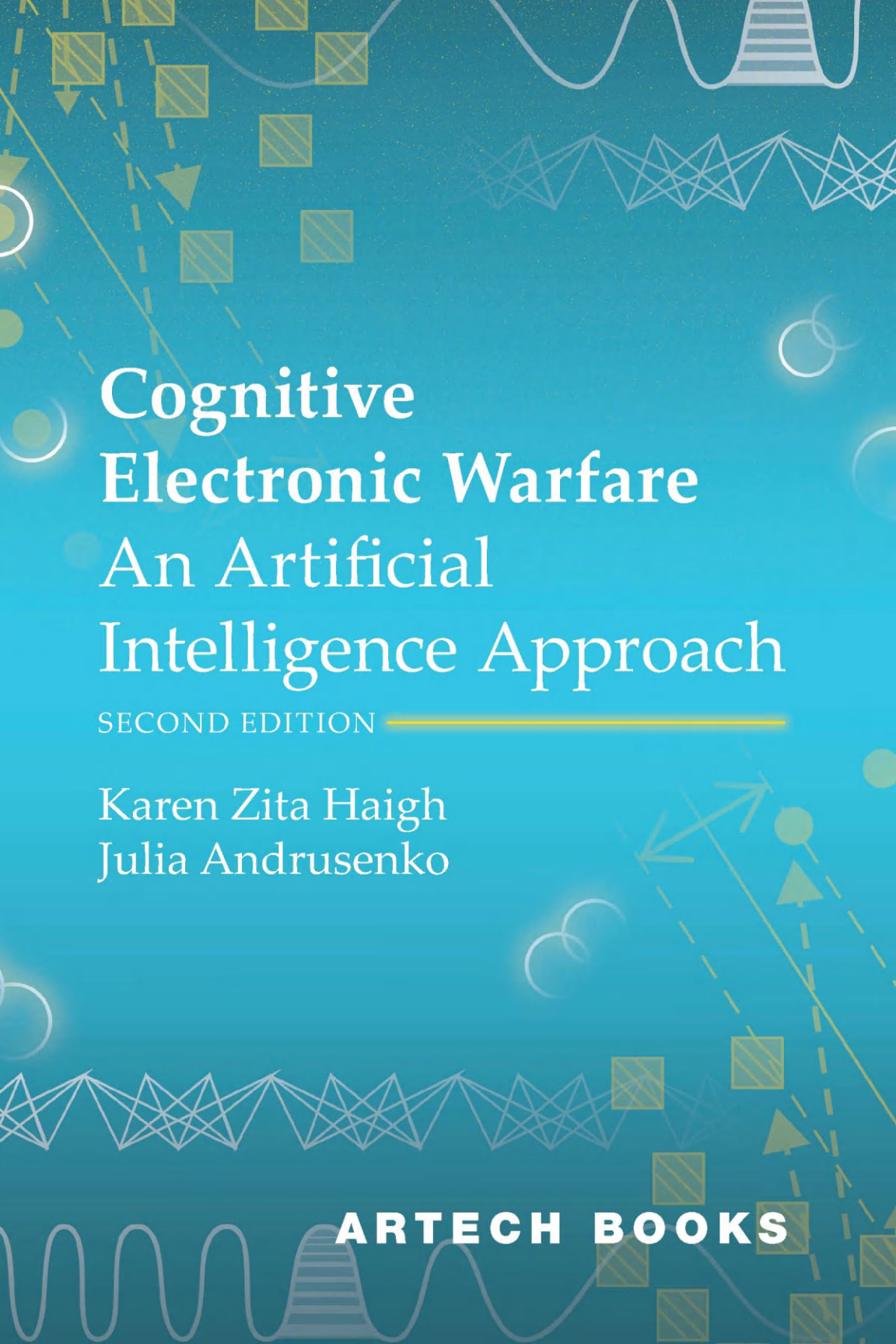


The background is a solid blue color with various abstract geometric patterns. In the top left, there are several yellow squares with diagonal lines and dashed lines with arrows pointing downwards. In the top right, there is a white sine wave and a white trapezoid with horizontal lines. In the middle right, there is a white zigzag line. In the bottom left, there is a white sine wave and a white trapezoid with horizontal lines. In the bottom right, there are several yellow squares with diagonal lines and dashed lines with arrows pointing upwards. The title is written in a large, white, serif font, centered on the page.

Cognitive Electronic Warfare An Artificial Intelligence Approach

SECOND EDITION

Karen Zita Haigh
Julia Andrusenko

ARTECH BOOKS

Cognitive Electronic Warfare

An Artificial Intelligence Approach

Second Edition

The first edition offered an approachable foundation in cognitive EW, and this eagerly awaited second edition updates the content to tackle current critical issues. It further provides practical AI examples to accelerate signal identification and an accessible introduction to RF principles for AI specialists. **If you are working in EW, this is the one reference book that you should have on your shelf.**

—Tim Fountain

*Head of Aerospace & Defense Test Market Segment
Rohde & Schwarz, France*

The second edition delivers timely and crucial information on cognitive EW for experts across sectors, incorporating significant updates on AI/ML-driven technologies and adaptive systems highlighted by the Ukraine War. This seminal work underscores the vital role of cognitive EW in today's and future battlefield scenarios against near-peer adversaries. **Timely, critical, and a must-know for EW, academia, industry, and government alike.**

—Sean Pascoli

*Deputy Program Manager
US Army DEVCOM Army Research Laboratory*

An excellent introduction to the use and potential of artificial intelligence in electronic warfare—now further strengthened in its second edition with updated concepts reflecting recent research, and an increased focus on practical examples and first steps. **A valuable resource for both EW specialists and those new to the field.**

—Vincent Voogt

*Senior Scientist, Electronic Warfare department (Unit Defence,
Safety and Security), Netherlands Organisation for
Applied Scientific Research (TNO)*

The increasingly congested and contested electromagnetic battlespace demands sophisticated tools, making cognitive EW essential for maintaining control amidst prioritizing civilian access. Drawing from lessons in the Russia-Ukraine conflict, this approach, leveraging AI and machine learning, offers the necessary precision and timeliness for effective sensing, decision-making, and maneuver of EW effects. **Current military activities demonstrate the critical need for EMS control by means of EW—we have no time to lose. Dr. Haigh and Ms. Andrusenko offer the foundational knowledge needed to get this right.**

—Lt Col Erik A. Bamford

*Director International Region I, Association of Old Crows, and
Branch Chief EW, Norwegian Armed Forces*

Like the first edition, this second edition will become the standard work for this fascinating discipline. Its readable and concise yet comprehensive style will appeal to both the specialist and general reader alike. This work should be required reading from the foxhole to the boardroom and ministerial office. **The right book at the right moment.**

—Dr. Thomas Withington, *EW Writer and Analyst*

For a complete listing of titles in the
Artech House Electronic Warfare Library,
turn to the back of this book.

Cognitive Electronic Warfare

An Artificial Intelligence Approach

Second Edition

Karen Zita Haigh

Julia Andrusenko



**ARTECH
HOUSE**

BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data

A catalog record for this book is available from the U.S. Library of Congress.

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library.

Cover design by Joi Garron

ISBN 13: 978-1-68569-103-5

Distribution Statement A: Approved for public release, distribution unlimited.

© 2025 ARTECH HOUSE

685 Canton Street

Norwood, MA 02062

All rights reserved. Printed and bound in the United States of America. No part of this book may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without permission in writing from the publisher.

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Artech House cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

10 9 8 7 6 5 4 3 2 1

*To the most important people in and of our lives,
your love, wisdom, and support kept us going*

K. Z. H.

J. A.

Contents

	Foreword	xv
	Preface	xvii
	Foreword to the 1st Edition	xxi
	Preface to the 1st Edition	xxiii
1	Introduction to Cognitive EW	1
1.1	The Vision of Cognitive EW	2
1.2	What Is a Cognitive System?	3
1.3	A Brief Introduction to AI	6
1.4	A Brief Introduction to EW	9
1.5	Cognition in EW	13
1.6	Civilian EW	15
1.7	Cognitive Radio, Cognitive Radar, and Cognitive EW	18
1.8	EW System Design Questions	19

1.9	EW Domain Challenges Viewed from an AI perspective	20
1.9.1	Situation Assessment for Electronic Support and EW Battle Damage Assessment	20
1.9.2	Decision Making for Electronic Attack, Electronic Protect, and Electronic Battle Management	22
1.9.3	User Requirements	24
1.10	Reader's Guide	25
1.11	Conclusion	25
	References	28
2	Objective Function	31
2.1	Observables that Describe the Environment	32
2.1.1	Clustering Environments	34
2.2	Control Parameters to Change Behavior	38
2.3	Metrics to Evaluate Performance	40
2.4	Creating a Utility Function	44
2.5	Utility Function Design Considerations	47
2.6	Example Observables, Controllables, and Metrics	49
2.7	Conclusion	52
	References	52
3	Machine Learning Primer	55
3.1	Common ML Algorithms	57
3.1.1	Support Vector Machines (SVMs)	58
3.1.2	Artificial Neural Networks	59
3.1.3	Ensemble Methods	62
3.1.4	Other ML Algorithms	63
3.2	Generalization to Surprise	64
3.3	Hybrid ML	66

3.4	Open-Set Classification	69
3.5	Metalearning	70
3.6	Generative AI	73
3.7	Embedded ML	75
3.8	Training an ML Model	77
3.9	Algorithmic Trade-Offs	78
3.10	The Speed of AI	81
3.11	Conclusion	85
	References	85
4	Electronic Support	97
4.1	Detection, Localization, and Signal Separation	98
4.2	Emitter Classification and Characterization	101
4.2.1	Feature Engineering and Behavior Characterization	102
4.2.2	Waveform Classification	104
4.2.3	Specific Emitter Identification	106
4.3	Performance Estimation	107
4.4	Multisensor Data Fusion	110
4.4.1	Data Fusion Approaches	111
4.4.2	Example: 5G Data Fusion for Localization	113
4.4.3	Distributed Data Fusion	115
4.5	Anomaly Detection	115
4.6	Causal Relationships	117
4.7	Intent Recognition	118
4.8	Conclusion	121
	References	122

5	Electronic Protect and Electronic Attack	135
5.1	Optimization	138
5.1.1	Multiobjective Optimization	139
5.1.2	Searching through the Performance Landscape	141
5.2	Scheduling	147
5.3	Reward Hacking	152
5.4	Anytime Algorithms	154
5.5	Centralized, Distributed, and Decentralized Optimization	155
5.5.1	Centralized Optimization	156
5.5.2	Distributed Optimization	157
5.5.3	Decentralized Optimization	158
5.5.4	Summary	159
5.6	Conclusion	159
	References	159
6	Electronic Battle Management	173
6.1	Planning	175
6.1.1	Planning Basics: Problem Definition, and Search	177
6.1.2	Hierarchical Task Networks	178
6.1.3	Action Uncertainty	180
6.1.4	Information Uncertainty	183
6.1.5	Temporal Planning and Resource Management	186
6.1.6	Multiple Timescales	188
6.2	Game Theory	189
6.3	Human-Machine Interface	191
6.3.1	System Designers	193
6.3.2	Commanders, Mission Planners, and EWOs	194
6.3.3	Understanding the Human Users	196
6.4	Conclusion	197
	References	197

7	<u>Real-Time In-Mission Planning and Learning</u>	207
7.1	Execution Monitoring	208
7.1.1	EW Battle Damage Assessment	209
7.2	In-Mission Replanning	214
7.3	In-Mission Learning	216
7.3.1	Updating Models	218
7.3.2	Exploitation, Exploration, and Active Learning	219
7.3.3	Reinforcement Learning Approaches	220
7.4	Conclusion	230
	References	230
8	<u>Data Management</u>	241
8.1	Data Quality Control	242
8.1.1	Data Curation	243
8.1.2	Bias and Noise	246
8.1.3	Traceability	248
8.2	Data Modeling: Ontologies, Metadata, and Schemas	249
8.2.1	Ontologies	249
8.2.2	Metadata	251
8.2.3	Schemas	257
8.3	Data Management Practice	257
8.3.1	Data in an Embedded System	257
8.3.2	Data Diversity	258
8.3.3	Data Augmentation	261
8.3.4	Forgetting Data	265
8.3.5	Data Privacy and Security	265
8.4	Conclusion	267
	References	267

9	Architecture	273
9.1	Software Architecture: Interprocess	273
9.2	Software Architecture: Intraprocess	278
9.3	Language Choices	278
9.4	Hardware Choices	281
9.5	Conclusion	284
	References	284
10	Test & Evaluation	287
10.1	Paradigm Shifts	289
10.2	Validating the Learning Process	291
10.2.1	Iterative Design and Test	292
10.2.2	Learning Assurance Process	292
10.3	Evaluate Learning Goals	295
10.4	Determine Range of Operational Effectiveness	296
10.5	Mixed-Fidelity Closed-Loop Testing	298
10.5.1	Closed-Loop Test Framework	298
10.5.2	Multifidelity Vertically Integrated Test Framework	299
10.5.3	Mixed-Fidelity Test Framework	301
10.5.4	Integrating Third-Party Tools for System Analysis	301
10.6	Behavior-Based Models with Closed-Loop Effects	301
10.6.1	Stimulus-Response Effects	302
10.6.2	High-Level Behaviors	302
10.6.3	Detailed Transceiver Characteristics	303
10.6.4	Possible Future Capabilities	303
10.7	Smart Experimental Design	304

10.8	Computing Accuracy and Adequacy	305
10.8.1	Regression and Normalized RMSE	305
10.8.2	Classification and Confusion Matrices	307
10.8.3	Evaluating Strategy Performance	310
10.8.4	Adequacy: Strategy Performance over a Scenario	311
10.9	Ablation Testing	315
10.10	Verification Approaches	319
10.10.1	Formal Verification Methods	320
10.10.2	Empirical and Semiformal Verification Methods	322
10.11	Example Implementation: Scenario Driver	324
10.12	Conclusion	328
	References	329
11	Getting Started: First Steps	335
11.1	Engineering Resilient Systems	336
11.2	Development Considerations	339
11.3	Choices: AI or Traditional?	342
11.4	ML Toolkits	345
11.5	RF Datasets and RF Data-Generation Tools	345
11.5.1	RF Datasets for ML	346
11.5.2	RF Data-Generation Tools	348
11.6	Projects	350
11.6.1	Project Idea: RF Fingerprinting	351
11.6.2	Project Idea: Open-Set Recognition	352
11.6.3	Project Idea: Detect Causal Connections	352
11.6.4	Project Idea: Describe Emitters	353
11.6.5	Project Idea: Classify WiFi and Bluetooth Interference	353
11.6.6	Project Idea: GNSS Signal Integrity and Spoofing Detection	354
11.6.7	Project Idea: Multisensor Fusion for Localization	354
11.6.8	Project Idea: Multiplatform Route Planning	355
11.6.9	Project Idea: Decision Loop and Online Learning	355
11.6.10	Project Idea: EW Decision Making with 5G Technologies	357

11.6.11 Project Idea: Augment Existing Data	357
11.6.12 Project Idea: Evaluate ML Models	360
11.6.13 Project Idea: Implement a DeepNet on an FPGA	361
11.6.14 Project Idea: Data Compression	362
11.7 Conclusion	362
References	363
Acronyms	371
About the Authors	373
Index	375

Foreword

The challenges facing electronic warfare (EW) exceed the capabilities of traditional systems, necessitating the application of an artificial intelligence (AI) approach. Authors Dr. Karen Zita Haigh and Julia Andrusenko build upon their earlier work in this area to examine the use of AI in conducting such warfare. This work introduces RF practitioners to AI techniques for EW. AI is pervasive, impacting various industries, including autonomous vehicles, transportation, logistics, intelligence gathering, threat detection, damage assessment, and cybersecurity. It will also play a significant role in the evolution of EW. The speed and rhythm of contemporary warfare mean that AI tools and techniques are the only viable means to respond effectively and maintain any advantage. We have transitioned from an information-sharing approach of “publish and subscribe” with many humans involved, to a “sense and respond” environment with reduced human intervention. The application of AI is essential for this evolution in the conduct of electronic warfare.

In my work at both the state and national levels in public safety emergency communications and data/systems interoperability, the ultimate goal has always been to achieve real-time situation awareness (RTSA). Whether for public safety first responders—police officers, firefighters, paramedics—or for their partners in transportation, public works, utilities, and public health, and also for the warfighter, the need for RTSA is universal.

RTSA involves collecting and detecting data about the environment, fusing it, and transforming it into actionable information for those who need it when they need it. To achieve RTSA, one must integrate a variety of disparate information from numerous systems via multiple communication channels. RTSA does not stem from knowing a single piece of information; rather, it requires spatial context, temporal scale, and visual confirmation. As one state police practitioner told me, “There are only four questions you want to answer when you are having a ‘bad day’... they are: Where is it? What do we know about it? Can we talk about it? And, Can we see it?” These questions, while somewhat simplistic, have significant implications for geographic information system (GIS) and global positioning system (GPS) data, video data, access to both static and real-time sensors, and the status of communication networks. When fused, this data can support RTSA about incidents

in public safety and battlefield environments. Previously, accessing and integrating disparate data for dissemination posed challenges for identifying key information, publishing it in a standardized way, and allowing users to subscribe to the product. However, advancements in communication, coverage, and computing now create greater opportunities for RTSA in public safety and military contexts. Today, basic interoperability is less of an obstacle, and the traditional *access, publish, subscribe* model is too cumbersome for the volume and speed of data flow. AI and machine learning tools are now available to analyze data, identify patterns, draw inferences, and guide decisions. We are transitioning to more of an *access, sense, respond* paradigm where cognition, learning, and adaptation are essential.

Collaboration between the military, civilian, and private sectors is increasingly important for force protection, border security, and EW. While EW may seem limited to the military realm, the underlying technology is equally vital to the public safety sector. Like the military, the communications environment for public safety has become more complex, utilizing commercial assets and technologies such as near field communications (NFC), Bluetooth, WiFi, land mobile radio (LMR), 5G, LTE, television broadcast/datacasting, and satellites. Public safety aims to improve situation awareness before, during, and after an incident. However, the communications environment can also hinder information flow due to data interoperability and interactions with bad actors. Cognitive systems can help mitigate these challenges.

My colleagues Karen and Julia have continued their work from the first edition, published in 2021, to explore the changes in technology that have occurred since then. They have expanded the discussion to include generative AI, deep reinforcement learning, and metasurfaces. They have done this in a “plain language” manner that is genuinely valuable to the reader. Additionally, they expanded the discussion of data management and testing approaches, which underpin much of the envisioned functionality. They also included various projects to illustrate the concepts and help people get started with this technology. This publication is a landmark work with significant implications for how we design, monitor, and defend radio and radar systems, as well as effectively conduct EW. It is a valuable resource that can foster innovation and shape the future of cognitive EW.

John M. Contestabile

Director, Public Safety Solutions, Skyline Technology Solutions

Former Deputy Director, Governor's Office of Homeland Security, Maryland

April 2025

Preface

Cognitive electronic warfare (CogEW) has undergone a remarkable transformation in the past four years, propelled by advancements in artificial intelligence (AI), machine learning (ML), and software-defined radios. Once reliant on static, rule-based systems, modern EW now embraces adaptive, autonomous technologies capable of sensing, learning, and dynamically responding to complex electromagnetic environments.

Breakthroughs in deep reinforcement learning now empower systems to optimize jamming and interception strategies in rapid real time. Spectrum situation awareness and signal classification enhance situation awareness, while edge computing accelerates decentralized decision making. Techniques such as transfer learning enable rapid adaptation to new signal types with limited data, while emerging methods like zero-shot learning show potential for identifying previously unseen signals based on high-level signal features. At the same time, adversarial ML introduces both new vulnerabilities and innovative methods to counter AI-based threats.

This edition captures the changes in technology that have taken place since the last edition in 2021. Most of these changes are in AI, which is exploding in a tidal wave of change; a few developments are in EW or hardware. The book went from 246 pages to 407, with lots of new material and restructuring of previous material. The major changes are:

- In the first edition, we expected that most of our readers would be radio frequency (RF) experts, and that we didn't need to explain the concepts of the electromagnetic spectrum (EMS) or how radio or radars work. We learned from AI experts that this was the first time someone had explained the RF problem in words that they could understand. For that reason, we added more discussion of the EMS, including a conversation of civilian EW. The mapping between cognitive radio, cognitive radar, and cognitive EW is now more highlighted.
- We added conversation about some of the modern AI tools, including spiking neural networks, large language models, generative AI, and deep reinforcement learning. We also added a discussion about how to build models that

work on the edge, with limited compute, power, and communications. There is now more explicit conversation about reward hacking.

- We significantly rewrote the “Test & Evaluation” chapter to better capture the challenges of testing systems that learn after they have been deployed. Testing AI-enabled systems is a popular topic currently; we take the extra step of discussing the difference between *AI-enabled* and *cognitive*, and what that means for assuring cognitive systems.
- We added a variety of projects with the goal of making practical, bite-size examples to get you started. Each idea could grow into larger, deployable systems. In the same vein, we added more examples of how the cognitive concepts would appear in an RF system, including systems development and a longer conversation about EW battle damage assessment (EW BDA).
- We added a discussion on some of the newer hardware, including neuromorphics, photonics, and quantum. Metasurfaces—the technology behind invisibility cloaks—are already a reality, playing a crucial role in 6G. In EW, they are unlocking innovations in sensing, stealth, communications, and radar. Metamaterials are perhaps the strongest motivating factor for fast real-time optimization and learning in EW.
- We expanded our discussion of data management, including semantics, metadata, and schemas. We also grew the list of RF datasets; while the number of public RF datasets is still small, it is no longer embarrassingly short.
- We added a full section on the speed of AI, trying to describe the rate of change. Perhaps the strongest indicator of the speed of change is that there were daily updates. DeepSeek, the model that shook the world in mid-Jan 2025, was overtaken by two other models in less than two weeks. It felt like quicksand: By the time we had completed a sentence, it would already be outdated. Almost 40% of the citations in this edition are to works published in 2021 or later, and we barely scratched the surface of what’s available.

The lesson learned for readers is that any citation or recommendation that a technique works well should be taken with a grain of salt: There’s a *very* good chance that something better has come out. We hope that we’ve given you enough of the ideas and keywords to be able to think creatively and search effectively.

This past 5 years has been a whirlwind of conversations and learning. Speaking to academics, industry, and operational users globally has been enlightening. We can’t name all the individuals who have enriched our lives and our understanding

of this domain, especially those whose lives are directly impacted by these capabilities. The following individuals have directly contributed to this edition of book, with written text or reviews: David “Flipper” Brown, Joe Guerci, Jarrett Holcomb, Haruko Kawahigashi, Rafael Licursi, Jeff Mahon, Haydn Nelson, Trung Nguyen, James Stewart, Darrell “V” Venture, Vincent Voogt, John Waddell, William “\$” Young, and Amy Zhang. We wish to give our heartfelt thanks to Hatsuo Kimura, who translated the first edition into Japanese. His dedicated efforts ensured that the translation accurately conveyed our ideas. His insightful questions about our text helped us identify areas where we could enhance clarity and better express important concepts.

It has been truly rewarding talking with readers and understanding their needs, capabilities, and gaps. Niche technical book authors do not earn much from our books; it’s about 3¢ per hour for us. Our reward function (to use reinforcement learning terminology) is that readers find the information helpful. Being asked to write the second edition is deeply rewarding. However, it also represents a partnership with the publisher, which means they need rewards too. They need to know how many books have sold; unfortunately, pirated copies don’t count on their scales. Small publishers are a great way for niche authors like us to disseminate knowledge, but they need your help. Help them stay afloat so they can find other authors who have things to say. If you found this book valuable, please support us and our publisher by spreading the word. Recommend it to colleagues, buy the book, cite us in your work, leave a review, or simply let us know how it helped you.

Writing this book, and then sharing its concepts through teaching and discussions, has been an amazing journey for both of us. A few years ago, Karen received a thoughtful gift from a friend representing her dream job: a keychain stating *Chief Evangelist*. Her trip to Kenya was especially rewarding, where she spoke about CogEW in the context of anti-poaching efforts. We are deeply grateful for these experiences and the inspiring community that continues to shape our work.

As CogEW evolves, its potential to transform spectrum operations is becoming increasingly clear. The integration of AI-driven adaptability, FPGA acceleration, and software-defined architectures is opening new frontiers for smarter, more efficient signal processing and defense mechanisms. Future systems will respond to challenges and proactively optimize performance in dynamic environments. Advances in AI, ML, edge computing, and autonomous decision-making will further enhance the adaptability and resilience of EW systems, enabling them to anticipate threats, refine countermeasures, and operate with unprecedented efficiency. These advancements will extend beyond EW, shaping secure communications, resilient network operations, and the future of adaptive spectrum management.

This second edition highlights these developments, offering insights to inspire innovation in this rapidly growing field. We hope it serves as a foundation for further exploration, as these advancements are only the tip of the iceberg. With continued research, CogEW will unlock more resilient, intelligent, and adaptive spectrum solutions, emphasizing trustworthiness and explainability for reliable decision making.

—Karen Haigh and Julia Andrusenko

Foreword to the 1st Edition

Advances in electronic warfare (EW) have determined the winners of wars. The research, development, and adoption of these technical advances shapes the tactics, operational maneuvers, and strategies employed by commanders at all levels in the battlespace. Advanced research and development efforts contribute substantially to deterrence. The Department of Defense in the United States invests approximately \$7B annually in EW; I led the development and acquisition of these systems in the Pentagon and continue to advance these systems as a Chief Technology Officer in our national security innovation base.

Cognitive EW is one of these critical advances that will determine the outcomes of future battles. An EW researcher from World War II would recognize many of the systems operationally fielded today. The application of artificial intelligence (AI) to make EW systems cognitive is a critical advance to allow our systems to adapt and to learn during a mission. In a digital world of software-defined capability, EW systems must respond to previously unknown signals. In the densely connected battlespace of today, the so-called internet of military things, feedback can be known, or at least estimated, continuously during a mission. Through aggregated sensor feeds and understanding, EW systems are capable of adapting based on this real-time feedback. With appropriate automation, this feedback and learning can occur faster than humans can reason on data. This adaptation during an individual mission or engagement can allow our soldiers and our airmen to successfully complete their missions, or in some cases survive an engagement.

While it's easy to state that an EW system will learn from its surroundings in real time, it's quixotic to assume that just any system is effective, and robust, in performing its function. Society, including its military elements, is excited at the future AI will enable. To realize this future, we need experts like Karen and Julia to link the theory to the application. While it's easy to showcase a demo, military systems require robustness in their operations and must undergo critical testing and evaluation.

The application of AI into our lives will change how we perform functions each day. Military applications of cognitive technology require security as well to protect their functionality. Machine learning and adaptivity begin to change this

process for the military. While there are diverse interesting philosophical questions around protecting algorithms, specific implementations, training data, and real-time learning, the academic foundation is quite robust and provides valuable insights for practitioners in the field. Given the rapid rate of innovation and advancements in AI, military systems must leverage these advances at the speed of relevance.

Karen and Julia do an excellent job in this book at providing the framework and enabling a developer to include these recent advances into their capabilities. A core understanding of AI provides a foundation, followed by a detailed discussion of objective functions for cognitive EW. Each EW function is unique and the nuances are described in necessary depth. Every acquisition professional knows that sustaining weapon systems is where the costs are truly borne. In this case, the management of data, building the right architectures and testing these new systems is covered. Lastly, Karen and Julia provide a gateway to these new system functions. The balance of breadth and depth ensures alignment and success to meet the lofty goal of helping nations deter and, when needed, win wars.

*Dr. William G. Conley
Chief Technology Officer, Mercury Systems
Former Director of EW the Office of the Secretary of Defense
July 2021*

Preface to the 1st Edition

I often hear that cognitive EW is too big a landscape and that people don't know where to start. I always say: "AI is like Math: One day it will be everywhere." Everybody, no matter their role or their age, should know enough about AI to know where it applies to the problems in their daily lives. AI is not just "autonomous vehicles" or "Terminator;" AI is face recognition on a camera, purchase recommendations when shopping, and predictions about hurricane paths. AI belongs in EW systems, just as math and physics belong in EW systems. This book aims at lighting the path, and igniting a passion for solving EW problems that can't be solved using only traditional methods.

Julia and I wrote the book for radio frequency (RF) people—experts in EW, cognitive radio, and/or cognitive radar—who want to learn more about how and where to use AI. Our goal is to help triage and guide EW system designers in choosing AI and machine learning (ML) solutions. ML is a key component, but AI is much more than just ML.

If it seems like we flip-flop between communications and radar, or electronic attack and electronic protect, we do. From an AI standpoint, these problems truly are interchangeable. We use the same AI techniques in all four of those areas. There are some important differences, such as EW battle damage assessment (BDA), but AI can bring these ideas together, solve common problems, and move from coexistence to codesign.

We use a running example throughout the book: the BBN Strategy Optimizer (BBN SO). It is a communications electronic protect effort that learns how to handle novel environments using in-mission learning at mission-relevant timescales, and then performs hard real-time optimization and decision making to choose mitigation strategies. Example 1.1 presents the essential concept, and many of the discussion points use examples derived from this work.

We'd like to thank the people who directly contributed to the book in some concrete way: Aaron Walker, Chris Geib, Matt Markel, Bill Conley, Pete Reese, Feng Ouyang, Shaun McQuaid, and the anonymous reviewer from Artech House.

The first two people who started me thinking about the AI-for-RF problem are Chris Ramming who presented the initial problem, and Jim Freebersyser who

funded the first effort from his Honeywell internal research budget. Without these two individuals, I would never have developed the interest or momentum to see it through. Greg Troxel and John Tranquilli patiently helped educate me—that weird AI person—teaching me how to spell RF and think about what makes things hard in this space.

Many others have helped over the years. On an effort like this, I am guaranteed to miss people who have helped mature my thinking and develop AI-RF systems that actually work in the field. My apologies to those whom I missed. These include Louis Bouchard, Michael Cook, Amber Dolan, Dianne Egnor, Regina Hain, Brian Hood, David Landon, Li Lin, Scott Loos, Allan Mackay, Hala Mostafa, Andrew Nelson, Jason Redi, Vikram Sardana, Choon Yik Tang, Chris Vander Valk, Srivatsan Varadarajan, and Fusun Yaman. I'd also like to thank the program managers and fellow performers on my DARPA, AFRL, ONR, and transition contracts.

Julia has been an amazing partner on this effort, providing different perspectives, arguing with me when I needed it, and cheering me on when things weren't going well. I'd like to thank my Ph.D. advisor Manuela Veloso, who taught me to stick with an idea, even when the rest of society isn't ready for it yet. Finally, my family, who understand most of the nouns in this book but not the phrases, and still support me.

—Karen Haigh

Karen and I first met in November, 2019 at the IEEE Military Communications Conference, Norfolk, VA. It is then when she asked for my help with this book's write-up. I believe my first answer was “no,” then, “let me think about it.” Karen, thanks for your persistence in bringing me on this wild journey with you. I echo Karen in expressing my immense gratitude to all of the people that helped us tackle this difficult, multi-faceted problem of cognitive EW. Lastly, from my side, I wanted to thank my brilliant Johns Hopkins University Applied Physics Laboratory Colleagues for their continual support of all of my endeavors. Thank you all.

—Julia Andrusenko

1

Introduction to Cognitive EW

The challenges of modern electronic warfare (EW) are beyond the ability of traditional approaches to solve. Incorporating artificial intelligence (AI) techniques into EW systems is the only way to manage the complexity of this problem domain, its rapid timescales, and novel emitters. This book describes how AI techniques can help address the challenges of modern EW. We expect readers to be familiar with at least one of EW, cognitive radio, or cognitive radar domains, and will focus instead on the AI techniques and their associated challenges and trade-offs. In the future, AI will be part of every EW system, recording and analyzing system previous performance and then adapting behavior according to the current situation. AI—not just machine learning (ML)—is the heart of future CogEW solutions.

Cognitive radio concepts have existed since at least 1999, when Joe Mitola III [1] introduced the term, while the idea of cognitive radar has been around since at least 2006 [2]. Fielded examples, however, are few and far between, and limited in their cognitive capabilities. A significant reason for this gap is due to the sense that building a cognitive system is a huge undertaking, exacerbated by the misconception that it is impossible to evaluate a cognitive system.

This book aims to address this gap. It introduces RF practitioners to relevant AI techniques for EW: situation assessment, decision making, and ML. It covers situation assessment techniques for *electronic support (ES)*, including detection, characterization, classification, anomaly detection, causal reasoning, and intent recognition. The book discusses decision-making techniques for *electronic protect (EP)*, *electronic attack (EA)*, and *electronic battle management (EBM)*, including planning, optimization, and scheduling, and how to manage the temporal trade-offs and distributed nature of the problem. It describes where to incorporate ML to improve both situation assessment and decision making. The book also examines a significant area of interest for EW system designers: real-time in-mission learning to recognize novel environments and respond to surprises.

To address the barrier-to-adoption created by the lack of understanding of how to evaluate a cognitive system, the book describes how to manage data and models

to ensure quality and presents a learning assurance process for validating the results and building trusted systems.

Finally, this book provides a few recommendations on how to get started with building an actual CogEW system.

*Creating a CogEW system is not a
huge “all-or-nothing” hurdle.*

From an AI standpoint, EP and EA differ only in their objectives: defines objectives with respect to oneself, while EA defines objectives with respect to the adversary. Likewise, AI is agnostic to whether the solutions apply to radar or communications problems. This book therefore treats EP/EA and radar/communications equally, highlighting the few places where the problem domain impacts the choice of data or algorithm.

The AI techniques presented here apply to other related disciplines such as cybersecurity, information warfare, position, navigation, and timing, and intelligence, surveillance, and reconnaissance. We do not directly address related areas.

1.1 The Vision of Cognitive EW

Traditional approaches to EW are failing in modern engagements for three primary reasons:

- First, the *timeline is too fast*. Humans reason at seconds or slower, and EW operates at seconds or faster.
- Second, the *domain complexity is too high*. Humans can naturally reason about two or three simultaneous objectives; with training, a specialist in a field can handle five or six. EW, meanwhile, has many competing objectives, and possibly a thousand actions available on each platform. Across a network of nodes, humans cannot optimally configure nodes, even at slow-time.
- Third, there are no good methods to *handle novel emitters*. Radar EW systems typically use a library of known systems, while comms EW systems have preprogrammed waveforms. In both cases, humans design and implement all changes.

The goal for a CogEW system is therefore to automatically select actions that improve performance towards objectives, even in novel RF environments. Achieving this goal requires three complementary capabilities. The system must: (1) characterize the RF conditions, (2) choose the best actions to improve mission performance, and (3) learn the performance of available strategies.

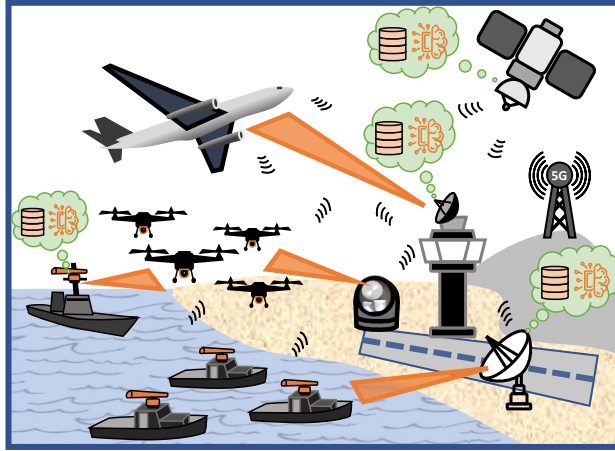


Figure 1.1 The vision of a CogEW system is one that can see through the clutter, reason about the complex electromagnetic environment, and learn how to respond to novel waveforms at mission-relevant timescales.

CogEW understands and predicts the EMS environment, makes goal-directed decisions to improve the performance of an EW system, and learns from its own actions at mission-relevant timescales with minimal human supervision.

To be considered cognitive, it must interact with the environment, and use that experience to change the way it makes decisions.

Figure 1.1 illustrates a motivating scenario with clutter and novel waveforms. A traditional system may have difficulty identifying key emitters within the clutter, and be completely unable to identify or respond effectively to novel systems.

1.2 What Is a Cognitive System?

Despite myths in our community to the contrary, the concept of *cognition* is very well defined by human psychology and animal behavior specialists:

Cognition refers to the mechanisms by which animals acquire, process, store, and act on information from the environment. These include perception, learning, memory, and decision-making.

— Sara Shettleworth, 2010 [3]

This definition of a cognitive system applies to any domain, biological or synthetic: toddlers, teenagers, engineers, crows, elephants, octopods, or robots.

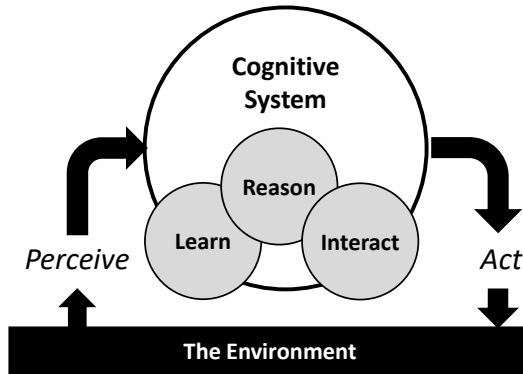


Figure 1.2 A cognitive system perceives the environment, reasons about the situation, and acts to accomplish goals. It learns from the interaction with the environment.

Each of these systems exhibits cognition in different ways that depend on the environment it occupies. Toddlers exhibit enormous learning ability. Teenagers are developing decision-making capabilities and exploring different levels of risk-taking and effectiveness. Our late acknowledgment of octopod intelligence is due in part to our expectation of human-centric decision-making approaches.

A *cognitive system*, or *intelligent agent*, perceives its environment, reasons about what it has seen, and takes actions to achieve its goals, as illustrated in Figure 1.2. It reasons and understands at a higher level, dealing with symbolic and conceptual information, to make accurate decisions in complex situations. Cognitive systems are aware of context, handle uncertainty, and make judgments autonomously. Cognitive systems are:

- *Iterative*, because they perceive, reason, and act throughout their lifetime;
- *Interactive*, because every action changes their environment;
- *Deliberative*, because they are goal-directed; and
- *Transformative*, because they learn from their experiences.

Cognitive systems may *collaborate* by sharing information among nodes, and they may *coordinate* when they accomplish tasks together.

The key difference between a cognitive system and an AI-enabled adaptive system is that cognitive systems learn after deployment. An *adaptive* system is one whose behavior changes based on current conditions, i.e., $a = f(c)$ for action a and conditions c . Radio and radar waveforms are highly adaptive, using fixed rules or programming to adapt to the environment. A *cognitive* system, on the other hand,

is one whose interaction with the environment allows it to change the way it makes decisions, i.e., it can change the function f .

An AI-enabled system can use AI techniques to perform its adaptive tasks. For example, an AI-enabled EW system can learn a model that identifies different modulations, or generate a plan for deploying assets to a mission. A system that uses AI techniques to understand its environment, or to generate an action plan, but does not directly interact with the environment, has cognitive *features*, but is not a cognitive *system*. It becomes a CogEW system only when we add the ability for the system to directly interact with the environment to update its own decision-making process.

Figure 1.3 illustrates the cognition loop using AI terminology. Other mappings exist, including Haykin's *Perception-Action Cycle* [4] and Mitola's *Observe-Orient-Plan-Decide-Act-Learn (OODAL)* model [1]. The cognition loop is similar to the *Observe, Orient, Decide, and Act (OODA)* loop commonly used in military environments [5]. The key difference between the OODA loop and the cognition loop is that OODA describes a process; cognition happens in the humans following that process. For example, situation assessment performed by a cognitive system feeds the situation awareness of the human operators.

Figure 1.4 lays out these three functions to highlight increasing levels of cognition. A simple, traditional EW system uses the ES to identify known signals, and chooses a response based on a library lookup. Each AI capability increases the overall level of cognition. This layout also provides a means to evaluate and compare EW system capability. Horne et al. present a cognition mapping that

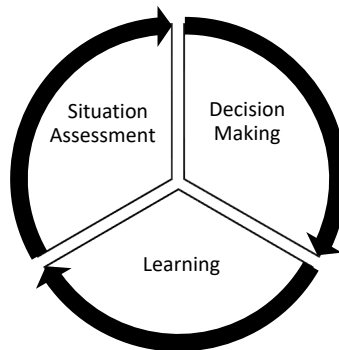


Figure 1.3 A cognitive system iteratively assesses the situation, plans and decides its actions, and learns from its experience.

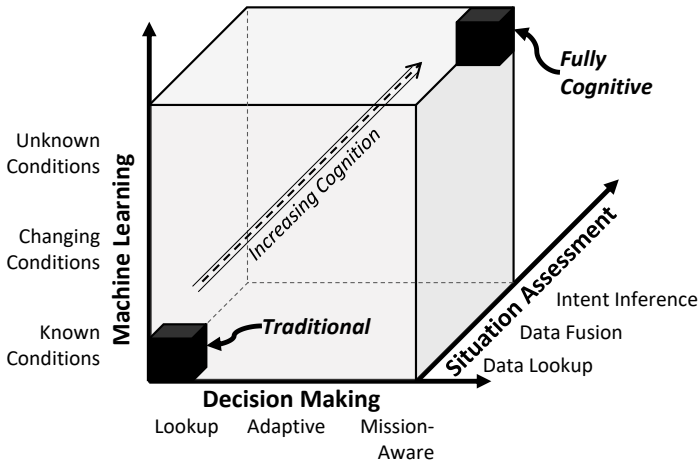


Figure 1.4 AI functions contribute to increasing levels of cognitive function, according to the concepts laid out in Figure 1.3.

further breaks down the decision-making components by *memory*, *forethought*, and *algorithmic sophistication* [6].

Cognition depends on the specific physical and social environments of the entity [7]. Different species perceive and reason differently, as do different individuals. As the human brain develops, our cognitive capability changes. Immediate context also impacts our cognition; for example, we will likely perform better in familiar environments than in new ones, or perform better when well-rested than when sleep-deprived. The same holds true for synthetic cognitive systems. A soldier hand-held radio needs different cognitive capabilities than an airport management system. There is no one perfect cognitive state. The environment shapes requirements and shapes the desired cognition.

1.3 A Brief Introduction to AI

The first key component to a CogEW system is the AI that supports the cognition. AI is the field of computer science that attempts to mimic human behaviors. AI was founded as an academic discipline in 1956 at the Dartmouth Conference [8], and draws from a wide variety of fields including mathematics, computer science, engineering, linguistics, and cognitive psychology.

AI is like math. One day, it will be everywhere.

As with physics, math, and engineering, AI comprises many subfields. Four key research areas form the pillars of core activity in AI:

- *Knowledge management* organizes the knowledge in a structured framework that links related information. It contextualizes information for specific needs. Key challenges include knowledge acquisition, organizing unstructured data, ensuring data quality, scalability, and tailoring knowledge to special requirements.
- *Situation assessment* is the understanding of environmental elements and events with respect to time or space, the comprehension of their meaning, and the projection of their future status. (Situation awareness is a result of a situation assessment module.) It is a critical foundation for successful decision making. Key steps are to *collect* the raw data, *validate* the observations, *fuse* the data into higher-level concepts, *analyze the impact* of the situation, and *infer the intent* of users (both friendly and adversarial). *Execution monitoring* is the activity within situation assessment that focuses on evaluating how well-planned actions succeeded. Key challenges for a situation assessment module include diversity in the data, latency of when the data is received, and how observable the environment is.
- *Decision making* involves setting goals and determining feasible methods of achieving them. The cognitive system must prioritize goals, understand the benefits and costs of different actions, and resolve conflicts. Key challenges include managing resources, multiple actors, and nondeterministic actions, especially in problem-solving domains where the environment is constantly changing.
- *Machine learning (ML)* extracts information from prior experience to improve future performance. ML techniques may extract rules about how to interpret observations or behave, or they may build functions that approximate the performance of the data. ML updates static models by extracting information from observations of the dynamic environment. Key challenges include data heterogeneity, missing data, and managing bias.

Consider the following example of an autonomous vehicle, where these four pillars might work together as follows. Knowledge management maintains a comprehensive database of information, including the map, traffic rules, and landmarks. Situation assessment assesses current road conditions, predicts disruptions, and estimates travel time. Decision making uses the static road map and assessed road conditions to suggest routes to the user, balancing factors such as travel time and

Artificial Intelligence							
Pillars	Knowledge Management		Situation Assessment		Decision Making		Machine Learning
Functions	Data Mining	Speech Recognition		Machine Vision	...	Robotics	Anomaly Detection
Domains	Medical AI	Financial AI	...	Legal AI	Educational AI	Autonomous Vehicles	Retail AI

Figure 1.5 AI approaches can be layered as a set of four pillars, many functional areas, and vertical solutions for specific domains.

distance. ML analyzes historic and real-time data to create models that capture dynamic properties of the road system such as rush hour and construction.

The four pillars of AI (knowledge management, situation assessment, decision making, and ML) combine to support the ultimate goal of AI: to create a fully autonomous intelligent agent.

The four core research areas are utilized in a variety of functional areas including data mining, robotics, natural language understanding, recommendation systems, and predictive analytics. These concepts are then adopted by specialized domains that develop vertical solutions, including medicine, finance, security, and manufacturing. Figure 1.5 illustrates these layers.

Typically, when academic AI algorithms and models are presented with real-world data, they do not completely solve the problem. After domain-specific developers update and improve the algorithms to create working real-world solutions, the changes influence the broader community. For example, the success of many-to-many machine translation models has impacted machine vision research. Similar to how multilingual models transfer learning across languages, few-shot learning in vision benefits from multitask techniques where the model learns from related tasks or object categories, improving recognition with limited labeled data. This cross-disciplinary effect leads to better performance in diverse, real-world scenarios.

Math, physics, engineering, and AI are each fields of study in their own right. One would not say that “Math is autonomy” or “Engineering is cognitive.” Likewise, one should not say that “AI is autonomy” or “AI is cognitive.” Cognition and autonomy are capabilities that require math, physics, engineering, and also AI.

This book is emphatically not a treatise on AI. It is a sampling of AI techniques that are most immediately applicable to CogEW. The book *AI: A*

Modern Approach [9] has an in-depth discussion on AI concepts, written from the perspective of building a complete intelligent agent. For a deeper understanding of AI's evolution, current advancements, and future directions, see *Scientific Progress in Artificial Intelligence: History, Status, and Futures* [10], which explores key developments, challenges, and their broader societal impact.

ML is broadly relevant throughout the EW system; Chapter 3 introduces ML in more depth. Chapters 5 and 6 discuss the AI techniques of planning, optimization, distributed AI, and human factors concepts because these form a logical unit within decision making. Chapter 8 presents knowledge management.

1.4 A Brief Introduction to EW

The first key component to a CogEW system is an understanding of the EW domain. Electromagnetic spectrum (EMS) operations comprise all coordinated military actions to exploit, attack, protect, and manage the electromagnetic environment to achieve a commander's objectives. EW focuses on how to control the spectrum or attack an enemy using the EMS, including radio, radar, infrared, optical, and microwave [11–14].

Electronic warfare (EW) is any action involving the use of the EMS or directed energy to control the spectrum, attack an enemy, or impede enemy assaults. The purpose of EW is to deny the opponent the advantage of, and ensure friendly unimpeded access to, the EMS. EW can be applied from air, sea, land, and/or space by manned and unmanned systems, and can target humans, communication, radar, or other assets (military and civilian).

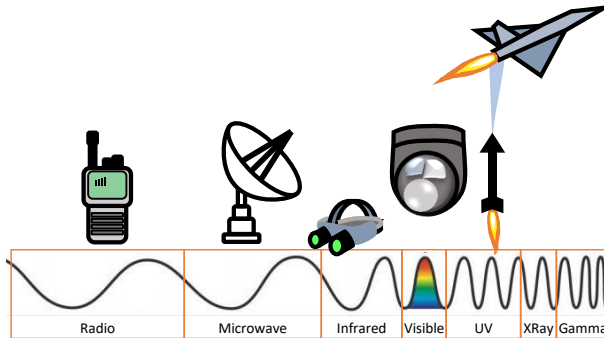
— *Chairman of the Joint Chiefs of Staff, 2012* [11]
Joint Publication 3-13.1: Electronic Warfare

The term *electronic warfare* traditionally refers only to communications and radar systems, while *electromagnetic warfare* refers to broader concepts including the full EMS, cybersecurity, collaborative effects, and strategic goals. Moreover, “electronic” denotes a device, while “electromagnetic” describes an effect, regardless of whether the mechanism is electronic in nature or not. From an AI perspective, distinguishing between these concepts is not important: Both domains interact with the EMS to accomplish goals, and it is the AI algorithms' responsibility to choose the correct actions to accomplish those goals. Callout 1.1 describes key EMS frequencies and some of their trade-offs.

EW comprises the following core concepts:

- *Electronic support (ES)* understands the spectrum—who is using it and how, when, and where. ES needs to detect, intercept, identify, and locate EMS

Callout 1.1 The EMS ranges from low frequency radio waves, through visible light, to high frequency gamma rays.



Different points in the EMS offer trade-offs to system designers. Lower frequency signals propagate longer distances. Higher frequency signals have higher bandwidth, using smaller antennas and typically less power.

- Low frequency waves (3 Hz to 30 kHz) can propagate through seawater, but have very low bandwidth.
- AM radio waves (540 kHz to 1700 kHz) travel farther than FM radio waves (88 MHz to 108 MHz) due to lower frequency. In 5G networks, low-band frequencies provide better coverage, but slower speeds, while high-band frequencies are faster, but have limited coverage.
- Microwave frequencies (300 MHz to 30 GHz) are used for long range communications and radar[†] due to good penetration through obstacles. In satellite communications, low frequencies are more reliable in bad weather, while higher frequencies offer better data rates. In radar, low frequencies detect long-range targets, while higher frequencies provide better angle accuracy and resolution.
- Millimeter waves (30 GHz to 300 GHz) are ideal for short-range high-bandwidth applications including 5G, 6G, and automotive radar, but are susceptible to atmospheric interference.
- Infrared (IR) frequencies (300 GHz to 400 THz), detected as heat, are used in remote controls, night-vision devices, and infrared photography. Lidar systems are essentially radars that predominantly operate in the infrared frequency ranges.
- Visible light is the section of the EMS visible to the human eye, ranging from 400 THz (red) to 790 THz (violet).
- Ultra-violet sensors (750 THz to 3 PHz) detect UV radiation from jet engine combustion and missile plumes. UV and IR are often combined in multispectral systems because they detect different EMS wavelengths.

[†] RADAR was originally an acronym for *radio detection and ranging*.

energy, understand how it is being used, and determine whether there are any identifiable patterns that can be exploited.

- *Electronic protect (EP)* involves actions taken to protect the friendly nodes from any undesirable effects due to changes in the spectrum such as jamming or noise. This activity chooses strategies or countermeasures to maintain communications or radar performance. EP conversations revolve around anti-jamming and radar countermeasure techniques; in this book we use EP equally to discuss techniques used to protect communication and radar systems. Strategies may include frequency agility, waveform design, antenna direction, and signal processing to reduce the impact of jamming.
- *Electronic attack (EA)* denies the adversary access to its own RF spectrum. EA uses offensive EM energy to degrade or deny the adversary's access to the spectrum, or to deceive the adversary by conveying misleading information.

Deny, degrade, disrupt, deceive, destroy

- *Electronic battle management (EBM)* oversees all aspects of EMS operations to increase mission effectiveness, including managing changing mission priorities, coordinating effects, and collaborating with other elements of mission command. A key aspect is to interact with and support the EW officer.
- *Electronic warfare battle damage assessment (EW BDA)* assesses the effectiveness of the EA, and provides feedback that allows the operator or system to create more effective attacks. EW BDA modules feed into, and are often part of, the ES module.
- *Electronic warfare reprogramming (EW reprogramming)* changes the self-defense systems, offensive weapons systems, and intelligence-collection systems. EW reprogramming activities fall into three major categories—tactics, software, and hardware—and provide commanders with a timely capability to respond to changes in adversary threat systems, correct system deficiencies, and tailor equipment to meet unique theater or mission requirements.

Figure 1.6 illustrates the interactions between transmitters and receivers in communications and radar settings, showing the different goals of the corresponding EP and EA system. Figure 1.7 highlights some of the activities performed by ES, EP, and EA. The illustration suggests a strong separation between ES, EP, and EA, but in reality each action could potentially support each objective. For example, deceptive jamming could be an EP activity, and filtering techniques could improve

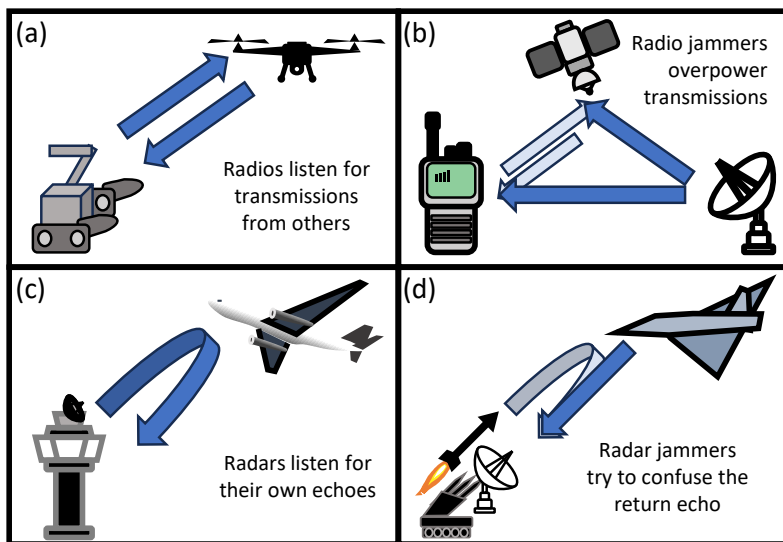


Figure 1.6 (a) Radios in a communications network perform EP to maintain communications. (b) Jammers performing communications EA transmit signals to overpower or confuse the receivers. (c) Radars listen for reflections of their own transmissions to estimate distance to objects based on the time difference from transmit to receive. (d) Radar jammers use EA techniques to overpower the reflection or try to confuse the timing estimates of the radar.

ES. Chaff, as another example, can be considered EP to defend against a threat, or EA to confuse an adversarial radar.

From an AI standpoint, EP and EA can be treated the same way; the only difference is that EP defines objectives with respect to oneself, while EA defines objectives with respect to the adversary (and thus is harder to measure).

The same applies to communications and radar systems, in that the available actions are similar, with expected effects and associated costs. Communications systems have more challenging distributed decision-making requirements, with more latency, more coordination, and a more nuanced set of performance metrics. Action choices will differ between radar and communications systems due to their mission requirements and principles of operation. Radar ES focuses on analyzing pulse characteristics, direction finding, and tracking emitters, while communications ES focuses more on demodulation and content analysis, signal classification, and protocol analysis. Radar EP uses techniques like pulse compression, polarization diversity, and moving target indicators, while communications EP tends to use encryption,

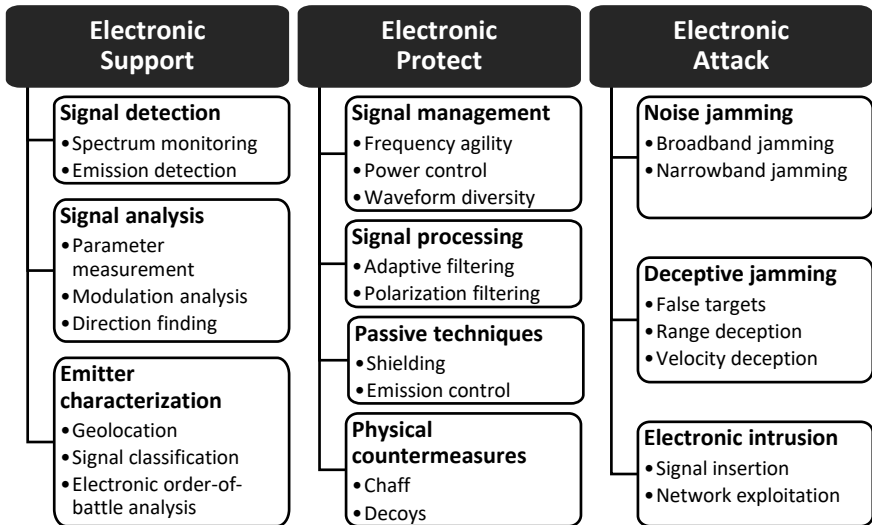


Figure 1.7 These example techniques illustrate how ES describes the spectrum, EP protects the system against undesirable effects, and EA denies the adversary access to its own EMS. Each technique may support multiple objectives.

error-correcting codes, and extensive frequency hopping. Radar EA uses actions such as deceptive jamming, physical chaff dispersion to obscure, and mainlobe and sidelobe jamming for signal disruption, while communications EA tends to use actions like spoofing to deceive the adversary, denial-of-service to deny access to the spectrum, and precision jamming to disrupt communication.

Making Sense of Radio Frequency is an easy-to-read introduction to RF concepts [15]. The *EW Handbook* [16] and *EW Fundamentals* [17] are good references for many EW equations and concepts.

1.5 Cognition in EW

Table 1.1 shows the relationship between EW concepts and the common AI terms introduced in Section 1.3. Many ideas are similar; for example the EW kill chain of *find-fix-track-target-engage-assess* is similar to the OODA and cognition loops.

Figure 1.8 illustrates how different AI concepts and techniques might appear in a traditional EW functional diagram. The illustration is not prescriptive; instead it aims to show that AI techniques can potentially contribute to solutions anywhere and everywhere within an EW system. Cognitive approaches apply equally to the

Table 1.1
Mapping Between EW Activities and AI Concepts

AI Term	EW Term
Situation Assessment	Electronic Support (ES)
Decision Making	Electronic Protect and Attack (EP and EA)
	Electronic Battle Management (EBM)
Execution Monitoring	Electronic Battle Damage Assessment (EW BDA)
	Effects Analysis
Learning	EW Reprogramming (of data and software)

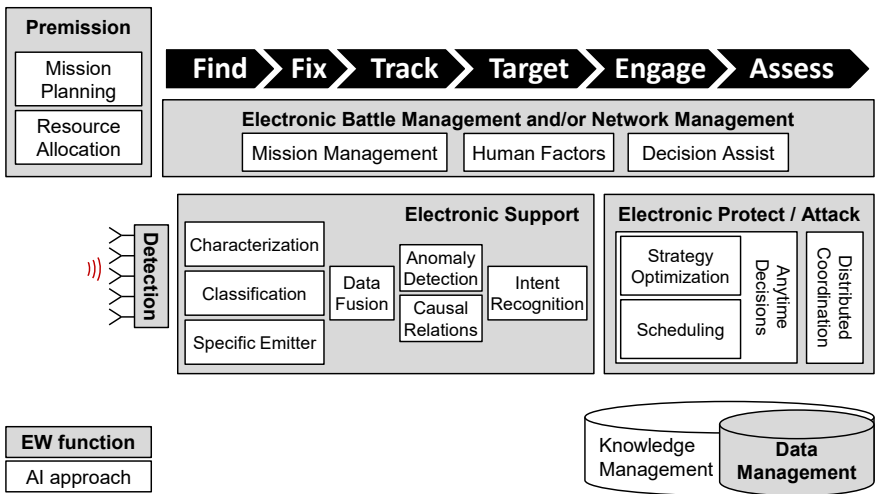


Figure 1.8 AI situation assessment, decision making, and learning capabilities are relevant for all EW functions.

sensors and to the actuators; we can use these techniques to improve both the receive and transmit activities. Chapter 11 describes an incremental development approach, where the goal is to identify pieces of the EW system that are particularly weak or brittle, and replace them with AI algorithms. Each of the major function boxes have corresponding sections in this book: ES in Chapter 4, EP and EA in Chapter 5, pre-mission planning, EBM, and human interactions in Chapter 6, and data management in Chapter 8.

EW is similar to a distributed robotics or swarming problem, in that it is a complex, dynamic, multiagent, resource-constrained problem. EW adds several additional complexities:

- Task negotiation goes beyond user-level tasks, and includes negotiating how to communicate, meaning that nodes may fall into non-interoperable configurations where they are unable to coordinate.
- Observations may be fleeting, at the order of nano- or pico-seconds.
- The decision timeline is extremely fast, usually subsecond and often submillisecond.
- A large antenna might collect a petabyte of data every minute. (Most of today's systems cannot *record* most of it.)
- A single system might have 1000 or more controllable actions available.

The complexity of EMS data means that no two platforms or two moments in time will record exactly the same observation, and mobility intensifies the problem.

AI techniques are being adopted as solutions to many of these problems. A CogEW system must characterize the environment, choose appropriate responses, and evaluate and learn from the effects within milliseconds of an observation. Each observation could be novel, outside the domain-of-expertise from the training data. A CogEW system, therefore, must learn after deployment and must learn from one or two examples in submillisecond timeframes. As the CogEW community applies AI techniques to the EW problem, CogEW should and will become another of the vertical AI domains mentioned in Figure 1.5. Solutions to our challenges will benefit the broader AI community.

In addition to the 3D representation of Figure 1.4, another way to view cognition in RF systems is through the levels outlined in Table 1.2. Level 1 represents traditional, static threat systems with no cognitive capability; levels 2-5 have increasing adaptivity and possibly some basic cognition; levels 6 and higher represent increasing cognitive capability.

1.6 Civilian EW

A common impression from both civilian and military communities alike is that EW is a “military” activity. However, it is becoming increasingly difficult to distinguish between military and civilian EW needs. The 2022 Japanese National Defense Strategy [20] recognizes this blend, stating “In the current security environment, the boundaries between military and civilian, peacetime and emergency, have become blurred.” Nowhere is this vision more true than in EW.

Civilian uses of EW techniques were first commonly seen by the police. Unintentional jamming of police radios started in the 1930s as a form of accidental

Table 1.2
Cognitive Capability Levels (From [18, 19])

Level	Cognitive Capability
1: Static parametrics	Select parameters and behaviors from a known static library; well-established models and simulations available.
2: Variable parametrics	Vary parameters without regard for environmental conditions; perform behaviors from a known static library.
3: Hidden or unknown modes	Vary parameters outside what was collected in the training set, but perform behaviors from a known static library; identification and/or type can be determined from behavior.
4: Adaptive behavior	Vary parameters and behaviors outside what was collected in the training set; type and/or intent may be discerned from behavior.
5: Environment adaptation	Change parametrics and behaviors according to the environment.
6: Cognitive responses	Recognize adversarial actions; react to mitigate effects.
7: Active engagement	Take actions to improve model quality, for example with active learning.
8: Counter-cognitive	Take actions to hide cognitive capabilities.

interference. By the 1950s, organized crime and activist groups used radio jammers to disrupt police coordination. The first civilian radar detector was invented in 1968 by Dale Smith, an electronic expert in Ohio, who had experience producing radar units for police [21]. He had received a speeding ticket that he believed was unfair.

In some ways, the September 11th terrorist attacks on New York City could be considered the first large-scale civilian EW event (albeit unintentional). The cellular network was overwhelmed by the number of calls made to loved ones. First responders resorted to a variety of alternatives, including landlines, two-way radios, satellite phones, and even human runners (“sneaker net”). While unintentional, the effect was essentially a denial-of-service attack on the cellular net.

We are seeing more EW in the coast guard, border patrol, antiterrorism, and antipoaching. EarthRanger is an antipoaching system developed under the Allen Institute for AI [22]. EarthRanger fuses data over many sensors to track animal and human movements, including Internet of Things (IoT) devices, animal

collars, handheld radios, camera traps, remote imagery, vehicle trackers, and aircraft trackers. There are many success stories worldwide in wildlife protection.

A good civilian scenario that demonstrates the critical need to automatically learn new waveforms is in counter-drone activities at airports. The first drone incident was reported at Gatwick airport in London, UK in 2018. In 2022 there were 2,554 drone incidents globally, and in 2024 the US Federal Aviation Administration reported approximately 150 incidents per month [23]. Pilots have had to take evasive action, and airports have been shut down.

These drone incidents highlight the need for a CogEW solution. The drone threat is evolving beyond the traditional signal types—primarily DJI Lightbridge (Lightstream) and 900MHz/2.4GHz MAVLink—toward the use of 5G-enabled¹ modems that resemble smartphones and may implement aspects of the still-developing 3GPP standards (e.g., Release 17 and UAV applications). A classical drone-protection system would be inherently reactive and require reprogramming and requalification with each new variant of a communications waveform. A cognitive system, meanwhile, would adapt its approach, but might have to learn that cell phones carried by helicopter passengers are not a threat. The cognitive system must respect guardrails while learning to increase its effectiveness.

In addition, to address evolving threats to 5G, 6G, and non-terrestrial networks (NTNs), CogEW solutions are essential. As these networks rely on both terrestrial and satellite infrastructure, they face a growing range of threats, including jamming, global navigation satellite system (GNSS) spoofing, denial-of-service attacks, and signal degradation. 5G and 6G systems, with their high throughput, ultra-low latency, and millimeter-wave frequencies, are particularly vulnerable to interference that can degrade performance or compromise security. Similarly, NTNs, which use satellite systems for global coverage, face additional risks such as vulnerabilities in satellite communications links and GNSS-based synchronization that can be disrupted or spoofed. As 5G, 6G, and NTN networks become central to IoT, critical infrastructure, and autonomous vehicles, securing them with CogEW solutions is vital.

The civilian and military EW problem has blurred sufficiently that the US's Cybersecurity and Infrastructure Security Agency (CISA) has published a set of EW recommendations for local public safety practitioners [24]. As commercial vendors develop more flexible software-defined EMS systems, EW will continue becoming more of a combined civilian/military challenge.

1. 5G is a set of fifth-generation commercial cellular standards developed by the 3rd Generation Partnership Project (3GPP) standardization body.

Table 1.3	
The Benefits of Cognitive Radio Systems and How They Transfer to Cognitive EW	
Cognitive Radio Benefits	Cognitive EW Benefits
RF communications, seamless to the user.	→ EMS operations must also be seamless to the user. Moreover, networked radar EW systems require underlying communications.
Self-regulating radio networks.	→ EW systems must also be self-configuring and self-regulating, particularly given the rapid requirements of the EW kill chain.
Better performance in terms of cost, speed, power consumption, availability, timeliness, and resource use.	→ Better performance in terms of cost, speed, power consumption, availability, timeliness, and resource use. EA activity, in communications or in radar, is also measured by effectiveness (EW BDA).

1.7 Cognitive Radio, Cognitive Radar, and Cognitive EW

Cognitive radio, cognitive radar, and cognitive EW can be viewed as three sides of the same coin. All three rely on a comprehensive understanding of the channel to determine appropriate actions needed to accomplish specific objectives. A fully adaptive cognitive RF system must probe the channel to accurately characterize RF conditions. The primary difference between EW applications and cognitive radio or radar applications is a direct acknowledgment of the challenging operating conditions, whether those conditions are intentionally created by adversaries or arise unintentionally from the environment.

As an example, consider cognitive radio networks (CRNs), which accomplish the EP objectives of CogEW systems in communications. Table 1.3 outlines some of the potential and well-known benefits of cognitive radio that are directly applicable to CogEW. Cognitive radio concepts were developed to address RF spectrum usage efficiency [25–27]. Cognitive radio technology is a transformative force within spectrum management. All EMS operations can leverage these novel spectrum management approaches and opportunistic spectrum access. Furthermore, opportunistic spectrum access implies that cognitive radio engines have achieved a certain level of spectrum situation awareness, which is an important aspect of an intelligent EW system as well. CRNs also look at heterogeneous radio frameworks’ management, a concept also relevant for multiagent, heterogeneous EW systems [28–31]. 5G also has similarities to the EW environment; Callout 11.2 outlines use cases and some cognitive approaches. Much of the published CRN work deals with multiaccess

issues, not spectrum use in an adversarial environment. Moreover, CRN research usually assumes cooperative entities with similar goals, while EW scenarios are always adversarial. Chapter 5 describes approaches to optimizing in the adversarial setting.

Just as cognitive radio leverages opportunistic spectrum access to optimize communications, cognitive radar exploits real-time environmental awareness to maximize detection and tracking performance in the presence of adversarial interference [2, 32, 33]. Unlike traditional radar systems, which operate with predetermined waveforms and processing techniques, cognitive radar dynamically adapts its transmission and signal processing strategies based on environmental feedback. Techniques for cognitive radar such as data fusion, waveform design, resource allocation, and spectrum sharing map directly into EW applications. Moreover, multifunction systems that fuse communications and radar are becoming more prevalent, for example in 5G networks that perform radar functions for localization (Table 4.2). Situation assessment, decision making, and ML come together to support cognitive radar capabilities in the same way as for CogEW. Finally, by integrating cognitive radar concepts into EW frameworks, systems can enhance capabilities by countering sophisticated jamming techniques, improving target discrimination, and optimizing resource allocation in contested environments.

1.8 EW System Design Questions

CogEW system design must provide situation assessment, decision making, and learning. System requirements drive decisions about which components to incorporate, and what questions to ask during system design [25, 32, 34, 35]:

- *Sensing*: How do we assess the accuracy of spectrum sensing capabilities? Can we improve sensor accuracy? How do we leverage cooperative sensing mechanisms across spatially diverse nodes without incurring untenable latency? Can we leverage redundancy in data feeds to compensate or correct for failures? How do we address other topics such as long-range RF sensing, multifunctional RF sensing, and passive RF sensing? Section 4.4 discusses some data fusion concepts, while Section 8.2.2 discusses the need to record not only the data, but also its uncertainty and origin.
- *Decision making*: Should decisions be centralized or distributed? What decision-making algorithms should be used? How do we define optimization functions? Can we leverage the physics of the domain and progression of the engagement to reduce the state space? Chapter 5 describes decision making for EP and EA, while Chapter 6 describes decision making for EBM, and Chapter 7 describes in-mission decision making.

- *Learning*: How do we define appropriate learning tasks? Should learning be supervised, unsupervised, semisupervised, or self-supervised? What measurements (features) should form the basis of the learning task? How do we handle the combinatorial explosion of the data that captures all past observations and decisions? Chapter 4 describes learning tasks for ES, Chapter 7 describes in-mission learning, and Chapter 8 describes data management.
- *Security*: When do we need to encrypt data? How do we ensure policies to avoid security violations, particularly given the highly dynamic environment and missions? How do we protect data and models without jeopardizing performance, particularly accuracy and latency? Can we ensure that even if a model is compromised, it cannot be reverse-engineered? Section 8.3.5 talks about some security issues.
- *Software architectures*: How do we address global optimization of an EW system, both globally across the network of platforms, and internally within each node across each module? Section 9.1 describes software architectures in more detail. Section 5.5 discusses distributed optimization.
- *Hardware design*: How do we design RF systems that are amenable to cognitive control? Can we effectively design multifunction hardware systems that can handle cognitive configurations? What memory and compute capabilities can we incorporate within the SWaP constraints of the platform? Because not all data can be recorded, what data compaction techniques effectively minimize data loss? Section 9.4 outlines some of the hardware considerations.

Chapter 11 presents a developer's guide that answers these questions.

1.9 EW Domain Challenges Viewed from an AI perspective

Like any other intelligent system, CogEW must overcome challenges associated with each AI concept or stage of the cognition loop. The domain is challenging to understand, and the decision space is large and complex. User requirements add an additional layer of complexity to the system.

1.9.1 Situation Assessment for Electronic Support and EW Battle Damage Assessment

The dynamics and complexity of the RF/EW domain create a number of challenges for situation assessment for ES and EW BDA.

- *Dynamic*: Nothing in a distributed CogEW system is static. Most assessments of the current situation are valid only fleetingly, and may have even expired before decision-making activity begins.

- *Ambiguous*: Some observations may have multiple root causes or imply multiple conclusions. Detection and understanding of a change in situation is not always simple. For example, how does the system automatically tell the difference between short-term fade versus entering a building?
- *Partially observable*: Many factors that impact the EMS cannot be observed. Few platforms, for example, incorporate a “fog” sensor. EW BDA is notoriously difficult to accomplish accurately. Some nodes may be able to sense an event, while other nodes cannot.
- *Complex interactions*: Many control parameters have poorly understood interactions with each other and with system performance. While specific pair-wise interactions can often be identified, such as that increased power reduces battery life, generally these pair-wise interactions are studied in laboratory environments, and the conclusions do not transfer directly to a fielded system. System calibration in the field addresses some pair-wise interactions, but is typically limited to the most significant performance parameters because it is time-consuming and expensive to perform. Interactions across many parameters are rarely studied, even in the lab.
- *Complex temporal feedback loops*: Within a node, certain activities occur at very rapid speeds requiring a very tight feedback loop to support cognitive control. Other activities occur on a longer timescale that require analyzing a wider range of factors over broader windows of time. Between nodes, there is yet a longer feedback loop between decisions and their effects. The variety of temporal loops and their dramatic feedback latency differences means that correlating cause and effect of actions is particularly challenging.

These characteristics combine to create the significant challenge of *lack of training data*. In a complex EW domain, it is impossible to collect sufficient examples of how the system behaves. While it may be possible to collect many examples of WiFi performance, where nodes are somewhat mobile, this is not true for highly mobile settings, multiple weather patterns, complex missions, and novel waveforms. Synthetic data can represent some of this variation (e.g., channel effects), but this data will be neither realistic enough or comprehensive enough to cover all situations. Chapter 8 addresses the training data challenge.

This challenge is exacerbated given the adversarial setting: A true representation of adversary tactics and capabilities is likely not available for day 0 operations, and becomes increasingly true for day 1 to N as the situation in a conflict changes.

The EW system must be able to learn from very small numbers of novel examples, even during a mission.

1.9.2 Decision Making for Electronic Attack, Electronic Protect, and Electronic Battle Management

Many EW domain characteristics also create challenges for decision making for EA, EP, and EBM. The operating environment and available equipment impose constraints that the decision maker must conform to.

- *Dynamic*: Military missions change, user requirements change, platforms join or leave the ensemble, hardware fails, and mobility causes continuous fluctuations in communications connectivity. Each change increases the complexity of the decision-making activity.
- *Resource constrained*: Nodes typically have tight constraints on size, weight, and power (SWaP). Power management is a critical requirement for remote operations. A given node may have the ability to accomplish a given EW task but will exhaust its resources when doing so. One strategy to reduce the resource use on a single node is to distribute the task across multiple nodes, but communications is *also* a resource that the system must manage.
- *Distributed*: As a result of the limited communication and frequent disconnections, nodes must intentionally trade off the benefit of rapid local decision making against the advantages that can be gained by coordinating across nodes. A stand-alone EW system or platform has significantly lower latencies than an internetworked EW system.
- *Diverse*: Nodes in a distributed EW environment may have a wide variety of capabilities, from small handheld radios, to large platforms with satellite communications (SATCOM), or aircraft with modern radar systems. Nodes may have different RF antenna components and supporting computational infrastructure such as memory and compute. Diversity also exists in the operation of similar equipment, sensors, produced by different manufacturers or different revisions. This heterogeneity requires different solutions on different nodes, causing additional diversity in the software layers.
- *Massive scale*: There are many observable and controllable parameters (possibly continuous-valued) to configure *per node*. No current system exposes all of theoretically available control parameters. The maximum number of strategies that each node can take, per timestep, is $\prod_{\forall c} v_c$, where v_c is the number of possible values for the control parameter c . Even if all $\overline{c_n}$ control parameters are simply binary on/off, then there are $2^{\overline{c_n}}$ strategies per node, per timestep. When even one controllable is continuous-valued, there is an infinite number of configurations.

- *Macroscale impacts:* Elements essentially unrelated to the direct EW engagement can affect outcomes. For example, red platform maneuvers that exceed their antenna coverage, or lack of or incidental illumination during red threat engagement with other platforms both impact mission success.

These challenges create a distributed, heterogeneous, low-communication, partially observable, high latency, exponential optimization problem. Cognitive control in the general case is therefore never simple: The level at which symptoms appear may not be the level where node configuration changes must be made; symptoms may be ambiguous at one level or time and require more context; changes in one module may impact other modules and cause new issues; and the timing of changes may be critical. In practice, however, the challenge is not as great as it seems. First, from an engineering perspective, the “true” optimal is not required, and in practice there may be many “good enough” solutions (Section 5.4). Second, physics and progression of the engagement both constrain the problem.

In other domains, AI techniques have addressed the full richness of most of these challenges. In the cognitive radio and EW domains, AI techniques are just beginning to scratch the surface. AI techniques have addressed many of these challenges in physically embodied systems that have some similar characteristics to the EW domain (e.g., robotics). We need to bring these techniques into EMS operations, and address them in depth.

The one notable exception where AI has only just begun to explore is that of automatic heterogeneous intercommunication. There has historically been a very strong norm in the networking community that all nodes must be designed and (preferably statically) configured to interoperate [36]; typical ad hoc networks are built from a group of homogenous nodes. CRNs break this assumption: Each node can have an independent cognitive controller, and thus network nodes *may be heterogeneous, and may fall into noninteroperable configurations*.² Meanwhile traditional AI has always assumed that communication is “safe,” negotiating and coordinating only the application-level tasks [37, 38]; moreover, they also generally require very high communications overhead. These assumptions—homogeneity and safe communications—fall squarely into the set of eight distributed computing fallacies (Figure 1.9). ADROIT [40], by giving each node its own learning system, represented a *radical* departure from the traditional networking stance that requires homogeneous configurations. ADROIT was the first system to demonstrate an effective *heterogeneous* mobile ad hoc network (MANET) and, moreover, used

2. The alternative is to have one cognitive controller for several nodes, but communication overhead and latency increases dramatically and intelligent control is vulnerable to network partitions.

Fallacies of Distributed Computing	
1. The network is reliable	5. Topology doesn't change
2. Latency is zero	6. There is one administrator
3. Bandwidth is infinite	7. Transport cost is zero
4. The network is secure	8. The network is homogeneous

Figure 1.9 When designing a distributed CogEW system, one must not fall victim to the classical eight fallacies of distributed computing defined in 1994 by Peter Deutsch and James Gosling [39]. These fallacies are more acute in EW.

ML to configure the nodes. Section 5.5 discusses approaches to addressing the distributed coordination challenge. The solution is to design for fully disconnected operations, and leverage connectivity when available.

1.9.3 User Requirements

Finally, the following user requirements add another layer of complexity to the CogEW system:

- *Complex access policies:* Due to the heterogeneous nature of the data, nodes, and mission, access policies may restrict the set of nodes that are permitted to hold or transmit specific data.
- *Complex multiobjective performance requirements:* Multiple users have interacting requirements and policies, thus requiring a complex multiobjective function that captures mission and environmental standpoints [41]. It is rarely possible to optimize all objectives simultaneously. Instead, the system must choose a particular operating point that balances the trade-offs. Examples include resource use, joint optimization of ES/EP/EA, joint optimization of comms and radar, and accuracy such as probability of detection (Pd) versus probability of false alarms (Pfa).
- *Constrained operations:* All CogEW system designers desire real-time, in-mission ML. The concern from a user perspective is that the system may learn inappropriate behaviors. Requirements must simultaneously constrain emergent behavior, and yet address novel, unexpected situations.

Ultimately, these factors lead to how much the human stakeholders *trust* the information gathering, situation assessment, and decision making of the AI system. Stakeholders include the acquisitions community, who want “verifiable” and “guaranteed” performance, as well as the individuals—electronic warfare officers (EWOs), aircrew, sailors, and soldiers—whose lives depend on pragmatic actions. Section 5.1.1 discusses multiobjective optimization. Section 6.3 discusses the human-machine team, and Chapter 10 presents mechanisms for verification and validation of AI-enabled system.

1.10 Reader’s Guide

The chapters in this book walk through the concepts required for building a fully CogEW system, organized by the components in Figure 1.8.

- Chapter 2 describes the objective function that drives decision making.
- Chapter 3 presents a short primer³ on ML, including a discussion of algorithmic trade-offs.
- Chapter 4 explains how to assess an ES situation.
- Chapter 5 describes how to choose a strategy for EP and EA in time-constrained and distributed settings.
- Chapter 6 presents EBM and the human interface, dealing with planning, including resource management, uncertainty, and adversaries.
- Chapter 7 examines online replanning and learning, including EW BDA, which maps expected outcomes to observed outcomes.
- Chapter 8 presents data management processes and practice.
- Chapter 9 lists software and hardware architecture considerations.
- Chapter 10 presents evaluation: how to test and what to test.
- Chapter 11 condenses all the lessons learned into recommendations on how to get started building a CogEW system.

Much of the discussion in this book is based on the *BBN Strategy Optimizer (BBN SO)*, a communications EP system that learns how to respond to any interference in milliseconds on a PPC440 CPU with no floating-point coprocessor. Example 1.1 presents the core capabilities of the BBN SO.

1.11 Conclusion

EW is no longer a hardware-only world. Software is a key part of EW systems, requiring AI to configure parameters and optimize performance. Building a CogEW

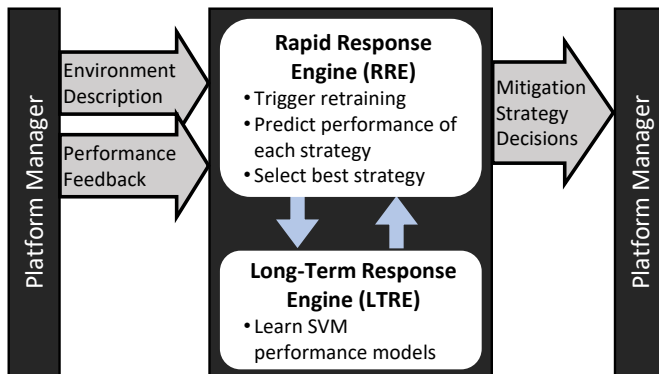
3. The word *primer* is pronounced *primmer* using the *i* from *sit* when it covers the basic elements of a subject. It is pronounced *pr-eye-mer* using the *i* from *five* when it is an underlayer of paint.

Example 1.1 The BBN Strategy Optimizer performs real-time in-mission learning and optimization for communications EP.

The BBN SO learns how to optimize network performance in the presence of previously unknown interference and jamming conditions [42–44]. The SO automatically recognizes conditions that affect communications quality, and learns to select a configuration that improves performance, even in highly dynamic missions. The SO is the first known communications EP system to use ML *in-mission*, at mission-relevant timescales.

The SO effectively learns how to mitigate communications interference, in real time, on limited hardware, from small numbers of examples, even when given no a priori training data.

The SO comprises a rapid response engine (RRE) that makes strategy decisions, and a long-term response engine (LTRE) that learns the models of how strategies perform.



The BBN SO demonstrates:

- *Rapid adaptive decision making* that selects actions in real time to optimize mission performance;
- *Incremental reinforcement learning* that learns the effectiveness of available actions in complex, changing, and novel environments; and
- A *semantically agnostic architecture* that supports easy deployment to new platforms and domains.

The SO has been tested against many different interference conditions, from distributed interference to complex jammers, in simulation, in lab emulation, and in the field. Results are shown on a PPC440 with a hard real-time operating system [42].

Given a set of training data (possibly empty), the LTRE constructs a model f_k of the performance surface for each metric m_k : $m_k = f_k(o, s)$ for observables o and strategy s . The SO uses SVMs (Section 3.1.1) as a regression tool to estimate performance (Section 4.3). The SO uses SVMs because they can learn from small numbers of training examples, and because they can be implemented on limited hardware. Because memory is limited on this device, the LTRE manages data diversity (Section 8.3.2) and forgets old instances (Section 8.3.4).

The RRE uses these SVM models f_k to make rapid real-time strategy decisions during the mission using Algorithm 5.1. When it detects an error, the RRE instructs the LTRE to relearn the models. The SO therefore uses RL to perform in-mission learning at mission-relevant timescales.

The SO is agnostic to platform and problem domain. All observables, controllables, and metrics are listed in a configuration XML. Controllables list their valid settings, and metrics list their associated weights and costs. The learning and optimization approach is therefore transferable to other platforms, problem sets, and utility metrics.

The BBN SO is part of a family of research into ML for RF systems starting in 2006 [34, 36, 41–48]. ADROIT [46] was the first to use ML to control a real-world (not simulated) MANET, demonstrating control of network, MAC, and PHY parameters for an application-level map quality metric. The BBN SO [42] was the first communications EP system to use ML to learn at real time during the mission, demonstrating control of antenna, PHY, and MAC layer parameters for EP. Early work used ANNs and achieved the accuracy at the expense of training time and large training sets. SVMs solved both of those challenges. The BBN SO demonstrates that:

- The system *must* learn in-mission to handle novel emitters;
- The model *can* learn from a single example;
- A diverse dataset produces models that *generalize* to new experiences; and
- A well-curated small dataset that fits with embedded system memory limits can capture the *complexity* of the domain.

system requires understanding what and where AI can help: situation assessment for ES and understanding the RF environment, decision making for choosing EP/EA/EBM actions, and ML for continuous improvement.

This high-tempo complex environment is well-suited to the application of AI, but there are challenges to developing fieldable CogEW systems that can operate in that environment. A fully CogEW system needs algorithmic advancements for learning and inference, developments in decision making at tempo, data management approaches that annotate characteristics, and architectures that support cognitive reasoning.

However, you can start with a small capability, and grow it for the problem at hand; Chapter 11 describes the incremental steps.

A journey of a thousand miles begins with a single step.

— Chinese proverb, *Dào Dé Jīng* by LǎoZi, 6th century BCE

References

- [1] Mitola III, J., “Cognitive Radio for Flexible Multimedia Communications,” in *International Workshop on Mobile Multimedia Communications*. IEEE, 1999. DOI: 10.1109/MOMUC.1999.819467.
- [2] Haykin, S., “Cognitive Radar: A Way of the Future,” *IEEE Signal Processing Magazine*, Vol. 23, No. 1, 2006. DOI: 10.1109/MSP.2006.1593335.
- [3] Shettleworth, S., *Cognition, Evolution, and Behavior*, 2nd Edition. Oxford, UK: Oxford University Press, 2009.
- [4] Haykin, S., *Cognitive Dynamic Systems: Perception-action Cycle, Radar, and Radio*. Cambridge, UK: Cambridge University Press, 2012.
- [5] Boyd, J., *Destruction and Creation*, 1976. Online: http://www.goalsys.com/books/documents/DESTRUCTION_AND_CREATION.pdf. Unpublished essay.
- [6] Horne, C., M. Ritchie, and H. Griffiths, “Proposed Ontology for Cognitive Radar Systems,” *IET Radar, Sonar and Navigation*, Vol. 12, No. 12, 2018. DOI: 10.1049/iet-rsn.2018.5280.
- [7] Bräuer, J., et al., “Old and New Approaches to Animal Cognition: There Is Not ‘One Cognition’,” *Journal of Intelligence*, Vol. 8, No. 3, 2020. DOI: 10.3390/jintelligence8030028.
- [8] Kline, R., “Cybernetics, Automata Studies, and the Dartmouth Conference on Artificial Intelligence,” *IEEE Annals of the History of Computing*, Vol. 33, No. 4, 2011. DOI: 10.1109/MAHC.2010.44.
- [9] Russell, S., and P. Norvig, *Artificial Intelligence: A Modern Approach*. London, UK: Pearson Education, 2015.
- [10] Horvitz, E., and T. M. Mitchell, “Scientific Progress in Artificial Intelligence: History, Status, and Futures,” in *Realizing the Promise and Minimizing the Perils of AI for Science and the Scientific Community*. 2024. Online: <https://lnkd.in/gXFJFnAC>.
- [11] Chairman of the Joint Chiefs of Staff, *Joint Publication 3-13.1: Electronic Warfare*, 2012. Online: <https://fas.org/irp/doddir/dod/jp3-13-1.pdf>.

- [12] De Martino, A., (Ed.), *Introduction to Modern EW Systems*. Norwood, MA: Artech House, 2013. Online: <https://studylib.net/doc/18347338/introduction-to-electronic-warfare-scenarios>.
- [13] US Air Force, Curtis E. Lemay Center for Doctrine Development and Education, *Annex 3-51: Electromagnetic Warfare and Electromagnetic Spectrum Operations*. 2019.
- [14] Chairman of the Joint Chiefs of Staff, *Joint Publication 3-85: Joint Electromagnetic Spectrum Operations*, 2020. Online: <https://tinyurl.com/af-jemso-202>.
- [15] Cambridge Radio Frequency Systems, *Making Sense of Radio Frequency (RF)*, 2023. Online: <https://www.crfs.com/making-sense-of-radio-frequency>.
- [16] Avionics Department, "Electronic Warfare and Radar Systems Engineering Handbook," Naval Air Warfare Center Weapons Division, Tech. Rep. NAWCWD TP 8347, 2013. Online: <https://tinyurl.com/navy-ew-2013>.
- [17] Air Combat Command (ACC) Training Support Squadron (TRSS), D. 8., *Electronic Warfare Fundamentals*, 2000. Online: <https://falcon.blu3wolf.com/Docs/Electronic-Warfare-Fundamentals.pdf>.
- [18] Plyler, D., "Graduated Complexity in Cognitive Radar Threat Simulations for Cognitive EW T&E," in *Dixie Crows EW Symposium*, Warner Robins, GA, 2023.
- [19] Haigh, K. Z., and T. Nguyen, "Challenges of Testing Cognitive EW Systems," in *IEEE AutoTestCon*, 2023. DOI: 10.1109/AUTOTESTCON47464.2023.10296319.
- [20] Kishida administration, *National Security Strategy of Japan*, 2022. Online: <https://www.cas.go.jp/jp/siryou/221216anzenhoshou/nss-e.pdf>.
- [21] Santo, B., *The Consumer Electronics Hall of Fame: Electrolert Fuzzbuster Radar Detector*, 2018. Online: <https://tinyurl.com/civilian-radar-detector>.
- [22] Vulcan LLC, *EarthRanger*. Online: <https://www.earthranger.com/>. Accessed 2024-11-20.
- [23] Federal Aviation Administration, *UAS Sightings Report*. Online: https://www.faa.gov/uas/resources/public_records/uas-sightings-report. Accessed 2024-11-20.
- [24] Cybersecurity and Infrastructure Security Agency, "Radio Frequency Interference Best Practices Guidebook," Tech. Rep., 2020. Online: <https://tinyurl.com/rf-interference-guide>.
- [25] Nguyen, V. T., F. Villain, and Y. Le Guillou, "Cognitive Radio RF: Overview and Challenges," *VLSI Design*, Vol. 2012, 2012. Online: <https://tinyurl.com/CR-RF-overview>.
- [26] Mitola, J., and G. Maguire, "Cognitive Radio: Making Software Radios More Personal," *IEEE Personal Communications*, Vol. 6, No. 4, 1999. Online: <https://tinyurl.com/cr-mitola-personal>.
- [27] Haykin, S., "Cognitive Radio: Brain-Empowered Wireless Communications," *IEEE Journal on Selected Areas in Communications*, Vol. 23, No. 2, 2005. DOI: 10.1109/JSAC.2004.839380.
- [28] Trigui, E., M. Essegir, and L. M. Boulahia, "On Using Multi Agent Systems in Cognitive Radio Networks: A Survey," *International Journal of Wireless & Mobile Networks*, Vol. 4, No. 6, 2012. DOI: 10.5121/ijwmn.2012.4601.
- [29] Rizk, Y., M. Awad, and E. W. Tunstel, "Decision Making in Multiagent Systems: A Survey," *IEEE Transactions on Cognitive and Developmental Systems*, Vol. 10, No. 3, 2018. DOI: 10.1109/TCDS.2018.2840971.
- [30] Dorri, A., S. S. Kanhere, and R. Jurdak, "Multi-Agent Systems: A Survey," *IEEE Access*, Vol. 6, 2018. DOI: 10.1109/ACCESS.2018.2831228.
- [31] Tang, H., and S. Watson, "Cognitive Radio Networks for Tactical Wireless Communications," Defence Research, and Development Canada-Ottawa Research Centre Ottawa, Tech. Rep., 2014. Online: <https://apps.dtic.mil/sti/pdfs/AD1004297.pdf>.

- [32] Guerci, J., *Cognitive Radar: The Knowledge-Aided Fully Adaptive Approach*, 2nd Edition. Norwood, MA: Artech House, 2020.
- [33] Mishra, K., B. Shankar, and M. Rangaswamy, (Eds.), *Next-Generation Cognitive Radar Systems*. Raleigh, NC: SciTech Publishing, 2023.
- [34] Haigh, K. Z., "AI Technologies for Tactical Edge Networks," in *MobiHoc Workshop on Tactical Mobile Ad Hoc Networking*, 2011. Online: <http://www.cs.cmu.edu/~khaigh/papers/Haigh-MobiHoc2011.pdf>, Keynote.
- [35] Čabrić, D., "Cognitive Radios: System Design Perspective," Technical Report No. UCB/EECS-2007-156, Ph.D. dissertation, Electrical Engineering, and Computer Sciences, University of California at Berkeley, 2007. Online: <https://www2.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-156.pdf>.
- [36] Haigh, K. Z., et al., "Rethinking Networking Architectures for Cognitive Control," in *Microsoft Research Cognitive Wireless Networking Summit*, 2008. Online: <http://www.cs.cmu.edu/~khaigh/papers/haigh-msr08.pdf>.
- [37] Modi, P., et al., "ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees," *Artificial Intelligence*, Vol. 161, No. 1-2, 2005. DOI: 10.1016/j.artint.2004.09.003.
- [38] Zhang, X., V. Lesser, and S. Abdallah, "Efficient Management of Multi-Linked Negotiation Based on a Formalized Model," *Autonomous Agents and Multi-Agent Systems*, Vol. 10, No. 2, 2005. Online: <http://mas.cs.umass.edu/paper/384>.
- [39] Thampi, S., *Introduction to Distributed Systems*, 2009. Online: <https://arxiv.org/abs/0911.4395>.
- [40] Troxel, G., et al., "Enabling Open-Source Cognitively-Controlled Collaboration Among Software-Defined Radio Nodes," *Computer Networks*, Vol. 52, No. 4, 2008. DOI: 10.1016/j.comnet.2007.11.010.
- [41] Haigh, K. Z., O. Olofinboba, and C. Y. Tang, "Designing an Implementable User-Oriented Objective Function for MANETs," in *Networking, Sensing and Control*. IEEE, 2007. Online: <http://www.cs.cmu.edu/~khaigh/papers/Haigh07-ICNSC.pdf>.
- [42] Haigh, K. Z., et al., "Parallel Learning and Decision Making for a Smart Embedded Communications Platform," BBN Technologies, Tech. Rep. BBN-REPORT-8579, 2015. Online: <http://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-SO.sm.pdf>.
- [43] Haigh, K. Z., et al., "Machine Learning for Embedded Systems: A Case Study," BBN Technologies, Tech. Rep. BBN-REPORT-8571, 2015. Online: <https://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-Embedded.pdf>.
- [44] Haigh, K. Z., et al., "Optimizing Mitigation Strategies: Learning to Choose Communication Strategies to Mitigate Interference," in *Classified US Military Communications*, 2013.
- [45] Haigh, K. Z., S. Varadarajan, and C. Y. Tang, "Automatic Learning-based MANET Cross-Layer Parameter Configuration," in *Workshop on Wireless Ad Hoc and Sensor Networks*. IEEE, 2006. Online: <https://www.cs.cmu.edu/~khaigh/papers/haigh06a-configuration.pdf>.
- [46] Troxel, G., et al., "Cognitive Adaptation for Teams in ADROIT," in *GLOBECOM*. IEEE, 2007. doi: 10.1109/GLOCOM.2007.923, Invited.
- [47] Haigh, K. Z., et al., "Modeling RF Interference Performance," in *Collaborative Electronic Warfare Symposium*, 2014.
- [48] Haigh, K. Z., "Learning to Optimize a Network Overlay Router," in *Learning Methods for Control of Communications Networks*, 2017. Online: <https://www.cs.cmu.edu/~khaigh/papers/2017-haigh-rldm.pdf>.

2

Objective Function

The EW decision-making challenge is to build a cognitive controller that automatically maintains near-optimal configurations in highly dynamic environments. Every decision-making problem has:

- A set of *objectives* for the system to accomplish; each objective has an established importance;
- A set of *actions* is available for the system to use to accomplish the objectives; and
- A set of *methods* that evaluate the state to determine consequences (e.g., reinforce success and handle adverse consequences).

Armed with these three components, the system can choose the most appropriate actions for the evolving mission conditions. This chapter describes the heart of the EW decision maker: the set of objectives. Objectives must be operationally relevant, and optimizable in practice [1–4]. We must therefore capture mission goals and constraints in a mathematically optimizable *objective function*, also known as a *utility function*. Goals and constraints come from user and mission needs, environmental constraints, and equipment capabilities.

The utility function measures all the factors by which to evaluate a decision, so the system can choose the “best” solution. Consider an EW system with N nodes; each node $n \in N$ has:

- A set of *observable parameters* $\mathbf{o}_n$ that describe the RF environment (Section 2.1),
- A set of *controllable parameters* $\mathbf{c}_n$ that the decision maker can use to change system behavior (Section 2.2),
- One or more *metrics* $\mathbf{m}_n$ that provide feedback on how well it is doing, along with their corresponding *weights* $\mathbf{w}_n$ (Section 2.3), and
- A *utility function* $\tilde{U}$ that combines the metrics and weights into a single scalar-valued result (Section 2.4).

A *strategy* s_n is a combination of control parameters $\mathbf{c}_n$. The goal is to have each node n choose its strategy s_n to maximize utility $\tilde{U}_n$ of the system.

Table 2.1
Symbols That Support the Construction of a Utility Function

Symbol	Definition
$n \in N$	A node n in the set of nodes N
t	Timestamp
$\mathbf{o}_n$	Observables o for node n ; $\mathbf{o}_n(t)$ is the value of o_n at time t ; Denote $\overline{o_n} \triangleq o_n $
$\mathbf{z}$	Unobservable contextual information
$\mathbf{c}_n$	Controllables c for node n ; $\mathbf{c}_n(t)$ is the value of c_n at time t ; Denote $\overline{c_n} \triangleq c_n $
$\mathbf{m}_n$	Metrics m for node n ; Denote $\overline{m_n} \triangleq m_n $
$\mathbf{w}_n$	Weights corresponding to each metric in m_n ; $ w_n = \overline{m_n}$
s_n	Strategy s for node n ; combination of controllable values c_n
$\mathcal{U}$	The true utility function with no corresponding exact analytical expression
$\tilde{U}_n$	Node n 's local estimate of the utility
$\tilde{\mathcal{F}}_n$	Node n 's local model of the performance surface
U_n	Observed utility at node n
$\hat{U}_n$	Best-measured (or optimal) utility at node n
g_n	A function over metrics m_n and weights w_n to evaluate utility $\tilde{U}_n$
f_{nk}	A function over observables o_n and controllables c_n to evaluate the k^{th} metric of node n : $m_{nk} = f_{nk}(o_n, c_n) = f_{nk}(o_n, s_n)$ or simply $m_k = f_k(o, s)$

Table 2.1 collects these mathematical symbols. In the main text of this book, we generally use informal notation for these concepts; Section 2.4 presents a formal definition of the true utility function $\mathcal{U}$, and its approximation per node $\tilde{U}_n$.

Depending on system requirements, a strategy may contribute to ES objectives, EP objectives, or EA objectives, or all three. Likewise, a strategy may contribute to communications objectives, or radar objectives, or both. AI is agnostic to the problem domain, and does not discriminate between objective types or problem domains. It is the problem definition that determines problem-specific observables, controllables, objectives, and heuristics.

*AI techniques apply equally to ES, EP, and EA,
and communications or radar. The utility function
ties AI techniques to the problem domain.*

2.1 Observables that Describe the Environment

In EW, observable features, or *observables*, are general descriptors of the signal environment, and any relevant contextual information. Observables are computed by ES modules. Each node $n \in N$ has its own set of observables, $\mathbf{o}_n$, and these may differ from other nodes $n' \in N$. There are $\overline{o_n}$ observables at each node. Observables

may include raw values, computations from physics models, deduced abstractions, or inferred concepts:

- Raw receiver in-phase/quadrature (I/Q) values.
- Descriptions of all detected emitters (friendly, neutral, or adversarial). These features might include noise levels, error rates, Gaussianness, repetitiveness, similarity to own communications signal, or geolocation information, emitter capabilities, and current operational mode.
- Descriptions of all detected targets, including carrier, pulse statistics, and Doppler.
- Descriptions of receiver state, such as saturation, or antenna characteristics.
- Descriptions of internal software state. In a cognitive radio setting, these include statistics from the IP stack or end user applications, such as message error rate, queue lengths, and neighborhood size.
- Descriptions of the mission, including known or forecasted locations, tasks, users, and commander's intent.
- Descriptions of the environment, such as temperature, dust, or topography.

Figure 2.1 illustrates a handful of observables that capture different communication jammer types. Each detected emitter is associated with its own vector of observables.

Generally, when derived or inferred features can be abstracted or normalized relative to expectations, the decision maker can more easily determine the relative importance of different features. For example, observables might range from -1 to 1 (strongly “is not” to strongly “is”) relative to expectations. Partially observable values, or values not computed, can be captured with NaN.

There are many different visualization tools useful for understanding how each environment is different. Figure 2.2 shows four different ways to look at the data; each may be useful for different reasons. The bar charts use the mean values of each observable to give a quick gestalt sense. The box plots shows the distribution of values. The scatter plot shows detailed distributions of each observable for every observation of the environment.

Note that there may be *unobservable* contextual information, z . These hidden variables may include broad environmental factors like politics, wind farms, or weather, or detailed concepts like undetected emitters or failing hardware components. In theory, every butterfly could impact the performance of the system; in practice, designers should attempt to capture all relevant factors. Unobservable factors are the primary source of modeling error in ES. Models of physical systems

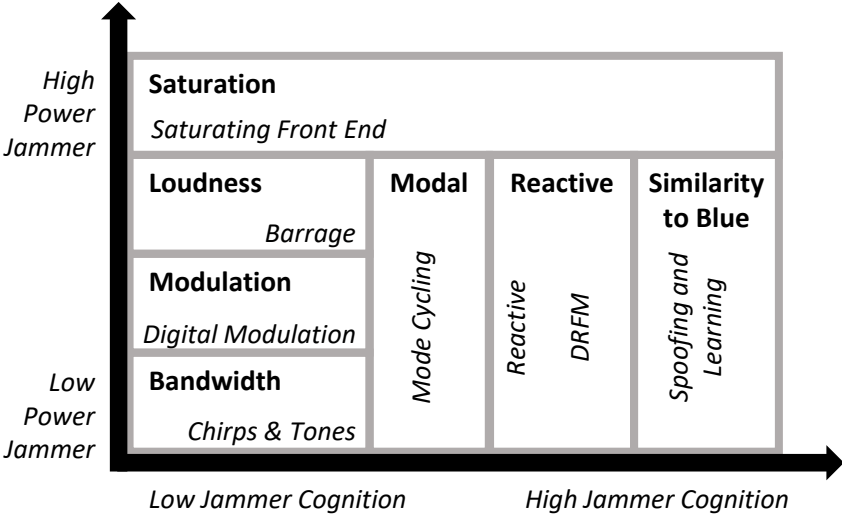


Figure 2.1 Observables are the features that allow a system to make decisions; these observables are useful to characterize different jammer types.

are inherently incomplete, and the apparent randomness of a system depends on unseen or unmeasurable variables.

All models are wrong, but some are useful.
— George Box, 1979 [5]

2.1.1 Clustering Environments

As our first foray into using ML in an EW system, we can use *clustering* to determine environment similarity. Clustering is an unsupervised ML method that groups similar items based on their features [6, 7]. We cluster the observable feature vectors into *RF environments*, and then draw a corresponding dendrogram to visualize how dis/similar each environment is to the others. An RF environment is the EMS in which EW operations take place, encompassing RF signals or reflections that can be detected, interfered with, or exploited, including the number of emitters, their locations and their activity, along with all propagation effects.

Figure 2.3 shows a dendrogram built from 23 different RF environments, and three of the corresponding boxplots. The shorter the lines connecting two

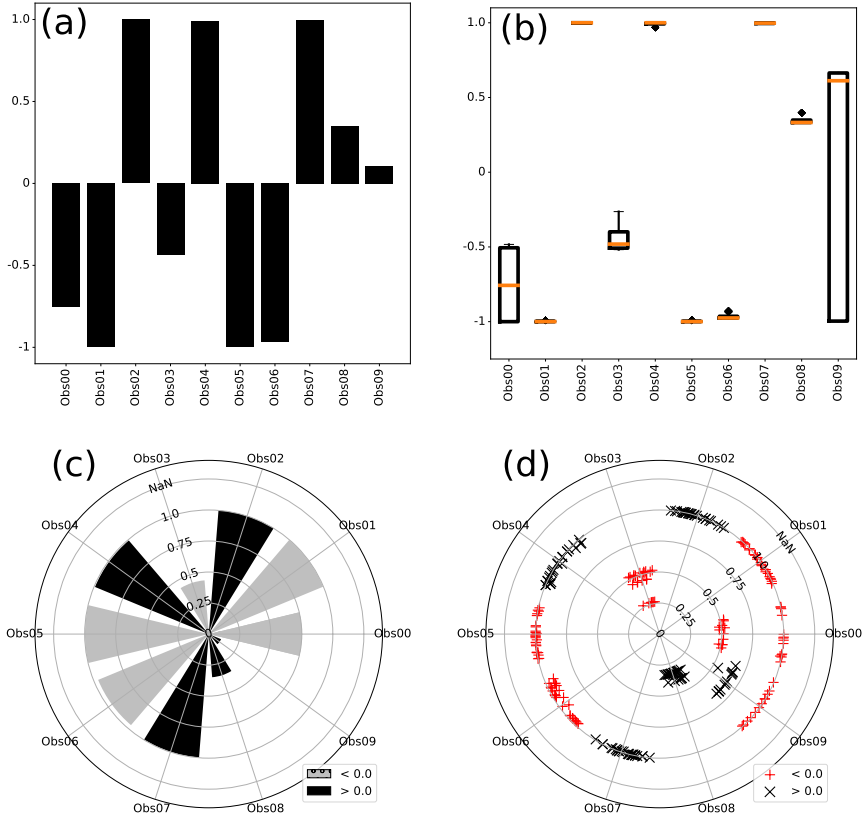


Figure 2.2 Four different ways to visualize a set of environmental observations for features such as those in Figure 2.1: (a) a bar chart that shows the mean of the non-NaN values, (b) a box-and-whiskers plot showing the distribution, (c) a bar chart plotted on a polar axis, and (d) a scatter plot of values on a polar axis. Each of these visualizations represent the same underlying observable data. (*Figure 7.4 explains boxplots.*)

environments, the more similar they are. Environments 06 and 05 are the two most-similar environments and form the first cluster; these two then become a cluster with the pair (env16 and env13). The boxplots for env16 and env06 show that they are, in fact, very similar. Env21 is very different, and is very far away in the dendrogram.

Many clustering methods are available in standard ML libraries (Section 11.4). Algorithm 2.1 shows how to plot a simple dendrogram using scikit-learn. Scikit-

Algorithm 2.1 This scikit-learn code plots a dendrogram like the one in Figure 2.3 to show the similarity of RF environments.

```
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import linkage
#####
# Load a dataset: columns are (Environ) label and multiple (obs)
def getData():
    environs = pd.read_csv( 'radardata.csv' )
    environs = environs.set_index('Environ')
    return environs
#####
# Efficient method to replace NaNs with the mu for that environ/obs
def cleanNaNs(df):
    newDF = pd.DataFrame()
    environs = np.unique( df.index )
    for environ in environs:
        # Instances for this environ
        instances = df.loc[ df.index == environ ]
        # Only replace NaN in columns with NaN values
        nanCols = instances.isnull().any(axis=0)
        for i in instances.columns[nanCols]:
            # Replace NaN in this col with mean of other vals
            instances[i].fillna( instances[i].mean(),
                               inplace=True )
        newDF = newDF.append(instances)
    return newDF
#####
if __name__ == '__main__':
    df = getData()
    df = cleanNaNs(df)
    nEnviron = len(np.unique( df.index ))
    fig,ax = plt.subplots(figsize=(12,8))
    Z = linkage(df,method='ward')
    dendrogram(Z,truncate_mode='lastp',
               p=nEnviron,orientation='left')
    fig.tight_layout()
    plt.savefig('dendrogram.pdf')
    plt.close()
```

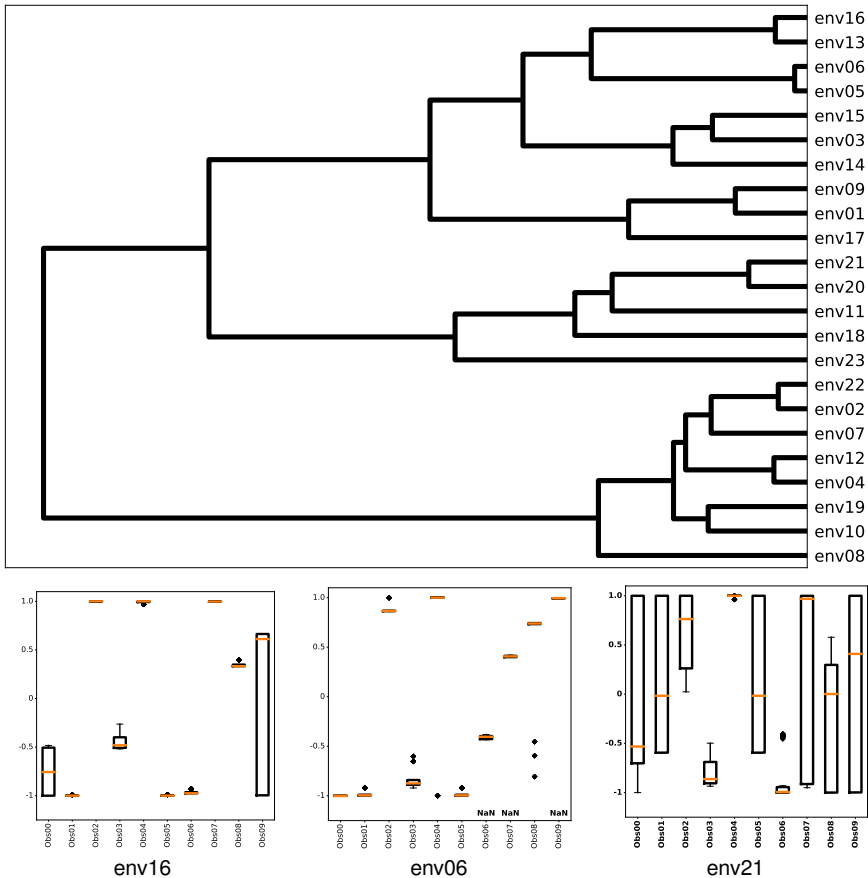


Figure 2.3 A dendrogram visualizes how similar items are to each other. Each environment is described by the multiple features shown in the boxplots. The x -axis in the dendrogram indicates the distance between environments. (*Figure 7.4 explains boxplots.*)

Learn's dendrogram functions require the data to contain no NaN values. The `cleanNaNs()` function therefore replaces NaN values by values: For each environment, for each observable, replace NaNs with the mean value of the non-NaN values of that observable in that environment.

Standard ML libraries will not utilize computations for multiple purposes and will need reimplementation for the embedded environment (Section 9.3). K-means clustering is efficient and appropriate for embedded domains [8]; known labels can be used as initial cluster seeds. If the system maintains distances between examples (e.g., to support training an SVM or maintaining diversity in the dataset) then hierarchical clustering methods are also appropriate. The final number of clusters can be determined by available memory or by intercluster distances.

2.2 Control Parameters to Change Behavior

Control parameters, or *controllables*, can include any action available to the node. A decision maker chooses controllables to accomplish desired objectives. Each node n has a set of controllables, c_n , that conform to the platform's available capabilities; these are the platform's "knobs," or degrees of freedom. There are $\overline{c_n}$ controllables at each node. A strategy s_n is a combination of control parameters, wherein each controllable has an assigned value. The maximum number of strategies per node, per timestep, is $\prod_{\forall c} v_c$, where v_c is the number of possible values for the control parameter c ; if all $\overline{c_n}$ controllables are binary on/off, then there are $2^{\overline{c_n}}$ strategies, well beyond the ability of a human to manage. For example, if there are five controllables: c_1, c_2, c_3 are binary on/off, c_4 can take 3 values, and c_5 has 10, then there are a total of $2^3 \times 3 \times 10 = 240$ different strategies. When controllables can take continuous values, or many discrete values, there may be an infinite number of strategies per node.

Control parameters are intentionally exposed by the platform for tuning; cross-layer issues are implicitly captured by selecting control parameters from multiple components. For a cognitive radio, these parameters are typically found in the protocol stack or on the radio, such as those settable through a management information base (MIB) or the mission data file (MDF).

The choice of algorithm or module allows for online reconfiguration of software and firmware flow. Modeled as a binary on/off, each available module has a control parameter x , and $x = 1$ when the module should be invoked, and $x = 0$ when the module should not operate [9].

Hierarchical control parameters are those parameters that are only valid when affiliated parameters or algorithms are enabled. For example, a proactive routing protocol requires the periodicity of *hello* messages, while a reactive protocol requires *time-to-live* thresholds.

We assume a team of heterogeneous nodes. Each node has an optimizer that can configure its capabilities, including techniques in the RF hardware, the FPGAs,

the IP stack, and even outside the RF system when appropriate, such as telling the platform to move, or requesting action from a human user. Some examples include:

- Antenna techniques such as beamforming, nulling, and sensitivity time control;
- RF front end techniques such as analog tunable filters or frequency-division multiplexing;
- Physical layer parameters such as transmit power, notch filters, or number of Fourier transform bins;
- Medium access control (MAC) layer parameters such as dynamic spectrum access, frame size, carrier sense threshold, reliability mode, unicast/broadcast, timers, contention window algorithm (e.g., linear or exponential), neighbor aggregation algorithm, dwell time, pulse repetition interval, and pulse compression length;
- Network layer or multinode coordination parameters such as neighbor discovery algorithm, thresholds, timers, number of transmitter/receiver pairs to combine in a multistatic radar, and number of receivers to jam;
- Encryption settings such as cipher and hash function;
- Application layer parameters such as compression (e.g., jpg 1 vs 10), method (e.g., audio vs video), scan pattern, or how to weight individual radar returns;
- Radar/counterradar items such as modulation type, antenna scan rate and beam pointing sequence, receiver bandwidth and dwell duration, space-time adaptive processing parameters, and EA technique(s);
- Concepts outside the RF framework, such as when interacting with a human or the platform (e.g., deploying chaff to change the electromagnetic properties of the air, or deploying a towed array to increase spatial diversity).

As an example use case, Szymanski presents a list of controllable actions that a satellite might take for self-defense [10]. These include actions in the EMS such as sensing or transmitting, in the software such as rescheduling or reassessing conditions, and physical actions such as closing shutters or maneuvering.

Controllables may have associated *costs* that must be accounted for in the metrics. Costs can include ramp-up, maintenance, or ramp-down. Costs may reflect any relevant metric, including time, power, or memory usage. Costs may depend on the value of the controllable; for example, frequent *Hello* interval timers will cost more power and bandwidth than infrequent ones. Costs in a set of controllables may be combined in different ways, for example latency-to-implement is a parallel cost, while transmit-power is summed.

Controllable values may be *mutually exclusive*. For example, a synthetic aperture radar can control integration time *or* azimuth resolution, but not both. The configuration may explicitly state this interaction, or the system can learn it.

Controllables may not always be available. Different phases of a mission, for example, may enable or disable platform capabilities (e.g., the information-gathering phase of a mission may disable actions that reveal the presence of the node, and then the prosecution phase enables them).

2.3 Metrics to Evaluate Performance

Metrics quantify how well the EW system satisfies requirements from mission and situation standpoints. There are $\overline{m}_n$ metrics at each node. A decision maker chooses controllables $c_n(t)$ to impact the metrics $m_{nk}(t' > t)$. Metrics evaluate the performance of the subcomponents under different operational conditions, and are directly optimizable, either at design time or online through self-assessment, learning, and changing behavior.

There are multiple layers of metrics, but the two that matter most are those that impact *mission success* and *EW performance*. Mission success metrics, including platform survival and probability of mission kill, are most relevant to the military decision makers and users, but may not be directly optimizable by the system. Example performance metrics include:

- *Effectiveness concepts* such as throughput, latency, sensitivity, bit-error-rate (BER), message error rate (MER), probability of detecting an event (Pd), probability of false alarms (Pfa), range, azimuth, or Doppler resolution, clutter-to-noise ratio, broad concepts of jamming effectiveness (J/S or EW BDA), and system availability. Note that jamming effectiveness is an inferred metric of adversary systems, and cannot be directly measured.
- *Cost factors* such as time, power, control overhead, timeline usage, probability of detection, risk tolerance, or even platform wear-and-tear.
- *Other concepts* such as *decision making uncertainty*, *heuristic preference*, or *flexibility to surprise*. For example, designers may prefer FPGA-based actions over software-based actions, or for the system to remain relatively stable over time (not changing actions at each time step). The *value of information* (Section 6.1.4) can also be an important metric. When the system uses ML to learn the models $f_k(o, s)$, *decision confidence* or *applicability domain* may become important metrics.¹ Also, risk can act as a cost factor in

1. The applicability domain of a statistical model is the subset of the underlying patterns on which the model is expected to give reliable predictions.

EW systems by, for example, optimizing strategies to balance time, power, and detection probability, minimizing exposure and resource use, while ensuring mission success.

Haigh et al. [1] examines four potential metrics for CRNs: (1) application-level quality-of-service requirements of multiple heterogeneous traffic flows, (2) node-level constraints on battery lifetime, (3) node-level constraints on probability of detection, and (4) cost in terms of how much bandwidth the network wastes as a by-product of application-level data transfer. Table 4.2 outlines some metrics for 5G localization. An EW system might optimize the following:

- *Low probability of detection*: Minimizes detection by adversaries
- *Low probability of intercept*: Prevents signal interception by adversaries
- *Jammer-to-signal ratio*: Disrupts enemy communications effectively
- *Response time*: Rapid threat detection and counteraction
- *Spectrum agility*: Adapts to contested environments
- *Detection accuracy*: Reduces false positives/negatives
- *Coverage*: Expands operational reach
- *Interference mitigation*: Protects blue forces from disruptions
- *Power efficiency*: Optimizes energy use

Metrics may be computed from a model, or empirically learned per Section 4.3 and Chapter 7. For example, when a communications node changes frequency, it must announce the change to its immediate neighbors, taking a minimum of one transmission; this cost can usually be computed for a given waveform. If the entire network must utilize the same frequency, the cost will depend on factors such as the network topology, medium access control protocols, and synchronization requirements; this cost should be learned empirically.

Each metric may have an associated *weight* that specifies how the metric impacts the overall performance. A scalar value could be used to accumulate a weighted sum of metrics. A lambda function could specify more flexible structures to handle metrics that are not uniform or linear. For example, if BER is above a threshold, or latency is beyond an acceptable time limit, then the metric can be zeroed. Figure 2.4 sketches some possible weight functions.

Metric values should be normalized, so that large values are evaluated at the same level as small values. Metrics may be measured in linear scale or logarithmic scale (e.g., dB). The normalization approach may differ based on scale of choice. Weights control their relative values. When maximizing utility, effectiveness metrics would have positive weights, while costs would have negative weights.

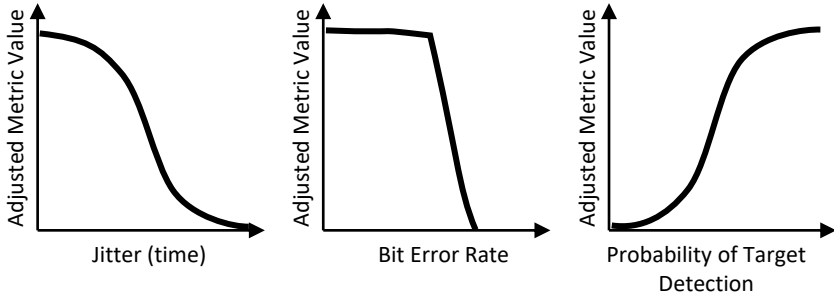


Figure 2.4 Weights can modify metric values in different ways, and moreover may change depending on the mission, for example whether the emitter is in search, acquisition, or track mode.

System requirements directly drive the process of identifying appropriate metrics in a complex system. Choosing a good set of metrics is possibly the hardest part of creating a system that effectively chooses actions autonomously. Each metric must capture the connection between available actions and system high-level requirements. The set should be as complete as possible, and include no extraneous measurements. Metrics should be relevant, measurable, responsive, and resourced [11]:

- *Relevant*: Measures of performance (MOP) and measures of effectiveness (MOE) must align with the task, desired outcomes, and commander's objectives, avoiding unnecessary data collection and redundancy.
- *Measurable*: Clear qualitative or quantitative standards are needed, with a baseline established before execution for accurate assessment.
- *Responsive*: The assessment process must quickly detect changes to enable timely adjustments and decisions, accounting for action implementation and indicator development time.
- *Resourced*: Adequate resources for data collection and analysis are essential to ensure effectiveness, prevent task duplication, and maintain operational efficiency.

Section 6.3 describes some approaches for translating high-level commander's intent to optimizable objectives.

The objectives defined in EW doctrine may need to be adjusted or elaborated to include concepts not traditionally considered. Examples include disrupting or degrading the adversary's decision-making ability, by defeating aircraft track automation or reducing the number of targets that an adversary operator can individually manage.

The most effective metrics are ones with rapid, measurable feedback. Feedback is computed by ES/BDA and reflects the observed value of the metric to compare against the value estimated by learned models. The key factors are:

- *Measurability.* Can the metric be directly measured? BER can be directly measured, whereas jamming effectiveness is inferred. Radar track quality meanwhile, is partially inferred from factors such as angle, range, velocity, and the covariance matrix estimate. Wang et al. [12] suggest an approach to inferring radar jamming effectiveness that combines over-the-air observables with engineering expertise. The system can use both types, but a directly measurable one is significantly easier to develop and leverage.
- *Locality.* How many nodes contribute to the measurement? BER can be measured on a single node; roundtrip time is measured by all of the nodes in a single flow, and jamming effectiveness could be constructed locally or over a small team of nodes. Generally, fewer nodes means that the computation is faster and more accurate. Multinode measurements can be computed with techniques such as consensus propagation [13] but also introduce latency.
- *Latency.* How rapid is the feedback? BER is relatively instantaneous, throughput has delay usually measured in seconds, target tracking accuracy is usually measured in minutes, and number of lives lost on a mission is measured in hours or days. As the feedback interval increases, it is harder for the system to connect specific actions to their effect. While in theory, the system could automatically determine these relationships, the amount of time and data required become prohibitive.

It is also useful to determine whether the metric value changes with time or other context. For example, target tracking in a surveillance radar is measured in minutes, but fire control only needs seconds to generate a firing solution. Moreover, accuracy may matter for only the initial shot selection and mid-course/end-game aspects of weapons guidance. If the EW system deceives an enemy missile guidance radar for 90% of flight out time, but did not deny the final 10% of the fly out, it still fails. Metrics must be evaluated when they are most applicable to ensure they are meaningful in the context of the overall engagement outcome in terms of success.

Some of these concerns can be handled with lambda functions; others may require changing the structure of the utility function.

There is an inherent tension between metrics and observables, in that many items could be used either way. BER, for example, may be an observable to support a low probability of intercept / low probability of detection metric, or it may be a metric that the EP system aims to minimize. The choice of whether to call a feature a metric or an observable is a function of the mission and the problem definition. When building the system, it is frequently most appropriate to start with the simplest metric (i.e., easiest to measure), while using more complex or inferred features as observables. Once the system has stabilized, one can convert features from observables to metrics. For example, the initial system may optimize BER, and subsequently augment to broader concepts of throughput and latency.

2.4 Creating a Utility Function

In systems that autonomously choose actions, a utility function captures the objectives of the stakeholders. The structure of the utility function depends on the specific tasks and capabilities of the system. The function combines the metrics in a way that is operationally meaningful and optimizable in practice.

Roth et al. [4] break down high-level utilities to the lower-level specifications that the system needs to run. Haigh et al. [1] describe policy-based methods for a human expert and an approach to turning them into an implementable, operationally meaningful objective function. A satisfactory objective function must be (1) operationally meaningful, (2) sufficiently flexible, (3) easy to modify, and (4) implementable and optimizable in practice.

Callout 2.1 presents a formal definition of a utility function, and how to simplify it so that it can be optimized in practice. Generally, a comms objective function will be more nuanced than a radar objective function because there are more metrics available. Joint optimization (ES/ EP/EA or comms/radar) adds another layer of complexity. In a complex system, requirements drive the process of choosing metrics, weights, and how to combine them. Goals and constraints come from user and mission needs, from environmental constraints, and from equipment capabilities. Haigh et al. [1] present an approach to allow in-mission manipulation of the utility function through *policy bundles*. It is important to capture goals from all stakeholders.

A simple objective function computes the weighted sum of the metrics:

$$\tilde{U}_n = \sum_{k=1}^{\overline{m}_n} (w_{nk} \times m_{nk})$$

Callout 2.1 Formal Problem Definition: the true utility function $\mathcal{U}$ must be simplified to make it optimizable in practice.

Consider an EW system with N heterogeneous nodes. Each node $n \in N$ has:

- A set of $\overline{o_n}$ observable features, denoted $\mathbf{o}_n \triangleq (o_{n1}, o_{n2}, \dots, o_{n\overline{o_n}})$.
- A set of $\overline{c_n}$ control parameters, denoted $\mathbf{c}_n \triangleq (c_{n1}, c_{n2}, \dots, c_{n\overline{c_n}})$.

Denote unobservable contextual information as $\mathbf{z}$.

To capture changes over time, denote $\mathbf{c}_n(t)$ to be the value of $\mathbf{c}_n$ at time t , and do likewise for $\mathbf{o}_n$ and $\mathbf{z}$.

Associated with the system is a real-valued scalar utility measure $\mathcal{U}(t)$ that characterizes the global, network-wide performance measure at time t . This measure is a function $\mathcal{F}$ of all the control parameters, observable features, and unobservable factors, for all nodes, since the beginning of the mission:

$$\mathcal{U}(t+1) = \mathcal{F}(\forall n \in N(\mathbf{o}_n(0), \dots, \mathbf{o}_n(t), \mathbf{c}_n(0), \dots, \mathbf{c}_n(t)), \mathbf{z}(0), \dots, \mathbf{z}(t))$$

The goal is to solve the following distributed optimization problem:

Design a fully distributed algorithm where every node n determines its control parameter values $\mathbf{c}_n(t)$ using only its own previous observable features $\mathbf{o}_n(0), \dots, \mathbf{o}_n(t)$ and control values $\mathbf{c}_n(0), \dots, \mathbf{c}_n(t-1)$ such that $\mathcal{U}(t+1)$ is maximized (or minimized) for each t .[†]

The challenge is that we cannot define an exact analytical expression for $\mathcal{F}$, due to the unobservable factors $\mathbf{z}$ and complex intra- and internode interactions. Therefore, the algorithms described in this book and in the related work generally simplify the problem by using local, memory-less approximations of $\mathcal{F}$: each node approximates the true utility $\mathcal{U}$ with $\tilde{\mathcal{U}}_n$.

$$\begin{aligned} \tilde{\mathcal{U}}_n(t+1) &= \tilde{\mathcal{F}}_n(\mathbf{o}_n(t), \mathbf{c}_n(t)) \\ \forall n \in N, \mathcal{U}(t) &\approx \tilde{\mathcal{U}}_n(t) \end{aligned}$$

Essentially, node n assumes that decisions made historically ($\mathbf{c}_n(t' < t)$), and by nearby nodes ($n' \neq n$), will be *implicitly* observable in $\mathbf{o}_n(t)$. For example, if

[†] The optimization goal can be expanded to compute the utility of the entire mission, rather than for each consecutive instantaneous time value t .

a neighbor n' increases data rate, node n will observe increased congestion. When nodes $n' \neq n$ explicitly share the previous observables or control settings with node n , i.e., $\mathbf{o}_{n' \neq n}(t' < t)$ or $\mathbf{c}_{n' \neq n}(t' < t)$, these values become additional features in $\mathbf{o}_n(t)$.

To configure the nodes, at each timestep t , each node selects a strategy $\mathbf{s}_n(t)$ that is a particular setting of controllables $\mathbf{c}_n(t)$ that optimize performance on this surface:

$$\mathbf{s}_n(t) = \operatorname{argmax}_{\mathbf{c}_n(t)} \tilde{\mathcal{F}}_n(\mathbf{o}_n(t), \mathbf{c}_n(t))$$

(Or alternatively, *argmin*.) There are $\prod_{v_{c_n}} v_{c_n}$ candidate strategies for node n at each timestep, where v_{c_n} is the number of values that a given controllable c_n can assume. This number becomes infinity when controllables are continuous-valued.

To support changing the system's behavior during the mission, we express $\tilde{U}_n$ as a set of $\overline{m}_n$ metrics, m_n , allowing a user to change corresponding weights w_n during the mission.[§] Thus for a node $n \in N$, $\tilde{U}_n(t)$ is a function of its metrics and their weights, and each metric is in turn a function of the observables and controllables at that node:

$$\begin{aligned} m_{nk}(t+1) &= f_{nk}(\mathbf{o}_n(t), \mathbf{c}_n(t)) \\ \phi_{nk}(t) &= (w_{nk}(t), m_{nk}(t)) \\ \tilde{U}_n(t) &= g_n(\phi_{n1}(t), \phi_{n2}(t), \dots, \phi_{n\overline{m}_n}(t)) \end{aligned}$$

The functions f_{nk} are models created by hand or empirically learned. The model f_{nk} at time t estimates the metric value $m_{nk}(t+1)$ from the observables $\mathbf{o}_n(t)$ and controllables $\mathbf{c}_n(t)$, allowing a decision maker on node n to select the best control settings.

Elsewhere in this book, we rarely need this level of detailed notation. We therefore use the following shorthand: $m_k = f_k(o, s)$, or simply $m = f(o, s)$. Likewise, $\tilde{U}(s_i)$ is shorthand for the utility of candidate strategy s_i .

Unlike more common notations, where $y = f(x)$, we explicitly distinguish between observables and controllables to expose the system's ability to make decisions. This notation is similar to that in reinforcement learning, where the reward is expressed as a function of state and action: $U = R(s, a)$, in Section 6.1.3.

[§] Weights do not need to be scalar values; lambda functions may be appropriate for some metrics.

where each weight w_{nk} is a scalar value, and each metric m_{nk} is a function of the node's observables and controllables, $m_{nk} = f_{nk}(\mathbf{o}_n, \mathbf{c}_n)$. When weights are lambda functions, a possible function is:

$$\tilde{U}_n = \sum_{k=1}^{\overline{m}_n} w_{nk}(m_{nk})$$

When metrics are not independent, alternate structures may be appropriate. For example, any communications flow before a deadline should be evaluated using jitter, but after the deadline, there is no benefit for delivering data. If latency $m_1 < 3$ is a hard constraint, and jitter $m_2 = 0$ is a soft constraint, then a good mathematical representation of this concept is:

$$\tilde{U}_n = w_1(3 - m_1) \cdot e^{-w_2 m_2^2}$$

where $w_1(\cdot)$ is the unit step function and $w_2 > 0$ is a parameter that characterizes how soft the jitter constraint is. Figure 2.5 shows the utility values from this function. Note that estimates for m_1 and m_2 may still depend on learned models $f_1(o, s)$ and $f_2(o, s)$, respectively.

As an example, ES systems have a goal of 100% for probability of intercept (POI). The only way to achieve 100% is to construct a staring architecture that monitors the spectrum continuously. Architectures that cannot provide continuous monitoring must implement scan patterns with sequential tuning and dwelling to intercept emitters when they are transmitting and while the receiver is dwelling at their frequency. A simplified utility function in this case is $\text{POI} = f(\text{RF environment observables, scan rate, revisit patterns, dwell, } \dots, \text{frequency})$. As with many BDA metrics, POI cannot be directly measured, and must be inferred by ES/BDA functions.

Section 5.1.1 describes different approaches for using and optimizing a multi-objective utility function. Game theory (Section 6.2) may add a layer of probabilities over the utilities to handle environments with selfish-but-rational individuals.

2.5 Utility Function Design Considerations

From an architectural design and implementation perspective, consider the following suggestions:

- Choose a single, rapid, easy-to-measure metric, such as BER, to test the decision pipeline.

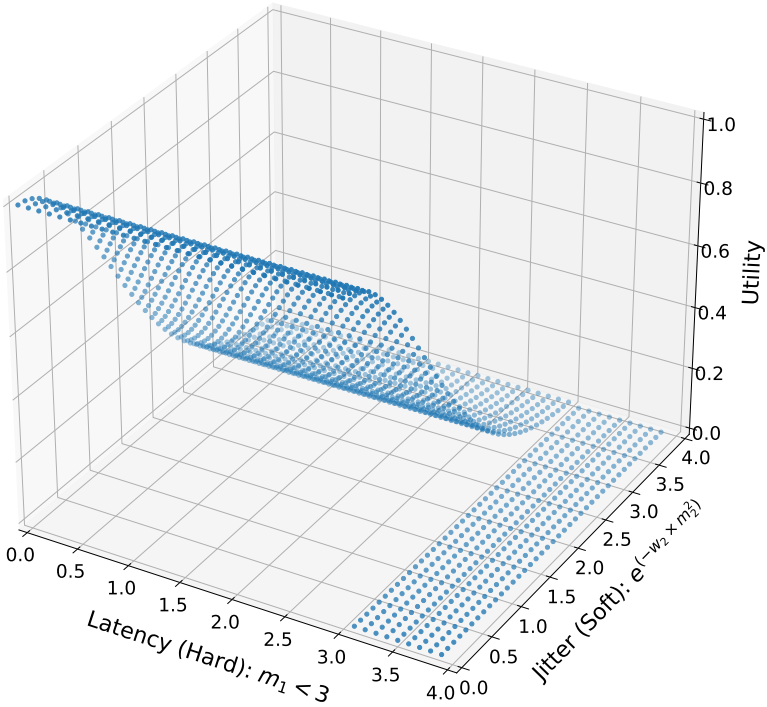


Figure 2.5 This quality-of-service metric combines the hard deadline of latency and a soft constraint of jitter [1]; here we use $w_2 = 0.3$.

- Structure the function so it is easy to add new metrics and new requirements, thus handling both planned and unplanned changes in the system.
- Keep each metric computationally independent from other metrics until the last logical moment where they need to be combined, thereby supporting rapid changes in priorities *during a mission*. Notably, this separation of concerns is critical if you use ML techniques to learn the relationship between the controllables and the metrics $m_k = f_k(o, s)$. Some systems will build a single model $\tilde{U}_n = \tilde{F}_n(o, s)$, burying the component metrics; this approach means either that the system can never be retasked, or that previous experience has to be completely restructured.

As an example, by keeping probability of detection (Pd) separate from probability of false alarms (Pfa), the system can dynamically choose its

operating point along the receiver operating characteristic (ROC) curve as described in Section 5.1.1.

Section 4.3 describes approaches to learning metrics from empirical data, while Chapter 7 describes the process of learning *during* a mission.

- Because different nodes might have different capabilities and requirements, different nodes (or types of nodes) may have different utility functions. There is no need for the utility function to be identical on every node: Each node may have its own unique function. Utility across a heterogeneous set of nodes is additionally complicated because of intermittent connectivity and dynamic network membership. The true utility function $\mathcal{U}$ takes into account a fully heterogeneous team, wherein each node has different capabilities and tasks. The approximate utility function $\tilde{U}_n$ captures utility for a specific node n , and may use different metrics, weights, observables, and controllables from any other node.
- Identify mechanisms for the nodes to share information efficiently when a metric is an inherently multinode concept. Consensus propagation is an effective approach that minimizes measurement latency, and does not require the system to know how many nodes are in the network [1, 13]. Each node n computes its local contribution to the shared metric, incorporating estimates from other nodes n' when available. Section 6.1.4 discusses knowledge sharing.

2.6 Example Observables, Controllables, and Metrics

Two easy metrics to start an initial exploration into decision optimization are BER or link stability. Figure 2.1 lists some example observables and Table 5.1 lists some example controllables for optimizing BER. Table 2.2 lists some example observables and controllables for link stability. Table 2.3 describes a possible set of observables, controllables, and metrics for a CRN. CRNs generally perform communications EP, and optimize data integrity, communication quality and efficiency. Newman outlines some of the dependencies among them, namely which observables and controllables impact which metrics [14]. Ramzan et al. present a utility function combining five of these metrics [15]. Table 5.1 outlines several techniques to mitigate a tone jammer.

Table 2.4 describes a possible set of observables, controllables, and metrics for a radar EA system. Charlish et al. present a variety of objective functions for cognitive radar, including task-based, information-theoretic, and quality-of-service [16]. Guerci presents several optimization examples, including signal-to-interference-plus-noise, signal-to-clutter, and target identification [17]. Radar EA

Table 2.2
Example Observables and Controllables to Optimize Comms Link Stability

Observables	Controllables
BER (moving average; hysteresis)	Implement or deactivate retransmission
Packet error rate (moving average; hysteresis)	Modify waveform modulation
Link disruptions (time of most recent; hysteresis)	Modify network coding parameters
Correlation of link disruptions to context (e.g., geography, EW activity)	Implement or deactivate network coding
	Change the physical node position

Table 2.3
Example Observables, Controllables, and Metrics for a CRN

Observables	Controllables	Metrics
Received signal power	Transmission power	Minimize BER
Noise power density	Modulation scheme	Minimize packet error rate
Residual energy in the batteries	and symbol rate	Minimize frame error rate
Spectral occupancy and information	Coding scheme	Maximize throughput
Consumed power	Duplexing scheme	Minimize power consumption
Interference power	Bandwidth	Minimize interference
Channel state information	Carrier frequency	Maximize spectral efficiency
Power delay profile	Frame size	
Doppler effects	Beamforming parameters	
Primary user detection metrics		

aims to hide mission-critical assets by disrupting or misleading a target radar’s operation. The jammer uses system-level control parameters such as antenna gain and symbol rate to support tactical-level actions like sidelobe jamming, range- and velocity-gate pull-off, and spoofing. If engineering analysis can identify factors like the target radar’s system information such as anti-jamming features or detection and tracking algorithms, the radar EA system can also use these features to specialize an attack. Additionally, while some operational modes may be inferred directly based on the target radar’s emissions, the full range of modes may be difficult to determine. Wang et al. present a complete cognitive radar EA system [12].

Figure 2.6 sketches how observables, controllables, and metrics might come together in a library. Each emitter or RF environment is captured with a set of observables. Each available technique is a controllable; combinations of techniques are strategies. Each environment/technique pair has a set of properties, including

Table 2.4
Example Observables, Controllables, and Metrics for a Radar EA System

Observables	Controllables	Metrics
Frequency	Carrier frequency	Maximize target concealment
Pulse repetition frequency	Modulation scheme	Maximize own-system stealth
Signal strength	Waveform characteristics	Minimize tracking accuracy
Antenna beamwidth and pattern	(e.g., pulse width, pulse repetition interval)	Maximize range and velocity ambiguity
Modulation scheme	Pulse repetition frequency	Maximize false target generation
Doppler effects	Beamforming parameters	Minimize effective detection range
Waveform characteristics	Duplexing scheme	Minimize radar's time-to-engage
Radar sensitivity and noise	Transmission power	Minimize situation awareness
Scan type and rate	Antenna gain	Maximize resource efficiency
Environment and clutter effects	Bandwidth	
Operational mode	Frame size	
	Coding scheme	
	Symbol rate	

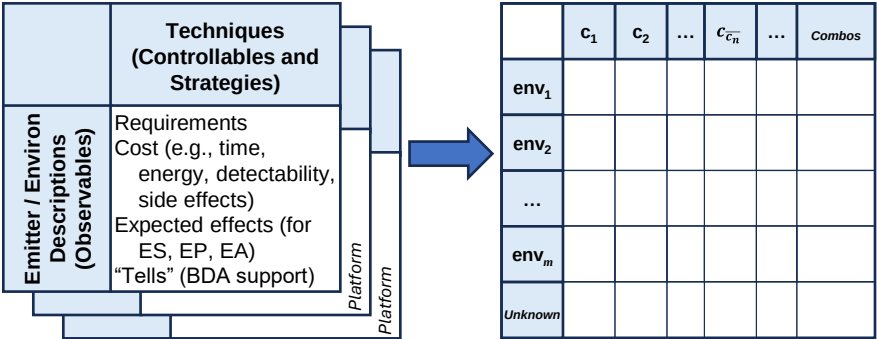


Figure 2.6 A template library captures the interactions between environments, techniques, and effects. The library can be learned and used by decision makers operationally. (Table 7.1 shows a similar table that tracks what the decision maker knows.)

the requirements to use the technique, expected costs, expected effects, and a set of expected “tells” that can be used by BDA. An initial template can be created by system designers when they create new techniques, or when analysts characterize new threats. Operationally, decision makers (human or AI) can use the templates to choose appropriate strategies to accomplish desired objectives, and BDA methods can use the templates to evaluate effectiveness of recent actions. Learning,

meanwhile, can update the templates: more detailed or accurate environment descriptions; better understanding of requirements, cost, expected effects, or tells; or performance for different platforms or when components are replaced. Learning can also create new templates when the system encounters new environments, characterize performance for combinations of techniques, and transfer learned knowledge across environments, techniques, or platforms.

2.7 Conclusion

The goal of a fully capable, effective EW system rests on a foundation of a utility function that measures system performance, so that it can select good strategies for the mission. The utility function captures user objectives, mission goals, and environmental constraints in a mathematically optimizable approach.

Chapter 4 describes ES and the observable features upon which to base decisions; the structure of the objective function supports observables that are raw values, derived from physics models, or inferred by learned models. Chapters 5 and 6 show how to use the objective function for decision making; namely how to choose strategies for EP and EA using optimization and scheduling techniques, and the longer-term EBM decision-making horizon using planning approaches.

References

- [1] Haigh, K. Z., O. Olofinboba, and C. Y. Tang, "Designing an Implementable User-Oriented Objective Function for MANETs," in *Networking, Sensing and Control*. IEEE, 2007. Online: <http://www.cs.cmu.edu/~khaigh/papers/Haigh07-ICNSC.pdf>.
- [2] Jouini, W., C. Moy, and J. Palicot, "Decision Making for Cognitive Radio Equipment: Analysis of the First 10 Years of Exploration," *EURASIP Journal on Wireless Communications and Networking*, No. 26, 2012. DOI: 10.1186/1687-1499-2012-26.
- [3] Arrow, K., "Decision Theory and Operations Research," *Operations Research*, Vol. 5, No. 6, 1957. DOI: 10.1287/opre.5.6.765.
- [4] Roth, E., et al., "Designing Collaborative Planning Systems: Putting Joint Cognitive Systems Principles to Practice," in *Cognitive Systems Engineering: A Future for a Changing World*, Ashgate Publishing, 2017, Ch. 14. Online: <https://tinyurl.com/collab-planning-2017>.
- [5] Box, G., "Robustness in the Strategy of Scientific Model Building," in *Robustness in Statistics*, Academic Press, 1979. DOI: 10.1016/B978-0-12-438150-6.50018-2.
- [6] Rokach, L., and O. Maimon, "Clustering Methods," in *Data Mining and Knowledge Discovery Handbook*, New York, NY: Springer, 2005. DOI: 10.1007/0-387-25465-X_15.
- [7] Bair, E., "Semi-Supervised Clustering Methods," *Wiley interdisciplinary reviews. Computational statistics*, Vol. 5, No. 5, 2013. DOI: 10.1002/wics.1270.
- [8] MacQueen, J., "Some Methods for Classification and Analysis of Multivariate Observations," in *Berkeley Symposium on Mathematical Statistics and Probability*, 1967. Online: https://projecteuclid.org/download/pdf_1/euclid.bsm/1200512992.

-
- [9] Haigh, K. Z., S. Varadarajan, and C. Y. Tang, "Automatic Learning-based MANET Cross-Layer Parameter Configuration," in *Workshop on Wireless Ad Hoc and Sensor Networks*. IEEE, 2006. Online: <https://www.cs.cmu.edu/~khaigh/papers/haigh06a-configuration.pdf>.
 - [10] Szymanski, P., *Automated Satellite Self-Defense Courses of Action*, 2024. Online: <https://tinyurl.com/szymanski2024>. Accessed 2024-12-28.
 - [11] US Army Headquarters, "Targeting," Tech. Rep. ATP 3-60, 2015. Online: <https://fas.org/irp/doddir/army/atp3-60.pdf>.
 - [12] Wang, S., et al., *Principle and Technology of Cognitive Electronic Warfare*. Beijing, China: National Defense Industry Press, 2018.
 - [13] Moallemi, C., and B. Van Roy, "Consensus Propagation," *Transactions on Information Theory*, Vol. 52, No. 11, 2006. Online: <https://arxiv.org/abs/cs/0603078>.
 - [14] Newman, T., "Multiple Objective Fitness Functions for Cognitive Radio Adaptation," Ph.D. dissertation, University of Kansas, Lawrence, KS, 2008. Online: <https://core.ac.uk/reader/213386062>.
 - [15] Ramzan, M., et al., "Multi-Objective Optimization for Spectrum Sharing in Cognitive Radio Networks: A Review," *Pervasive and Mobile Computing*, Vol. 41, 2017. DOI: 10.1016/j.pmcj.2017.07.010.
 - [16] Charlish, A., et al., "Stochastic Control for Cognitive Radar," in, Mishra, K., B. Shankar, and M. Rangaswamy, (Eds.), Raleigh, NC: SciTech Publishing, 2023.
 - [17] Guerri, J., *Cognitive Radar: The Knowledge-Aided Fully Adaptive Approach*, 2nd Edition. Norwood, MA: Artech House, 2020.

3

Machine Learning Primer

AI is a field within computer science that draws from mathematics, engineering, and neuroscience. Section 1.3 describes the four pillars of AI: knowledge management, situation assessment, decision making, and ML. Despite the media's attempt to equate ML with AI, ML is just one of AI's many subfields, artificial neural networks (ANNs) are one of many families of techniques within ML, and deep learning is simply a type of ANN.

“AI” is not synonymous with “Deep Learning.”
 $AI \supset ML \supset ANNs \supset Deep\ Learning$

ML is broadly relevant throughout the EW system. CogEW techniques rely on ML to model the spectrum, understand the participants, and learn how to effectively plan and optimize. In-mission learning is an essential part of a complete cognitive EW system; without this capability, the system will never handle novel emitters.

A learning algorithm uses empirical data to learn a model of the space. ML is a type of curve fitting or function approximation. Trend lines, as used by engineers and financial analysts alike, are a simple method for estimating performance and future values. ML models predict values for all input values, *even if they were not present in the training data*. If the training data had one data point, there are an infinite number of trend lines. With two data points, the trend is linear. With three or more samples, the trend is a curve. In general, more training data yields a more accurate trend. The goal of an ML model is to minimize the error between the true function and the estimated curve.

Learning algorithms are commonly grouped into *supervised* and *unsupervised* approaches, where the algorithm is given the ground truth labels Y for training, or not, respectively. A supervised learner creates a function $f : X \rightarrow Y$ from an input space X to an output space Y , using a finite number of instances $(X, f(X))$. The model f approximates the true output space $\mathcal{F}$. An unsupervised learner uses only X to build the model that captures hidden patterns in the unlabeled data. For example, a supervised modulation classifier would label every observation of the RF environment with the true transmitted modulation. An unsupervised

Table 3.1
Common Applications for ML Algorithms

Application	Description	EW Example
Clustering	Grouping similar instances together (unsupervised)	Section 2.1.1
Classification	Assigning a label from a set of discrete classes (supervised)	Section 4.2.3
Regression	Estimating a numerical value (supervised)	Section 4.3
Outlier Detection	Identifying instances that are very different from typical examples (unsupervised)	Section 4.5

modulation clustering algorithm would group similar observations without knowing their labels, and the output Y is a value corresponding to the cluster number for each observation. *Semi-supervised* learning approaches have a small number of labeled examples, and many unlabeled ones. In *reinforcement learning (RL)* or *self-supervised* approaches, the learner gathers the labels itself by taking actions in the environment. RL forms the basis of in-mission learning (Chapter 7).

Table 3.1 lists some of the common uses of ML; this book presents many other examples for EW. Section 3.9 presents some of the trade-offs that an EW engineer should consider when choosing an ML algorithm. Section 11.4 lists ML toolkits useful for rapid prototyping; supporting datasets are described in Section 11.5.

ML techniques offer several key advantages over the traditional function approximation approaches, including:

- Scalability to very large datasets,
- Adaptability to new data,
- Automated feature extraction,
- Automatic modeling of feature interactions (including nonlinear relationships),
- Versatility to different data types, including numerical, categorical, textual, and visual,
- Better handling of missing features and unstructured data,
- Regularization and overfitting control,
- Knowledge transfer via transfer learning, and
- Robustness to noise.

Traditional approaches such as signal processing or statistical models are often grounded in well-established mathematical or physical principles. Therefore

they may be more interpretable, have better sample efficiency, and be able to provide theoretical performance guarantees.

Many of these methods can also offer *closed-form solutions*, which are precise formulae that provide exact solutions without iterative approximation. For example, linear regression and Fourier transforms often have closed-form solutions, making them computationally efficient and straightforward to implement. This ensures accurate, interpretable results when assumptions are satisfied. However, these approaches can be limited when faced with complex, high-dimensional, or nonlinear problems, where closed-form solutions are often impractical or unavailable. In such cases, ML methods, which often rely on iterative optimization instead of closed-form solutions, provide greater flexibility and scalability. Section 11.3 discusses some of the considerations of when to use AI or traditional approaches.

Purely data-driven ML models can sometimes lack transparency and efficiency. Hybrid approaches (Section 3.3) bridge this gap by integrating symbolic knowledge, such as human-readable rules or domain-specific heuristics, into ML models. This combination can improve feature construction, reduce search complexity, and enhance interpretability, allowing models to “bootstrap” the learning process and leverage domain expertise while retaining the adaptability of ML.

Lastly, when choosing an ML model, it’s important to consider its trustworthiness. The ability for a model to estimate its own confidence is a key factor. Section 10.10 discusses verification methods to assure the system, including for example methods for a model to assess its own uncertainty.

3.1 Common ML Algorithms

There are many ML algorithms; perhaps 100 are widely recognized across different tasks. For a deeper analysis, refer to *The Hundred-Page Machine Learning Book* [1], or even deeper to *The Elements of Statistical Learning* [2]. Alternatively, Vink and de Haan [3] have short descriptions of ML algorithms in the context of target recognition, and Kulin et al. [4] presents some of the math around ML for spectrum learning. Kuzdeba [5] provides an in-depth exploration of ML in the RF domain, with a strong focus on deep learning.

Instance-based methods store all training instances in memory, and the inference phase compares new instances against all training instances. There is no training time, but inference time is a function of the size of the training data. Table lookup, hashing, and nearest neighbor methods are common instance-based approaches [6]. If an EW system automatically adds new threats to the library, then the traditional EW activity of “looking up the threat in the library” is an instance-based learning method.

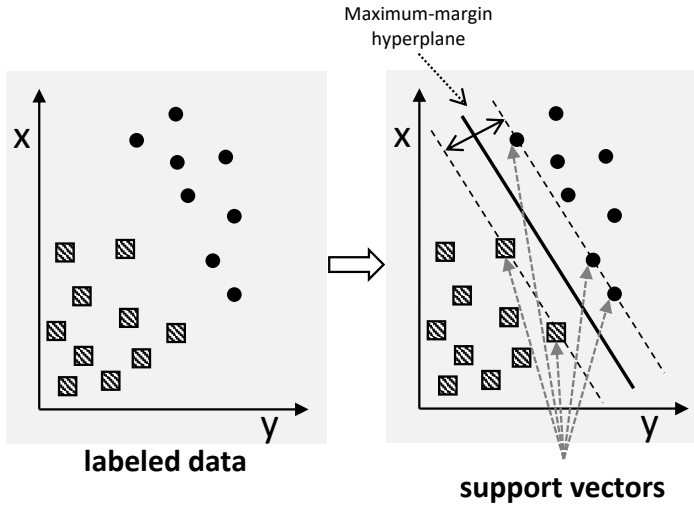


Figure 3.1 SVMs efficiently learn complex functions with limited data.

Model-based methods use the training data to create a model that has parameters; the model size (number of parameters) does not change with the amount of data. Training time is a function of the size of the training data, while inference time is constant. SVMs (Section 3.1.1) and ANNs (Section 3.1.2) are two that have particular relevance for EW. Ensemble methods increase robustness by combining the predictions from multiple models (Section 3.1.3). Many other ML models exist, applicable for many different settings (Section 3.1.4).

3.1.1 Support Vector Machines (SVMs)

SVMs are a class of machine learning methods used for clustering, classification, regression, and outlier detection [7–9]. SVMs have high generalization performance, particularly for complex nonlinear relationships and when the available training data is limited.

SVMs use a subset of training data points (called support vectors) in the decision function of the SVM algorithm. Each SVM constructs a decision function that estimates the decision boundary, known as the *maximum-margin hyperplane*, and an acceptable hard or soft margin around that hyperplane. The support vectors are the samples on the margin, illustrated in Figure 3.1.

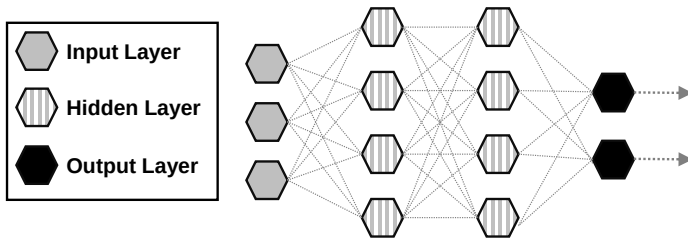


Figure 3.2 ANNs consist of input, hidden, and output layers. The number of layers refers to the depth of the network.

SVMs proficiently accomplish nonlinear classification by using kernel functions or “tricks” [10]. With a kernel trick, nonlinear classifiers map their inputs into high-dimensional feature spaces.

3.1.2 Artificial Neural Networks

ANNs are a class of algorithms inspired by the way biological nervous systems, such as the brain, process information. These algorithms simulate neurons and their interconnections to replicate how the brain learns. ANNs have their conceptual roots in the 1940s [11], and in 1958, Rosenblatt designed the *perceptron*, an algorithm for pattern recognition [12]. Werbos’s 1975 *backpropagation* algorithm enabled practical training of multilayer networks: It distributes the error term back up through the layers by modifying the weights at each node [13].

ANNs consist of input, hidden, and output layers, illustrated in Figure 3.2. The number of layers refers to the *depth* of the network. Modern ANNs typically have many layers, and are thus referred to as *deep learning*, *deep neural networks*, or simply *DeepNets*. DeepNets can discover patterns or *latent features* that are too complex for a human to extract or design. DeepNets excel at modeling complex, nonlinear relationships, but can also incorporate traditional features in a hybrid fashion, such as those from signal processing, when needed.

Deep learning has achieved remarkable breakthroughs in natural language and image processing, largely due to three factors: bigger computers, labeled data, and insights in how to connect the networks. Many DeepNet architectures exist, with new ones emerging regularly. The most comprehensive references are by Bengio, Hinton, and LeCun, who won the Turing award for their work in 2018 [14, 15]. Common architectures include:

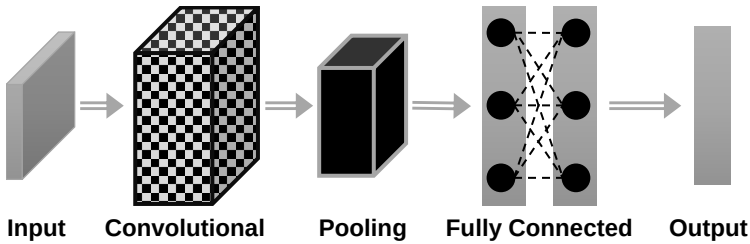


Figure 3.3 CNNs process data with a known grid-like topology.

- *Convolutional neural networks (CNNs)* [16, 17] are ANNs for processing data with a known grid-like topology. A typical CNN consists of three types of layers: convolutional, pooling, and fully connected, illustrated in Figure 3.3. *Convolutional layers* allow the network to learn filters which respond to local regions of the input. *Pooling layers* streamline data processing: They reduce the outputs of neuron clusters at one layer into a single neuron in the next layer. *Fully connected layers* connect every neuron in one layer to every neuron in the next layer.
- *Recurrent neural networks (RNNs)* [18] belong to a family of ANNs for processing sequential data [15]. RNNs have feedback connections (i.e., memory). They address the temporal nature of their input data by maintaining internal states that have memory [19]. RNNs are valuable in contexts where their sequential structure and memory mechanism are beneficial, particularly in resource-limited environments, tasks that require step-by-step processing, or long-term dependencies [20].
- *Temporal CNNs* handle sequence-related, time-related, and memory-related deep learning [21], and outperform RNNs in many applications [22]. Temporal CNNs compute convolutions in parallel, leading to faster training and inference times; they can capture hierarchical representations, and effectively capture short- to medium-term dependencies.
- *Autoencoders* learn efficient data encodings, building a model with a bottleneck layer; the bottleneck is the efficient encoding, and the output layer is the reconstruction of the input, as illustrated in Figure 3.4. The autoencoder tries to generate, from the reduced encoding, a representation as close as possible to its original input. They effectively learn to eliminate noise (i.e., they learn the common case), and are often used for dimensionality reduction

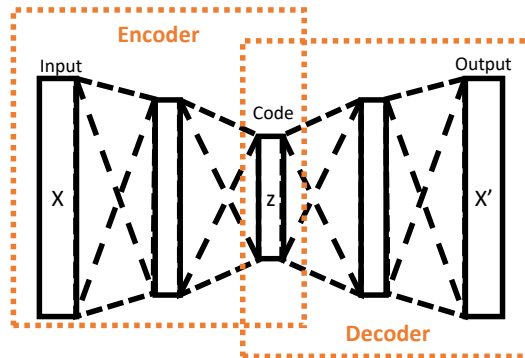


Figure 3.4 In an autoencoder the bottleneck layer learns a compressed encoding, and the output layer reconstructs the input.

and anomaly detection [15, 23]. Additive autoencoders, for example, can reduce data dimensions by combining bias, linear trend, and nonlinear residual estimation [24]. Alnaseri et al. [25] highlight the potential of deep learning autoencoders for next-generation communication systems, noting challenges like data requirements, overfitting, and computational complexity.

- *Siamese neural nets* use the same weights on multiple networks, while training each on different input data. They require less training data than other methods [26].
- *Kohonen networks*, also known as *self-organizing maps*, produce low-dimensional representations of the input space, and are used for dimensionality reduction and visualization [27].
- *Generative adversarial networks (GANs)* are a system of two competing neural networks each trying to improve the accuracy of its predictions [28]. Illustrated in Figure 3.5, a GAN sets a *generative network* against a *discriminative network* in a competition: The generative network's goal is to fool the discriminative network. GANs are often used to create synthetic data (Section 8.3.3).
- *Spiking neural networks (SNNs)*, inspired by biological neurons, communicate through discrete spikes rather than continuous signals. Gerstner and Kistler [29] established the foundational framework for spiking neuron models, while Pfeiffer and Pfeil [30] examined the integration of deep learning

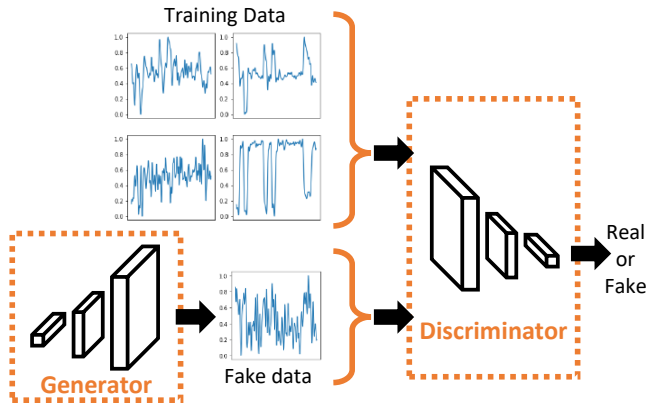


Figure 3.5 In GANs, a generative network competes with a discriminative network, aiming to produce synthetic data that can fool the discriminator.

with SNNs. SNNs are energy efficient due to their event-driven sparse calculations [31, 32]. SNNs show promise in integrated sensing and communications [33] and adaptable neuromorphic architectures like *reconfigurable architecture for neuromorphic computing (RANC)* [34]. Chen et al. [35] provide a comprehensive overview of ML applications in wireless communications, discussing various neural network architectures, including SNNs, and their potential roles in next-generation wireless networks.

- *Large language models (LLMs)* utilize *transformer* architecture to process and generate text, excelling in tasks like translation, summarization, and content creation [36]. While they offer significant advantages across various industries, their use requires substantial computational resources. Section 3.6 describes the use of LLMs in generative AI.

Popular ANN architectures are available online (e.g., [37–39]), and many have been presented in the context of RF [5, 35, 40].

3.1.3 Ensemble Methods

Using multiple classifiers in an *ensemble* [6, 41–46] can improve prediction accuracy and increase robustness to adversarial attacks because an ensemble combines the predictions from multiple diverse models. Election predictions, for example, combine the results from many different polls, with the hope that flaws in one poll

Table 3.2
Commonly Used ML Algorithms and Associated Applications

Algorithm	Classif.	Regr.	Clustering	Anomaly Detection	Dimen. Reduction	Time Series
ANNs	✓	✓	✓	✓	✓	✓
SVMs	✓	✓		✓		✓
GMMs			✓	✓	✓	✓
HMMs	✓			✓		✓
Bagging	✓	✓				✓
Boosting	✓	✓				
Random forest	✓	✓				✓
Decision trees	✓	✓				
kNN	✓		✓			
kMeans			✓		✓	
Naïve Bayes	✓					
Logistic regr.		✓				
Gaussian process		✓				
Hierarchical clustering			✓			
DBScan			✓			
Isolation forest				✓		
PCA					✓	
AutoRegression						✓
Kalman filters						✓

will compensate for flaws in other models. Section 7.1.1 shows a simple example for an EW BDA task.

Common ensemble approaches include majority voting, bagging, boosting, and Bayesian model averaging; most ML toolkits offer additional techniques. Simple *majority voting* chooses the most common prediction [47]. *Bagging* gives each model in the ensemble equal weight [47, 48]. *Boosting* incrementally builds the ensemble by training each new model to emphasize training instances that previous models misclassified [49]. *Bayesian model averaging* uses weights based on the posterior probabilities of each model [50, 51].

3.1.4 Other ML Algorithms

Table 3.2 lists a few of the more commonly used techniques, and the applications they tend to be used for. ML libraries such as MATLAB and Python include these techniques, along with many others; documentation describes when and where they are applicable. Since the list in Table 3.2 is extensive, we will highlight only a few algorithms. For example, *Gaussian mixture models (GMMs)* are probabilistic models that represent data as a mixture of multiple Gaussian distributions, making

them effective for clustering and density estimation [52]. Similarly, *hidden Markov models (HMMs)* are sequence-based models that infer hidden states from observed data, widely applied in speech recognition and time-series analysis [53]. *k-nearest neighbor (kNNs)* is an instance-based ML technique that assigns labels based on the majority class or value of the closest k datapoints, and therefore effectively learns from k or fewer instances [54]. Meanwhile, *density-based spatial clustering of applications with noise (DBScan)* is a clustering algorithm that groups data points based on density, allowing it to identify arbitrarily shaped clusters and detect outliers [55]. *Principal component analysis (PCA)* is a dimensionality reduction technique that transforms data into uncorrelated principal components, commonly used for feature extraction and noise reduction [56–58].

Some ML methods are naturally suited to working with limited data. They either inherently build models from limited data, leverage prior domain knowledge, or transfer insights from well-understood areas of the domain to less-familiar ones. Models that *inherently* work with limited data include algorithms like k NNs and SVMs (Section 3.1.1), which classify new instances based on their similarity to known examples. The BBN SO (Example 1.1) uses SVMs because they can efficiently learn complex functions from limited training sets. Gaussian processes and randomized search techniques like genetic algorithms (Section 5.1.2) handle limited data by relying on probabilistic modeling and iterative optimization. Decision trees and case-based or analogical reasoning further support learning in data-scarce environments by structuring decisions around known cases. Methods incorporating *prior domain-knowledge*, such as Bayesian methods and neurosymbolic AI (Section 3.3), enhance predictions by integrating prior probabilities or structured reasoning. Finally, *knowledge transfer* techniques, such as few-shot learning (Section 3.4), enable models to generalize from only a handful of examples by leveraging meta-learning or transfer learning strategies.

3.2 Generalization to Surprise

Generalization refers to a model's ability to adapt properly to new, previously unseen data, that is not drawn from the same distribution as the one used to create the model. A table-lookup, for example, has no ability to generalize. A good model will use the available empirical data to approximate the true data, and will be robust to unexpected inputs.

Overfitting means that the model captures the training data too well, and therefore is unlikely to do well with new data, while *underfitting* doesn't even capture the training data well (Figure 3.6). Overfitting is a result of excessive model complexity, trying to fit the noise or anomalies, too little training data, or

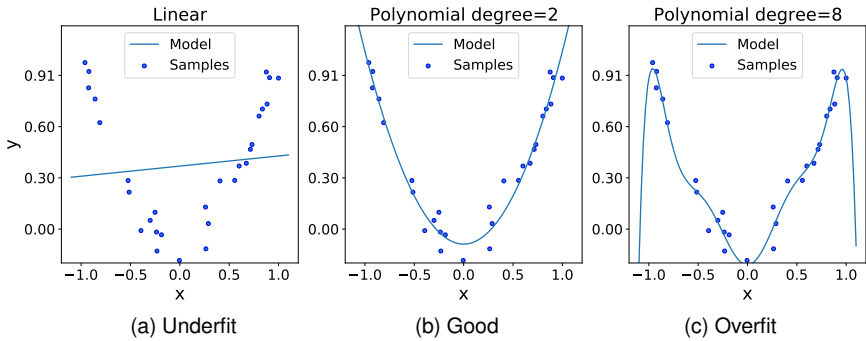


Figure 3.6 Linear models often underfit, and models tend to overfit if there are many more features than training instances. (*Underlying data is $y = x^2 + \epsilon$.*)

over-training that leads to memorizing the environment. Underfitting, meanwhile, might occur when there aren't enough features or data to capture the pattern, or the model makes strong assumptions that limit its flexibility (e.g., using a linear model for nonlinear data). Good generalization means finding the right trade-off between under- and overfitting the data.

The *domain-of-expertise* indicates what the model should be expected to capture well: When a new observation falls within the domain-of-expertise, the model should do better than when it falls outside. In Figure 3.6, for example, the training data falls in the range $[-1.0, 1.0]$. A value of 2.0 is outside the domain-of-expertise. The two-degree polynomial generalizes well outside this range, but the other two models generate increasing error the further away from the expected range. Ideally, the model should compute its own confidence.

Over- and underfitting are controlled by tuning *hyperparameters* that control how the algorithm works. Each algorithm has its own hyperparameters, for example decision trees tune *MaxDepth* and *MinSamples*, while SVMs tune C and γ (cost for misclassified points, and influence of single instances, respectively), and DeepNets might use early stopping and activation dropout. If using one of the standard ML toolkits (Section 11.4), hyperparameters are usually the arguments to the function.

Section 10.7 describes an experimental approach that fine-tunes hyperparameters of a model by adaptively selecting new data points. Other approaches to improve generalization include ensemble methods (Section 3.1.3), hybrid models (Section 3.3), calibrating models [59], and training approaches like batchnorm [60,

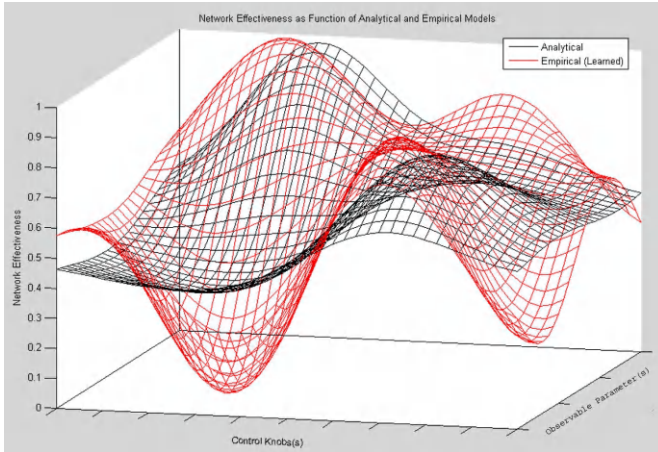


Figure 3.7 Analytical models estimate the performance surface as a function of $\overline{o}_n$ dimensions of observables and $\overline{c}_n$ dimensions of controllables; empirical models refine the prediction. (From [66].)

61], Langevin methods [62], or entropy filtering [63]. Managing the data with diversity, augmentation, and forgetting are also important steps (Section 8.3). Section 3.5 talks more about the concept of metalearning.

3.3 Hybrid ML

Symbolic AI approaches manipulate symbols (often human-readable), while *non-symbolic* approaches operate on raw data. Decision trees are frequently symbolic in nature, while DeepNets are usually nonsymbolic. In recent years, *hybrid* approaches combine the two; they use symbolic knowledge to construct features, reduce the search space, improve search efficiency, and explain resulting models. Hybrid approaches exploit domain-specific knowledge and heuristics to find solutions more quickly [64–74]. Hybrid approaches are also known as *knowledge-based AI*, *neuro-symbolic AI*, or *scientific ML*.

Hybrid approaches essentially bootstrap or constrain the learning process. Figure 3.7 illustrates the idea: An analytical model provides an initial framework or educated guess about the result, and the empirical data refines the prediction to match observed experience. For example, the free-space path loss model (Callout 3.1) captures the loss for a perfect environment. A hybrid learner can use $1/R^2$

Callout 3.1 The inverse square law of electromagnetic propagation is a good starting point for rapidly learning real-world path loss.

The inverse square law of electromagnetic propagation is a good physics-based model. It accounts for the loss of signal strength over space (called free-space path loss), where the amount of energy received at distance R diminishes as $1/R^2$. In a perfect environment, the path loss is only affected by the transmission frequency and the distance R between the transmitter and receiver. The power density W at the wavefront is inversely proportional to R^2 :

$$W = \frac{P_t}{4\pi R^2}$$

where P_t is the transmitted signal power. Since radar relies on the reflection of a transmitted signal, radar systems suffer path loss that is proportional to R^4 rather than the R^2 for the one-way propagation of a communication system [75].

In reality, the environment is not perfect and path loss is impacted by additional factors, including absorption, diffraction, and reflection. Other variables include the environment, terrain, medium of propagation, and antenna properties.

as the initial model, and then use empirical data to refine path loss estimates to the current operating environment.

The hybrid approach enables the learner to work adequately even with no training data, and work well after real-world training. Figure 3.8 sketches how DeepNets, classic ML, and hybrid approaches perform as a function of available data. Classical feature-based ML models perform well with little or no data; $1/R^2$ could be one of the features. DeepNets require a lot of data before they perform well, but will eventually do better than a classical ML model. Combining both approaches in a hybrid model has been shown, in all domains, to outperform either approach individually.

Hybrid models that incorporate symbolic deduction, attention, and context outperform pure DeepNet models, while reducing computation costs and memory footprint [76]. Neurosymbolic approaches enhance interpretability, robustness, and trustworthiness of DeepNets, while learning from much less data [77, 78]. Hybrid models reason over knowledge represented symbolically, allowing them to generalize from fewer examples. Neurosymbolic methods can explain their decisions in the context and language of experts in the domain. Neurosymbolic approaches are also more resistant to errors because they must conform to the rules of the domain; for example, they cannot learn things that are scientifically impossible [79]. Traditional methods are often grounded in well-established mathematical or physics principles,

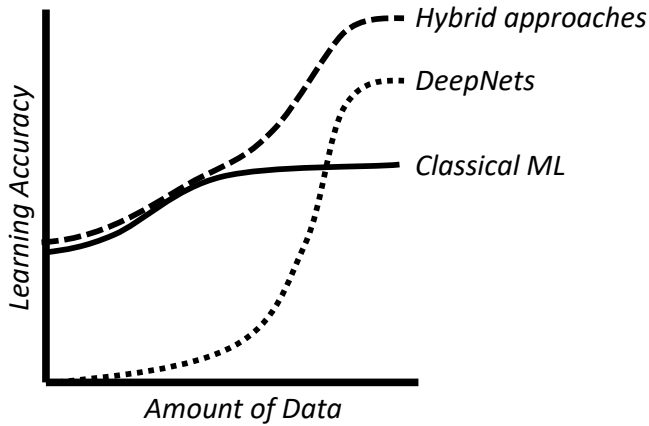


Figure 3.8 DeepNets identify latent features in the data, whereas classical ML approaches rely on traditional feature engineering. Hybrid approaches leverage the benefits of both.

and therefore have better sample efficiency while also providing theoretical performance guarantees.

Combining empirical approaches with domain knowledge improves diagnostic accuracy, improves airline safety, enhances online shopping, and monitors regulatory compliance [80]. A climate model that incorporates conventional atmospheric physics has demonstrated superior performance in long-range weather and climate forecasting, offering faster and more accurate predictions while being less resource-intensive than traditional methods [81]; their model is *five orders of magnitude* faster than other approaches. Examples of hybrid approaches in this book include:

- Section 4.2.1 presents an cartoon example of how to combine traditional features with DeepNet models. This approach is possibly the easiest and most intuitive way to combine approaches.
- Section 4.2.2 describes competition results for modulation recognition.
- Section 5.1.2 describes hybrid approaches for optimization.
- Section 6.3 leverages human expertise in model design.
- Section 7.1.1 presents several hybrid approaches for BDA using formal analytics and over-the-air observables.
- Section 7.3.3.1 presents a hybrid approach for rapid in-mission-learning.

There are many approaches for combining domain expertise, science, and empirical data. These include changing the internal structure of the model, reducing

the state space or creating heuristics, and feature construction [66]. van Bekkum et al. [82] present common design patterns for hybrid ML, and Moseley [78] describes common approaches for physics-inspired ANNs. Moseley incorporates the scientific constraints into the architecture of the ML model, uses the physics to minimize residual error, or uses empirical data to correct the outputs of a traditional model (similar to Figure 3.7). Work in hybrid approaches is exploding, as evidenced by recent reports and new journals on the topic [72, 76, 83].

3.4 Open-Set Classification

Open-set classification techniques create new classes on the fly, after the initial model is trained. They handle unknown data encountered during the mission, and that was not anticipated during the training phase. One option is to choose an ML model that effectively learns from one example; k NN and SVMs are good choices (Section 3.1.4). Data augmentation (Section 8.3.3) can also help ensure that the original training data covers more of the unknown classes.

Scheirer et al. [84] explore the challenges of open-set recognition, where models must identify both known and unknown classes. They formalize the problem as a constrained minimization task and introduce a measure called *openness*, which quantifies the difficulty of an open-set problem based on the ratio of known to unknown classes. This paper also proposes solutions to improve recognition performance in open-set scenarios. Geng et al. [85] provide a comprehensive survey of open-set recognition techniques, including thresholding, distance-based classification, and generative models, emphasizing the need to manage uncertainty in novel inputs.

Deep learning methods do not perform well with limited or out-of-distribution data (outside the domain-of-expertise) [86, 87]. As a result, a lot of research has focused on approaches to addressing this challenge, including autoencoders, zero- or low-shot learning [88–93] and anomaly detection (Section 4.5). Low-shot learning, for example, uses the training data to construct a *latent embedding* of the important features, and then uses this embedding to create new classes on the fly. Metric learning meanwhile learns similarity functions between well-represented classes and limited-data classes [94], and memory-augmented neural nets store learned patterns for rapid adaptation [95].

A different perspective is offered by Gkanatsios et al. [96], who introduce *analogical networks* for 3D scene segmentation. Their method retrieves and composes structures from memory, enabling models to generalize to unseen object categories without requiring retraining. Apfeld and Charlish present an approach that leverages hierarchy in the structure [97]. Meanwhile, Sanh et al. [98] propose

multitask-prompted training as a means to facilitate zero-shot generalization. Their approach aligns with open-set classification by equipping models to handle novel data distributions through fine-tuning on diverse tasks. Kumar et al. [99] present a *metalearning* framework designed for class-incremental learning. Their method mitigates catastrophic forgetting while dynamically integrating new classes, drawing parallels to pseudoclass strategies that generate synthetic labels to improve adaptability in open-set scenarios.

A related concept is that of *self-supervised learning (SSL)*, which generates supervisory labels from the data itself [100, 101] by predicting missing parts of the input or contrasting similar vs dissimilar pairs, and thereby learns to predict the underlying structure or properties of the data. These features are not specialized for solving a particular supervised task, but rather encapsulate richer statistics for downstream tasks. In EW applications, Lin et al. [102] apply transfer learning and decision fusion to radar waveform classification, achieving over 80% accuracy on unknown signals by leveraging CNN-extracted features and fusing time-frequency representations. Metric learning has been used to identify radar signals [103] and automatically identify pulse repetition intervals [104]. Huang et al. [105] use SSL and domain adaptation to pretrain masked autoencoders on signals from various RF domains; the learned representations can be transferred to domains with limited annotated data. Milososheski et al. [106] use SSL to extract features and monitor spectrum activity; their result dramatically reduces the number of features and the size of the models, while outperforming supervised approaches.

In summary, open-set classification enables models to dynamically recognize and incorporate new classes when encountering unseen data. Some models, such as k NN and SVMs, rapidly adapt to new classes, while data augmentation enhances robustness by expanding coverage of potential unknowns in evolving environments. Adaptive learning strategies—including transfer learning, structural analogy, and multitask learning—further improve classification in open-set scenarios. Generative models and anomaly detection help distinguish unknown inputs, while low-shot learning improves classification with minimal data. Additionally, pseudoclass methods simulate novel categories during training, strengthening recognition of unseen data. Finally, metalearning facilitates continuous class integration while addressing catastrophic forgetting, supporting the adaptability of open-set models.

3.5 Metalearning

Metalearning is the process of learning-to-learn [107–112]. Metalearning is an awareness and understanding of the phenomenon of learning, rather than learning the subject knowledge. A metalearner changes the operational parameters of an

algorithm so that its expected performance increases with experience. There is some terminology awkwardness because *learning* involves creating rules to assess or decide more effectively, while *metalearning* is learning-to-learn; many learning algorithms are optimizers, therefore metalearning can learn to optimize a learner that improves an optimizer.

Metalearning learns how to exploit structure in the problems of interest in an automatic way, so that they no longer need to be tuned by hand. Metalearning offers a variety of benefits:

- *Speed*: learn or optimize faster,
- *Ease of use*: automate the process of tuning algorithms,
- *Reduced data*: use available data more efficiently,
- *Solution quality*: learn models that generalize better to surprise, and
- *Adaptability*: agile to environmental changes and transfer knowledge from one context another.

Early research in metalearning looked at bias-management to select between different algorithms [113], meta-genetic programming to evolve a better genetic program [114], and learning to optimize learning rules in neural networks [115]. For newer work, the book *Metalearning* is a good overview of metalearning across many application areas [116]. The University of Amsterdam developed a tutorial on metalearning [117], focused on few-shot learning for image recognition. Metalearning can improve performance of all capabilities in a system, including:

- *Data management*: Metalearning can transform features [118, 119], select features [120], reduce dimensionality of the task [121, 122], detect concept drift [123, 124], improve the data labeling process [125–127], infer values for missing data [128, 129], perform data augmentation [130–132] (see also Section 8.3.3), and select relevant training examples (Section 8.3.2).
- *ML techniques*: Metalearning can improve model selection [133, 134], tune hyperparameters [135], adapt architectures dynamically [114, 136], and optimize training strategies. The survey *Domain Generalization through Meta-Learning* [87] addresses how DeepNets struggle with out-of-distribution data (outside the domain-of-expertise) and explores metalearning as a solution. The paper introduces a taxonomy of generalization methods and a decision graph for model selection, outlining strategies for enhancing DeepNet robustness.

- *Decision making:* Metalearning can prioritize the search for strategies, optimize hyperparameters [135, 137–139], choose the optimization algorithm [113, 140], adapt the search gradient [141], and refine decision heuristics based on past performance.
- *RL techniques:* Metalearning can learn reward functions [142, 143], guide exploration [144–146] (see also Section 7.3.2), identify effective reward shaping techniques [147–149], optimize hyperparameters dynamically during training [150], and facilitate transfer learning by recognizing similarities across tasks and adapting policies accordingly [151].
- *Evaluation:* Metalearning can determine model sensitivity to hyperparameter changes [152, 153], generate adversarial scenarios [154, 155], and learn from past evaluations to optimize test coverage (Section 10.7).

As a case example, metalearning for optimization can help prioritize the search for strategies, starting with the most important controllable (i.e., those that contribute most to solution quality). The algorithm searches over options for a small handful of important controllable, while using default values or the values selected in the previous timestep, for the less important ones. The algorithm searches for less important controllable only after fixing these important values. An example in communication jamming would be the use of an appropriate duty cycle for causing interference as a major controllable, with minor controllable being the details of a jamming waveform like frequency offsets and particular modulation techniques. Metalearning can also analyze the search process to optimize hyperparameters and search more efficiently [135, 137–139]. A metalearner optimizes a set of hyperparameters θ that control the optimization of $\tilde{U}_n$. The metalearner uses a meta-Utility function $\mathcal{J}$, where $\mathcal{J}$ evaluates qualities of the optimizer, such as the speed to convergence, as illustrated in Figure 3.9.

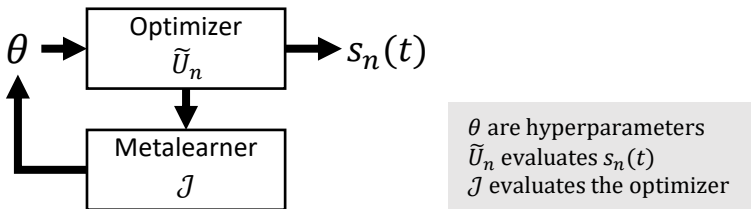


Figure 3.9 Metalearning optimizes the hyperparameters θ of an optimizer using a metaUtility function $\mathcal{J}$.

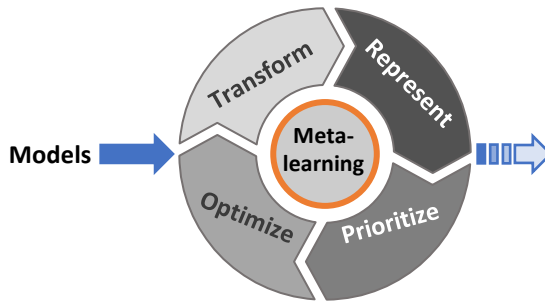


Figure 3.10 Metalearning changes the representation of the data, prioritizes search, adjusts hyperparameters, and transforms the utility function to improve solution quality.

Metalearning approaches can be combined, as illustrated in Figure 3.10, with the overall purpose of improving the generalization of ML models and the efficiency and accuracy of decision-making engines.

3.6 Generative AI

Generative AI (GenAI) is a branch of AI involving algorithms designed to create new synthetic data or outputs, such as text, images, audio, and videos, by analyzing and learning patterns from large datasets [156, 157]. It includes techniques such as GANs, variational autoencoders (VAEs), and transformers, which are widely used across various fields due to their ability to generate new data.

GANs operate by utilizing a generator to create data and a discriminator to refine it (Section 3.1.2), making them ideal for data-generation tasks such as image and video generation. Beyond these applications, research has leveraged GANs for adaptive radar waveform design, where the generator creates waveforms that blend into background noise while remaining effective for sensing [158].

VAEs, on the other hand, encode and decode data to generate structured yet diverse outputs, widely applied in creative fields like art and design. Kingma & Welling [159] formally introduced VAEs as a framework that merges probabilistic graphical models with deep learning. This approach enables the generation of diverse and coherent outputs by learning latent representations in a probabilistic manner. This work also introduced the reparameterization trick, a technique that incorporates randomness into the model without affecting its differentiability, thus enabling optimization of the model's parameters. Although VAEs are often highlighted in the context of image generation, their applications span a wide range of AI tasks, from anomaly detection to the generation of new drug molecules [160].

Transformers understand and process sequence-based tasks. Transformers, particularly in *large language models (LLMs)*, learn from vast datasets to generate coherent text and speech, which may prove effective in signal generation and processing for EW tasks. Models like *generative pretrained transformer (GPT)* leverage attention mechanisms to understand context and produce meaningful text. The transformer architecture revolutionized natural language processing by utilizing attention mechanisms to enhance context understanding [161].

Despite their potential, LLMs face challenges such as inherent biases from training data, significant computational requirements, and limited interpretability of their outputs. These challenges extend more broadly to GenAI as a whole, though their severity varies depending on the model type and application. While biases are a concern for most GenAI models trained on real-world data, text-based models often exhibit them more prominently than domain-specific generators [162].

Concerns include computational demands, limited interpretability, inherent biases, discriminatory outcomes, and privacy risks in GenAI models and applications [163–165]. Computational demands can range from the immense resource requirements of large-scale diffusion models to more lightweight generative techniques [166]. Interpretability remains a common issue, particularly in deep-learning-based generators, though some structured generative approaches offer greater transparency [167].

GenAI has many practical tactical and strategic applications in CogEW. GenAI can improve EW capabilities by generating adversarial signal patterns to train defense systems, improving their ability to detect and counter emerging threats (Section 8.3.3). In anti-jamming applications, generative models can produce adaptive interference signals that evade conventional defenses, ensuring EW systems remain resilient against evolving countermeasures. Additionally, GenAI can support real-time decision-making by simulating potential scenarios and providing operators with clear, actionable strategies.

In adaptive jamming, GANs can simulate adversarial signals to disrupt enemy communications and radar systems [168–170]. Transformers can generate synthetic radar and communication signals to test system performance in real-time scenarios. Transformers can also support beamforming to optimize communication performance [171], or perform radar signal deinterleaving [172].

Anomaly detection is another important application. VAEs can model normal signal patterns and identify deviations, such as spoofing or interference, which can help maintain the integrity of communication and radar systems [173, 174]. Similarly, spectrum management can leverage LLMs to dynamically allocate resources

and minimize interference, enhancing efficiency in complex and contested environments [175, 176].

3.7 Embedded ML

TinyML, also known as *embedded AI*, *edge AI*, or *on-device ML*, focuses on running ML models on small, resource-constrained devices such as microcontrollers and low-power embedded systems [177]. In environments such as IoT, autonomous systems, and the tactical edge, devices often have limited power, memory, and processing capacity, making ML implementation more challenging. The book *TinyML* [178] explores the deployment of ML models on ultra-low-power microcontrollers, enabling AI-driven applications on edge devices with minimal computational resources.

In CogEW applications, tinyML enables on-device data processing and decision-making, reducing reliance on cloud resources. TinyML capability allows for faster, more secure, and adaptive responses to dynamic RF environments. By reducing latency, improving data privacy, and lowering energy consumption, tinyML can be used for tasks such as signal detection, jamming resistance [179], and spectrum monitoring.

TinyML, while often associated with deep learning, can be applied to a diverse range of ML models, enhancing their efficiency for deployment on resource-constrained devices. It employs popular compression techniques such as pruning, quantization, knowledge distillation, decomposition, mixture-of-experts, and lightweight model design, which are just as valuable for deep learning as they are for models like decision trees, SVMs, and ensemble methods [5, 180–183]. Some of the most commonly used methods include:

- *Pruning*: Reduces the size of a trained model by eliminating unnecessary parameters, connections, or structures, while preserving its performance. This process is akin to removing noncritical parts of a model, such as layers or neurons, to decrease complexity. Pruning is particularly beneficial for over-parameterized networks, as it enhances efficiency by discarding redundant components, much like weeding a garden, eliminating unwanted growth to allow the flowers to thrive.
- *Quantization*: Reduces the precision of numerical representations in deep neural networks while preserving performance. By representing weights and activations with fewer bits, quantization lowers memory usage, computational costs, and model size, enabling faster inference, for example 1-bit LLMs [184].

- *Knowledge distillation*: Focuses on transferring insights from a large, complex teacher model to a smaller, more efficient student model. Knowledge distillation is also relevant for robust verification of ML models (Section 10.10) and privacy preservation (Section 8.3.5).
- *Decomposition*: Simplifies parts of a model to achieve sufficient performance while reducing complexity.
- *Mixture-of-experts*: Uses multiple expert submodels, but only a few are activated per inference step based on an intelligent gating mechanism.
- *Lightweight model design*: creates models optimized to operate efficiently within a fixed computational budget while adhering to device constraints. The BBN SO (Example 1.1) is an example of this approach.

Pruning and distillation demonstrate both significant memory and compute improvements, while maintaining accuracy [185]. DeepSeek, the model that shook the world in January 2025, used distillation, quantization, and mixture-of-experts to create high-performing models at a fraction of the cost of larger models. DeepSeek also eliminates the value function by computing the rewards over a group of samples, and therefore needs to only train one network [186].

TinyML will play an important role in CogEW by enabling real-time adaptability and decision-making. For example, lightweight models such as MobileGPT [187] can support on-device generation of operational insights from sensor data. Similarly, embedded GANs [188], used in real-time image processing, can synthesize deceptive signals or simulate adversarial RF environments for enhanced resilience. SSL can further aid EW by analyzing spectrum scans to detect anomalies and classify patterns with minimal labeled data. For example, Chen et al. [189] propose a self-supervised GAN approach where the discriminator learns an additional task, such as detecting image rotation, alongside distinguishing real from fake data. This approach improves model stability and performance using unlabeled data.

In the same vein, the Sub-Linear Deep Learning Engine (SLIDE) [190, 191] enhances tinyML's potential by enabling more efficient DeepNet training, achieving over 3.5x speedup compared to Tesla V100 (2019) and a further 7x improvement with CPU optimizations (2021). Its hash-based sparsity and CPU efficiency approaches allow low-power deep learning on resource-constrained edge devices, making it particularly well-suited for CogEW applications.

Efficient inference techniques are also being developed for edge devices [192–194]. Combining these tinyML techniques will support rapid, energy-efficient, real-time ML on edge devices.

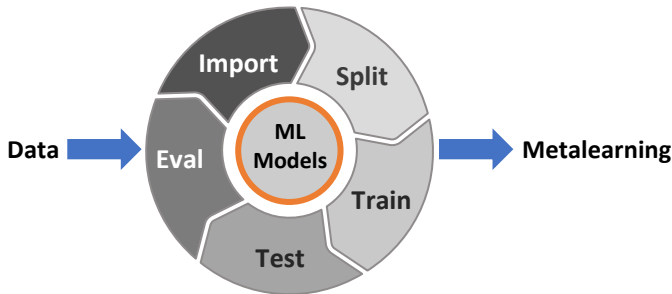


Figure 3.11 Every ML model is trained and tested with the same underlying process.

3.8 Training an ML Model

Figure 3.11 indicates the core steps for building an ML model. Every ML model is trained and tested with the same basic iterative process. Always experiment with multiple data formats (and features), and types of models, to determine which is most effective for the problem: accuracy, data, goals, and constraints (Section 3.9).

The first step in training an ML model is a diverse and representative dataset. High-quality data enhances model performance, while biases, noise, or imbalances can lead to poor generalization. Effective training data should capture a wide range of scenarios, including typical cases, anomalies, and variations, to improve robustness. Preprocessing techniques such as cleaning, normalization, and augmentation are often necessary to make the data suitable for learning. Chapter 8 covers data collection, management, and preprocessing methods, explaining their role in model training. For example datasets, refer to Chapter 11, where we explore various data sources and discuss the initial steps in implementing an ML algorithm.

Figure 3.12 illustrates where these steps occur during design and operation in a radar classification task. The design phase creates the data formats, develops the model, and evaluates its performance. Algorithm 4.1 shows an example of code underlying this loop. When the model meets requirements, it is used to classify new radars during operation. If there is an in-mission learning step (the dotted line), the learning process must also be validated using the iterative process of Section 10.2.2.

In domains where novel situations are the common case, this train/test process is inadequate. Most AI literature describes a test process wherein the majority of the data is used for training the model, and the remainder for evaluating it. In EW, however, this process is insufficient. EW is a domain where novel classes are common, and the system must learn from new examples using only a few samples.

Table 3.3

Questions to Address When Choosing an Algorithmic Solution to Fit an ML Task

Factor	Questions to consider
Task	Are you trying to understand the environment? Predict what will happen? Control actions? Adapt continuously? Scott Page, in <i>The Model Thinker</i> [195], presents the uses of models (REDCAPE): • <i>Reason</i>: To identify conditions and deduce logical implications. • <i>Explain</i>: To provide (testable) explanations for empirical phenomena. • <i>Design</i>: To choose features of institutions, policies, and rules. • <i>Communicate</i>: To relate knowledge and understandings. • <i>Act</i>: To guide policy choices and strategic actions. • <i>Predict</i>: To make numerical and categorical predictions of future and unknown phenomena. • <i>Explore</i>: To investigate possibilities and hypotheticals.
Data	How much training data is available (pre-mission and in-mission)? How well labeled is the data? Is it partially observable (i.e., can features be missing)? Is the data numeric or categorical? What features can be computed by a priori models? Has data changed since training (e.g., concept drift, component updates)?
Goals	How accurate must the solution be? How fast must the system arrive at conclusions? Do false positives and false negatives impact performance differently? Are confidence levels appropriate or necessary as annotations on decisions? Does the solution need to be interpretable or explainable to an EW operator? Does the solution need to be scalable to more emitters, more environments, or more tasks? What security/privacy considerations matter?
Constraints	What hard real-time requirements must be satisfied? What computation is available (CPU, GPU, FPGA, custom ASIC)? How much storage is available, for the model and for the data? How much of the data must be persistent over very long durations (e.g., mission-to-mission)?

As one example, DeepNets effectively identify latent features in the data. DeepNets, however, depend on a large set of diverse well-labeled training data [197]. While there might be a lot of data available for cellular or WiFi networks, EW domains generally do not have this advantage. Figure 3.8 illustrates the performance trade-off in which DeepNets perform better when there is a lot of data, while classical ML approaches do better when data is limited, in large part because features are crafted for the problem. Section 4.2.1 describes how DeepNets and SVMs might be combined effectively in a deployed system, so that DeepNets compute general features of the RF domain, while SVMs perform the in-mission model updates.

In EW missions where the mission expects to encounter novel emitters during the mission, system design must incorporate techniques for learning novel models, from one or two examples, at mission-relevant timescales. In Section 3.1.4, we highlight ML models that work well on limited data. Section 3.4 discusses open-set classification methods that enable models to recognize and adapt to new classes when encountering previously unseen data. Data augmentation and adversarial training approaches can often compensate for the lack of data (Section 8.3.3). Data augmentation increases the diversity of data, without actually collecting new data.

Table 3.4 summarizes a few of the design trade-offs for common ML algorithms. *This list is not comprehensive*, but simply illustrates the need to analyze the task, data, goals, and constraints of the EW system before settling on one particular approach. Some general rules of thumb include [1]:

- When predicting the future, larger complex models tend to capture the performance surface more accurately. When trying to explain, smaller more concise models are better.
- When the number of training instances is limited, feature engineering can be very helpful, and classical ML approaches are more likely to be useful.
- When the number of training instances is large, and there is ample time for training, DeepNets can be effective.
- When inference-time is limited, model-based techniques are better than instance-based techniques. Model-based techniques may suffer from long training times, but inference is fast. Instance-based techniques do not have a training phase, instead incurring computation time at inference.

Lim et al. [198] evaluate 33 algorithms using metrics of training time and accuracy. A longer list of metrics appears in Callout 10.2. Table 3.5 lists the computational complexity of several ML algorithms. For example, k NNs have no retraining time because all the computation happens at inference. CNNs, meanwhile, are extremely computationally expensive for both training and inference. Fuscone highlights computational resources for several algorithms [199]. Computational learning theory analyzes complexity and other formal mathematical concepts around ML [200].

Requirements for in-mission learning adds additional considerations to system design. Section 10.3 discusses learning goals, ensuring that the chosen models perform well even when pretrained on little or no data. Figure 11.4 shows how the performance of a static regression model trained on a lot of data may not be a good indicator of the performance of the same model if it must learn in-mission; in-mission performance must be separately evaluated. Section 10.10 discusses verification approaches for learning systems.

Table 3.4
Design Trade-Offs for Common ML Algorithms

ML Algorithm	Common Uses	Advantages	Disadvantages
SVMs	Stock market; RF quality; radar emitter signal recognition; modulation recognition; anomalies	Excellent accuracy on little data; extremely efficient; good for many features	Primarily numerical data
DeepNets	Image and text understanding; signal characteristics; modulation recognition; target tracking; anomalies; SEI	Extracts latent features in the data; inference can be very fast	Requires a lot of training data; computationally intensive; hard to explain
Logistic regression	Risk; process failure	Efficient; interpretable; computes feature relevance	Requires clean data; one-dimensional output
Naïve Bayes classifier	Sentiment analysis; document categorization; spam filtering	Categorical input data; relatively little training data; probability of outcomes	Requires conditional independence
Decision trees	Design decisions; sentiment analysis	Easy to explain; very strong with discrete data	Fragile with many features; weaker with numerical data
Causal models	Relationships	Patterns when experimentation is not possible	Hard to control for unobservable features

3.10 The Speed of AI

AI is developing at a rate never seen for any previous technology. Some estimates suggest that AI (not just ML) is doubling every two months. Consider the following examples:

- In 2017, the paper *Attention is All You Need* [201] took the natural language processing world by storm; only a few months after the paper was released, it had replaced almost all previous approaches. At the time, it was the fastest-ever transition from research to production; today, the pace is even higher.

Table 3.5
Computational Complexity of Some ML Algorithms

Algorithm	Training	Inference
Logistic regression	$O(nf)$	$O(f)$
k NN	$O(1)$ k NN is a lazy learner	$O(k \log n)$
Decision trees	$O(nf \log f)$	$O(\log f)$
Naïve Bayes	$O(nf)$	$O(f)$
SVMs	$O(n^2 f)$ to $O(n^3 f)$ Complexity depends on kernel	$O(mf)$ $m \leq n$ is the number of support vectors
Random forest	$O(tun \log n)$ t is number of trees $u \leq f$ is number of features considered for splitting	$O(t \log n)$
CNNs	$O(nmdhwk)$ m is number of layers d is depth of layers h and w are height and width of input data k is number of filters in each layer	$O(mdhw)$

n is number of training instances; f is number of features.

- When OpenAI’s text-to-image tool *Dall-E* was released January 2021, there was skepticism that AI would ever be able to generate video from text. It was only 11 months when Microsoft’s NÜWA model came out.
- The release of stability.ai’s *Stable Diffusion* image generator in November 2022 is the first significant technological open-source disclosure allowing users complete flexibility over the capability [202].

The pace of development and free sense of exploration from users is proceeding at such a dizzying speed that it’s difficult to see very far ahead.

— Martin Anderson, 2022 [202]

- The 2022 *AI Index* [203] shows that AI publications doubled during the preceding 10 years, AI patents grew 77% per year since 2015, training times dropped 27x, and cost dropped 223x.

- Andrew Ng’s course *Generative AI for Everyone* enrolled 43,000 people in the first seven days when it was released November 2023 [204].
- The annual *State of AI Report* summarizes the major research advances; in 2024 these include hybrid models, DeepNets on edge devices, synthetic data generation, AI for biotech, and foundational models [76].
- The *Abstraction and Reasoning Corpus* (ARC) for Artificial General Intelligence (AGI) evaluates AI on benchmark tasks that require human-like fluid intelligence [205]. ARC is designed to measure adaptability to novel tasks. In June 2020, OpenAI’s GPT-3 achieved a 0% score on ARC: it could not solve any ARC’s benchmark problems. In May 2024, OpenAI’s GPT-4o achieved a 5% score on ARC-AGI puzzles. By December 2024—less than 7 months later—OpenAI’s o3 model achieved a 75.7% score.
- *Cursor.AI* is an AI-powered code editor that allows almost anyone to become a developer. They reached US\$100M revenue in just 12 months (calendar year 2024), making it among the fastest-growing products in history. Only a few products, such as COVID-19 vaccines and the iPhone, have been faster.
- *DeepSeek* (released 10 January 2025) shocked the world with its efficiency. Before DeepSeek could fade out of headlines, *Qwen2.5-Max* from Alibaba (released 29 January) and *Tulu 3* from AI2 (released 30 January) outperformed it for accuracy, memory footprint, and computational efficiency.

Nothing highlights the speed of AI development than the progress that China has made in the last 15 years. Until approximately 2010, China was not a contender in the global AI research community. China caught up with the US in terms of AI research in the mid-2010s [206]. In 2017, China released its *New Generation Artificial Intelligence Development Plan* [207, 208], aiming to become the global leader in AI by 2030. By 2020, China excelled in areas like facial recognition, speech recognition, and computer vision, often surpassing NATO countries in the practical application of these technologies, and by 2021, China was producing the largest volume of high-quality global AI research [209].

According to the Australian Strategic Policy Institute’s *Critical Technology Tracker* [210], China is now the lead country in 57 of 64 key technologies, including AI, EW, space, robotics, advanced materials, and quantum; several of these are highlighted in Table 3.6. For some of these technologies, all of the world’s top-10 leading research institutions are in China. As of 2023, ASPI’s analysis of global research in EW for the previous 5 years estimates that China is producing 51% of high-impact research; the US is in second place with only 12%. China overtook

Table 3.6
Technologies from ASPI's *Critical Technology Tracker* [210] Related to CogEW

Technology	Lead Country	#Top 10 Orgs [†]	% Global Publications [‡]	Monopoly Risk
Electronic warfare	China	10	51.5%	High
Radar	China	10	42.7%	High
Undersea wireless comms	China	9	51.5%	High
Optical communications	China	9	41.0%	High
Directed energy	China	7	43.7%	Medium
RF communications	China	6	31.9%	Medium
Quantum communication	China	6	33.6%	Low
Mesh networks	China	6	29.2%	Low
ML	China	9	36.5%	Medium
Data analytics	China	9	33.2%	Medium
AI algorithms and HW accelerators	China	6	30.9%	Medium
Adversarial AI	China	7	31.1%	Low
Autonomous underwater vehicles	China	10	66.8%	High
Drones and swarming robots	China	8	38.4%	High
Robotics	China	7	34.5%	Low
Autonomous systems operation	China	7	34.3%	Low
Hypersonic detection and tracking	China	10	72.9%	High
Multi- and hyperspectral imaging	China	9	53.7%	High
Novel metamaterials	China	9	51.7%	High

[†] #Top 10 Orgs: Lead country's share of the world's top-10 research institutions.
[‡] % Global publications: For publication impact, ASPI uses the top 10% most-cited papers from the Web of Science Core Collection database, which reduces the influence of low-impact journals. They do not incorporate Chinese-language publications, which is now the primary venue for Chinese AI research [208], but may inflate counts from self-citation.

the combined research outputs of NATO countries in 2018. ASPI concludes that China's lead is the product of deliberate design and long-term policy planning.

In a humorous nod to the tidal wave of change, Togelius and Yannakais present a list of survival strategies for depressed AI academics [211]. The first citation in this paper is to Rick Astley's song *Never Gonna Give You Up*.

Kai-Fu Lee, a visionary in AI, has stated:

Artificial intelligence could be the most transformative technology in the history of mankind.

— Kai-Fu Lee, 2018 [212]

Lee completed his doctorate at Carnegie Mellon University (CMU) in speech recognition in 1988, and can therefore be described as the grandfather of Siri and Alexa. Lee ran Google China for a while, then started his own venture capital company. By 2018, Sinovation Ventures' asset management reached US\$2 billion and has invested over 300 portfolios primarily in China, primarily in AI companies. This quote is unusual, in that it is not fear mongering. It is simply an observation of the current world, and a prediction of the future.

3.11 Conclusion

AI is a subject area as broad as mathematics or engineering, with new capabilities being developed almost daily. The key thing to remember about ML is that it is a concept *within* AI, and that DeepNets are a set of techniques *within* ML: AI is not equivalent to ML, and AI is not equivalent to DeepNets. ML is more than just Deep Learning. When choosing an ML algorithm, there are always trade-offs that must be considered to ensure that the approach is appropriate for the desired task. DeepNets have their place for EW problems, but do not neglect or ignore classical ML approaches, or any of the other AI approaches, simply because DeepNets currently have the broadest visibility.

Lastly, in addition to considering trade-offs, prioritize trustworthiness by focusing on the ML model's ability to estimate its own confidence, which is crucial for ensuring reliable AI/ML performance.

References

- [1] Burkov, A., *The Hundred-Page Machine Learning Book*. Québec, Canada: True Positive Inc, 2019.
- [2] Hastie, T., R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY: Springer, 2009.
- [3] Vink, J., and G. de Haan, "Comparison of Machine Learning Techniques for Target Detection," *Artificial Intelligence Review*, Vol. 43, No. 1, 2015. DOI: 10.1007/s10462-012-9366-7.
- [4] Kulin, M., et al., "End-to-end Learning from Spectrum Data," *IEEE Access*, Vol. 6, 2018. DOI: 10.1109/ACCESS.2018.2818794.
- [5] Kuzdeba, S., *Radio Frequency Machine Learning: A Practical Deep Learning Perspective*. Norwood, MA: Artech House, 2025.
- [6] Russell, S., and P. Norvig, *Artificial Intelligence: A Modern Approach*. London, UK: Pearson Education, 2015.
- [7] Cristianini, N., and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, UK: Cambridge University Press, 2000.
- [8] Schölkopf, B., and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2002.

- [9] *Support Vector Machines*. Online: <https://scikit-learn.org/stable/modules/svm.html>. Accessed 2025-02-01.
- [10] Üstün, B., W. Melssen, and L. Buydens, "Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel," *Chemometrics and Intelligent Laboratory Systems*, Vol. 81, No. 1, 2006. DOI: 10.1016/j.chemolab.2005.09.003.
- [11] McCulloch, W., and W. Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, Vol. 5, No. 4, 1943. DOI: 10.1007/BF02478259.
- [12] Rosenblatt, F., "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, Vol. 65, 1958. DOI: 10.1037/h0042519.
- [13] Werbos, P., *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Cambridge, MA: Harvard University, 1975.
- [14] LeCun, Y., Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, Vol. 521, 2015. DOI: 10.1038/nature14539.
- [15] Goodfellow, I., Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016. Online: <http://www.deeplearningbook.org>.
- [16] LeCun, Y., et al., "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, Vol. 86, No. 11, 1998. DOI: 10.1109/5.726791.
- [17] Khan, A., et al., "A Survey of the Recent Architectures of Deep Convolutional Neural Networks," *Artificial Intelligence Review*, Vol. 53, 2020. Online: <https://arxiv.org/abs/1901.06032>.
- [18] Rumelhart, D., et al., "Schemata and Sequential Thought Processes in PDP Models," in *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, Vol. 2, 1987. Online: <https://stanford.edu/~jlmcc/papers/PDP/Chapter14.pdf>.
- [19] Miljanovic, M., "Comparative Analysis of Recurrent and Finite Impulse Response Neural Networks in Time Series Prediction," *Indian Journal of Computer Science and Engineering*, Vol. 3, No. 1, 2012. Online: <https://tinyurl.com/nn-timeseries>.
- [20] Stryker, C., *What is a Recurrent Neural Network (RNN)?* 2024. Online: <https://www.ibm.com/think/topics/recurrent-neural-networks>. Accessed 2025-02-01.
- [21] Baevski, A., et al., *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations*, 2020. Online: <https://arxiv.org/abs/2006.11477>.
- [22] Vorhies, W., *Temporal Convolutional Nets (TCNs) Take Over from RNNs for NLP Predictions*, 2018. Online: <https://www.datasciencecentral.com/temporal-convolutional-nets-tcns-take-over-from-rnns-for-nlp-pred/>. Accessed 2025-02-01.
- [23] Kingma, D., and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends in Machine Learning*, Vol. 12, No. 4, 2019. Online: <https://arxiv.org/abs/1906.02691>.
- [24] Kärkkäinen, T., and J. Hänninen, "Additive Autoencoder for Dimension Estimation," *Neurocomputing*, Vol. 551, 2023. Code at <https://github.com/TommiKark/AdditiveAutoencoder>. DOI: 10.1016/j.neucom.2023.126520.
- [25] Alnaseri, O., et al., *A Review on Deep Learning Autoencoder in the Design of Next-Generation Communication Systems*, 2024. DOI: 10.48550/arXiv.2412.13843.
- [26] Chicco, D., "Siamese Neural Networks: An Overview," in *Artificial Neural Networks*, New York, NY: Springer, 2020. DOI: 10.1007/978-1-0716-0826-5_3.
- [27] Kohonen, T., *Self-Organizing Maps*, 3rd Edition. Springer, 2001. DOI: 10.1007/978-3-642-56927-2.

- [28] Goodfellow, I., et al., “Generative Adversarial Nets,” *Advances in Neural Information Processing Systems*, Vol. 27, 2014. DOI: 10.48550/arXiv.1406.2661.
- [29] Gerstner, W., and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.
- [30] Pfeiffer, M., and T. Pfeil, “Deep Learning with Spiking Neurons: Opportunities and Challenges,” *Frontiers in Neuroscience*, Vol. 12, 2018. DOI: 10.3389/fnins.2018.00774.
- [31] Roy, K., A. Jaiswal, and P. Panda, “Towards Spike-Based Machine Intelligence with Neuromorphic Computing,” *Nature*, Vol. 575, 2019. DOI: 10.1038/s41586-019-1677-2.
- [32] Luo, Z., et al., “Radar Emitter Recognition Based on Spiking Neural Networks,” *Remote Sensing*, Vol. 16, No. 14, 2024. DOI: 10.3390/rs16142680.
- [33] Chen, J., N. Skatchkovsky, and O. Simeone, “Neuromorphic Integrated Sensing and Communications,” *IEEE Wireless Communications Letters*, Vol. 12, No. 3, 2023. DOI: 10.1109/LWC.2022.3231388.
- [34] Mack, J., et al., “RANC: Reconfigurable Architecture for Neuromorphic Computing,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 40, No. 11, 2021. DOI: 10.1109/TCAD.2020.3038151.
- [35] Chen, M., et al., “Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial,” *IEEE Communications Surveys & Tutorials*, Vol. 21, No. 4, 2019. DOI: 10.1109/COMST.2019.2926625.
- [36] Rothman, D., *Transformers for Natural Language Processing: Build Innovative Deep Neural Network Architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and More*. Birmingham, UK: Packt Publishing, 2021.
- [37] Culurciello, E., *Analysis of Deep Neural Networks*, 2018. Online: <https://tinyurl.com/medium-anns>. Accessed 2025-02-01.
- [38] Culurciello, E., *Neural Network Architectures*, 2017. Online: <https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba>. Accessed 2025-02-01.
- [39] Canziani, A., A. Paszke, and E. Culurciello, “An Analysis of Deep Neural Network Models for Practical Applications,” in *CVPR*. IEEE, 2016. Online: <https://arxiv.org/abs/1605.07678>.
- [40] Majumder, U., E. Blasch, and D. Garren, *Deep Learning for Radar and Communications Automatic Target Recognition*. Norwood, MA: Artech House, 2020.
- [41] Bauer, E., and R. Kohavi, “An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants,” *Machine Learning*, Vol. 36, 1999. Online: <http://robotics.stanford.edu/~ronnyk/vote.pdf>.
- [42] Sagi, O., and L. Rokach, “Ensemble Learning: A Survey,” *Data Mining and Knowledge Discovery*, Vol. 8, No. 4, 2018. DOI: 10.1002/widm.1249.
- [43] Ardabili, S., A. Mosavi, and A. Várkonyi-Kóczy, “Advances in Machine Learning Modeling Reviewing Hybrid and Ensemble Methods,” in *Engineering for Sustainable Future*, InterAcademia, 2020. DOI: 10.1007/978-3-030-36841-8_21.
- [44] Dogan, A., and D. Birant, “A Weighted Majority Voting Ensemble Approach for Classification,” in *Computer Science and Engineering*, 2019. DOI: 10.1109/UBMK.2019.8907028.
- [45] Abbasi, M., et al., “Toward Adversarial Robustness by Diversity in an Ensemble of Specialized Deep Neural Networks,” in *Canadian Conference on AI*, 2020. DOI: 10.1007/978-3-030-47358-7_1.

- [46] Tramèr, F., et al., "Ensemble Adversarial Training: Attacks and Defenses," in *ICLR*, 2018. Online: <https://arxiv.org/abs/1705.07204>.
- [47] Du, K.-L., et al., "Foundations and Innovations in Data Fusion and Ensemble Learning for Effective Consensus," *Mathematics*, Vol. 13, No. 4, 2025. DOI: 10.3390/math13040587.
- [48] Breiman, L., "Bagging Predictors," *Machine Learning*, Vol. 24, No. 2, 1996. DOI: 10.1007/BF00058655.
- [49] Freund, Y., and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences*, Vol. 55, No. 1, 1997. DOI: 10.1006/jcss.1997.1504.
- [50] Fragoso, T. M., W. Bertoli, and F. Louzada, "Bayesian Model Averaging: A Systematic Review and Conceptual Classification," *International Statistical Review*, Vol. 86, No. 1, 2018. DOI: 10.1111/insr.12243.
- [51] Hoeting, J. A., et al., "Bayesian Model Averaging: A Tutorial," *Statistical Science*, Vol. 14, No. 4, 1999. DOI: 10.1214/ss/1009212519.
- [52] Reynolds, D., "Gaussian Mixture Models," in *Encyclopedia of Biometrics*, Li, S. Z., and A. Jain, (Eds.) Boston, MA: Springer US, 2009. DOI: 10.1007/978-0-387-73003-5_196.
- [53] Rabiner, L., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, Vol. 77, No. 2, 1989. DOI: 10.1109/5.18626.
- [54] Cover, T., and P. Hart, "Nearest Neighbor Pattern Classification," *IEEE Transactions on Information Theory*, Vol. 13, No. 1, 1967. DOI: 10.1109/TIT.1967.1053964.
- [55] Ester, M., et al., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Knowledge Discovery and Data Mining*, Portland, Oregon: AAAI Press, 1996. DOI: <https://dl.acm.org/doi/10.5555/3001460.3001507>.
- [56] Pearson, K., "On Lines and Planes of Closest Fit to Systems of Points in Space," *Philosophical Magazine*, Vol. 2, No. 11, 1901.
- [57] Hotelling, H., "Analysis of a Complex of Statistical Variables Into Principal Components," *Journal of Educational Psychology*, Vol. 24, 1933.
- [58] Bharadiya, J., "A Tutorial on Principal Component Analysis for Dimensionality Reduction in Machine Learning," *International Journal of Innovative Research in Science Engineering and Technology*, Vol. 8, 2023. DOI: 10.5281/zenodo.8002436.
- [59] Guo, C., et al., *On Calibration of Modern Neural Networks*, 2017. DOI: 10.48550/arXiv.1706.04599.
- [60] Principe, J., et al., "Learning from Examples with Information Theoretic Criteria," *VLSI Signal Processing-Systems for Signal, Image, and Video Technology*, Vol. 26, 2000. DOI: 10.1023/A:1008143417156.
- [61] Santurkar, S., et al., *How Does Batch Normalization Help Optimization?* 2019. Online: <https://arxiv.org/abs/1805.11604>.
- [62] Barbu, A., and S.-C. Zhu, *Monte Carlo Methods*. New York, NY: Springer, 2020. DOI: 10.1007/978-981-13-2971-5.
- [63] Chaudhari, P., et al., "Entropy-SGD: Biasing Gradient Descent into Wide Valleys," in *ICLR*, 2017. Online: <https://arxiv.org/abs/1611.01838>.
- [64] Aha, D., "Integrating Machine Learning with Knowledge-Based Systems," in *New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, 1993. DOI: 10.1109/ANNES.1993.323058.

- [65] Mao, J., et al., “The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences from Natural Supervision,” in *ICLR*, 2019. Online: <https://arxiv.org/abs/1904.12584>.
- [66] Haigh, K. Z., S. Varadarajan, and C. Y. Tang, “Automatic Learning-based MANET Cross-Layer Parameter Configuration,” in *Workshop on Wireless Ad Hoc and Sensor Networks*. IEEE, 2006. Online: <https://www.cs.cmu.edu/~khaigh/papers/haigh06a-configuration.pdf>.
- [67] Towell, G., and J. Shavlik, “Knowledge-Based Artificial Neural Networks,” *Artificial Intelligence*, Vol. 70, No. 1, 1994. DOI: 10.1016/0004-3702(94)90105-8.
- [68] Yu, T., S. Simoff, and D. Stokes, “Incorporating Prior Domain Knowledge Into a Kernel Based Feature Selection Algorithm,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2007. DOI: 10.1007/978-3-540-71701-0_120.
- [69] Muralidhar, N., et al., “Incorporating Prior Domain Knowledge into Deep Neural Networks,” in *Big Data*, 2018. Online: <http://people.cs.vt.edu/naren/papers/PID5657885.pdf>.
- [70] Agrell, C., et al., “Pitfalls of Machine Learning for Tail Events in High Risk Environments,” in *Safety and Reliability—Safe Societies in a Changing World*, 2018. DOI: 10.1201/9781351174664-381.
- [71] d’Avila Garcez, A., et al., “Neural-Symbolic Computing: An Effective Methodology for Principled Integration of Machine Learning and Reasoning,” *Journal of Applied Logics*, Vol. 6, No. 4, 2019. Online: <https://arxiv.org/abs/1905.06088>.
- [72] *Neurosymbolic Artificial Intelligence*. Online: <http://www.neural-symbolic.org/>. Workshops, conferences, and resources. Accessed 2025-02-12.
- [73] Willard, J., et al., “Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems,” *ACM Computing Surveys*, Vol. 55, No. 4, 2022. DOI: 10.1145/3514228.
- [74] Hitzler, P., M. Sarker, and A. Eberhart, (Eds.), *Compendium of Neurosymbolic Artificial Intelligence*. Clifton, VA: IOS Press, 2023.
- [75] MathWorks, *Modeling the Propagation of Radar Signals*. Online: <https://tinyurl.com/matlab-path-loss>. Accessed 2025-02-11.
- [76] Benaich, N., and A. Chalmers, *State of AI Report*, 2024. Online: <https://www.stateof.ai/>. Accessed 2025-01-01.
- [77] Wan, Z., et al., *Towards Cognitive AI Systems: A Survey and Prospective on Neuro-Symbolic AI*, 2023. Online: <https://arxiv.org/abs/2401.01040>.
- [78] Moseley, B., “Physics-Informed Machine Learning: From Concepts to Real-World Applications,” Ph.D. dissertation, University of Oxford, Oxford, UK, 2022. DOI: 10.13039/501100000266.
- [79] Dolean, V., et al., “Multilevel domain decomposition-based architectures for physics-informed neural networks,” *Computer Methods in Applied Mechanics and Engineering*, Vol. 429, 2024. DOI: 10.1016/j.cma.2024.117116.
- [80] Forbes Technology Council, “Neurosymbolic AI: 20 Practical Real-World Applications,” *Forbes*, 2024. Accessed 2025-02-11. Online: <https://tinyurl.com/forbes-neurosymbolic>.
- [81] Kochkov, D., et al., “Neural General Circulation Models for Weather and Climate,” *Nature*, Vol. 632, No. 8027, 2024. DOI: 10.1038/s41586-024-07744-y.
- [82] van Bekkum, M., et al., “Modular Design Patterns for Hybrid Learning and Reasoning Systems,” *Applied Intelligence*, Vol. 51, No. 9, 2021. DOI: 10.1007/s10489-021-02394-3.

- [83] *Neurosymbolic Artificial Intelligence*. Online: <https://neurosymbolic-ai-journal.com/>. Journal. Accessed 2025-03-03.
- [84] Scheirer, W., et al., "Toward Open Set Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 35, No. 7, 2013. DOI: 10.1109/TPAMI.2012.256.
- [85] Geng, C., S.-J. Huang, and S. Chen, "Recent Advances in Open Set Recognition: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 43, No. 10, 2021. DOI: 10.1109/tpami.2020.2981604.
- [86] Gharoun, H., et al., *Meta-learning approaches for few-shot learning: A survey of recent advances*, 2023. Online: <https://arxiv.org/abs/2303.07502>.
- [87] Khoei, A. G., Y. Yu, and R. Feldt, "Domain Generalization Through Meta-learning: A Survey," *Artificial Intelligence Review*, Vol. 57, No. 10, 2024. DOI: 10.1007/s10462-024-10922-z.
- [88] Mattei, E., et al., "Feature Learning for Enhanced Security in the Internet of Things," in *Global Conference on Signal and Information Processing*. IEEE, 2019. DOI: 10.1109/GlobalSIP45357.2019.8969222.
- [89] Robyns, P., et al., "Physical-Layer Fingerprinting of LoRa Devices Using Supervised and Zero-Shot Learning," in *Conference on Security and Privacy in Wireless and Mobile Networks*, 2017. DOI: 10.1145/3098243.3098267.
- [90] Tokmakov, P., Y.-X. Wang, and M. Hébert, "Learning Compositional Representations for Few-Shot Recognition," in *CVPR*. IEEE, 2019. Online: <https://arxiv.org/abs/1812.09213>.
- [91] Wang, W., et al., "A Survey of Zero-Shot Learning: Settings, Methods, and Applications," *ACM Transactions on Intelligent Systems and Technology*, Vol. 10, No. 12, 2019. DOI: 10.1145/3293318.
- [92] Geng, C., S.-J. Huang, and S. Chen, "Recent Advances in Open Set Recognition: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. DOI: 10.1109/TPAMI.2020.2981604.
- [93] Cao, A., Y. Luo, and D. Klabjan, *Open-Set Recognition with Gaussian Mixture Variational Autoencoders*, 2020. Online: <https://arxiv.org/abs/2006.02003>.
- [94] Zhang, L., et al., "From Sample Poverty to Rich Feature Learning: A New Metric Learning Method for Few-Shot Classification," *IEEE Access*, Vol. 12, 2024. DOI: 10.1109/ACCESS.2024.3444483.
- [95] Khosla, S., Z. Zhu, and Y. He, *Survey on Memory-Augmented Neural Networks: Cognitive Insights to AI Applications*, 2023. Online: <https://arxiv.org/abs/2312.06141>.
- [96] Gkanatsios, N., et al., "Analogy-Forming Transformers for Few-Shot 3D Parsing," in *ICLR*, 2023. DOI: 10.48550/arXiv.2304.14382.
- [97] Apfeld, S., and A. Charlish, "Recognition of Unknown Radar Emitters with Machine Learning," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 57, No. 6, 2021. DOI: 10.1109/TAES.2021.3098125.
- [98] Sanh, V., et al., *Multitask Prompted Training Enables Zero-Shot Task Generalization*, 2022. DOI: 10.48550/arXiv.2110.08207.
- [99] Kumar, S., et al., "Meta-Learning for Real-World Class Incremental Learning: A Transformer-Based Approach," *Scientific Reports*, Vol. 14, 2024. DOI: 10.1038/s41598-024-71125-8.
- [100] Jing, L., and Y. Tian, "Self-Supervised Visual Feature Learning with Deep Neural Networks: A Survey," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, Vol. 43, No. 11, 2021. DOI: 10.1109/TPAMI.2020.2992393.

- [101] Gidariss, S., et al., *Tutorial on Leave Those Nets Alone: Advances in Self-Supervised Learning*, 2021. Online: <https://gidariss.github.io/self-supervised-learning-cvpr2021/>.
- [102] Lin, A., et al., "Unknown Radar Waveform Recognition Based on Transferred Deep Learning," *IEEE Access*, Vol. 8, 2020. DOI: 10.1109/ACCESS.2020.3029192.
- [103] Huang, J., et al., "Few-Shot Learning for Radar Emitter Signal Recognition Based on Improved Prototypical Network," *Remote Sensing*, Vol. 14, No. 7, 2022. DOI: 10.3390/rs14071681.
- [104] Feng, H. C., B. Tang, and T. Wan, "Radar pulse repetition interval modulation recognition with combined net and domain-adaptive few-shot learning," *Digital Signal Processing*, Vol. 127, 2022. DOI: 10.1016/j.dsp.2022.103562.
- [105] Huang, Z., et al., *Radar Signal Recognition through Self-Supervised Learning and Domain Adaptation*, 2025. Online: <https://arxiv.org/abs/2501.03461>.
- [106] Milosheski, L., et al., "Self-supervised Learning for Clustering of Wireless Spectrum Activity," *Computer Communications*, Vol. 212, 2023. DOI: 10.1016/j.comcom.2023.10.009.
- [107] Maudsley, D., "A Theory of Meta-Learning and Principles of Facilitation: An Organismic Perspective," Ph.D. dissertation, University of Toronto, Ontario, Canada, 1979.
- [108] Brazdil, P., and C. Giraud-Carrier, "Metalearning and Algorithm Selection: Progress, State of the Art and Introduction to the 2018 Special Issue," *Machine Learning*, Vol. 107, 2018. DOI: 10.1007/s10994-017-5692-y.
- [109] Vanschoren, J., "Meta-Learning," in *Automated Machine Learning*, New York, NY: Springer, 2019. DOI: 10.1007/978-3-030-05318-5_2.
- [110] Hospedales, T., et al., *Meta-Learning in Neural Networks: A Survey*, 2020. Online: <https://arxiv.org/abs/2004.05439>.
- [111] Vettoruzzo, A., et al., "Advances and Challenges in Meta-Learning: A Technical Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 46, No. 7, 2024. DOI: 10.1109/TPAMI.2024.3357847.
- [112] Colelough, B., and W. Regli, "Neuro-Symbolic AI in 2024: A Systematic Review," in *Logical Foundations of Neuro-Symbolic AI*, 2024. Online: <https://ceur-ws.org/Vol-3819/paper3.pdf>.
- [113] Rendell, L., R. Seshu, and D. Tchong, "More Robust Concept Learning Using Dynamically-Variable Bias," in *Workshop on Machine Learning*, 1987. DOI: 10.1016/B978-0-934613-41-5.50011-8.
- [114] Schmidhuber, J., "Evolutionary Principles in Self-Referential Learning, or on Learning How to Learn: The Meta-Meta-... Hook," M.S. thesis, Technische Universität München, Munich, Germany, 1987. Online: <http://people.idsia.ch/~juergen/diploma.html>.
- [115] Bengio, Y., S. Bengio, and J. Cloutier, "Learning a Synaptic Learning Rule," in *IJCNN*, Vol. ii, 1991. Online: <http://bengio.abracadoudou.com/publications/pdf/bengio-1991-ijcnn.pdf>.
- [116] Brazdil, P., et al., *Metalearning: Applications to Automated Machine Learning and Data Mining*, 2nd Edition. New York, NY: Springer, 2022.
- [117] Lippe, P., *Tutorial 16: Meta-Learning - Learning to Learn*. Online: https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial16/Meta_Learning.html. Accessed 2025-02-12.
- [118] Amorim, L. B. V. de, G. D. C. Cavalcanti, and R. M. O. Cruz, "Meta-Scaler: A Meta-Learning Framework for the Selection of Scaling Techniques," *IEEE Transactions on Neural Networks and Learning Systems*, 2024. DOI: 10.1109/TNNLS.2024.3366615.

- [119] Balaji, Y., S. Sankaranarayanan, and R. Chellappa, "MetaReg: Towards Domain Generalization using Meta-Regularization," in *NeurIPS*, Vol. 31, Curran Associates, Inc., 2018. DOI: <https://dl.acm.org/doi/10.5555/3326943.3327036>.
- [120] Parmezan, A. R. S., H. D. Lee, and F. C. Wu, "Metalearning for Choosing Feature Selection Algorithms in Data Mining: Proposal of a New Framework," *Expert Systems with Applications*, Vol. 75, 2017. DOI: 10.1016/j.eswa.2017.01.013.
- [121] Rusu, A., et al., "Meta-Learning with Latent Embedding Optimization," in *ICLR*, 2019. Online: <https://arxiv.org/abs/1807.05960>.
- [122] Zintgraf, L., et al., "Fast Context Adaptation via Meta-Learning," in *ICLR*, 2019. Online: <https://arxiv.org/abs/1810.03642>.
- [123] Yu, H., et al., "Meta-ADD: A Meta-learning Based Pre-trained Model for Concept Drift Active Detection," *Information Sciences*, Vol. 608, 2022. DOI: 10.1016/j.ins.2022.07.022.
- [124] Aguiar, G. J., and A. Cano, "Enhancing Concept Drift Detection in Drifting and Imbalanced Data Streams through Meta-Learning," in *BigData*, 2023. DOI: 10.1109/BigData59044.2023.10386364.
- [125] Pham, H., et al., "Meta Pseudo Labels," in *CVPR*, 2021.
- [126] Zheng, G., A. H. Awadallah, and S. Dumais, "Meta Label Correction for Noisy Label Learning," *AAAI*, Vol. 35, No. 12, 2021. DOI: 10.1609/aaai.v35i12.17319.
- [127] Settles, B., "Active Learning Literature Survey," University of Wisconsin-Madison, Madison, WI, Tech. Rep. 1648, 2009.
- [128] Liu, Q., and M. Hauswirth, "A Provenance Meta Learning Framework for Missing Data Handling Methods Selection," in *Ubiquitous Computing, Electronics & Mobile Communication*, 2020. DOI: 10.1109/UEMCON51285.2020.9298089.
- [129] Fatyanosa, T. N., et al., "Meta-learning for Vessel Time Series Data Imputation Method Recommendation," *Expert Systems with Applications*, Vol. 251, 2024. DOI: 10.1016/j.eswa.2024.124016.
- [130] Cubuk, E., et al., "AutoAugment: Learning Augmentation Strategies from Data," in *CVPR*, 2019. DOI: 10.1109/CVPR.2019.00020, Code available at <https://paperswithcode.com/method/autoaugment>.
- [131] Tamkin, A., M. Wu, and N. Goodman, "Viewmaker Networks: Learning Views for Unsupervised Representation Learning," in *ICLR*, 2020. Online: <https://arxiv.org/abs/2010.07432>.
- [132] Ni, R., et al., "Data Augmentation for Meta-Learning," in *ICML*, 2021. Online: <https://proceedings.mlr.press/v139/ni21a.html>.
- [133] Kotlar, M., et al., "Novel Meta-Features for Automated Machine Learning Model Selection in Anomaly Detection," *IEEE Access*, Vol. 9, 2021. DOI: 10.1109/ACCESS.2021.3090936.
- [134] Finn, C., and S. Levine, "Meta-Learning and Universality: Deep Representations and Gradient Descent can Approximate any Learning Algorithm," in *ICLR*, 2018.
- [135] Brazdil, P., et al., "Metalearning for Hyperparameter Optimization," in *Metalearning*, 2022. DOI: 10.1007/978-3-030-67024-5_6.
- [136] Madireddy, S., A. Yanguas-Gil, and P. Balaprakash, "Neuromorphic Architecture Optimization for Task-Specific Dynamic Learning," in *Neuromorphic Systems*, 2019. DOI: 10.1145/3354265.3354270.
- [137] Chen, Y., et al., "Learning to Learn without Gradient Descent by Gradient Descent," in *ICML*, Vol. 70, 2017. Online: <http://proceedings.mlr.press/v70/chen17e.html>.

-
- [138] Li, K., and J. Malik, "Learning to Optimize," in *ICLR*, 2017. Online: <https://openreview.net/pdf?id=ry4Vrt5gl>.
 - [139] Vilalta, R., and Y. Drissi, "A Perspective View and Survey of Meta-Learning," *Artificial Intelligence Review*, Vol. 18, 2005. DOI: 10.1023/A:1019956318069.
 - [140] Ma, Z., et al., "MetaBox: A Benchmark Platform for Meta-Black-Box Optimization with Reinforcement Learning," in *NeurIPS*, 2023. Online: <https://neurips.cc/virtual/2023/poster/73497>.
 - [141] Amos, B., and D. Yarats, "The Differentiable Cross-Entropy Method," in *ICML*, 2020. Online: <https://arxiv.org/abs/1909.12830>.
 - [142] Ng, A., and S. Russell, "Algorithms for Inverse Reinforcement Learning," in *ICML*, 2000.
 - [143] Kaufmann, T., et al., "A Survey of Reinforcement Learning from Human Feedback," *Submitted to Transactions on Machine Learning Research*, 2024. Online: <https://arxiv.org/abs/2312.14925>.
 - [144] Finn, C., P. Abbeel, and S. Levine, *Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks*, 2017. Online: <https://arxiv.org/abs/1703.03400>.
 - [145] Burega, B., J. Martin, and M. Bowling, "Learning to Prioritize Planning Updates in Model-based Reinforcement Learning," in *Workshop on Meta-learning at NeurIPS*, 2022.
 - [146] Schrum, M., et al., *Meta-Active Learning in Probabilistically Safe Optimization*, 2022. DOI: 10.48550/arXiv.2007.03742.
 - [147] Ibrahim, S., et al., "Comprehensive Overview of Reward Engineering and Shaping in Advancing Reinforcement Learning Applications," *IEEE Access*, Vol. 12, 2024. DOI: 10.1109/ACCESS.2024.3504735.
 - [148] Zou, H., et al., *Reward Shaping via Meta-Learning*, 2019. Online: <https://arxiv.org/abs/1901.09330>.
 - [149] Hu, Y., et al., *Learning to Utilize Shaping Rewards: A New Approach of Reward Shaping*, 2020. Online: <https://arxiv.org/abs/2011.02669>.
 - [150] Xu, Z., et al., "Meta-Gradient Reinforcement Learning with an Objective Discovered Online," in *NeurIPS*, 2020. Online: <https://arxiv.org/abs/2007.08433>.
 - [151] Duan, Y., et al., "RL²: Fast Reinforcement Learning via Slow Reinforcement Learning," in *ICLR*, 2017. Online: <https://arxiv.org/abs/1611.02779>.
 - [152] Mantovani, R. G., et al., "To Tune or Not to Tune: Recommending When to Adjust SVM Hyper-Parameters via Meta-Learning," *IJCNN*, 2015. DOI: 10.1109/IJCNN.2015.7280644.
 - [153] Adkins, J., M. Bowling, and A. White, "A Method for Evaluating Hyperparameter Sensitivity in Reinforcement Learning," in *NeurIPS*, 2024. Online: <https://openreview.net/forum?id=4OJdZhcwBb>.
 - [154] Luedtke, A., I. Chung, and O. Sofrygin, "Adversarial Monte Carlo Meta-Learning of Optimal Prediction Procedures," *Journal of Machine Learning Research*, Vol. 22, No. 1, 2021. Online: <https://www.jmlr.org/papers/volume22/20-1065/20-1065.pdf>.
 - [155] Zhang, C., et al., "Adversarial Task Up-sampling for Meta-learning," in *Learning Representations*, 2022. Online: <https://openreview.net/forum?id=pFqgUJxXXz>.
 - [156] Lawton, G., *What is Generative AI?* 2024. Online: <https://www.techtarget.com/searchenterpri/seai/definition/generative-AI>. Accessed 2024-12-11.

- [157] Akhtar, Z., "Unveiling the Evolution of Generative AI (GAI): A Comprehensive and Investigative Analysis Toward LLM Models (2021?2024) and Beyond," *Journal of Electrical Systems and Information Technology*, Vol. 11, No. 22, 2024. DOI: 10.1186/s43067-024-00145-1.
- [158] Ziemann, M., and C. Metzler, "Adaptive LPD Radar Waveform Design with Generative Deep Learning," in *Signals, Systems, and Computers*, 2023. DOI: 10.48550/arXiv.2403.12254.
- [159] Kingma, D., and M. Welling, "Auto-Encoding Variational Bayes," in *ICLR*, 2014. DOI: 10.48550/arXiv.1312.6114.
- [160] Bergmann, D., and C. Stryker, *What Is a Variational Autoencoder?* 2024. Online: <https://www.ibm.com/think/topics/variational-autoencoder>. Accessed 2024-12-11.
- [161] Vaswani, A., et al., "Attention Is All You Need," in *NeurIPS*, 2017. DOI: 10.48550/arXiv.1706.03762.
- [162] Weidinger, L., et al., "Ethical and Social Risks of Harm from Language Models," 2021. Online: <https://dl.acm.org/doi/10.1145/3531146.3533088>.
- [163] Nah, F. F.-H., et al., "Generative AI and ChatGPT: Applications, Challenges, and AI-Human Collaboration," *Journal of Information Technology Case and Application Research*, Vol. 25, No. 3, 2023. DOI: 10.1080/15228053.2023.2233814.
- [164] Kenthapadi, K., H. Lakkaraju, and N. Rajani, "Generative AI meets Responsible AI: Practical Challenges and Opportunities," in *Knowledge Discovery and Data Mining*, 2023. DOI: 10.1145/3580305.3599557.
- [165] Manduchi, L., et al., *On the Challenges and Opportunities in Generative AI*, 2024. DOI: 10.48550/arXiv.2403.00025.
- [166] Mei, K., et al., "Bigger is not Always Better: Scaling Properties of Latent Diffusion Models," *Transactions on Machine Learning Research*, 2024. DOI: 10.48550/arXiv.2404.01367.
- [167] Zhu, M., et al., *LatentExplainer: Explaining Latent Representations in Deep Generative Models with Multimodal Large Language Models*, 2025. DOI: 10.48550/arXiv.2406.14862.
- [168] Jiang, Y., et al., "Jamming Waveform Generation Method Based on Generative Adversarial Network Model," *IEEE Signal Processing Letters*, Vol. 31, 2024. DOI: 10.1109/LSP.2024.3386491.
- [169] Paialunga, P., *Hands-on Generative Adversarial Networks (GAN) for Signal Processing, with Python*. Online: <https://tinyurl.com/gan-for-signals>. Accessed 2025-02-17.
- [170] MathWorks, *What Are GANs?* 2025. Online: <https://www.mathworks.com/discovery/generative-adversarial-networks.html>. Accessed 2025-02-17.
- [171] Cui, Y., et al., "Sensing-Assisted High Reliable Communication: A Transformer-Based Beamforming Approach," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 18, No. 5, 2024. DOI: 10.1109/JSTSP.2024.3405859.
- [172] Zhou, Z., et al., "Radar Signal Sorting with Multiple Self-Attention Coupling Mechanism Based Transformer Network," *IEEE Signal Processing Letters*, Vol. 31, 2024. DOI: 10.1109/LSP.2024.3421948.
- [173] Esmaeili, F., et al., "Anomaly Detection for Sensor Signals Utilizing Deep Learning Autoencoder-Based Neural Networks," *Bioengineering*, Vol. 10, No. 4, 2023. DOI: 10.3390/bioengineering10040405.
- [174] Wai, I. W. C., et al., "Joint Interference Cancellation and Signal Detection Using Latent Space Representations in VAE," *IEEE Transactions on Consumer Electronics*, Vol. 70, No. 1, 2024. DOI: 10.1109/TCE.2023.3324979.

- [175] Brown, C., and B. Rong, "Enhanced IoT Spectrum Utilization: Integrating Geospatial and Environmental Data for Advanced Mid-Band Spectrum Sharing," *Sensors*, Vol. 24, No. 18, 2024. DOI: 10.3390/s24185885.
- [176] Lee, W., and J. Park, *LLM-Empowered Resource Allocation in Wireless Communications Systems*, 2024. Online: <https://arxiv.org/abs/2408.02944>.
- [177] Edge AI Foundation. Online: <https://www.edgeaifoundation.org/>. Accessed 2025-02-17.
- [178] Warden, P., and D. Situnayake, *TinyML*. Sebastopol, CA: O'Reilly Media, Inc., 2019.
- [179] Zahra, F. T., Y. S. Bostanci, and M. Soyuturk, "Real-Time Jamming Detection in Wireless IoT Networks," *IEEE Access*, Vol. 11, 2023. DOI: 10.1109/ACCESS.2023.3293404.
- [180] Kim, J., S. Chang, and N. Kwak, "PQK: Model Compression via Pruning, Quantization, and Knowledge Distillation," in *Interspeech*, 2021. DOI: 10.21437/Interspeech.2021-248.
- [181] Li, Z., H. Li, and L. Meng, "Model Compression for Deep Neural Networks: A Survey," *Computers*, Vol. 12, No. 3, 2023. DOI: 10.3390/computers12030060.
- [182] Bibi, U., et al., "Advances in Pruning and Quantization for Natural Language Processing," *IEEE Access*, Vol. 12, 2024. DOI: 10.1109/ACCESS.2024.3465631.
- [183] Malihi, L., and G. Heidemann, "Matching the Ideal Pruning Method with Knowledge Distillation for Optimal Compression," *Applied System Innovation*, Vol. 7, No. 4, 2024. DOI: 10.3390/asi7040056.
- [184] Ma, S., et al., "The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits," *CoRR*, Vol. abs/2402.17764, 2024. Online: 10.48550/arXiv.2402.17764.
- [185] Sreenivas, S. T., et al., *LLM Pruning and Distillation in Practice: The Minitron Approach*, 2024. Online: <https://arxiv.org/abs/2408.11796>.
- [186] Shao, Z., et al., *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*, 2024. Online: <https://arxiv.org/abs/2402.03300>.
- [187] Lee, S., et al., *Explore, Select, Derive, and Recall: Augmenting LLM with Human-like Memory for Mobile Task Automation*, 2024. DOI: 10.48550/arXiv.2312.03003.
- [188] Danopoulos, D., et al., "FPGA Acceleration of Generative Adversarial Networks for Image Reconstruction," in *Modern Circuits and Systems Technologies*, 2021. DOI: 10.1109/MOCAS T52088.2021.9493361.
- [189] Chen, T., et al., "Self-Supervised GANs via Auxiliary Rotation Loss," 2019. DOI: 10.48550/arXiv.1811.11212.
- [190] Chen, B., et al., "SLIDE: In Defense of Smart Algorithms over Hardware Acceleration for Large-Scale Deep Learning Systems," in *Machine Learning and Systems*, Dhillon, I., D. Papailiopoulos, and V. Sze, (Eds.), Vol. 2, 2020. DOI: 10.48550/arXiv.1903.03129.
- [191] Daghighi, S., et al., "Accelerating SLIDE Deep Learning on Modern CPUs: Vectorization, Quantizations, Memory Optimizations, and More," in *Machine Learning and Systems*, Vol. 3, 2021. DOI: 10.48550/arXiv.2103.10891.
- [192] Mudraje, I., K. Vogelgesang, and T. Herfet, *A 1-D CNN Inference Engine for Constrained Platforms*, 2025. Online: <https://arxiv.org/abs/2501.17269>.
- [193] Shuvo, M. M. H., et al., "Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review," *Proceedings of the IEEE*, Vol. 111, No. 1, 2023. DOI: 10.1109/JPROC.2022.3226481.
- [194] Daghero, F., D. J. Pagliari, and M. Poncino, "Chapter Eight - Energy-efficient deep learning inference on edge devices," in *Hardware Accelerator Systems for Artificial Intelligence and*

- Machine Learning*, Kim, S., and G. C. Deka, (Eds.), Vol. 122, Elsevier, 2021. DOI: 10.1016/bs.adcom.2020.07.002.
- [195] Page, S., *The Model Thinker*. New York, NY: Hachette, 2018.
 - [196] Wolpert, D., "The Lack of A Priori Distinctions Between Learning Algorithms," *Neural Computation*, 1996. DOI: 10.1162/neco.1996.8.7.1341.
 - [197] Golestaneh, P., M. Taheri, and J. Lederer, "How Many Samples are Needed to Train a Deep Neural Network?" In *ICLR*, 2024. Online: <https://arxiv.org/abs/2405.16696>.
 - [198] Lim, T.-S., W.-Y. Loh, and Y.-S. Shih, "A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms," *Machine Learning*, Vol. 40, 2000. Online: <https://tinyurl.com/classifier-comparison>.
 - [199] Fuscone, S., *Have you Ever Wondered Why Some ML Models Take Forever to Train?* 2025. Online: <https://tinyurl.com/fuscone-bigO>. Accessed 2025-02-23.
 - [200] Kearns, M., U. Vazirani, and U. Vazirani, *An Introduction to Computational Learning Theory*. Cambridge, MA: MIT Press, 1994. DOI: 10.7551/mitpress/3897.001.0001.
 - [201] Vaswani, A., et al., "Attention Is All You Need," in *NeurIPS*, 2017. DOI: 10.48550/arXiv.1706.03762.
 - [202] Anderson, M., *Three Challenges Ahead for Stable Diffusion*, 2022. Online: <https://www.unite.ai/three-challenges-ahead-for-stable-diffusion/>.
 - [203] Clark, J., and R. Perrault, *The AI Index Report: Measuring trends in Artificial Intelligence*, 2022. Online: <https://aiindex.stanford.edu/ai-index-report-2022/>.
 - [204] Ng, A., *Generative AI for Everyone Enrolls 43K Users in First 7 Days*, 2023. Online: <https://x.com/AndrewYNg/status/1724118433414156624>. Accessed 2025-01-01.
 - [205] Chollet, F., "On the Measure of Intelligence," *Computing Research Repository*, 2019. Online: <http://arxiv.org/abs/1911.01547>.
 - [206] Lee, K.-F., *AI Superpowers: China, Silicon Valley, and the New World Order*. New York, NY: Harper Business, 2018.
 - [207] China's State Council, *A New Generation Artificial Intelligence Development Plan*, 2017. Online: <https://d1y8sb8igg2f8e.cloudfront.net/documents/translation-fulltext-8.1.17.pdf>.
 - [208] Hannas, W. C., and H.-M. Chang, (Eds.), *Chinese Power and Artificial Intelligence*. New York, NY: Routledge, 2023.
 - [209] Benaich, N., and I. Hogarth, *State of AI Report*, 2021. Online: <https://www.stateof.ai/>. Accessed 2025-01-01.
 - [210] Wong-Leung, K., S. Robin, and D. Cave, *ASPI's Two-decade Critical Technology Tracker: The Rewards of Long-term Research Investment*, 2024. Online: <https://aspi.org.au/report/aspi-two-decade-critical-technology-tracker>. Accessed 2025-01-01.
 - [211] Togelius, J., and G. Yannakakis, "Choose Your Weapon: Survival Strategies for Depressed AI Academics," Vol. 12, 2024. DOI: 10.1109/JPROC.2024.3364137.
 - [212] Lee, K.-F., *How AI Will Completely Change the Way We Live in the Next 20 Years*, 2021. Online: <https://time.com/6097625/kai-fu-lee-book-ai-2041/>.

4

Electronic Support

The first step in every CogEW system is *electronic support (ES)* to understand the RF spectrum. Known as situation assessment in the AI community, ES determines who is using the spectrum, where and when they are using it, and whether there are patterns that can be exploited. This chapter discusses AI/ML techniques that can be used for feature estimation, emitter characterization and classification, data fusion, anomaly detection, and intent recognition. ES analyzes the environment and creates the observables that drive decision making. The chapter is organized according to the increasing complexity implied by Figure 1.8.

Howland et al. [1] define spectrum situation awareness (SSA) as “a means of collecting disparate information about spectrum use and processing this information to produce a fused spectrum picture.” SSA collects, organizes, and processes the spectrum data required for EW. In addition to pre- and post-mission analysis, SSA needs to occur in rapid real time, according to the needs of the decision maker.

There are five key steps in SSA (illustrated in Figure 9.1):

- *Collect*: Sense the environment and gather raw data; EW data is streaming and large volume, so much of this data is discarded.
- *Validate*: Verify the accuracy, reliability, and relevance of the collected data.
- *Fuse*: Integrate and combine validated data from multiple sources to create a unified understanding of the situation.
- *Determine Impact*: Evaluate the potential consequences of the current situation, including who is affected and how, and impact on outcomes.
- *Infer Intent*: Deduce the objectives of participants and infer their possible future actions.

To reduce brittleness and handle novel emitters and adversaries, AI and ML capabilities can improve SSA at every level. Figure 4.1 depicts a view of these AI/ML technologies, in the context of other relevant SSA technologies. A complete EW system must have multifaceted SSA.

Future SSA systems can be trained with deep learning models for latent feature generation, classical ML models for in-mission updates, and hybrid models to offset limited data (Section 4.2). Moreover, SSA does not have to solely rely

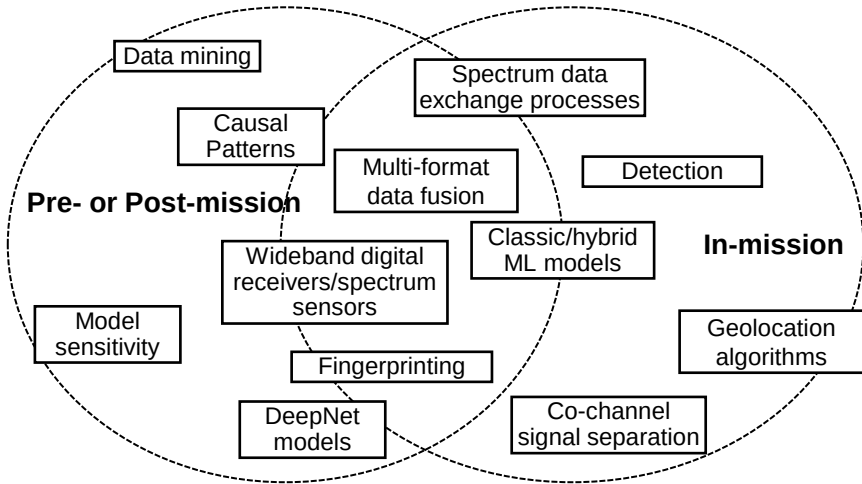


Figure 4.1 SSA must combine a variety of supporting technologies, both traditional and cognitive. One of the challenges lies in the integration and demonstration of relevant technologies, only a few of which are cognitive.

on the RF data: It can be fused with non-RF data such as video and still imagery, free-space optics, or open-source, tactical, or operational intelligence (Section 4.4). Distributed data fusion across multiple heterogeneous sources must create a coherent battlespace spectrum common operating picture that is accurate in space, time, and frequency. Anomaly detection (Section 4.5), causal reasoning (Section 4.6), and intent inference (Section 4.7) complete the picture to understand the impact of events and support decision making.

4.1 Detection, Localization, and Signal Separation

Detection and *localization* are core functions in CogEW, enabling situational awareness, threat tracking, and, subsequently, EA in complex electromagnetic environments. Detection identifies the presence of an emitter, while localization determines its position, velocity, and trajectory. These functions use active or passive methods, each with trade-offs in precision, stealth, and resilience.

Active detection involves transmitting signals and analyzing their reflections, as in radar-based systems. Cognitive radars enhance resilience using adaptive waveforms, beamforming, and power control to counter various EW threats [2–4]. Techniques such as pulse-Doppler processing, synthetic aperture radar (SAR), and

low probability of intercept (LPI) radar improve detection accuracy while reducing vulnerability [5, 6]. Multistatic radar further enhances robustness by using multiple receivers to mitigate jamming [7]. However, active detection reveals the emitter's presence, making it susceptible to adversary geolocation and attack.

Passive detection, by contrast, does not transmit but intercepts adversary emissions, enabling covert monitoring of RF signals. It employs RF spectrum monitoring, cyclostationary detection, and ML-based classification to identify threats in contested environments [8, 9]. Passive detection on mobile platforms will play a crucial role in ES by leveraging ambient signals for stealthy, adaptive threat detection. Cristallini and O'Hagan [10] highlight challenges in compensating for motion-induced Doppler shifts in passive radar, essential for real-time cognitive adaptation. Krückemeier [11] demonstrates its effectiveness in drone and aircraft surveillance, a key aspect of autonomous EW systems. Passive radar can also enhance drone detection by exploiting existing RF emissions [12] and improve automotive situational awareness through 5G signal reflections [13]. In summary, passive detection evades direct jamming but depends on signal availability, making it susceptible to disruptions like frequency hopping and deceptive emissions. Additionally, overcoming challenges such as noise uncertainty, low signal-to-noise ratios, and channel fading requires robust and adaptive solutions [9].

Localization refines detection by determining an emitter's position. Active methods, such as radar range measurement [14] and Doppler shift estimation [15], provide high accuracy but expose the observer. SAR and inverse SAR further improve localization by generating high-resolution target images [16, 17]. Passive localization techniques, including time-difference-of-arrival, frequency-difference-of-arrival, and angle-of-arrival, enable covert geolocation by analyzing signals at multiple receiver sites [18]. Hybrid methods combine these techniques to improve accuracy in dynamic environments [19]. In addition, CNNs and transformers enable joint target tracking and classification by extracting observation features and target states [20]. Sensor fusion further integrates active and passive localization, optimizing precision [21] while minimizing exposure. Section 4.4.2 discusses localization in the context of 5G.

A major challenge in signal processing is *deinterleaving*, also known by a variety of names including *pulse train separation* or *signal separation*, which distinguishes multiple overlapping signals and splits them into separate groups, for example, distinguishing multiple overlapping pulse streams from different radar sources. Traditional methods include independent component analysis, principal

component analysis, time-frequency analysis, wavelets, and second- and higher-order statistics for activities such as time-of-arrival clustering and pulse repetition interval analysis [22, 23]. These have been enhanced or replaced by modern ML techniques improving emitter classification in contested environments [24–26]. Density-based clustering effectively deinterleaves radar pulses even with 30% jitter [27]. SVMs [28] worked well to distinguish emitters with complex parameters, but overlapping parameter ranges limit effectiveness in more complex environments. Gaussian mixture models can address this challenge, and also recognize out-of-distribution pulses [29]. Deinterleaving of radar pulses has been approached using implicit feature extraction through LSTM networks for pulse sequence trajectory tracking [30]. Liu et al. proposed a method based on RNNs to classify and denoise pulse train patterns by extracting temporal features from multicomponent pulse sequences [31]. Other DeepNet approaches include self-organizing maps [32, 33], autoencoders [34], transformers [35, 36], and a variety of hybrid methods [37–40].

Deinterleaving also plays a crucial role in localization by ensuring correct signal-source association before applying geolocation algorithms [41]. For example, Kocamış et al. present a deep learning-based radar signal deinterleaving method using object detection on amplitude patterns from radar beam motions, improving signal-source association for geolocation [39].

Irregularly spaced time series, where events occur unpredictably and with nonuniform gaps, can be useful in CogEW for analyzing adversarial signal patterns. In many cases, precise timing information is unavailable or unreliable due to interference, environmental factors, or the stochastic nature of emissions. Traditional time-series models struggle in these scenarios, necessitating specialized approaches. Temporal point processes provide a framework for modeling sequences of discrete, stochastic events [42]. Hawkes processes [43], for instance, capture self-exciting behaviors where one event increases the likelihood of subsequent ones, making them useful for radar emission pattern prediction, electronic countermeasures (ECM), and adversarial signal analysis. Inhomogeneous Poisson and renewal processes can help model-independent and time-dependent emissions, while neural point processes, such as the neural Hawkes process or transformer-based temporal models, can offer deep learning-driven adaptability in recognizing and responding to complex, irregular event patterns [44]. ML models tailored for irregular time series can enhance CogEW capabilities. Time-aware recurrent neural networks, like temporal LSTMs, can incorporate elapsed time to improve signal classification and tracking [45]. Neural ordinary differential equations (ODEs) and latent ODEs [46]

can help with estimating latent trajectories from sparse, incomplete emissions, providing predictive power in dynamic environments. Graph neural networks can be effective when structuring EW data [47].

For missing or sparse data, imputation and resampling techniques can play a critical role in reconstructing signals [48]. In addition, Gaussian processes can provide uncertainty-aware interpolation [49], while adaptive binning and kernel smoothing [50] can aggregate emissions into structured time windows for analysis.

Detecting significant shifts in signal emissions is crucial for adaptive responses in EW. Bayesian online change-point detection can help in identifying sudden changes in EA patterns or jamming strategies [51]. Burst detection methods, such as Kleinberg's algorithm [52], can reveal clusters of activity in radar emissions or communication networks. Interpulse interval analysis can aid in distinguishing pulse repetition frequency variations, such as staggered or jittered emissions [53]. Time-frequency analysis methods, including wavelet transforms, Hilbert-Huang transform, and short-time Fourier transform, may offer adaptive ways to extract intelligence from nonstationary electromagnetic signals [54].

By leveraging these traditional and ML-enhanced advanced techniques, Cog-EW systems can better interpret irregular emissions, detect anomalies, and adapt to dynamic adversarial tactics.

4.2 Emitter Classification and Characterization

A foundational component of ES is to understand what emitters are in the environment, and what they are doing. Emitter *classification* identifies known categories of the signal, such as modulation classification or platform type. In AI, classification refers to *any* discrete category or label; thus the EW concept of *specific emitter identification (SEI)* is also a form of classification. *Characterization*, on the other hand, refers to capturing behaviors of the signal environment, without necessarily placing a named label on the behavior. A system might for example recognize that a signal is loud, or is Gaussian, or repeats over a given time interval.

Characterization and classification can feed each other, in the sense that a given behavior may support a given classification, or that a given classification may indicate a need to search for a particular behavior characteristic. A variety of challenges can impact the accuracy of emitter classification and characterization:

- *Dynamic properties*, where signal types change over time, require emitter characterization and long-term pattern analysis.
- *Spoofing*, wherein cognitive emitters replay signals to mask their identity, can be mitigated with RF fingerprints that capture microimperfections unique to the emitter.

- *Unknown emitters*, for which there is no a priori training data, usually have known behavior characteristics such as canonical modulation, frequency, pulse parameters, but combine them in unexpected ways or use them for different purposes. Open-set classification techniques recognize and cluster these novel feature groups to construct new classes on the fly.

4.2.1 Feature Engineering and Behavior Characterization

Behavior characterization describes the signal environment, including *features* such as instantaneous energy, frequency, scattering, and repetition patterns, and their *probabilities*. Traditionally, *feature engineering* is computed via a suite of a priori modules, corresponding to characteristics that experts know to be relevant for the expected signals. These mature signal intelligence approaches can be very accurate, even at low signal-to-noise (SNR) ratios, because they rely on underlying immutable properties of the domain, such as physics of signal propagation or kinematics of the targets. However, classical signal processing approaches tend to be brittle in a complex problem domain, and typically cannot incorporate novel emitters or conditions.

Traditional methods use statistical analysis of features such as pulse width, pulse repetition interval, and frequency, and map them to a database of known signals. These may be *channel-specific*, such as channel impulse response, or *transmitter-specific*, such as signal encoding. They may be *time-varying*, or *time-independent*. Fractal features can provide a deeper understanding of the properties of the signal [55–57].

DeepNet approaches can identify *latent* features in the signal environment when trained with sufficient data, potentially capturing characteristics at a much higher fidelity than hand-generated computations. These approaches can capture behavior characteristics that are hard to model with explicit mathematical expressions, or that vary significantly with environmental conditions. They also tend to be more effective when the domain is partially observable (i.e., when known observables $\mathbf{o}_n$ cannot be measured temporarily, or for unobservable features $\mathbf{z}$).

Hand-generated features are a better choice when little training data is available, or when the known model closely captures the empirical performance (Section 3.3). Features can also be pulled from non-RF data sources, such as topography [58], road networks [59], and weather conditions [60]. Section 4.4 discusses some approaches to data fusion.

Combining approaches is an effective way to accelerate learning accuracy. Radio signal strength, for example, generally follows the inverse square law with distance, but is impacted by factors such as antenna gain, multipath, and fading

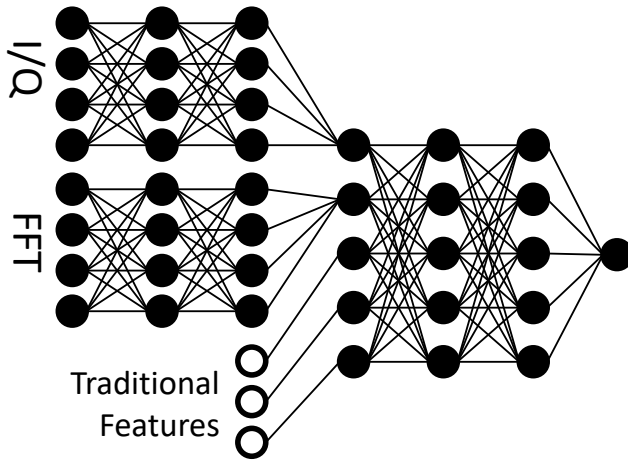


Figure 4.2 A possible hybrid architecture uses independent DeepNets to analyze I/Q data and FFTs, then hierarchically combines the generated features with the traditional features.

(Callout 3.1). One way to create a hybrid model in RF uses both traditional features and DeepNets. The DeepNets would learn the latent features, while the traditional features would include physics, human expertise, and other metadata. Figure 4.2 illustrates a concept of how the overall network could be constructed, combining I/Q data with a fast Fourier transform (FFT) and traditional features.

Feature engineering should focus on those that lead to better assessment of the environment and thus actionable inferences.

For example, features that indicate a loss of performance for EP BDA include (1) reduced comms network throughput that suggests changing to a more robust modulation technique, for example switching from QAM64 to QPSK, or (2) a loss of radar track indicated by cessation of track waveforms or detection of acquisition waveforms suggests increasing the beam revisit rate or changing the waveform, such as longer pulse widths. These engineered features contain actionable information that can be used to guide decision making. Section 7.1.1 mentions similar considerations in the context of inferred BDA.

Metalearning can transform features [61, 62], select features [63, 64], and reduce dimensionality of the task [65, 66]. An interesting characterization problem

determines the degree of sophistication of observed emitters [67–70]. Extended observation of the emitter computes its adaptivity and behavior model, thus supporting inference of its strategy and probable next actions (Section 4.7).

4.2.2 Waveform Classification

One of the most popular current uses for ML techniques is in the area of waveform classification. Older work frequently used SVMs for radar emitter signal recognition [71, 72], radar antenna scan type recognition for EW applications [73], and automatic digital modulation recognition [74, 75]. Newer work is migrating to using deep networks, largely because of their ability to identify latent features of the RF signals [76–78]. Examples include classifying radio emitters using CNNs [79–81], or Siamese ANNs [82], classifying radar emitters using CNNs [83], and capturing temporal patterns with RNNs [84]. GANs are effective methods for augmenting the data, so that a model can be trained with fewer examples [85, 86].

Unfortunately, many RF researchers resort to using images, such as spectrograms, for their models. The argument is that object detection in images is a very mature field with robust foundational models, while the same is not true for I/Q data. However, rendering I/Q into images adds latency and reduces information, making these image-based approaches inherently unsuitable for EW.

In 2018, the US Army Rapid Capabilities Office (RCO)¹ conducted a Signal Classification Challenge to promote advanced research to explore deep learning algorithms for blind signal classification and characterization [87]. While accuracies demonstrated by this challenge were not ready for fielding, research exploded, and perhaps more importantly, the importance of data quality and diversity has been recognized by the community. Hybrid ML approaches may solve some of the open problems. Figure 4.2 illustrates one possible hybrid architecture; Section 7.3.3.1 describes a different structure appropriate when a priori training data is abundant, but the system needs to rapidly update during a mission. Notably, the winning team used a hybrid approach, combining traditional features, ensemble learning, and DeepNets [88, 89].

Algorithm 4.1 outlines the basic steps required to build a radar class detector. The train/test cycle is the same for every model type (Section 3.8). This algorithm uses a naïve Bayes classifier; other ML models may yield more accurate results. *Step 1.* Create a set of *emitter descriptor words (EDWs)* from ground truth observed parameters including the duty cycle, the frequency, the pulse repetition interval, and the pulse width. Assign each EDW its associated class. Most ML algorithms expect

1. The Army RCO has since been renamed the Army Rapid Capabilities and Critical Technologies Office (RCCTO).

Algorithm 4.1 To build a radar class detector, train a model on some of the data, and validate the model on the rest of the data. Every ML model is trained and tested with the same process (Section 3.8).

Step 1: Load the dataset of known radars. Split the dataset into a *train* and *test* set (approximately 80%/20%).

```
from sklearn.model_selection import train_test_split
radars = pandas.read_csv( 'radar-classes.csv' )
X = radars.drop(columns=['Class'])
y = radars[ 'Class' ]
xTrain,xTest,yTrain,yTest = train_test_split( X, y,
                                              test_size=0.20)
# xTrain, meansPerCol = cleanNaNs( xTrain )
# xTest = replaceNaNs( xTest, meansPerCol )
```

Step 2: Train the algorithm. Select a classification algorithm, for example naïve Bayes, and train the model on the training data.

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
trainedGNB = gnb.fit( xTrain, yTrain )
```

Step 3: Validate the results. Evaluate the model with the test data. Compare the predicted classes against the known test classes using a confusion matrix (Section 10.8.2).

```
from sklearn.metrics import confusion_matrix
yPred = trainedGNB.predict( xTest )
accuracy = (yTest==yPred).sum() / len(yTest) * 100
confMtx = confusion_matrix( yTest, yPred )
```

Step 4: Classify new radars and estimate class probabilities. We assume no new radar classes.

```
newRadars = pandas.read_csv( 'newRadars.csv' ) # No 'Class'
yPred = trainedGNB.predict( newRadars )
yProbs = trainedGNB.predict_proba( newRadars )
```

no NaN-values in the data. We therefore replace NaNs for each feature with the non-NaN mean of that feature (Algorithm 2.1). Do not replace NaNs in the complete dataset (i.e., before splitting it for train and test data). Instead, first split the data, and use only the training data to compute the means. Use those mean values to replace the NaNs in the test data. *Step 2.* Train a naïve Bayes algorithm, which computes the probability of each possible class. Using Bayes' rule, the posterior probability

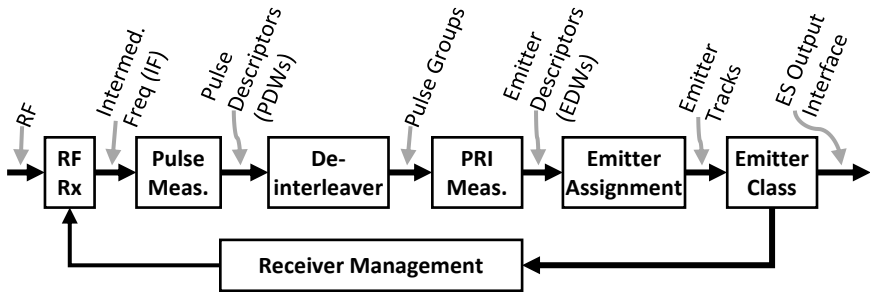


Figure 4.3 The radar classifier is the last step of the common RF processing chain. (Other AI techniques could use the information for further downstream analysis.)

$P(y|x)$ is the probability that the radar whose EDW is x is actually a member of the class y : $P(y|x) = P(x|y) \times P(y) \div P(x)$ where $P(x|y)$ is the likelihood of the EDW given the class, $P(y)$ is the prior probability of the class, and $P(x)$ is the prior probability of the EDW. *Step 3.* Evaluate the model with the test data and generate a confusion matrix (Section 10.8.2). *Step 4.* Classify the new radars. We assume no new radar classes; open-set learning techniques (Section 3.4) should be evaluated with ablation trials (Section 10.9). Figure 4.3 illustrates where the classifier belongs in the RF processing chain.

4.2.3 Specific Emitter Identification

Specific emitter identification (SEI), or *RF fingerprinting*, is the process of uniquely identifying each RF emitter, independent of receiver properties, or how the transmitter changes its transmissions, its mobility, environmental factors. In older work, Chen et al. [90] use clustering and infinite hidden Markov random fields to handle both time-dependent and time-independent features for SEI. Nguyen et al. [91] use Gaussian mixture models to cluster and then classify signals.

A lot of current work feeds I/Q samples to a DeepNet, moving away from hand-generated features. These models often do not perform channel estimation, and do not assume prior knowledge of the system. Example includes feeding the I/Q samples to the CNN [92–95], adding a finite input response [96], structural causal models [97], and using a wavelet transform to handle temporal patterns [93]. RiftNet is a DeepNet approach that scales to 10,000 devices [78]. Several surveys cover DeepNet approaches [78, 98–100]. Figure 10.7 shows results from a specific emitter task with twelve emitters on six radios (i.e., two transmitters per radio). The models were trained on raw I/Q data, and the two DeepNet architectures achieved

different results. The most common misclassification error was confusing the two emitters on the same radio. A key consideration is how the model performs as a function of how many observations, and how many I/Q samples per observation, are needed before accuracy is sufficient for the task [93].

Existing work has not yet achieved the accuracy of traditional approaches for highly dense, complex real-world environments, particularly for low signal strengths. Hybrid approaches will likely yield interesting results in the future.

4.3 Performance Estimation

One can also use ML techniques to capture the relationship between observations o and performance metrics m , usually as regression models: $m = f(o, s)$. Section 2.3 presents possible metrics, organized by effectiveness and cost. Example techniques include SVMs [101], ANNs [102–107], Bayesian networks [108], and Markov decision processes (MDPs) [109, 110]. Kulin et al. survey ML-based performance estimation for wireless networks [111]. In radar systems, common metrics are resource use and tracking accuracy [112]. Jamming effectiveness is another useful feature to predict. Because these approaches measure system performance, many of them are incorporated into RL frameworks, where the learned model is then used to control the decision process (see Chapter 7).

Performance estimation learns the effectiveness of each technique, using the set of observables that describe the environment (see Section 2.1.1). As an example, consider Figure 4.4, using synthetic, clean data. There are three observables, each with two core values: o_1 centers around 0.0 or 1.0, o_2 centers around 2.0 or 3.0, and o_3 centers around 4.0 or 5.0. This configuration yields a total of eight RF environments. Table 4.1 outlines notional metric values for each of two controllables, c_0 and c_1 . The best strategy for each environment is highlighted; note that the *best* strategy for a given environment may not be a *good* strategy. Traditionally, we collect these values using controlled experiments in a lab and write rules such as *if* $((o_1 \approx 0.0) \text{ and } (o_2 \approx 2.0) \text{ and } (o_3 \approx 4.0))$, *then use* s_3 .

However, this approach does not scale to many observables, noisy observables, many controllables, noisy metrics, or novel environments. The goal of ML for estimating strategy performance is to fill in the values of this table, based on real-world execution experience. Some initial values may be provided by lab experiments or engineering analysis; these samples become the initial training data. Some of the cell values may change, such as when components are upgraded or fail. Some of the cell values may not be easy to collect in the lab, especially when there are many controllables. Novel RF environments become a new row in the table, with

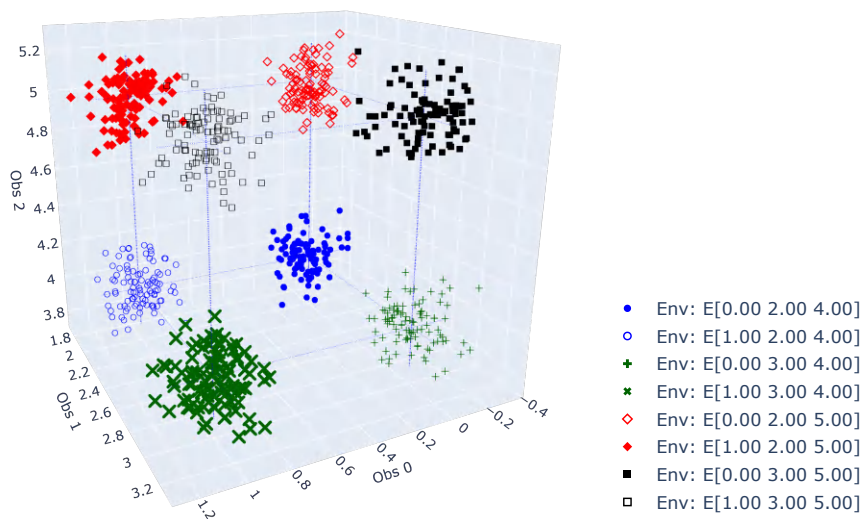


Figure 4.4 Three observables, each with two core values, yields eight RF environments.

Table 4.1 Synthetic Example Showing Performance Metrics for Eight RF Environments and Four Strategies				
Environment	$c_0=0, c_1=2$ $s_0=01$	$c_0=1, c_1=2$ $s_1=12$	$c_0=0, c_1=3$ $s_2=03$	$c_0=1, c_1=3$ $s_3=13$
E[0.0 2.0 4.0]	0.116	0.022	0.161	0.704
E[1.0 2.0 4.0]	0.430	0.674	0.018	0.318
E[0.0 3.0 4.0]	0.229	0.594	0.463	0.081
E[1.0 3.0 4.0]	0.532	0.075	0.940	0.054
E[0.0 2.0 5.0]	0.303	0.244	0.434	0.037
E[1.0 2.0 5.0]	0.602	1.000	0.011	0.164
E[0.0 3.0 5.0]	0.225	0.640	0.207	0.023
E[1.0 3.0 5.0]	0.003	0.157	0.022	0.000

no initial cell values. These samples can be added to the training data incrementally during the run (Section 7.3), and then used in future engagements.

Figure 4.5 presents example results from estimating the performance of different controllables against different RF environments using real data. Only

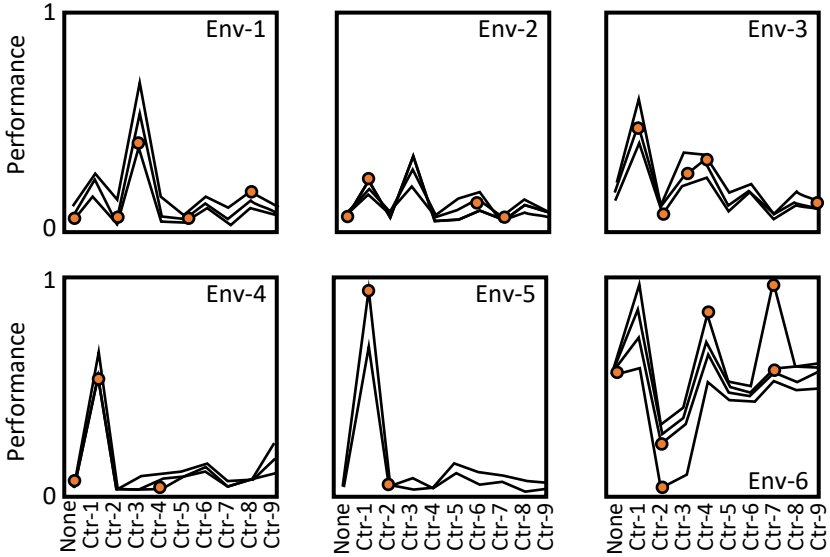


Figure 4.5 The ML model f_k learns to estimate the performance m_k of different controllables c against different environments. (Real data from Example 1.1. Dots represent observations; all other points are estimated by the model. Figure 8.6 shows plots like these in a 3D layout.)

the points marked with a dot are known a priori; the values would be stored in a table similar to Table 4.1 or Table 10.9. The ML model estimates other performance values based on observable features of the different environments. This example uses an SVM to model the performance space, and traditional features to describe the environments. Each observation of an environment is described by a corresponding set of observables o . Each use of a strategy s in that environment creates feedback of the metric m_k , i.e., an example of $m_k = f_k(o, c)$, and marked with a dot on the graphic. The trained model then estimates the performance of all strategies against all environments, generalizing from the known cases to other conditions. In Env-5, c_1 is the best strategy, and there is supporting data. In Env-2, the model estimates c_3 as best strategy, but even the best strategy might not have particularly good performance. In Env-6, c_7 has supporting data that shows wide variation in expected performance. Because the estimated performance profiles for Env-4 and Env-5 are similar, the two environments likely have similar observables and behavior characteristics. Figure 7.6 extends this example to show how the model can incrementally learn during a mission.

Table 5.1 outlines several techniques to mitigate a tone jammer. The learner can model both the associated benefits and associated costs. In this case, BER is a good metric to evaluate performance. ML can also predict performance in terms of jamming effectiveness [113, 114], learning which EA technique to use for which observed emitter behaviors. Performance feedback on these predictions come from EW BDA (Section 7.1.1).

4.4 Multisensor Data Fusion

For optimal situation awareness, decision makers need to interpret data from multiple sources (e.g., radars, unmanned aerial, ground, or underwater systems, ships, space assets, fighter jets, antennas, sensor networks). To support this capability, a CogEW system must perform data fusion.

Data fusion is the process of data integration from multiple sources to produce more accurate inferences. Illustrated in Figure 4.6, multisensor data fusion compares, correlates, and combines data from multiple sources of different types to improve accuracy and make more specific inferences than can be achieved by a single sensor alone [115–117]. Accurate data fusion is challenging for a variety of reasons [118, 119], including:

- *Data quality issues*, such as measurement uncertainty, sensor calibration, and bias can skew accuracy and confidence. Data curation (Section 8.1.1) helps mitigate this issue.

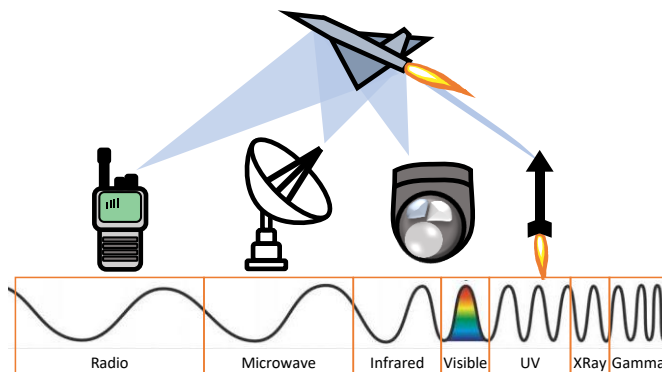


Figure 4.6 Data fusion combines information from multiple sources of different types to produce better inferences.

- *Conflicting data* can produce counterintuitive results. Traceability (Section 8.1.3) and managing information uncertainty (Section 6.1.4) facilitate correct decisions.
- *Data association* connects the dots between concepts, and if overeager, can produce ridiculous results. Causal modeling (Section 4.6) techniques help identify correct relationships.
- *High-dimensional data* makes it difficult to find good items to fuse, akin to looking for the needle in the haystack. Section 3.5 describes some techniques for dimensionality reduction.
- *Operational timing* can cause relevant information to be delayed or arrive out-of-sequence. Metadata is *crucial* here: Time is probably the most important feature to record faithfully so the fusion engine doesn't fuse inconsistent items. Section 4.4.3 delves more into this issue.

In all cases, metadata tagging (Section 8.2.2) tracks the upstream sources of data, allowing downstream consumer modules to compensate for error.

Data fusion techniques can be grouped into three nonexclusive categories: (1) data association, (2) state estimation, and (3) decision fusion [119]. The Joint Directors of Laboratories / Data Fusion Information Group (JDL/DFIG) model comprises seven levels of data fusion shown in Figure 4.7 [120, 121]. Each level increases the breadth of data and scope of analysis. Some of the JDL/DFIG model's benefits include exploration of situation awareness, user refinement, and mission management [120]. The JDL/DFIG model has also been successfully used for the visualization of the data fusion process, discussion facilitation, and common understanding [121], and systems-level data fusion design [120, 122]. The latest advances in the data fusion field are captured by the International Society of Information Fusion (ISIF) [123].

4.4.1 Data Fusion Approaches

The book *Sensor and Data Fusion* [124] presents a comprehensive compilation of research papers on various topics related to sensor and data fusion. Some of the addressed problems include target tracking air traffic control, remote sensing, anomaly detection and behavior prediction, and sensor networks.

Data fusion techniques have been widely used in multisensor environments with the objective of amalgamating data from different sensors. The advantages of multisensor data fusion predominately include enhancements in data authenticity, reliability, and availability [116, 118, 119, 125–127]. Examples of data authenticity enhancements consist of improved detection, confidence, reliability, and reduction

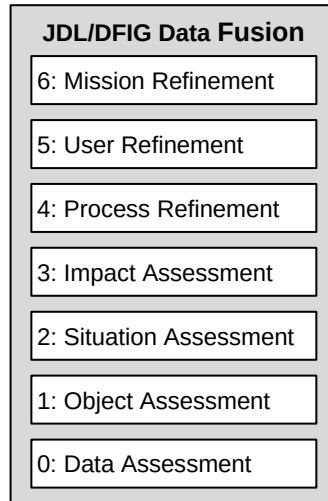


Figure 4.7 The JDL/DFIG model comprises seven levels of data fusion.

in data ambiguity. Data reliability addresses gaps that occur when sensors fail, ensuring continuity in information flow. Data availability enhancements focus on extending spatial and temporal multisensor coverage.

Data fusion helps solve scalability issues in wireless sensor networks that are caused by a large number of sensor nodes, packet collisions, and redundant data transmissions. Data fusion in the routing process (i.e., where sensor data is merged and only the result is forwarded) reduces the number of messages, avoids collisions, and reduces energy consumption [116, 118, 125]. Wireless sensors typically rely on battery power for energy that can be consumed by computation, communication, sensing and idling states [128]. Data fusion in EW combines sensors such as radio, radar, mmWave, visible light, lidar (e.g., [129–131]).

Some interesting applications of multisensor data techniques in EW pull features from other types of data sources. Brandfass et al. [58] develop an application that adds map data to radar. Katsilieris and Charlish use road networks for ground moving target indicator [59]. Zhang et al. use weather data to improve path loss predictions [60]. Levashova et al. present an approach to data fusion to predict terrorist threat activity for applications such as area surveillance, command and control, communications, and access rights [132].

Table 4.2
5G's Location-Based KPIs and Some Related Use Cases

KPI	Description
Position accuracy (PA)	Difference between the user equipment (UE) estimated position and its true position
Speed accuracy (SA)	Difference between the estimated magnitude of the UE's velocity to the true magnitude
Bearing accuracy (BA)	Difference between the UE measured bearing and its true bearing
Latency (L)	Time elapsed between the event that triggers the determination of the position-related data and their availability at the positioning system interface
Time to first fix (TTFF)	Time elapsed between the event triggering the determination of the position-related data and their availability at the positioning system interface
Update rate (UR)	Rate at which the position-related data is generated by localization
Power consumption (PC)	Electrical power (typically in milliwatts) used by localization to produce the position-related data
Energy per fix (EPF)	Electrical energy (typically in millijoules per fix) used by localization to produce the position-related data
System scalability (SS)	Number of devices for which the positioning system can determine the position-related data in a given time unit, and/or for a specific update rate

Using these KPIs, 3GPP outlined several use cases and their potential requirements:

- *First Responders Indoor*: PA: 1 m horizontal, 2m vertical; Availability: 95%; TTFF: 10 sec; L: 1 sec
- *Augmented Reality Outdoor*: PA: 1 - 3 m horizontal, 0.1 - 3m vertical; Velocity: 2 m/s; 10 degrees; Availability: 80%; TTFF: 10 sec; L: 1 sec; UR: 0.1 - 1 sec; PC: low energy
- *Traffic Monitoring and Control Outdoor*: PA: 1 - 3m horizontal, 2.5m vertical; Availability: 95%; UR: 0.1 sec; TTFF: 10 sec; L: 30 msec; antispooofing; antitampering
- *Asset Tracking and Management Outdoor*: PA: 10 - 30m horizontal; Velocity: 5m/s; Availability: 99%; UR: 300 sec - 1 day; antispooofing; antitampering; out of coverage; EPF=20 mJ/fix
- *UAV: (Data Analysis) Outdoor*: PA: 0.1m horizontal, 0.1m vertical; Availability: 99%; TTFF: 10 sec; low energy; antispooofing; antitampering

4.4.2 Example: 5G Data Fusion for Localization

One concrete example of how ML can assist with the multi-intelligence data fusion is 5G localization and the concept of *context awareness*. Context-aware mobile devices such as smart phones may become aware of their location, or attempt to make assumptions about the end user's current situation. Table 4.2 shows 5G's location-based key performance indicators (KPIs) [133, 134] and some related use cases. Note that *localization* in the context of 5G is radar function, for

Table 4.3
ML-Enabled Radio Access Technology (RAT)-Dependent and
RAT-Independent Data Fusion Techniques and Measurements

RAT-Dependent	RAT-Independent
4-4.5G: Cell ID, enhanced cell ID, observed time-difference-of-arrival, uplink time-difference-of-arrival, RF pattern matching [134]	<i>Traditional:</i> GNSS, radio-frequency identification (RFID), terrestrial beacon systems, Bluetooth, wireless local area network, sensors (e.g., RF, acoustic, infrared, radar, laser, inertial, vision tech, barometric), ultra-wideband [134]
<i>Novel:</i> Multipath-assisted localization [137], single-anchor localization with massive millimeter-wave antenna arrays [138], cooperative localization (e.g., device-to-device, vehicle-to-everything) [139]	<i>Novel:</i> Distributed soft information-based localization from heterogeneous sensors plus context [135], such as digital maps, dynamic channel models, geographic information system data, and real-time traffic data

example localizing humans in the environment, inspecting critical infrastructure, or providing aerial maps.

Table 4.3 summarizes different ML-enabled data fusion techniques and measurements that can fulfill the complex localization KPI requirements in 5G scenarios [134–136]. The reliability of 5G localization also requires efficient network strategies such as node prioritization, node activation, and node deployment. Cooperative operation, robustness guarantees, and distributed design of network operation are also important in localization because they affect energy consumption and determine localization accuracy [140].

One of the novel localization approaches for 5G is soft information-based localization from heterogeneous sensors and contextual information. Conventional localization methods rely on *single-value estimates* such as time-of-arrival, observed time-difference-of-arrival, angle-of-arrival, or received signal strength indicator (RSSI). Novel techniques, however, rely on a *set of possible values* rather than on single-distance estimates, for example, soft range information [135]. ML-enabled data fusion approaches include data association, state estimation, decision fusion, classification, prediction/regression, unsupervised ML, dimensionality reduction, and statistical inference and analytics [136].

4.4.3 Distributed Data Fusion

Distributed data fusion is the process of intelligent agents (1) sensing their local environment, (2) communicating with other agents, and (3) collectively trying to infer knowledge about a particular process [141]. Collaboration is a process of sharing information, while coordination is a process of sharing tasks. Distributed data fusion is an important area to consider as it must respect many key constraints imposed by the EW domain, particularly limited communication.

Distributed analysis is a powerful way to reduce data ambiguity and increase accuracy of slower-time conclusions.

Lang [142] surveys work that addresses collective decision making with incomplete knowledge. Makarenko et al. [143] examine distributed data fusion as a distributed inference problem, and apply Shafer-Shenoy and Hugin junction tree algorithms to target tracking. Decentralized estimation increases robustness against node and link failures [144]. The book collection edited by Hall et al. describes a variety of data fusion approaches for distributed network-centric operations [145]. Munir et al. bring these ideas to edge networks [146].

High-quality metadata (Section 8.2.2) is a key requirement of distributed data fusion. The challenge is not accuracy in spectrum per se, but rather the challenge of collecting multiple signals across spectrum on multiple nodes, and aggregating them into a single consistent view of the observed emitters. This step requires consistent signal interpretation and emitter behavior aggregation via sharing, and a very consistent behavior representation. Additionally, provenance and credibility (Section 8.1.3) notably impact data transformation, inferences made, and downstream modules' functionalities.

Collectively, distributed data fusion may change the overall sensitivity of the data. For example, temperature at a singular sensor X may not be sensitive, but the results from the sensor array are.

4.5 Anomaly Detection

Anomaly or outlier detection aims to find patterns in data that do not follow known behavior [147]. The statistics community has been studying anomaly detection in data as early as the 19th century [148], and many anomaly detection techniques exist. Figure 4.8 provides a simple example of anomalies in a two-dimensional dataset. Anomalies in RF can be due to factors such as spurious noise, measurement error, and new observed behaviors.

Chandola et al. [147] distinguish between simple anomalies and complex anomalies. *Simple* or *point* anomalies address individual data instances that are

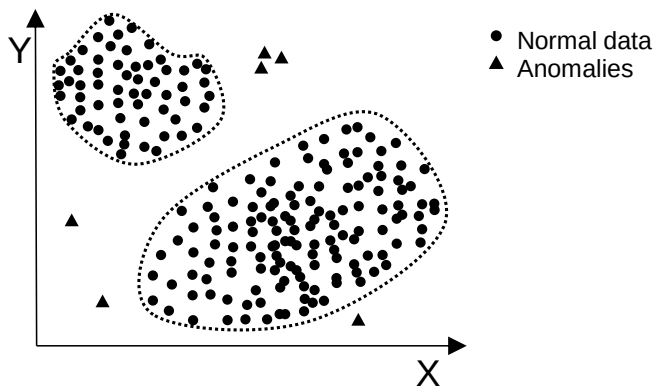


Figure 4.8 Anomaly detection aims to find data that does not follow conventional behavior. (Previously seen well-behaved data is depicted by black circles, while anomalies are shown as triangles.)

anomalous with respect to the rest of the data (e.g., a radar deliberately using a different frequency or making a spurious error). The majority of anomaly detection research focuses on point anomalies.

Complex anomalies can be further subdivided into contextual and collective anomalies. A *contextual* or *conditional* anomaly is an anomalous data instance in a specific context but not otherwise [149]. For example, seeing a known emitter in an unexpected location. A *collective* anomaly is a collection of related data instances anomalous with respect to the entire dataset. Individual data instances in a collective anomaly may not be anomalies independently, but their combined occurrence is anomalous. For example, when a radar uses a particular pulse burst (e.g., defined parametrics, RF, pulse descriptor word) with a specific scan pattern normally used for target acquisition (say, box search), and instead uses it for tracking.

Anomaly detection techniques rely on ML techniques such as classification, nearest neighbor, clustering, statistics, information theory, and spectral theory [147, 150–157]. Common classification-based anomaly detection techniques often involve SVMs, Bayesian networks, rule-based methodologies, and DeepNets, particularly autoencoders. Anomaly detection is commonly employed for network intrusion detection and categorization using various algorithms including feed-forward ANNs [158], RNNs [159], autoencoders [160], variational autoencoders [161], SVMs [162, 163], linear regression and random forest for multicloud environments [164], and restricted Boltzmann machines [165]. Dästner et al. explore big

data trajectory analysis and ML for radar surveillance, focusing on anomaly detection, target classification, and adaptive signal processing [166]. They discuss data preprocessing, smoothing techniques, and sliding windows, highlighting their role in enhancing detection and tracking. Bauw et al. applied semi-supervised anomaly detection to identify unusual radar targets, focusing on ML techniques for high-resolution range profiles to enhance anomaly detection in radar surveillance [167]. Ruff et al. introduced deep semi-supervised anomaly detection (Deep SAD) that leverages both labeled and unlabeled data to improve anomaly detection by refining decision boundaries [168].

An interesting use for anomaly detection at runtime is detecting evolving normal data where the current (normal) state is not sufficiently representative of the future state. This idea is known as *concept drift* in the AI literature because the current concept of normal has drifted away from the concept at the beginning of the mission. Concept drift approaches include statistical tests, adaptive learning, metalearning, and hybrid approaches [169–172]. One way to handle concept drift is to estimate the properties of statistical distribution models that capture the range of observations. When the distribution(s) begin to separate with a weight (support) of “anomalous” signals, then the distribution models can be split to capture the statistics of all modes observed. This idea lies behind much of the open-set classification research (Section 3.4). In this way, one does not lose the history of known behavior and get distracted by the new anomalies or fail to adapt to new behavior. It is not a precise solution since there are trade-offs on when to spawn a new distributional model component but having a persistent memory as well as mechanisms to learn new behavior is critical.

4.6 Causal Relationships

Causal inference attempts to extract causal relationships from the observed data, typically when there is no explicit way to perform an experiment to validate a hypothesis [173, 174]. The causal hierarchy groups models according to the kinds of questions that each layer is capable of answering:

- *Association*. Observing. How does my belief in a concept y depend on an observation x : $P(y|x)$?
- *Intervention*. Acting. How does doing an action a impact the outcome y : $P(y|\text{do}(a))$
- *Counterfactuals*. Imagining. What would have happened under other circumstances x' : $P(y|x')$

Causal models are commonly used for analysis of medical, warfare, and other social settings. Effective for cybersecurity [175], they have also been used for radar

and air traffic safety [176]. The goal is to detect sequences of events that lead to likely outcomes, such as a precursor “red flag” of imminent danger. Chernozhukov et al. present a variety of causal inference methods [177].

In RF, Martin and Chang use probabilistic structural causal models and multi-attribute utility theory for dynamic spectrum access (DSA) situation awareness and decision making [178]. The DSA causal inference system develops and maintains situation awareness based on spectral occupancy, spectrum user location estimates, and path loss characterizations. The authors derive the relationship between situation assessment uncertainty and DSA system performance yielding path loss estimates that enable DSA systems to operate with acceptable levels of risk. Structural causal models also help with SEI [97]. Causal analysis can identify vulnerabilities in 5G [179].

The most common use of causal analysis in RF is for network topology estimation. For example, graph neural networks can structure EW data as interconnected nodes for mapping adversarial communication networks or performing coordinated signal emissions [47]. Testi and Giergetti use causal analysis for blind source separation [41]. Cousins et al. [180] used statistical pattern analysis to build a conversation probability matrix. The matrix estimates who is talking to whom, even when the packets are encrypted or attempts are made to mask the timing. The approach detects ingress/egress nodes and key nodes, such as the hub of a hub-and-spoke network. For example, terrorists may all use the same Iridium satellite phone infrastructure, but each terrorist cell is independent. Figure 4.9 illustrates a network where the central vehicle and the satellite are key nodes in the network.

Other techniques that analyze likely patterns include Granger causality [181, 182], hysteresis [183], precursor detection [184], n -gram analysis to determine common phrases, multivariate Hawkes processes [185, 186], and HMMs to determine likely transitions [68]. Intent recognition takes the pattern analysis “up” one level, determining full grammars and hierarchical structure (Section 4.7).

4.7 Intent Recognition

Plan recognition, activity recognition, and intent recognition all make inferences about another actor by observing their interactions with the environment and each other [187–189]. In a CogEW system, such techniques can be applied to observe and understand the plans, activities, and goals of all the participants in the theater, including adversaries, coalition partners, and human EWOs. Such inference actually requires three separate tasks.

First, *activity recognition* identifies the lowest-level actions being executed by an agent. As such, it focuses on the problem of processing raw, noisy, frequently

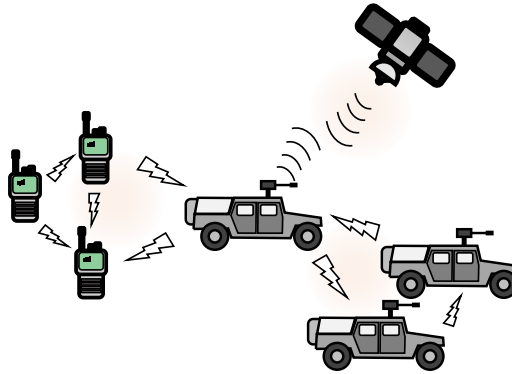


Figure 4.9 Statistical pattern analysis can detect causal relationships such as subnetworks and key nodes.

continuous data into high-confidence discrete events (often with much longer duration). For example, signal-processing algorithms remove noise from a continuous stream of sensor values to provide a clear unambiguous identification of the event that is happening.

Second, *goal recognition* identifies the highest level desired end or goal states of the observed agent. Such a process asks at a high level what the agent is trying to accomplish. In the abstract, activity and goal recognition can be seen as the same problem and may be amenable to similar algorithms. That is, both of these problems are instances of labeling: Given a sequence of observations, both of these tasks produce a single label to identify either the activity being performed or the goal being achieved.

Finally, and in contrast, rather than producing a single label for a sequence of activities, *plan recognition* produces a structured (hierarchical) representation that relates observed activities to the desired goal states of the agent [190]. Thus, a plan recognition system takes as input the sequences of events identified by activity recognition. It organizes the sequence of observations into structured representations that captures the causal relations for the particular sequence of recognized activities and their final objective. This inference of the final objective means goal recognition is frequently performed as a byproduct of plan recognition. This process also usually requires not only capturing the relations between the observed activities, but also hypothesizing future activities that the agent will have to undertake if

the algorithm is correct in identifying the actor's plan and goal. As such, it can enable prediction of future actions at multiple levels of abstraction. These predictions can then be used as a comparison feature for EW BDA (Section 7.1.1).

Algorithms for plan recognition fall into two categories [191]:

- Plan recognition as *planning* describes observed actions/activities as functions from states to states and captures the changes in the world state caused by their execution. As such, plan recognition is a world-state-based search for a sequence of actions that will provably carry the actor from a known initial state of the world to a goal state of the world.
- Plan recognition as *parsing* views the problem as a search in the space of possible plans. Such systems explicitly represent the set of possible plans (often using a formal grammar similar to those used in natural language processing) and then search this space for the plan that is most consistent with the input observations. This process constructs hierarchical structures to capture the plans.

As an example, the *probabilistic hostile agent task tracker (PHATT)* [192] used a parsing-based approach for diverse domains, including network intrusion detection [193] and insider threat/misuse detection. To represent the plans to be recognized, it used a probabilistic context-free tree grammar most closely associated with *Greibach normal form* grammars. These grammars are commonly used in compilers for programming languages. While PHATT was very effective, its wide applicability was limited by its computational costs. It made early commitments to high-level goals and required a complete search of the space of consistent plans. For example, imagine observing an airplane take off, PHATT would immediately consider all possible flight paths to all possible destinations, no matter how wildly unlikely. This problem prevented its scaling to large, real-world deployment.

Geib's more recent work on the LEXrec component of the *Engine for LEXicalized Intent Reasoning (ELEXIR)* [194] addressed this limitation. Instead of tree grammars, LEXrec uses probabilistic *combinatory categorial grammars* from state-of-the-art natural language processing research. LEXrec supports recognizing multiple concurrent and interleaved plans, partially ordered plans, and partial observability in the domain.

In addition, LEXrec leverages Monte Carlo tree search—the algorithm behind AlphaGo [195]—to perform heuristic search of the space of possible plans in an “anytime” fashion (Section 5.4). This approach has allowed LEXrec to process thousands of observations in minutes in the task of recognizing grey zone operations. Further, the grammars for LEXrec can be configured to further accelerate

processing, and thus promising for large-scale deployment on real-world problems. LEXrec's search can be modified so that more likely explanations for the observations are searched first. In such a case, high probability explanations that include hostile plans will be explored before lower-probability benign explanations.

Similar grammar-based systems have been successfully applied to the multifunction radar problem [196, 197]. Further, the plan recognition community has developed both supervised and unsupervised learning techniques to learn new grammars (potential plans) based on sequences of observations [198–200].

A system like LEXrec could take as input a large number of categorized radar tracks and observations from diverse and distributed multifunction radars. On the basis of these observations, and a grammar capturing both the potential individual and joint plans that the targets could be following, the system could build a set of hypothesized plans and goals (both individual and team) and establish conditional probabilities over these possible plans. Such plans could capture complex radar modes and temporally extended plans. Further, the system could predict the future course of the targets, not just on the basis of their immediate trajectory, but based on a higher-level understanding of the target's goals.

Adversarial intent recognition is a challenge for plan recognition algorithms because the actor(s) attempt to hide or disguise their actions or deliberately change their patterns of activity to make recognition harder. However, given a model of what the adversary can do, the process is no different for adversarial actions than for benign actions. The algorithm must be agnostic about whether an action is benign or deceptive until an additional action differentiates the two. Recent work has started to incorporate game-theoretic approaches to recognize plans where the observed agent actively tries to avoid detection [201–203]. The adversarial plan recognition task consists of inferring a probability distribution over the set of possible adversarial plans, and a strategy formulation task selects the most appropriate response by computing a Nash equilibrium. Section 6.2 discusses other uses for game theory.

4.8 Conclusion

Electronic support is at the heart of every effective EW system. EP and EA performance is directly tied to the quality of the ES. ES analyzes the environment and creates the observables that drive decision making. This introduced a sampling of situation assessment techniques that address some of the particular challenges of an ES system.

Classical ML techniques have had a presence in EW systems for years. ML is exploding with new insights and results, driven primarily by developments in

the deep learning community. The RF community can exploit these remarkable breakthroughs to address similar problems in ES.

Embracing deep learning however, does not mean that classical ML techniques, RF signal processing, or expert knowledge of the problem should be dropped. Use the approach most suited for the task at hand that meets system goals and requirements.

References

- [1] Howland, P., S. Farquhar, and B. Madahar, "Spectrum Situational Awareness Capability: The Military Need and Potential Implementation Issues," Defence Science, and Technology Lab Malvern (United Kingdom), Tech. Rep., 2006. Online: <https://tinyurl.com/howland-ssa>.
- [2] Gurbuz, S. Z., et al., "An Overview of Cognitive Radar: Past, Present, and Future," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 34, No. 12, 2019. DOI: 10.1109/MAES.2019.2953762.
- [3] Rojhani, N., and G. Shaker, "Comprehensive Review: Effectiveness of MIMO and Beamforming Technologies in Detecting Low RCS UAVs," *Remote Sensing*, Vol. 16, No. 6, 2024. DOI: 10.3390/rs16061016.
- [4] Song, Y., et al., "Joint Waveform Design and Resource Allocation Strategy for Cognitive Radar Target Situation Awareness," *IET Radar, Sonar & Navigation*, Vol. 18, No. 8, 2024. DOI: 10.1049/rsn2.12575.
- [5] Kulkarni, S., et al., "A Comprehensive Review of Direction of Arrival (DoA) Estimation Techniques and Algorithms," *Journal of Electronics and Electrical Engineering*, 2025. DOI: 10.37256/jee.4120255708.
- [6] Huang, Z., et al., *RadDet: A Wideband Dataset for Real-Time Radar Spectrum Detection*, 2025. Online: <https://arxiv.org/abs/2501.10407>.
- [7] Zhang, G., et al., "Power Allocation Scheme for Multi-Static Radar to Stably Track Self-Defense Jammers," *Remote Sensing*, Vol. 16, No. 15, 2024. DOI: 10.3390/rs16152699.
- [8] Horstmann, S., D. Ramírez, and P. J. Schreier, "Multistatic Passive Detection of Cyclostationary Signals," in *Acoustics, Speech and Signal Processing*, 2024. DOI: 10.1109/ICASSP48485.2024.10447335.
- [9] Nouredine El-haryqy, Z. M., and Y. Zouine, "A Review of Deep Learning Techniques for Enhancing Spectrum Sensing and Prediction in Cognitive Radio Systems: Approaches, Datasets, and Challenges," *International Journal of Computers and Applications*, Vol. 46, No. 12, 2024. DOI: 10.1080/1206212X.2024.2414042.
- [10] Cristallini, D., and D. O'Hagan, (Eds.), *Passive Radars on Moving Platforms*. Stevenage, UK: The Institution of Engineering and Technology, 2023. DOI: 10.1049/sbra536e.
- [11] Krückemeier, M., *A Mobile Passive Radar System*. Göttingen, Germany: Cuvillier Verlag, 2022. Online: <https://cuvillier.de/en/shop/publications/8694-a-mobile-passive-radar-system>.
- [12] Maksymiuk, R., et al., "5G Network-Based Passive Radar for Drone Detection," in *International Radar Symposium*, 2023. DOI: 10.23919/IRS57608.2023.10172437.
- [13] Maksymiuk, R., et al., "5G-Based Passive Radar on a Moving Platform - Detection and Imaging," *IET Radar, Sonar & Navigation*, Vol. 18, No. 12, 2024. DOI: 10.1049/rsn2.12559.

- [14] Hou, E., R. Greenwood, and P. Kumar, "Machine Learning Models for Improved Tracking from Range-Doppler Map Images," in *FUSION*, 2024. DOI: 10.23919/FUSION59988.2024.10706353.
- [15] Xie, N., et al., "Doppler Shift Estimation Method for Frequency Diverse Array Radar Based on Graph Signal Processing," *Remote Sensing*, Vol. 17, No. 5, 2025. DOI: 10.3390/rs17050765.
- [16] Yigit, E., S. Demirci, and U. Özkaya, "Recognizing SAR Images from ISAR Training Data," in *Applied Engineering and Natural Sciences*, Konya, Turkey, 2021. Online: https://www.researchgate.net/publication/356475143_Recognizing_SAR_Images_from_ISAR_Training_Data.
- [17] Lang, P., et al., *A Comprehensive Survey of Machine Learning Applied to Radar Signal Processing*, 2020. DOI: 10.48550/arXiv.2009.13702.
- [18] Elgamoudi, A., et al., "A Survey for Recent Techniques and Algorithms of Geolocation and Target Tracking in Wireless and Satellite Systems," *Applied Sciences*, Vol. 11, No. 13, 2021. DOI: 10.3390/app11136079.
- [19] Chen, H., L. Cao, and Y. Yue, "TDOA/AOA Hybrid Localization Based on Improved Dandelion Optimization Algorithm for Mobile Location Estimation Under NLOS Simulation Environment," *Wireless Personal Communications*, Vol. 131, 2023. DOI: 10.1007/s11277-023-10578-y.
- [20] Yu, W., et al., "A Deep Learning Algorithm for Joint Direct Tracking and Classification of Manoeuvring Sources," *IET Radar, Sonar & Navigation*, Vol. 16, No. 7, 2022. DOI: 10.1049/rsn2.12254.
- [21] Zhang, Y., et al., "A Multi-Active and Multi-Passive Sensor Fusion Algorithm for Multi-Target Tracking in Dense Group Clutter Environments," *Remote Sensing*, Vol. 16, No. 22, 2024. DOI: 10.3390/rs16224120.
- [22] Lee, C.-h., and W. Wolf, "Blind Signal Separation for Cognitive Radio," *Journal of Signal Processing Systems*, Vol. 63, No. 1, 2011. DOI: 10.1007/s11265-009-0400-1.
- [23] Luo, Z., C. Li, and L. Zhu, "A Comprehensive Survey on Blind Source Separation for Wireless Adaptive Processing: Principles, Perspectives, Challenges and New Research Directions," *IEEE Access*, Vol. 6, 2018. DOI: 10.1109/ACCESS.2018.2879380.
- [24] Wang, J., et al., "A Novel Pulse Stream Representation and Affinity Measurement Strategy for Radar Recognition With Distribution Pattern Mining," *IEEE Transactions on Instrumentation and Measurement*, Vol. 74, 2025. DOI: 10.1109/TIM.2024.3500053.
- [25] Lesieur, L., et al., "An Overview and Classification of Machine Learning Approaches for Radar Signal Deinterleaving," *IEEE Access*, Vol. 13, 2025. DOI: 10.1109/ACCESS.2025.3539589.
- [26] Peng, Y., et al., "Radar Emitter Identification Based on Co-clustering and Transfer Learning," in *Industrial Electronics and Applications*, 2021. DOI: 10.1109/ICIEA51954.2021.9516285.
- [27] Dadgarnia, A., and M. T. Sadeghi, "A Novel Method of Deinterleaving Radar Pulse Sequences Based on a Modified DBSCAN algorithm," *China Communications*, Vol. 20, No. 2, 2023. DOI: 10.23919/JCC.2023.02.012.
- [28] Mu, H., J. Gu, and Y. Zhao, "A Deinterleaving Method for Mixed Pulse Signals in Complex Electromagnetic Environment," in *Control, Automation and Information Sciences*, IEEE, 2019. DOI: 10.1109/ICCAIS46528.2019.9074573.
- [29] Hedvall, P., "Out-of-distribution Recognition and Classification of Time-Series Pulsed Radar Signals," M.S. thesis, KTH, Royal Institute of Technology, Stockholm, Sweden, 2022.

- [30] Guo, Q., et al., "Deinterleaving of Radar Pulse Based on Implicit Feature," *Journal of Systems Engineering and Electronics*, Vol. 34, No. 6, 2023. DOI: 10.23919/JSEE.2023.000032.
- [31] Liu, Z.-M., and P. S. Yu, "Classification, Denoising, and Deinterleaving of Pulse Streams with Recurrent Neural Networks," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 55, No. 4, 2019. DOI: 10.1109/TAES.2018.2874139.
- [32] Wen, J., et al., "An Improved De-interleaving Algorithm of Radar Pulses Based on SOFM with Self-adaptive Network Topology," *Journal of Systems Engineering and Electronics*, Vol. 31, No. 4, 2020. DOI: 10.23919/JSEE.2020.000046.
- [33] Li, Z., et al., "A Spectral Clustering Sorting Method for Radar Signals Based on SOM Anchor Extraction And Graph Fusion," in *Signal, Information and Data Processing*, 2024. DOI: 10.1109/ICSIDP62679.2024.10868666.
- [34] Li, X., Z. Liu, and Z. Huang, "Deinterleaving of Pulse Streams with Denoising Autoencoders," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 6, 2020. DOI: 10.1109/TAES.2020.3004208.
- [35] Liu, H., L. Wang, and G. Wang, "Emitter Signal Deinterleaving Based on Single PDW with Modulation-Hypothesis-Augmented Transformer," *Remote Sensing*, Vol. 16, No. 20, 2024. DOI: 10.3390/rs16203830.
- [36] Zhou, Z., et al., "Radar Signal Sorting with Multiple Self-Attention Coupling Mechanism Based Transformer Network," *IEEE Signal Processing Letters*, Vol. 31, 2024. DOI: 10.1109/LSP.2024.3421948.
- [37] Chao, W., et al., "A Radar Signal Deinterleaving Method Based on Semantic Segmentation with Neural Network," *IEEE Transactions on Signal Processing*, Vol. 70, 2022. DOI: 10.1109/TSP.2022.3229630.
- [38] Can, O., et al., "Blind Deinterleaving of Signals in Time Series with Self-Attention Based Soft Min-Cost Flow Learning," in *Acoustics, Speech and Signal Processing*, IEEE, 2021. DOI: 10.1109/icassp39728.2021.9415025.
- [39] Kocamış, B., A. Orduyılmaz, and S. Taşcıoğlu, "Object Detection-Based Deinterleaving of Radar Signals Using Deep Learning for Cognitive EW," *Signal, Image and Video Processing*, Vol. 18, No. 11, 2024. DOI: 10.1007/s11760-024-03428-2.
- [40] Liu, H., et al., "SemHybridNet: A Semantically Enhanced Hybrid CNN-Transformer Network for Radar Pulse Image Segmentation," *Complex & Intelligent Systems*, Vol. 10, No. 2, 2024. DOI: 10.1007/s40747-023-01294-y.
- [41] Testi, E., and A. Giorgetti, "RSS-Based Localization of Multiple Radio Transmitters via Blind Source Separation," *IEEE Communications Letters*, Vol. 26, No. 3, 2022. DOI: 10.1109/LCOMM.2021.3137598.
- [42] Shchur, O., et al., *Neural Temporal Point Processes: A Review*, 2021. DOI: 10.48550/arXiv.2104.03528.
- [43] Lima, R., *Hawkes Processes Modeling, Inference and Control: An Overview*, 2021. DOI: 10.48550/arXiv.2011.13073.
- [44] Omi, T., N. Ueda, and K. Aihara, *Fully Neural Network based Model for General Temporal Point Processes*, 2020. DOI: 10.48550/arXiv.1905.09690.
- [45] Baytas, I. M., et al., "Patient Subtyping via Time-Aware LSTM Networks," in *Knowledge Discovery and Data Mining*, 2017. DOI: 10.1145/3097983.3097997.
- [46] Rubanova, Y., R. T. Q. Chen, and D. Duvenaud, *Latent ODEs for Irregularly-Sampled Time Series*, 2019. DOI: 10.48550/arXiv.1907.03907.

- [47] Suárez-Varela, J., et al., *Graph Neural Networks for Communication Networks: Context, Use Cases and Opportunities*, 2022. DOI: 10.48550/arXiv.2112.14792.
- [48] Emmanuel, T., et al., "A Survey on Missing Data in Machine Learning," *Journal of Big data*, Vol. 8, 2021. DOI: 10.1186/s40537-021-00516-9.
- [49] Kocev, B., et al., "Uncertainty-Aware Asynchronous Scattered Motion Interpolation Using Gaussian Process Regression," *Computerized Medical Imaging and Graphics*, Vol. 72, 2019. DOI: 10.1016/j.compmedimag.2018.12.001.
- [50] Ebeling, H., D. A. White, and F. V. N. Rangarajan, "ASMOOTH: A Simple and Efficient Algorithm for Adaptive Kernel Smoothing of Two-Dimensional Imaging Data," *Monthly Notices of the Royal Astronomical Society*, Vol. 368, No. 1, 2006. DOI: 10.1111/j.1365-2966.2006.10135.x.
- [51] Adams, R., and D. MacKay, "Bayesian Online Changepoint Detection," 2007. DOI: 10.48550/arXiv.0710.3742.
- [52] Kleinberg, J., "Bursty and Hierarchical Structure in Streams," *Data Mining and Knowledge Discovery*, Vol. 7, No. 4, 2003. DOI: 10.1023/A:1024940629314.
- [53] Svensson, A., "Classification of Radar Emitters Based on Pulse Repetition Interval using Machine Learning," English, Master's thesis, KTH, Royal Institute of Technology, Stockholm, Sweden, 2022. Online: <https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-320777>.
- [54] Zhu, Q., Y. Wang, and G. Shen, "Comparison and Application of Time-Frequency Analysis Methods for Nonstationary Signal Processing," *Communications in Computer and Information Science*, 2011. DOI: 10.1007/978-3-642-21783-8_47.
- [55] Aghababae, H., J. Amini, and Y. Tzeng, "Improving Change Detection Methods of SAR Images Using Fractals," *Scientia Iranica*, Vol. 20, No. 1, 2013. DOI: 10.1016/j.scient.2012.11.006.
- [56] Dudczyk, J., "Specific Emitter Identification Based on Fractal Features," in *Fractal Analysis - Applications in Physics, Engineering and Technology*, IntechOpen, 2017. Online: <https://tinyurl.com/dudczyk-sei>.
- [57] Shen, L., Y.-s. Han, and S. Wang, "Research on Fractal Feature Extraction of Radar Signal Based on Wavelet Transform," in *Advances in Intelligent Systems and Interactive Applications*, 2017. DOI: 10.1007/978-3-319-69096-4_46.
- [58] Brandfass, M., et al., "Towards Cognitive Radar via Knowledge Aided Processing for Airborne and Ground based Radar Applications," in *International Radar Symposium*, 2019. DOI: 10.23919/IRS.2019.8768133.
- [59] Katsilieris, F., and A. Charlish, "Knowledge Based Anomaly Detection for Ground Moving Targets," in *IEEE Radar Conference*, 2018. DOI: 10.1109/RADAR.2018.8378660.
- [60] Zhang, Y., et al., "Path Loss Prediction Based on Machine Learning: Principle, Method, and Data Expansion," *Applied Sciences*, 2019. Online: <https://www.mdpi.com/2076-3417/9/9/1908>.
- [61] Amorim, L. B. V. de, G. D. C. Cavalcanti, and R. M. O. Cruz, "Meta-Scaler: A Meta-Learning Framework for the Selection of Scaling Techniques," *IEEE Transactions on Neural Networks and Learning Systems*, 2024. DOI: 10.1109/TNNLS.2024.3366615.
- [62] Balaji, Y., S. Sankaranarayanan, and R. Chellappa, "MetaReg: Towards Domain Generalization using Meta-Regularization," in *NeurIPS*, Vol. 31, Curran Associates, Inc., 2018. DOI: <https://dl.acm.org/doi/10.5555/3326943.3327036>.

- [63] Parmezan, A. R. S., H. D. Lee, and F. C. Wu, "Metalearning for Choosing Feature Selection Algorithms in Data Mining: Proposal of a New Framework," *Expert Systems with Applications*, Vol. 75, 2017. DOI: 10.1016/j.eswa.2017.01.013.
- [64] Ylmaz Eroglu, D., and U. Akcan, "An Adapted Ant Colony Optimization for Feature Selection," *Applied Artificial Intelligence*, Vol. 38, No. 1, 2024. DOI: 10.1080/08839514.2024.2335098.
- [65] Rusu, A., et al., "Meta-Learning with Latent Embedding Optimization," in *ICLR*, 2019. Online: <https://arxiv.org/abs/1807.05960>.
- [66] Zintgraf, L., et al., "Fast Context Adaptation via Meta-Learning," in *ICLR*, 2019. Online: <https://arxiv.org/abs/1810.03642>.
- [67] Krishnamurthy, V., *Adversarial Radar Inference. From Inverse Tracking to Inverse Reinforcement Learning of Cognitive Radar*, 2020. Online: <https://arxiv.org/abs/2002.10910>.
- [68] Haigh, K. Z., et al., "Modeling RF Interference Performance," in *Collaborative Electronic Warfare Symposium*, 2014.
- [69] Topal, O., S. Gecgel, and E. Eksioglu, *Identification of Smart Jammers: Learning based Approaches Using Wavelet Representation*, 2019. Online: <https://arxiv.org/abs/1901.09424>.
- [70] Krishnamurthy, V., et al., "Adversarial Radar Inference: Inverse Tracking, Identifying Cognition, and Designing Smart Interference," in, Mishra, K., B. Shankar, and M. Rangaswamy, (Eds.), Raleigh, NC: SciTech Publishing, 2023.
- [71] Zhang, G., W. Jin, and L. Hu, "Radar Emitter Signal Recognition Based On Support Vector Machines," in *Control, Automation, Robotics and Vision Conference*. IEEE, Vol. 2, 2004. DOI: 10.1109/ICARCV.2004.1468946.
- [72] Zhang, G., H. Rong, and W. Jin, "Application of Support Vector Machine to Radar Emitter Signal Recognition," *Journal of Southwest Jiaotong University*, Vol. 41, No. 1, 2006. DOI: 10.1007/978-3-540-37258-5.16.
- [73] Barshan, B., and B. Eravci, "Automatic Radar Antenna Scan Type Recognition in Electronic Warfare," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 48, No. 4, 2012. DOI: 10.1109/TAES.2012.6324669.
- [74] Mustafa, H., and M. Doroslovacki, "Digital Modulation Recognition Using Support Vector Machine Classifier," in *Asilomar Conference on Signals, Systems and Computers*. IEEE, Vol. 2, 2004. DOI: 10.1109/ACSSC.2004.1399565.
- [75] Park, C.-S., et al., "Automatic Modulation Recognition Using Support Vector Machine in Software Radio Applications," in *Advanced Communication Technology*. IEEE, Vol. 1, 2007. DOI: 10.1109/ICACT.2007.358249.
- [76] Li, X., et al., "Deep Learning Driven Wireless Communications and Mobile Computing," *Wireless Communications and Mobile Computing*, 2019. DOI: 10.1155/2019/5629572.
- [77] Majumder, U., E. Blasch, and D. Garren, *Deep Learning for Radar and Communications Automatic Target Recognition*. Norwood, MA: Artech House, 2020.
- [78] Robinson, J., and S. Kuzdeba, "RiftNet: Radio Frequency Classification for Large Populations," in *Consumer Communications & Networking Conference*, 2021. DOI: 10.1109/CCNC49032.2021.9369455.
- [79] O'Shea, T., J. Corgan, and T. Clancy, "Convolutional Radio Modulation Recognition Networks," in *Engineering Applications of Neural Networks*. Springer, 2016. Online: <https://arxiv.org/abs/1602.04105>.

- [80] O'Shea, T., T. Roy, and T. Clancy, "Over-The-Air Deep Learning Based Radio Signal Classification," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 12, No. 1, 2018. Online: <https://arxiv.org/abs/1712.04578>.
- [81] Shi, Y., et al., "Deep Learning for RF Signal Classification in Unknown and Dynamic Spectrum Environments," in *International Symposium on Dynamic Spectrum Access Networks*. IEEE, 2019. Online: <https://arxiv.org/abs/1909.11800>.
- [82] Jiang, R., et al., "A Generalized Radio Frequency Fingerprint Based Wireless Device Identification Using Siamese-Based Neural Network," *IEEE Sensors Journal*, 2024. DOI: 10.1109/JSEN.2024.3411696.
- [83] Sun, J., et al., "Radar Emitter Classification Based on Unidimensional Convolutional Neural Network," *IET Radar, Sonar and Navigation*, Vol. 12, No. 8, 2018. DOI: 10.1049/iet-rsn.2017.0547.
- [84] Notaro, P., et al., *Radar Emitter Classification with Attribute-Specific Recurrent Neural Networks*, 2019. Online: <https://arxiv.org/abs/1911.07683>.
- [85] Li, M., et al., "Generative Adversarial Networks-Based Semi-Supervised Automatic Modulation Recognition for Cognitive Radio Networks," *Sensors*, 2018. Online: <https://www.mdpi.com/1424-8220/18/11/3913/pdf-vor>.
- [86] Shi, Y., K. Davaslioglu, and Y. Sagduyu, *Generative Adversarial Network for Wireless Signal Spoofing*, 2019. Online: <https://arxiv.org/abs/1905.01008>.
- [87] Urrego, B., "Army Signal Classification Challenge," in *GNURadio Conference*, 2018. Online: <https://tinyurl.com/army-signal-challenge>.
- [88] Logue, K., et al., "Expert RF Feature Extraction to Win the Army RCO AI Signal Classification Challenge," in *Python in Science*, 2019. Online: <https://tinyurl.com/logue-signal-classification>.
- [89] Vila, A., et al., "Deep and Ensemble Learning to Win the Army RCO AI Signal Classification Challenge," in *Python in Science*, 2019. Online: <https://tinyurl.com/vila-signal-classification>.
- [90] Chen, F., et al., *On Passive Wireless Device Fingerprinting Using Infinite Hidden Markov Random field*, 2012. Online: <https://tinyurl.com/chen2012fingerprint>.
- [91] Nguyen, N., et al., "Device Fingerprinting to Enhance Wireless Security Using Nonparametric Bayesian Method," in *INFOCOM*, 2011. DOI: 10.1109/INFOCOM.2011.5934926.
- [92] Sankhe, K., et al., "ORACLE: Optimized Radio Classification through Convolutional neural Networks," in *INFOCOM*, 2019. DOI: 10.1109/INFOCOM.2019.8737463, Dataset available at <https://genesys-lab.org/oracle>.
- [93] Youssef, K., et al., *Machine Learning Approach to RF Transmitter Identification*, 2017. Online: <https://arxiv.org/abs/1711.01559>.
- [94] Wong, L., et al., "Clustering Learned CNN Features from Raw I/Q Data for Emitter Identification," in *MILCOM*, 2018. Online: <https://ieeexplore.ieee.org/document/8599847>.
- [95] Jian, T., et al., "Deep Learning for RF Fingerprinting: A Massive Experimental Study," *IEEE Internet of Things Magazine*, Vol. 3, No. 1, 2020. DOI: 10.1109/IOTM.0001.1900065.
- [96] Restuccia, F., et al., *DeepRadioID: Real-Time Channel-Resilient Optimization of Deep Learning-based Radio Fingerprinting Algorithms*, 2019. Online: <https://tinyurl.com/deepRadioID>.
- [97] Tang, P., et al., "Causal Learning for Robust Specific Emitter Identification Over Unknown Channel Statistics," *IEEE Transactions on Information Forensics and Security*, Vol. 19, 2024. DOI: 10.1109/TIFS.2024.3393237.

- [98] Jagannath, A., J. Jagannath, and P. S. P. V. Kumar, "A Comprehensive Survey on Radio Frequency (RF) Fingerprinting: Traditional Approaches, Deep Learning, and Open Challenges," *Computer Networks*, Vol. 219, 2022. DOI: 10.1016/j.comnet.2022.109455.
- [99] Xie, L., et al., "Radio Frequency Fingerprint Identification for Internet of Things: A Survey," *Security and Safety*, Vol. 3, 2024. Published Online: 18 September 2023, Review. DOI: 10.1051/sands/2023022.
- [100] Kuzdeba, S., *Radio Frequency Machine Learning: A Practical Deep Learning Perspective*. Norwood, MA: Artech House, 2025.
- [101] Haigh, K. Z., et al., "Parallel Learning and Decision Making for a Smart Embedded Communications Platform," BBN Technologies, Tech. Rep. BBN-REPORT-8579, 2015. Online: <http://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-SO.sm.pdf>.
- [102] Baldo, N., et al., "A Neural Network based Cognitive Controller for Dynamic Channel Selection," in *International Conference on Communications*. IEEE, 2009. DOI: 10.1109/ICC.2009.5198636.
- [103] Haigh, K. Z., S. Varadarajan, and C. Y. Tang, "Automatic Learning-based MANET Cross-Layer Parameter Configuration," in *Workshop on Wireless Ad Hoc and Sensor Networks*. IEEE, 2006. Online: <https://www.cs.cmu.edu/~khaigh/papers/haigh06a-configuration.pdf>.
- [104] Katidiotis, A., K. Tsagaris, and P. Demestichas, "Performance Evaluation of Artificial Neural Network-Based Learning Schemes for Cognitive Radio Systems," *Computers and Electric Engineering*, Vol. 36, No. 3, 2010. DOI: 10.1016/j.compeleceng.2009.12.005.
- [105] Troxel, G., et al., "Adaptive Dynamic Radio Open-Source Intelligent Team (ADROIT): Cognitively-Controlled Collaboration Among SDR Nodes," in *Workshop on Networking Technologies for Software Defined Radio (SDR) Networks*. IEEE, 2006. DOI: 10.1109/SDR.2006.4286321.
- [106] Haviluddin, A., et al., "Modelling of Network Traffic Usage Using Self-Organizing Maps Techniques," in *Science in Information Technology*, 2016. DOI: 10.1109/ICSITech.2016.7852658.
- [107] Qu, X., et al., "A Survey on the Development of Self-Organizing Maps for Unsupervised Intrusion Detection," *Mobile Networks and Applications*, 2019. DOI: 10.1007/s11036-019-01353-0.
- [108] Demestichas, P., et al., "Enhancing Channel Estimation in Cognitive Radio Systems by Means of Bayesian Networks," *Wireless Personal Communications*, Vol. 49, 2009. DOI: 10.1007/s11277-008-9559-1.
- [109] Selvi, E., et al., "On the Use of Markov Decision Processes in Cognitive Radar: An Application to Target Tracking," in *IEEE Radar Conference*, 2018. DOI: 10.1109/RADAR.2018.8378616.
- [110] Thornton, C., et al., *Experimental Analysis of Reinforcement Learning Techniques for Spectrum Sharing Radar*, 2020. Online: <https://arxiv.org/abs/2001.01799>.
- [111] Kulin, M., et al., *A Survey on Machine Learning-based Performance Improvement of Wireless Networks: PHY, MAC and Network Layer*, 2021. DOI: 10.3390/electronics10030318.
- [112] Yi, W., et al., "Recent Advances in Multi-Radar Collaborative Surveillance: Cognitive Tracking and Resource Scheduling Algorithms," *Journal of Radars*, Vol. 12, No. 3, 2023. DOI: 10.12000/JR23036.
- [113] Wang, S., et al., *Principle and Technology of Cognitive Electronic Warfare*. Beijing, China: National Defense Industry Press, 2018.

- [114] Lee, G.-H., J. Jo, and C. H. Park, "Jamming Prediction for Radar Signals Using Machine Learning Methods," *Security and Communication Networks*, 2020. Online: <https://www.hindawi.com/journals/scn/2020/2151570/>.
- [115] English, D., *C4ISRNET the Compass/Net Defense Blogs: How multi-INT Enables Deciphering the Indecipherable*, 2015. Online: <https://tinyurl.com/c4isrnet>. Accessed 2025-02-17.
- [116] Hall, D., and J. Llinas, "An Introduction to Multisensor Data Fusion," *Proceedings of the IEEE*, Vol. 85, No. 1, 1997. DOI: 10.1109/5.554205.
- [117] White, F., "JDL, Data Fusion Lexicon," *Technical Panel for C*, Vol. 3, 1991. Online: <https://apps.dtic.mil/sti/pdfs/ADA529661.pdf>.
- [118] Khaleghi, B., et al., "Multisensor Data Fusion: A Review of the State-of-the-Art," *Information Fusion*, Vol. 14, No. 1, 2013. DOI: 10.1016/j.inffus.2011.08.001.
- [119] Castanedo, F., "A Review of Data Fusion Techniques," *The Scientific World Journal*, Vol. 2013, 2013. DOI: 10.1155/2013/704504.
- [120] Blasch, E., and D. Lambert, *High-Level Information Fusion Management and Systems Design*. Norwood, MA: Artech House, 2012.
- [121] Liggins II, M., D. Hall, and J. Llinas, *Handbook of Multisensor Data Fusion: Theory and Practice*, 2nd Edition. Boca Raton, FL: CRC press, 2008.
- [122] Blasch, E., et al., "Revisiting the JDL model for information Exploitation," in *FUSION*. IEEE, 2013. Online: <https://ieeexplore.ieee.org/document/6641155>.
- [123] ISIF.org, *International Society of Information Fusion*, 2025. Online: <http://isif.org/>. Accessed 2025-02-17.
- [124] Milisavljević, N., (Ed.), *Sensor and Data Fusion*. London, UK: InTechOpen, 2009. DOI: 10.5772/102.
- [125] Waltz, E., *Data Fusion for C3I: A Tutorial*. London, UK: Argus Business, 1986.
- [126] Fabeck, G., and R. Mathar, "Kernel-Based Learning of Decision Fusion in Wireless Sensor Networks," in *FUSION*. IEEE, 2008. Online: <https://ieeexplore.ieee.org/document/4632207>.
- [127] Ansari, N., et al., "Adaptive Fusion by Reinforcement Learning for Distributed Detection Systems," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 32, No. 2, 1996. Online: <https://web.njit.edu/~ansari/papers/96AES.pdf>.
- [128] Stankovic, J., and T. He, "Energy Management in Sensor Networks," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 370, No. 1958, 2012. DOI: 10.1098/rsta.2011.0195.
- [129] Wang, S., et al., "Multi-Modal Fusion Sensing: A Comprehensive Review of Millimeter-Wave Radar and Its Integration With Other Modalities," *IEEE Communications Surveys & Tutorials*, Vol. 27, No. 1, 2025. DOI: 10.1109/COMST.2024.3398004.
- [130] Liu, S., et al., "Data Fusion of Roadside Camera, LiDAR, and Millimeter-Wave Radar," *IEEE Sensors Journal*, Vol. 24, No. 20, 2024. DOI: 10.1109/JSEN.2024.3448428.
- [131] Samadzadegan, F., A. Toosi, and F. Dadrass Javan, "A Critical Review on Multi-Sensor and Multi-Platform Remote Sensing Data Fusion Approaches: Current Status and Prospects," *International Journal of Remote Sensing*, Vol. 46, No. 3, 2024. DOI: 10.1080/01431161.2024.2429784.
- [132] Levashova, T., et al., "Situation Detection Based on Knowledge Fusion Patterns," in *International Workshop on Ontologies and Information Systems*, 2014. Online: <http://ceur-ws.org/Vol-1230/paper-02.pdf>.

- [133] 3GPP, "Study on NR Positioning Support (Release 16)," Standard 3GPP TSG RAN TR 38.855, 2019.
- [134] Bartoletti, S., et al., "5G Localization and Context-Awareness," in *5G Italy White eBook: from Research to Market*, 2018. Online: <https://www.5gitaly.eu/2018/wp-content/uploads/2019/01/5G-Italy-White-eBook-5G-Localization.pdf>.
- [135] Conti, A., et al., "Soft Information for Localization-of-Things," *Proceedings of the IEEE*, Vol. 107, No. 11, 2019. DOI: 10.1109/JPROC.2019.2905854.
- [136] Lau, B., et al., "A Survey of Data Fusion in Smart City Applications," *Information Fusion*, Vol. 52, 2019. DOI: 10.1016/j.inffus.2019.05.004.
- [137] Witrals, K., et al., "High-Accuracy Localization for Assisted Living: 5G Systems Will Turn Multipath Channels from Foe to Friend," *IEEE Signal Processing Magazine*, Vol. 33, No. 2, 2016. DOI: 10.1109/MSP.2015.2504328.
- [138] Guerra, A., F. Guidi, and D. Dardari, "Single-Anchor Localization and Orientation Performance Limits Using Massive Arrays: MIMO vs. Beamforming," *IEEE Transactions on Wireless Communications*, Vol. 17, No. 8, 2018. DOI: 10.1109/TWC.2018.2840136.
- [139] Wymeersch, H., et al., "5G mmWave Positioning for Vehicular Networks," *IEEE Wireless Communications*, Vol. 24, No. 6, 2017. DOI: 10.1109/MWC.2017.1600374.
- [140] Win, M., et al., "Network Operation Strategies for Efficient Localization and Navigation," *Proceedings of the IEEE*, Vol. 106, No. 7, 2018. DOI: 10.1109/JPROC.2018.2835314.
- [141] Campbell, M., and N. Ahmed, "Distributed Data Fusion: Neighbors, Rumors, and the Art of Collective Knowledge," *IEEE Control Systems Magazine*, Vol. 36, No. 4, 2016. DOI: 10.1109/MCS.2016.2558444.
- [142] Lang, J., "Collective Decision Making under Incomplete Knowledge: Possible and Necessary Solutions," in *IJCAI*, 2020. Online: <https://www.ijcai.org/Proceedings/2020/0680.pdf>.
- [143] Makarenko, A., et al., "Decentralised Data Fusion: A Graphical Model Approach," in *FUSION*, 2009. Online: <https://ieeexplore.ieee.org/document/5203740>.
- [144] Thompson, P., and H. Durrant-Whyte, "Decentralised Data Fusion in 2-Tree Sensor Networks," in *FUSION*, 2010. DOI: 10.1109/ICIF.2010.5712031.
- [145] Hall, D., et al., (Eds.), *Distributed Data Fusion for Network-Centric Operations*. Boca Raton, FL: CRC Press, 2012.
- [146] Munir, A., et al., "Artificial Intelligence and Data Fusion at the Edge," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 36, No. 7, 2021. DOI: 10.1109/MAES.2020.3043072.
- [147] Chandola, V., A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, Vol. 41, No. 3, 2009. DOI: 10.1145/1541880.1541882.
- [148] Edgeworth, F., "On Observations Relating to Several Quantities," *Hermathena*, Vol. 6, No. 13, 1887. Online: <https://www.jstor.org/stable/23036355>.
- [149] Song, X., et al., "Conditional Anomaly Detection," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 19, No. 5, 2007. DOI: 10.1109/TKDE.2007.1009.
- [150] Agyemang, M., K. Barker, and R. Alhajj, "A Comprehensive Survey of Numeric and Symbolic Outlier Mining Techniques," *Intelligent Data Analysis*, Vol. 10, No. 6, 2006. Online: <https://dl.acm.org/doi/10.5555/1609942.1609946>.
- [151] Ahmed, M., A. N. Mahmood, and J. Hu, "A Survey of Network Anomaly Detection Techniques," *Journal of Network and Computer Applications*, Vol. 60, 2016. DOI: 10.1016/j.jnca.2015.11.016.

- [152] Bakar, Z., et al., "A Comparative Study for Outlier Detection Techniques in Data Mining," in *Conference on Cybernetics and Intelligent Systems*. IEEE, 2006. DOI: 10.1109/ICCIS.2006.252287.
- [153] Beckman, R., and R. Cook, "Outlier.....s," *Technometrics*, Vol. 25, No. 2, 1983. DOI: 10.2307/1268548.
- [154] Hodge, V., and J. Austin, "A Survey of Outlier Detection Methodologies," *Artificial Intelligence Review*, Vol. 22, No. 2, 2004. Online: <https://tinyurl.com/outlier-detection>.
- [155] Markou, M., and S. Singh, "Novelty Detection: A Review—Part 1: Statistical Approaches," *Signal Processing*, Vol. 83, No. 12, 2003. DOI: 10.1016/j.sigpro.2003.07.018.
- [156] Markou, M., and S. Singh, "Novelty Detection: A Review—Part 2: Neural Network Based Approaches," *Signal Processing*, Vol. 83, No. 12, 2003. DOI: 10.1016/j.sigpro.2003.07.019.
- [157] Patcha, A., and J.-M. Park, "An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends," *Computer Networks*, Vol. 51, No. 12, 2007. DOI: 10.1016/j.comnet.2007.02.001.
- [158] Chawla, S., "Deep Learning Based Intrusion Detection System for Internet of Things," Ph.D. dissertation, University of Washington, 2017. Online: <http://hdl.handle.net/1773/39829>.
- [159] Yin, C., et al., "A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks," *IEEE Access*, Vol. 5, 2017. DOI: 10.1109/ACCESS.2017.2762418.
- [160] Bowman, B., "Anomaly Detection using a Variational Autoencoder Neural Network with a Novel Objective Function and Gaussian Mixture Model Selection Technique," M.S. thesis, Naval Postgraduate School, 2019. Online: <https://apps.dtic.mil/sti/pdfs/AD1080012.pdf>.
- [161] Esmaili, F., et al., "Anomaly Detection for Sensor Signals Utilizing Deep Learning Autoencoder-Based Neural Networks," *Bioengineering*, Vol. 10, No. 4, 2023. DOI: 10.3390/bioengineering10040405.
- [162] Rajasegarar, S., et al., "Centered Hyperspherical and Hyperellipsoidal One-Class Support Vector Machines for Anomaly Detection in Sensor Networks," *IEEE Transactions on Information Forensics and Security*, Vol. 5, No. 3, 2010. DOI: 10.1109/TIFS.2010.2051543.
- [163] Zhang, M., B. Xu, and J. Gong, "An Anomaly Detection Model Based on One-Class SVM to Detect Network Intrusions," in *Mobile Ad-hoc and Sensor Networks*. IEEE, 2015. DOI: 10.1109/MSN.2015.40.
- [164] Salman, T., et al., "Machine Learning for Anomaly Detection and Categorization in Multi-Cloud Environments," in *Cyber Security and Cloud Computing*. IEEE, 2017. DOI: 10.1109/CSCloud.2017.15.
- [165] Fiore, U., et al., "Network Anomaly Detection with the Restricted Boltzmann Machine," *Neurocomputing*, Vol. 122, 2013. DOI: 10.1016/j.neucom.2012.11.050.
- [166] Dästner, K., et al., "Machine Learning Support for Radar-Based Surveillance Systems," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 36, No. 7, 2021. DOI: 10.1109/MAES.2020.3001966.
- [167] Bauw, M., et al., "From Unsupervised to Semi-supervised Anomaly Detection Methods for HRRP Targets," in *IEEE Radar Conference*, 2020. DOI: 10.1109/RadarConf2043947.2020.9266497.
- [168] Ruff, L., et al., "Deep Semi-Supervised Anomaly Detection," in *ICLR*, 2020. Online: <https://openreview.net/forum?id=HkgH0TEYwH>.
- [169] Yu, H., et al., "Meta-ADD: A Meta-learning Based Pre-trained Model for Concept Drift Active Detection," *Information Sciences*, Vol. 608, 2022. DOI: 10.1016/j.ins.2022.07.022.

- [170] Aguiar, G. J., and A. Cano, "Enhancing Concept Drift Detection in Drifting and Imbalanced Data Streams through Meta-Learning," in *BigData*, 2023. DOI: 10.1109/BigData59044.2023.10386364.
- [171] Hinder, F., V. Vaquet, and B. Hammer, "One or Two Things We Know About Concept Drift—A Survey on Monitoring in Evolving Environments. Part A: Detecting Concept Drift," *Frontiers in Artificial Intelligence*, Vol. 7, 2024. DOI: 10.3389/frai.2024.1330257.
- [172] Hovakimyan, G., and J. M. Bravo, "Evolving Strategies in Machine Learning: A Systematic Review of Concept Drift Detection," *Information*, Vol. 15, No. 12, 2024. DOI: 10.3390/info15120786.
- [173] Pearl, J., *Causality*, 2nd Edition. Cambridge, UK: Cambridge University Press, 2009.
- [174] Pearl, J., "An Introduction to Causal Inference," *International Journal of Biostatistics*, 2010. Online: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2836213/>.
- [175] Tople, S., A. Sharma, and A. Nori, *Alleviating Privacy Attacks via Causal Learning*, 2020. Online: <https://arxiv.org/abs/1909.12732>.
- [176] Ali, B., et al., "A Causal Factors Analysis of Aircraft Incidents Due to Radar Limitations: The Norway Case Study," *Air Transport Management*, Vol. 44-45, 2015. DOI: 10.1016/j.jairtraman.2015.03.004.
- [177] Chernozhukov, V., et al., *Applied Causal Inference Powered by ML and AI*. 2024. Online: <https://causalml-book.org/>.
- [178] Martin, T., and K. Chang, "Development and Analysis of a Probabilistic Reasoning Methodology for Spectrum Situational Awareness and Parameter Estimation in Uncertain Environments," in *FUSION*, 2015. Online: <https://tinyurl.com/probabilistic-ssa>.
- [179] Peng, Y., et al., "RAFT: A Real-Time Framework for Root Cause Analysis in 5G and Beyond Vulnerability Detection," in *Consumer Communications & Networking Conference*, 2024. DOI: 10.1109/CCNC51664.2024.10454824.
- [180] Cousins, D., et al., "Understanding Encrypted Networks through Signal and Systems Analysis of Traffic Timing," in *IEEE Aerospace*, Vol. 6, 2003. DOI: 10.1109/AERO.2003.1235227.
- [181] Tilghman, P., and D. Rosenbluth, "Inferring Wireless Communications Links and Network Topology from External Using Granger Causality," in *MILCOM*, 2013. DOI: 10.1109/MILCOM.2013.219.
- [182] Chen, Y., et al., "Analyzing Multiple Nonlinear Time Series with Extended Granger Causality," *Physics Letters A*, Vol. 324, No. 1, 2004. DOI: 10.1016/j.physleta.2004.02.032.
- [183] Morris, K., "What is Hysteresis?" *Applied Mechanics Reviews*, Vol. 64, 2011. DOI: 10.1115/1.4007112.
- [184] Deshmukh, R., et al., "Reactive Temporal Logic-Based Precursor Detection Algorithm for Terminal Airspace Operations," Vol. 28, No. 4, 2020. DOI: 10.2514/1.D0182.
- [185] Salmond, D., "Blind estimation of wireless network topology and throughput," in *Conference on Information Sciences and Systems (CISS)*, 2019. DOI: 10.1109/CISS.2019.8692903.
- [186] Song, Y., et al., "Topology tracking of dynamic UAV wireless networks," *Chinese Journal of Aeronautics*, Vol. 35, No. 11, 2022. DOI: 10.1016/j.cja.2021.08.012.
- [187] Keren, S., R. Mirsky, and C. Geib, *Resources for Plan, Activity, Behavior, and Intent Recognition Research*. Online: <http://www.planrec.org/>. Accessed 2025-02-17.
- [188] Suktharankar, G., et al., *Plan, Activity, and Intent Recognition*. Amsterdam, Netherlands: Elsevier, 2014. DOI: 10.1016/C2012-0-01169-6.

- [189] Mirsky, R., S. Keren, and C. Geib, *Introduction to Symbolic Plan and Goal Recognition*. New York, NY: Springer, 2021.
- [190] Schmidt, C., N. Sridharan, and J. Goodson, "The Plan Recognition Problem: An Intersection of Psychology and Artificial Intelligence," *Artificial Intelligence*, Vol. 11, No. 1, 1978. DOI: 10.1016/0004-3702(78)90012-7.
- [191] Keren, S., R. Mirsky, and C. Geib, *Plan Activity and Intent Recognition Tutorial*, 2019. Online: http://www.planrec.org/Tutorial/Resources_files/pair-tutorial.pdf.
- [192] Geib, C., and R. Goldman, "A Probabilistic Plan Recognition Algorithm Based on Plan Tree Grammars," *Artificial Intelligence*, Vol. 173, No. 11, 2009. DOI: 10.1016/j.artint.2009.01.003.
- [193] Geib, C., and R. Goldman, "Plan Recognition in Intrusion Detection Systems," in *DARPA Information Survivability Conference and Exposition*, Vol. 1, 2001. DOI: 10.1109/DISCEX.2001.932191.
- [194] Geib, C., "Delaying Commitment in Plan Recognition Using Combinatory Categorical Grammars," in *IJCAI*, 2009. Online: http://www.planrec.org/Geib/Papers_files/IJCAI09.pdf.
- [195] Fu, M., "AlphaGo and Monte Carlo Tree Search: The Simulation Optimization Perspective," in *Winter Simulation Conference*, 2016. DOI: 10.1109/WSC.2016.7822130.
- [196] Latombe, G., E. Granger, and F. Dilkes, "Graphical EM for On-Line Learning of Grammatical Probabilities in Radar Electronic Support," *Applied Soft Computing*, Vol. 12, No. 8, 2012. DOI: 10.1016/j.asoc.2012.02.022.
- [197] Wang, A., and V. Krishnamurthy, "Threat Estimation of Multifunction Radars: Modeling and Statistical Signal Processing of Stochastic Context Free Grammars," in *Acoustics, Speech and Signal Processing*. IEEE, 2007. DOI: 10.1109/ICASSP.2007.366799.
- [198] Kantharaju, P., et al., "Combinatory Categorical Grammar Learning for Plan Recognition in Domains with Type Trees," in *Workshop on Plan, Activity, and Intent Recognition (PAIR)*, 2020. Online: <https://www.sift.net/sites/default/files/publications/pair-2020-paper-camera-ready.pdf>.
- [199] Li, R., et al., "Automatically Learning HTN Methods from Landmarks," 1, Vol. 37, 2024. DOI: 10.32473/flairs.37.1.135625.
- [200] Kantharaju, P., "Learning Decomposition Models for Hierarchical Planning and Plan Recognition," Ph.D. dissertation, Drexel University, Philadelphia, PA, 2020.
- [201] Lisý, V., et al., "Game-Theoretic Approach to Adversarial Plan Recognition," in *Frontiers in Artificial Intelligence and Applications*, Vol. 242, 2012. DOI: 10.3233/978-1-61499-098-7-546.
- [202] Le Guillarme, N., et al., "A Generative Game-Theoretic Framework for Adversarial Plan Recognition," in *Workshop on Distributed and Multi-Agent Planning*, 2015. Online: <https://tinyurl.com/adversary-planrec>.
- [203] Braynov, S., "Adversarial Planning and Plan Recognition: Two Sides of the Same Coin," in *Secure Knowledge Management Workshop*, 2006. Online: <https://tinyurl.com/adversarial-plan-rec>.

5

Electronic Protect and Electronic Attack

An EW system must choose actions to accomplish mission objectives, given whatever context it knows about the environment and the tasks: The platform(s) have a set of capabilities, and the cognitive decision maker composes these into strategies to achieve the desired performance. It is through these knobs, or degrees of freedom, that the EW system can accomplish its goals. Because the goal of this book is not to describe EP/EA, but rather to highlight what challenges AI and ML can help address, we therefore present solutions from the perspective of AI decision making.

The majority of EW literature separates the discussion of EP from EA, arguing that they are fundamentally different purposes. From an AI standpoint, however, they use the same underlying algorithmic solutions.

The only difference between EP and EA is the mission objective:

EP defines objectives with respect to oneself, while

EA defines objectives with respect to others.

Moreover, missions often need to accomplish objectives of both types, so it is unwise to decouple the concepts. Ignoring one objective at the expense of another leads to mission failure, for example jamming for purposes of EA can lead to fratricide and failure of the EP mission.

Figure 5.1 outlines a spectrum of strategies that can accomplish both EP and EA objectives. Decision making applies to ES as well, for example reconfiguring a sensor, cross-sensor cueing, or changing the sensor schedule. Any given action can apply to any of the ES, EP, or EA objectives. Figure 1.7 outlines several techniques,

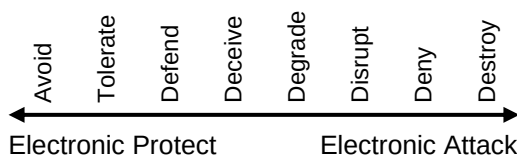


Figure 5.1 EP and EA lie on a spectrum, where a chosen strategy can be effective for multiple mission objectives.

each of which could contribute to multiple objectives. Jamming, for example, would normally be considered an action for EA, but jamming can be used for ES, EP, or EA. Precision jamming can increase ES understanding of the RF environment and emitter capabilities, deceptive jamming for EP can confuse a threat, and jamming for EA can deny the adversary access to their own spectrum. A single transmission could potentially contribute to all three objectives. Embedding information in the signal can even accomplish both radar and communications objectives in the same transmission. Likewise, when an EWO tells a pilot to *break left*, the aircraft's rapid turn alters its radial velocity relative to a tracking radar, taking it through the Doppler notch so that the radar struggles to track the target. While this physical action is generally considered EP, it could also be an ES action testing the radar's limits, or performing deception more consistent with EA. Table 6.3 outlines other active sensing actions to support ES.

Two key reasons for AI-based decision making in EW are *time* and *complexity*. Decision-making time requirements are faster than humans are capable of handling. Moreover, the domain has too many inputs for a human to understand quickly, and too many choices for a human to analyze.

Figure 5.2 illustrates a notional flow. The decision maker takes input from the ES system, consisting of the observables o and recent control feedback on the metrics m , and chooses a new strategy s to implement. If there is an online learning

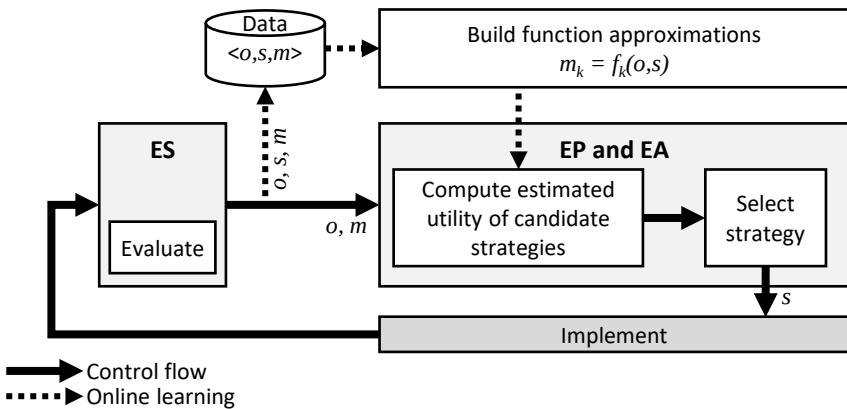


Figure 5.2 The decision maker chooses a strategy s based on recent observables o and performance feedback of the metrics m . An optional online learning step updates performance models based on empirical data. (Based on Example 1.1.)

loop (dotted line; see also Chapter 7), the system can update models on the fly. As a simple example, imagine choosing commute routes to work. Observables are day of week, time of day, and weather conditions. Strategies involve the route and vehicle. Metrics are time and distance. The optimizer chooses the best strategy for current conditions, and a learning component updates the model as needed (e.g., if a road goes under construction). Note that there might not be a solution that meets all metrics. A key ability of AI-style decision making is its ability to choose what to constraints or objectives to relax and help soften the blow of failures.

Traditional look-up tables and one-to-one mapping of environment to technique are inferior to AI-based decision making in every realistic case with many performance metrics, capabilities, platforms, missions, and complex jammers. AI systems have largely abandoned these rule-based and expert-system based approaches because of their inherent rigidity, brittleness, inability to generalize to new settings, and difficulty to maintain. The equivalent techniques in EW—mission data files and regular expressions—are inappropriate for modern settings. Instead, systems need a dynamic library where concepts can be human- or machine-edited, and changed and extended as the system encounters new conditions. The system could initially start with a hand-created mapping of environments to techniques as illustrated in Figure 2.6; this library supports a decision maker in choosing the best strategy for a given setting. In-mission learning then updates the library by capturing complex interactions and changing conditions, enabling greater flexibility, generalization, and the ability to adapt to new information without extensive manual intervention.

Choosing a strategy is best solved using *decision theory* approaches because it follows a rigorous approach to reasoning about action choices. Decision theory is the science of choosing actions based on their expected outcomes, given constraints and assumptions about the problem, as illustrated in Figure 5.3.

Decision theory has a long history; renowned economist Kenneth Arrow describes early work [1]. Russell and Norvig [2] provide a good introductory discussion of decision theory, discussing rational decision making, how to handle

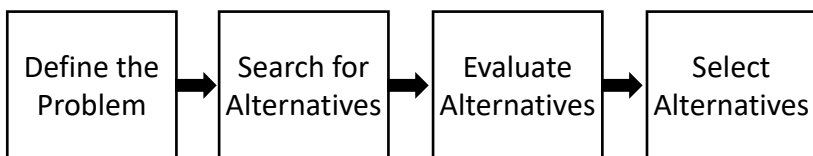


Figure 5.3 Decision theory gives a structural framework for thinking about the problem and making choices.

uncertainty and partially observable environments, how to compute utility over time, and how to handle adversaries. In a complete CogEW system, decision theory appears in the context of three highly connected concepts: planning, scheduling, and optimization.

- *Planning* synthesizes a sequence of actions that result in a desired goal state. Planning is *what to do*, and *in what order*, as a partially ordered graph. Planning is more strategic, more global. An EBM system plans how many platforms to deploy, which resources each gets, and where they will go. Chapter 6 describes the EW planning problem, which is at a higher level than scheduling and optimization.
- *Optimization* evaluates multiple plans to choose the “best” plan. Optimization is more tactical, more local. An EW system optimizes ES, EP, and EA metrics like power usage, probability of detection, and EW BDA. Section 5.1 describes optimization, including multiple objectives.
- *Scheduling* maps a partially ordered plan to specific resources and timeslots. Scheduling worries about *when* and *how to do things*. Scheduling drives down into the specifics of when to transmit and when to receive. Section 5.2 describes scheduling.

These activities are not clearly delineated. Traditionally, humans create the EW plan, delegating detailed scheduling and automation to the system. When planning is also automated, these activities are closely tied together; in fact, solution feasibility can only be guaranteed by dynamically harmonizing the layers [3, 4].

Chapter 2 presented the objective function, and Chapter 4 (ES) explained how observables describe the environment. This chapter describes *how to choose a strategy*, with a focus on optimization and scheduling. Section 5.1 describes optimization approaches, and Section 5.2 discusses scheduling. Section 5.3 discusses the concept of reward hacking, namely over-optimizing to the wrong function. Section 5.4 describes a desirable attribute for decision making: that implementations need to be interruptible. Finally, Section 5.5 considers methods to optimize across a distributed network of nodes.

5.1 Optimization

An optimization problem consists of choosing a set of values that maximize (or minimize) a utility function. Optimization has been a subject of mathematics since at least 1646 [5], with modern cross-disciplinary work falling in many fields including operations research, economics, and AI. AI confers *heuristics* and *randomized search*, allowing systems to solve previously impossible problems.

AI tackles the challenge of searching the exponential performance landscape. Section 5.1.1 describes the challenges of multiobjective optimization. Section 5.1.2 highlights some of the approximation approaches developed by the AI community. Additionally, Section 3.5 presents a few metalearning techniques being used to improve optimization.

5.1.1 Multiobjective Optimization

It is rarely possible to optimize all objectives simultaneously. Usually, the system must trade off the various objectives, making adequate performance toward all of them, without being optimal in any. For example, there is a rate/range/power trade-off, and a trade between robustness and efficiency. Ramzan et al. [6] survey multiobjective optimization approaches for spectrum sharing. A classic example in radar is probability of detection (Pd) versus probability of false alarms (Pfa): the receiver operating characteristic (ROC) curve defines the trade-off [7, 8]. Joint optimization of radar and communications performance is another common multi-objective problem [9], as is resource management on the platform [10, 11].

When the desired balance between objectives is not known, the *Pareto-optimal* frontier is the set of noninferior solutions for each objective, as illustrated in Figure 5.4. For example, the metrics in Table 2.3 (for CRNs) and Table 2.4 (for radar EA) cannot be optimized independently, but rather must be balanced and jointly optimized to accomplish overall mission success. The “best” decision becomes a choice along the Pareto frontier, using one of these common approaches:

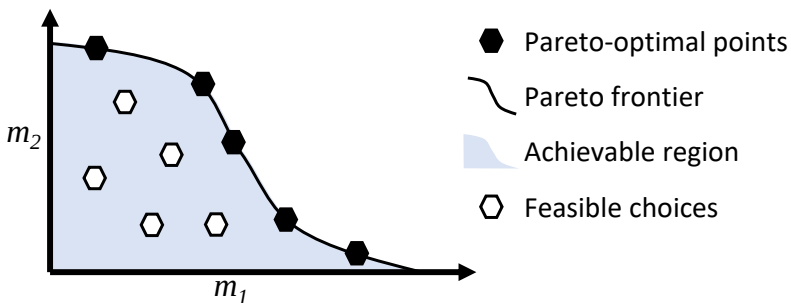
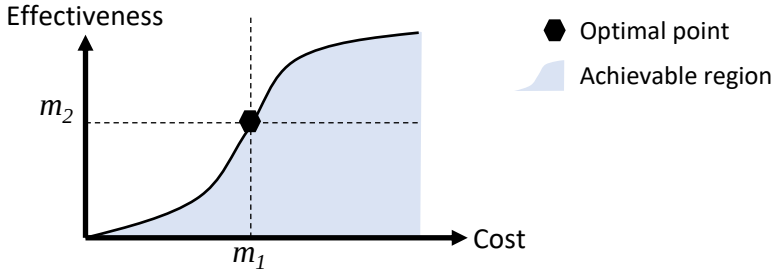


Figure 5.4 Pareto-optimal points occur when no one metric can be improved without degrading other metrics: A higher value for m_1 reduces the value for m_2 .

Callout 5.1 Constraint models support decision making along a Pareto frontier. In contrast, a single objective function would combine all metrics, using negative weights for costs.



When the system operates under minimal operating requirements, it can be beneficial to formulate the problem using one of two methods:

- Maximize Effectiveness subject to constraints on Cost, or
- Minimize Cost subject to constraints on Effectiveness.

A goal of $\text{Cost} \leq m_1$ and $\text{Effectiveness} \geq m_2$ determines an optimal operating point. While mathematically equivalent, these two formulations lead to different practical issues [12, 13].

For example, the *Navy Mission Planner* [14] creates logistically supportable ship employment plans to maximize anticipated mission accomplishment. The linear integer directed network flow optimization model maximizes the total value while satisfying constraints on mobility, mission completion, resource use, and varying effectiveness of combatants. Radar task scheduling can likewise be formulated as a constraint satisfaction problem (e.g., [15]).

1. Balancing mission parameters that define the desired balance in the form of an objective function, for example through weights on the metrics (Section 2.4);
2. Solving a constraint model, where some metrics must meet minimum criteria while others are optimized (Callout 5.1);
3. Computing a probability distribution over the options, for example game theory (Section 6.2); or
4. If there's enough time, showing feasible options to a human operator (Section 6.3).

Table 5.1
Different Techniques to Mitigate a Tone Jammer with Different Expected Effectiveness and Costs in a Communications EP Setting

Strategy	Effectiveness for EP	Cost
Notch filter	Good, but as power goes up, an adaptive notch may remove too much signal	Low
Beamforming	Usually good; poor if interference is distributed	Computation and multiple antennas
DSA	Good, but jammer may follow	Sensing; coordination time
Narrowband hopping	Depends on bandwidth of tone	Coordination time
Redundant packets	Good if tone is intermittent; poor otherwise	Power (and half the throughput)
Spreading	Poor	More gain in receive path increases vulnerability to saturation
Routing around	Medium	Power (multiple nodes); latency

As an example, consider a tone jammer. There are many different techniques that can mitigate the interference it causes; each technique has a different expected benefit and cost for EP objectives (Table 5.1). Even in this simple case, the choice is not always straightforward. First, specific details of the jammer or the environment may reduce the expected benefit of the technique, and therefore the system should learn to estimate the performance based on these observables (Section 4.3). Second, different costs may be more or less tolerable to the platform and mission. Finally, different stages of the mission may have different priorities and thus mandate a different choice, even for techniques that have equal expected effectiveness. Adding to the decision-making complexity, each of these EP techniques may also have benefits or costs against ES or EA objectives.

5.1.2 Searching through the Performance Landscape

AI tackles the challenge of searching the exponential performance landscape. Traditional mathematics approaches focused on complete, optimal solutions, but computing a solution for most real-world problems (including EW) is impossible. Per our definition of the objective function, EW has an exponential number of strategies ($\prod_{\forall c} v_c$, where v_c is the number of possible values for the control parameter c), quickly approaching infinity. When the number of strategies is small, the system can estimate the utility of every strategy, as outlined in Algorithm 5.1. This code is

Algorithm 5.1 When the number of candidate strategies is small, node n can exhaustively compute the estimated utility for each strategy. This approach effectively chooses EP interference-mitigation approaches for a fielded communications network (Example 1.1, [16]).

For each candidate strategy s_i :	//All candidate strategies for node n
For each metric m_k :	
$m_k = f_k(o, s_i)$	//Estimated metrics
Compute $\tilde{U}(s_i)$ from all metrics m_k	//Estimated utility
Select $s = \operatorname{argmax}_{s_i} \tilde{U}(s_i)$	//Best strategy

compact and efficient, and therefore well-suited to an embedded system that must meet formal safety-certification requirements.

As the number of strategies grows, however, alternative approaches become necessary. If the objective function is smooth and has relatively few local maxima, gradient-based searches are efficient and effective methods to finding a solution.

EW objectives, however, are rarely smooth, not necessarily continuous, and frequently have many local maxima, making *randomized search methods* appropriate [17]. Also known as *metaheuristic methods*, *Monte Carlo methods*, or *stochastic algorithms*, they statistically guarantee an optimal solution; there is typically no way to know whether they have found an optimal solution.

*Optimality, however, is rarely critical;
usually “good” is good enough.*

In EW, the domain is partially observable, and changes so fast that an approximate solution is sufficient. In other words, one can overthink the problem. Nobel Prize-winning economist and father of AI, Herbert A. Simon, developed the concept of *bounded rationality* [18], which says that the rationality of individuals is limited by the information they have, the limitations of their minds, and *the finite amount of time they have to make a decision*. Sections 5.3 and 5.4 continue this line of thought.

Randomized search methods compute the quality of individuals in a population over several iterations before selecting a final choice (Algorithm 5.2). In our case, the population $\mathcal{P}$ is a subset of all candidate strategies available at the node, and $\mathcal{P}$ is much smaller than the total number of available strategies. Randomized algorithms differ in the metaheuristic they use to compute each successive iteration.

Algorithm 5.2 Metaheuristic methods explore the performance landscape using randomized methods that search through a small population of individuals. The utility function $\tilde{U}_n$ evaluates the quality of the individuals.

```

 $\mathcal{P}$  = Compute initial population of candidate strategies for node  $n$ 
For each iteration:                                     //Convergence or max iterations
    For each candidate strategy  $s_i \in \mathcal{P}$ :
        For each metric  $m_k$ :
             $m_k = f_k(o, s_i)$                              //Estimated metrics
        Compute  $\tilde{U}_n(s_i)$                                  //Estimated utility
     $\mathcal{P}$  = Update population (location or individuals)
Select  $s = \operatorname{argmax}_{s_i \in \mathcal{P}} (\tilde{U}_n(s_i))$            //Best strategy in last population

```

This following list of randomized optimization algorithms is *far* from being comprehensive. Here we list ones that have shown promise to the EW space, and mention a few relevant examples. Beheshti and Shamsuddin [19] review population-based metaheuristics, including mathematical formulations and trade-offs. Jamil and Yang [20] present 175 different benchmark problems that validate the performance of optimization algorithms, and (importantly) the characteristics of each. Ramzan et al. [6] present a survey of optimization approaches in the context of CRNs. The NATO *Cognitive Radar* report presents many approaches and applications [21]. These techniques are merely a starting point, and can be combined and augmented with other approaches. Example randomized search methods include:

- *Ant colony optimization* (ACO) is a robust, versatile metaheuristic for solving combinatorial optimization problems, wherein a population of artificial “ants” randomize the search to a solution [22, 23]. As the ants move through the landscape, they leave “pheromones;” higher probability paths have denser pheromones. Updating the population in Algorithm 5.2 for ACO means “moving” the ant to a different strategy.
 - *Pros*: ACO is inherently parallelizable, guarantees convergence, and can be used in dynamic applications.
 - *Cons*: ACO requires careful parameter tuning. Because ACO is a randomized search, time to convergence is uncertain, it is hard to analyze theoretically, and it doesn’t guarantee an optimal solution. Good solutions in the previous iteration increase the probability that

solutions in subsequent iterations will follow a similar path, reducing the diversity of the population.

ACO is used in CRNs to build routing trees [24, 25], assign channels [26], optimize load balancing [27], and tune transmission parameters [28]. ACO can steer nulls and synthesize antenna patterns [29–31]. Ye et al. [32] combines ACO with Tabu search to make collaborative and cooperative jamming decisions. ACO filtering has also been used to track drones [33] and design antennas [34].

- *Particle swarm optimization* (PSO) uses a population (swarm) of candidate solutions (particles) to move through the search space [35]. Particle movements are guided by their own estimate of their position, and the swarm's best-known position.
 - *Pros*: PSO has a simple structure that is easy to encode. PSO is more efficient than genetic algorithms, especially in problems with continuous variables. PSO can operate in dynamic applications.
 - *Cons*: The choice of PSO parameters can have a large impact on optimization performance, and can converge prematurely to local optima, especially with complex problems. Stochastic variability can be high.

PSOs have been used to optimize performance and resource allocation in CRNs [6, 36, 37]. PSO works well for radar pulse compression codes [38, 39], and designing discrete frequency waveforms [40, 41]. PSO has been used for jamming suppression [42], designing active decoy clusters [43], optimizing range-velocity gate pull-off jamming [44] jointly optimizing flight plans and jamming resources [45], and topology estimation for target surveillance [46].

- *Genetic algorithms* (GAs) are another population-based randomized search method in which solutions are encoded as “chromosomes” [47]. This meta-heuristic uses mutation, crossover, and selection to generate high-quality solutions to complex problems.
 - *Pros*: GAs are naturally suited for discrete-valued problems, and are easily parallelized.
 - *Cons*: GAs are prone to premature convergence; maintaining diversity in the population is critical [48–50]. GA problem-encoding is problem-specific, and these encodings can be difficult to create. GAs are less computationally efficient than PSO [51].

GAs are used to compute multiparameter solutions for transmission, with various goals including power management, transmission performance, and shared spectrum usage [6, 52]. GAs have been used for radar design [53], waveform selection [54], waveform design [55, 56], target identification [57, 58], radar scheduling [59], jammer suppression [60], distributed jamming [61], and cellular antenna and radar placement [62, 63].

- *Bayesian optimization* (BO) is a global optimization technique that models an unknown objective function using a probabilistic surrogate model, typically a Gaussian process [64, 65]. It iteratively selects new sample points based on an acquisition function that balances exploration and exploitation.
 - *Pros*: BO is efficient for expensive black-box functions where evaluations are costly. It provides an uncertainty-aware search strategy, avoiding premature convergence. BO is particularly well-suited for low-dimensional problems with noisy evaluations.
 - *Cons*: BO scales poorly with high-dimensional search spaces, as Gaussian process regression is computationally expensive. It requires careful selection of the acquisition function and can struggle in dynamic environments where the objective function changes over time.

BO won the 2020 NeurIPS black-box optimization competition [66–68]; the code is open-source. BO has been used in RF for clustering in sensor networks [69], resource allocation [70–72], localization [73], radar cross section minimization [74], synthesizing of predistortion filters for optical transmitters [75], and to optimize DeepNets for beamforming [76]. BO has also been used to design amplifiers [77] and integrated systems [78].

- *Simulated annealing* is another search metaheuristic that probabilistically decides whether to move from one solution to another [79]. Unlike GAs and PSO, there is only one search instance. The probability function is usually chosen so that the probability of deciding to move decreases when the difference in solution quality increases; that is, small uphill moves are more likely than large uphill moves. It can be used within GAs and PSO to improve search time.
 - *Pros*: Simulated annealing is relatively easy to code, and generally gives a good solution. (It statistically guarantees finding an optimal solution.) Because simulated annealing has only one individual, it requires significantly less memory than population-based approaches.

- *Cons:* For the same reason, it is more likely to end up in the same valley it started in; it doesn't know whether it has found an optimal solution.

Simulated annealing has been used for topics as varied as detecting faults in antennas [80], meeting the quality-of-service for a radio network [81], estimating radar cross-sections [82], time-domain coding for radar coexistence [83], localizing jammers [84], and suppressing jamming [85, 86]. Simulated annealing and GAs have been combined for power allocation in CRNs [87], radar imaging [88], clutter suppression [89], and designing a MIMO sensor array [90]. Simulated annealing and PSO merge to optimize waveforms for sparse frequency-agile radars [91]. Simulated annealing has been combined with other AI techniques, for example case-based reasoning, to increase the convergence rate [92].

- *Cross-entropy method* (CEM) is a randomized search method for importance sampling; it draws a sample from a probability distribution, and then minimizes the cross-entropy between this distribution and the target distribution to produce a better sample in the next iteration [93, 94]. Amos and Yarrats [95] describe a method to differentiate CEM so that it can be used with gradient descent methods.
 - *Pros:* CEM defines a precise mathematical framework—in some sense optimal learning rules—which enable strong theoretical guarantees for convergence. It is robust to parameter settings. CEM is valuable for rare event simulation, where very small probabilities need to be accurately estimated.
 - *Cons:* Time to convergence is uncertain; the stopping condition is a heuristic. It requires a large number of samples to estimate probability distributions accurately. The original CEM method has high storage and computational requirements [96]. CEM is less commonly used for real-time or dynamic problems because it optimizes a probability distribution based on past samples, which may become outdated before convergence.

CEM is used to estimate and optimize network reliability [93, 97], find network paths [98], track multiple targets [99], and place nulls [100]. Naeem et al. [101] use CEM to optimize cooperative communication and greenhouse gas emissions. CEM can also be used in the antenna design problem [102, 103], and in RL to learn more quickly [94, 96]. Like simulated annealing,

CEM can be combined with the population-based approaches to parallelize solutions [104, 105]. CEM accelerates convergence by guiding search with an adaptive probability model rather than relying on pure heuristics. It can also provide theoretical justifications and improve the stability, predictability, and robustness of metaheuristic-driven solutions.

One of the more interesting applications of AI-based optimization is for design and configuration of reconfigurable intelligent surfaces (RIS) and metamaterials. Callout 5.2 describes this emerging application.

Despite optimization being a mature field, new approximation techniques are developed regularly. Multiple conferences host regular optimization competitions, including GECCO [123, 124], IEEE CEC [125, 126], and NeurIPS [67, 127]; the code is usually released open-source. Moreover, individual optimization approaches can be combined to get better performance (speed or solution quality). Many other approaches can likewise help, including hybrid approaches (Section 3.3) and metalearning (Section 3.5). For example, while the performance surface is in theory infinitely large, EW engagements are constrained by physics and progression of engagement state that can be updated by collection or denial of information (see Section 6.1.4). Heuristics can leverage this connected state space, and use randomized methods and active learning (Section 7.3.2) to explore the remaining unknown regions.

5.2 Scheduling

Scheduling maps a partially ordered plan to specific resources and timeslots. Scheduling is concerned with when and how to do things. In EW, scheduling commonly furnishes the specifics of when to transmit and when to receive.

A resource scheduler on a single EW node may decide when to activate sensors, when and how to transmit, when and how to avoid specific electromagnetic signatures, and/or when and how to receive. For distributed EW systems, such particulars (of how and when) only grow in complexity. An EW resource scheduler needs (1) a sequence of actions and their resource requirements, (2) a set of resources to use, (3) constraints, and (4) an objective function.

Critical path methods (CPM) build a directed graph relating actions; the critical path is the one with the total longest duration, and determines the maximum duration of the entire EW mission. Any task that lays on the critical path is called a *critical task*. Critical tasks and their interdependencies can be represented with a Gantt chart or similar approaches. Critical tasks are spectrum sensing, spectrum analysis, and spectrum decision (Figure 5.5); an EW mission encompasses numerous cognition cycle iterations.

Callout 5.2 Metasurfaces enable precise manipulation of electromagnetic waves, unlocking innovations in sensing, stealth, communications, radar, and EW.

Metasurfaces are engineered materials composed of strategically placed unit cells, designed to manipulate electromagnetic waves and exhibit tailored electromagnetic responses. Imagine a mirror that is engineered to only reflect certain wavelengths or angles of light; all other reflected colors and phases are omitted (via absorption or scattering). Where a traditional mirror reflects all visible wavelengths, a metasurface mirror is selective about what it reflects. The proper engineering of their tiny, resonant unit cells enables customizable responses in the transmission, reflection, and absorption of energy [106].

By carefully designing unit cells according to Maxwell's laws of electromagnetics, each unit cell can embed reconfigurable components addressed independently through *feeding lines* (the RF power conduit between the radio and antenna), allowing for the tailoring of electromagnetic wavefronts [107]. By designing the gradient index (GRIN) of the selected material, one can create a parabolic phase gradient allowing for custom beam shaping [108].

Metasurfaces can manipulate polarization through selective absorption or phase control [109, 110]. They enable asymmetric transmission [111], imposing different effects on waves depending on their direction of travel. In other words, metasurfaces allow engineers to create “magic” mirrors for various wavelengths of energy, such as in antenna radomes that permit outgoing transmissions while blocking incoming signals.

Any sufficiently advanced technology is indistinguishable from magic.

— *Arthur C. Clarke, 19 Jan 1968 [112]*

These engineered surfaces can dynamically increase or decrease the intensity of the electromagnetic response of a target, reducing its detectability [113] or enhancing its signature for radar tracking [114]. Metasurfaces can perform frequency-selective beam-splitting, directing different frequency components along distinct paths, enabling compact direction-finding systems [115]. These functionalities, among others, unlock new capabilities for EW engineers.

Metasurfaces have a low size, weight, power, and cost (SWaP-C) profile, making them scalable, cost-effective, and compatible with conventional electronic components when implemented using printed circuit board technology. This manufacturing approach enables the development of compact, multiband, and multifunctional antennas for lightweight drones.

As a result, advanced EW capabilities become economically feasible for drone swarms [116], while also providing increased sensitivity for off-axis incident energy or signals [108]. For example, metasurfaces can enhance a drone's electromagnetic signature to mislead tracking systems, even deceiving resilient radar technologies such as monopulse systems. In a swarm scenario, metasurfaces can manipulate radar perception, enabling drones to evade detection or mislead surveillance systems. Moreover, controlling which drones are detectable can create confusing scenarios for cognitive radars.

The low SWaP-C features of metasurfaces also make them candidates for multiagent cognitive applications with drone swarms. In RL terminology, highlighting the drones represents an action a affecting the target radar's perception. Perceiving the various operating modes of the target radar—its state s —requires sensing capabilities in the drones.

Another application of metasurfaces lies in dynamically shaping electromagnetic wavefronts to optimize communication and sensing systems. Strategically placing metasurfaces in urban environments allows precise control over wave propagation, optimizing signal delivery and reducing interference in communication networks. Emerging 6G research identifies this approach as a key element of the physical layer to optimize user device connectivity and capacity [117]. This optimization occurs in a closed-loop setting where a controller dynamically adjusts metasurface configurations based on real-time feedback from the transmitter and receiver [118]. Metasurfaces can improve signal interception while minimizing the likelihood of detection. Furthermore, metasurfaces can selectively manipulate spectrum access, potentially preventing adversaries from establishing reliable communication links. Ultimately, the optimization loop must leverage AI to predict how the metasurfaces should be adjusted, based on partial information regarding the communication environment [116, 119].

AI will play a critical role as we realize the utility of fully reconfigurable *phase changing metamaterials (PCMMs)* for EW and camouflage, concealment and deception related systems and platforms. A PCM is a material with the natural ability to transition its state (physical or chemical) such as a polymer that can switch from a hard plastic to a gaseous state; a PCMM is an engineered structure using PCMs. PCMMs exploit PCMs to achieve tunable electromagnetic properties. PCMMs controlled and commanded by AI will tailor phase and state changes based on both learned and instantaneous data sets. Various randomized search algorithms have shown utility in solving this problem [120–122], often combined with RL (Section 7.3).

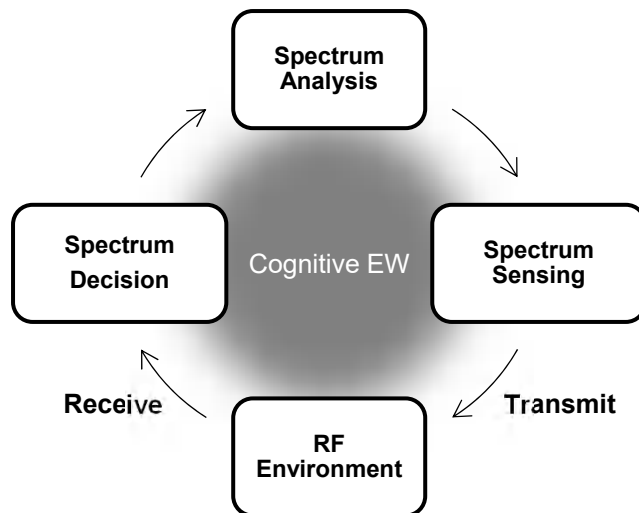


Figure 5.5 Each of the high-level tasks—sensing, analysis, and decision making—drive the transmit/receive schedule.

In computing, the first scheduling approaches were taken from the field of operations management. Assembly lines and many other human aspects of life require scheduling, sharing the same concerns with computing, including a desire for efficiency. The development of scheduling policies is concerned with determining what the key assumptions should be and what metrics are important. In computer science, metrics include *workload*, *jobs*, *turnaround time*, and *fairness* [128]. Today, many scheduling disciplines exist and are found in solutions for network routers, operating systems, disk drives, embedded systems, and packet-switched wireless networks.

In EW systems, the scheduler chooses actions that will ultimately enable the EW system to transmit and to receive. The scheduler can perform actions in a cyclical or linear manner. Linear schedules are finite, whereas cyclical ones naturally repeat. Ideally, an awareness of all possible states is necessary for proper task scheduling; however, partial observability of some states could make it difficult to accomplish a given task. To address numerous constraints such as output power, energy efficiency, and time requirements, the scheduler may assign weights or costs to these various constraints thus eliminating the need to preplan manually, an arduous task and often impossible given the timing constraints of the system.

In automatic scheduling, all tasks may have to undergo an automatic prioritization of execution order, but the nature of tasks themselves might have to be predetermined by the CogEW system designer. The designer's goal is to create a scheduler that achieves all of its scheduling objectives within the mission and operating constraints. In distributed systems, the decision should be based upon nodes' responses (or lack thereof) to determine if some tasks need to be abandoned altogether or whether another node should be allowed to carry out the task. Lack of true broad spectrum "visibility" in spacetime will also determine the boundaries of one's scheduling algorithm. Awareness of many states, past, current, or pre-planned/predicted future, allows the scheduler to adjust its parameters when executing the tasks (i.e., adjusting its overall master schedule). For real-time CogEW systems, time is of the essence. Thus, near real-time decisions must determine the appropriate order of tasks to execute, based on many internal and external factors such as sensor data, weather, antenna array availability or, generally, resource availability. A scheduler should adjust the schedule based on resource availability.

Modern commercial communications systems, for example, have schedulers that allow them to prioritize when to transmit and receive and in what frequencies and time slots, such as frequency division multiple access (FDMA) or time division multiple access (TDMA) schemes. DSA is also an attractive feature for any EW system to possess and should be a part of the system's scheduling algorithm.

The scheduler decides in which order to execute system tasks, while assessing resource availability and determining how to utilize these resources. The scheduler must rely on situation assessment information and reorder tasking based on the information received. While a simple scheduler can be entirely deterministic, such as rule-based or hard-coded with predetermined inputs and outputs, the chosen approach must support scheduling to account for unexpected, unknown, or partially observable states.

Lastly, overall awareness of the mission environment is important for successful execution of system tasks (whatever they may be) in the best possible order. Flexibility is key; prioritization and reprioritization are also equally important. Valuable information may sometimes arrive at an inconvenient time, but cannot be ignored and should be recognized by the scheduler. The scheduler may also benefit from having a "fight or flight" capability or an emergency mode when the ongoing schedule might have to be abandoned in lieu of an emergency to instantiate a new tasks' order to protect the system.

Table 5.2 provides several examples from the literature on various schedulers that are relevant to CogEW. Many of these algorithms use randomized search methods and/or RL as part of the solution approach.

Table 5.2
Examples of EW Scheduling Solutions

Category	Examples
Comms	Resource allocation [6]; spectrum sensing [129, 130]; transmission scheduling for MANETs, CRNs, 5G, and satellites [131–137]; energy management [138]; uplink timing and power-control [139]; user scheduling for multi-antenna systems and covert communication [140, 141]; routing (CPM) [142, 143]; task management in edge networks (CPM) [144]; scheduling for distributed sensor networks (CPM) [145, 146].
Radar	Phased array task scheduling [147]; active electronically scanned array (AESA) antenna scheduling [148]; multifunctional radar network scheduling [149]; multifunction phased array radar scheduling [150]; multifunction radar [151] resource allocation for both transmit and receive [152]; task scheduling [15, 153, 154]; multiradar collaborative surveillance [155].
EW	Jamming for spectrum sharing [156]; game-based approach for jamming resources and strategy optimization [157]; FSR and power control for EP [142]; airborne ECM [158]; periodic sensor scheduling for ES [159]; risk-based multisensor scheduling for target threat assessment [160]; multitarget tracking [161–163]; jamming networked radars [164].
Multifunction	Task scheduling for joint communication and radar [165–168]; minimizing communication interference from radar signals [169]; queue scheduling [170]; spectrum sharing [171].
General framework	Scheduling based on knowledge-based engineering [172]; heterogeneity-driven task scheduling and end-to-end synchronized scheduling algorithms for networked embedded systems [173]; utility accrual real-time scheduler based on polynomial-time heuristic algorithm [174]; RL scheduler for heterogeneous distributed systems [175]; generalized ML-based solution for large-scale resource scheduling [176]; online resource scheduling algorithm using Deep RL [177]; ML approaches to learning heuristics for combinatorial optimization [178]; scheduling approaches using evolutionary computation [179]; temporal reasoning for planning and scheduling [180].

5.3 Reward Hacking

Reward hacking occurs when the decision maker exploits a flaw in the objective function to achieve high rewards in unintended—and usually unacceptable—ways [181, 182]. For example, if a car salesman is told to “sell many cars,” they might price the cars at \$1.00, exploiting a loophole in the objective. If a CogEW system on an airplane has been trained on the objective to “avoid getting hit by a missile,” it may choose to crash; the learner perfectly accomplishes the poorly

specified objective. The optimizer can find loopholes in the objective function, exploit bugs in the models or simulators, and fall prey to the training process (e.g., insufficient exploration) [183].

It is very hard to specify good objective functions, so loopholes tend to be many and varied. Defining objective functions is hard because it requires specifying all acceptable behaviors, which is impractical. Reward specification is hard even for experts, and making them align with end-users' expectations is particularly challenging due to the varied and latent user preferences [184]. It is much easier for humans to identify unacceptable behavior than it is to enumerate all of the acceptable options.

The following strategies can reduce the likelihood of encountering catastrophic failure operationally:

- *Develop the algorithm in rapid increments.* The development of ML algorithms, especially those that learn in-mission, must proceed iteratively in small increments. It is not only *expected* that the learner will creatively find unacceptable solutions, it is also *encouraged*. We want to ensure that unacceptable behavior occurs early in the development effort, in a low-fidelity modeling and simulation environment, so we can correct problems in the objective function before the system becomes operational.
- *Monitor for aberrant behavior.* Humans can identify and flag undesirable behavior during iterative development and test. Anomaly detection techniques can also mitigate misalignment.
- *Learn the objective function.* Inverse RL can learn the hidden reward function (Section 6.3.3). The learner can learn a near-optimal function, even from non expert teachers. If a trusted (but not-optimal) reward is available, it can be used to support the training process [182].
- *Optimize only to “good enough” rather than “optimal.”* There are operational, mathematical, and engineering reasons to not always find the optimal solution.

This final point bears further discussion. From the *operational* perspective, decisions that seem non optimal “in the moment” can lead to better outcomes over the course of a mission. For example, sacrificing a pawn in a chess game may appear to be a poor choice at first, but if it helps save the queen, it could ultimately lead to winning the game. Moreover, game theory approaches intentionally select semi-random actions to prevent the adversary from predicting future moves (Section 6.2).

From the *mathematical* perspective, optimality is hard to measure. Physics models may suggest an optimal solution, but in reality, it is an incomplete model, and

therefore true optimality can never be determined. Moreover, the domain is partially observable and changes extremely fast (in picoseconds). Any optimal solution is likely to be irrelevant before it can be implemented.

From the *engineering* perspective, many EW systems are specified to have certain performance against the most stressing threats. There is a lot of design margin. For example, a radar employing adaptive beamforming may be specified to create null depth of, say 50 dB, to counter a noise jammer. The 50 dB is typically chosen to counter the most powerful jammer at the closest range—creating the highest effective isotropic radiated power—at the closest angle to the desired target. Therefore, if the learning system chooses adaptive beamforming as the correct EP action, it will likely still be effective against a very large population of jammers because of the extra margin built into the system for this mode, even if some of the parameters aren't optimal, for example sample support for secondary data estimation.

Second, multiple objectives create trade-offs that usually require none to be independently optimal. Being perfect in one objective leads to other objectives failing, for example having perfect EA leads to fratricide in the EP mission.

Third, in the networked environment, it is often better to make a locally good decision than to attempt to find a globally optimal decision that is irrelevant before it can be implemented. The objective function needs to be constructed so that the agents can be fully independent, collaborating and coordinating when communications allow (Section 5.5).

Finally, when the objective function isn't quite right, over-optimization leads to reward hacking. It is better to have a good, generalizable solution than to have an optimal solution that memorizes specific conditions.

5.4 Anytime Algorithms

EW decision-making algorithms operate in rapidly changing environments with hard real-time requirements. New priorities can “pop up” at any time, and resources may be unexpectedly depleted. A desirable trait for a decision maker is that it can generate solutions quickly, or, when time is available, spend more time deliberating. While it may be a goal to have zero-latency for decisions, in practice, a longer integration time (and waiting a second or two) may yield a better result.

Anytime algorithms find better and better solutions the longer they run. Invented by Dean and Boddy in the 1980s [185], anytime algorithms can be interrupted at any point during computation to return a result whose utility is a function of computation time. The similar concept of *flexible computation* [186] explicitly balances the benefits of additional computation time against the costs

of acting with a partial solution. Both ideas have roots in Herb Simon's concept of *satisficing* [187], which involves searching for a solution until an acceptability threshold is met; satisficing is particularly useful when optimal solutions cannot be determined because of computational intractability or lack of information. Longer observation times may generate better quality observations, and longer compute times may yield better optimization. Anytime algorithms recognize that the time required to find the optimal solution typically reduces the overall utility, similar to the law of diminishing returns.

Most metaheuristic algorithms satisfy this condition: Each iteration improves the quality of the best-available solution, and therefore can be stopped as needed. Adding to the list of citations from Section 5.1.2, anytime algorithms are effective in domains where approximate solutions need to be generated quickly, including progressive signal detection [188], incrementally designing anti-jamming strategies [189], refining adversarial attacks over time [190], and adaptive weather-control radars [191].

Zilberstein looks at the problem of metacontrol: compiling, controlling, and managing multiple anytime algorithms [192, 193]. Zilberstein, who used a radar threat analysis and target-assignment problem as one of his motivating scenarios, said:

The radar component, designed to track objects, and the planning component, designed to perform target assignment, are clearly interdependent: the quality of the first clearly affects the quality of the second. The use of conditional performance profiles and dynamic scheduling seems essential for solving such problems.

— Shlomo Zilberstein, 1993 [192]

Metalearning (Section 3.5) augments this work to learn and utilize performance predictions [194–196].

5.5 Centralized, Distributed, and Decentralized Optimization

An EW system may need to optimize across multiple decision makers on a single platform, or across platforms. A single node of a distributed EW system may be aware of all of its states within its world's purview yet it may not be aware of other nodes' or actors' states. Partial accomplishment of tasks can be achieved in time, locally, on each node, but coordination among nodes is crucial as nodes might not be striving toward the same goal and might end up being in conflict with each other's objectives. For example, the decision maker must be cognizant of limited node connectivity and visibility, validate that tasks have been successfully accomplished, and manage accountability for wrongly executed tasks (reward and punishment).

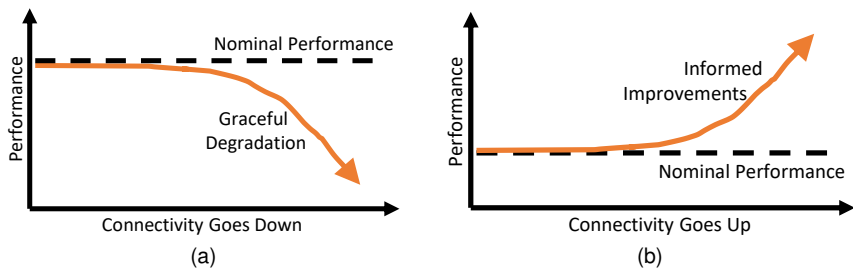


Figure 5.6 (a) Graceful degradation assumes that communications are safe and plentiful with occasional problems, while (b) informed improvements assumes a no-communication baseline, and leverages communications if available.

Both radar and comms systems require distributed decision making, but comms systems have greater latency and more coordination. Approaches differ according to the nature of the solution:

- *Centralized*: A single decision maker finds a solution for all components and all nodes;
- *Distributed*: Decision makers use local communications to coordinate actions;
- *Decentralized*: Decision makers are fully independent, and do not rely on communications for coordination.

One of our failings as a community is that we design systems for *graceful degradation*. This approach assumes that communications are safe and plentiful. This assumption is not true in EW where communications is limited or denied due to jamming, mobility disconnects, or decisions to reduce RF emissions. Instead, we must design systems for *informed improvements*, operating under the assumption is that there will be no communications at all, leveraging communications if and when available. Figure 5.6 illustrates the difference between these two concepts.

Figure 5.7 illustrates the potential impact of communications on planning for a suppression of enemy air defense (SEAD) task. As communications become more available, more of the lower-priority threats can be neutralized. Decentralized optimization approaches can structure the problem so that even under limited or no communication, more of the common objectives can be fulfilled.

5.5.1 Centralized Optimization

On a single platform, a single optimizer will find generally a solution that is closer to optimal than multiple optimizers because it can search for a single

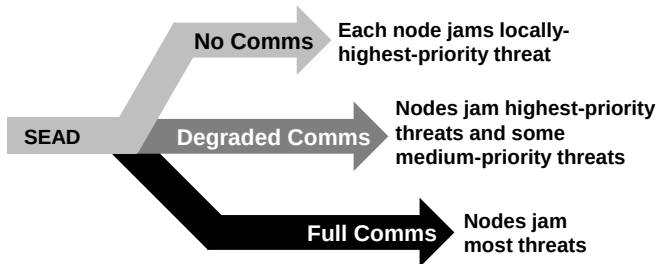


Figure 5.7 Task accomplishment depends on available communications for coordination. Designing for decentralized operation, and leveraging communication if available, improves the ability to accomplish tasks.

global solution. However, a single optimizer is more challenging to design due to differing timescales, differing types of information, and amount of information to synchronize.

Across the network of platforms, a single, centralized optimizer is inappropriate. The critical concern is that of a single bottleneck: If the centralized optimizer fails, the entire system fails. Additional concerns include the latency of decision making, information inconsistencies and overload, information privacy, and detectable emissions.

*It is better to make a locally good decision quickly
than it is to attempt to find a globally optimal decision
that is irrelevant before it can be implemented.*

Figure 5.8 illustrates the practical limits of a centralized optimization approach.

In EW, centralized nodes can capture “slower-time” information, including changes to the broader mission policy, EWO interactions (Section 6.3), and slower information dissemination and task management [197, 198] (Section 6.1.6).

5.5.2 Distributed Optimization

Distributed optimization approaches communicate across local neighborhoods to coordinate actions. Section 6.1.4 discusses approaches for planning when to communicate—managing the benefit of information sharing against the cost of transmission. Several surveys describe distributed optimization solutions, noting communications overhead, memory usage, and optimality [199–201].

Gossip algorithms, such as *consensus propagation* [202] are a lightweight communications approach that allows each node to share its local estimate of the global utility, and quickly compute the global average, despite not knowing how

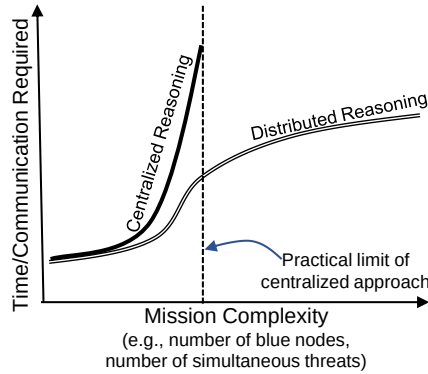


Figure 5.8 Centralized reasoning quickly reaches a practical limit.

many nodes are in the network. ACO algorithms likewise have low communications demands.

5.5.3 Decentralized Optimization

Fully decentralized coordination requires each node to independently evaluate its contribution to the global solution.

Nodes can coordinate without communicating.

One option is to formulate the utility function so that each node's local optimum accurately captures its contribution to the team's global optimum [203–206]. Up to a certain group-level achievement, it is possible to achieve a consensus for agents with heterogeneous preferences using decentralized model-free learning [207]; after that level, “free riders” appear and the solution does not converge to a group optimum. The formal problem definition of Callout 2.1 uses a fully local approximation of the true utility; the BBN SO uses this function to choose EP interference-mitigation approaches for a communications network in an online learning environment (Example 1.1, [16]). Molzhan et al. [208] conducted a survey on decentralized optimization in electric power systems, against the backdrop of classical control theory and real-time feedback control.

Communication-free learning [209, 210] formulates the problem so that local sensing is sufficient to determine whether the solution is valid; the authors describe how to model graph coloring, channel assignment with channel-dependent interference, intersession network coding, and decentralized transmission scheduling. Each

node needs to know only its own assignments; it doesn't need to know the choices of the other nodes.

Nodes can also use *implicit communication* to send information. This approach assumes that other nodes can observe actions or changes to the state, and infer what happened without needing to explicitly interact. This method dovetails with the active communication concepts of Section 6.1.4.

5.5.4 Summary

Distributed and decentralized optimization approaches support collaboration (sharing information) and coordination (sharing tasks). Examples include information fusion [191], resource allocation [197, 209, 211], and scheduling [145]. Section 6.1.4 highlights more of the information sharing activities. A common application for distributed optimization is timing synchronization, which then can be used for data fusion, duty cycling, cooperative localization, and coordinated actions such as distributed beamforming [212–214].

Many distributed and decentralized optimization solutions are naturally anytime algorithms, making them inherently suitable for generating satisficing solutions in this dynamic domain.

5.6 Conclusion

Just as there is no free lunch for ML and statistical inference [215], there is also no free lunch for optimization [216]: If an algorithm performs well on a certain class of problems then it necessarily pays for that with degraded performance on the set of all remaining problems.

This chapter presented some of the optimization approaches shown to be effective for EW problems:

- They support multiple objectives;
- They rapidly search the exponential performance surface;
- They handle a dynamic environment;
- They improve solution quality when given more time; and
- They operate well for multiple nodes under limited communication.

Chapter 6 discusses more strategic decision making: long-term battle planning and management.

References

- [1] Arrow, K., "Decision Theory and Operations Research," *Operations Research*, Vol. 5, No. 6, 1957. DOI: 10.1287/opre.5.6.765.

- [2] Russell, S., and P. Norvig, *Artificial Intelligence: A Modern Approach*. London, UK: Pearson Education, 2015.
- [3] Chien, S., et al., "Automated Planning and Scheduling for Goal-Based Autonomous Spacecraft," *Intelligent Systems*, Vol. 13, No. 5, 1998. DOI: 10.1109/5254.722362.
- [4] R-Moreno, M., et al., "RFID Technology and AI Techniques for People Location, Orientation and Guiding," in *Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 2009. DOI: 10.1007/978-3-642-02568-6_40.
- [5] Du, D., P. Pardalos, and W. Wu, "History of Optimization," in *Encyclopedia of Optimization*, Floudas, C., and P. Pardalos, (Eds.), New York, NY: Springer, 2009. Online: <https://tinyurl.com/optimization-encyclopedia>.
- [6] Ramzan, M., et al., "Multi-Objective Optimization for Spectrum Sharing in Cognitive Radio Networks: A Review," *Pervasive and Mobile Computing*, Vol. 41, 2017. DOI: 10.1016/j.pmcj.2017.07.010.
- [7] Cassady, P., *Bayesian Analysis of Target Detection with Enhanced Receiver Operating Characteristic*, 2019. Online: <https://arxiv.org/abs/1903.08165>.
- [8] Chalise, B. K., M. G. Amin, and B. Himed, "Performance Tradeoff in a Unified Passive Radar and Communications System," *IEEE Signal Processing Letters*, Vol. 24, No. 9, 2017. DOI: 10.1109/LSP.2017.2721639.
- [9] Chiriyath, A., et al., "Radar Waveform Optimization for Joint Radar Communications Performance," *Electronics*, Vol. 8, No. 12, DOI: 10.3390/electronics8121498.
- [10] AlQerm, I., and B. Shihada, "Adaptive Multi-Objective Optimization Scheme for Cognitive Radio Resource Management," in *GLOBECOM*. IEEE, 2014. DOI: 10.1109/GLOCOM.2014.7036916.
- [11] Mitchell, A., et al., "Cost Function Design for the Fully Adaptive Radar Framework," *IET Radar, Sonar and Navigation*, 2018. DOI: 10.1049/iet-rsn.2018.5327.
- [12] Haigh, K. Z., O. Olofinboba, and C. Y. Tang, "Designing an Implementable User-Oriented Objective Function for MANETs," in *Networking, Sensing and Control*. IEEE, 2007. Online: <http://www.cs.cmu.edu/~khaigh/papers/Haigh07-ICNSC.pdf>.
- [13] Charlish, A., and F. Hoffman, "Cognitive Radar Management," in *Novel Radar Techniques and Applications*, Klemm, R., et al., (Eds.), Raleigh, NC: SciTech Publishing, 2017. DOI: 10.1049/SBRA512G_ch5.
- [14] Brown, G., and J. Kline, "Optimizing Navy Mission Planning," *Military Operations Research*, Vol. 26, No. 2, 2021.
- [15] Akbar, S., A. Hebb, and R. Adve, "Development of Cognitive Approaches to Radar Scheduling, Detection and Tracking," Defence R&D Canada, Tech. Rep. DRDC-RDDC-2023-C199, 2023. Online: https://cradpdf.drdc-rddc.gc.ca/PDFS/unc437/p817000_A1b.pdf.
- [16] Haigh, K. Z., et al., "Parallel Learning and Decision Making for a Smart Embedded Communications Platform," BBN Technologies, Tech. Rep. BBN-REPORT-8579, 2015. Online: <http://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-SO.sm.pdf>.
- [17] Zabinsky, Z., "Random Search Algorithms," in *Wiley Encyclopedia of Operations Research and Management Science*, 2011. DOI: 10.1002/9780470400531.eorms0704.
- [18] Simon, H., "A Behavioral Model of Rational Choice," *Quarterly Journal of Economics*, Vol. 69, No. 1, 1955. DOI: 10.2307/1884852.

- [19] Beheshti, Z., and S. Shamsuddin, "A Review of Population-Based Meta-Heuristic Algorithm," *International Journal of Advances in Soft Computing and its Applications*, Vol. 5, No. 1, 2013. Online: <https://tinyurl.com/metaheuristics>.
- [20] Jamil, M., and X.-S. Yang, "A Literature Survey of Benchmark Functions for Global Optimization Problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, Vol. 4, No. 2, 2013. Online: <https://arxiv.org/abs/1308.4008>.
- [21] Task Group SET-227, "Cognitive Radar," NATO Science, and Technology, Tech. Rep. TR-SET-227, 2020. Online: <https://tinyurl.com/nato-set-227-cr>.
- [22] Dorigo, M., "Optimization, Learning and Natural Algorithms," Ph.D. dissertation, Politecnico di Milano, Milan, Italy, 1992.
- [23] Katiyar, S., I. Nasiruddin, and A. Ansari, "Ant Colony Optimization: A Tutorial Review," in *National Conference on Advances in Power and Control*, 2015. Online: <https://tinyurl.com/aco-tutorial>.
- [24] Almasoud, A., and A. Kama, "Multi-Objective Optimization for Many-to-Many Communication in Cognitive Radio Networks," in *GLOBECOM*. IEEE, 2015. Online: <https://www.ece.iastate.edu/~kamal/Docs/mk15b.pdf>.
- [25] Zhang, Q., Q. He, and P. Zhang, "Topology Reconfiguration in Cognitive Radio Networks Using Ant Colony Optimization," in *Vehicular Technology Conference*. IEEE, 2012. DOI: 10.1109/VTCFall.2012.6398990.
- [26] He, Q., and P. Zhang, "Dynamic Channel Assignment Using Ant Colony Optimization for Cognitive Radio Networks," in *Vehicular Technology Conference*. IEEE, 2012. DOI: 10.1109/VTCFall.2012.6398951.
- [27] Alam, M., et al., "Ant Colony Optimization-Based Solution to Optimize Load Balancing and Throughput for 5G and Beyond Heterogeneous Networks," *EURASIP Journal on Wireless Communications and Networking*, Vol. 2024, No. 1, 2024. DOI: 10.1186/s13638-024-02376-2.
- [28] Waheed, M., and A. Cai, "Cognitive Radio Parameter Adaptation in Multicarrier Environment," in *Wireless and Mobile Communications*. IEEE, 2009. DOI: 10.1109/ICWMC.2009.72.
- [29] Andalani, A., and S. Baghaiee, "Application of Ant Colony Optimization Algorithm to Pattern Synthesis of Uniform Circular Antenna Array," *Applied Computational Electromagnetics Society*, Vol. 30, No. 8, 2015.
- [30] Karaboga, N., K. Güneş, and A. Akdagli, "Null Steering of Linear Antenna Arrays with Use of Modified Touring Ant Colony Optimization Algorithm," *International Journal of RF and Microwave Computer-Aided Engineering*, Vol. 12, No. 4, 2002. DOI: 10.1002/mmce.10034.
- [31] Zhu, Z., Y. Wu, and J. Hao, "Research on MIMO Beam Integration Optimization Method Based on PredRNN Prediction and Ant Colony Algorithm," in *Antennas, Propagation and EM Theory*, 2024. DOI: 10.1109/ISAPE62431.2024.10840496.
- [32] Ye, F., F. Che, and L. Gao, "Multiobjective Cognitive Cooperative Jamming Decision-Making Method Based on Tabu Search-Artificial Bee Colony Algorithm," *International Journal of Aerospace Engineering*, 2018. DOI: 10.1155/2018/7490895.
- [33] Mandal, P., L. P. Roy, and S. K. Das, "Tracking of Invader Drone using Hybrid Unscented Kalman-Continuous Ant Colony Filter (HUK-CACF)," *ISA Transactions*, Vol. 152, 2024. DOI: 10.1016/j.isatra.2024.06.018.
- [34] Bahlouane, I., et al., "Design and Optimization of Microstrip Patch Antenna using Ant Colony Optimization," in *Global Aeronautical Engineering and Satellite Technology*, 2024. DOI: 10.1109/GAST60528.2024.10520747.

- [35] Kennedy, J., and R. Eberhart, "Particle Swarm Optimization," in *Neural Networks*, Vol. 4, 1995. DOI: 10.1109/ICNN.1995.488968.
- [36] Zou, S., et al., "Efficiency-optimized 6G: A Virtual Network Resource Orchestration Strategy by Enhanced Particle Swarm Optimization," *Digital Communications and Networks*, Vol. 10, No. 5, 2024. DOI: 10.1016/j.dcan.2023.06.008.
- [37] Agrawal, A., et al., "Particle Swarm Optimization versus Simulated Annealing for 5G and 6G Wireless Networks: A Comparative Study," in *Wireless Personal Multimedia Communications*, 2024. DOI: 10.1109/WPMC63271.2024.10863550.
- [38] Hafez, A., and M. El-latif, "New Radar Pulse Compression Codes by Particle Swarm Algorithm," in *IEEE Aerospace Conference*, 2012. DOI: 10.1109/AERO.2012.6187187.
- [39] Li, B., "Particle Swarm Optimization for Radar Binary Phase Code Selection," in *Radar Sensor Technology*, 2018. DOI: 10.1117/12.2305768.
- [40] Reddy, B., and U. Kumari, "Performance Analysis of MIMO Radar Waveform using Accelerated Particle Swarm Optimization Algorithm," *Image Processing an International Journal*, Vol. 3, No. 4, 2012. DOI: 10.5121/sipij.2012.3416.
- [41] Praveena, A., and V. Narasimhulu, "Design of DFC Waveforms for MIMO Radar using Accelerated Particle Swarm Optimization Algorithm," *International Journal of Engineering Trends and Technology*, Vol. 33, No. 2, 2016. Online: <http://www.ijettjournal.org/archive/ijett-v33p220>.
- [42] Pan, Y., et al., "Optimization of Intermittent Sampling and Repeating Jamming Waveform Based on DPSO," in *Signal, Information and Data Processing*, 2024. DOI: 10.1109/ICSIDP62679.2024.10869226.
- [43] Wu, Z., Y. Luo, and S. Hu, "Optimization of Jamming Formation of USV Offboard Active Decoy Clusters Based on an Improved PSO Algorithm," *Defence Technology*, Vol. 32, 2024. DOI: 10.1016/j.dt.2023.03.017.
- [44] Ma, Z., et al., "Strategy Optimization for Range-Velocity Gate Pull-off Jamming," *IEEE Transactions on Aerospace and Electronic Systems*, 2024. DOI: 10.1109/TAES.2024.3514593.
- [45] Jiang, Z., et al., "Joint Optimization of Flight Planning and Jamming Resource Allocation in UAV Swarm against Radar Network," in *Signal, Information and Data Processing*, 2024. DOI: 10.1109/ICSIDP62679.2024.10868622.
- [46] Wei, J., et al., "Topology Optimization of UAV Network for Target Surveillance Task with Support Jamming," *Signal Processing*, Vol. 224, 2024. DOI: 10.1016/j.sigpro.2024.109583.
- [47] Holland, J., "Genetic Algorithms and Adaptation," in *Adaptive Control of Ill-Defined Systems*, Vol. 16, 1984. DOI: 10.1007/978-1-4684-8941-5_21.
- [48] Meadows, B., et al., "Evaluating the Seeding Genetic Algorithm," in *Australasian Joint Conference on Artificial Intelligence*, 2013. DOI: 10.1007/978-3-319-03680-9_24.
- [49] Watson, T., and P. Messer, "Increasing Diversity in Genetic Algorithms," in *Developments in Soft Computing*, 2001. DOI: 10.1007/978-3-7908-1829-1_14.
- [50] Chuang, C.-Y., and S. Smith, "Diversity Allocation for Dynamic Optimization Using the Extended Compact Genetic Algorithm," in *Congress on Evolutionary Computation*. IEEE, 2013. DOI: 10.1109/CEC.2013.6557745.
- [51] Hassan, R., et al., *A Copmarison [sic] of Particle Swarm Optimization and the Genetic Algorithm*, 2004. Online: <https://tinyurl.com/ps0-ga-compare>.
- [52] Mehboob, U., et al., "Genetic Algorithms in Wireless Networking: Techniques, Applications, and Issues," *Soft Computing*, 2016. DOI: 10.1007/s00500-016-2070-9.

- [53] Barteel, J., "Genetic Algorithms As A Tool for Phased Array Radar Design," M.S. thesis, Naval Postgraduate School, Monterey, CA, 2002. Online: <https://apps.dtic.mil/dtic/tr/fulltext/u2/A404735.pdf>.
- [54] Capraro, C., et al., "Using Genetic Algorithms for Radar Waveform Selection," in *IEEE Radar Conference*, 2008. DOI: 10.1109/RADAR.2008.4720947.
- [55] Lellouch, G., A. K. Mishra, and M. Inggs, "Design of OFDM Radar Pulses Using Genetic Algorithm Based Techniques," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 52, No. 4, 2016. DOI: 10.1109/Taes.2016.140671.
- [56] Sen, S., G. Tang, and A. Nehorai, "Multiobjective Optimization of OFDM Radar Waveform for Target Detection," *IEEE Transactions on Signal Processing*, Vol. 59, No. 2, 2011. DOI: 10.1109/TSP.2010.2089628.
- [57] Jouny, I., "Radar Target Identification Using Genetic Algorithms," in *Automatic Target Recognition*, 1998. DOI: 10.1117/12.323832.
- [58] Smith-Martinez, B., A. Agah, and J. Stiles, "A Genetic Algorithm for Generating Radar Transmit Codes to Minimize the Target Profile Estimation Error," *Journal of Intelligent Systems*, Vol. 22, No. 4, 2013. DOI: 10.1515/jisys-2013-0028.
- [59] Dam, T. M., et al., "Efficient Radar Scheduling Using Genetic Algorithms and Stochastic Heuristic Initialization," in *Intelligent Data Engineering and Automated Learning*, 2024. Online: https://link.springer.com/chapter/10.1007/978-3-031-77731-8_18.
- [60] Zhou, C., F. Liu, and Q. Liu, "An Adaptive Transmitting Scheme for Interrupted Sampling Repeater Jamming Suppression," *Sensors*, Vol. 17, No. 11, 2017. DOI: 10.3390/s17112480.
- [61] Tang, C., J. Ding, and L. Zhang, "LEO Satellite Downlink Distributed Jamming Optimization Method Using a Non-Dominated Sorting Genetic Algorithm," *Remote Sensing*, Vol. 16, No. 6, 2024. DOI: 10.3390/rs16061006.
- [62] Calles-Esteban, F., et al., "Optimizing Antenna Positioning for Enhanced Wireless Coverage: A Genetic Algorithm Approach," *Sensors*, Vol. 24, No. 7, 2024. DOI: 10.3390/s24072165.
- [63] Tema, E. Y., S. Sahmoud, and B. Kiraz, "Radar placement optimization based on adaptive multi-objective meta-heuristics," *Expert Systems with Applications*, Vol. 239, 2024. DOI: 10.1016/j.eswa.2023.122568.
- [64] Frazier, P., *A Tutorial on Bayesian Optimization*, 2018. Online: <https://arxiv.org/abs/1807.02811>.
- [65] Wang, X., et al., "Recent Advances in Bayesian Optimization," *ACM Comput. Surv.*, Vol. 55, No. 13s, 2023. DOI: 10.1145/3582078.
- [66] HEBO: Heteroscedastic Evolutionary Bayesian Optimization. Online: <https://www.youtube.com/watch?v=Jrujn9PHpyQ>. Code available at <https://github.com/huawei-noah/HEBO>.
- [67] *NeurIPS Competition Track*, 2020. Online: <https://neurips.cc/Conferences/2020/CompetitionTrack>. Accessed 2025-02-21.
- [68] Cowen-Rivers, A., et al., "HEBO: Pushing The Limits of Sample-Efficient Hyperparameter Optimisation," Vol. 74, 2022. DOI: 10.1613/jair.1.13643.
- [69] Zhang, T., et al., "Bayesian-Optimization-Based Peak Searching Algorithm for Clustering in Wireless Sensor Networks," *Journal of Sensor and Actuator Networks*, Vol. 7, No. 1, 2018. DOI: 10.3390/jsan7010002.
- [70] Maggi, L., A. Valcarce, and J. Hoydis, "Bayesian Optimization for Radio Resource Management: Open Loop Power Control," *IEEE Journal on Selected Areas in Communications*, Vol. 39, No. 7, 2021. DOI: 10.1109/JSAC.2021.3078490.

- [71] De Freitas, A., et al., "Global Optimization for Resource Allocation in a Networked-Surveillance Radar," in *Information Fusion*, 2019. DOI: 10.23919/FUSION43075.2019.9011303.
- [72] Yan, J., Q. Lu, and G. B. Giannakis, "Bayesian Optimization for Online Management in Dynamic Mobile Edge Computing," *IEEE Transactions on Wireless Communications*, Vol. 23, No. 4, 2024. DOI: 10.1109/TWC.2023.3307875.
- [73] Jagadeesan, N., and B. Krishnamachari, "A Unifying Bayesian Optimization Framework for Radio Frequency Localization," *IEEE Transactions on Cognitive Communications and Networking*, Vol. 4, No. 1, 2018. DOI: 10.1109/TCCN.2017.2768065.
- [74] Bilal, A., et al., "Fast Wideband Wide-Angle RCS Minimization Using ISAR Imaging and Bayesian Optimization," in *Asia-Pacific Microwave Conference*, 2024. DOI: 10.1109/APMC60911.2024.10867659.
- [75] Sena, M., et al., "Bayesian Optimization for Nonlinear System Identification and Pre-Distortion in Cognitive Transmitters," *Journal Lightwave Technology*, Vol. 39, No. 15, 2021. Online: <https://opg.optica.org/jlt/abstract.cfm?URI=jlt-39-15-5008>.
- [76] Yoo, H., et al., "Deep Learning-Based Beamforming Design with Bayesian Optimization," *The Journal of Korean Institute of Electromagnetic Engineering and Science*, Vol. 36, No. 1, 2025. DOI: 10.5515/KJKIEES.2025.36.1.86.
- [77] Chen, P., B. M. Merrick, and T. J. Brazil, "Bayesian Optimization for Broadband High-Efficiency Power Amplifier Designs," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 63, No. 12, 2015. DOI: 10.1109/TMTT.2015.2495360.
- [78] Torun, H. M., et al., "A Global Bayesian Optimization Algorithm and Its Application to Integrated System Design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 26, No. 4, 2018. DOI: 10.1109/TVLSI.2017.2784783.
- [79] Kirkpatrick, S., C. Gelatt Jr., and M. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, 1983. DOI: 10.1126/science.220.4598.671.
- [80] Boopalan, N., A. Ramasamy, and F. Nagi, "Faulty Antenna Detection in a Linear Array Using Simulated Annealing Optimization," *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 19, No. 3, 2020. DOI: 10.11591/ijeecs.v19.i3.pp1340-1347.
- [81] Kaur, K., M. Rattan, and M. Patterh, "Optimization of Cognitive Radio System Using Simulated Annealing," *Wireless Personal Communications*, Vol. 71, 2013. DOI: 10.1007/s11277-012-0874-1.
- [82] White, R., "Simulated Annealing Algorithm for Radar Cross-Section Estimation and Segmentation," in *Applications of Artificial Neural Networks V*. International Society for Optics, and Photonics, Vol. 2243, 1994. DOI: 10.1117/12.169970.
- [83] Hillbom, V., *Simulated Annealing Design of Time-Domain Coding for FMCW Radar Coexistence*, 2024. Online: <http://lup.lub.lu.se/student-papers/record/9166760>. Student Paper, Lund University, Sweden.
- [84] Liu, Z., et al., "Error Minimizing Jammer Localization through Smart Estimation of Ambient Noise," in *Mobile Ad-Hoc and Sensor Systems*, 2012. DOI: 10.1109/MASS.2012.6502530.
- [85] Wang, Y., and S. Zhu, "Main-Beam Range Deceptive Jamming Suppression with Simulated Annealing FDA-MIMO Radar," *IEEE Sensors Journal*, Vol. 20, No. 16, 2020. DOI: 10.1109/JSEN.2020.2982194.

- [86] Xu, Z., et al., "The GNSS Blind Anti-Jamming Method Based on Simulated Annealing Algorithm," in *Electronics and Communication, Network and Computer Technology*, 2024. DOI: 10.1109/ECNCT63103.2024.10704272.
- [87] Zhao, J., X. Guan, and X. Li, "Power Allocation Based on Genetic Simulated Annealing Algorithm in Cognitive Radio Networks," *Chinese Journal of Electronics*, Vol. 22, No. 1, 2012. Online: <https://tinyurl.com/crn-power-allocation>.
- [88] Yang, G., et al., "W-Band MIMO Radar Array Optimization and Improved Back-projection Algorithm for Far-Field Imaging," in *Infrared, Millimeter, and Terahertz Waves*, 2019. DOI: 10.1109/IRMMW-THz.2019.8874531.
- [89] Shi, J., et al., "Pulse Interval Optimization for Doppler Ambiguity Clutter Suppression in Missile-Borne STAP Radar," *IEEE Geoscience and Remote Sensing Letters*, Vol. 21, 2024. DOI: 10.1109/LGRS.2024.3371438.
- [90] Sayin, A., E. G. Hoare, and M. Antoniou, "Design and Verification of Reduced Redundancy Ultrasonic MIMO Arrays Using Simulated Annealing & Genetic Algorithms," *IEEE Sensors Journal*, Vol. 20, No. 9, 2020. DOI: 10.1109/JSEN.2020.2964774.
- [91] Chen, S., et al., "A Waveform Optimization Method for Sparse Frequency Agile Radar," in *Signal, Information and Data Processing*, 2024. DOI: 10.1109/ICSIDP62679.2024.10868098.
- [92] Liu, Y., et al., "A Self-Learning Method for Cognitive Engine Based on CBR and Simulated Annealing," in *Advanced Materials and Engineering Materials*, Vol. 457, 2012. DOI: 10.4028/www.scientific.net/AMR.457-458.1586.
- [93] Rubinstein, R., and D. Kroese, *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. New York, NY: Springer, 2004.
- [94] de Boer, P.-T., et al., *A Tutorial on the Cross-Entropy Method*, 2003. Online: http://web.mit.edu/6.454/www/www_fall_2003/gew/CEtutorial.pdf.
- [95] Amos, B., and D. Yarats, "The Differentiable Cross-Entropy Method," in *ICML*, 2020. Online: <https://arxiv.org/abs/1909.12830>.
- [96] Joseph, A., and S. Bhatnagar, "Revisiting the Cross Entropy Method with Applications in Stochastic Global Optimization and Reinforcement Learning," in *European Conference on AI*, 2016. DOI: 10.3233/978-1-61499-672-9-1026.
- [97] Kroese, D., K.-P. Hui, and S. Nariai, "Network Reliability Optimization via the Cross-Entropy Method," *IEEE Transactions on Reliability*, Vol. 56, No. 2, 2007. DOI: 10.1109/TR.2007.895303.
- [98] Heegaard, P., B. Helvik, and O. Wittner, "The Cross Entropy Ant System for Network Path Management," *Teletronikk*, 2008. Online: <https://tinyurl.com/cem-network>.
- [99] Qiang, G., "Bearings Fusion using the Cross Entropy Method for Underwater Multitarget Tracking," in *Information Communication and Signal Processing*, 2024. DOI: 10.1109/ICICSP62589.2024.10809069.
- [100] Bian, L., "Parameters Analysis to Pattern Null Placement Based on the Cross Entropy Method," *Physics Procedia*, Vol. 24, No. B, 2012. DOI: 10.1016/j.phpro.2012.02.214.
- [101] Naeem, M., et al., "Green Cooperative Cognitive Radio: A Multiobjective Optimization Paradigm," *IEEE Systems Journal*, Vol. 10, No. 1, 2016. DOI: 10.1109/JSYST.2014.2301952.
- [102] Minvielle, P., et al., "Sparse Antenna Array Optimization with the Cross-Entropy Method," *IEEE Transactions on Antennas and Propagation*, Vol. 59, No. 8, 2011. DOI: 10.1109/TAP.2011.2158941.

- [103] Connor, J., "Antenna Array Synthesis Using the Cross Entropy Method," Ph.D. dissertation, Florida State University, Tallahassee, FL, 2008. Online: http://purl.flvc.org/fsu/fd/FSU_migr_etd-3444.
- [104] Heegaard, P., et al., *Distributed Asynchronous Algorithm for Cross-Entropy-Based Combinatorial Optimization*, 2003. Online: <https://tinyurl.com/ce-ants-2003>.
- [105] Helvik, B., and O. Wittner, "Using the Cross-Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks," in *Workshop on Mobile Agents for Telecommunication Applications*, 2001. DOI: 10.1007/3-540-44651-6.24.
- [106] Bukhari, S., J. Vardaxoglou, and W. Whittow, "A Metasurfaces Review: Definitions and Applications," *MDPI Applied Sciences*, Vol. 9, No. 2727, 2019. DOI: 10.3390/app9132727.
- [107] Liu, C., et al., "Reconfigurable Metasurface: A Systematic Categorization and Recent Advances," *Electromagnetic Science*, Vol. 1, No. 4, 2023. DOI: 10.23919/emsci.2023.0002.
- [108] Stewart, J., et al., "Holographic Optics for Increased Detection Efficiency," in *SPIE Defense and Commercial Sensing*, 2021. DOI: 10.1117/12.2587830.
- [109] Begaud, X., et al., "Ultra-wideband and Wide-angle Microwave Metamaterial Absorber," *MDPI Materials*, Vol. 11, No. 10, 2018. DOI: 10.3390/ma11102045.
- [110] Yuan, Y., et al., "Independent Phase Modulation for Quadruplex Polarization Channels Enabled by Chirality-assisted Geometric-phase Metasurfaces," *Nature Communications*, Vol. 11, No. 4186, 2020. DOI: 10.1038/s41467-020-17773-6.
- [111] Ji, W., et al., "High-Efficiency and Ultra-Broadband Asymmetric Transmission Metasurface Based on Topologically Coding Optimization Method," *Optics Express*, Vol. 27, No. 3, 2019. DOI: 10.1364/OE.27.002844.
- [112] Clarke, A. C., "Clarke's Third Law on UFO's," *Science*, No. 3812, 1968.
- [113] Zhao, J., et al., "A Tunable Metamaterial Absorber Using Varactor Diodes," *New Journal of Physics*, Vol. 15, No. 4, 2013. DOI: 10.1088/1367-2630/15/4/043049.
- [114] Taravati, S., and G. Eleftheriades, "Full-Duplex Reflective Beamsteering Metasurface Featuring Magnetless Nonreciprocal Amplification," *Nature Communications*, Vol. 12, No. 4414, 2021. DOI: 10.1038/s41467-021-24749-7.
- [115] Huang, M., et al., "Machine-Learning-Enabled Metasurface for Direction of Arrival Estimation," *Nanophotonics*, Vol. 11, No. 9, 2022. DOI: 10.1515/nanoph-2021-0663.
- [116] Gonçalves Licursi de Mello, R., *Metasurface-Driven Electronic Warfare*. Hoboken, NJ: Wiley, 2024.
- [117] Basar, E., et al., "Reconfigurable Intelligent Surfaces for 6G: Emerging Hardware Architectures, Applications, and Open Challenges," *IEEE Vehicular Technology Magazine*, Vol. 19, No. 3, 2024. DOI: 10.1109/MVT.2024.3415570.
- [118] Mohjazi, L., et al., "An Outlook on the Interplay of Artificial Intelligence and Software-Defined Metasurfaces: An Overview of Opportunities and Limitations," *IEEE Vehicular Technology Magazine*, Vol. 15, No. 4, 2020. DOI: 10.1109/MVT.2020.3023550.
- [119] Qian, C., I. Kaminer, and H. Chen, "A Guidance to Intelligent Metamaterials and Metamaterials Intelligence," *Nature Communications*, Vol. 16, No. 1, 2025. DOI: 10.1038/s41467-025-56122-3.
- [120] Zhang, L., et al., "RIS-Assisted Simultaneous Legitimate Monitoring and Jamming for Industrial Wireless Networks," *CoRR*, Vol. abs/2412.18749, 2024. DOI: 10.48550/ARXIV.2412.18749.

- [121] Zhang, C., et al., "Compatible Metasurface for Ultra-wideband Radar and Switchable Infrared Stealth," *Opt. Express*, Vol. 32, No. 18, 2024. DOI: 10.1364/OE.533691.
- [122] Aziz, M. M., A. Habib, and A. Zafar, *Waveform Design for Radar Anti Jamming with RIS Using Reinforcement Learning*, 2024. Available at SSRN: <https://ssrn.com/abstract=4861098> or <http://dx.doi.org/10.2139/ssrn.4861098>.
- [123] Škvorc, U., T. Eftimov, and P. Korošec, "GECCO Black-box Optimization Competitions: Progress from 2009 to 2018," in *Genetic and Evolutionary Computation Conference Companion*, 2019. DOI: 10.1145/3319619.3321996.
- [124] *GECCO Competitions*, 2025. Online: <https://gecco-2025.sigevo.org/Competitions>. Accessed 2025-02-21.
- [125] *IEEE Congress on Evolutionary Computation Competitions*, 2023. Online: <https://2023.ieee-ccc.org/competitions/>. Accessed 2025-02-21.
- [126] *IEEE CEC Competition on Dynamic Optimization Problems*, 2025. Online: <https://competition-hub.github.io/GMPB-Competition/>. Accessed 2025-02-21.
- [127] *NeurIPS Machine Learning for Combinatorial Optimization*, 2021. Online: <https://neurips.cc/virtual/2021/competition/32547>. Accessed 2025-02-21.
- [128] Arpaci-Dusseau, R., and A. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*. Madison, WI: Arpaci-Dusseau Books, 2018.
- [129] Chauhan, P., et al., "Utility Driven Cooperative Spectrum Sensing Scheduling for Heterogeneous Multi-channel Cognitive Radio Networks," *Telecommunication Systems*, Vol. 78, No. 1, 2021. DOI: 10.1007/s11235-021-00785-y.
- [130] Salih, S., M. Suliman, and A. Mohammed, "A Novel Spectrum Sensing Scheduler Algorithm for Cognitive Radio Networks," in *Computing, Electrical and Electronic Engineering*, 2013. DOI: 10.1109/ICCEEE.2013.6633961.
- [131] Bayhan, S., and F. Alagoz, "Scheduling in Centralized Cognitive Radio Networks for Energy Efficiency," *IEEE Transactions on Vehicular Technology*, Vol. 62, No. 2, 2013. DOI: 10.1109/TVT.2012.2225650.
- [132] Liang, J.-C., and J.-C. Chen, "Resource Allocation in Cognitive Radio Relay Networks," *Selected Areas in Communications*, Vol. 31, No. 3, 2013. DOI: 10.1109/JSAC.2013.130313.
- [133] Rainish, D., and A. Freedman, *Air Interface for Low Power Operation of a Satellite Terminal*, 2015. Online: <https://www.satixfy.com/vlnsr-implementation-for-mobile/>.
- [134] Zhu, L., et al., "The Design of Scheduling Algorithm for Cognitive Radio Networks Based on Genetic Algorithm," in *IEEE Computational Intelligence Communication Technology*, 2015. DOI: 10.1109/CICT.2015.131.
- [135] Bollapragada Subrahmanya, V., and H. Russell, "RMTS: A Novel Approach to Transmission Scheduling in Ad Hoc Networks by Salvaging Unused Slot Transmission Assignments," *Wireless Communications and Mobile Computing*, 2018. DOI: 10.1155/2018/2484897.
- [136] Kilic, S., and O. Ozkan, "Modeling and Optimization Approaches for Satellite Broadcast Scheduling Problem," *IEEE Transactions on Engineering Management*, 2019. DOI: 10.1109/TEM.2019.2937947.
- [137] Jaradat, A. M., et al., "Radar-Aided Communication Scheduling Algorithm for 5G and Beyond Networks," *IEEE Access*, Vol. 10, 2022. DOI: 10.1109/ACCESS.2022.3205641.
- [138] El Gamal, A., et al., "Energy-Efficient Scheduling of Packet Transmissions over Wireless Networks," in *Computer and Communications Societies. IEEE*, Vol. 3, 2002. DOI: 10.1109/INFCOM.2002.1019431.

- [139] Dahlman, E., S. Parkvall, and J. Sköld, "Chapter 11 - Uplink Physical-Layer Processing," in *4G LTE/LTE-Advanced for Mobile Broadband*, Academic Press, 2011. DOI: 10.1016/B978-0-12-385489-6.00011-4.
- [140] Pattanayak, P., and P. Kumar, "Computationally Efficient Scheduling Schemes for Multiple Antenna Systems Using Evolutionary Algorithms and Swarm Optimization," in *Evolutionary Computation in Scheduling*. 2020. DOI: 10.1002/9781119574293.ch5.
- [141] Shi, J., et al., "User Scheduling Design for Covert Communication in Cooperative Cognitive Radio System," *IEEE Sensors Journal*, Vol. 23, No. 5, 2023. DOI: 10.1109/JSEN.2023.3238002.
- [142] Grilo, A., et al., "Electronic Protection and Routing Optimization of MANETs Operating in an Electronic Warfare Environment," *Ad Hoc Networks*, Vol. 5, No. 7, 2007. DOI: 10.1016/j.adhoc.2006.05.002.
- [143] Varma, A. K., and D. Mitra, "A Novel Cloud Based Self-adaptive Cognitive Radio Network Architecture," *AEU - International Journal of Electronics and Communications*, Vol. 106, 2019. DOI: 10.1016/j.aecue.2019.04.016.
- [144] Xu, B., et al., "Iterative Dynamic Critical Path Scheduling: An Efficient Technique for Offloading Task Graphs in Mobile Edge Computing," *Applied Sciences*, Vol. 12, No. 6, 2022. DOI: 10.3390/app12063189.
- [145] Zhang, W., et al., "Distributed Stochastic Search and Distributed Breakout: Properties, Comparison and Applications to Constraint Optimization Problems in Sensor Networks," *Artificial Intelligence*, Vol. 161, No. 1, 2005. DOI: 10.1016/j.artint.2004.10.004.
- [146] Le, D.-T., et al., "Critical-Path Aware Scheduling for Latency Efficient Broadcast in Duty-Cycled Wireless Sensor Networks," *Wireless Communications and Mobile Computing*, Vol. 2018, 2018. DOI: 10.1155/2018/5017319.
- [147] Zhang, H., et al., "A Hybrid Adaptively Genetic Algorithm for Task Scheduling Problem in the Phased Array Radar," *European Journal of Operational Research*, 2019. DOI: 10.1016/j.ejor.2018.07.012.
- [148] Sahin, S., and T. Girici, "A Method for Optimal Scheduling of Active Electronically Scanned Array (AESA) Antennas," 2019. Online: https://tgirici.files.wordpress.com/2019/05/ew2019_sevdasahin.pdf.
- [149] Li, X., et al., "A Scheduling Method of Generalized Tasks for Multifunctional Radar Network," in *Control, Automation and Information Sciences*. IEEE, 2019. DOI: 10.1109/ICCAIS46528.2019.9074597.
- [150] Orman, A., et al., "Scheduling for a Multifunction Phased Array Radar System," *European Journal of Operational Research*, Vol. 90, No. 1, 1996. DOI: 10.1016/0377-2217(95)00307-X.
- [151] Shaghghi, M., R. Adve, and Z. Ding, "Multifunction Cognitive Radar Task Scheduling Using Monte Carlo Tree Search and Policy Networks," *IET Radar, Sonar and Navigation*, Vol. 12, No. 12, 2018. Online: <https://arxiv.org/abs/1805.07069>.
- [152] Nguyen, H., et al., "Cognitive Radar Scheduler Architecture," in *IEEE Radar Conference*, 2024. DOI: 10.1109/RadarConf2458775.2024.10548605.
- [153] Xu, L., and T. Zhang, "Reinforcement Learning based Dynamic Task Scheduling for Multifunction Radar Network," in *IEEE Radar Conference*, 2020. DOI: 10.1109/RadarConf2043947.2020.9266684.
- [154] George, T., K. Wagner, and P. Rademacher, "Deep Q-Network for Radar Task-Scheduling Problem," in *IEEE Radar Conference*, 2022. DOI: 10.1109/RadarConf2248738.2022.9764230.

-
- [155] Yi, W., et al., "Recent Advances in Multi-Radar Collaborative Surveillance: Cognitive Tracking and Resource Scheduling Algorithms," *Journal of Radars*, Vol. 12, No. 3, 2023. DOI: 10.12000/JR23036.
 - [156] Jiang, X., et al., "Intelligent Jamming Strategies for Secure Spectrum Sharing Systems," *IEEE Transactions on Communications*, Vol. 70, No. 2, 2022. DOI: 10.1109/TCOMM.2021.3140082.
 - [157] Ren, Y., et al., "A Novel Cognitive Jamming Architecture for Heterogeneous Cognitive Electronic Warfare Networks," in *Information Science and Applications*, Vol. 621, New York, NY: Springer, 2020. DOI: 10.1007/978-981-15-1465-4.11.
 - [158] Song, H., et al., "A POMDP Approach for Scheduling the Usage of Airborne Electronic Countermeasures in Air Operations," *Aerospace Science and Technology*, Vol. 48, 2016. DOI: 10.1016/j.ast.2015.11.001.
 - [159] Vaughan, I., and L. Clarkson, "Optimal Periodic Sensor Scheduling in Electronic Support," in *Defence Applications of Signal Processing*, 2005. Online: <http://staff.itee.uq.edu.au/vaughan/Publications/dasp-04a.pdf>.
 - [160] Zhang, Y., and G. Shan, "A Risk-Based Multisensor Optimization Scheduling Method for Target Threat Assessment," *Mathematical Problems in Engineering*, 2019. DOI: 10.1155/2019/2043727.
 - [161] Sun, H., et al., "Joint Radar Scheduling and Beampattern Design for Multitarget Tracking in Netted Colocated MIMO Radar Systems," *IEEE Signal Processing Letters*, Vol. 28, 2021. DOI: 10.1109/LSP.2021.3108675.
 - [162] Dai, J., et al., "Integrated Trajectory Planning and Resource Scheduling for Multiple Target Tracking in Airborne Radar Network," *IEEE Sensors Journal*, Vol. 24, No. 9, 2024. DOI: 10.1109/JSEN.2024.3379384.
 - [163] Hu, T., et al., "Resource Scheduling for Multitarget Imaging in a Distributed Netted Radar System Based on Maximum Scheduling Benefits," *Sensors*, Vol. 22, No. 17, 2022. DOI: 10.3390/s22176400.
 - [164] Zhang, D., et al., "Joint Jamming Beam and Power Scheduling for Suppressing Netted Radar System," in *IEEE Radar Conference*, 2021. DOI: 10.1109/RadarConf2147009.2021.9455337.
 - [165] Ju, H., et al., "Adaptive Scheduling for Joint Communication and Radar Detection: Tradeoff Among Throughput, Delay, and Detection Performance," *IEEE Transactions on Vehicular Technology*, Vol. 71, No. 1, 2022. DOI: 10.1109/TVT.2021.3123618.
 - [166] Zhang, J. A., et al., "Enabling Joint Communication and Radar Sensing in Mobile Networks—A Survey," *IEEE Communications Surveys & Tutorials*, Vol. 24, No. 1, 2022. DOI: 10.1109/COMST.2021.3122519.
 - [167] Rong, J., et al., "Adaptive Task Scheduling Algorithm for Multifunction Integrated System with Joint Radar–Communications Waveform," *Electronics*, Vol. 12, No. 7, 2023. DOI: 10.3390/electronics12071560.
 - [168] Xie, Z., et al., "Optimal Scheduling Policy for Time-Division Joint Radar and Communication Systems: Cross-Layer Design and Sensing for Free," *IEEE Internet of Things Journal*, Vol. 10, No. 23, 2023. DOI: 10.1109/JIOT.2023.3305977.
 - [169] Su, Y., et al., "Communication-awareness Adaptive Resource Scheduling Strategy for Multiple Target Tracking in a Multiple Radar System," *IET Signal Processing*, 2022. DOI: 10.1049/sil2.12097.

- [170] Yang, H., et al., "Queue-Aware Dynamic Resource Allocation for the Joint Communication-Radar System," *IEEE Transactions on Vehicular Technology*, Vol. 70, No. 1, 2021. DOI: 10.1109/TVT.2020.3042551.
- [171] Tolley, J., C. Makin, and C. B. Dietrich, "Systematization of Knowledge: Spectrum Sharing Between Radar and Communications," *IEEE Access*, Vol. 11, 2023. DOI: 10.1109/ACCESS.2023.3339579.
- [172] Rajpathak, D., "Intelligent Scheduling—A Literature Review," Knowledge Media Institute, The Open University, Milton Keynes, UK, Tech. Rep. KMI-TR-119, 2001. Online: <https://tinyurl.com/kmi-tr-119>.
- [173] Xie, G., R. Li, and K. Li, "Heterogeneity-Driven End-to-End Synchronized Scheduling for Precedence Constrained Tasks and Messages on Networked Embedded Systems," *Journal of Parallel and Distributed Computing*, Vol. 83, 2015. DOI: 10.1016/j.jpdc.2015.04.005.
- [174] Balli, U., et al., "Utility Accrual Real-Time Scheduling under Variable Cost Functions," *IEEE Transactions on Computers*, Vol. 56, No. 03, 2007. DOI: 10.1109/TC.2007.52.
- [175] Orhean, A., F. Pop, and I. Raicu, "New Scheduling Approach Using Reinforcement Learning for Heterogeneous Distributed Systems," *Journal of Parallel and Distributed Computing*, Vol. 117, 2018. DOI: 10.1016/j.jpdc.2017.05.001.
- [176] Yang, R., et al., "Intelligent Resource Scheduling at Scale: A Machine Learning Perspective," in *Symposium on Service-Oriented System Engineering*. IEEE, 2018. DOI: 10.1109/SOSE.2018.00025.
- [177] Ye, Y., et al., *A New Approach for Resource Scheduling with Deep Reinforcement Learning*, 2018. Online: <https://arxiv.org/abs/1806.08122>.
- [178] Mirshekarian, S., and D. Sormaz, "Machine Learning Approaches to Learning Heuristics for Combinatorial Optimization Problems," *Procedia Manufacturing*, Vol. 17, 2018. DOI: 10.1016/j.promfg.2018.10.019.
- [179] Gandomi, A., et al., (Eds.), *Evolutionary Computation in Scheduling*. Hoboken, NJ: Wiley, 2020.
- [180] Barták, R., R. Morris, and B. Venable, "An Introduction to Constraint-Based Temporal Reasoning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Vol. 8, No. 1, 2014. DOI: 10.2200/S00557ED1V01Y201312AIM026.
- [181] Skalse, J., et al., "Defining and Characterizing Reward Hacking," 2022. DOI: 10.48550/arXiv.2209.13085.
- [182] Pan, A., K. Bhatia, and J. Steinhardt, "The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models," in *ICLR*, 2022. DOI: 10.48550/arXiv.2201.03544.
- [183] *Specification Gaming Examples in AI - Master List*. Online: <https://tinyurl.com/reward-hacking-examples>. Accessed 2025-02-22.
- [184] Abouelazm, A., J. Michel, and J. M. Zöllner, "A Review of Reward Functions for Reinforcement Learning in the Context of Autonomous Driving," in *IEEE Intelligent Vehicles Symposium*, 2024. DOI: 10.1109/IV55156.2024.10588385.
- [185] Dean, T., and M. Boddy, "An Analysis of Time-Dependent Planning," in *AAAI*, 1988. Online: <https://dl.acm.org/doi/10.5555/2887965.2887974>.
- [186] Horvitz, E., "Reasoning about Beliefs and Actions under Computational Resource Constraints," in *Workshop on Uncertainty in Artificial Intelligence*, 1987. Online: <http://erichorvitz.com/u87.htm>.

- [187] Simon, H., "Rational Choice and the Structure of the Environment," *Psychological Review*, Vol. 63, No. 2, 1956. DOI: 10.1037/h0042769.
- [188] Pemasiri, A., et al., "Automatic Radar Signal Detection and FFT Estimation using Deep Learning," in *Signal Processing and Communication System*, 2024. Online: <https://arxiv.org/abs/2402.19073>.
- [189] Li, H., et al., "Counterfactual Regret Minimization for Anti-Jamming Game of Frequency Agile Radar," in *Sensor Array and Multichannel Signal Processing Workshop*, 2022. DOI: 10.1109/SAM53842.2022.9827883.
- [190] Ma, R., et al., *Concealed Electronic Countermeasures of Radar Signal with Adversarial Examples*, 2023. Online: <https://arxiv.org/abs/2310.08292>.
- [191] Kim, Y., M. Krainin, and V. Lesser, "Application of Max-Sum Algorithm to Radar Coordination and Scheduling," in *Workshop on Distributed Constraint Reasoning*, 2010. Online: <http://mas.cs.umass.edu/paper/486>.
- [192] Zilberstein, S., "Operational Rationality through Compilation of Anytime Algorithms," Ph.D. dissertation, University of California, Berkeley, CA, 1993. Online: <https://www2.eecs.berkeley.edu/Pubs/TechRpts/1993/6276.html>.
- [193] Zilberstein, S., "Using Anytime Algorithms in Intelligent Systems," *AI Magazine*, 1996. Online: <http://rbr.cs.umass.edu/shlomo/papers/Zaimag96.pdf>.
- [194] Svegliato, J., K. Wray, and S. Zilberstein, "Meta-Level Control of Anytime Algorithms with Online Performance Prediction," in *IJCAI*, 2018. DOI: 10.24963/ijcai.2018/208.
- [195] Gagliolo, M., and J. Schmidhuber, "Learning Dynamic Algorithm Portfolios," *Annals of Mathematics and Artificial Intelligence*, Vol. 47, 2006. DOI: 10.1007/s10472-006-9036-z.
- [196] López-Ibáñez, M., and T. Stützle, "Automatically Improving the Anytime Behaviour of Optimisation Algorithms," *European Journal of Operational Research*, Vol. 235, No. 3, 2014. Extended version at <https://core.ac.uk/download/pdf/208141673.pdf>. DOI: 10.1016/j.ejor.2013.10.043.
- [197] Smith, S., et al., "Robust Allocation of RF Device Capacity for Distributed Spectrum Functions," *Journal of Autonomous Agents and Multi-Agent Systems*, Vol. 31, No. 3, 2017. Online: <https://link.springer.com/article/10.1007/s10458-016-9329-5>.
- [198] Gillen, M., et al., "Beyond Line-Of-Sight Information Dissemination for Force Protection," in *MILCOM*, 2012. DOI: 10.1109/MILCOM.2012.6415700.
- [199] Fioretto, F., E. Pontelli, and W. Yeoh, "Distributed Constraint Optimization Problems and Applications: A Survey," *Journal of Artificial Intelligence Research*, Vol. 61, No. 1, 2018. DOI: 10.1613/jair.5565.
- [200] Faltings, B., "Distributed Constraint Programming," in *Handbook of Constraint Programming*, Amsterdam, Netherlands: Elsevier, 2006. Online: <http://cse.unl.edu/~choueiry/Documents/Faltings2006-DisCSP.pdf>.
- [201] Shoham, Y., and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, UK: Cambridge University Press, 2009. Online: <http://www.masfoundations.org/download.html>.
- [202] Moallemi, C., and B. Van Roy, "Consensus Propagation," *Transactions on Information Theory*, Vol. 52, No. 11, 2006. Online: <https://arxiv.org/abs/cs/0603078>.
- [203] Iscen, A., and K. Turner, "Decentralized Coordination via Task Decomposition and Reward Shaping," 2013.

- [204] Vezhnevets, A. S., et al., "FeUdal Networks for Hierarchical Reinforcement Learning," in *ICML*, Sydney, NSW, Australia: JMLR.org, 2017.
- [205] Schilling, M., et al., "Decentralized Control and Local Information for Robust and Adaptive Decentralized Deep Reinforcement Learning," *Neural Networks*, Vol. 144, 2021. DOI: 10.1016/j.neunet.2021.09.017.
- [206] Dong, S., et al., "Decentralized Counterfactual Value with Threat Detection for Multi-Agent Reinforcement Learning in mixed cooperative and competitive environments," *Expert Systems with Applications*, Vol. 257, 2024. DOI: 10.1016/j.eswa.2024.125116.
- [207] Köster, R., et al., *Model-free Conventions in Multi-agent Reinforcement Learning with Heterogeneous Preferences*, 2020. DOI: 10.48550/arXiv.2010.09054.
- [208] Molzahn, D., et al., "A Survey of Distributed Optimization and Control Algorithms for Electric Power Systems," *IEEE Transactions on Smart Grid*, Vol. 8, No. 6, 2017. DOI: 10.1109/TSG.2017.2720471.
- [209] Duffy, K., C. Bordenave, and D. Leith, "Decentralized Constraint Satisfaction," *IEEE/ACM Transactions on Networking*, Vol. 21, No. 4, 2013. DOI: 10.1109/TNET.2012.2222923.
- [210] Yazicioglu, A. Y., M. Egerstedt, and J. S. Shamma, "Communication-Free Distributed Coverage for Networked Systems," *IEEE Transactions on Control of Network Systems*, Vol. 4, 2015. Online: <https://api.semanticscholar.org/CorpusID:18647127>.
- [211] Di Lorenzo, P., and S. Barbarossa, "Distributed Resource Allocation in Cognitive Radio Systems Based on Social Foraging Swarms," in *International Workshop on Signal Processing Advances in Wireless Communications*. IEEE, 2010. DOI: 10.1109/SPAWC.2010.5670805.
- [212] Stanković, M., S. Stanković, and K. Johansson, "Distributed Time Synchronization in Lossy Wireless Sensor Networks," in *IFAC Workshop on Estimation and Control of Networked Systems*, 2012. DOI: 10.3182/20120914-2-US-4030.00054.
- [213] Yan, H., et al., "Software Defined Radio Implementation of Carrier and Timing Synchronization for Distributed Arrays," in *Aerospace Conference*, 2019. DOI: 10.1109/AERO.2019.8742232.
- [214] Kim, M., H. Kim, and S. Lee, "A Distributed Cooperative Localization Strategy in Vehicular-to-Vehicular Networks," *Sensors*, Vol. 20, No. 5, 2020. DOI: 10.3390/s20051413.
- [215] Wolpert, D., "The Lack of A Priori Distinctions Between Learning Algorithms," *Neural Computation*, 1996. DOI: 10.1162/neco.1996.8.7.1341.
- [216] Wolpert, D., and W. Macready, "No Free Lunch Theorems for Optimization," *Transactions on Evolutionary Computation*, Vol. 1, No. 67, 1997. DOI: 10.1109/4235.585893.

6

Electronic Battle Management

An EBM system plans how many platforms to deploy, which resources each gets, and where they will go. It coordinates activity across electromagnetic spectrum domains, and with broader issues such as logistics. EW plans should identify the desired electromagnetic profile, identify missions, tasks, and resources, evaluate threats, and meet the constraints and goals of relevant policies [1]. Factors affecting the plan include available assets, desired effects, placement limitations (altitude, range, time, or loads); frequency deconfliction, anticipated EW missions from other services; and authentication requirements [2]. Figure 6.1 captures some of the inputs and outputs of an EBM system, desired effects, platform limitations, rules of engagement, and anticipated EW missions from other services.

EBM is part of a complete *joint all-domain command and control (JADC2)* solution, supporting the strategic vision of *multidomain operations (MDO)*. MDO

Inputs

Rules of Engagement	Environment	Operating Constraints	Available Assets	Cooperation with Other Organizations
Commander Intent	Mission Model	Expected Threats	Attack Library	Desired Effects

Outputs

Asset Allocations	Spectrum Sensing Plan	EP and EA Planning	Human Interactions	Inter-operability

Figure 6.1 To support deliberative EBM planning, inputs include relatively static descriptions of the mission, and outputs include high-level tasking. (*As planning becomes more automated, dynamic inputs will also become relevant.*)

seeks to integrate capabilities across land, sea, air, space, and cyberspace for seamless coordination. JADC2 is the technological framework enabling MDO that integrates sensors, actuators, and decision makers across all domains. EBM is the specific subset of JADC2 that addresses the EMS.

DoD doctrine explains the traditional approach to EW planning, noting that “joint EW is centrally planned and directed and decentrally executed...” [1], and that “...this planning requires a multidisciplinary approach with expertise from operations (ground, airborne, space), intelligence, logistics, weather, and information.” [2]

Moving to an *automated* EBM planning system creates a more interactive system that can respond to events as they occur during the mission. As noted in Chapter 5, automated planning activities overlap optimization and scheduling, and will become fully interactive integrated systems in the future. Section 6.1 presents AI planning approaches, discussing issues such as uncertainty, resource allocation, and multiple timescales. Section 6.2 touches on game theory approaches for team coordination and reasoning about adversarial actions. Section 6.3 discusses the human-machine interface (HMI), including how to leverage human expertise within the system, extract human-level goals for optimization, and explain decisions to human users. (Section 7.2 describes how to update plans during a mission.)

In a complete EW system, there will be multiple decision makers. Planners might be task-specific, algorithm-delineated, or geographically or temporally separated. This separation-of-concerns makes system design significantly easier. Examples of task-specific planners include ROGUE, which had a task planner and a route planner [3], and Tactical Expert Mission Planner (TEMPLAR), which had four planners (Example 6.1).

Example 6.1 The Tactical Expert Mission Planner (TEMPLAR) generates air tasking orders.

There are four distinct planning modules in TEMPLAR [4]:

- The package planner, which generates mission lines on Target Planning Worksheets.
- The flow planner, which generates mission lines on unit schedules for Close Air Support (CAS) and Defensive Counter Air (DCA) missions.
- The air-air refueling planner, which generates mission lines on Refueling Planning Worksheets.
- The mission line planner, which fills in mission number, call sign, Selective Indication Feature (SIF) code, and other information on existing mission lines.

Example 6.2 An AI planner and scheduler controls astronomy tasks on the Hubble Space Telescope.

The Hubble Space Telescope was one of the first fielded, large-scale, automated planning and scheduling systems [5, 6]. The *Heuristic Scheduling Testbed System* constructs observation schedules based on goal expansion (planning) and resource allocation (scheduling). Astronomers formulate observation programs with temporal constraints specifying the collection of light from a celestial object with one of the six scientific instruments. The resulting plan must accomplish tasks for multiple astronomers. Multiple layers of abstraction allow the planner to manage the combinatorics of the problem: The abstract level manages telescope availability, reconfiguration time, and target visibility, while the detail level manages specific exposures.

6.1 Planning

Planning operates over a broader scope and with a slower time envelope than optimization or scheduling. Planning evaluates actions over longer time horizons, and handles broader asset and resource types. The Hubble Space Telescope is automatically controlled with a planning and scheduling system built in the early 1990s, and highlights the tight coupling between a planner and a scheduler (Example 6.2).

Planning focuses on the causal reasoning for finding a sequence of actions needed to achieve the goal(s). The plan is usually a partially ordered set of actions that does not specify resource or a schedule; scheduling assigns time and resources to the plan. Figure 6.2 shows how different challenges drive selection of a planning approach.

Classical planning takes as input a description of the current situation (an initial state), a set of actions, and a description of the goal(s), and generates as output a sequence of actions that lead from the current situation to the goals. *Probabilistic planning* handles nondeterministic and partially observable environments. *Temporal planning* handles actions with duration, or can be concurrent; temporal planning techniques usually handle resources because resource use is frequently temporal in nature (e.g., a resource may be used by at most one action at a time). *Conditional planning*, or *contingent planning*, generates plans with conditional branches: Rather than a sequence of actions, it generates a contingency plan that chooses actions based on perception of the environment, such as the one illustrated in Figure 6.3. The more mission-critical the failure, the more important it is to prepare in advance for contingencies [7]. These techniques are not mutually exclusive, and moreover usually multiple techniques typically need to be present in any given solution; for

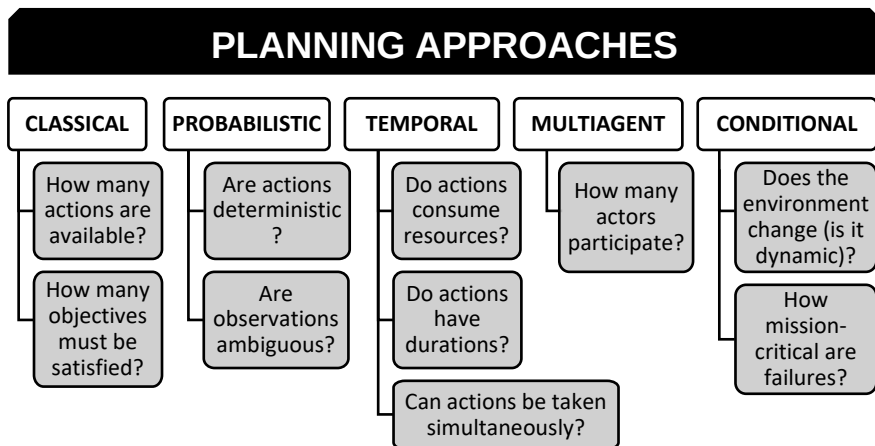


Figure 6.2 Different problem domain challenges require different considerations in the planning approach (not mutually exclusive).

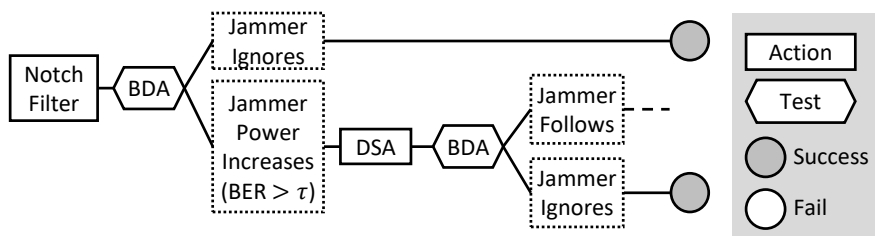


Figure 6.3 A contingent plan generates actions with conditional branches based on expected outcomes. In simple cases such as this plan for a tone jammer (Table 5.1), they look like decision trees or rules systems.

example, task allocation in multivehicle systems combines temporal planning with contingent planning [8].

Several good books exist on planning [9–11]. Benaskeur et al. [7] describe different planning approaches for naval-engagement scenarios. The International Planning Competition [12], run roughly biennially since 1998, runs tracks for different types of planners on realistic (and sometimes real) real-world problems, such as satellite-based Earth observation, wildlife poaching, and data network task scheduling. Notably, the source code from these planners is public. Yuen [13] presents a

graphic showing the evolution of IPC competition tracks and the important trade-offs to consider for different types of problems. Yuen then presents a detailed cyber red-teaming use case.

Search and planning algorithms are judged on the basis of *completeness*, *optimality*, *time complexity*, and *space complexity* [9]. This chapter presents search algorithms that search through the solution space in a *deliberative* manner. Section 5.1.2 presented *randomized* search engines, and uses these same criteria to evaluate their appropriateness to EW.

6.1.1 Planning Basics: Problem Definition, and Search

A planning problem can be defined formally with the following components:

- An initial state, $s_0 \in S$,
- A set of possible actions, A ,
- A description of what each action does, as a function that returns the state $s' \in S$ that results from doing action a in state s , $\rho : (s, a) \rightarrow s'$,
- A goal state, $s_g \in S$, and
- A cost function that assigns a cost to each step between states, $c(s, a, s')$.

A solution to the planning problem $\mathcal{P}$ is to find a sequence of actions $\pi = (a_1, a_2, \dots, a_n)$ from s_0 that produces a sequence of states $(s_1, s_2, \dots, s_g)$. An optimal solution has the lowest cost path from s_0 to s_g .

Classical planning domains are commonly represented in *PDDL*, the Planning Domain Definition Language [14]. PDDL syntax encodes the domain: states, actions, and compound actions. States are represented as a conjunction of positive literals *predicate*(*arg1*,...,*argN*), for example, $s = in(pilot, airplane) \text{ AND } has\_fuel(airplane)$. Actions have *preconditions* and *effects*; an action a is applicable in a state s only if its precondition literals $p_i \in s$ are true. Several variants of PDDL capture planning problems of different natures and complexities, including constraints, conditional effects, numeric and temporal features, and action costs.

Search algorithms look for the sequence of actions by creating a search tree whose root is the initial state. Each *branching step* expands the current state by applying each legal action to the state, thereby generating new states. As an example, a simple robot moving in a grid-world can move to any adjacent grid square, resulting in four possible future states. (A robot moving on a flat surface has a 2D infinite space of future states. An aerial drone has a 3D infinite space.) Continuous values can be discretized, abstracted, or presented with statistics such as mean and variance. *Pruning steps* eliminate illegal or repeated states to reduce the search effort.

Uninformed algorithms build this search tree without guidance from the domain, expanding nodes until they reach the goal. Common strategies include breadth-first search, depth-first search, iterative deepening, and bidirectional search. Given the complexity of EW, we do not present these in more detail here, because domain knowledge is critical to finding solutions.

Informed search strategies use *heuristics* to guide the search, thus making search more efficient or more effective [15]. Heuristic rules can be generated by experts, or created by ML from empirical data. One of the best-known heuristic algorithms is A^* , a best-first search that expands the node n with the lowest estimated path cost: $g(n) + h(n)$, where $g(n)$ is the cost from the initial state s_0 to n , and $h(n)$ is the estimated cost from n to the goal state s_g .

Nearly every successful planning system for real-world domains uses domain-specific heuristics.

The randomized metaheuristic search algorithms of Section 5.1.2 are one approach to finding solutions in a complex space, and many of the schedulers in Section 5.2 also use heuristics. Typically, each possible plan is encoded as a sequence of actions that can be manipulated by the randomized search engine. For example, plans become chromosomes for GAs [16, 17], and graphs capture hazards in fire escape routes for ACO [18].

6.1.2 Hierarchical Task Networks

Hierarchical Task Network (HTN) planning embodies a different view of planning [11, 19–21]: Instead of searching for goals by changing the state of the world model, the HTNs are a set of abstract tasks to be done, and a set of methods for each task that represent different ways in which they can be carried out. The dependency among actions is represented with hierarchically structured networks.

HTN planners develop plans at increasing levels of abstraction. HTNs are the most widely used form of planning in practical applications for two reasons. First, they effectively manage computational complexity. Second, they naturally correspond to how humans think about problems. This characteristic means that humans can describe relationships among tasks more easily, and the user interface inherently explains task progress. These properties lead to increased user trust [22].

An HTN planner reasons over a *task network* representing the problem to be solved. The network is a collection of *tasks* to carry out and associated *constraints* such as their order, and what preconditions must be true, as illustrated in Figure 6.4. There are three types of tasks:

- *Goal tasks*, capturing the desired end state;

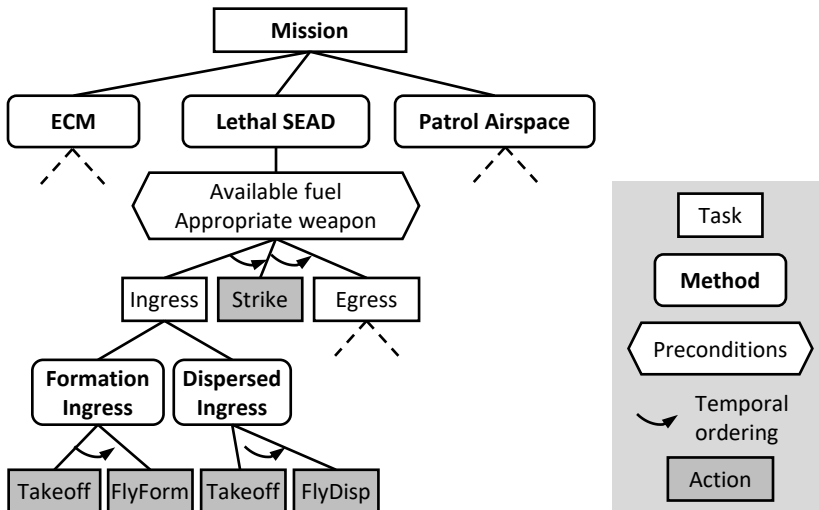


Figure 6.4 HTNs contain tasks, their preconditions, and associated constraints such as ordering. The mission goal has three possibly independent tasks: ECM, suppression of enemy air defense (SEAD), or Patrol Airspace. The SEAD method decomposes into three ordered tasks: ingress, strike, and egress. (Derived from [23].)

- *Primitive tasks*, which are actions that can be directly executed in the state, with associated preconditions and expected effects; and
- *Compound tasks*, showing how to accomplish goals with primitive tasks. A *method* shows how to decompose a compound task into partially ordered subtasks (primitive or compound).

A solution to an HTN problem is thus an executable sequence of primitive tasks that can be obtained from the initial task network by decomposing compound tasks into their set of simpler tasks, and by inserting ordering constraints.

HTN planners are used in many problem domains [21], including medicine, production planning, logistics, crisis management, air campaign planning, naval command and control [24], aircraft carrier scheduling [25], and unmanned combat robot teams [23]. Example 6.3 presents a more detailed example for coalition battle planning. Planning competitions have been run on HTN planning; problems and solution code are available online [29, 30]. Because HTN planners encode domain knowledge within the task networks, they can solve much larger planning problems, and solve them more quickly, than domain-independent planners.

Example 6.3 The Course-of-Action Display and Evaluation Tool (CADET) integrates HTN planning with scheduling to create coalition battle plans.

CADET assists military planners in coalition environments [26–28]. CADET is a knowledge-based tool capable of producing automatically (or with human guidance) battle plans with realistic degree of detail and complexity. CADET has been integrated with several battle management systems for DARPA and the US Army [27].

The human planner defines the key goals for a tactical course-of-action (COA), and CADET expands them into a detailed plan/schedule of the operation. CADET includes technology for interleaved planning, scheduling, routing, attrition, and resource consumption. CADET models assets and tasks, handles the adversarial environment, coordinates team efforts, and supports autonomous action (through commander's intent).

The authors state “The integration of planning and scheduling is achieved via an algorithm for tightly interleaved incremental planning and scheduling. The HTN-like planning step produces an incremental group of tasks by applying domain-specific ‘expansion’ rules to those activities in the current state of the plan that require hierarchical decomposition. The scheduling step performs temporal constraint propagation (both lateral and vertical within the hierarchy) and schedules the newly added activities to the available resources and time periods.” [27]

HTNs can be translated to other planning representations to improve their performance, for example PDDL [31], MDPs [32], and conditional and temporal planners [33–35]. In this way, the domain-independent planners can leverage domain knowledge encoded in the HTN. The ability to use domain knowledge is essential for solving large problems in complex domains. HTNs also support risk-aware planning [36] and nondeterministic planning [37].

6.1.3 Action Uncertainty

There are several approaches for planning in domains where actions have nondeterministic outcomes, including:

- *Conditional planning*: Decisions are made based on explicit “if-then” conditions that specify different actions depending on observed outcomes [38, 39].
- *Stochastic satisfiability*: Uses satisfiability solvers to handle probabilistic constraints in decision-making problems [40, 41].

Table 6.1
Planning Approaches in Domains with Nondeterministic Outcomes

Algorithm	Strengths	Weaknesses
Conditional planning	Simplicity, explainability, partial observability	Inflexibility, scalability
Stochastic satisfiability	Precision, formal guarantees, satisfiability	Computational complexity, limited expressiveness
First-order decision diagrams	Compact, decomposable	Complexity, limited to problems where relationships between variables are crucial
Belief space planning	Partial observability	Computational complexity
Monte Carlo tree search	Exploration, adapts dynamically during execution	Sample inefficiency, time-consuming
MDPs	Robust stochasticity, mathematical foundation	State explosion, limited to Markovian problems

- *First-order decision diagrams*: Combines decision diagrams with first-order logic to handle complex decision processes [42].
- *Belief-space planning*: Operates in the space of probability distributions over states, used to handle uncertainty in partially observable environments [43].
- *Monte Carlo tree search (MCTS)*: Incrementally builds a decision tree through random simulations, refining strategies to maximize expected outcomes in uncertain or large state spaces [44].
- *Markov decision processes (MDPs)*: A mathematical framework for modeling decision-making in environments with stochastic outcomes, where actions are chosen to maximize expected utility.

Table 6.1 outlines the strengths and weaknesses of these approaches.

Because MDPs are one of the most widely used of these, they are the focus of this section. (Section 6.1.4 discusses the orthogonal issue of *information* uncertainty.) MDPs are a framework for modeling and guiding sequential decision making under uncertainty [9, 45, 46]. Markov models describe systems that transition probabilistically between a finite set of states. When the domain is only partially observable, planning corresponds to a *partially observable Markov decision process (POMDP)*. MDPs augment the problem definition of Section 6.1.1 with *transition probabilities* $P(s'|s, a)$ that denote the probability of reaching state $s' \in S$ if action

$a \in A$ is done in state $s \in S$. A *reward* R captures the expected reward when action a is done in state s : $R(s, a)$.[†]

If models of adversary behavior are available from ES, they can be incorporated into the transition function to reflect the fact that the state of threats (which is part of each $s_i \in S_i$) is affected by both friendly and adversarial actions.

The policy π specifies what agents should do in every state; $\pi(s)$ is the action recommended by π for state s . In a *stationary policy*, once the policy is found, the action is fixed for each state; in a *nonstationary policy*, the policy changes over time (i.e., actions depend on history).

Because the environment is stochastic, the quality of a policy is measured by its *expected utility* from executing it many times. An optimal policy π^* is one that yields the highest expected utility. The planning problem is to find a good (or optimal) policy. Two standard algorithms that return a stationary optimal policy after a finite number of iterations are *value iteration* and *policy iteration* [48].

A common utility function is that of *delayed* or *discounted rewards*, where the utility of a state sequence is discounted over time, for the discount factor $\gamma \in [0, 1]$. The expected utility, or total return, of a policy π starting in state s_0 is:

$$V^\pi(s_0) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$$

where $a_t = \pi(s_t)$. $\pi^*(s_t)$ is then the action that maximizes the expected utility of the subsequent state s_{t+1} .[‡]

A variety of action-choice functions may be appropriate for the mission:

- *Maximax*: Choose the action with largest expected reward, even if it is unlikely (optimistic; risky; best possible outcome).

†. The notation we use in this book, $m = f(o, s)$ is similar in that it decomposes the state from the action, but we explicitly do not use $R(s, a)$ for two reasons. (1) We would like to avoid the assumption that we use sample-inefficient MDPs. Any ML model can estimate the value function [46]. (2) Also, our utility function is a combination of multiple metrics, which is not implied in the standard RL reward function. Our work predates the emergence of multiobjective RL research, which began gaining traction in the mid-2010s [47].

‡. Using the notation of Callout 2.1, we can augment the utility function to give each metric its own delayed reward, γ_k . The estimated discounted value for metric k at time t becomes:

$$M_k(t) = \mathbb{E} \left[\sum_{t'=t}^{\infty} \gamma_k^{t'-t} \cdot m_k(t' + 1) \right] = \mathbb{E} \left[\sum_{t'=t}^{\infty} \gamma_k^{t'-t} \cdot f_k(o_{t'}, s_{t'}) \right]$$

We redefine $\phi_k \triangleq (\gamma_k, w_k, M_k)$, so that the utility function for each node n remains unchanged: $\tilde{U}_n(t) = g_n(\phi_{n1}(t), \dots, \phi_{n\overline{m}n}(t))$

- *Maximum likelihood*: Pick the action with largest expected value by combining probability of outcome with expected reward (pragmatic; most probable scenario).
- *Maximin*: Choose the action with maximum of the minimum values (pessimist; risk-averse; worst-case safe bet).
- *Minimax regret*: Choose the action that minimizes worst-case regret, which is the opportunity-loss for making the wrong decision (cautious; avoids future disappointment).

A *myopic best response* selects the reward-maximizing action in each state, ignoring any possible future reward (i.e., $\gamma = 0$).

MDP planning is very computationally inefficient [48–50], but MDPs are the most widely used framework to formulate probabilistic plans. For this reason, much research focuses on how to improve MDP planning performance [32, 51–57].

Decentralized POMDPs are a common solution for a team of coordinating nodes [58], but are NEXP-complete [59], and may require doubly exponential time to solve in the worst case. Moreover, states and actions of node n include knowledge and actions from nodes n' . In practice, nodes reduce problem size by using heuristics or exploiting problem structure. *Locality of interaction*, for example, assumes nodes only need to coordinate with a limited subset of other nodes (i.e., nodes do not need to know every detail of every other node) [58, 60, 61]. Section 5.5 presents additional distributed optimization approaches.

MDPs and POMDPs are also frequently used in RL settings (Chapter 7), where RL techniques learn the probabilities and/or rewards of the model. There are three key reasons why RL is not synonymous with MDPs: (1) MDPs are models that specify a policy, while RL takes actions to update that policy, (2) in MDP planning, the planner must decide when to terminate planning, while RL terminates only when the mission is over [52], and (3) other ML models can be used for RL including SVMs and ANNs. Section 7.3.3 outlines some common RL algorithms, including multiarmed bandit and Q-learning, and their computational trade-offs.

6.1.4 Information Uncertainty

In complex environments, information is rarely known perfectly. Decision-making engines must incorporate and reason about the uncertainty in the information leading to decisions. With this information, the planner can use active sensing and communication to improve the information quality or certainty. Approaches to managing information uncertainty include:

- *Dempster-Shafer theory*, which measures *belief* in a fact and computes the probability that the evidence supports the proposition [62, 63];

Table 6.2
Active Sensing and Communication Actions Driven By Information Utility

Factor	Active Sensing	Deliberate Communication
Why	Improve info quality	Improve info quality
Who	Collaborative sensing	With whom to communicate
What	What to gather	What to transmit; Abstraction level
Where	Spatial and spectral	Team geography
When	Sensing schedule	Criticality of information
How	What sensors to use	Implicit or explicit interactions

- *Fuzzy logic*, which represents the truth value of facts [64]; and
- *Argumentation*, which explicitly constructs the relationships connecting evidence and conclusions, making it possible to reason about these relationships directly [65], for example to resolve conflicting information, or to allow planners to reason about causality [66].

These approaches support evaluating the utility of information.

Information utility enables being deliberative about whether, when, with whom, and what to communicate or sense.

Table 6.2 summarizes these active sensing and communication actions. The concepts of provenance and credibility are closely connected (Section 8.1.3), as is data fusion (Section 4.4). Table 6.3 suggests possible active sensing actions the system might take depending on the desired information. After performing an active sensing action, the execution monitor must determine whether the effect was an inherent function of the environment or whether the recent actions caused the observed behavior. Active sensing is a decision made to support ES rather than EP or EA.

Note also that communication can be interplatform, and also intraplatform, (i.e., among multiple decision-makers on a single platform). This section focuses on interplatform communication, but similar reasoning applies to the simpler problem of intraplatform communication.

When time and bandwidth are available, nodes can exchange observations and decisions to improve or recalibrate their understanding of the situation. Data fusion resolves ambiguities, reduces uncertainty in the models, and can accelerate the rate of identifying previously unknown threats. For example, a team of nodes can combine estimates of threats that can only be seen with fleeting glimpses.

In the broad context of performing distributed optimization under limited communications, traditional approaches assume that communications is safe and

Table 6.3
Sensing Actions Elicit Information About Unknown Emitters in the Environment

Domain	Desired Info	Active Sensing Action
Radar	Number of radar(s); Location of radar(s)	Generate ghosts; Directional probes for blind spots
Radar	Max detection range; Sensitivity; G-force limits	Move toward target; Signal amplification, PRI & Doppler; Increase replay rate
Communications	Node response time; Network recovery time; Network recovery methods	Surgical packet jamming; Self-collapse trigger attacks; Piecewise attack during recovery
Communications	Network critical nodes	Isolate nodes
Communications and Radar	ES recognition ability; EP methods; EA agility & precision	Honeypot sequences; Camouflage signatures

plentiful. In EW, this assumption is false: Communications are limited or denied due to jamming, mobility disconnects, or deliberate choices to reduce RF emissions. Reasoning about when to communicate can achieve near-optimal results [67–72]. The system can minimize the operational and computational costs of communication based on a quantitative estimate of the benefit of communicating or sensing. Some of the decision maker’s choices include:

- *Types of information:* Surprising outcomes; high-penalty outcomes; critical coordination; ambiguous or conflicting observations; uncertain variables, predictions, or inferences; unfamiliar concepts; assumptions (testing hypotheses).
- *With whom:* Nodes in coordinated actions; nodes with temporal dependencies; nearby nodes.
- *Level of abstraction:* State and action abstractions depend on task.
- *Costs:* Energy; exposure; congestion; coordination; cost of being wrong.

The level of detail at which nodes communicate determines the level of coordination they can achieve if/when needed and the kind of coordinated actions they can execute. It also impacts the computational and communication cost of solving the problem. (Even in settings where instantaneous and cost-free communication is available, broadcasting all information to all nodes doesn’t scale [58].) A richer communication language is informative (e.g., incorporating detailed current

state [73]), leading to better coordinated decisions, but results in a larger problem size. In a simpler language, the nodes can exchange commitments over actions with nonlocal effects [74]. In the simplest language, a node can signal the completion of an action with nonlocal effects, but for this information to be informative, the recipient needs to have some knowledge of what the completion of this action implies regarding the sender's future intentions. *Ontologies* are a way to represent information (Section 8.2), and can be used jointly with the planner communication languages.

The exploration concepts from RL can be a very effective approach to acquiring information because they often select the most informative actions to take to reduce uncertainty (Section 7.3).

6.1.5 Temporal Planning and Resource Management

Protracted engagements require awareness and coordination over shared resources (including communications). Nodes are heterogeneous, resources are constrained (and possibly nonrenewable), and distributed tasks must be coordinated. Augmenting the planning with resource management ensures a unified approach to resource optimization and EW strategy selection. (Ignoring resources creates plans that may not be achievable, and/or need to be updated based on execution monitoring per Section 7.1.) The planner must make the best use of resources to handle current and anticipated threats according to the mission model, for the timescale over which the resources are needed. Temporal planning and resource planning are tightly coupled, because resource use is frequently temporal in nature. Temporal overlap between actions is limited if they need the same resource in conflicting ways.

The critical feature of a cognitive radar is that various constraints have to be fulfilled resulting from the hardware, the platform or the environment. In reality, the constraining requirements are often contradicting. The art-of-radar-system engineering consists of handling these constraints by matching the different hardware and software components of the system to fulfill the requirements in a good compromise.

— *Richard Klemm et al., 2017 [75]*

Consider a heterogeneous team of nodes. Airborne assets must coexist and share tasks with ground-based EW systems. The appropriate assignment of tasks to assets depends on available resources (such as transmit power, frequency band, range, look angle, and time-to-target) and friendly considerations (such as fratricide minimization). For example, while ground-based systems may have longer time-on-station, they also have more limited range and greater vulnerability. The planner

must ensure that these multiobjective trade-offs are effectively managed when tasks are assigned.

Resource management ensures availability and appropriate usage of resources, both locally at a single node, and as needed by coordinating groups of nodes. The planner does this by (1) tracking current and expected resource usage, (2) determining what resource usage information should be shared with other nodes, and (3) sharing this information in an efficient and timely manner.

Resources have associated *capacity*, which can be a categorical value (e.g., available/unavailable) or a numerical value that tracks consumption. Consumable resources may be renewable or disposable, and are either shareable by multiple tasks, or exclusive for one task.

By incorporating resource consumption considerations into the planner, we ensure effective decisions that address threat conditions while respecting long-term distributed resource constraints. Resource constraints have been successfully incorporated into decision-theoretic models for both single [76] and multiple [77] decision makers. Different approaches to resource modeling have different impacts on resource reasoning. Extending the definitions of Section 6.1.3, we augment the rewards R or state descriptions s . Table 6.4 outlines the options:

- *Managing local resource consumption through costs.* The simplest approach is to incorporate local resource consumption directly into the reward function, enabling simultaneous optimization of rewards and resource usage.
- *Incorporating levels of local resources in state.* A richer expression incorporates resource profiles into the state, allowing the planner to reason about

Table 6.4
Different Approaches to Resource Modeling

Network Scope	Temporal Scope	Approach
Local	Immediate	Consider resource utilization of actions: $\bar{R}_n(s_n, a_n) = R_n(s_n, a_n) - \sum_{k=1}^K c_k \times \text{needs}(a_n, k)$
Local	Long-term	Capture local resources in state: $\bar{s}_n = \langle s_n, q_1, \dots, q_r \rangle \bar{P}_n(\bar{s}_n' \bar{s}_n, a_n)$
Global	Long-term	Capture shared resources in state: $\bar{P}_n(\bar{s}_n' \bar{s}_n, \vec{a})$

s_n = state of node n ; $\bar{s}_n$ = augmented state of node n ; a_n = action of node n ; c_k = cost of resource k ; $\text{needs}(a_n, k)$ = action a_n requires resource k , K = number of resources.

the long-term cumulative resource effects of actions. Consumable resources, such as fuel or the number of available decoys, must be managed so that they are used when they are most effective for the mission. An augmented state $\bar{s}_n$ for node n is $\bar{s}_n = \langle s_n, q_1, \dots, q_K \rangle$, where each q_k is the available remaining amount of resource k and s_n is the original state. The transition function P operates on the augmented state. The resulting function $\bar{P}_n(\bar{s}_n' | \bar{s}_n, a_n)$ ensures that the only possible transitions are the ones consistent with resource consumption of the actions:

$$\bar{P}_n(\bar{s}_n' | \bar{s}_n, a_n) = \begin{cases} 0 & \text{if } \exists k \ \bar{s}_n.q_k - \text{needs}(a_n, k) \neq \bar{s}_n'.q_k \\ P_n(s_n' | s_n, a_n) & \text{otherwise} \end{cases}$$

- *Incorporating levels of shared resources in state.* Shared resources are not entirely under the node's control, thus requiring coordination. When nodes coordinate, they can exchange abstract information that includes probabilities of using a shared resource. $\bar{P}_n$ thus accounts for another node taking an action that affects a shared resource. For example, consider two nodes who adjust their power level during transmission to have their combined SNR ratio meet a desired level at the target. As the nodes (friendly or adversarial) move, the nodes need to update their power levels.

Game theoretic approaches are also relevant (Section 6.2) (e.g., for finding fair resource allocations). Various optimization and scheduling approaches can also accomplish the task; see surveys [7, 78–80], or Chapter 5. Multiarmed bandit (Section 7.3.3.2) is also useful.

This unified approach to decision making and resource management allows the planner to simultaneously consider resource impact and attack effectiveness. The planner considers the impact of actions on immediate, near-term, and long-term mission goals.

6.1.6 Multiple Timescales

One of the system design questions to answer is how to handle decision making at different timescales. Decisions may have different lead times, actions may have different durations, and metrics may have different latencies before they can be measured. Figure 6.5 illustrates the temporal groups. Priority tasks are often in conflict, both within platforms and across platforms, and architectural choices at lower levels constrain higher-level capabilities.

The most effective approach is to completely decouple the planners; for example have one planner that operates over units of days, a second that operates

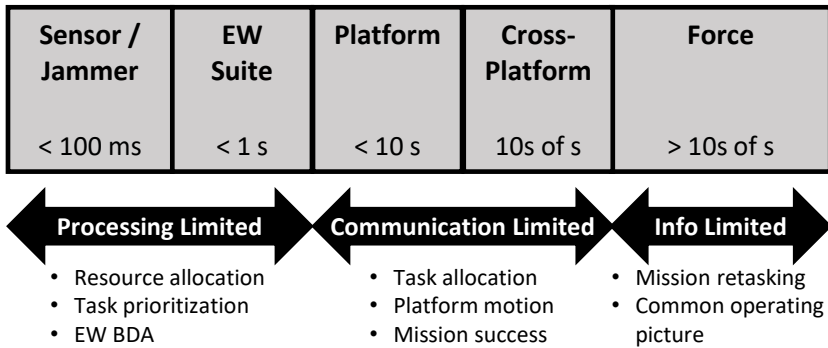


Figure 6.5 Different temporal requirements drive different architectural choices.

over minutes, and a third that operates over milliseconds. Each layer of temporal abstraction places constraints on the layer(s) below. This approach is simple to design, implement, and test, and supports specialized reasoning at different scales.

Within a single planner, it may be beneficial to reason over abstractions, increasing the resolution in each iteration. This approach, while being harder to design than decoupled planners can lead to significantly better solutions because all dependencies are explicit.

6.2 Game Theory

Game theory is a set of analytical tools used to model strategic, complex interactions among multiple entities. Game theory supports decision making in uncertain environments when the uncertainty in the outcome of an action depends on how others react [81, 82].

Game theory characterizes both cooperation and competition. It is effective for cooperative teams, for resource allocation, and for adversarial settings. *Cooperative games* allow individuals to form agreements before choosing actions, while *noncooperative games* model selfish-but-rational individuals.

In the teamwork setting, game theory addresses self-organizing, decentralized networks where individuals have self-interest, but must cooperate to maximize performance [83–88]. Cooperative self-interest extends to distributed resource use. Game theory has been applied to the RF domain in power control, admission control, waveform design [89] and network management [79, 84, 85, 90, 91].

Game theory is an effective method to handle adversarial (competitive) situations. Ho et al. [92] examine the role of game theory in defense, covering resource

allocation, information warfare, weapons control, and adversary monitoring. It is a popular approach in security settings, modeling attacker and defender, such as cybersecurity [93–95], target tracking and missile guidance [92], spectrum sharing in CRNs [96], spectrum sharing between radar and communications [97], and EW for countering jammers [98–108]. He and Su [109] review game theory for radar anti-jamming using power allocation, network coding design, waveform design, signal detection, target tracking, and radar communication. Sundaram et al. [110] examined the case where defense EP actions have costs, while Blum [111] examined electronic countermeasures including bluffing to create uncertainty.

The mixed-strategy game-theoretic decision can be viewed as rolling biased dice: Traditionally, the decision maker chooses the single optimal action, while in an adversarial game, the decision maker chooses a semi-random action based on a probability that is a function of the direct utility and the adversary's possible responses. The randomness makes it harder for the adversary to anticipate actions. As an example in human operations, high-risk activities such as improvised explosive device (IED) disposal, the principle of never repeating the exact same steps is a critical safeguard against adversarial adaptation. Adversaries learn from previous encounters, and if disposal teams follow the same procedures each time, insurgents can predict and counteract their methods. The same concept applies to decision making in all adversarial contexts; game theory provides a formal framework for cognitive systems.

To incorporate game theory in an EW decision maker, two significant challenges must be addressed: (1) confidence about adversary goals and strategies, and (2) computational cost of solving game-theoretic models in real time.

In adversarial settings, the approach relies on an estimate of the adversary's utility. *Zero-sum games* assume that the loss and gain for each participant is exactly balanced by the losses or gains of other participants (i.e., that the total losses and gains for all players sum to zero). Zero-sum games can be solved with linear programming. EW usually assumes that the adversary wins when friendly forces lose, but this simplification may not be true. Moreover, games that are zero-sum in one dimension may not be zero-sum in another, increasing the complexity of the decision-making problem.

Algorithmic game theory studies the computational aspects of game theory, focusing on finding solutions with reasonable limits on computational complexity. Generally, game-theoretic solutions are computationally difficult, although approximate or problem-specific solutions can be constructed.

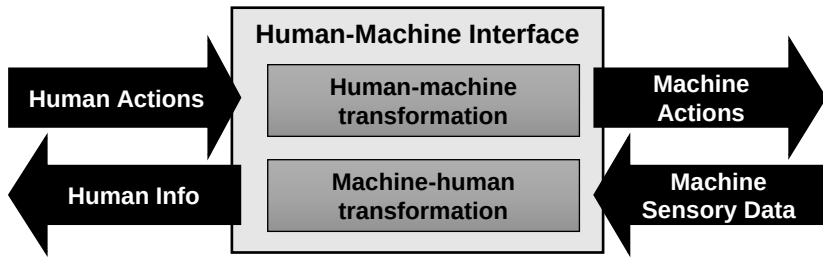


Figure 6.6 An HMI translates between human and machine (adapted from [112]).

6.3 Human-Machine Interface

A successful design of the HMI needs to be intuitive, flexible, extensible, and highly responsive. The main objectives of the HMI system should be to (1) improve operational performance, (2) increase effectiveness, and (3) alleviate tasking overload on human pilots/drivers, EWOs, and mission planners. The system should provide a set of wide-ranging and intuitive displays and controls for an operator's timely interaction with EW system functions and data. A well-considered HMI design is critical for maintaining situation awareness.

Customarily, HMI transforms human operator's actions to command actions for the machine and then transforms the machine's sensory data back to the human operator as illustrated in Figure 6.6 [112]. Information from the AI supports common understanding of the operating picture, while the human provides decision guidance. In traditional HMIs, these transformations are static and do not change based on the state of the human operator, the machine, or the environment.

Human-machine teaming (HMT) addresses this gap: Transformation processes are no longer static and may employ intelligent agents and ML. HMT is a relationship between the human operator and machine that goes beyond a human operating or supervising machine(s) [112, 113]. This evolving discipline aims to develop operative teams where human operator and automation are complementary team members [114–117].

HMT is not, however, a panacea to all teaming relationships. Vaccaro et al. [118] analyzed hundreds of HMI studies and found that, on average, human + AI collaborations performed worse than either humans alone or AI alone. Notably, AI outperformed human-AI teams in decision-making tasks, while collaborative teams excelled in creative tasks like content generation. Adding an AI teammate can reduce team coordination, communication, and trust; human-AI teams frequently

underperform due to poor team cognition and mutual understanding [119]. To design an ethical and trustworthy AI system, the HMT framework should therefore be based around four themes [120]:

1. Accountability to humans,
2. Cognizance of speculative risks and benefits,
3. Respectfulness and security, and
4. Honesty and usability.

Transparency and explainability of AI agents are crucial aspects for enabling shared cognition and establishing trust. The interrelationship between humans and machines will have five dominant aspects or technological challenges to contend with [121]:

- *Human state sensing and assessment:* Performance, behavior, physiological factors;
- *Human-machine interaction:* Communication and information sharing between human and machine;
- *Task and cognitive modeling:* Establishing workload and decision-making balance by task and function allocation;
- *Mutual training:* Adaptive learning and extended mutual training between human and machine; and
- *Data fusion and understanding:* Integration of human and machine data to generate a shared world model.

An EBM system plans how many platforms to deploy, which resources each gets, and where they will go. To achieve all of these objectives, the EBM system can benefit from multiple HMT-enabled HMIs and may have to establish interaction points with every notional block of the EBM system depicted in Figure 1.8. It may also be beneficial to have an overarching, global HMI providing the end-to-end capability for the entire EBM system. Trust must be established at every HMT-based HMI. Figure 6.7 depicts the potential HMI interaction points for the EBM components of the CogEW system.

These interaction points vary among the CogEW system participants (i.e., designers, commanders, mission planners, EWOs, and the cognitive system). Below we highlight some of the concepts/interaction points that each stakeholder type may be concerned with.

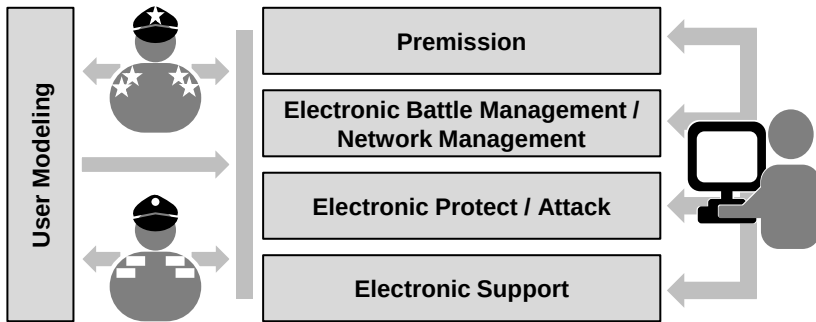


Figure 6.7 Permission planners, EWOs, and system designers interact with the system at all points; the system must accept their commands and inputs, track their tasks, learn their preferences, and explain its own decisions.

6.3.1 System Designers

System designers have the greatest impact on how the system works, and are often forgotten as key members of the human-machine team. Leveraging human expertise improves the performance of the AI components, thus supporting EWOs and the EW mission. Planning systems have leveraged human expertise since their initial inception, for example showing the power of domain-guidance in HTN planners (Section 6.1.2). Haigh et al. [122] discuss hybrid ML models in RF, and neurosymbolic AI approaches seek to blend numerical approaches (e.g., neural nets) with symbolic approaches (i.e., concepts that a human can understand) [123, 124]. Designers can provide:

- Assumptions about types of missions, users, and platforms.
- Concepts to represent, for example via ontologies (see Section 6.1.4).
- Choice of system observables, controllables, and metrics.
- Abstract feature construction for ES, for example, cross-layer interactions or effect of topography on signal propagation. Haigh et al. [122] use distance-to-neighbors to bootstrap a rough estimate of MANET throughput.
- Statespace reduction and guidance for optimization: for example relevant or critical parameters, current operating space, constraints.
- Type of learned models: for example decision trees vs DeepNets as a function of how explainable the results need to be [125].
- Within learned models to match data types, inference requirements, speed, or accuracy: for example changing the form of the model.

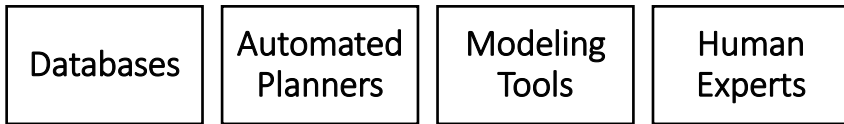


Figure 6.8 A variety of decision aids can help human EW planners.

- Search heuristics to reduce search effort. Experts and offline simulations can create guidelines and rules, expressed in a language appropriate to the decision maker. For example, Mitola [126] suggested radio knowledge representation language (RKRL), and a policy description language manages spectrum sharing in radio networks [127].

6.3.2 Commanders, Mission Planners, and EWOs

Commanders control and guide all participants in the mission. The system must capture their objectives and vision. *Commander's intent* is a critical concept for the entire EW mission team, and describes concepts such as acceptable risk, possible methods, and success conditions. Human participants and machine-generated decision support need to have the ability to communicate and interpret a shared intent [128, 129]. For example, Gustavsson et al. [128] propose a model that relates commander's intent to effects, and supports both traditional military planning and effects-based operations. Utility captures objectives in a way that is operationally meaningful and optimizable in practice (Section 2.4). EW resources for mission planners include databases for intelligence and operational data, (automated) planners, spatial and propagation modeling tools, and reachback support if required information is not available immediately [1] (Figure 6.8).

Mission planners and EWOs interact with the EW system both in pre-planning and in real-time. They must be able to guide and control the automation, and understand all feedback. The HMT approach should capitalize on the unique strengths of humans and automation; Irandoust and Benaskeur [130] demonstrate this for threat evaluation and response planning.

Mixed-initiative planning is an approach to developing plans interactively, for example the automated planner presents trade-offs, and the human planner selects from amongst the choices [4, 24, 131, 132]. Figure 6.9 illustrates the major components of a mixed-initiative planner. Mixed-initiative planning supports common situation awareness, group-related information, rationale for decisions, and keeps the operator engaged. *Playbook interfaces* support mixed-initiative planning by allowing a user to call a "play" and then tailoring it to the situation [133, 134].

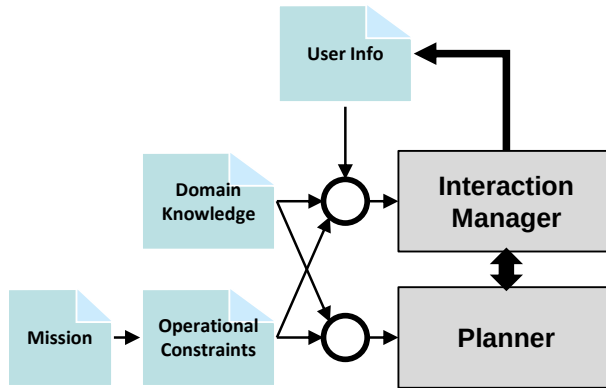


Figure 6.9 Mixed-initiative planning supports creative HMT problem solving.

Example 6.4 Raytheon’s EW Planning and Management Tool (EWPMT) uses plays to interact with the EW personnel.

Raytheon’s EWPMT software assists the maneuver commander’s ability to plan, coordinate and synchronize EW, manage the spectrum, and perform cyber operations [135–137]. EWPMT has been a US Army program of record since 2014.

EWPMT provides EW mission planning, EW targeting, and simulation capabilities to support course-of-action (COA) development. It develops the electromagnetic order of battle, deconflicts EW and communications assets, and displays the electromagnetic operational environment [136]. EWPMT takes in data from sensors in real time, and provides automation and analytics. It allows users to easily identify a new threat, geolocate that threat, and share that information [135].

Automated “plays” help relieve the cognitive burden on electronic warfare personnel in the field, enabling automated actions when certain conditions are met, such as turning off sensors on a specific frequency if they’re detected by the sensing system [135].

Influenced by sports playbooks, the playbook assumes each player knows the broad outline of a play, and each play has been well-practiced by the team. The shared task model provides a means of human-automation communication about plans, goals, methods, and resource usage. Raytheon’s EW planning and management tool (EWPMT) uses a playbook interface (Example 6.4). Playbook interfaces have an additional benefit: They map naturally to HTN planning (Section 6.1.2). For example, Goldman et al. [23] propose a mixed-initiative planning system with a

playbook interface to generate, check, and modify plans for teams of heterogeneous unmanned aerial vehicles.

The concept of *adjustable autonomy* allows a system to operate along a continuum of autonomy, often by imposing or relaxing constraints on its behavior [138, 139]. For example, a human may choose a low level of autonomy, where she “joysticks” an aerial vehicle, or a high level of autonomy, where she geotags a region of interest that the platform should monitor. Basich et al. [140] develop a competence-aware framework that models its own proficiency at different levels of autonomy from human feedback, while Bakirtzis et al. [141] uses a neurosymbolic approach to dynamically balance autonomy and human intervention. Adjustable autonomy improves human-machine collaboration, offloads human workload while maintaining human oversight, and increases efficiency, task performance, safety, and risk management.

Argumentation is a natural way to present conflicting concepts to humans, showing the reasons why an observation might be true or a decision might be good, and the reasons against (including deception) [66, 142, 143]. An international competition on argumentation is held every two years [144]; all source code is open-source. Presenting the pros/cons of a threat-evaluation system resulted in reduced stress and cognitive overload, while also building operator trust [145].

Finally, in addition to these planning and control approaches, humans can support ES, for example by labeling unknown signals, or providing supporting evidence such as environmental conditions (e.g., topography, weather, or threats). Geng et al. describe how to incorporate human sensors into the signal processing framework [146].

6.3.3 Understanding the Human Users

Another under-recognized concept in a HMT is that of the machine understanding the human users. *Intent Recognition* (Section 4.7) attempts to understand what goals the user is trying to achieve; this information can be used to anticipate user needs and customize the display.

Preference learning captures a user’s preferences in cases where they are hard to state explicitly [147]. Preference learning approaches can learn HTN models [148], utility functions (learning a function that captures why *a* is better than *b*), or preference relations (the partially ordered graph between options). Preference learning can take explicit or implicit feedback [149]. Bantouna et al. [150] use Bayesian statistics to learn user preferences for quality-of-service.

Inverse RL (IRL) is an ML technique where an agent infers the reward function of an expert by observing their behavior [151–153]. Unlike standard RL,

which learns a policy to maximize a given reward function, IRL works in reverse: It assumes expert demonstrations (actions, states) are optimal and attempts to deduce the hidden reward function that led to those decisions. IRL is particularly useful in complex decision-making environments where defining an explicit reward function is difficult, including EW. Used in EW for identifying cognitive radars or estimating interference by observing drone swarm movements [154, 155], the same approach can be used to explain human decision making. IRL could be used to learn what decisions an EWO would make in a given setting.

Imitation learning or *apprenticeship learning* learns to perform tasks by mimicking human demonstrations, rather than relying on well-specified objective functions [156, 157]. The approach can learn to schedule [158], learn how to collaborate in a team [159], and moreover does not require an expert teacher [160].

6.4 Conclusion

Automated EBM planning systems must support human planners at all stages of EW planning—long-range planning, midterm planning, and real-time mission management. Multiple tightly coupled decision makers can handle separate concerns.

HTN planners work for complex domains, and naturally support transparency, explainability, and control required for HMT. In-mission replanning (Section 7.2) ensures that the plan remains feasible despite surprises and changes in mission goals.

Planners must address both action and information uncertainty, manage temporal and resource constraints, and choose what and when to communicate with team members. Domain-specific heuristics and metalearning (Section 3.5) help improve the search process. Game-theoretic approaches assist with cooperative team management and competitive adversaries.

References

- [1] Chairman of the Joint Chiefs of Staff, *Joint Publication 3-13.1: Electronic Warfare*, 2012. Online: <https://fas.org/irp/doddir/dod/jp3-13-1.pdf>.
- [2] US Air Force, *Air Force Doctrine: Electronic Warfare, Document 2-5.1*, 2002. Online: <https://tinyurl.com/af-ew-2002-pdf>.
- [3] Haigh, K. Z., and M. Veloso, “Planning, Execution and Learning in a Robotic Agent,” in *Artificial Intelligence Planning Systems*, 1998. Online: <https://www.aaai.org/Papers/AIPS/1998/AIPS98-015.pdf>, Summary of Haigh’s Ph.D. dissertation.
- [4] Siska, C., B. Press, and P. Lipinski, “Tactical Expert Mission Planner (TEMPLAR),” TRW Defense Systems, Tech. Rep. RADC-TR-89-328, 1989. Online: <https://apps.dtic.mil/sti/citations/ADA285521>.
- [5] Miller, G., “Planning and Scheduling the Hubble Space Telescope,” *Robotic Telescopes*, 1995. Online: <http://adsabs.harvard.edu/full/1995ASPC...79..173M>.

- [6] Adler, D., D. Taylor, and A. Patterson, "Twelve Years of Planning and Scheduling the Hubble Space Telescope: Process Improvements and the Related Observing Efficiency Gains," in *Observatory Operations to Optimize Scientific Return III*. International Society for Optics, and Photonics, Vol. 4844, SPIE, 2002. DOI: 10.1117/12.460653.
- [7] Benaskeur, A., E. Bossé, and D. Blodgett, "Combat Resource Allocation Planning in Naval Engagements," Defence R&D Canada, Tech. Rep. DRDC Valcartier TR 2005-486, 2007. Online: <https://apps.dtic.mil/sti/pdfs/ADA605291.pdf>.
- [8] Carreno, Y., "Towards Full-Scale Autonomy for Multi-Vehicle Systems Planning and Acting in Extreme Environments," Awarded jointly with The University of Edinburgh, Ph.D. dissertation, Heriot-Watt University, Edinburgh, UK, 2023.
- [9] Russell, S., and P. Norvig, *Artificial Intelligence: A Modern Approach*. London, UK: Pearson Education, 2015.
- [10] Helmert, M., *Understanding Planning Tasks: Domain Complexity and Heuristic Decomposition*. New York, NY: Springer, 2008. Revised version of Ph.D. thesis.
- [11] Wilkins, D., *Practical Planning*. St. Louis, MO: Morgan Kaufmann, 1998.
- [12] *Automated Planning and Scheduling Competitions*. Online: <https://www.icaps-conference.org/competitions/>. Accessed 2025-02-12.
- [13] Yuen, J., "Automated Cyber Red Teaming," Defence Science, and Technology Organisation, Australia, Tech. Rep. DSTO-TN-1420, 2015. Online: <https://apps.dtic.mil/sti/citations/ADA618584>.
- [14] Haslum, P., et al., "An Introduction to the Planning Domain Definition Language," in *An Introduction to the Planning Domain Definition Language*, Morgan & Claypool, 2019. DOI: 10.2200/S00900ED2V01Y201902AIM042.
- [15] Korf, R., "Heuristic Evaluation Functions in Artificial Intelligence Search Algorithms," *Minds and Machines*, Vol. 5, 1995. DOI: 10.1007/BF00974979.
- [16] Liang, Y.-q., X.-s. Zhu, and M. Xie, "Improved Chromosome Encoding for Genetic-Algorithm-Based Assembly Sequence Planning," in *International Conf. on Intelligent Computing and Integrated Systems*, 2010. DOI: 10.1109/ICISS.2010.5655004.
- [17] Brie, A., and P. Morignot, "Genetic Planning Using Variable Length Chromosomes," in *ICAPS*, 2005. Online: <https://cdn.aaai.org/ICAPS/2005/ICAPS05-033.pdf>.
- [18] Goodwin, M., O. Granmo, and J. Rianti, "Escape Planning in Realistic Fire Scenarios with Ant Colony Optimisation," *Applied Intelligence*, Vol. 42, 2015. DOI: 10.1007/s10489-014-0538-9.
- [19] Erol, K., J. Hendler, and D. Nau, "HTN Planning: Complexity and Expressivity," in *AAAI*, 1994. Online: <https://www.aaai.org/Papers/AAAI/1994/AAAI94-173.pdf>.
- [20] Erol, K., J. Hendler, and D. Nau, "Complexity Results for HTN planning," Institute for Systems Research, Tech. Rep., 2003. DOI: 10.1007/BF02136175.
- [21] Georgievski, I., and M. Aiello, "HTN Planning: Overview, Comparison, and Beyond," *Artificial Intelligence*, Vol. 222, 2015. DOI: 10.1016/j.artint.2015.02.002.
- [22] Alnazer, E., et al., "A Role for HTN Planning in Increasing Trust in Autonomous Driving," in *IEEE International Smart Cities Conference*, 2022. DOI: 10.1109/ISC255366.2022.9922427.
- [23] Goldman, R., et al., "MACBeth: A Multi-Agent Constraint-Based Planner," in *Digital Avionics Systems Conference*, 2000. DOI: 10.1109/DASC.2002.1052934.

- [24] Mitchell, S., "A Hybrid Architecture for Real-time Mixed-Initiative Planning and Control," in *Innovative Applications of Artificial Intelligence*, 1997. Online: <https://www.aaai.org/Papers/IAAI/1997/IAAI97-205.pdf>.
- [25] Qi, C., and D. Wang, "Dynamic Aircraft Carrier Flight Deck Task Planning based on HTN," *IFAC-PapersOnLine*, Vol. 49, No. 12, 2016. Manufacturing Modelling, Management and Control. DOI: 10.1016/j.ifacol.2016.07.810.
- [26] Kott, A., et al., "Toward Practical Knowledge-Based Tools for Battle Planning and Scheduling," in *AAAI*, 2002. Online: <https://www.aaai.org/Papers/AAAI/2002/AAAI02-132.pdf>.
- [27] Ground, L., A. Kott, and R. Budd, "Coalition-based Planning of Military Operations: Adversarial Reasoning Algorithms in an Integrated Decision Aid," *Computing Research Repository*, 2016. Online: <http://arxiv.org/abs/1601.06069>.
- [28] Kott, A., et al., *Decision Aids for Adversarial Planning in Military Operations: Algorithms, Tools, and Turing-test-like Experimental Validation*, 2016. Online: <https://arxiv.org/abs/1601.06108>.
- [29] Behnke, G., D. Höller, and P. Bercher, (Eds.), *International Planning Competition: Planner and Domain Abstracts – Hierarchical Task Network (HTN) Planning Track*, 2021. Online: <https://bercher.net/publications/2021/Behnke2021IPC-Booklet.pdf>.
- [30] Alford, R., G. Behnke, and D. Schreiber, (Eds.), *HTN Planning Track of the International Planning Competition*, 2024, 31 pp. DOI: 10.5445/IR/1000177594.
- [31] Alford, R., U. Kuter, and D. Nau, "Translating HTNs to PDDL: A Small Amount of Domain Knowledge Can Go a Long Way," in *IJCAI*, 2009. Online: <https://www.cs.umd.edu/~nau/papers/alford2009translating.pdf>.
- [32] Kuter, U., and D. Nau, "Using Domain-Configurable Search Control for Probabilistic Planning," in *AAAI*, 2005. Online: <https://tinyurl.com/htn2mdp-2005>.
- [33] Victoria, J., "Automated Hierarchical, Forward-Chaining Temporal Planner for Planetary Robots Exploring Unknown Environments," Ph.D. dissertation, Technical University of Darmstadt, Darmstadt, Germany, 2016. Online: <https://tinyurl.com/htln-2016>.
- [34] Liu, D., et al., "Hierarchical Task Network-based Emergency Task Planning with Incomplete Information, Concurrency and Uncertain Duration," *Knowledge-Based Systems*, Vol. 112, No. C, 2016. DOI: 10.1016/j.knosys.2016.08.029.
- [35] Zhao, P., *Probabilistic Contingent Planning Based on HTN for High-Quality Plans*, 2023. Online: <https://arxiv.org/abs/2308.06922>.
- [36] Alnazer, E., I. Georgievski, and M. Aiello, "Risk Awareness in HTN Planning," 2022. DOI: 10.48550/arXiv.2204.10669.
- [37] Chen, D., and P. Bercher, "Fully Observable Nondeterministic HTN Planning – Formalisation and Complexity Results," in *ICAPS*, Vol. 31, 2021. DOI: 10.1609/icaps.v31i1.15949.
- [38] Andersen, M., T. Bolander, and M. Jensen, "Conditional Epistemic Planning," in *Logics in Artificial Intelligence*, 2012. DOI: 10.1007/978-3-642-33353-8_8.
- [39] To, S., T. Son, and E. Pontelli, "Contingent Planning as AND/OR Forward Search with Disjunctive Representation," in *ICAPS*, 2011. Online: <https://www.cs.nmsu.edu/~sto/ICAPS11.pdf>.
- [40] Majercik, S., and M. Littman, "Contingent Planning Under Uncertainty via Stochastic Satisfiability," in *AAAI*, 1999. DOI: 10.1016/S0004-3702(02)00379-X.
- [41] Ferraris, P., and E. Giunchiglia, "Planning as Satisfiability in Nondeterministic Domains," in *AAAI*, 2000. Online: <https://www.aaai.org/Papers/AAAI/2000/AAAI00-115.pdf>.

- [42] Joshi, S., and R. Khardon, "Probabilistic Relational Planning with First Order Decision Diagrams," *Journal of Artificial Intelligence Research*, Vol. 41, 2011. DOI: 10.1613/jair.3205.
- [43] Barenboim, M., and V. Indelman, "Adaptive Information Belief Space Planning," in *IJCAI*, 2022. Online: <https://arxiv.org/abs/2201.05673>.
- [44] Świechowski, M., et al., "Monte Carlo Tree Search: a Review of Recent Modifications and Applications," *Artificial Intelligence Review*, 2022. DOI: 10.1007/s10462-022-10228-y.
- [45] Mausam, and A. Kolobov, *Planning with Markov Decision Processes: An AI Perspective*. San Rafael, CA: Morgan & Claypool, 2012.
- [46] Sutton, R., and A. Barto, *Reinforcement Learning: An Introduction*. Denver, CO: Bradford, 2018.
- [47] Hayes, C., et al., "A Practical Guide to Multi-objective Reinforcement Learning and Planning," *Autonomous Agents and Multi-Agent Systems*, Vol. 36, No. 1, 2022. DOI: 10.1007/s10458-022-09552-y.
- [48] Littman, M., T. Dean, and L. Kaelbling, "On the Complexity of Solving Markov Decision Problems," in *Uncertainty in AI*, 1995. Online: <https://dl.acm.org/doi/10.5555/2074158.2074203>.
- [49] Gupta, A., and S. Kalyanakrishnan, "Improved Strong Worst-case Upper Bounds for MDP Planning," in *IJCAI*, 2017. Online: <https://www.ijcai.org/Proceedings/2017/0248.pdf>.
- [50] Jinnai, Y., et al., "Finding Options that Minimize Planning Time," in *ICML*, 2019. Online: <https://tinyurl.com/min-time-mdp>.
- [51] McMahan, B., and G. Gordon, "Fast Exact Planning in Markov Decision Processes," in *ICAPS*, 2005. Online: <https://www.cs.cmu.edu/~ggordon/mcmahan-gordon.fast-mdp.pdf>.
- [52] Taleghan, M., et al., "PAC Optimal MDP Planning with Application to Invasive Species Management," *Journal of Machine Learning Research*, Vol. 16, No. 1, 2015. Online: <https://dl.acm.org/doi/10.5555/2789272.2912119>.
- [53] Buşoniu, L., et al., "Optimistic Planning for Sparsely Stochastic Systems," in *Adaptive Dynamic Programming*, 2011. DOI: 10.1109/ADPRL.2011.5967375.
- [54] van Seijen, H., and R. Sutton, "Efficient Planning in MDPs by Small Backups," in *ICML*, 2013. DOI: 10.13140/2.1.4274.3043.
- [55] Zhang, J., M. Dridi, and A. Moudni, "A Stochastic Shortest-Path MDP Model with Dead Ends for Operating Rooms Planning," in *Automation and Computing*, 2017. DOI: 10.23919/IconAC.2017.8081974.
- [56] Yoon, S., A. Fern, and R. Givan, "FF-Replan: A Baseline for Probabilistic Planning," in *ICAPS*, 2007. Online: <https://engineering.purdue.edu/~givan/papers/icaps07-ffreplan.pdf>.
- [57] Teichteil-Koenigsbuch, F., G. Infantes, and U. Kuter, "RFF: A Robust, FF-based MDP Planning Algorithm for Generating Policies with Low Probability of Failure," in *ICAPS*, 2008. Online: <http://www.cs.umd.edu/~nau/others-papers/rff.pdf>.
- [58] Oliehoek, F., and C. Amato, *A Concise Introduction to Decentralized POMDPs*. New York, NY: Springer, 2016. DOI: 10.1007/978-3-319-28929-8.
- [59] Bernstein, D., et al., "The Complexity of Decentralized Control of Markov Decision Processes," *Mathematics of Operations Research*, Vol. 27, No. 4, 2002. Online: <https://arxiv.org/abs/1301.3836>.

- [60] Varakantham, P., et al., "Exploiting Coordination Locales in Distributed POMDPs via Social Model Shaping," in *ICAPS*, 2009. Online: <http://www.cs.cmu.edu/~pscerri/papers/ICAPS-09.pdf>.
- [61] Nair, R., et al., "Networked Distributed POMDPs: A Synthesis of Distributed Constraint Optimization and POMDPs," in *AAAI*, 2005. Online: <https://www.aaai.org/Papers/AAAI/2005/AAAI05-022.pdf>.
- [62] Denoeux, T., "40 Years of Dempster-Shafer Theory," *Approximate Reasoning*, 2016. DOI: 10.1016/j.ijar.2016.07.010.
- [63] Liu, L., and R. Yager, "Classic Works of the Dempster-Shafer Theory of Belief Functions: An Introduction," *Studies in Fuzziness and Soft Computing*, Vol. 219, 2008. DOI: 10.1007/978-3-540-44792-4_1.
- [64] Nikraves, M., "Evolution of Fuzzy Logic: From Intelligent Systems and Computation to Human Mind," in *Forging New Frontiers: Fuzzy Pioneers I. Studies in Fuzziness and Soft Computing*, Vol. 217, New York, NY: Springer, 2007. DOI: 10.1007/978-3-540-73182-5_3.
- [65] Mogdil, S., and H. Prakken, "The ASPIC* Framework for Structured Argumentation: A Tutorial," *Argument and Computation*, Vol. 5, No. 1, 2014. DOI: 10.1080/19462166.2013.869766.
- [66] Collins, A., D. Magazzeni, and S. Parsons, "Towards an Argumentation-Based Approach to Explainable Planning," in *Explainable Planning*, 2019. Online: <https://nms.kcl.ac.uk/simon.parsons/publications/conferences/xaip19.pdf>.
- [67] Mostafa, H., and V. Lesser, "Offline Planning for Communication by Exploiting Structured Interactions in Decentralized MDPs," in *Web Intelligence and Intelligent Agent Technology*, 2009. DOI: 10.1109/WI-IAT.2009.150.
- [68] Mogali, J., S. Smith, and Z. Rubinstein, "Distributed Decoupling of Multiagent Simple Temporal Problems," in *IJCAI*, 2016. Online: <https://tinyurl.com/multiagent-time>.
- [69] Melo, F., and M. Veloso, "Decentralized MDPs with Sparse Interactions," *Artificial Intelligence*, Vol. 175, 2011. DOI: 10.1016/j.artint.2011.05.001.
- [70] Yordanova, V., "Intelligent Adaptive Underwater Sensor Networks," Ph.D. dissertation, University College London, London, UK, 2018. Online: <https://discovery.ucl.ac.uk/id/eprint/10042482/>.
- [71] Ren, P., et al., *A Survey of Deep Active Learning*, 2020. DOI: 10.48550/arXiv.2009.00236.
- [72] Yang, P., et al., "Decentralized Over-the-Air Federated Learning by Second-Order Optimization Method," *IEEE Transactions on Wireless Communications*, Vol. 23, No. 6, 2024. DOI: 10.1109/TWC.2023.3327610.
- [73] Becker, R., V. Lesser, and S. Zilberstein, "Decentralized Markov Decision Processes with Event-Driven Interactions," in *International Joint Conference on Autonomous Agents and MultiAgent Systems*, 2004. Online: <http://rbr.cs.umass.edu/camato/decpomdp/pub/aamas04a.pdf>.
- [74] Xuan, P., and V. Lesser, "Incorporating Uncertainty in Agent Commitments," in *Agent Theories, Architectures, and Languages*, 1999. Online: https://link.springer.com/chapter/10.1007/10719619_5.
- [75] Klemm, R., et al., (Eds.), *Novel Radar Techniques and Applications*. Raleigh, NC: SciTech Publishing, 2017.
- [76] Altman, E., *Constrained Markov Decision Processes*. Boca Raton, FL: Chapman Hall, 1999.

- [77] Dolgov, D., and E. Durfee, "Stationary Deterministic Policies for Constrained MDPs with Multiple Rewards, Costs, and Discount Factors," in *IJCAI*, 2005. Online: <https://dl.acm.org/doi/10.5555/1642293.1642504>.
- [78] Muhammad, N., et al., "Resource Allocation Techniques in Cooperative Cognitive Radio Networks," *IEEE Communications Surveys and Tutorials*, Vol. 16, No. 2, 2013. DOI: 10.1109/SURV.2013.102313.00272.
- [79] Ramzan, M., et al., "Multi-Objective Optimization for Spectrum Sharing in Cognitive Radio Networks: A Review," *Pervasive and Mobile Computing*, Vol. 41, 2017. DOI: 10.1016/j.pmcj.2017.07.010.
- [80] Jorswieck, E., and M. Butt, "Resource Allocation for Shared Spectrum Networks," in *Spectrum Sharing*, 2020. DOI: 10.1002/9781119551539.ch13.
- [81] Shoham, Y., and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge, UK: Cambridge University Press, 2009. Online: <http://www.masfoundations.org/download.html>.
- [82] Nisan, N., et al., (Eds.), *Algorithmic Game Theory*. Cambridge, UK: Cambridge University Press, 2007. Online: <https://www.cs.cmu.edu/~sandholm/cs15-892F13/algorithmic-game-theory.pdf>.
- [83] Saad, W., et al., "Coalitional Game Theory for Communication Networks: A Tutorial," *IEEE Signal Processing Magazine*, 2009. Online: <https://arxiv.org/abs/0905.4057>.
- [84] MacKenzie, A., and L. DaSilva, *Game Theory for Wireless Engineers*. San Rafael, CA: Morgan & Claypool, 2006. DOI: 10.2200/S00014ED1V01Y200508COM001.
- [85] Han, Z., et al., *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge, UK: Cambridge University Press, 2012. The course <http://www2.eegr.uh.edu/~zhan2/game.theory.course/> contains slides based on the book.
- [86] Sun, Z., et al., "Applications of Game Theory in Vehicular Networks: A Survey," *IEEE Communications Surveys & Tutorials*, Vol. 23, No. 4, 2021. DOI: 10.1109/COMST.2021.3108466.
- [87] Naseer, S., et al., "A Game Theoretic Power Control and Spectrum Sharing Approach Using Cost Dominance in Cognitive Radio Networks," *PeerJ Computer Science*, Vol. 7, 2021. DOI: 10.7717/peerj-cs.617.
- [88] Tinh, B. T., et al., "Practical Optimization and Game Theory for 6G Ultra-Dense Networks: Overview and Research Challenges," *IEEE Access*, Vol. 10, 2022. DOI: 10.1109/ACCESS.2022.3146335.
- [89] Wang, H., et al., "Radar Waveform Strategy Based on Game Theory," *Radio Engineering*, Vol. 28, No. 4, 2019. Online: https://www.radioeng.cz/fulltexts/2019/19_04_0757_0764.pdf.
- [90] Zayen, B., A. Hayar, and G. Noubir, "Game Theory-Based Resource Management Strategy for Cognitive Radio Networks," *Multimedia Tools and Applications*, Vol. 70, No. 3, 2013. DOI: 10.1007/s11042-012-1211-0.
- [91] Mkiramweni, M., et al., "Game-Theoretic Approaches for Wireless Communications with Unmanned Aerial Vehicles," *IEEE Wireless Communications*, 2018. DOI: 10.1109/MWC.2017.1700250.
- [92] Ho, E., et al., "Game Theory in Defence Applications: A Review," *Sensors*, Vol. 22, No. 3, 2022. DOI: 10.3390/s22031032.

- [93] Snyder, M., R. Sundaram, and M. Thakur, "A Game-Theoretic Framework for Bandwidth Attacks and Statistical Defenses," in *Local Computer Networks*, 2007. DOI: 10.1109/LCN.2007.11.
- [94] Schramm, H., et al., "A Game Theoretic Model of Strategic Conflict in Cyberspace," *Military Operations Research*, Vol. 19, No. 1, 2014. Online: <https://www.jstor.org/stable/24838493>.
- [95] Kasmarik, K., et al., "A Survey of Game Theoretic Approaches to Modelling Decision-Making in Information Warfare Scenarios," *Future Internet*, Vol. 8, 2016. DOI: 10.3390/fi8030034.
- [96] Harihara Gopalan, S., et al., "Cognitive Radio Spectrum Allocation using Nash Equilibrium with Multiple Scheduling Resource Selection Algorithm," *Ain Shams Engineering Journal*, Vol. 15, No. 5, 2024. DOI: 10.1016/j.asej.2024.102688.
- [97] Shi, C., et al., "Applications of Game Theory in Cognitive Radar," in, Mishra, K., B. Shankar, and M. Rangaswamy, (Eds.), Raleigh, NC: SciTech Publishing, 2023.
- [98] Liu, X., et al., "SPREAD: Foiling Smart Jammers using Multi-layer Agility," in *INFOCOM*, 2007. DOI: 10.1109/INFCOM.2007.316.
- [99] Yang, D., et al., "Coping with a Smart Jammer in Wireless Networks: A Stackelberg Game Approach," *IEEE Transactions on Wireless Communications*, Vol. 12, No. 8, 2013. DOI: 10.1109/TWC.2013.071913121570.
- [100] Wonderley, D., T. Selee, and V. Chakravarthy, "Game Theoretic Decision Support Framework for Electronic Warfare Applications," in *IEEE Radar Conference*, 2016. DOI: 10.1109/RADAR.2016.7485124.
- [101] Mneimneh, S., et al., "A Game-Theoretic and Stochastic Survivability Mechanism Against Induced Attacks in Cognitive Radio Networks," *Pervasive and Mobile Computing*, Vol. 40, 2017. DOI: 10.1016/j.pmcj.2016.12.010.
- [102] Firouzbakht, K., G. Noubir, and M. Salehi, "Multi-Carrier Jamming Mitigation: A Proactive Game Theoretic Approach," in *Proactive and Dynamic Network Defense*. 2019. DOI: 10.1007/978-3-030-10597-6_10.
- [103] Li, K., B. Jiu, and H. Liu, "Game Theoretic Strategies Design for Monostatic Radar and Jammer Based on Mutual Information," *IEEE Access*, Vol. 7, 2019. DOI: 10.1109/ACCESS.2019.2920398.
- [104] Zhang, X., et al., "Game Theory Design for Deceptive Jamming Suppression in Polarization MIMO Radar," *IEEE Access*, Vol. 7, 2019. DOI: 10.1109/ACCESS.2019.2931604.
- [105] Jia, L., et al., "Game-theoretic Learning Anti-jamming Approaches in Wireless Networks," *IEEE Communications*, 2022. Online: <https://arxiv.org/abs/2207.00159>.
- [106] Feng, C., et al., "A Radar Anti-Jamming Strategy Based on Game Theory With Temporal Constraints," *IEEE Access*, Vol. 10, 2022. DOI: 10.1109/ACCESS.2022.3200761.
- [107] Geng, J., et al., "Radar and Jammer Intelligent Game under Jamming Power Dynamic Allocation," *Remote Sensing*, Vol. 15, No. 3, 2023. DOI: 10.3390/rs15030581.
- [108] He, B., et al., "Game Theory and Reinforcement Learning in Cognitive Radar Game Modeling and Algorithm Research: A Review," *IEEE Sensors Journal*, Vol. 24, No. 20, 2024. DOI: 10.1109/JSEN.2024.3454121.
- [109] He, B., and H. Su, "A Review of Game Theory Analysis in Cognitive Radar Anti-jamming," *Journal of Electronics & Information Technology*, Vol. 43, No. 5, 2021. Online: <https://api.semanticscholar.org/CorpusID:245742182>.
- [110] Sundaram, R., et al., "Countering Smart Jammers: Nash Equilibria for Asymmetric Agility and Switching Costs," in *Classified US Military Communications*, 2013.

- [111] Blum, D., "A Game-Theoretic Analysis of Electronic Warfare Tactics with Applications to the World War II Era," M.S. thesis, Massachusetts Institute of Technology, Cambridge, MA, 2004. Online: <http://hdl.handle.net/1721.1/28756>.
- [112] Thanoon, M., "A System Development for Enhancement of Human-Machine Teaming," Ph.D. dissertation, Tennessee State University, 2019. Online: <https://digitalscholarship.tnstate.edu/dissertations/AAI13856376/>.
- [113] McDermott, P., et al., "Human-Machine Teaming Systems Engineering Guide," MITRE, Tech. Rep. AD1108020, 2018. Online: <https://apps.dtic.mil/sti/citations/AD1108020>.
- [114] Urlings, P., and L. Jain, "Teaming Human and Machine," in *Advances in Intelligent Systems for Defence*, World Scientific, 2002. DOI: 10.1142/9789812776341_0002.
- [115] Taylor, R., "Towards Intelligent Adaptive Human Autonomy Teaming," Defence Science, and Technology Laboratory, UK, Tech. Rep. STO-MP-HFM-300, 2018. Online: <https://tinyurl.com/HAT-2018-NATO>.
- [116] Demirjian, E., et al., "Superteams: The Development and Effect of Artificial Intelligence Teammates on Organizational Performance," *Naval Engineers Journal*, Vol. 136, No. 1-2, 2024. Online: <https://tinyurl.com/demirjian-superteams>.
- [117] Zhang, L., Z. Ji, and B. Chen, "CREW: Facilitating Human-AI Teaming Research," *Transactions on Machine Learning Research*, 2024. Online: <https://openreview.net/pdf?id=ZRXwHRXm8i>.
- [118] Vaccaro, M., A. Almaatouq, and T. Malone, "When Combinations of Humans and AI Are Useful: A Systematic Review and Meta-Analysis," *Nature Human Behaviour*, Vol. 8, No. 12, 2024. DOI: 10.1038/s41562-024-02024-1.
- [119] Schmutz, J., et al., "AI-teaming: Redefining Collaboration in the Digital Era," *Current Opinion in Psychology*, Vol. 58, 2024. DOI: 10.1016/j.copsyc.2024.101837.
- [120] Smith, C., *Ethical Artificial Intelligence (AI)*, 2020. DOI: 10.1184/R1/c.4953576.v1.
- [121] Overholt, J., and K. Kearns, *Air Force Research Laboratory Autonomy Science & Technology Strategy*, 2014. Online: <https://tinyurl.com/afrl-autonomy>.
- [122] Haigh, K. Z., S. Varadarajan, and C. Y. Tang, "Automatic Learning-based MANET Cross-Layer Parameter Configuration," in *Workshop on Wireless Ad Hoc and Sensor Networks*. IEEE, 2006. Online: <https://www.cs.cmu.edu/~khaigh/papers/haigh06a-configuration.pdf>.
- [123] Hilario, M., "An Overview of Strategies for Neurosymbolic Integration," in *Connectionist-Symbolic Integration*, Mahwah, NJ: Lawrence Erlbaum, 1997. Online: <https://tinyurl.com/neurosymbolic-1997>.
- [124] D'Avila Garcez, A., L. Lamb, and D. Gabbay, *Neural-Symbolic Cognitive Reasoning*. New York, NY: Springer, 2009.
- [125] Confalonieri, R., et al., "A Historical Perspective of Explainable Artificial Intelligence," *Data Mining and Knowledge Discovery*, 2020. DOI: 10.1002/widm.1391.
- [126] Mitola III, J., "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio," Ph.D. dissertation, Royal Institute of Technology (KTH), Kista, Sweden, 2000. Online: <http://www.diva-portal.org/smash/get/diva2:8730/FULLTEXT01.pdf>.
- [127] Berleemann, L., S. Mangold, and B. Walke, "Policy-based Reasoning for Spectrum Sharing in Radio Networks," in *Symposium on New Frontiers in Dynamic Spectrum Access Networks*, 2005. DOI: 10.1109/DYSPAN.2005.1542612.

- [128] Gustavsson, P., et al., "Machine Interpretable Representation of Commander's Intent," in *International Command and Control Research and Technology Symposium: C2 for Complex Endeavors*, 2008. Online: <https://tinyurl.com/commander-intent>.
- [129] Gustavsson, P., et al., "Formalizing Operations Intent and Effects for Network-Centric Applications," in *System Sciences*, 2009. DOI: 10.1109/HICSS.2009.219.
- [130] Irandoust, H., and A. Benaskeur, "Human-autonomy Teaming for Critical Command and Control Functions," in *Human-Machine Systems*, IEEE, 2020. DOI: 10.1109/ICHMS49158.2020.9209513.
- [131] Heilemann, F., and A. Schulte, "Interaction Concept for Mixed-Initiative Mission Planning On Multiple Delegation Levels in Multi-UCAV Fighter Missions," in *Intelligent Human Systems Integration*. Springer, New York, NY, 2019. DOI: 10.1007/978-3-030-11051-2_106.
- [132] Irandoust, H., et al., "A Mixed-Initiative Advisory System for Threat Evaluation," in *International Command and Control Research and Technology Symposium*, 2010. Online: <https://tinyurl.com/mixed-init-threats>.
- [133] Miller, C., M. Pelican, and R. Goldman, "'Tasking' Interfaces for Flexible Interaction with Automation: Keeping the Operator in Control," in *Human Interaction with Complex Systems*, 2000. Online: <https://tinyurl.com/tasking-interface>.
- [134] Miller, C., et al., "The Playbook™ Approach to Adaptive Automation," in *Human Factors and Ergonomics Society*, 2005. DOI: 10.1177/154193120504900105.
- [135] Pomerleau, M., "Here's What the Army is Saying about Its New Electronic Warfare Solution," *C4ISRNet*, 2018. Online: <https://tinyurl.com/pomerleau-c4isrnet>.
- [136] US Army, *Electronic Warfare Planning and Management Tool (EWPMT)*. Online: <https://tinyurl.com/ewpmt-army>. Accessed 2020-09-19.
- [137] Raytheon, *Electronic Warfare Planning Management Tool (EWPMT)*. Online: <https://tinyurl.com/ewpmt-rtn>. Accessed 2020-09-19.
- [138] Mostafa, S., M. Ahmad, and A. Mustapha, "Adjustable Autonomy: A Systematic Literature Review," *Artificial Intelligence Review*, No. 51, 2019. DOI: 10.1007/s10462-017-9560-8.
- [139] Reinmund, T., et al., "Variable Autonomy through Responsible Robotics: Design Guidelines and Research Agenda," *ACM Transactions on Human-Robot Interaction*, Vol. 13, No. 1, 2024. DOI: 10.1145/3636432.
- [140] Basich, C., et al., "Learning to Optimize Autonomy in Competence-Aware Systems," in *International Joint Conference on Autonomous Agents and MultiAgent Aystems*, 2020.
- [141] Bakirtzis, G., M. Chiou, and A. Theodorou, *Negotiating Control: Neurosymbolic Variable Autonomy*, 2024. Online: <https://arxiv.org/abs/2407.16254>.
- [142] Parsons, S., et al., "Argument Schemes for Reasoning About Trust," *Argument and Computation*, Vol. 5, No. 2-3, 2014. DOI: 10.1080/19462166.2014.913075.
- [143] Sarkadi, S., et al., "Modelling Deception using Theory of Mind in Multi-Agent Systems," *AI Communications*, Vol. 32, No. 4, 2019. DOI: 10.3233/AIC-190615.
- [144] *International Competition on Computational Models of Argumentation*. Online: <https://argumentationcompetition.org/>. Accessed 2025-02-17.
- [145] Irandoust, H., et al., "A Mixed-initiative Advisory System for Threat Evaluation," in *Command and Control Research and Technology Symposium*, 2010.

- [146] Geng, B., P. Varshney, and M. Rangaswamy, "Information Integration from Human and Sensing Data," in, Mishra, K., B. Shankar, and M. Rangaswamy, (Eds.), Raleigh, NC: SciTech Publishing, 2023.
- [147] Fürnkranz, J., and E. Hüllermeier, *Preference Learning*. Heidelberg, Germany: Springer Verlag, 2011.
- [148] Chen, K., et al., "Learning Hierarchical Task Networks with Preferences from Unannotated Demonstrations," in *Robot Learning*, Vol. 155, 2021. Online: <https://proceedings.mlr.press/v155/chen21d.html>.
- [149] Kostkova, P., G. Jawaheer, and P. Weller, "Modeling User Preferences in Recommender Systems," *ACM Transactions on Interactive Intelligent Systems*, Vol. 4, 2014. DOI: 10.1145/2512208.
- [150] Bantouna, A., et al., "An Overview of Learning Mechanisms for Cognitive Systems," *Wireless Communications and Networking*, No. 22, 2012. DOI: 10.1186/1687-1499-2012-22.
- [151] Ng, A., and S. Russell, "Algorithms for Inverse Reinforcement Learning," in *ICML*, 2000.
- [152] Arora, S., and P. Doshi, "A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress," *Artificial Intelligence*, Vol. 297, 2021. DOI: 10.1016/j.artint.2021.103500.
- [153] Kaufmann, T., et al., "A Survey of Reinforcement Learning from Human Feedback," *Submitted to Transactions on Machine Learning Research*, 2024. Online: <https://arxiv.org/abs/2312.14925>.
- [154] Pattanayak, K., V. Krishnamurthy, and C. Berry, "Meta-Cognition. An Inverse-Inverse Reinforcement Learning Approach for Cognitive Radars," in *Information Fusion*, 2022. DOI: 10.23919/FUSION49751.2022.9841368.
- [155] Kim, K. J., et al., "An Application of Inverse Reinforcement Learning to Estimate Interference in Drone Swarms," *Entropy*, Vol. 24, No. 10, 2022. DOI: 10.3390/e24101364.
- [156] Zare, M., et al., "A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges," *IEEE Transactions on Cybernetics*, Vol. 54, No. 12, 2024. DOI: 10.1109/TCYB.2024.3395626.
- [157] Celemin, C., et al., "Interactive Imitation Learning in Robotics: A Survey," *Foundations and Trends® in Robotics*, Vol. 10, No. 1-2, 2022. DOI: 10.1561/23000000072.
- [158] Gombolay, M., et al., "Apprenticeship Scheduling: Learning to Schedule from Human Experts," in *IJCAI*, 2016. Online: <https://dspace.mit.edu/handle/1721.1/116062>.
- [159] Grover, R., et al., "Better Apprenticeship Learning with LLM Explanations," in *AAAI Spring Symposium*, 2024.
- [160] Chen, L., R. Paleja, and M. Gombolay, *Towards Sample-efficient Apprenticeship Learning from Suboptimal Demonstration*, 2021. Online: <https://arxiv.org/abs/2110.04347>.
- [161] Mishra, K., B. Shankar, and M. Rangaswamy, (Eds.), *Next-Generation Cognitive Radar Systems*. Raleigh, NC: SciTech Publishing, 2023.

7

Real-Time In-Mission Planning and Learning

Situation assessment and decision making are key components to all EW systems. Chapters 4–6 presented approaches for these activities without putting them in context of *in-mission* performance.

Traditional approaches to EW replanning assumed lengthy timescales between creating EP/EA plans and updating ES models. EW reprogramming is even slower because it requires a human-in-the-loop, often requiring days or months between the identification of a new signal and the ability to characterize and respond to it. In a fully integrated modern EW system the system can, and *must*, respond to unexpected events that happen during the mission, and learn from its experiences.

Figure 7.1 shows the four activities that must happen on-board the platform(s) with minimal human supervision. The ES module updates histories and probabilities based on mission progress, and identifies novel conditions. BDA compares expected effects to observed effects. The decision-making modules of EP and EA must perform both (1) replanning to update actions based on changing conditions,

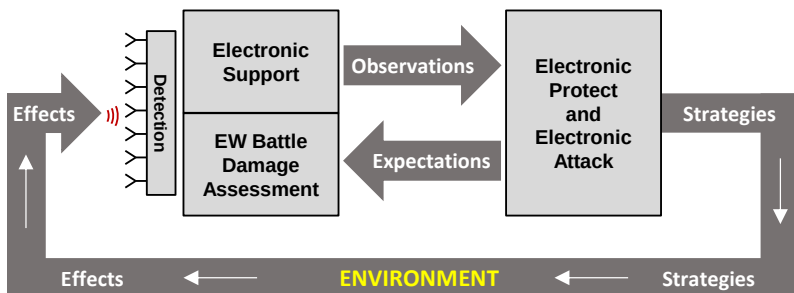


Figure 7.1 In-mission activity includes real-time updates to ES, BDA, and decision making. Note that strategies may be internal to the system, and do not require transmission, for example changing ES parameters.

and (2) learning to update models based on in-mission experience. Section 7.1 describes execution monitoring, which tracks the progress of all in-mission activities including EBM, ES, EP, and EA (including BDA). Section 7.2 describes replanning, and Section 7.3 describes in-mission learning.

7.1 Execution Monitoring

Execution monitoring tracks the outcomes of actions, giving a planner the ability to adapt to changes, and a learner the ability to incorporate experience. An EW system must monitor execution at all levels: ES, EP/EA, and EBM. Monitoring must occur on each platform, and across the team of multiple nodes. Execution monitoring requires ES capabilities, and becomes an introspective component of ES.

ES describes a situation or environment.

In contrast, execution monitoring compares the observed environment to the expected environment.

Execution monitoring is broader than BDA in that it monitors *all* progress toward mission goals, while BDA has traditionally only meant evaluating the effect of jamming actions. Monitoring actions illustrated in Figure 7.2 include:

- *Sensor monitoring*: Is the raw data correct? Are the sensors operating correctly? Is the inferred current situation rare or anomalous? Can the sensor information be trusted? Is there an adversarial attack on the sensors?

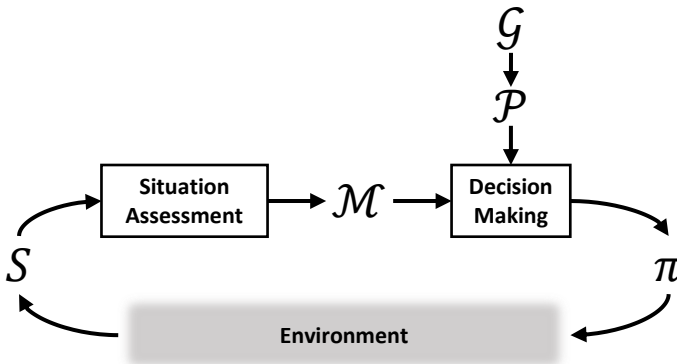


Figure 7.2 In uncertain or dynamic domains, execution monitoring must analyze sensors S , models M , actions π , plans P , and goals G .

- *Model monitoring*: Are the ES models correct? Is the situation assessment correct? Does the learned model cover the current situation; is the situation within the domain-of-expertise? Is there an adversarial attack on the situational inferences? When the system also learns in-mission, is the concept drift acceptable?
- *Action monitoring*: Are the assumptions or preconditions still true for an action? After acting, did one of the expected outcomes happen? How well did the action perform? Were there minor, but possibly critical, variations on the expected outcome(s)?
- *Plan monitoring*: Is the remainder of the plan still feasible? Can the system still accomplish its goals? Do any tasks need to be reallocated? Will the costs or time to accomplish the mission change?
- *Goal monitoring*: Before acting, do the long-term goals still make sense? Has the mission changed in a meaningful way? Are there new opportunities to accomplish additional objectives?

A failure of one of these questions might trigger *replanning* (Section 7.2) or trigger *learning* (Section 7.3). Replanning involves updating the plan to handle the new situation; learning updates and corrects the underlying models.

ES/BDA capabilities evaluate expected conditions against the current conditions. When the feedback matches that of the learned models, the feedback enables in-mission learning updates. For example, execution monitoring can compute the model's prediction error to trigger retraining, as in Example 1.1.

By tying actions to observations, execution monitoring can compute the causal relationships between events. This analysis allows the system to determine whether events are an inherent function of the observed environment, or whether actions are true precursors. The system can then use precursor events as "red flags" of imminent danger. Section 4.6 describes how to detect these patterns.

7.1.1 EW Battle Damage Assessment

EW BDA is a type of action monitoring, usually associated with evaluating the performance of a jamming action. EW BDA has traditionally been performed using our understanding of physics combined with engineering constraints. EWOs use their training and experience to develop a "gut instinct" of whether an action succeeded. A hand-engineered rules-based BDA system might set thresholds for alerts, or construct rules. Each rule might contribute a score; scores can be combined arithmetically or used as votes for the overall effectiveness of the action. As emitters become more complex, hand-generated rules quickly become brittle. The transition to automated on-board BDA capabilities requires a similar approach: Training a ML

model to make good inferences about the effectiveness of actions. Using ground truth collected during a predeployment test phase, an ML model can be trained to infer these values, and then used at runtime.

In EP missions, the performance metric can often be directly *measured*, (e.g., BER or throughput). For an EA system, ES must instead *infer* the effectiveness of the attack. (See Section 2.3 for more details.) Examples of inferred values include jamming effectiveness [1], radar mode, Pfa, Pd, or POI.

An EW BDA system can infer effectiveness by observing whether a threat's mode or other behavior changed. These features—akin to a “tell” in gambling—act as indicators that something is influencing the victim's performance. Detecting these subtle changes can help assess the impact of the jamming, much like reading a gambler's tell can reveal hidden intentions. Both rely on recognizing small but significant deviations from expected behavior to draw conclusions about the situation at hand. For example, if the system changes its frequency hopping pattern, that may indicate that it detected interference and is attempting to evade it. Likewise, changing to a more robust modulation technique, such as switching from QAM64 to QPSK, might indicate reduced comms network throughput. Different radar modes vary different parameters to accomplish different tasks [2–5], including for example pulse width to improve search capability, PRI to change detection range [3], or frequency agility to enhance EP properties. Changes in these characteristics may indicate EA effectiveness [5–11]. As a case example, an observing receiver can detect when a transmitting radar has shifted from long-range to short-range detection by observing changes such as:

- *Increased PRF.* Long-range radar uses a lower PRF to avoid range ambiguity and allow longer listening times between pulses, while short-range radar requires a higher PRF to increase update rates and improve target tracking.
- *Pulse-width reduction.* Long-range radar uses longer pulse widths to maximize transmitted energy and improve detection at greater distances. Short-range radar often uses shorter pulses for better range resolution and target discrimination.
- *Carrier frequency shift.* A shift to higher frequencies may indicate a focus on short-range detection, as they provide better resolution at the cost of range.
- *Beamwidth changes.* Long-range radars use narrower beams to focus energy over greater distances. Short-range radars may employ wider beams for broader coverage.
- *Change in scan pattern.* Long-range radars may use slower, wide-area search patterns, while short-range radars often increase scan rates for quicker updates on fast-moving nearby targets.

- *Change in modulation or waveform.* Long-range detection may use simple pulses or chirped signals for maximizing range, while short-range modes may use higher-resolution waveforms for improved accuracy.

Historical patterns and causal event analysis can add to the evidence. For example, when a radar parameter changes several times in a row, the system can increase its confidence in the effectiveness of the EA technique.

Recall that feature engineering should focus on features that lead to better assessment of the environment and thus actionable inferences (Section 4.2). For BDA, Wang et al. [12] use features with the following characteristics:

- *Systematicity:* Reflect the comprehensive characteristics of the system being evaluated.
- *Simplicity:* Eliminate excess elements and retain indicators with greater significance.
- *Objectivity:* Align with the system being evaluated, ensuring clarity in what each indicator represents.
- *Timeliness:* Evolve based on the changing requirements of the user.
- *Measurability:* Allow for quantitative description when possible.
- *Completeness:* Capture the entire capability of the system being evaluated and how the factors influence one another.
- *Consistency:* The relationship between the system's effect targets and each indicator should be clear and consistent.

Additionally, the features must reflect the response of the threat object following interference; they should remain unaffected by the jammer's orientation and distance to the threat object; and the signal parameters should be minimally influenced by environmental factors or anti-jamming measures [12].

Rather than relying on brittle hand-generated rules, incorporate these features into an ML model. Engineering expertise and understanding of the physics of EW can help select effective features, and an ML can combine them to infer BDA. Wang et al. [12] use a *grey analytical hierarchy process* [13, 14] to infer radar jamming effectiveness: Engineering expertise selects over-the-air observables, grey theory determines the relative importance of the features, and ANNs and SVMs learn the relationship between the features and the jamming effect, similar to the approach in Section 4.3. Chu et al. [15] use an SVM to estimate BDA in networked radar systems, and Ling et al. [16] use data fusion to estimate the effectiveness of range deception jamming on a ground-based radar network. A similar analytic process can be used in cybersecurity by analyzing the effects on different attack categories [17] and thus forming the basis for ML-based BDA in cyberspace [18]; Kim et al. [19]

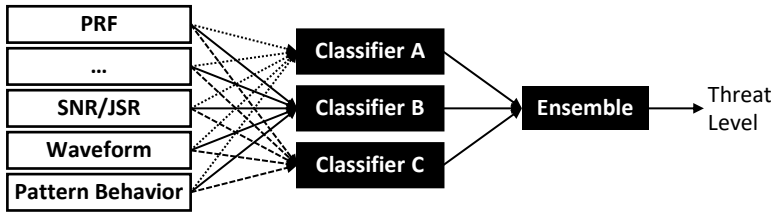


Figure 7.3 Ensemble methods can increase accuracy by selecting from or combining multiple hypotheses. Features can be hand-generated or learned, and any ML method can be used as an intermediate classifier.

found that a k NN had the best results. Jamming effects can likewise be estimated for electro-optical systems [20, 21]. Zeng et al. [22] use the analytic hierarchy process to determine individual factors, grey theory to determine their relative importance, a fuzzy expert system to recommend solutions, and human judgment to increase solution flexibility and validation.

Inverse RL can determine one aspect of whether a system is cognitive, namely if it uses a constrained utility maximizer [23, 24]. Krishnamurthy et al. show how to compute other adversarial characteristics, including inverse tracking, the adversary’s sensor gain, or how it computes signal-to-interference-plus-noise [25].

Any ML model, including rule learning, may be appropriate as a BDA scoring engine to estimate effectiveness. Figure 7.3 illustrates a possible ML architecture. Each feature (e.g., modulation, behavior) could be derived by a hand-generated algorithm, or inferred by ML (traditional, DeepNet, or hybrid). Multiple classifiers use these features to compute their local threat mode estimates, and the ensemble method (Section 3.1.3) chooses the best threat mode. Finally, change detection compares the current mode to the previous mode(s). While this figure illustrates a classification model, a regression model can also be used for numerical estimation.

Algorithm 7.1 shows a corresponding code listing for an ensemble of different classifiers chosen from `scikit-learn`: a decision tree, an extra tree, a Gaussian naïve Bayes, and two k NN models using different values of k . Figure 7.4 shows the results.

The 10-fold cross-validation in `evalModel()` yields 10 scores per model, presented in a box-and-whisker plot. A k -fold cross-validation trains k models; each model uses a different random $1/k$ of the dataset as its test examples. In a box-and-whisker plot, the central box shows the range of the middle half of the data (first quartile to third quartile), and whiskers correspond to 1.5 times the interquartile

Algorithm 7.1 Ensemble methods can achieve higher accuracy than independent models. Scikit-learn contains multiple classifiers and ensemble methods.

```

from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.ensemble import VotingClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score, KFold
from matplotlib import pyplot
import numpy as np

def getData(): # Returns a fake dataset
    return make_classification( n_samples=1000,n_features=25,
                               n_informative=15, n_redundant=5)

def getModels():
    return [ ('CART', DecisionTreeClassifier()),
             ('RandTree', ExtraTreeClassifier()),
             ('GNB', GaussianNB()),
             ('kNN-3', KNN(n_neighbors=3)),
             ('kNN-7', KNN(n_neighbors=7))]

def evalModel(name, model, X, y):
    cv = KFold(n_splits=10)
    scores = cross_val_score(model, X, y, scoring='accuracy',
                              cv=cv, error_score='raise')
    s = '{:10s} mu={:.3f} '.format( name,np.mean(scores) )
    s = s+'std={:.3f} scores={}'.format( np.std(scores),scores )
    print(s)
    return scores

if __name__ == '__main__':
    names,results=[],[]
    X,y = getData()
    models = getModels()
    for name,model in models:
        results.append( evalModel( name, model, X, y ) )
        names.append(name)
    ensemble = VotingClassifier( estimators=models )
    results.append( evalModel( 'Ensemble', ensemble, X, y ) )
    names.append('Ensemble')
    pyplot.boxplot(results, labels=names)
    pyplot.savefig('ensemble.png')

```

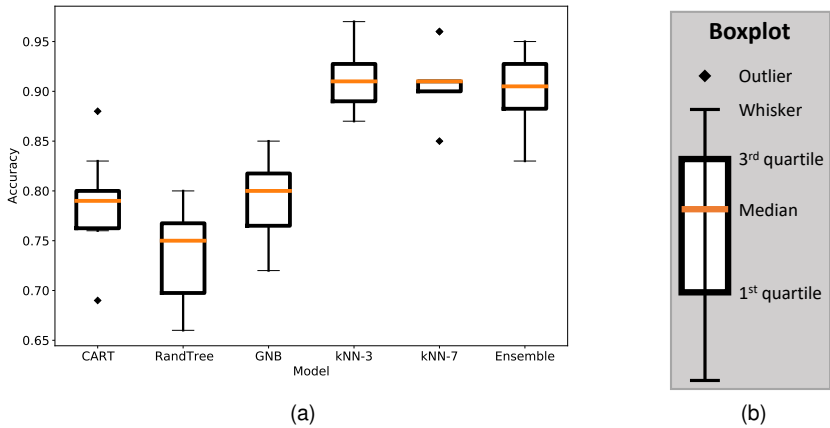


Figure 7.4 Ensemble methods can achieve higher accuracy than independent models. (a) Each boxplot shows the 10-fold cross validation scores for the model. (b) A *box-and-whisker* plot shows a distribution of values.

range (99.3% of the data if the data is normally distributed). Outliers are shown individually.

Building an ML model to estimate BDA is an improvement over traditional approaches that operate at human-time with human senses. On-board ML that operates submillisecond can incorporate features beyond a human EWO’s ability to perceive. The model can combine our understanding of physics and engineering with human gut instinct to infer at fast real time whether an action succeeded. Moreover, feedback obtained during a mission allows the system to observe performance after each action and under different conditions, thereby collecting a richer dataset for post-mission analysis.

7.2 In-Mission Replanning

No plan of operations reaches with any certainty beyond the first encounter with the enemy’s main force.

— *Helmuth von Moltke, 1880, Kriegsgeschichtliche Einzelschriften*
Often misquoted as “No plan survives contact with the enemy.”

In-mission replanning ensures that the plan remains feasible, despite surprises and changes in mission goals. New priorities can emerge, and resources may be

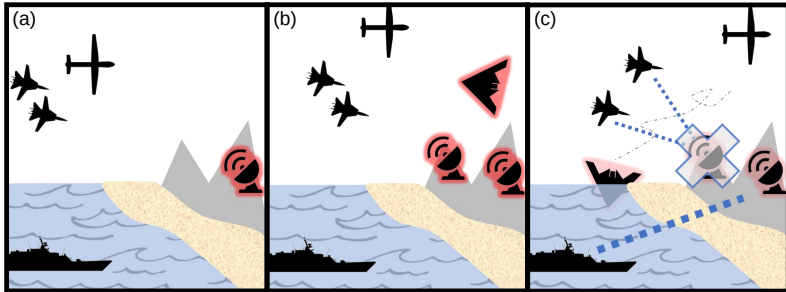


Figure 7.5 In-mission replanning must adjust to unexpected situations: (a) a priori electronic intelligence anticipated one enemy radar; (b) in-mission collaborative ES detected two enemy radars and an aerial drone; and (c) in-mission replanning adjusts, and the nodes neutralize the enemy.

unexpectedly depleted. Some planning approaches, such as conditional planning and MDPs, incorporate options for *expected* outcomes. The planner must also always update the plan to handle *unexpected* outcomes (both positive and negative). Figure 7.5 illustrates a possible scenario where replanning adapts to the situation.

In all cases, execution monitoring triggers the replanning process. An action might have failed to achieve expected results and it must be repeated or replaced, for example, if BDA indicates that the chosen EA method wasn't sufficiently effective and a different technique(s) must be used. The existing plan may no longer be achievable, such as when a platform failed and tasks need to be reallocated. The existing plan may no longer be relevant because the mission goals changed, and a completely new plan must be generated.

Engineers can often estimate the *types* of failures, even if *specific* failures are hard to foresee. For example, one can anticipate platform attrition, but not know which platforms (and thus which tasks) will be compromised. In conditional planning, the planner prepares actions for multiple possible outcomes (usually the most mission-critical), leaving other actions for dynamic replanning because they are less important, less likely, or easy to handle at runtime [26]. Planners can incorporate flexibility to surprise as part of the decision criteria. Planners can also plan to communicate to handle expected changes in mission goals [27].

Planners have the option to perform a complete *replan* from the current state, or *repair* the existing plan [28–30]. The decision can depend, for example, on the algorithmic approach, computation time, or desire for plan stability. Anytime approaches manage decision making when time is limited (Section 5.4). Plan repair techniques usually depend on the chosen planning approach, for example:

- Interleaving planning and execution [31, 32],
- Contingent planning with sensing [33–35],
- HTN planning [36–40],
- Probabilistic planning [41],
- Task allocation [42, 43],
- Graph planning [44, 45],
- Analogical planning [46], and
- Plan editing [47, 48].

A common theme of these approaches is that of *plan stability*, wherein the new plan is similar to the original [45–49]. This approach reduces computation, reduces coordination among participants, and supports human operators more effectively: Humans do not like significant changes to plans without clear justification [50]. AFRL’s Distributed Operations (DistrO) program addresses this concept directly, looking at plan creation and plan repair [51]. Components assess progress toward goals, identify deviations from expected outcomes, and recommend adjustments to the plan for forward nodes in an anti-access area-denial (A2/AD) environment. A key concept is to minimize adjustments to the plan because nodes have made commitments, and may not be able to communicate any changes. It may also be useful to factor the cost of changing the plan into the planning process.

7.3 In-Mission Learning

Reinforcement learning (RL), in its broadest definition, is a goal-directed learning approach wherein individuals interact with the environment to improve its performance over time [52]. The environment becomes the teacher. In EW, RL means that the system can take an action in the environment, collect feedback, and evaluate its own performance; for this reason, RL is also called *self-supervised learning*.¹

Real-world environments are usually too complex to collect data that covers all expected situations. In EW, moreover, systems will encounter novel conditions that cannot be captured in any lab setting. RL is related to direct adaptive control of nonlinear systems [53] and model-predictive control (MPC) [54, 55]. The critical difference is that MPC is model-based, while RL is not. RL systems do not have a differentiable objective function, and must take an action in the environment to learn the function. RL allows the system to learn in situ, where learning is most beneficial, and most needed.

RL can update action descriptions to capture what actually happens in the environment, for example updating transition probabilities $P(s'|s, a)$, or action

1. In contrast, a supervised learner is passive and given the examples; see Chapter 3.

preconditions (p_1, p_2, p_3) to (p_1, p_2, p_4) , for $p_i \in \text{state } s$. RL is the only way to accurately learn performance models $m = f(o_n, c_n)$ for novel RF conditions, as introduced in Section 4.3.

Figure 7.6 presents a simplified example of in-mission incremental learning. At time $t - \delta$, the optimizer uses the previously learned model f to estimate the performance m of each candidate strategy s , given the known observables o :

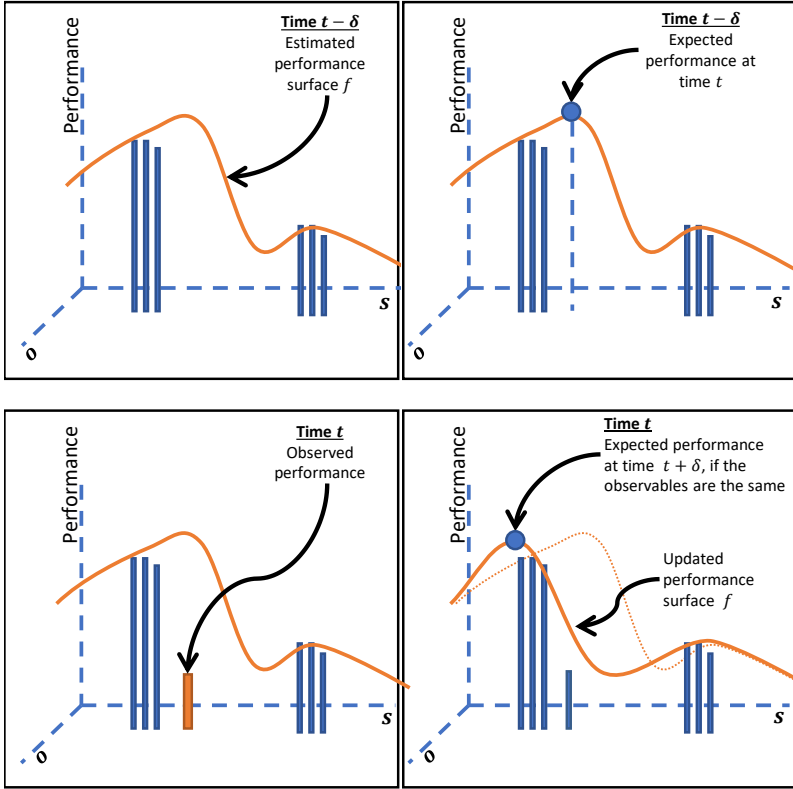


Figure 7.6 At time $t - \delta$, the BBN SO (Example 1.1, [56]) uses the model $m_k = f_k(o, s)$ to estimate m_k based on the (known) observables and all (candidate) strategies o, s , and chooses the best strategy. The learner updates f_k for the next iteration. (To simplify the illustration, the x - and y -axes are multidimensional: x captures all values of all controllables, and y captures all values of all observables. Note also that the sketch shows f_k only for the given $o_{t-\delta}$ and o_t , but f_k is a model for all possible observable values and all strategies for metric m_k .)

$m = f(\mathbf{o}_n, \mathbf{c}_n) = f(o, s)$. After choosing and then executing the best strategy $s = \operatorname{argmax}_{s_i} \tilde{U}_n(s_i)$, the system observes the actual performance of s_i . The learner adds the result to the dataset, updates the model f , and the optimizer starts using the updated function f . In the sketch, if $\mathbf{o}_n(t) \approx \mathbf{o}_n(t - \delta)$, then the optimizer chooses the indicated best strategy. Figure 5.2 illustrates an example control flow that incorporates in-mission learning of performance metrics.

The two key actions in every RL system are (1) when to update models, presented in Section 7.3.1, and (2) what actions to take, presented in Section 7.3.2. Section 7.3.3 presents some RL approaches.

7.3.1 Updating Models

Most RL systems update models at every time step, but the choice can be deliberate. For example, the system might trigger retraining in batches if compute cycles are limited, or if there is new information. Essentially, every expectation or prediction failure is an opportunity to learn. New information includes performance metric feedback for new actions in a known environment, known actions in new environments, or unexpected feedback for a known action in a known environment. Edge cases, strange sensor readings, and adversarial attacks are also candidates.

The BBN SO (Example 1.1) updates the models on error as illustrated in Figure 7.7. When the predicted performance is different from the observed performance, the RRE instructs the LTRE to retrain the models. If the estimated performance of $m_k(t)$ differs sufficiently from the observed performance $\hat{m}_k(t + \delta$

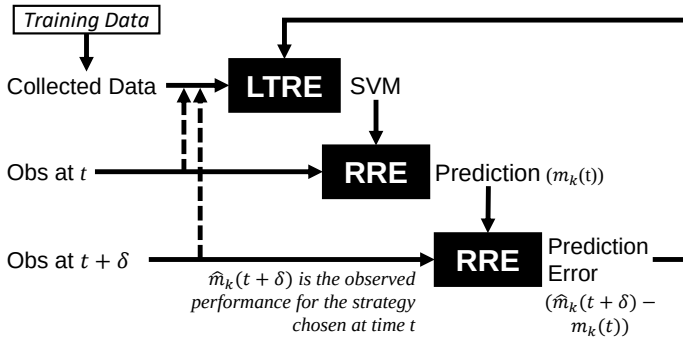


Figure 7.7 The BBN SO updates the SVM models when it detects an error in the model.

δ),² then the RRE triggers a retraining event, and the LTRE rebuilds the SVM for m_k using the new training data. Figure 10.9 illustrates a scenario in which each of the eight environments is novel to the BBN SO; the SO only retrains three times because the model can generalize from the known environments to the novel ones.

7.3.2 Exploitation, Exploration, and Active Learning

In terms of choosing actions, the RL-based decision maker makes an explicit explore-exploit trade-off. *Exploitation* means that the decision maker chooses the strategy with the highest expected utility, while *exploration* means that it chooses actions that will acquire new knowledge. (The utility function can also incorporate value of information directly, per Section 6.1.4.) The decision balances reward maximization based on previously acquired knowledge with attempting new actions to increase knowledge. Even in human decision making, the willingness to fail increases learning rate [57].

Active learning is the RL concept of deliberately choosing what actions to take, namely choosing actions that are most useful to learning. The learner identifies the most informative actions to learn from, rather than exploring randomly. The actions may reduce uncertainty, inspect edge cases or challenging states, or have a high likelihood of increasing the expected performance. Section 6.1.4 described the concepts of planning to sense or planning to communicate; these are both forms of active learning. The concept of *probing* in EW is a type of active learning. Deep active learning approaches use information uncertainty, diversity, and expected model change to select experiments [58].

A good simple exploration approach is for the decision maker to choose strategies that it knows little or nothing about. It can manage bias (Section 8.1.2) and maintain data diversity (Section 8.3.2) by recording which strategies it has used against which environments, using a sparse matrix such as Table 7.1. The decision maker clusters environments using observable vectors (Section 2.1.1), adding new classes as needed. Exploration decisions help fill out this matrix.

In Figure 4.5, when the decision maker is operating in environments 1, 3, 4, or 5, there is a clear exploitation choice (c_3 , c_1 , c_1 , and c_1 , respectively). The strategy to exploit in environment 2 is less clear because it has no data for c_3 , and performance estimates may be below possible performance for c_1 , but overall c_3 is a good choice. Environment 6, meanwhile, the best strategy to exploit could be any of c_1 , c_4 , or c_7 . In exploration mode, the decision maker has many more choices. In general, *there is no wrong answer when exploring*. While random exploration can

2. The BBN SO uses $\delta = 1$ for immediate feedback. Section 6.1.3 described how to incorporate delayed rewards.

Table 7.1
Example of a Sparse Matrix Used by a Decision Maker to Track
Strategies Used Against RF Environments

Env	None	One Value					Two Values					Three Values	
	000	001	002	010	100		011	012	101	102	110	111	112
Env1	Y	Y	Y	Y	Y		Y	-	-	-	Y	-	-
Env2	Y	Y	Y	Y	Y		-	-	Y	-	Y	-	-
Env3	Y	Y	Y	Y	Y		-	-	Y	Y	-	-	-
Unknown	-	-	-	-	-		-	-	-	-	-	-	-

In this example, controllables c_1 and c_2 are binary on/off, and c_3 can take the values 0, 1, and 2, yielding 12 strategies. "Y" indicates when the system has performance results for that strategy in that environment, "-" otherwise. (Figure 2.6 shows other information that might be captured in this matrix.)

be beneficial, active learning requires a reasonable justification for a given choice. In environment 2, c_3 is likely the best choice for exploration because it may lead to the best performance. In environment 6, c_1 is likely the best choice, but c_4 and c_7 are also good choices, from the perspective of quantifying the range of possible outcomes. In environment 4, c_9 is an interesting choice because it appears to be a good second-best strategy; moreover, given the similarity between environment 4 and 5, if c_9 yields much higher performance than estimated, it may impact the performance surface for environment 5 as well.

7.3.3 Reinforcement Learning Approaches

Much of the common research in RL is based on MDPs to the extent that RL is almost synonymous with MDP. It's not: MDPs may or may not be the underlying model.

RL is not defined by the learning method, but by the learning problem and direct interaction with the environment [52].

In fact, MDP-based RL is generally inappropriate for in-mission EW due to computational complexity and the number of training samples required.

In principle, [value prediction] can use any method for supervised learning from examples, including ANNs, decision trees, and various kinds of multivariate regression. However, not all function approximation methods are equally well suited for use in RL. The most sophisticated ANN and statistical methods all assume a static training set over which multiple passes are made. In RL, however, it is important that learning be able to occur on-line,

while interacting with the environment or with a model of the environment. To do this requires methods that are able to learn efficiently from incrementally acquired data. In addition, RL generally requires function approximation methods to handle nonstationary target functions (target functions that change over time).

— Richard Sutton and Andrew Barto, 2018 [52]
Turing award winners, 2024

Callout 10.2 presents evaluation metrics and Section 3.9 discusses some of the algorithmic trade-offs to consider when choosing a model, and what to learn in-mission. Figure 10.10 shows the performance of multiple ML models as a function of how much a priori training data they received, and Figure 11.4 shows the trade-off between performance and computation time.

Table 7.2 summarizes some of the most common RL methods, and compares them by the model type, management of the explore/exploit trade-off, training complexity, and common applications. *Value-based RL* estimates the expected return of actions to derive a policy (e.g., Q-learning), while *policy-based RL* directly optimizes the policy without needing a value function (e.g., policy-gradient methods). The ϵ -greedy method is a simple strategy that balances exploration and exploitation by choosing a random action with probability ϵ and the best-known action otherwise and is commonly used in value-based methods.

Multiarmed bandit (MAB) and SVM models are well-suited for EW applications because they operate in single-state decision-making scenarios (i.e., without explicit state transitions), enabling a lower-complexity approach that can learn from limited data. While Q-learning and Deep Q-learning are the most popular RL techniques in the literature, both are computationally intensive and require substantial a priori training data before performing well. Actor-critic approaches are effective in continuous action spaces, where MAB and Q-learning might be inefficient. Monte Carlo tree search (MCTS) dynamically builds a tree through execution, making it useful for environments with limited a priori data. Bayesian RL and cognitive architectures are the only ones with an explicit model of uncertainty.

Wang et al. [59] surveys deep RL, and the book *Deep RL Hands-On* [60] presents Python examples. Mismar presents code on github for deep RL in 5G [61]. Kuzdeba presents a variety of deep RL applications for RF [62], and includes a discussion of deployment concerns.

Complementary work includes structuring the utility function to understand how carefully the reward function needs to be maintained [63], optimizing the utility function while satisfying constraints [64, 65], and using Kalman filters to improve

Table 7.2
Comparison of Common RL Methods

Method	Model	Exploration vs Exploitation	Complexity
SVMs	Hyperplane for classification or regression	Exploration via action selection or ϵ -greedy	Low to moderate (simple actions)
Multiarmed bandit	No model (single state, arms only)	Exploration via action selection	Low to moderate (simple actions)
Randomized search	Population of potential solutions	High exploration via population diversity	Moderate to high (population-based optimization)
Monte Carlo tree search	Tree-based search with rollouts	Balanced via upper confidence bound	Moderate to high (tree expansion)
Policy-gradient	Directly optimize policy	Balanced via policy gradient updates	Moderate to high (single network)
Actor-critic models	Learn policy and value functions	Balanced via policy and value updates	Moderate to high (dual networks)
Q-learning	Learn action-value function	Balanced via ϵ -greedy	Moderate (value function or Q-values)
Deep Q-learning	Q-function learned by DeepNet	Balance via ϵ -greedy and experience replay	High (requires neural network training)
Bayesian RL	Models environment uncertainty	Exploration guided by uncertainty	High (involves posterior updates)
Cognitive architectures	Rules and cognitive decision models	Perceptual attention and action selection	Moderate to high (multiple cognitive layers)

stability [66] or handle more complex settings [67]. Delayed or discounted reward is discussed in Section 6.1.3.

7.3.3.1 Support Vector Machines and Other Traditional ML Models

SVMs (Section 3.1.1) are a particularly effective method for RL value prediction in EW. They learn from small numbers of training examples (even just a single example), and do not require large compute capability. The BBN SO uses SVMs to perform real-time in-mission learning for communications EP [56], and triggers retraining based on prediction error (Example 1.1, Section 7.3.1, and Figure 10.9).

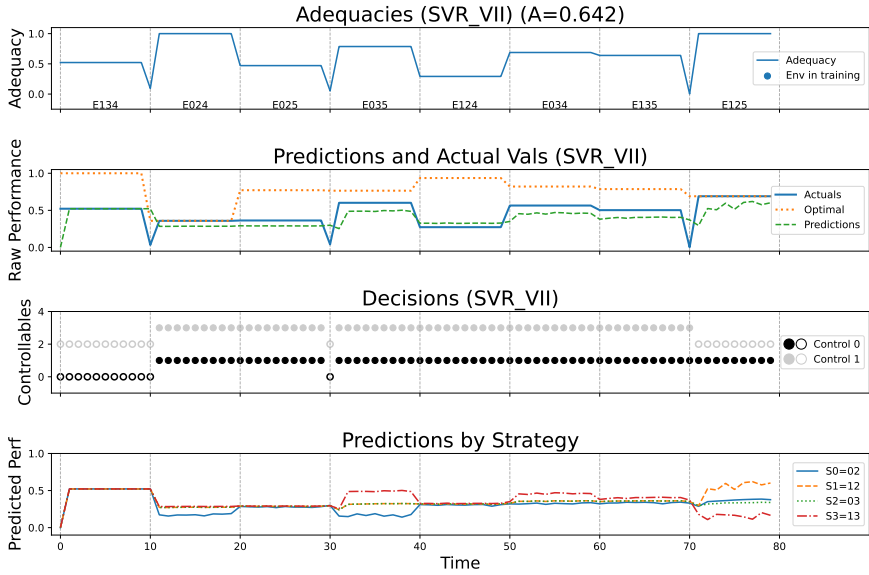


Figure 7.8 An SVM with the Pearson VII kernel achieves an Adequacy score of 0.642 by incrementally learning new environments, even when pretrained on no data; all RF environments are surprises. Each of the eight environments is present for ten time units; the decision maker can choose one of four strategies. (Compare to Figure 11.3 for decision trees and some pretraining.) *Top row:* Adequacy at each time step; no environments were used for a priori training. *Second row:* Predicted performance of the chosen strategy, actual performance of the chosen strategy, and optimal performance of the best strategy. *Third row:* Control decisions. *Bottom row:* Predictions of how well each of the four strategies will perform.

Despite encountering eight novel RF environments, in-mission learning only requires three retraining events to achieve an Adequacy score of 0.88, compared to an adaptive system that only achieves 0.28. The BBN SO uses SVMs with the Pearson VII kernel [68]. The Pearson VII performs better than other kernels in most domains. Even when presented with no a priori training data—a completely empty dataset—the BBN SO accomplishes an Adequacy score of 0.642 because it is able to learn in-mission from direct experience (Figure 7.8). The model only retraining when the actual performance is different from the observed performance, and there is no exploration mode that might potentially find better solutions. For example, the first environment, E134, it doesn't find the optimal solution because the prediction is exactly what was observed.

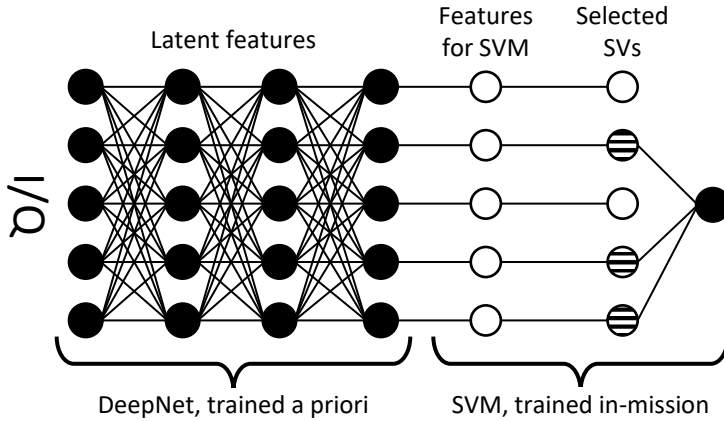


Figure 7.9 Before the mission, a DeepNet learns from a large set of typical environments to extract latent features. During the mission, the system uses the trained DeepNet for inference, and then uses these features to train (and update) an SVM model.

All other traditional ML approaches can be used as function approximators using the same approach as SVMs, including k NNs and decision trees (Figure 11.3). Figure 10.10 shows that SVMs perform better than many other ML models when learning how to handle novel RF environments, even when the initial training data was completely empty. Figure 11.4 shows that SVMs outcompete many other models for timing performance as well.

Combining DeepNets with SVMs is a promising area for research [69–71]. The DeepNet model can be trained a priori on a large corpus of data to extract latent features of RF signals. One would then replace the output layer of the DeepNet with an SVM that can be updated in mission, at timescales relevant to RF missions, to learn how to classify novel emitters. Figure 7.9 illustrates this concept. SVMs learn rapidly from very little data, and model updates can be computed on FPGAs or CPUs in submillisecond timeframes.

7.3.3.2 Multiarmed Bandit

Multiarmed bandit (MAB) is a classic RL approach where the rewards have probability distributions rather than fixed amounts [72, 73]. Each action (strategy s) is an “arm” and has a reward distribution R_s with an associated mean value μ_s . MABs model a situation where the environment is static and the same decision has

to be made repeatedly. A bandit can be viewed as an MDP with only one state and multiple available actions; equivalently, an MDP is a set of MAB problems because the state changes. MAB converges faster than MDPs. A useful variation of MAB adds context to each action, thus learning the reward distribution associated with the context; i.e., for $m = f(o, s)$, the distribution of values for m , when using strategy s under observables o . MABs have been used for a wide variety of applications, including channel selection [74–76], anti-jamming [77, 78], jamming [79, 80], searching in space surveillance radar [81], and multiplayer CRN management [82]. Takeuchi et al. [83] develop a simplified MAB algorithm for channel selection that rapidly forgets past parameters and thus reduces computational costs.

Several anytime MAB algorithms exist [84–86], correlated to the idea of how much information is enough [87]. Metalearning can also help choose the best MAB heuristic [88], and MAB can become a metalearner to choose effective algorithms [89, 90]. MAB is also used for smart experimental design (Section 10.7).

7.3.3.3 Randomized Search Methods

Randomized search methods (Section 5.1.2), especially genetic algorithms (GAs), search for the optimal policy through a population-based search, without explicitly involving policy gradients or value functions. They are particularly useful in complex, unstructured optimization problems, including multiobjective optimization and evolving controllers for physically embodied systems such as robots. They explore large, unstructured search spaces by maintaining diversity in the population, and therefore can adapt to more complex environments. These methods can be used to evolve ANN structure for deep RL tasks, and can also be combined with Bayesian methods for improved exploration and uncertainty handling.

7.3.3.4 Policy-Gradient Methods

Policy-gradient methods directly optimize the policy by estimating the gradient of the expected reward with respect to the policy parameters. They directly optimize the policy using gradient ascent on expected rewards. The policy is usually stochastic, in that it defines a probability distribution over actions, and the agent samples actions according to this distribution. Policy-gradient methods can suffer from high variance in the gradient estimates, and they may require variance-reduction techniques like the approach developed in REINFORCE [91].

Deterministic policy gradients (DPG) [92] introduced the approach that handled continuous action spaces better than standard stochastic policy gradients. *Trust*

region policy optimization (TRPO) updates policies by taking the largest step possible to improve performance, while satisfying a constraint that ensures stable learning [93]. *Proximal policy optimization (PPO)* simplifies TRPO with a simpler first-order method that is significantly easier to implement, and still perform as well as TRPO [94]. *Deep deterministic policy gradient (DDPG)* [95] is based on deterministic policy gradients, enhanced with DeepNets.

Group relative policy optimization (GRPO), the optimization approach introduced by DeepSeekMath [96], eliminates the value function, and instead normalizes rewards over a group of samples. This approach makes GRPO simpler to train and reduces memory and computational complexity. GRPO also incorporates the KL divergence term into the loss function,³ leading to a more stable training process and improved overall performance.

7.3.3.5 Monte Carlo Tree Search

Monte Carlo tree search (MCTS) is a model-free, tree-based RL method that balances exploration and exploitation through statistical sampling. It incrementally builds a search tree by simulating or executing action sequences and using the results to refine future decisions. MCTS is effective in sequential decision-making problems where the state transition dynamics are unknown or too complex to model explicitly. Unlike value-based methods such as Q-learning, MCTS does not require extensive prior training data but instead relies on iterative simulations (or real-world execution) to estimate action values. Its computational complexity grows with the branching factor and the number of simulations, making it resource intensive.

The foundational MCTS method uses four main steps [97, 98]: selection, expansion, simulation, and backpropagation. The selection step navigates the existing search tree using a selection policy such as upper confidence bound for trees (UCT) [99], which balances exploration and exploitation. Expansion adds new nodes to the tree, allowing the method to refine its search over time. In the simulation step, random rollouts are performed from a newly expanded node to estimate long-term rewards. Finally, the backpropagation step updates the estimated action values for all nodes along the visited path.

AlphaGo [100] used MCTS with DeepNets, where a deep policy network guides the tree search, and a value network improves rollout efficiency. MCTS has been used for radar scheduling [101], multifunction radar surveillance [102],

3. Kullback-Leibler divergence measures how much a model's probability distribution Q differs from a true probability distribution P . Most approaches use KL divergence as a diagnostic tool to monitor and limit policy divergence, which is not as effective or efficient as incorporating it directly into the loss function.

waveform design for radar tracking [103], anti-jamming [104], wireless network optimization [105], and bandwidth slicing [106].

7.3.3.6 Actor-Critic Methods

Actor-critic methods combine value-based and policy-based methods. The *actor* learns a policy network that decides which action to take based on the current state. It outputs a probability distribution over actions. The *critic* estimates the function that evaluates how good the current state-action pair is, estimating the state-value function or the action-value function. It helps to improve the policy by providing feedback to the actor. The actor decides the next action to take based on the policy, while the critic evaluates the chosen action's effectiveness, similar to how BDA works in EW. Actor-critic methods provide a low-variance estimate of expected returns, leading to more stable and sample-efficient learning. To encourage exploration, some actor-critic methods incorporate entropy regularization, which prevents the policy from becoming overly deterministic too quickly.

One of the earliest actor-critic method was *Tabular Actor-Critic* [107], which used a discrete action-state space without function approximation. *Linear function approximators* [108] enabled handling of large or continuous state spaces while still keeping the policy and value functions interpretable. Williams [91] introduced *advantage actor-critic (A2C)*, which reduces the variance by subtracting the state value from the reward estimate. The *asynchronous advantage actor-critic (A3C)* method is a parallel training method for deep RL that improves sample efficiency and stability [109, 110].

7.3.3.7 Q-Learning and SARSA

Q-learning and *state-action-reward-state-action (SARSA)* are model-free RL approaches to learn quality (Q-value) of actions. They estimate the action-value function for a given state-action pair, which represents the expected cumulative reward that can be obtained by taking an action from a state and following the respective policy thereafter.

Q-learning is an *off-policy* approach for situations when being optimal quickly is important. It updates the value function based on the maximum value of the next state, reflecting an assumption that the best action will always be taken. It will therefore learn the optimal policy, regardless of the actual actions taken, as long as it does enough exploration. Q-learning finds the optimal policy for any finite MDP given infinite exploration time [111]. The key word here is *infinite*: Q-learning is sample-inefficient, and moreover has high computational complexity [112, 113].

SARSA, meanwhile, is *on-policy*: It updates the value function based on the actions actually taken, and thus might not converge to the optimal policy if the exploration strategy is badly designed. SARSA can do better with fewer examples when it has a good exploration strategy. It performs better in situations where exploration is critical, where outcomes are stochastic, where the optimal policy might change over time, and in safety-critical applications.

Q-learning and SARSA have been widely used for various applications, including managing spectrum sharing in radar [113], predicting interference [114], anti-jamming [115], resource management [116], trajectory optimization for mobile wireless access points [117], and even understanding user behavior [118].

7.3.3.8 Artificial Neural Networks and Deep Q-Learning

When time and data are plentiful, *ANNs* (Section 3.1.2) effectively learn a variety of tasks in RF in an RL environment [119]. Many examples in Chapter 4 are based on ANNs; the RL framework trains these models using real environmental feedback rather than static precollected data. Some examples of pre-DeepNet work include learning performance [120, 121], interference control in CRNs [122], and selecting tasks [123].

Deep Q-Networks (DQN) use a DeepNet to estimate the Q-values of all possible actions of each MDP state [124], and thus reduce the computational burden (and number of samples required) of conventional Q-learning. DQNs have been used for a variety of RF tasks including signal classification [125, 126], detecting and organizing attack types [127], coexistence [128, 129], spectrum sharing radar [113], task scheduling [130], resource scheduling [131–133], jamming [134–136], and anti-jamming [137–144].

The main challenges limiting use of ANNs for RL are: (1) processing speed (low-latency inference); (2) data scarcity (ANNs need a lot of training data); and (3) learning from a single example. Qu et al. [127] suggests techniques to address some of the challenges, including computational efficiency and asymmetric data (class imbalance). Methods are being developed to maintain accuracy while dramatically reducing memory size and inference time, such as [62, 145, 146]. Low-shot learning (Section 3.4) is a promising approach to learning from very few examples.

Multiagent deep RL (MADRL) extends deep RL to settings where multiple agents interact to achieve their goals. The agents operated in a shared environment, and may cooperate or compete, directly matching EW, where multiple platforms operate in the same theater. Example implementations include enhancing RF fingerprinting [147], distributed jamming [148, 149], and anti-jamming [150–152]. MADRL has several key challenges, including scalability, credit assignment, and

nonstationarity where the environment is constantly changing. Other RL approaches implicitly (rather than explicitly) handle the challenges of distributed RL.

7.3.3.9 Bayesian RL

Bayesian RL uses Bayesian methods to handle uncertainty in MDPs and exploration strategies [153, 154]. Bayesian RL maintains a distribution over key concepts, including model parameters, the policy, or the reward function. The Bayesian posterior naturally captures the full state of knowledge, subject to the chosen parametric representation, and thus, the agent can select actions that maximize the expected gain with respect to this information state. Bayesian RL has a natural exploration-exploitation balance because it favors exploring actions that reduce uncertainty. The uncertainty helps guide the exploration of the environment. Bayesian RL has been used in RF applications (e.g., [155–157]) and can be combined with other RL approaches.

7.3.3.10 Cognitive Architectures

Many *cognitive architectures* learn rules of one form or another to help make decisions. Many of these have been used in physical domains, such as robot soccer and fighter pilots [158–163]. RL formalizes the explore-exploit trade-off; the cognitive architecture literature uses similar concepts of *perceptual attention* and *action selection*. (Haykin uses this terminology for the *perception-action cycle* [164].) Modern cognitive architectures have incorporated ANNs, embraced probabilistic reasoning, and implemented concepts of motivation [165–169]. Cognitive architectures seen limited use in RF [164, 170, 171], but have multiple advantages that should be explored:

- Structured decision making using symbolic reasoning, memory, and goals.
- Goal-driven behavior allowing for dynamic reprioritization of objectives.
- Faster learning via prior domain knowledge and transfer learning.
- Memory and long-term dependencies with explicit working memory and episodic memory.
- Robustness to sparse rewards due to heuristics, rules, and memory.
- Hybrid reasoning that integrates symbolic AI concepts like logic and rules.
- Explainability and transparency due to symbolic reasoning.

Cognitive architectures are ideal for goal-driven, explainable, memory-based, and logical reasoning tasks.

7.4 Conclusion

No plan survives contact with the enemy, and no model accurately captures every detail of the environment. An EW system must respond to these unexpected, in-mission events, and learn from its experiences. This chapter has discussed ways of achieving real-time in-mission planning and learning. Specifically, it focused on concepts of execution monitoring, in-mission replanning, and in-mission (machine) learning. Unexpected observations, inferences, and changes must be promptly conveyed to the human users via the HMI (Section 6.3).

Interaction with the real environment mandates constant oversight for planning, and it is the replanning that keeps the mission on track. Moreover, interaction with the real environment enables ML to improve empirical models; in-mission learning improves performance based on real experience. Finally, execution monitoring closes the loop, because it ties (1) the actions to the observations, (2) the decision making to the situation assessment, and (3) the EP/EA/EBM to the ES.

References

- [1] Lee, G.-H., J. Jo, and C. H. Park, "Jamming Prediction for Radar Signals Using Machine Learning Methods," *Security and Communication Networks*, 2020. Online: <https://www.hindawi.com/journals/scn/2020/2151570/>.
- [2] Melvin, W., and J. Scheer, *Principles of Modern Radar: Radar Applications*. Raleigh, NC: SciTech, 2013. Online: <https://www.iqpc.com/media/7813/18055.pdf>.
- [3] Aslan, M., "Emitter Identification Techniques in Electronic Warfare," M.S. thesis, Middle East Technical University, 2006. Online: <https://etd.lib.metu.edu.tr/upload/12607675/index.pdf>.
- [4] Morgan, T., *How Do the Different Radar Modes of a Modern Fighter Aircraft Work?* 2016. Online: <https://tinyurl.com/radarmodes>. Accessed 2021-02-02.
- [5] Avionics Department, "Electronic Warfare and Radar Systems Engineering Handbook," Naval Air Warfare Center Weapons Division, Tech. Rep. NAWCWD TP 8347, 2013. Online: <https://tinyurl.com/navy-ew-2013>.
- [6] Chairman of the Joint Chiefs of Staff, *Joint Publication 3-09: Joint Fire Support*, 2019. Online: https://fas.org/irp/doddir/dod/jp3_09.pdf.
- [7] Zhou, D., et al., "Battle Damage Assessment with Change Detection of SAR Images," in *Chinese Control Conference*, 2015. DOI: 10.1109/ChiCC.2015.7259637.
- [8] Basseville, M., and I. Nikiforov, *Detection of Abrupt Changes: Theory and Application*. Hoboken, NJ: Prentice-Hall, 1993.
- [9] Choi, S., et al., "A Case Study: BDA Model for Standalone Radar System (Work-in-Progress)," *Software Security and Assurance*, 2018. DOI: 10.1109/ICSSA45270.2018.00020.
- [10] US Army, *Electronic Warfare Battle Damage Assessment (A14-065)*, 2014. Online: <https://legacy.www.sbir.gov/node/561555>. Accessed 2025-03-05.
- [11] Parker, M., *Digital Signal Processing 101: Everything You Need to Know to Get Started*, 2nd Edition. Newnes, UK: Newnes, 2017.
- [12] Wang, S., et al., *Principle and Technology of Cognitive Electronic Warfare*. Beijing, China: National Defense Industry Press, 2018.

- [13] Saaty, R., "The Analytic Hierarchy Process—What it is and How it is Used," *Mathematical Modelling*, Vol. 9, No. 3, 1987. DOI: 10.1016/0270-0255(87)90473-8.
- [14] Hu, X., *Electronic Jamming Effect Evaluation Method*, 2011. (Not available in English).
- [15] Chu, Z., N. Xiao, and J. Liang, "Jamming Effect Evaluation Method Based on Radar Behavior Recognition," *Journal of Physics Conference Series*, Vol. 1629, No. 1, 2020. DOI: 10.1088/1742-6596/1629/1/012001.
- [16] Ling, Q., et al., "Range Deception Jamming Performance Evaluation for Moving Targets in a Ground-Based Radar Network," *Electronics*, Vol. 12, No. 7, 2023. DOI: 10.3390/electronics12071614.
- [17] Kim, S., et al., "Study on Cyber Attack Damage Assessment Framework," *IEEE Access*, 2022. DOI: 10.1109/ACCESS.2022.3179977.
- [18] Choi, S., et al., "Method for Effectiveness Assessment of Electronic Warfare Systems in Cyberspace," *Symmetry*, Vol. 12, No. 12, 2020. DOI: 10.3390/sym12122107.
- [19] Kim, D., et al., "Cyber Battle Damage Assessment Framework and Detection of Unauthorized Wireless Access Point Using Machine Learning," in *Frontier Computing*, Singapore: Springer, 2018.
- [20] Gao, W., "Evaluation Method of Interference Effect on Electro-Optical Imaging System," *Optoelectronic Engineering*, Vol. 33, No. 2, 2006.
- [21] Liu, S., et al., "A Novel Effectiveness Evaluation Method of Smokescreen Jamming Based on the Fusion of Directional Gradients and Local Texture Features," *AIP Advances*, Vol. 12, No. 11, 2022. DOI: 10.1063/5.0114734.
- [22] Zeng, F., et al., "Hybridising Human Judgment, AHP, Grey Theory, and Fuzzy Expert Systems for Candidate Well Selection in Fractured Reservoirs," *Energies*, Vol. 10, No. 4, 2017. DOI: 10.3390/en10040447.
- [23] Pattanayak, K., V. Krishnamurthy, and C. Berry, "Meta-Cognition. An Inverse-Inverse Reinforcement Learning Approach for Cognitive Radars," in *Information Fusion*, 2022. DOI: 10.23919/FUSION49751.2022.9841368.
- [24] Krishnamurthy, V., et al., "Identifying Cognitive Radars - Inverse Reinforcement Learning Using Revealed Preferences," *IEEE Transactions on Signal Processing*, Vol. 68, 2020. DOI: 10.1109/TSP.2020.3013516.
- [25] Krishnamurthy, V., et al., "Adversarial Radar Inference: Inverse Tracking, Identifying Cognition, and Designing Smart Interference," in, Mishra, K., B. Shankar, and M. Rangaswamy, (Eds.), Raleigh, NC: SciTech Publishing, 2023.
- [26] Benaskeur, A., E. Bossé, and D. Blodgett, "Combat Resource Allocation Planning in Naval Engagements," Defence R&D Canada, Tech. Rep. DRDC Valcartier TR 2005–486, 2007. Online: <https://apps.dtic.mil/sti/pdfs/ADA605291.pdf>.
- [27] Yordanova, V., "Intelligent Adaptive Underwater Sensor Networks," Ph.D. dissertation, University College London, London, UK, 2018. Online: <https://discovery.ucl.ac.uk/id/eprint/10042482/>.
- [28] Cushing, W., and S. Kambhampati, "Replanning: A New Perspective," in *ICAPS*, 2005. Online: <http://rakaposhi.eas.asu.edu/replan-will.pdf>.
- [29] Garrido, A., C. Guzman, and E. Onaindia, "Anytime Plan-Adaptation for Continuous Planning," in *UK Planning and Scheduling Special Interest Group*, 2010. Online: <https://tinyurl.com/anytime-adapt>.

- [30] Babli, M., Ó. Sapena, and E. Onaindia, "Plan Commitment: Replanning versus Plan Repair," *Engineering Applications of Artificial Intelligence*, Vol. 123, 2023. DOI: 10.1016/j.engappai.2023.106275.
- [31] Haigh, K. Z., and M. Veloso, "Planning, Execution and Learning in a Robotic Agent," in *Artificial Intelligence Planning Systems*, 1998. Online: <https://www.aaai.org/Papers/AIPS/1998/AIPS98-015.pdf>, Summary of Haigh's Ph.D. dissertation.
- [32] Nourbakhsh, I., "Interleaving Planning and Execution," in *Interleaving Planning and Execution for Autonomous Robots*, Vol. 385, 1997. Online: <https://link.springer.com/book/10.1007/978-1-4615-6317-4>.
- [33] Brafman, R., and G. Shani, "Replanning in Domains with Partial Information and Sensing Actions," *Journal of Artificial Intelligence Research*, Vol. 45, 2012. Online: <https://arxiv.org/abs/1401.6048>.
- [34] Komarnitsky, R., and G. Shani, "Computing Contingent Plans Using Online Replanning," in *AAAI*, 2016. Online: <https://dl.acm.org/doi/10.5555/3016100.3016345>.
- [35] Maliah, S., R. Komarnitski, and G. Shani, "Computing Contingent Plan Graphs using Online Planning," *ACM Transactions on Autonomous and Adaptive Systems*, Vol. 16, No. 1, 2022. DOI: 10.1145/3488903.
- [36] Ayan, F., et al., "HOTRiDE: Hierarchical Ordered Task Replanning in Dynamic Environments," in *Planning and Plan Execution for Real-World Systems*, 2007. Online: <http://rpggoldman.goldman-tribe.org/papers/icans07-hotride.pdf>.
- [37] Lesire, C., et al., "A Distributed Architecture for Supervision of Autonomous Multi-Robot Missions," *Autonomous Robots*, Vol. 40, 2016. DOI: 10.1007/s10514-016-9603-z.
- [38] Höller, D., et al., "HTN Plan Repair via Model Transformation," in *Künstliche Intelligenz*, 2020. Online: https://link.springer.com/chapter/10.1007%2F978-3-030-58285-2_7.
- [39] Bansod, Y., et al., in *ICAPS Workshop on Hierarchical Planning*, 2021. Online: <https://www.cs.umd.edu/~nau/papers/bansod2021integrating.pdf>.
- [40] Goldman, R., et al., in *ICAPS Workshop on Hierarchical Planning*, 2024. Online: <https://www.cs.umd.edu/~nau/papers/goldman2024comparative.pdf>.
- [41] Chen, C., et al., "RPRS: A Reactive Plan Repair Strategy for Rapid Response to Plan Failures of Deep Space Missions," *Acta Astronautica*, Vol. 175, 2020. DOI: 10.1016/j.actaastro.2020.05.011.
- [42] Beal, J., et al., "Adaptive Task Reallocation for Airborne Sensor Sharing," in *Foundations and Applications of Self* Systems*, 2016. DOI: 10.1109/FAS-W.2016.46.
- [43] Buckman, N., H.-L. Choi, and J. How, "Partial Replanning for Decentralized Dynamic Task Allocation," in *Guidance, Navigation, and Control*, 2019. Online: <https://arxiv.org/abs/1806.04836>.
- [44] Gerevini, A., A. Saetti, and I. Serina, "Planning through Stochastic Local Search and Temporal Action Graphs," *Journal of Artificial Intelligence Research*, Vol. 20, 2003. DOI: 10.1613/jair.1183.
- [45] Fox, M., et al., "Plan Stability: Replanning Versus Plan Repair," in *ICAPS*, 2006. Online: <https://cdn.aaai.org/ICAPS/2006/ICAPS06-022.pdf>.
- [46] Veloso, M., *Planning and Learning by Analogical Reasoning*. Heidelberg, Germany: Springer Verlag, 1994.
- [47] van der Krogt, R., and M. de Weerd, "Plan Repair as an Extension of Planning," in *ICAPS*, 2005. Online: <https://aaai.org/papers/icans-05-017-plan-repair-as-an-extension-of-planning/>.

- [48] Bidot, J., B. Schattenberg, and S. Biundo, "Plan Repair in Hybrid Planning," in *Künstliche Intelligenz*, 2008. Online: https://link.springer.com/chapter/10.1007%2F978-3-540-85845-4_21.
- [49] Saetti, A., and E. Scala, "Optimising the Stability in Plan Repair via Compilation," in *Automated Planning and Scheduling*, Vol. 32, AAAI Press, 2022. DOI: 10.1609/icaps.v32i1.19815.
- [50] Vasileiou, S. L., et al., "A Logic-Based Explanation Generation Framework for Classical and Hybrid Planning Problems," *Journal of AI Research*, Vol. 73, 2022. Published: Apr 25, 2022. DOI: 10.1613/jair.1.13431.
- [51] Marsh, G., "Distributed Operations," Air Force Research Labs, Tech. Rep. BAA-AFRL-RIK-2016-0003, 2015. Online: <http://www.fbodaily.com/archive/2015/12-December/25-Dec-2015/FBO-03977354.htm>.
- [52] Sutton, R., and A. Barto, *Reinforcement Learning: An Introduction*. Denver, CO: Bradford, 2018.
- [53] Sutton, R., A. Barto, and R. Williams, "Reinforcement Learning Is Direct Adaptive Optimal Control," *IEEE Control Systems Magazine*, Vol. 12, No. 2, 1992. DOI: 10.1109/37.126844.
- [54] Ernst, D., et al., "Reinforcement Learning Versus Model Predictive Control: A Comparison on a Power System Problem," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 39, No. 2, 2009. DOI: 10.1109/TSMCB.2008.2007630.
- [55] Görges, D., "Relations between Model Predictive Control and Reinforcement Learning," *IFAC-PapersOnLine*, Vol. 50, No. 1, 2017. IFAC World Congress. DOI: 10.1016/j.ifacol.2017.08.747.
- [56] Haigh, K. Z., et al., "Parallel Learning and Decision Making for a Smart Embedded Communications Platform," BBN Technologies, Tech. Rep. BBN-REPORT-8579, 2015. Online: <http://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-SO.sm.pdf>.
- [57] Rzhetsky, A., et al., "Choosing Experiments to Accelerate Collective Discovery," *National Academy of Sciences*, 2015. DOI: 10.1073/pnas.1509757112.
- [58] Ren, P., et al., *A Survey of Deep Active Learning*, 2020. DOI: 10.48550/arXiv.2009.00236.
- [59] Wang, X., et al., "Deep Reinforcement Learning: A Survey," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 35, No. 4, 2024. DOI: 10.1109/TNNLS.2022.3207346.
- [60] Lapan, M., *Deep Reinforcement Learning Hands-On*. Birmingham, UK: Packt Publishing, 2018.
- [61] Mismar, F., *Deep Reinforcement Learning for 5G Networks (github)*. Online: <https://tinyurl.com/deep-rf-for-5g>. Accessed 2025-01-26.
- [62] Kuzdeba, S., *Radio Frequency Machine Learning: A Practical Deep Learning Perspective*. Norwood, MA: Artech House, 2025.
- [63] Regan, K., and C. Boutilier, "Regret-based Reward Elicitation for Markov Decision Processes," in *Uncertainty in AI*, 2009. DOI: 10.48550/arXiv.1205.2619.
- [64] Efroni, Y., S. Mannor, and M. Pirotta, *Exploration-Exploitation in Constrained MDPs*, 2020. Online: <https://arxiv.org/abs/2003.02189>.
- [65] Taleghan, M., and T. Dietterich, "Efficient Exploration for Constrained MDPs," in *AAAI Spring Symposium*, 2018. Online: <https://tinyurl.com/mdp-exploration>.
- [66] Mowlavi, S., and M. Benosman, "Reinforcement Learning-Based Estimation for Spatio-Temporal Systems," *Scientific Reports*, Vol. 14, No. 1, 2024. DOI: 10.1038/s41598-024-72055-1.

- [67] Marino, F., and G. Guglieri, "Beyond Static Obstacles: Integrating Kalman Filter with Reinforcement Learning for Drone Navigation," *Aerospace*, Vol. 11, No. 5, 2024. DOI: 10.3390/aerospace11050395.
- [68] Üstün, B., W. Melssen, and L. Buydens, "Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel," *Chemometrics and Intelligent Laboratory Systems*, Vol. 81, No. 1, 2006. DOI: 10.1016/j.chemolab.2005.09.003.
- [69] Ma, M., et al., "Ship Classification and Detection Based on CNN Using GF-3 SAR Images," *Remote Sensing*, Vol. 10, 2018. DOI: 10.3390/rs10122043.
- [70] Wagner, S., "SAR ATR by a Combination of Convolutional Neural Network and Support Vector Machines," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 52, No. 6, 2016. DOI: 10.1109/Taes.2016.160061.
- [71] Gao, F., et al., "Combining Deep Convolutional Neural Network and SVM to SAR Image Target Recognition," in *Internet of Things*. IEEE, 2017. DOI: 10.1109/iThings-GreenCom-CPSCoM-SmartData.2017.165.
- [72] Lattimore, T., and C. Szepesvari, *Bandit Algorithms*. Cambridge, UK: Cambridge University Press, 2019. Online: <https://tor-lattimore.com/downloads/book/book.pdf>.
- [73] Slivkins, A., *Introduction to Multi-Armed Bandits*. Delft, Netherlands: Now Publishers, 2019. Online: <https://arxiv.org/abs/1904.07272>.
- [74] Jouini, W., C. Moy, and J. Palicot, "On Decision Making for Dynamic Configuration Adaptation Problem in Cognitive Radio Equipments: A Multi-Armed Bandit Based Approach," 2010. Online: <https://tinyurl.com/mab-cog-radio>.
- [75] Liu, K., and Q. Zhao, "Channel Probing for Opportunistic Access with Multi-channel Sensing," *Conference on Signals, Systems and Computers*, 2008. DOI: 10.1109/ACSSC.2008.5074369.
- [76] Lu, J., et al., "Dynamic Multi-Arm Bandit Game Based MultiAgents Spectrum Sharing Strategy Design," in *Digital Avionics Systems Conference*, 2017. Online: <https://arxiv.org/abs/1711.04365>.
- [77] Li, H., J. Luo, and C. Liu, "Selfish Bandit-Based Cognitive Anti-Jamming Strategy for Aeronautic Swarm Network in Presence of Multiple Jammer," *IEEE Access*, 2019. DOI: 10.1109/ACCESS.2019.2896709.
- [78] Zhu, H., et al., "Online Decision-making Method for Frequency-agile Radar Based on Multi-Armed Bandit," *Journal of Radar*, Vol. 12, No. 1, 2023. DOI: 10.12000/JR23206.
- [79] Amuru, S., et al., "Jamming Bandits: A Novel Learning Method for Optimal Jamming," *IEEE Transactions on Wireless Communications*, Vol. 15, No. 4, 2016. DOI: 10.1109/TWC.2015.2510643.
- [80] ZhuanSun, S., J. Yang, and H. Liu, "An Algorithm for Jamming Strategy Using OMP and MAB," *EURASIP Journal on Wireless Communications and Networking*, 2019. DOI: 10.1186/s13638-019-1414-4.
- [81] Schwalm, C., H. Schily, and A. Charlish, "Using Multi-Armed Bandits for Search Applications in a Space Surveillance Radar Network," in *NEO and Debris Detection Conference*, 2023. Online: <https://conference.sdo.esoc.esa.int/proceedings/neosst2/paper/39/NEOSST2-paper39.pdf>.
- [82] Singh, R., et al., "Improving Decision-Making Policy for Cognitive Radio Applications Using Reinforcement Learning," in *Interdisciplinary Approaches in Technology and Management for Social Innovation*, Vol. 2, 2024. DOI: 10.1109/IATMSI60426.2024.10503303.

- [83] Takeuchi, S., et al., "Dynamic Channel Selection in Wireless Communications via a Multi-Armed Bandit Algorithm Using Laser Chaos Time Series," *Scientific Reports*, Vol. 10, Online: <https://www.nature.com/articles/s41598-020-58541-2>.
- [84] Jun, K.-S., and R. Nowak, "Anytime Exploration for Multi-Armed Bandits using Confidence Information," in *ICML*, 2016. Online: <http://proceedings.mlr.press/v48/jun16.pdf>.
- [85] Kleinberg, R., "Anytime Algorithms for Multi-Armed Bandit Problems," in *Symposium on Discrete Algorithms*, 2006. Online: <https://www.cs.cornell.edu/~rdk/papers/anytime.pdf>.
- [86] Besson, L., and E. Kaufmann, *What Doubling Tricks Can and Can't Do for Multi-Armed Bandits*, 2018. Online: <https://tinyurl.com/doubling-mab>.
- [87] Even-Dar, E., S. Mannor, and Y. Mansour, "Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems," *Journal of Machine Learning Research*, 2006. Online: <https://jmlr.csail.mit.edu/papers/volume7/evendar06a/evendar06a.pdf>.
- [88] Besson, L., E. Kaufmann, and C. Moy, "Aggregation of Multi-Armed Bandits Learning Algorithms for Opportunistic Spectrum Access," in *Wireless Communications and Networking Conference*, 2018. Online: https://perso.crans.org/besson/articles/BKM_...IEEE.WCNC.2018.pdf.
- [89] Gagliolo, M., and J. Schmidhuber, "Learning Dynamic Algorithm Portfolios," *Annals of Mathematics and Artificial Intelligence*, Vol. 47, 2006. DOI: 10.1007/s10472-006-9036-z.
- [90] Svegliato, J., K. Wray, and S. Zilberstein, "Meta-Level Control of Anytime Algorithms with Online Performance Prediction," in *IJCAI*, 2018. DOI: 10.24963/ijcai.2018/208.
- [91] Williams, R. J., "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Machine Learning*, Vol. 8, No. 3-4, 1992.
- [92] Silver, D., et al., "Deterministic Policy Gradient Algorithms," in *ICML*, 2014. Online: <https://proceedings.mlr.press/v32/silver14.pdf>.
- [93] Schulman, J., et al., "Trust Region Policy Optimization," in *ICML*, 2015. Online: <https://arxiv.org/abs/1502.05477>.
- [94] Schulman, J., et al., "Proximal Policy Optimization Algorithms," 2017. Online: <https://arxiv.org/abs/1707.06347>.
- [95] Lillicrap, T. P., et al., "Continuous Control with Deep Reinforcement Learning," in *ICLR*, 2016. Online: <https://arxiv.org/abs/1509.02971>.
- [96] Shao, Z., et al., *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*, 2024. Online: <https://arxiv.org/abs/2402.03300>.
- [97] Coulom, R., "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search," in *Computers and Games*, Herik, H. J. van den, P. Ciancarini, and J. Donkers, (Eds.), 2007.
- [98] Świechowski, M., et al., "Monte Carlo Tree Search: a Review of Recent Modifications and Applications," *Artificial Intelligence Review*, 2022. DOI: 10.1007/s10462-022-10228-y.
- [99] Kocsis, L., and C. Szepesvári, "Bandit Based Monte-Carlo Planning," in *European Conference on Machine Learning*, Fürnkranz, J., T. Scheffer, and M. Spiliopoulou, (Eds.), 2006.
- [100] Fu, M., "AlphaGo and Monte Carlo Tree Search: The Simulation Optimization Perspective," in *Winter Simulation Conference*, 2016. DOI: 10.1109/WSC.2016.7822130.
- [101] Akbar, S., A. Hebb, and R. Adve, "Development of Cognitive Approaches to Radar Scheduling, Detection and Tracking," Defence R&D Canada, Tech. Rep. DRDC-RDDC-2023-C199, 2023. Online: https://cradpdf.drdc-rddc.gc.ca/PDFS/unc437/p817000_A1b.pdf.

- [102] Vincent, M., "Reinforcement Learning for Multi-Function Radar Resource Management," Theses, Sorbonne Université, Paris, France, 2023. Online: <https://theses.hal.science/tel-04295965>.
- [103] Rybicki, B. W., and J. K. Nelson, "Voronoi Progressive Widening for Cognitive Radar Tracking with Large Waveform Libraries," in *IEEE Radar Conference*, 2024. DOI: 10.1109/RadarConf2458775.2024.10548077.
- [104] Wang, C., et al., "Anti-jamming Equilibrium Strategy Learning of Frequency Agile Radar based on Monte Carlo Tree Search," in *IEEE Radar Conference*, 2023. DOI: 10.1109/RADAR54928.2023.10371079.
- [105] Jian, X., and W. Xi, "Wireless Network Optimization Scheme Recommendation Using Monte Carlo Tree Search," in *ACM Big Data Technologies*, 2022. DOI: 10.1145/3565291.3565347.
- [106] Chen, Y.-H., "Federated Learning Architecture to Integrate AI Models from Different Internet Service Providers: Using Bandwidth Slicing Resource Management as Case Study," in *Control, Decision and Information Technologies*, 2023. DOI: 10.1109/CoDIT58514.2023.10284421.
- [107] Barto, A. G., R. S. Sutton, and C. W. Anderson, "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 13, No. 5, 1983.
- [108] Sutton, R. S., et al., "Policy Gradient Methods for Reinforcement Learning with Function Approximation," in *NeurIPS*, Vol. 12, 1999.
- [109] Mnih, V., et al., "Asynchronous Methods for Deep Reinforcement Learning," in *ICML*, 2016. Online: <https://arxiv.org/abs/1602.01783>.
- [110] Schulman, J., et al., "Trust Region Policy Optimization," in *ICML*, 2015. Online: <https://arxiv.org/abs/1502.05477>.
- [111] Melo, F., *Convergence of Q-Learning: A Simple Proof*, 2007. Online: <https://tinyurl.com/qlearn-proof>.
- [112] Tarbouriech, J., and A. Lazaric, "Active Exploration in Markov Decision Processes," in *Artificial Intelligence and Statistics*, 2019. Online: <http://proceedings.mlr.press/v89/tarbouriech19a/tarbouriech19a.pdf>.
- [113] Thornton, C., et al., *Experimental Analysis of Reinforcement Learning Techniques for Spectrum Sharing Radar*, 2020. Online: <https://arxiv.org/abs/2001.01799>.
- [114] Selvi, E., et al., "On the Use of Markov Decision Processes in Cognitive Radar: An Application to Target Tracking," in *IEEE Radar Conference*, 2018. DOI: 10.1109/RADAR.2018.8378616.
- [115] Wu, Q., et al., "Reinforcement Learning-Based Anti-Jamming in Networked UAV Radar Systems," *Applied Science*, Vol. 9, No. 23, DOI: 10.3390/app9235173.
- [116] Delaney, J., S. Dowe, and C.-T. Cheng, "Reinforcement-Learning-Based Robust Resource Management for Multi-Radio Systems," *Sensors*, Vol. 23, No. 10, 2023. DOI: 10.3390/s23104821.
- [117] Mohammadi, H., V. Marojevic, and B. Shang, "Analysis of Reinforcement Learning Schemes for Trajectory Optimization of an Aerial Radio Unit," in *IEEE Communications*, 2023. DOI: 10.1109/ICC45041.2023.10279738.
- [118] Alizadeh, P., "Elicitation and Planning in Markov Decision Processes with Unknown Rewards," Ph.D. dissertation, Sorbonne, Paris, France, 2016. Online: <https://tinyurl.com/mdp-elicitation>.
- [119] Yau, K.-L., et al., "Application of Reinforcement Learning in Cognitive Radio Networks: Models and Algorithms," *The Scientific World Journal*, 2014. DOI: 10.1155/2014/209810.

- [120] Troxel, G., et al., "Cognitive Adaptation for Teams in ADROIT," in *GLOBECOM*. IEEE, 2007. doi: 10.1109/GLOCOM.2007.923, Invited.
- [121] Tsagkaris, K., A. Katidiotis, and P. Demestichas, "Neural Network-Based Learning Schemes for Cognitive Radio Systems," *Computer Communications*, No. 14, 2008. DOI: 10.1016/j.comcom.2008.05.040.
- [122] Galindo-Serrano, A., and L. Giupponi, "Distributed Q-Learning for Aggregated Interference Control in Cognitive Radio Networks," *IEEE Transactions on Vehicular Technology*, Vol. 59, No. 4, 2010. DOI: 10.1109/TVT.2010.2043124.
- [123] Li, M., Y. Xu, and J. Hu, "A Q-Learning Based Sensing Task Selection Scheme for Cognitive Radio Networks," in *Wireless Communications & Signal Processing*, 2009. DOI: 10.1109/WCSP.2009.5371749.
- [124] Mnih, V., et al., "Human-Level Control through Deep Reinforcement Learning," *Nature*, Vol. 518, 2015. Online: <https://www.nature.com/articles/nature14236>.
- [125] Kulin, M., et al., "End-to-end Learning from Spectrum Data," *IEEE Access*, Vol. 6, 2018. DOI: 10.1109/ACCESS.2018.2818794.
- [126] Qu, Z., et al., "Radar Signal Intra-Pulse Modulation Recognition Based on Convolutional Neural Network and Deep Q-Learning Network," *IEEE Access*, Vol. 8, 2020. DOI: 10.1109/ACCESS.2020.2980363.
- [127] Qu, X., et al., "A Survey on the Development of Self-Organizing Maps for Unsupervised Intrusion Detection," *Mobile Networks and Applications*, 2019. DOI: 10.1007/s11036-019-01353-0.
- [128] Maglogiannis, V., et al., "A Q-Learning Scheme for Fair Coexistence Between LTE and Wi-Fi in Unlicensed Spectrum," *IEEE Access*, Vol. 6, 2018. DOI: 10.1109/ACCESS.2018.2829492.
- [129] Kozy, M., "Creation of a Cognitive Radar with Machine Learning: Simulation and Implementation," M.S. thesis, Virginia Polytechnic Institute, Blacksburg, VA, 2019. Online: <https://vtechworks.lib.vt.edu/items/e9b044ae-d970-4b46-af1f-5d50de7defbf>.
- [130] George, T., K. Wagner, and P. Rademacher, "Deep Q-Network for Radar Task-Scheduling Problem," in *IEEE Radar Conference*, 2022. DOI: 10.1109/RadarConf2248738.2022.9764230.
- [131] Ye, Y., et al., *A New Approach for Resource Scheduling with Deep Reinforcement Learning*, 2018. Online: <https://arxiv.org/abs/1806.08122>.
- [132] Li, R., et al., "Deep Reinforcement Learning for Resource Management in Network Slicing," *IEEE Access*, Vol. 6, 2018. DOI: 10.1109/ACCESS.2018.2881964.
- [133] Casterline, K., et al., "Applications of Machine Learning for Electronic Warfare Emitter Identification and Resource Management," *Johns Hopkins APL Technical Digest*, Vol. 36, No. 2, 2022. Online: <https://api.semanticscholar.org/CorpusID:251342401>.
- [134] Zhang, B., and W. Zhu, "DQN Based Decision-Making Method of Cognitive Jamming Against Multifunctional Radar," *Systems Engineering and Electronics*, Vol. 42, No. 4, 2020. Online: <https://www.sys-ele.com/EN/abstract/abstract7454.shtml>.
- [135] Pan, Z., et al., "Joint Optimization of Jamming Type Selection and Power Control for Countering Multifunction Radar Based on Deep Reinforcement Learning," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 59, No. 4, 2023. DOI: 10.1109/TAES.2023.3272307.
- [136] Rao, N., et al., "Fast Adaptive Jamming Resource Allocation Against Frequency-Hopping Spread Spectrum in Wireless Sensor Networks via Meta-Deep-Reinforcement-Learning," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 60, No. 6, 2024. DOI: 10.1109/TAES.2024.3418944.

- [137] Kang, L., et al., "Reinforcement Learning Based Anti-jamming Frequency Hopping Strategies Design for Cognitive Radar," in *Signal Processing, Communications and Computing*, 2018. doi: 10.1109/ICSPCC.2018.8567751.
- [138] Bi, Y., Y. Wu, and C. Hua, "Deep Reinforcement Learning Based Multi-User Anti-Jamming Strategy," in *International Conference on Communications*, 2019. doi: 10.1109/ICC.2019.8761848.
- [139] Ak, S., and S. Brüggewirth, *Avoiding Jammers: A Reinforcement Learning Approach*, 2019. Online: <https://arxiv.org/abs/1911.08874>.
- [140] Huynh, N., et al., "'Jam Me If You Can': Defeating Jammer with Deep Dueling Neural Network Architecture and Ambient Backscattering Augmented Communications," *IEEE Journal on Selected Areas in Communications*, 2019. doi: 10.1109/JSAC.2019.2933889.
- [141] Abuzainab, N., et al., "QoS and Jamming-Aware Wireless Networking Using Deep Reinforcement Learning," in *MILCOM*, 2019. Online: <https://arxiv.org/abs/1910.05766>.
- [142] Erpek, T., Y. Sagduyu, and Y. Shi, "Deep Learning for Launching and Mitigating Wireless Jamming Attacks," *IEEE Transactions on Cognitive Communications and Networking*, Vol. 45, No. 1, 2018. Online: <https://arxiv.org/abs/1807.02567>.
- [143] Li, Y., et al., "On the Performance of Deep Reinforcement Learning-Based Anti-Jamming Method Confronting Intelligent Jammer," *Applied Sciences*, Vol. 9, 2019. doi: 10.3390/app9071361.
- [144] Cheng, S., X. Ling, and L. Zhu, "Deep Reinforcement Learning-Based Anti-Jamming Approach for Fast Frequency Hopping Systems," *IEEE Open Journal of the Communications Society*, Vol. 6, 2025. doi: 10.1109/OJCOMS.2025.3529982.
- [145] Baietto, A., et al., "Lean Neural Networks for Autonomous Radar Waveform Design," *Sensors*, Vol. 22, No. 4, 2022. doi: 10.3390/s22041317.
- [146] Zheng, S., et al., "Radar-Jamming Decision-Making Based on Improved Q-Learning and FPGA Hardware Implementation," *Remote Sensing*, Vol. 16, No. 7, 2024. doi: 10.3390/rs16071190.
- [147] Carmack, J., S. Schmidt, and S. Kuzdeba, "Multi-Agent Reinforcement Learning Approaches to RF Fingerprint Enhancement," in *ACM Workshop on Wireless Security and Machine Learning*, 2021. doi: 10.1145/3468218.3469037.
- [148] Zhou, C., et al., "Frequency Diversity Array Radar and Jammer Intelligent Frequency Domain Power Countermeasures Based on Multi-Agent Reinforcement Learning," *MDPI Remote Sensing*, Vol. 16, No. 12, 2023. Online: <https://www.mdpi.com/2072-4292/16/12/2127>.
- [149] Ran, W., et al., "Research on Efficient Multiagent Reinforcement Learning for Multiple UAVs' Distributed Jamming Strategy," *MDPI Electronics*, Vol. 12, No. 18, 2023. doi: 10.3390/electronics12183874.
- [150] Zhou, Q., Y. Li, and Y. Niu, "Intelligent Anti-Jamming Communication for Wireless Sensor Networks: A Multi-Agent Reinforcement Learning Approach," *IEEE Open Journal of the Communications Society*, Vol. 2, 2021. doi: 10.1109/OJCOMS.2021.3056113.
- [151] Jiang, W., Y. Ren, and Y. Wang, "Improving Anti-Jamming Decision-Making Strategies for Cognitive Radar via Multi-Agent Deep Reinforcement Learning," *Digital Signal Processing*, Vol. 135, No. 1, 2023. doi: 10.1016/j.dsp.2023.103952.
- [152] Peng, W., et al., "Enhanced Radar Anti-Jamming with Multi-Agent Reinforcement Learning," *Signal Processing Letters*, 2024. doi: 10.1109/LSP.2024.3493797.

- [153] Vlassis, N., et al., "Bayesian Reinforcement Learning," in *Reinforcement Learning. Adaptation, Learning, and Optimization*, Vol. 12, New York, NY: Springer, 2012. DOI: 10.1007/978-3-642-27645-3_11.
- [154] Ghavamzadeh, M., et al., "Bayesian Reinforcement Learning: A Survey," *Foundations and Trends in Machine Learning*, Vol. 8, No. 5-6, 2015. Online: <https://arxiv.org/abs/1609.04436>.
- [155] Zhang, R., "A Cross-Layer Optimization Technique Based on Bayesian Reinforcement Learning Extended POMDP Model for Dynamic Spectrum Access," in *Communication Technology*, 2022. DOI: 10.1109/ICCT56141.2022.10073446.
- [156] Jonnabhatla, P., and T. Vimala, "Bayesian Reinforcement Learning for Beam Forming in Millimetre-Wave Networks for Multi-Target Detection in 5G Radio Resource Allocation," *SSRG International Journal of Electronics and Communication Engineering*, Vol. 12, No. 1, 2025. DOI: 10.14445/23488549/IJECE-V12I1P114.
- [157] Elbattrawy, A. N., et al., "Model-based Bayesian Reinforcement Learning for Enhancing Primary User Performance Under Jamming Attack," *Ad Hoc Networks*, Vol. 148, 2023. DOI: 10.1016/j.adhoc.2023.103206.
- [158] Jiménez, S., et al., "A Review of Machine Learning for Automated Planning," *The Knowledge Engineering Review*, Vol. 27, 2012. DOI: 10.1017/S026988891200001X.
- [159] Kotseruba, I., and J. Tsotsos, "A Review of 40 Years of Cognitive Architecture Research: Focus on Perception, Attention, Learning and Applications," *Computing Research Repository*, 2016. Online: <http://arxiv.org/abs/1610.08602>.
- [160] Nason, S., and J. Laird, "Soar-RL: Integrating Reinforcement Learning with SOAR," *Cognitive Systems Research*, Vol. 6, 2005. DOI: 10.1016/j.cogsys.2004.09.006.
- [161] de la Rosa, T., and S. McIlraith, "Learning Domain Control Knowledge for TLPlan and Beyond," in *ICAPS Workshop on Planning and Learning*, 2011. Online: <http://www.cs.toronto.edu/~sheila/publications/ros-mci-pal11.pdf>.
- [162] Minton, S., et al., "Acquiring Effective Search Control Rules: Explanation-Based Learning in the PRODIGY System," in *International Workshop on Machine Learning*, 1987. DOI: 10.1016/B978-0-934613-41-5.50016-7.
- [163] Ye, P., T. Wang, and F.-Y. Wang, "A Survey of Cognitive Architectures in the Past 20 Years," *IEEE Transactions on Cybernetics*, Vol. 48, No. 12, 2018. DOI: 10.1109/TCYB.2018.2857704.
- [164] Haykin, S., *Cognitive Dynamic Systems: Perception-action Cycle, Radar, and Radio*. Cambridge, UK: Cambridge University Press, 2012.
- [165] Ustun, V., et al., *Controlling Synthetic Characters in Simulations: A Case for Cognitive Architectures and Sigma*, 2021. Online: <https://arxiv.org/abs/2101.02231>. See also <https://cogarch.ict.usc.edu/>.
- [166] *OpenCog Hyperon: Realizing the AGI Dream*. Online: <https://hyperon.opencog.org/>. Accessed 2025-02-25.
- [167] Tromp, R., "Towards Motivated Agents: An Implemented Blueprint of a Computational Motivation Theory in Minecraft," M.S. thesis, Utrecht University, Utrecht, Netherlands, 2019. Online: <https://studenttheses.uu.nl/handle/20.500.12932/34258>.
- [168] Kugele, S., and S. Franklin, "Learning in LIDA," *Cognitive Systems Research*, Vol. 66, 2021. DOI: 10.1016/j.cogsys.2020.11.001.
- [169] Samsonovich, A. V., "Schema Formalism for the Common Model of Cognition," *Biologically Inspired Cognitive Architectures*, Vol. 26, 2018. DOI: 10.1016/j.bica.2018.10.008.

- [170] Schuette, J., et al., "Performance of RF Mapping using Opportunistic Distributed Devices," in *IEEE Military Communications Conference*, 2015. DOI: 10.1109/MILCOM.2015.7357677.
- [171] Guerci, J., *Cognitive Radar: The Knowledge-Aided Fully Adaptive Approach*, 2nd Edition. Norwood, MA: Artech House, 2020.

8

Data Management

Data collection and management is perhaps the hardest part of building a system based on AI and ML. A rule of thumb for all ML-enabled systems is that 80% of the work goes into collecting and curating the data, despite the fact that they are only two of the steps in the data lifecycle (Figure 8.1). Chapter 4 discussed the steps related to creating deductions and inferences from the data; this chapter describes how to create high-quality data from the beginning, and how to maintain data quality over time.

One of the biggest challenges in EW is data quality: Data is improperly curated, “dropped” because the legacy system doesn’t record it, and even gets deliberately destroyed. Data is frequently either not available, not annotated with the right metadata, or not trustworthy. Before creating (or using) a dataset, consider the following questions:

- What is the problem? What is the decision to be informed? What is the task: understand? predict? control? adapt continuously? If you know the question, you can be much more selective about what you need.

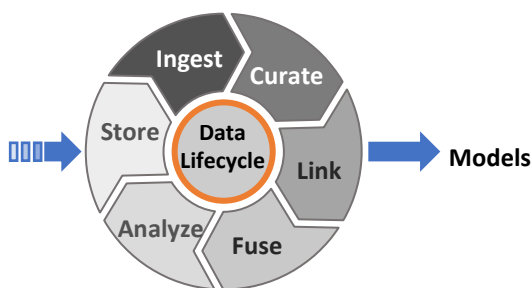


Figure 8.1 The data lifecycle involves collecting and processing the data; iterations increase its utility for multiple purposes.

Callout 8.1 Establish clear semantics and liberally annotate everything when performing an experiment.

Annotate all data with *metadata* that details how it was collected, including location, type of experiment, and participants.

- Ensure that synthetic and real data have the same structure.
- Version control *everything*: models, software, hardware, mission data, and scenario configurations.

Establish clear semantics to support interoperability across missions, across platforms, and over time.

- Identify all assumptions, both explicit and (if possible) implicit. If the data is synthetic, note all simplifying assumptions.
- Identify unusual cases, such as rare “edge” cases, or adversarial examples purposely constructed to exploit weaknesses.

- What data is available? Is it sufficient? Is it diverse? If not, can more be collected? What formats does it use? If you build the model, it may help identify where the data is lacking.
- How has the data been curated? Is it inherently biased? Has it been manipulated by an adversary? If you know where the data came from, you know where and how to use it.

System requirements rarely capture data needs, such as what data to record, how to tag it, or what properties it must have (e.g., diversity).

This chapter presents ideas for *data quality control* (how to curate EW data), *data modeling* (the semantics and tools for encoding the data), and *data management* (how to use it inside a system to obtain the desired results).

8.1 Data Quality Control

The data “supply chain” is a crucial driver of the quality of an AI/ML system. This supply chain includes the provenance and credibility of the data, its completeness, and its accessibility. If the data (or its metadata) is low quality, the ML will make poor inferences and, more importantly, be unaware of its own deficiencies. Callout 8.1 outlines the critical actions to take when performing an experiment and collecting data. The Data Safety Initiative Working Group published a *Data Safety Guidance* document that describes a full process for ensuring high quality data [1]. The steps cover data use and management through the entire lifecycle, establishing context and then identifying, analyzing and treating risks.

Table 8.1
Key Goals of the DoD's *Data Strategy* to Manage Data as a Strategic Asset [2]

Goal	Required Capabilities for Measurable Progress
Visible	Advertised; available; metadata standards; catalogued; common services to publish, search and discover data; governments make decision based on live visualizations
Accessible	Standard APIs; common platforms create and use data; data access is protected
Understandable	Preserves semantics; common syntax; elements aligned; catalogued
Linked	Discoverable and linked; metadata standards
Trustworthy	Protection; lineage; pedigree metadata; data quality
Interoperable	Exchange specifications; metadata and semantics; machine readable; translation across formats does not lose fidelity; data tagging
Secure	Granular privileges; approved standards; clear markings; authorized users

Table 8.1 summarizes the DoD's perspective for managing data as an asset. The goals support the purpose of data access, availability, and stewardship. The DoD's perspectives are in line with the FAIR guiding principles for data stewardship: Findability, Accessibility, Interoperability, and Reusability [3].

Data curation is the process of ensuring high-quality data. *Bias* is a key characteristic to manage in every dataset. *Traceability* ensures data integrity, tracking where the data came from and how credible it is. Together, these concepts enable accurate inference and decision making, experiment reproducibility, and sharing data across platforms and over time. Moreover, these concepts are key requirements when moving the data through security- or safety-assurance levels: They enable decisions to be made about whether it is appropriate to manipulate the data, whether to grant access to users, or whether the system meets accreditation requirements.

From an engineering perspective, modules must provide information that can be directly used by other modules, including data fusion, decision making, and the user interface. Figure 8.2 highlights the key concepts that support interoperability and cooperative processes.

8.1.1 Data Curation

Data curation is the activity of validating, organizing, and integrating data. Curation ensures high-quality data from which the system can draw high-quality conclusions.

Actionable	Additive	Auditable	Agnostic
Outputs are usable downstream	Outputs can be combined	Outputs have traceability	Data structure is platform-independent

Figure 8.2 To support interoperability, situation assessment and decision making modules should produce practical results that are actionable, additive, auditable, and agnostic. (Concepts originally introduced for probability management [4].)

Data quality is most commonly associated with a “fitness for use” principle, meaning that the quality of the data depends on the task for which it will be used.

Many attributes can be used to measure the quality of a dataset; Wang & Strong [5] identified 179. The most important of these are *accuracy*, *completeness*, *consistency*, *timeliness*, and *relevance*. The goals in the DoD Data Strategy cover many of these (Table 8.1), driving down to the level of measurable pragmatic implementation. The International Standards Organization (ISO) has a data quality model with 15 dimensions [6]; characteristics are organized by whether they are intrinsic to the data, or whether they are dependent on the system and usage (Figure 8.3). Wang et al. [7] identify 21 dimensions, and also discuss issues such as operation and assurance costs, legal responsibility, and user experience.

Wang & Strong [5] describe three approaches used to study data quality. The approach can be *intuitive*, based on the developers’ past experience or intuitive understanding of what dimensions are essential; it can be *theoretical*, studying data quality from the perspective of systems that consume and produce data; or it can be

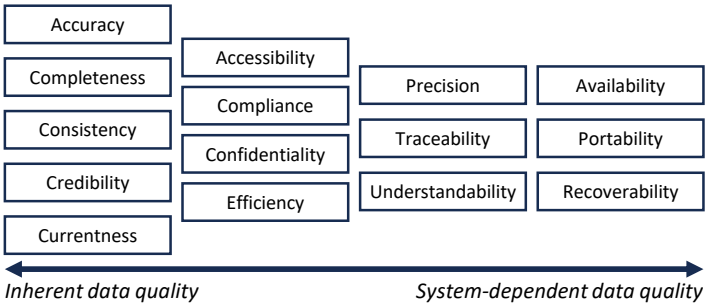


Figure 8.3 ISO defines 15 attributes to evaluate data quality [6].

empirical, asking consumers which characteristics they use to assess data for their tasks. Several tools exist to examine data quality against these attributes [8, 9].

Data curation is a crucial underpinning for CogEW systems, supporting a variety of tasks:

- *Accurate root-cause or ground-truth analysis*: Evaluating a CogEW system requires knowing what actually happened, for example knowing if a packet collision was intentional or accidental. Similarly, for an SEI task, recording the distance from transmitter to receiver enables the experimenter to determine that the model used latent features of the transmitter, and did not use received power as a surrogate for distance.
- *Experiment reproducibility*: Evaluating system accuracy requires the ability to rerun experiments, tweak experimental conditions, or evaluate ML models against each other. For example, the system can assess the relative benefit of separating signals by frequency or by cyclostationary processing.
- *Meta-analysis*: Determining data quality involves analyzing datasets for patterns such as distribution, completeness, and systemic bias. Constructing complex scenarios that combine features of (previously) independent datasets involves understanding where and how they interact. Meta-analysis also supports evaluating data integrity: ensuring that incoming or stored data is factual and complete.
- *Future-proofing*: Reducing data degradation (“bitrot”) requires capturing the details of the experiment so it is useful even as equipment or software changes.
- *Interoperability*: Reusing information across platforms or tools requires appropriate tagging of all collected, derived, or inferred knowledge.
- *Task reuse*: Knowing the purpose of a data collection allows developers to evaluate whether a dataset can be reused for a different purpose. For example, a dataset collected for the purpose of waveform classification is unlikely to be useful for SEI or identifying whether nodes are being jammed. The inverse, meanwhile, may not be true: Specific emitter data may very well be useful for waveform classification. Likewise, knowing which receiver recorded which data for SEI enables developing receiver-independent models.
- *Data sharing*: Sharing data with other organizations for research and operational purposes enables cooperative missions and fosters collaboration. Metadata helps determine whether and how to grant access to sensitive data.
- *Supply-chain evaluation*: Knowing the provenance and credibility of the data allows ES to make correct inferences, EP/EA/EBM to make correct

decisions, and system designers to determine whether to use the data at all. Manipulation can occur at any point in the supply chain, via hardware, software, in transit, or in storage.

Data curation involves managing and maintaining the data to ensure its usability, accessibility, and quality over time. The process involves the following steps:

- *Collect data.* Identify sources, gather data, and document provenance.
- *Assess data quality.* Check for data quality attributes and detect errors.
- *Clean data.* Correct errors, remove duplicates, and standardize the format.
- *Annotate data.* Create metadata and ensure compliance with standards.
- *Integrate data.* Merge data sources, resolve conflicts, and link related data.
- *Organize data.* Develop hierarchies and store in repositories.
- *Share data.* Document and publish formats, and ensure access control.
- *Maintain data.* Assess and update data quality, relevance, and accuracy; establish protocols to maintain consistency as technology evolves.

When treated as part of the data lifecycle (Figure 8.1), curation ensures that the data will continue to be a valuable resource for AI-enabled systems.

8.1.2 Bias and Noise

Bias is an unintended deviation from fairness, accuracy, or neutrality in the data, algorithms, or decisions produced by a system. It reflects *structural* and *systemic* characteristics of the system or environment in which the system operates, leading to inappropriate outcomes. Bias is a consequence of engineered factors during the design, development, and deployment of systems with AI. These biases are not random anomalies but are embedded within the processes that generate, process, and analyze data. The goal is therefore to eliminate bias and thus increase the quality of the data. Almost all datasets are biased, in that they do not accurately represent a model's use case, resulting in skewed outcomes, low accuracy levels, and analytical errors. For example, bias can lead to radar tracking errors, and multiple tracks generated for the same target. Common forms of data bias include:

- *Sample or selection bias:* When samples do not represent the underlying distribution of the data. This problem occurs due to faulty experiment design (missing key examples), or environmental factors that cause samples to be missed. For example, signal propagation over land is very different than over seawater due to ducting and surface-wave action; experiments and data collection must compensate for these factors.

Feedback loops (such as those created by RL) can exacerbate selection bias. For example, each favorable use of an ECM technique can reinforce the desire to use the same (or similar) technique, stepping along the gradient towards a local optimum. Without being deliberate about active learning and maintaining diversity, the system can become trapped in that local optimum and never discover the global optimum.

- *Class bias*: When data skews toward a particular set of classes. To the extent that the data covers all forms of RF signal (e.g., frequencies, modulations, protocols), the more likely that a learned model will capture all the latent features of RF, and not just the features within the overly narrow dataset. (Specialized models for a given type of RF signal are appropriate and should be tagged as such.)
- *Exclusion bias*: Post-collection exclusion of examples from the data, usually during data preprocessing. For example, if a system only records data of “interesting” emitters or those above/below a specific threshold for SNR ratio, critical examples are lost. A common structure in EW systems is that the legacy framework will act on the things it knows, and only forward the data it doesn’t know how to handle; this model becomes a significant challenge because the cognitive structures only receive the “scraps,” and don’t have sufficient history or context to make accurate conclusions.
- *Measurement bias*: Miscalibration, sensor bias or misregistration, faulty measurements, mislabeled examples, missing variables, or a difference between the data collected for training versus inference. Spatial misregistration, for example, will lead to false radar tracks. Resampling in digital signal processing can cause significant measurement bias. If the original data is downsampled to meet computation or memory requirements, then it may lose important features. Likewise, if decimating the data violates the Nyquist criteria, aliasing can create distortions that cannot be corrected downstream.
- *Recall bias*: Similar examples receive different labels, or the meaning of labels may change. In a dynamic system that evolves over time, this is a common problem: The concept may drift (the data changes), and the domain may change (e.g., when a component or algorithm changes). For example, signals separated by cyclostationary processing will have different characteristics than signals separated by frequency. Maintaining the metadata ensures that these meanings can be correctly interpreted.

- *Observer or confirmation bias*: Labels based on expected results, not actual results. A common cause of this problem is when the system fails to recognize novel examples (e.g., a new modulation), and classifies them as the most similar previously known example. Single source errors likewise create a risk that the data encodes artifacts of that particular source, such as when the data was collected by a single receiver.
- *Survivorship bias*: Only successful cases are included in the dataset, omitting failures. For example, models trained on only high-SNR detected threats may fail to recognize stealthy low-SNR signals.
- *Historical or temporal bias*: Data reflects past systemic issues or environmental changes. It can manifest as favoring recent data or giving too much weight to past data. For example, models trained on older systems may fail to detect newer advanced technologies.

Systemic bias is different from *noise*, which is the natural random variability of the domain. It is easier to measure and correct for noise than it is to measure and correct for bias. While it may be tempting to remove noise from the stored data, the particular noise-elimination technique may in itself be biased. In general, the dataset should contain the noise to reduce overfitting; data augmentation will, in fact, deliberately add noise (Section 8.3.3). The general expectation is that the distribution of the noise in the attributes of the training data will match the distribution of noise during inference [10, 11], and this hypothesis should be verified [12].

However, noise is also a leading error of classification errors because the model learns to fit the noise, and not the underlying concept. Regularization, diversity, and cross-validation are good techniques to ensure that the noise improves the quality of the model.

8.1.3 Traceability

Traceability is a necessary component for determining data integrity, both from a security perspective and a learning/inference perspective. The two key points are:

- The system must ensure that the data is not corrupted at ingest, during analysis, in transit, or in storage.
- The system must track the provenance and credibility of the data as it moves through hardware, firmware, software, communications, and humans.

The *provenance* of the data records where the data came from, while its *credibility* tracks how reliable the source is (i.e., that the data represents what it

is supposed to represent). While provenance and credibility are often considered a cybersecurity problem, the same issues apply to poorly collected data. In fact, if the system can determine that the data was deliberately manipulated, it can compensate and infer more accurately than when the data collection was sloppy or incomplete.

Never attribute to malice that which is
adequately explained by incompetence.

— Robert Hanlon, 1980, *Hanlon's Razor*

Good data collection and curation is the first step in creating credible data; each transformation of the data also transforms its credibility. For example, data fusion can fuse information from multiple low-credibility sources to create a relatively high-credibility conclusion. Another example would be an improved radar emitter signals recognition and classification via amalgamation of classifiers' predictions and drone surveillance data; surveillance can provide visual target confirmation and its location, further increasing the credibility of the predictions.

Traceability ensures that ES data fusion forms the correct conclusions (Section 4.4), and that decision-making algorithms manage information uncertainty correctly (Section 6.1.4). Traceability also enables anomalies and errors to be traced to their origin, components replaced, and better results achieved.

8.2 Data Modeling: Ontologies, Metadata, and Schemas

Well-established data semantics is a crucial piece of supporting tasks such as interoperability, task reuse, and data sharing. Good semantics allows data to be reused across systems and over time. The dataset can grow, even as components (software, hardware, platforms, and missions) evolve. Three tools are used to capture concepts and data for a domain. *Ontologies* are dictionaries that capture the meaning of and relationships between concepts. *Metadata* provides contextual information about the data itself, with domain- and task-specific fieldnames. *Schemas* are the formats that capture the data; a schema ensures the data is structured correctly. Table 8.2 highlights the key aspects of ontologies, metadata, and schemas.

8.2.1 Ontologies

An *ontology* represents information that captures semantics. An ontology is a formal naming and definition of the terms and categories to describe the concepts, data and entities of a problem domain, and the relationships among them. A *domain* ontology focuses on problem domains, while a *process* ontology captures the steps and constraints involved in a processing chain. Ontologies for cognitive radio [13–15], radar [16], and EW [17] exist. Each has a slightly different focus, for example

Table 8.2
Ontologies, Metadata, and Schemas Serve Different Complementary Purposes

Aspect	Ontology	Metadata	Schema
Purpose	Models knowledge by defining entities, relationships, and rules	Describes information about the data	Specifies data formats
Focus	Semantic meaning and inference	Data context and meaning	Data organization and validation
Scope	Broad conceptual models across systems and domains	Specific systems and datasets	Data integrity
Application	Knowledge representation, semantic search, AI reasoning	Data sharing and contextual understanding	Data validation, storage, and formatting

the SDR forum’s ontology [13] focuses on adaptive modulation, while Cooklev and Stanchev’s [15] supports radio topology. The goal of the CogEW ontology [17] was to create a framework for collaborative information exchange and an ontological language that facilitates the integration of knowledge from multiple sensors and processes, enabling the operator to quickly gain situation awareness of the spectral environment. An example query is “*What emitters within 3 miles of the White House are active within the TV White Space band?*”

The Web Ontology Language (OWL) is commonly used to capture the ontology [18], and the Protégé editor is used to create and manipulate it [19]. The main challenge when developing an ontology is to develop it only for the necessary task. It is easy to over-engineer the ontology and develop concepts that will never be used. A formal ontology may be excessive for the task. The crucial point is to establish a clear semantics that cannot be misinterpreted.

A major advantage afforded by ontologies is that they allow for concise representations in which much of the knowledge is implicit rather than explicitly stated. For example, if an ontology specifies that the class transceiver is a subclass of the class receiver, and Alpha is identified as being a transceiver, then it is not necessary to explicitly state that Alpha is also a receiver because that fact is implicit in the ontology and the axioms that define its semantics. Formal logical reasoning deduces facts from the known status in a specific scenario to the generation of suggestions for actions and questions to be sent to other participants. For example, if a sensor detects and characterizes an emitter, the ontology can hypothesize which

types of emitters might be making the waveforms, and further which types of platforms. The ontology helps focus the search for emitters, the prioritization of threats, and the selection of appropriate actions. An ontology can also help evaluate the quality of the data and enhance training of ML models [20].

8.2.2 Metadata

Metadata provides the contextual information about the data. Metadata captures all the information about the experiment (and the dataset) that is not inherently part of the dataset. It is the ground truth context of who, what, where, why, and when the data was created. One of the barriers to developing CogEW systems is that data is poorly collected, poorly tagged, and/or proprietary. Good metadata can help eliminate this barrier. Along with the completeness, bias, and appropriateness of the data for new tasks, metadata supports evaluation of the CogEW system.

Metadata should annotate the dataset's *macrofeatures*, such as version, author, and collection date, along with the purpose and goals of the scenario. Synthetic and real data should have the same structure. These global features include:

- Mission type and sequencing;
- Traffic type and patterns;
- Location (e.g., indoor, outdoor, lab, anechoic chamber, GPS coordinates, topography);
- Conditions (e.g., weather, mobility, rural/urban);
- Experiment design (e.g., controlled, “in-the-wild”);
- Experiment scenario (e.g., number of emitters and their types/roles);
- Hardware and software components (e.g., versions, capabilities, measurement error);
- Generation/collection style (e.g., over-the-wire, over-the-air, synthetic and the software used to generate it);
- Interdependencies, both within and between datasets;
- Collection protocols (e.g., continuous/sporadic experiments, reset/persistent configuration); and
- Assumptions, both explicit and (if possible) implicit, particularly simplifying assumptions for synthetic data.

The more detailed the annotations, the more useful the data will be for multiple scenarios. A detail like *this cable connected part A to part B inside radio C* may seem superfluous or overengineered during an initial collection, but having this record allows future analysis and reuse. A detail such as whether data was collected in binary, integer, fixed-point or n -bit floating point allows others to immediately

triage whether the data applies to their situation. (Many public datasets are recorded in floating point, which is not compatible with FPGAs.)

Metadata must follow two key principles. First, *annotate everything*. All observations and conclusions made by human observers and software components, such as the purpose of sending and the inferences made. For example, data collected at the beginning of the day may have different artifacts than after continuous usage when the hardware is warm. It's particularly useful to identify unusual cases, such as rare "edge" cases and adversarial cases purposely constructed to exploit weaknesses. Second, *version control everything*: models, software and hardware configurations, customizations and mission scenarios. It is hard to over-emphasize this requirement.

A cognitive radar requires a means for *evaluating and ensuring the reliability* of its knowledge sources: both sources of past knowledge, provided through access of databases, as well as knowledge learned through operational experience. This includes considering not only the possibility for *deception*, but also whether the validity of data *degrades* over time.

— Gürbüz et al., 2019 [21]

Figure 8.4 illustrates a notional (complex) example based on air traffic control radar. A final air traffic control display pulls together information from many different sources, including surface movement radar, airport surveillance radar, multilateration sensors around the airport, automatic dependent surveillance - broadcast (ADS-B) sensors, and flight plan data. Each inferred flight track is based on a complex web of connected components. Multiple platforms collect different types of raw data, make intermediate inferences, and communicate them to other platforms. Different data formats, communications media, and communications protocols move the inferences together, leading to final track inferences that support warning systems and air traffic control. *Each piece* of this complex system—hardware, firmware, software, communications, displays—must be certified to meet aviation safety standards because each piece impacts the accuracy of the final inferences and the degree to which stakeholders will trust the system. The degree to which the metadata is recorded impacts how easily the system can analyze the data and identify faults.

Northeastern University's (NEU) RF fingerprinting project provides a concrete example of metadata for a comprehensive experiment [22]. To evaluate RF fingerprinting and ensure repeatability, NEU records detailed features of every experiment. The metadata captures a description of the original dataset, the preprocessing steps, the model architecture, and methods for training and testing the data,

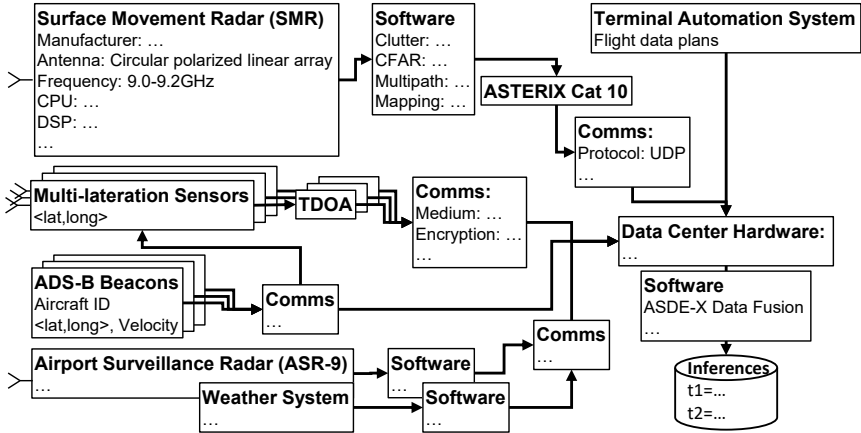


Figure 8.4 A complex web of components lead to high quality tracks and air traffic safety.

Table 8.3

NEU Records Detailed Features of Every Experiment to Ensure Repeatability [22]

Preprocessing	Model	Learning Task
Band Filtering	Full ANN architecture (e.g., layers, activation functions, batch normalization, dropout) Optimizer (type, metrics, loss) Training platform(s) (CPU, GPUs) Trained weights	Evaluation scenario
Partial Equalization		Training process (e.g., batches, epochs, normalization, stopping)
Slicing		Training population (e.g., devices, # examples)
		Validation population
		Test population
		Results (e.g., accuracy, timing)

as summarized in Table 8.3. Each component has associated parameters; for example *slicing* chooses random examples from a configurable fraction of data. From this structure, a given experiment can evaluate the impact on results of channel effects, number of I/Q samples per burst, and signal onset data. With a tag indicating that the training set contains transmitters $1, \dots, k$ and the test set contains transmitters $k + 1, \dots, n$, we can evaluate the impact of diversity on the ability to learn novel emitters [23].

The raw data itself should ideally record the details of each transmitter and each receiver, including information such as time-synchronous transmissions. These

details enable analysis of the impact of mobility, transmitter vs receiver chains, and simultaneous signals.

Time is one of the most important features to record faithfully.

When time is not recorded accurately, it becomes very challenging for the AI to recover information. For example, a summary of a pulse burst may have inconsistent time-of-arrival compared to the time-of-arrival of the constituent pulses.

Hardware layouts, and software versions, should likewise be detailed. A major gap to developing AI systems that understand their own limitations is that key information is lost along the processing chain. For example, while the system integrators know the angle and range measurement error for the radar system, this information needs to be available to the AI. If the metadata captures the uncertainty in raw measurements, software modules (especially data fusion) can estimate the error and confidence of the final inferences.

Figure 8.5 depicts a rudimentary, singular universal software radio peripheral (USRP) radio setup to demonstrate the locations for the USRP hardware driver (UHD) and FPGA-based applications. The metadata should record all of the details of the system. On the server side, there are several options to talk to UHD; raw C/C++ encoding has the best performance but requiring one to write all signal processing blocks from scratch. GNU radio packages (C/C++ or Python) are quicker to adopt, but may not have the same efficiency. On the USRP side, the RF network on chip (RFNoC) utility can be used to create FPGA applications to abstract some of the FPGA-based processing. The UHD software application programming interface (API) supports application development on all USRP software-defined radio SDR products [24]. Using a common software API improves code portability, allowing applications to transition seamlessly to other USRP SDR platforms when necessary. It reduces development efforts by allowing the users to preserve and reuse their legacy code so that they can focus on the development of new algorithms.

As an example use case, today we can search for data online and read a high-level description. If it seems to match our task needs, we download the metadata and inspect it. If the metadata is complete, then we can quickly select or eliminate the dataset from consideration. Assume researcher Alice is developing an SEI tool, and researcher Bob is developing a modulation recognition tool, and they wish to exchange their datasets. The key metadata attribute that Bob needs to know about Alice's data is whether signals are modulated, and how many modulations are used. Alice needs to know how many emitters are in Bob's modulation recognition dataset. Note that these emitters generally must be real, physical hardware: Multiple models that are synthetically generated might be considered multiple emitters,

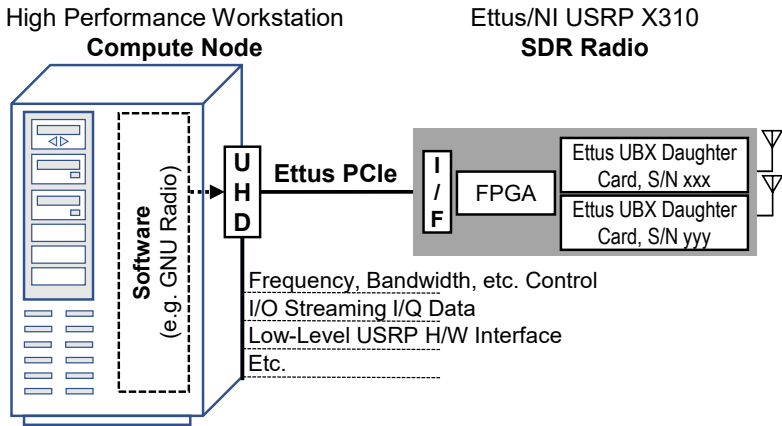


Figure 8.5 The metadata should contain details of the hardware, firmware, and software in this USRP structure, including versions and serial numbers. (See also Figure 11.2.)

but one signal generator that uses multiple models is *one* emitter. Knowing this information early reduces effort for data sharing, improves task reuse, and future-proofs the data.

NATO has developed two metadata formats for EMS information sharing and command and control [25]: *cooperative electronic support measure operations (CESMO)* [26] and *common electronic order of battle (C-EOB)* [27]. C-EOB is more strategic, describing known emitters for databases. CESMO shares real-time information about emitter observations and thus more tactical. These standards cover core elements of the EW domain, but do not yet comprehensively cover the full richness of the CogEW setting.

The *signal metadata format (SigMF)* [28, 29] is an open standard schema adopted by the GNU Radio community. It captures general information about the collection of samples (e.g., purpose, author, date), the characteristics of the system that generated the samples (e.g., hardware, software, scenario), and features of each signal (e.g., center frequency, timestamp). A variety of extensions to the base format exist. For example, the *antenna* format captures information such as model, frequency range, and gain patterns, while the *WiFi* extension adds information for frames and packets. An associated Python library `sigmf` makes it easy to use. Annotations allow comments, such as whether it was deliberately injected, or how it was classified by an ML model. Algorithm 8.1 contains a small example. The

Algorithm 8.1 SigMF captures information about the collection, the characteristics of the system, and features of each signal. (Example generated by code of README in [29].)

```
{
  "global": {
    "core:author": "jane.doe@domain.org",
    "core:datatype": "cf32_le",
    "core:description": "All zero example file.",
    "core:sample_rate": 48000,
    "core:sha512": "18790c279e0ca614c2b57a215fecc23...",
    "core:version": "0.0.2"
  },
  "captures": [
    {
      "core:datetime": "2020-08-19T23:58:58.610246Z",
      "core:frequency": 915000000,
      "core:sample_start": 0
    }
  ],
  "annotations": [
    {
      "core:comment": "example annotation",
      "core:freq_lower_edge": 914995000.0,
      "core:freq_upper_edge": 915005000.0,
      "core:sample_count": 200,
      "core:sample_start": 100
    }
  ]
}
```

GENESYS lab at NEU provides two SigMF-compatible datasets for RF fingerprinting of USRP X310 Radios [30].

VITA-49 is a complementary standard used for sending RF samples over a data transport [31, 32]; it is designed for data-in-motion, not long-term storage. Packet types include context, signal data, and control. *VITA-49* captures information such as waveform type, modulation type, location of emitters, and kinematics of the platforms.

8.2.3 Schemas

In contrast to ontologies and metadata, a *schema* is a structured framework that defines the formats for data elements and their permissible values. It ensures consistency and interoperability in data storage and usage.

Some schemas, such as XML and JSON, embed the structure in the message. JSON is more human-readable, has smaller filesizes, and is faster to parse, while XML can capture more complex information. While convenient for data transfer across systems that don't share the structure, both JSON and XML are heavyweight and inefficient encodings for many EW applications.

Schemas that place the structure external to the data—and therefore are more efficient encodings—include Protobuf, Cap'n Proto, and FlatBuffers. Google's *protocol buffers* (Protobuf) serialize and share structured data [33]. Protobuf's compactness emphasizes simplicity and performance (particularly for network communication), and maintains backward-compatibility. *Cap'n Proto* is a fast interchange format that uses in-memory data representation for serialization, thus avoiding parsing and encoding [34]. Developed by the original creator of Protobuf, it is much faster, and being used for high-performance communication in latency-critical systems. *FlatBuffers* by Google is a fast interchange that allows zero-copy deserialization [35]. It is optimized for high-performance real-time systems like games and mobile applications.

8.3 Data Management Practice

Data management, both in-mission and postmission, is the foundation to achieving the desired results in practice. Data management requirements are different between algorithm development and in-mission operation. During development, databases have relatively few space and timing constraints, and therefore should be comprehensive and thoroughly annotated.

In an *embedded* system, however, space and timing constraints must be considered. *Data diversity* ensures efficient memory use and rapid computation while achieving desired model quality. *Augmenting data* is a good way to ensure data diversity, and is particularly effective at handling adversarial settings. *Forgetting* irrelevant data helps ensure model relevance to the current situation. Finally, *data security* ensures that the data remains private.

8.3.1 Data in an Embedded System

Data management in an embedded system has very different requirements compared to huge data compute systems in a server farm. Moreover, many EW systems

operate in the tactical edge, where communications is limited or nonexistent, so storage and computation must be strictly local.

Controlling the size of the dataset solves both problems: memory usage and computation time. Controlling the size of the dataset ensures meeting memory constraints and improves computation time: Smaller datasets are less likely to trigger cache-miss penalties, and it is faster to train the models with smaller datasets. Approaches include:

- *Fewer instances.* Save only important data, maintaining the dataset's diversity with more relevant examples.
- *Fewer features.* Save fewer characteristics for each instance, capturing key behaviors.
- *Abstraction.* Simplify the data by removing details or precision, for example, by storing the channel and not the frequency.
- *Compression.* Simplify the data through compression or data distillation; when there is a known downstream task compression can be evaluated by impact to the task.

The BBN SO [36] (Example 1.1) uses a fixed-length circular buffer, maintaining the diversity of instances in memory and forgetting old instances. It records all examples in persistent storage for use in later missions, but these are not accessed in real time. The BBN SO stores k instances, each having $\overline{o_n} + \overline{c_n} + \overline{m_n}$ features (each feature is an 8-bit integer). It computes the dot products of the pairwise distances between instances, and stores the triangular similarity matrix, size $\frac{1}{2}k^2$. To improve efficiency for both memory and compute, the triangle matrix is a fixed-size, one-dimensional array that uses index arithmetic [37]; the index of $x[i; j]$ in the one-dimensional array is $\frac{1}{2}i(i + 1) + j$. This similarity matrix then becomes a basis for data diversity, model computation, and managing experiments for active learning. Matrix sizes are controlled and known a priori, and can be adjusted to suit the available memory on the platform.

Utilize long-term storage as available, running in a low-priority thread. This data is write-only, not accessed in real-time. It is useful for post-mission analysis.

8.3.2 Data Diversity

In embedded systems with limited memory, it is crucial to use the available memory as efficiently as possible. One of the keys to producing a model with good generalizability is to ensure data *diversity*. Diversity of the training data ensures that the data contains enough discriminative information to identify the important features (i.e., to maximize the amount of information contained in the data). The

data should include edge cases and adversarial examples. The goal is to have the model build high quality abstract representations. The better the generalization, the more likely that the model will perform well against novel environments. Moreover, diversity helps protect against adversarial attacks, as demonstrated by the data augmentation literature (Section 8.3.3).

The dataset does not need to be large, but it must be diverse.

Three examples of three different strategies in three different RF environments (27 examples) is far more useful than a million examples of one strategy in one RF environment, even though each observation of that environment will be slightly different. Figure 8.6 illustrates the 27 points; despite so few examples, it is clear that these examples cover the performance surface better than many examples for a single strategy or a single environment. By maintaining a diverse dataset, the learned models can achieve good performance even with limited examples.

A simple and effective method for computing diversity is simply the pairwise dot products between examples. When a new observation arrives, compute its distance to examples already in memory. If that distance is zero, then it is completely redundant with existing knowledge. Note that the dot product may compute *observables* only, *observables and strategy*, or *observables, strategy, and metrics*, depending on intended use. For example, computing the RF environments (Section 2.1.1) uses only the observables, while managing a diverse training dataset requires also using the strategy and metrics.

Mirzasoleiman et al. [38] develop a rigorous method for choosing a small subset S of the full training data V . The *coreset* of data is a weighted subset S that best approximates the full gradient of V , and can be efficiently found using a fast greedy algorithm. At every step, the greedy algorithm selects the element that reduces the upper bound on the estimation error the most. The size of the smallest subset S that estimates the full gradient by an error of at most ϵ depends on the structural properties of the data. The algorithm obtains significant speedup in training time while maintaining generalization accuracy.

In fact, diversity is so important that *simulated* diversity yields better results than a nondiverse real-world dataset. Essentially, while the simulated values do not capture the complexity or natural noise of the real environment, it increases the breadth of knowledge to draw inferences from. Real-world datasets are naturally imbalanced, and hidden war reserve modes exacerbate this imbalance. Simulated data can help “bootstrap” the learning process. Section 11.5.2 describes several RF data-generation tools. Work demonstrating the impact of diversity includes:

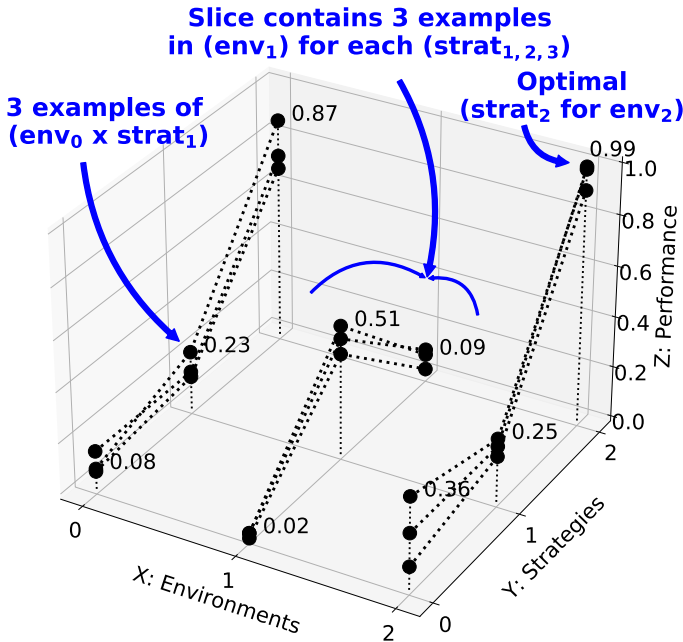


Figure 8.6 Three observations of each of three environments and three strategies yields 27 diverse training points (artificial data). An *environment* corresponds to a cluster of observable vectors o , and a *strategy* corresponds to a setting of controllables c . (Figure 4.5 shows several 2D environment “slices” from real environments, showing some environments with no observations where the model generalized from similar characteristics.)

- *Ablation trials* show that training the model using the most diverse dataset yields the best generalization to new environments (Section 10.9).
- *Pseudo classes* cause classes to repulse from each other so that the learned model chooses more discriminative features [39].
- *Diversified minibatch stochastic gradient descent* builds similarity matrices to suppress the co-occurrence of similar data points in the same training minibatch [40]. This approach generates more interpretable and diverse features for unsupervised problems, and in better classification accuracies for supervised problems.
- *Convex transductive experimental design* incorporates a diversity regularizer to ensure that selected samples are diverse [41, 42]. The approach excludes

highly similar examples with redundant information, consistently outperforming other active learning methods.

- *Informativeness analysis* uses pairwise gradient length as a measure of informativeness, and a *diversity analysis* forces a constraint on the proposed diverse gradient angle [41]. Empirical evaluations demonstrate the effectiveness and efficiency of the approach.
- *Entropy measurement and management* controls out-of-sample error; solutions with higher entropy are have lower error and handle novel situations better [43].

Diversity and uncertainty are effective methods to choose experiments in active learning (Section 7.3.2) [44, 45].

8.3.3 Data Augmentation

One of the most effective ways of creating data diversity is through data augmentation. Data augmentation enables ML practitioners to increase the diversity of the training data, without actually having to collect new data. Augmentation techniques include adding (to the existing dataset) slightly modified replicas of already existing data, or synthesizing new data. Data augmentation and adversarial training approaches can often compensate for the lack of data [46–49].

Data augmentation is closely related to the concept of *oversampling* in data analysis and can reduce *overfitting* during the training phase of the learning model [47]. For example, in image processing, some of the commonly used data augmentation techniques are padding, cropping, flipping, and changing lighting schemes. Data augmentation can be *domain-independent*, by relying on data-modification techniques such as noise, jitter, and rotation, or it can be *domain-dependent*, using techniques like channel effects and carrier frequency offsets.

Data augmentation techniques capture *data invariance*, the property of remaining unchanged regardless of changes in the conditions of measurement. For example, the surface area of an airplane remains unchanged if the airplane changes its pitch, yaw, and/or roll; thus the surface area of an airplane exhibits rotational invariance. Data augmentation ensures that the model is trained against this data invariance. Chung et al. [50] detect sample, class, and recall bias (Section 8.1.2) using correlation analysis, dynamic bucketization, unknown count estimation, and unknown value estimation. They use these results to augment the data.

A common theme to many data augmentation approaches is *adding noise* to the dataset. There can be noise in the *observables* and in the *metrics*. Model accuracy improves by injecting structured noise into the features [51–56]. Gaussian noise is the most common type of noise to add. Noise has a regularization effect

Table 8.4
RF Domain-Dependent Augmentations (Derived from [59])

Category	Examples
Noise	Channel, SNR, sample rate, background, cross-talk
Preprocessing	Filtering, dynamic range, bandwidth, decimation
Geometry	Reflections, changes in laydown, antenna orientation
Representation	I/Q imbalance, polarization change
Protocol	Modulation change, control sequence alteration
Variance	Control sequence delays, carrier changes, interleaving
Signal degradation	Subchannel nulling, clipping, path loss
Signal distortion	Fading, channel equalization, nonlinearities, intermodulation distortion
Deletions	Receiver failures, collisions, transmission dropouts

which improves the robustness of the model. Adding noise means that the model is less able to memorize training samples, resulting in smaller model weights and a more robust model that has lower generalization error (Section 3.2). Consider the following guidelines:

- Normalize the data before adding noise.
- Add noise only during the training phase.
- Noise can be added to the input data features.
- Noise can be added to outputs before training, especially in regression tasks over continuous target values. However, in the cases where it is possible to remove noise from outputs, better results can be obtained [57].
- Noise can be added to the activations, weights, and/or gradients [52].

Data augmentation doesn’t require knowledge of RF spectrum. Huang et al. [58] use rotation, flip, and Gaussian noise to improve classification accuracy for radio modulation recognition. They successfully classify modulations using shorter radio samples, leading to a simplified deep learning model and shorter inference time. They also show that augmentation improves accuracy at lower SNR.

Adding domain-dependent augmentation can further improve results [59]. Table 8.4 outlines some RF-specific augmentation approaches. Soltani et al. [60] propose a data augmentation technique for the training phase that exposes their RF fingerprinting DeepNet to many simulated channel and noise variations not present in the original dataset. The authors propose two approaches for data augmentation, one for the *transmitter* data (using pure signals without data distortion) and the other for the *receiver* data (using passive, available over-the-air transmitted signals). The

authors saw a 75% improvement with transmitter data augmentation and 32-51% improvement for receiver augmentation.

Sheeny et al. [61] present a radar data augmentation technique based on the measured properties of the radar signal. Their model achieved only 39% accuracy with nonaugmented data, 82% accuracy with translations and image mirroring, and 99.8% accuracy using data augmented with radar properties of signal attenuation and resolution change over range, speckle noise, and background shift.

SRC [62] trained models for pulse detection and denoising by blending live data with synthetic data to increase the diversity of the training set. Because the synthetic data essentially added samples that were outside the distribution of the live data, the model trained on blended data yielded better performance than models trained on either synthetic or live data alone.

Augmentation is particularly useful in cases where the initial training data is very limited. Figure 8.7 shows the confusion matrix results of an RF fingerprinting model on I/Q data collected from automatic dependent surveillance-broadcast (ADS-B) data. Figure 8.7 (a), (b), and (c) show the results on unaugmented data for training sizes of 2312, 289, and 10 samples, respectively. Figure 8.7 (d), (e), and (f) create 4 domain-independent augmentations on each of 10 original samples, demonstrating that augmentation can dramatically impact performance.

When augmenting data, relevant questions include: Which augmentation method works best on this data? Do you get the same results the same every time? What is the impact of number of initial instances, number of augmentations, or number of epochs? What happens when you combine methods? What is the impact of domain-independent methods vs domain-dependent methods?

Data augmentation also helps protect against adversarial attacks. Well-constructed examples train a model to defend against *white-box* attacks (where the adversary knows everything about the model), *black-box* attacks (where the adversary knows nothing), and intermediate *grey-box* attacks [64, 65].

GANs [66] have become a de facto method for generating adversarial examples and training models. A key benefit of GANs is that they do not require any information about RF properties. Adversarial training techniques sets a *generative network* against a *discriminative network* in a competition: The generative network's goal is to fool the discriminative network. The generative network creates artificial data designed to exploit weaknesses in the discriminative model. One of the most famous GANs generates photos of nonexistent celebrities [67]; humans are unable to distinguish these "photos" from reality. GANs improve accuracy for

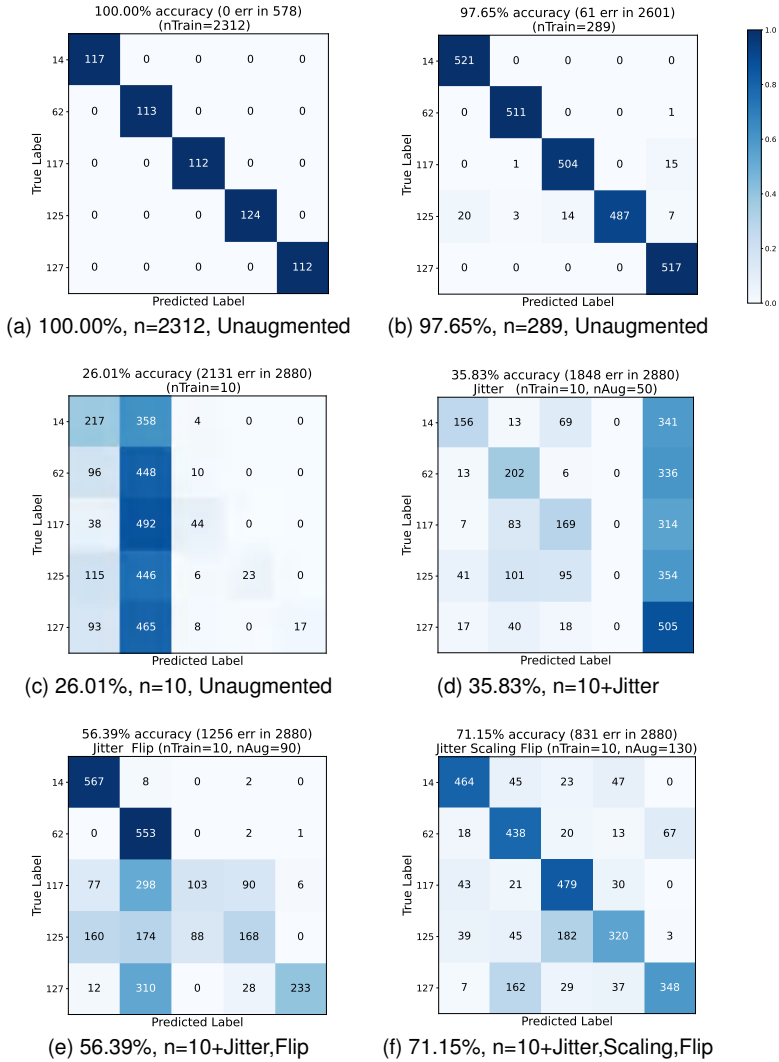


Figure 8.7 Domain-independent augmentation can dramatically improve prediction results on ADSB-IQ data. (Our results using data from [63].) *Top Row:* Unaugmented data; (a) train with 80% (n=2312) (b) train with 10% (n=289). *Middle Row:* n=10 samples of true data; (c) unaugmented, (d) augmented with 40 Jitter. *Bottom Row:* 10 samples of true data; (e) augmented with 40 each Jitter,Flip, (f) augmented with 40 each Jitter,Scaling,Flip.

classification tasks such as modulation recognition [68, 69] and radar target recognition [70]. The generative component has been used for signal spoofing and synthesizing new modulations [71, 72].

8.3.4 Forgetting Data

The counterpoint to data augmentation is forgetting. Reasons to deliberately drop data from a training set include limited memory, inaccurate information, and class imbalance. The key issue is to prevent total information loss within a category, either RF environment or strategy, i.e., *maintain the dataset's diversity*. During a mission, when memory and compute time are limited, simple strategies for choosing which examples to drop from the training dataset in memory include:

- *Distance*: Drop instances that are very similar to other items (observables, strategy, and metrics);
- *Class imbalance*: Drop instances that are well-represented, i.e., have many more examples than other classes;
- *Fidelity*: Drop instances used to bootstrap the learner, for example generated from simulated data, or collected on similar-but-not-identical platforms; and
- *Age*: Drop the oldest instances. This approach is responsive to changing mission conditions.

Note that all examples collected in-mission can (and should) be stored in persistent storage for postmission analysis. Postmission analysis can analyze data for missing data (e.g., performance that can't be explained usually means missing observables), patterns across systems and missions (e.g., something seen only fleetingly at single nodes, but regularly across the group), and best practices (e.g., different methods of curating data).

It is hard to justify *permanently* forgetting data. Even though data may degrade over time, older knowledge can help bootstrap a system when nothing is known about a set of conditions. This kind of knowledge transfer accelerates learning in novel conditions. Good metadata records the history and context of the data, which premission analysis uses to decide which data to use for training in future missions. We need to overcome the barrier of deliberately destroying data for the sake of security, and instead find ways to appropriately protect it.

8.3.5 Data Privacy and Security

A common issue in data storage is that of data sensitivity. The system must protect both the *data* so that adversaries cannot reconstruct the original data, and the *models* so that adversaries cannot determine system vulnerabilities.

Differential privacy is a mathematical definition of privacy used to prove guarantees of protection against privacy attacks [73]. Differential privacy has valuable properties to analyze protection [74], including (1) quantification of privacy loss, (2) group privacy, (3) immunity to post-processing, and (4) composition and control over multiple computations.

There are many different ways to achieve differential privacy; most involve modifications to the original dataset. Anonymizing the dataset provides privacy protection, and also facilitates sharing data to other forums. Approaches include:

- *Label anonymization*: Give features (observables, controllables, and metrics) anonymous names, for example “tone jammer” becomes “environment#1,” and “notch filter” becomes “technique#1.” Anonymous labeling has the advantage that the decision maker cannot exploit the semantic meaning of the labels themselves. In other words, the AI stays agnostic to the particular dataset in question, allowing the same code to transfer to different capabilities, platforms, and tasks.
- *Normalization*: Normalize the data over a fixed range, say `int8` representation -128 to 127. Moreover, most learning models expect data to be normalized so that large-valued features do not dwarf small-valued features; model generalization is better when trained on normalized data.
- *Generalization*: Round and reduce the precision of data.
- *Discretization*: Replace ranges of values with discrete labels.
- *Perturbation*: Replace values with a uniform function modifying the values. Add noise if appropriate.

There are also methods to learn models that preserve privacy. The goal is to build a model that does not reveal features of the original training data, while still ensuring that any inference based on a specific example does not change. Approaches include:

- The system partitions the sensitive data and trains an ensemble of “teacher” models; then, using unlabeled nonsensitive data, the system trains a “student” model to mimic the ensemble [75].
- The system learns models on fully encrypted data [76–82], and outputs an encrypted result.
- A federated learning system splits training data among many nodes, and then combines the model parameters in a central location before redistributing the blended model back to the nodes [83–85]. Each node learns only on its own observations, and the central model never sees the original data.

Methods also exist to do federated learning while limiting communications requirements [86].

Recent work is combining these techniques (e.g., [87, 88]).

A good security approach requires multiple layers, including standard cybersecurity approaches. One layer of a comprehensive solution is to protect the model and the data to maintain information privacy, even if the model itself is compromised.

8.4 Conclusion

Despite myths to the contrary, the system does *not* need a large dataset to learn effectively. Managing the dataset to maintain diversity is the most effective way to create high-quality generalized models that handle novel situations (out-of-sample) and are resistant to adversarial attacks. Smaller datasets also have the benefit of improving computation time.

A good data engineer will ensure that the data is high quality, and can be reused on different platforms, for different tasks, and over time. Detailed interactions and use-case development ensures that the EW community and AI community use the same language and have the same goals [89]; we must also address the social gap between the two communities. Detailed metadata ensures high-quality decision-making and root-cause analysis, allowing the system to propagate errors from measurements to final inferences.

Garbage-in, Garbage-out.

— Thomas McRae, 1964, *The Impact of Computers on Accounting*

Life is like a sewer: what you get out of it depends on what you put into it.

— Tom Lehrer, 1959, *We will all go together when we go*

References

- [1] SCSC Data Safety Initiative Working Group, “Data Safety Guidance (Version 3.6),” Tech. Rep. SCSC-127I, 2024. Online: <https://scsc.uk/r127I:1>.
- [2] US DoD, *US DoD Data Strategy*, 2020. Online: <https://tinyurl.com/dod-data-strategy-2020>. Accessed 2020-10-08.
- [3] Wilkinson, M., et al., “The FAIR Guiding Principles for Scientific Data Management and Stewardship,” *Scientific Data*, Vol. 3, 2016. DOI: 10.1038/sdata.2016.18.
- [4] Savage, S., and J. Thibault, “Towards a Simulation Network,” in *Winter Simulation Conference*, 2015. Online: <http://simulation.su/uploads/files/default/2015-savage-thibault.pdf>.

- [5] Wang, R., and D. Strong, "Management Information Systems," Vol. 12, No. 4, 1996. Accessed 2025-01-26. Online: <http://www.jstor.org/stable/40398176>.
- [6] ISO/IEC 25012 Data Quality Model, 2025. Online: <https://iso25000.com/index.php/en/iso-25000-standards/iso-25012>. Accessed 2025-01-26.
- [7] Wang, J., et al., "Overview of Data Quality: Examining the Dimensions, Antecedents, and Impacts of Data Quality," *Journal of the Knowledge Economy*, 2023. DOI: 10.1007/s13132-022-01096-6.
- [8] Papastergios, V., and A. Gounaris, "A Survey of Open-source Data Quality Tools: Shedding Light on the Materialization of Data Quality Dimensions in Practice," *Computing Research Repository*, 2024. DOI: 10.48550/ARXIV.2407.18649.
- [9] Ehrlinger, L., and W. Wöß, "A Survey of Data Quality Measurement and Monitoring Tools," *Frontiers in Big Data*, Vol. 5, 2022. This article is part of the Research Topic: Automated Data Curation and Data Governance Automation. DOI: 10.3389/fdata.2022.850611.
- [10] Quinlan, J., "The Effect of Noise on Concept Learning," in *Machine Learning, An Artificial Intelligence Approach*, Morgan Kaufmann, 1986.
- [11] Zhu, X., and X. Wu, "Class Noise vs. Attribute Noise: A Quantitative Study of Their Impacts," *Artificial Intelligence Review*, Vol. 22, 2004. Online: <http://www.cse.fau.edu/~xqzhu/papers/AIR.Zhu.2004.Noise.pdf>.
- [12] Cluzeau, J., et al., *Concepts of Design Assurance for Neural Networks (CoDANN)*, 2020. Online: <https://tinyurl.com/CoDANN-2020>. European Union Aviation Safety Agency.
- [13] Li, S., et al., *Now Radios Can Understand Each Other: Modeling Language for Mobility*, 2011. Online: <https://www.wirelessinnovation.org/assets/documents/MLM2011.pdf>. (Reference implementation from 2014 at <https://www.wirelessinnovation.org/reference-implementations>.)
- [14] Li, S., et al., "An Implementation of Collaborative Adaptation of Cognitive Radio Parameters Using an Ontology and Policy Based Approach," *Analog Integrated Circuits and Signal Processing*, Vol. 69, No. 2-3, 2011. DOI: 10.1007/s10470-011-9681-y.
- [15] Cooklev, T., and L. Stanchev, "A Comprehensive and Hierarchical Ontology for Wireless Systems," *Wireless World Research Forum Meeting*, Vol. 32, 2014. Online: <https://tinyurl.com/cr-ontology>.
- [16] Horne, C., M. Ritchie, and H. Griffiths, "Proposed Ontology for Cognitive Radar Systems," *IET Radar, Sonar and Navigation*, Vol. 12, No. 12, 2018. DOI: 10.1049/iet-rsn.2018.5280.
- [17] Moskal, J., et al., "Towards a SpectralSPARQL Standard for Exchanging EMS Knowledge," in *MILCOM*, 2017. Online: <https://tinyurl.com/spectral-sparql>.
- [18] World Wide Web Consortium (W3C), *Web Ontology Language (OWL)*, 2025. Online: <https://www.w3.org/OWL/>. Accessed 2025-03-03.
- [19] Musen, M., "The Protégé Project: A Look Back and A Look Forward," *ACM SIG in AI*, Vol. 1, No. 4, 2015. Online: <https://protege.stanford.edu/>.
- [20] Maher, M., and R. Orlando, "Solving the "Garbage in-Garbage out" Data Issue through Ontological Knowledge Graphs," Processus Group, and Kord Technologies, Tech. Rep., 2019.
- [21] Gürbüz, S., et al., "An Overview of Cognitive Radar: Past, Present, and Future," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 34, 2019. DOI: 10.1109/MAES.2019.2953762.
- [22] Jian, T., et al., "Deep Learning for RF Fingerprinting: A Massive Experimental Study," *IEEE Internet of Things Magazine*, Vol. 3, No. 1, 2020. DOI: 10.1109/IOTM.0001.1900065.

- [23] Youssef, K., et al., *Machine Learning Approach to RF Transmitter Identification*, 2017. Online: <https://arxiv.org/abs/1711.01559>.
- [24] Ettus Research, *UHD (USRP Hardware Driver)*, 2025. Online: <https://www.ettus.com/sdr-software/uhd-usrp-hardware-driver/>. Accessed 2025-03-03.
- [25] von Spreckelsen, M., and E. Bamford, *Future Command and Control of Electronic Warfare*, 2019. Online: <https://www.japcc.org/articles/future-command-and-control-of-electronic-warfare/>.
- [26] "Cooperative Electronic Support Measure Operations," NATO, Tech. Rep. STANAG 4658, 2022.
- [27] "Common Electromagnetic Order of Battle Exchange Format," NATO, Tech. Rep. STANAG 6545, 2022.
- [28] Hilburn, B., et al., "SigMF: The Signal Metadata Format," in *GNU Radio Conference*, 2018. Online: <https://pubs.gnuradio.org/index.php/grcon/article/view/52>.
- [29] GNU Radio Foundation, *Signal Metadata Format (SigMF)*. Online: <https://github.com/gnuradio/SigMF>. Accessed 2025-02-26.
- [30] Sankhe, K., et al., "ORACLE: Optimized Radio cAssification through Convolutional neural Networks," in *INFOCOM*, 2019. DOI: 10.1109/INFOCOM.2019.8737463, Dataset available at <https://genesys-lab.org/oracle>.
- [31] Cooklev, T., R. Normoyle, and D. Clendenen, "The VITA 49 Analog RF-Digital Interface," *IEEE Circuits and Systems Magazine*, 2012. Online: <https://tinyurl.com/vita49>.
- [32] VMEbus International Trade Association (VITA), *Vita: Open Standards*, 2025. Online: <https://www.vita.com/>. Accessed 2025-03-03.
- [33] Google, *Protocol Buffers*, 2025. Online: <https://protobuf.dev/>. Accessed 2025-01-27.
- [34] Varda, K., *Cap'n Proto Cerealization Protocol*. Online: <https://capnproto.org>. Accessed 2025-01-27.
- [35] Google, *FlatBuffers*. Online: <https://flatbuffers.dev/>. Accessed 2025-01-27.
- [36] Haigh, K. Z., et al., "Parallel Learning and Decision Making for a Smart Embedded Communications Platform," BBN Technologies, Tech. Rep. BBN-REPORT-8579, 2015. Online: <http://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-SO.sm.pdf>.
- [37] Haigh, K. Z., et al., "Machine Learning for Embedded Systems: A Case Study," BBN Technologies, Tech. Rep. BBN-REPORT-8571, 2015. Online: <https://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-Embedded.pdf>.
- [38] Mirzasoleiman, B., J. Bilmes, and J. Leskovec, "Coresets for Data-Efficient Training of Machine Learning Models," in *ICML*, 2020. Online: http://web.cs.ucla.edu/~baharan/papers/mirzasoleiman20data_long.pdf.
- [39] Gong, Z., et al., "An End-to-End Joint Unsupervised Learning of Deep Model and Pseudo-Classes for Remote Sensing Scene Representation," in *IJCNN*, 2019. Online: <https://arxiv.org/abs/1903.07224>.
- [40] Zhang, C., H. Kjellstrom, and S. Mandt, "Determinantal Point Processes for Mini-Batch Diversification," in *Uncertainty in AI*, 2017. Online: <http://auai.org/uai2017/proceedings/papers/69.pdf>.
- [41] You, X., R. Wang, and D. Tao, "Diverse Expected Gradient Active Learning for Relative Attributes," *IEEE Transactions on Image Processing*, Vol. 23, No. 7, 2014. DOI: 10.1109/TIP.2014.2327805.

- [42] Shi, L., and Y.-D. Shen, "Diversifying Convex Transductive Experimental Design for Active Learning," in *IJCAI*, 2016. Online: <https://www.ijcai.org/Proceedings/16/Papers/285.pdf>.
- [43] Zhang, Y., et al., "Energy-Entropy Competition and the Effectiveness of Stochastic Gradient Descent in Machine Learning," *Molecular Physics*, No. 16, 2018. DOI: 10.1080/00268976.2018.1483535.
- [44] Gong, Z., P. Zhong, and W. Hu, "Diversity in Machine Learning," *IEEE Access*, Vol. 7, 2019. Online: <https://ieeexplore.ieee.org/document/8717641>.
- [45] Ren, P., et al., *A Survey of Deep Active Learning*, 2020. DOI: 10.48550/arXiv.2009.00236.
- [46] Rizk, H., A. Shokry, and M. Youssef, *Effectiveness of Data Augmentation in Cellular-based Localization Using Deep Learning*, 2019. Online: <https://arxiv.org/abs/1906.08171>.
- [47] Shorten, C., and T. Khoshgoftaar, "A Survey on Image Data Augmentation for Deep Learning," *Big Data*, Vol. 6, No. 60, 2019. DOI: 10.1186/s40537-019-0197-0.
- [48] Sinha, R., et al., "Data Augmentation Schemes for Deep Learning in an Indoor Positioning Application," *Electronics*, Vol. 8, No. 554, 2019. DOI: 10.3390/electronics8050554.
- [49] Xie, F., et al., "Data Augmentation for Radio Frequency Fingerprinting via Pseudo-Random Integration," *IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. 4, No. 3, 2020. DOI: 10.1109/TETCI.2019.2907740.
- [50] Chung, Y., et al., *Learning Unknown Examples for ML Model Generalization*, 2019. Online: <https://arxiv.org/abs/1808.08294>.
- [51] Brownlee, J., *Train Neural Networks with Noise to Reduce Overfitting*, 2018. Online: <https://tinyurl.com/noise-and-overfitting>. Accessed 2020-11-22.
- [52] Goodfellow, I., Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016. Online: <http://www.deeplearningbook.org>.
- [53] Chen, T., et al., "Big Self-Supervised Models are Strong Semi-Supervised Learners," in *NeurIPS*, 2020. Online: <https://arxiv.org/abs/2006.10029v2>.
- [54] Sohn, K., et al., "FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence," in *NeurIPS*, 2020. Online: <https://arxiv.org/abs/2001.07685>.
- [55] Reed, R., and R. Marks II, *Neural Smthing: Supervised Learning in Feedforward Artificial Neural Networks*. Bradford, PA: Bradford Books, 1999.
- [56] Bishop, C., "Training with Noise is Equivalent to Tikhonov Regularization," *Neural Computation*, Vol. 7, No. 1, 2008. DOI: 10.1162/neco.1995.7.1.108.
- [57] Mirzasoleiman, B., K. Cao, and J. Leskovec, "Coresets for Robust Training of Neural Networks against Noisy Labels," in *NeurIPS*, 2020. Online: <https://tinyurl.com/training-corsets>.
- [58] Huang, L., et al., "Data Augmentation for Deep Learning-Based Radio Modulation Classification," *IEEE Access*, Vol. 8, 2020. DOI: 10.1109/ACCESS.2019.2960775.
- [59] Kuzdeba, S., *Radio Frequency Machine Learning: A Practical Deep Learning Perspective*. Norwood, MA: Artech House, 2025.
- [60] Soltani, N., et al., "More Is Better: Data Augmentation for Channel-Resilient RF Fingerprinting," *IEEE Communications Magazine*, Vol. 58, No. 10, 2020. DOI: 10.1109/MCOM.001.2000180.
- [61] Sheeny, M., A. Wallace, and S. Wang, "RADIO: Parameterized Generative Radar Data Augmentation for Small Datasets," *Applied Sciences*, Vol. 10, No. 11, 2020. DOI: 10.3390/app10113861.

- [62] O'Hara, S., "Data Management and Bootstrapping Processing for Machine Learning and Classification Development," US 11651290, 2023. Online: <https://patents.justia.com/patent/11651290>.
- [63] Tu, Y., et al., "Large-scale Real-world Radio Signal Recognition with Deep Learning," *Chinese Journal of Aeronautics*, Vol. 35, No. 9, 2022. Data accessed 2022-08-10 from <https://gitee.com/heu-linyun/Codes>. DOI: 10.1016/j.cja.2021.08.016.
- [64] Yuan, X., et al., "Adversarial Examples: Attacks and Defenses for Deep Learning," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 30, No. 9, 2019. Online: <https://arxiv.org/abs/1712.07107>.
- [65] Ren, K., et al., "Adversarial Attacks and Defenses in Deep Learning," *Engineering*, Vol. 6, No. 3, 2020. DOI: 10.1016/j.eng.2019.12.012.
- [66] Goodfellow, I., et al., "Generative Adversarial Nets," in *NeurIPS*, 2014. Online: <https://tinyurl.com/gans-2014>.
- [67] Karras, T., et al., "Progressive Growing of GANs for Improved Quality, Stability, and Variation," in *ICLR*, 2018. Online: https://research.nvidia.com/publication/2017-10_Progressive-Growing-of.
- [68] Li, M., et al., "Generative Adversarial Networks-Based Semi-Supervised Automatic Modulation Recognition for Cognitive Radio Networks," *Sensors*, 2018. Online: <https://www.mdpi.com/1424-8220/18/11/3913/pdf-vor>.
- [69] Davaslioglu, K., and Y. Sagduyu, "Generative Adversarial Learning for Spectrum Sensing," in *International Conference on Communications*, 2018. DOI: 10.1109/ICC.2018.8422223.
- [70] Majumder, U., E. Blasch, and D. Garren, *Deep Learning for Radar and Communications Automatic Target Recognition*. Norwood, MA: Artech House, 2020.
- [71] Shi, Y., K. Davaslioglu, and Y. Sagduyu, *Generative Adversarial Network for Wireless Signal Spoofing*, 2019. Online: <https://arxiv.org/abs/1905.01008>.
- [72] O'Shea, T., et al., "Physical Layer Communications System Design Over-the-Air Using Adversarial Networks," in *European Signal Processing Conference*, 2018. DOI: 10.23919/EUSIPCO.2018.8553233.
- [73] Wood, A., et al., "Differential Privacy: A Primer for a Non-Technical Audience," *Vanderbilt Journal of Entertainment & Technology Law*, Vol. 21, No. 1, 2018. Online: https://scholar.harvard.edu/files/salil/files/differential_privacy_primer_nontechnical_audience.pdf.
- [74] Dwork, C., and A. Roth, "The Algorithmic Foundations of Differential Privacy," in, 3-4, Vol. 9, 2014. DOI: 10.1561/04000000042.
- [75] Papernot, N., et al., "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data," in *ICLR*, 2017. Online: <https://arxiv.org/abs/1610.05755>.
- [76] Bost, R., et al., *Machine Learning Classification over Encrypted Data*, International Association for Cryptologic Research, 2014. Online: <https://eprint.iacr.org/2014/331.pdf>.
- [77] Dowlin, N., et al., "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy," in *ICML*, 2016. Online: <http://proceedings.mlr.press/v48/giladbachrach16.pdf>.
- [78] Tang, X., et al., *When Homomorphic Cryptosystem Meets Differential Privacy: Training Machine Learning Classifier with Privacy Protection*, 2018. Online: <https://arxiv.org/abs/1812.02292>.
- [79] Lou, Q., and L. Jiang, "SHE: A Fast and Accurate Deep Neural Network for Encrypted Data," in *NeurIPS*, 2019. Online: <https://tinyurl.com/encrypted-dn>.

- [80] Catak, F., et al., "Practical Implementation of Privacy Preserving Clustering Methods Using a Partially Homomorphic Encryption Algorithm," *Electronics*, Vol. 9, 2020. DOI: 10.3390/electronics9020229.
- [81] Kumbhar, H., and R. Srinivasa, "Machine Learning Techniques for Homomorphically Encrypted Data," in *Applied Computer Vision and Image Processing*, 2020. DOI: 10.1007/978-981-15-4029-5_25.
- [82] Lee, J.-W., et al., "Privacy-Preserving Machine Learning with Fully Homomorphic Encryption for Deep Neural Network," *IEEE Access*, Vol. 10, 2022. DOI: 10.1109/ACCESS.2022.3159694.
- [83] Geyer, R., T. Klein, and M. Nabi, "Differentially Private Federated Learning: A Client Level Perspective," in *NIPS Workshop on Machine Learning on the Phone and other Consumer Devices*, 2017. Online: <https://arxiv.org/abs/1712.07557>.
- [84] Oldenhof, M., et al., "Industry-Scale Orchestrated Federated Learning for Drug Discovery," in *Innovative Applications of Artificial Intelligence*, 2023. DOI: 10.1609/aaai.v37i13.26847.
- [85] Abasi, A. K., M. Aloqaily, and M. Guizani, "6G mmWave Security Advancements through Federated Learning and Differential Privacy," *IEEE Transactions on Network and Service Management*, 2025. DOI: 10.1109/TNSM.2025.3528235.
- [86] Yang, P., et al., "Decentralized Over-the-Air Federated Learning by Second-Order Optimization Method," *IEEE Transactions on Wireless Communications*, Vol. 23, No. 6, 2024. DOI: 10.1109/TWC.2023.3327610.
- [87] Aziz, R., et al., "Exploring Homomorphic Encryption and Differential Privacy Techniques towards Secure Federated Learning Paradigm," *Future Internet*, Vol. 15, No. 9, 2023. DOI: 10.3390/fi15090310.
- [88] Sébert, A., "Combining differential privacy and homomorphic encryption for privacy-preserving collaborative machine learning," Ph.D. dissertation, University Paris-Saclay, Orsay, France, 2023. Online: <https://theses.hal.science/tel-04223076/>.
- [89] Haigh, K. Z., et al., "Rethinking Networking Architectures for Cognitive Control," in *Microsoft Research Cognitive Wireless Networking Summit*, 2008. Online: <http://www.cs.cmu.edu/~khaigh/papers/haigh-msr08.pdf>.

9

Architecture

Using the deeper understanding of situation assessment and decision making developed in Chapters 1–8, we can redraw the components of the cognitive system from Figure 1.8 as illustrated in Figure 9.1. A modular architecture provides the backbone to these capabilities, allowing different techniques to provide different services and ensuring consistent flow of information and control. This chapter touches on software and hardware architecture concerns. Distributed-vs-centralized architectures were discussed in Section 5.5, and time layering concerns were in Section 6.1.6.

9.1 Software Architecture: Interprocess

While highly adaptive, some advanced radar and software-defined radio (SDR) architectures still rely on tailored APIs that expose each parameter separately. This approach is not amenable to real-time cognitive control in EW systems, because the tight coupling means that an AI can't make global decisions.

To eliminate this barrier, the constituent modules must be highly modular and composable. A general interface allows modules to expose their parameters and dependencies, and thus enables global optimization and computational load balancing across multiple processors [1–5]. A modular system also allows for real-time composability of the system, where modules can be swapped in/out depending on mission needs [1]. To this end, we need a *broker* that provides:

- *One consistent interface* to any and all modules, so that if a module changes, or a new module is created, none of its controlling or subordinate modules need to be modified. Figure 9.2 sketches the difference between a tightly coupled and a brokered interface. This modular approach solves the $m \times n$ problem (i.e., upgrades or replacements of modules).
- *Coordination of control* to ensure that multiple controllers don't overwrite each other. A module may not expose its parameters for outside control via any interface except the broker.

The first software architecture that provided this capability in RF was the ADROIT broker [1], using the abstract structure illustrated in Figure 9.3. Each

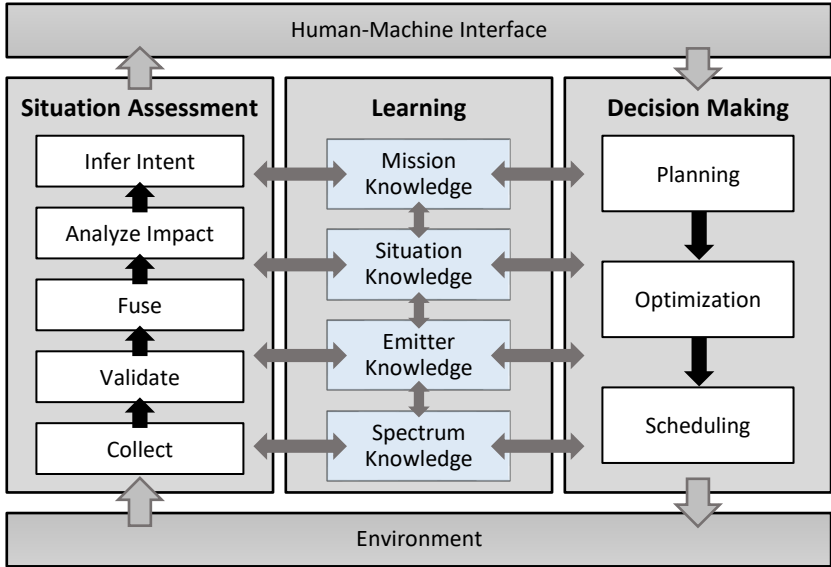


Figure 9.1 In the cognition loop, decision making critically depends on situation assessment. Learning captures the dynamic nature of the domain.

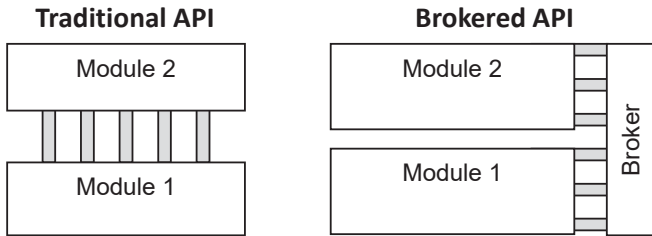


Figure 9.2 Adding or modifying modules in a tightly coupled traditional API requires updates to all connecting modules, while a brokered API doesn't impact other modules.

module exposes its parameters and their properties (especially read/write). When a module changes (e.g., adding a new parameter), it simply exposes the new parameter, `exposeParameter(name, properties)`, rather than adding a new API function for that new parameter. ADROIT instantiated the broker with the particular modules shown in Figure 9.4. ADROIT developers also performed

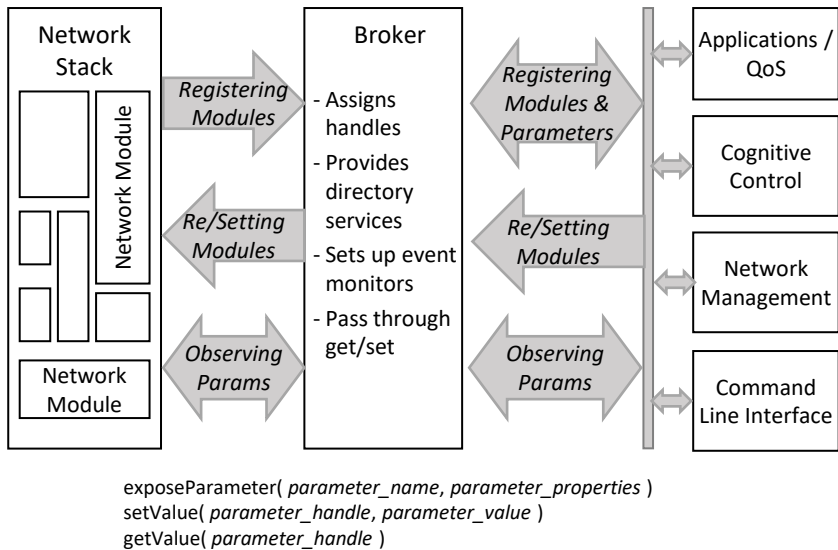


Figure 9.3 ADROIT was the first network architecture to support cognitive control through a broker. (From [1].)

several detailed walkthroughs showing the separation-of-concerns of networking modules versus a cognitive layer [1].

The advantages of this generic, modular approach is that it doesn't restrict the form of cognition, allowing designers to choose techniques appropriate to the problem.

- The architecture supports almost *any* cognitive technique,
- The architecture can contain *multiple* cognitive techniques, and
- The architecture doesn't *mandate* cognitive techniques.

The modularity allows for fine-grained decomposition of capability; for example, a module might compute and publish one (and only one) statistic (e.g., BER) that becomes an input to many modules.

An interesting side effect of a broker-based approach is that the distinction between "application" and "module" blurs. A given module may request changes not only "down" the stack, but also "up," for example, requesting that the video reduce its resolution to meet bandwidth limitations. In theory, any module can issue instructions to any other module.

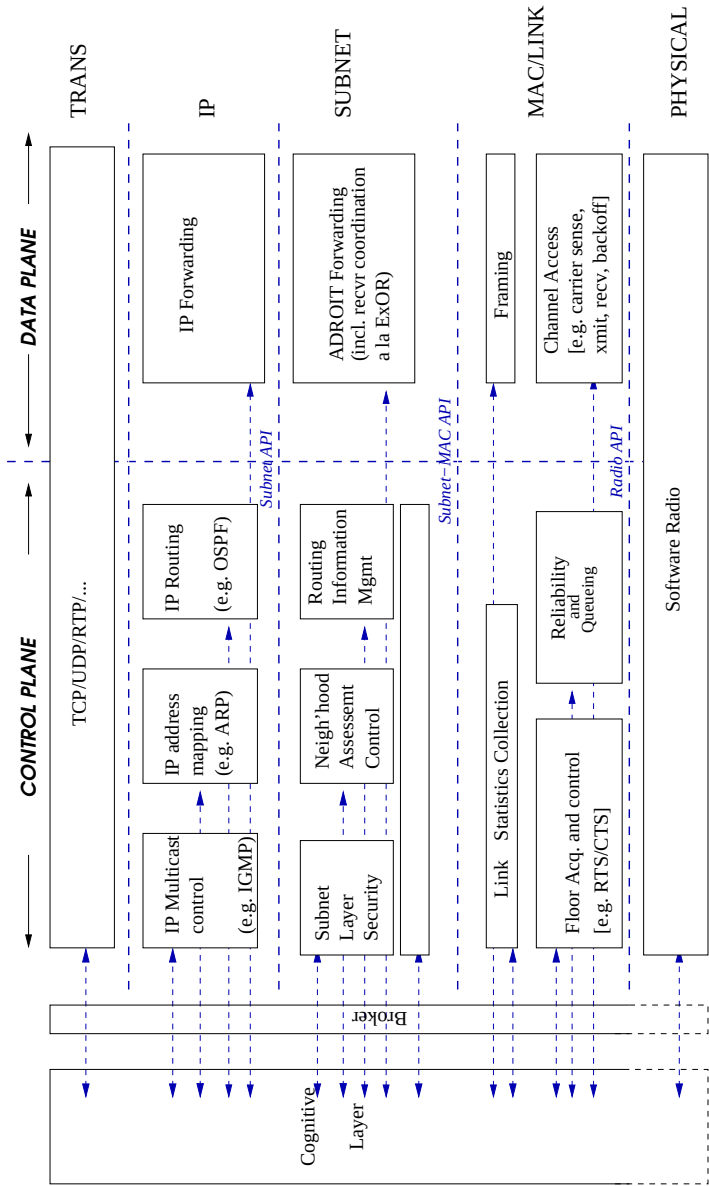


Figure 9.4 ADROIT used a broker to provide cognitive control for a modular networking stack. (From [1].)

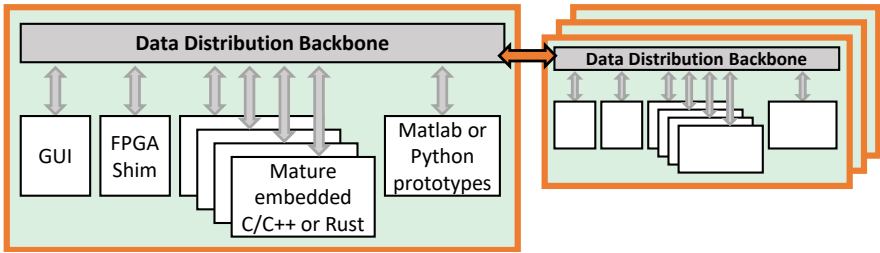


Figure 9.5 A data broker enables interprocess communication within a single node or across nodes in the network.

Modern RF systems use various *publish/subscribe* systems such as the Data Distribution Service (DDS) [6, 7] or Kafka [8]. Pub/Sub is a messaging service that decouples services that produce events (publishers) from services that process events (subscribers). DDS is asynchronous, while Kafka is a streaming service. While pub/sub systems are most commonly used to distribute services across a wide-area network, they are equally effective within a single CPU, as illustrated in Figure 9.5. This mechanism therefore decouples the software capabilities from the specific compute platform(s), allowing the system designers to allocate processes to compute resources as appropriate for the platform and mission needs. Services can naturally be extended across a bus or multiple cores on a single platform, or across multiple platforms, with increasing latency.

The broker-based approach is also a mechanism for evaluating and future-proofing the system. From a test perspective, the standard API allows apples-to-apples comparison of modules that provide identical functionality. For example, if multiple algorithms perform signal separation (deinterleaving), the system's performance can be evaluated with each algorithm, allowing for a direct and objective comparison of results. Performance of the system can be evaluated with each, and results can be directly compared. Moreover, the evaluators (who know the ground truth) can swap out the algorithmic solutions with a method that reports the true results, enabling comparison of the algorithms against the perfect solution. From a perspective of future-proofing, the standard API makes it easy to replace modules with newer alternatives, and to explore prototype concepts.

One of the challenges in implementing cognitive control is deconflicting with legacy systems. Ideally, every module in the system should subscribe to the same broker so that every module has the context-awareness it requires to make accurate inferences. In practice, however, legacy systems do not connect to a broker, and

more importantly, often do not allow the cognitive system to track all the incoming data. A common problem is that the legacy system will respond to all of the RF signals it “knows,” only passing on unknown signals. This approach leads to exclusion bias (Section 8.1.2) and incorrect conclusions due to lack of awareness of longer-term patterns.

9.2 Software Architecture: Intraprocess

Threading and shared memory approaches are appropriate when the interaction between processes needs to be more tightly coupled (e.g., memory access and similar computations). Any instance-based learning method that compares new instances directly against training instances is likely to fall into this category.

The SVMs of the BBN SO (Example 1.1), require the RRE and LTRE to be in separate threads of the same process [9]. Both RRE and LTRE require high-frequency, low-latency access to the same internal datastore; selected instances from the training data become the support vectors in shared memory. Figure 9.6 illustrates the distribution of function across threads, supporting the functions of Figure 5.2. Figure 9.7 illustrates the runtime behavior of the threads.

9.3 Language Choices

ML researchers have widely adopted Python as the prototyping language of choice. Unfortunately, the wide availability of GPU platforms has led to the belief that Python is appropriate for embedded systems. Pereira et al. performed an analysis of the energy, time, and memory efficiency of 27 different languages on 13 benchmark problems [10]. Table 9.1 shows the summary results for several of these languages. The best result receives a score of 1.00; other scores are relative to the best performer. If we assume that these factors are equally important, the *Sum* column shows the relative rankings of some of these 27 languages. C, C++, Rust, and Ada have good performance. Python’s performance shows that it is unequivocally inappropriate for embedded systems.

Several variants of Python exist for embedded systems, including CPython, PyMite, and MicroPython, but benchmark datasets have not been created, and formal language comparisons have not been performed. It is unlikely, however, that these would perform well when compared to C/C++ or Rust. First, there is no flavor of Python that supports threading; the code is converted to heavy-weight multiple processes.¹ Second, Python is an interpreted language, so must

1. As of release 3.13 (Oct 2024), CPython has experimental support for threads [11]. This release has several known limitations, for example it makes some objects immortal, in that they are never de-allocated from memory.

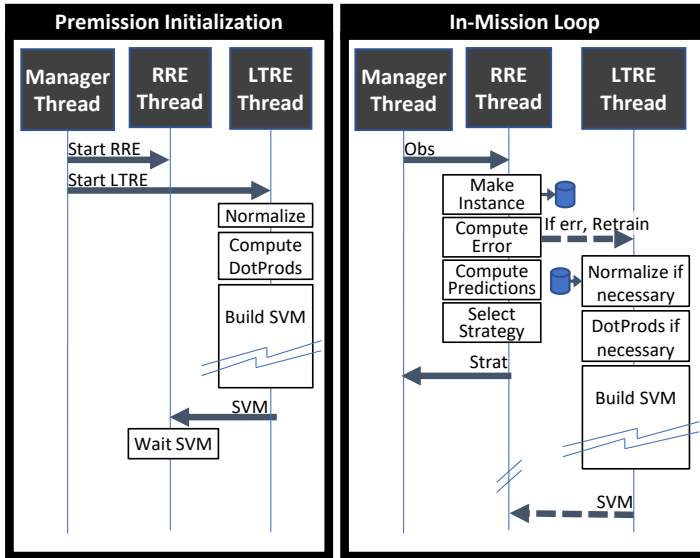


Figure 9.6 The BBN SO uses threads to share data between capabilities. (Example 1.1, [9].)

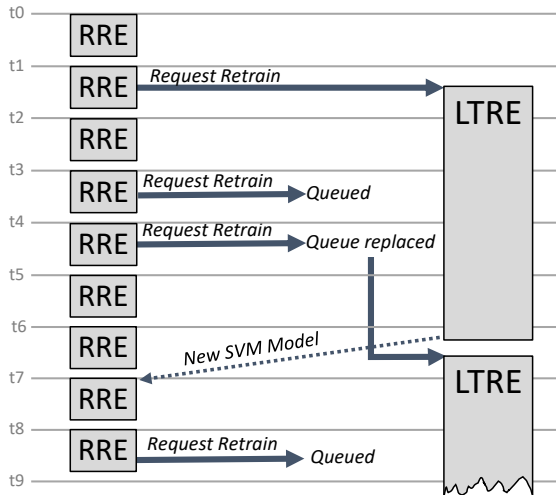


Figure 9.7 The RRE computes strategy choices on a hard real-time schedule, while the LTRE operates in a background thread when computation is available. (Example 1.1, [9].)

Table 9.1
C/C++ and Rust are Better for Embedded Systems than Python (Results from [10])

Language	Energy (J)	Time (ms)	Memory (Mb)	Sum
C	1.00	1.00	1.17	3.17
Rust	1.03	1.04	1.54	3.61
C++	1.34	1.56	1.34	4.24
Ada	1.70	1.85	1.47	5.02
Pascal	2.14	3.02	1.00	6.16
Fortran	2.52	4.20	1.24	7.96
Java	1.98	1.89	6.01	9.88
Lua	45.98	82.91	6.72	135.61
Python	75.88	71.90	2.80	150.58

allocate memory for the interpreter itself. Third, Python uses garbage-collecting for memory-management, and therefore is inappropriate for hard real-time systems. For compute-intensive applications, the guidance is to write a C module that gets called from Python. Libraries are being developed by the community, but at this time, unless a library has been developed for your specific hardware, you will have to write your own C module. Embedded variants of Python are a good way to introduce engineers to the challenges inherent in embedded systems, but not yet appropriate for final deployment.

I see [MicroPython] as the gateway drug of embedded development, you get pulled in by how easy it looks and then you move one to a more performant language when you need it.

— Edouard Belval (Amazon), Jan 2021 [12]

These results highlight one of the necessary activities in building the CogEW solution:

Translating from common ML libraries to embedded hard real-time software takes significant effort.

The BBN SO’s SVM model is derived from WEKA [13], which (at the time) was one of the fastest libraries available, and available in both Java and C/C++. The effort to translate the code for the embedded system involved removing unnecessary code, flattening or replacing object structures, and evaluating numerical representations [14]. The largest effort, by far, was to split the code to be multithreaded: to have the RRE operate in hard real-time with guaranteed timing for strategy choices, and the LTRE operate in a background thread when computation resources are

available. The embedded code ran at 5% of the runtime from the baseline C/C++ code, where baseline disabled the “easy-to-ablate” capabilities. The original (off-the-shelf) Java code had additional items that we could not ablate, such as shared computation across threads, and ran a 1,000,000 times slower than the embedded code. Putting the RRE and LTRE in separate *processes* would likely increase this time yet another >1000x because the processes would both compute shared values, and also the system would incur overhead for sharing the dataset across processes.

Embedded systems software engineers have a long history with C and C++, but even experienced individuals occasionally trip over some of the poor semantics. Rust, meanwhile, is a memory-safe language that avoids some of C/C++’s pitfalls [15, 16]. If starting a new project from scratch, it is therefore worth considering Rust as the implementation language.

9.4 Hardware Choices

Many CogEW solutions will be software (or firmware) overlays on existing legacy systems, in which case this section isn’t relevant. For those who have a choice, there are several trade-offs to consider. Callout 9.1 highlights some of the hardware design concerns for EW systems, and Table 9.2 summarizes the key features of CPUs, FPGAs, GPUs, and custom ASICs to consider.

Customers are discovering that there is no single ‘best’ piece of hardware to run the wide variety of AI applications, because there’s no single type of AI. The constraints of the application dictate the capabilities of the required hardware from data center to edge to device, and this reiterates the need for a more diverse hardware portfolio.

— Naveen Rao (Intel), 2018 [17]

Callout 9.1 Embedded EW systems, especially those at the tactical edge, create unique hardware considerations. Wide instantaneous bandwidth exacerbates the processing resources and power issue for EW.

Lack of access to cloud computing requires <i>local</i> processing	Limited power from generators and batteries requires <i>efficient</i> systems
Space and weight limitations require <i>compact</i> solutions	Mission-critical safety and security require <i>reliable</i> and <i>trusted</i> solutions
Upgrade lifecycle of 10+ years requires scalable, <i>modular</i> solutions	Harsh environments with varying temperatures, pressure, shock, and humidity require <i>ruggedized</i> systems

Table 9.2
The Key Features of CPUs, FPGAs, GPUs, and Custom ASICs

Processor	Characteristics	Pros/Cons
CPU	• Traditional processor for general-purpose applications • Single- and multicore, plus specialized blocks (e.g., floating point) • Lots of cache; best latency performance from memory to core • Can run independently and host an operating system • Optimized for sequential processing with limited parallelism	• Multiple mission modes; future-proofing • Data locality • Very reliable vendors • Industrial versions available • Software availability and programmer talent • High performance for non-DeepNet ML • High performance on inference that requires a lot of memory
FPGA	• Flexible collection of logical elements that can be changed in the field • Tailor-made architecture configured application • Higher performance, lower cost, and lower power consumption	• Reconfigurable • Guaranteed timing performance • Large datasets • Compute-intensive applications • Not good for floating-point operations • Harder to program than other platforms, therefore relatively inflexible
GPU	• Built to do many identical parallel operations (e.g., matrix math) • Thousands of identical processor cores	• Reliable vendors • Most software support for DeepNets • High power consumption • Computation improvement comes at a cost of memory transfer • Unreliable for hard real-time tasks • Black-box specifications • Shorter lifecycle than CPU or FPGA • Security features less advanced
ASIC	• Application-specific integrated circuitry • The most efficient performance-to-power solution	• Single mode (dedicated AI process) • Short lifespan due to rapid technology obsolescence (< 1 year) • Limited software support

CPUs have a long history of good performance in hard real-time systems, and are flexible to changes in the software applications over the decade-long lifespan. FPGAs are reconfigurable and provide huge power-efficient processing capabilities, reducing thermal management and space requirements. This feature allows the integration of acceleration hardware in small housings and extreme environments.

FPGAs outperform CPUs for fixed algorithms over streaming data, but the cost (dollars) can be prohibitive.

In the first edition, we stated that GPUs would likely be inappropriate for EW problems because of security concerns, power usage, and reliability on hard real-time tasks [18, 19]. The majority of these issues, particularly the security concerns, have been addressed due to demand by the automotive industry. Demonstrations showing timing requirements and reliability have also been performed in EW applications [20–23]. These articles note that the software needs to be redesigned to take advantage of the GPU features. It is still also worth considering the impact of hardware lifecycles: GPU lifecycles are significantly shorter than CPUs or FPGAs. In the context of defense programs, this issue makes sustainment more difficult and diminishing manufacturing sources and material shortages more likely. GPUs are not necessarily required for EW systems, because a lot of EW data is very sparse, and FPGAs often do the heavy lifting for the streaming data.

EW has a lot of data, but EW is not a “big-data” environment.

ASICs may have a role in EW, assuming that the short lifespan of the hardware technology, high upfront cost, and the single application it can host, is acceptable to the community.

Neuromorphics, photonics, and quantum computing are emerging and future hardware capabilities that could significantly improve signal processing, threat detection, and real-time decision making in complex EW environments.

Neuromorphic computers are systems designed to mimic the structure and functionality of the human brain using artificial neurons and synapses. A lot of research has focused on replicating the analog nature of biological computation and the role of neurons in cognition. These computers offer benefits such as energy efficiency, parallel processing capabilities, and adaptability, making them particularly well-suited for tasks like pattern recognition, signal processing, and real-time decision making in complex environments. Recent work has significantly advanced applications in communications, radar, and EW. Neuromorphic computing systems utilize spiking neural nets (SNNs) to perform RF sensing, data decoding, emitter recognition, and data fusion, enhancing efficiency and results of EMS systems [24–27]. Additionally, the development of reconfigurable architectures and virtualization for neuromorphic computing has facilitated the rapid prototyping of neuromorphic systems, enabling efficient signal processing for various applications, including radar and communications [28, 29]. These advancements demonstrate the potential of neuromorphic computing to enhance the performance and adaptability of systems in communications, radar, and EW domains.

Photonic integrated circuits eliminate analog-to-digital conversion, and offer ultra-low latency processing with lower power consumption through the use of light-based components [30–32]. Zhang et al.'s system-on-chip processor employs a fully integrated photonic signal pathway in the analog domain, enabling rapid demixing of received mixtures and recovering the signal-of-interest in under 15 picoseconds. This reduction in latency surpasses electronic counterparts by more than three orders of magnitude [32].

Quantum computing use *qubits* rather than bits to encode information; qubits can exist in multiple states simultaneously due to superposition and entanglement. This property enables quantum computers to perform complex calculations exponentially more efficiently than classical computers, thereby offering the ability to solve previously unsolvable complex problems, enabling the rapid processing of EW data [33, 34]. Other uses of quantum technologies include *quantum radars* that are resilient to countermeasures and operate with greater accuracy and sensitivity than traditional radars, and *quantum communication*, which is resistant to eavesdropping and interception while improving the reliability and range of communications. The first quantum network was built by BBN Technologies in 2005 in the Boston area over fiber covering 29 km [35]. In 2017, China demonstrated a satellite-to-ground quantum key distribution over optical links covering 1200 km, and in 2021 they reported transmission between mobile drones [36]. China is building a nation-wide quantum key distribution network; by 2021, this covered 4,600 km and at least Beijing, Shanghai, Jinan, and Hefei [37].

9.5 Conclusion

From the software architecture perspective, supporting system-wide cognitive control is best accomplished with a modular infrastructure, wherein each module connects to a backbone and exposes its parameters. Python is useful for early prototyping, while C/C++ and Rust are appropriate for the embedded setting.

From the hardware perspective, a combination of CPUs and FPGAs will generally be the most appropriate for EW due to their support for hard real-time operations and streaming computation. GPUs may be appropriate for the embedded EW environment, but do not make that assumption unconditionally.

References

- [1] Haigh, K. Z., et al., "Rethinking Networking Architectures for Cognitive Control," in *Microsoft Research Cognitive Wireless Networking Summit*, 2008. Online: <http://www.cs.cmu.edu/~khaigh/papers/haigh-msr08.pdf>.
- [2] Troxel, G., et al., "Enabling Open-Source Cognitively-Controlled Collaboration Among Software-Defined Radio Nodes," *Computer Networks*, Vol. 52, No. 4, 2008. DOI: 10.1016/j.comnet.2007.11.010.

- [3] Blossom, E., "GNU Radio: Tools for Exploring the Radio Frequency Spectrum," *Linux journal*, Vol. 2004, No. 122, 2004. Online: <https://www.linuxjournal.com/article/7319>.
- [4] Casimiro, A., J. Kaiser, and P. Verissimo, "An Architectural Framework and a Middleware for Cooperating Smart Components," in *Computing Frontiers*, 2004. DOI: 10.1145/977091.977098.
- [5] Hiltunen, M., and R. Schlichting, "The Cactus Approach to Building Configurable Middleware Services," in *Workshop on Dependable System Middleware and Group Communication*, 2000. Online: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.7555>.
- [6] Object Management Group, *Data Distribution Service (DDS), version 1.0*, 2004. Online: <https://www.omg.org/spec/DDS/1.0>.
- [7] Object Management Group, *Data Distribution Service Portal*, 2020. Online: <https://www.omg.org/omg-dds-portal/>. Accessed 2020-10-21.
- [8] Apache, *Kafka*, 2025. Online: <https://kafka.apache.org/>. Accessed 2025-03-03.
- [9] Haigh, K. Z., et al., "Parallel Learning and Decision Making for a Smart Embedded Communications Platform," BBN Technologies, Tech. Rep. BBN-REPORT-8579, 2015. Online: <http://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-SO.sm.pdf>.
- [10] Pereira, R., et al., *Ranking Programming Languages by Energy Efficiency*, 2021. DOI: 10.1016/j.scico.2021.102609.
- [11] Python Software Foundation, 2024. Online: <https://docs.python.org/3/howto/free-threading-python.html>. Accessed 2025-01-08.
- [12] Belval, E. Online: <https://news.ycombinator.com/item?id=25861584>. Accessed 2025-01-08.
- [13] University of Waikato, NZ, *WEKA: The Workbench for Machine Learning*. Online: <https://www.cs.waikato.ac.nz/ml/weka/>. Accessed 2025-03-03.
- [14] Haigh, K. Z., et al., "Machine Learning for Embedded Systems: A Case Study," BBN Technologies, Tech. Rep. BBN-REPORT-8571, 2015. Online: <https://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-Embedded.pdf>.
- [15] Klabnik, S., and C. Nichols, *The Rust Programming Language*, 2nd Edition. Burlingame, CA: No Starch Press, 2024.
- [16] The Rust Team, 2024. Online: <https://www.rust-lang.org/>.
- [17] Rao, N., *Intel AI – The Tools for the Job*, 2018. Online: <https://tinyurl.com/intel-hw-trades>.
- [18] Yang, M., et al., "Avoiding Pitfalls when Using NVIDIA GPUs for Real-Time Tasks in Autonomous Systems," in *Euromicro Conference on Real-Time Systems*, 2018. DOI: 10.4230/LIPIcs.ECRTS.2018.20.
- [19] Maceina, T., and G. Manduchi, "Assessment of General Purpose GPU Systems in Real-Time Control," Vol. 64, No. 6, 2017. DOI: 10.1109/TNS.2017.2691061.
- [20] Krijestorac, E., et al., "Machine Learning-Assisted Computationally Efficient Target Detection and Tracking in Massive Fully Digital Phased Arrays," *IEEE Transactions on Radar Systems*, Vol. 1, 2023. DOI: 10.1109/TRS.2023.3298340.
- [21] Liu, G., et al., "A GPU-Based Real-Time Processing System for Frequency Division Multiple-Input-Multiple-Output Radar," *IET Radar, Sonar, and Navigation*, Vol. 17, No. 10, 2023. DOI: 10.1049/rsn2.12439.
- [22] Zhao, X., et al., "GPU-Accelerated Signal Processing for Passive Bistatic Radar," *Remote Sensing*, Vol. 15, No. 22, 2023. DOI: 10.3390/rs15225421.

- [23] Yang, T., et al., "An Embedded-GPU-Based Scheme for Real-Time Imaging Processing of Unmanned Aerial Vehicle Borne Video Synthetic Aperture Radar," *Remote Sensing*, Vol. 16, No. 1, 2024. DOI: 10.3390/rs16010191.
- [24] Chen, J., N. Skatchkovsky, and O. Simeone, "Neuromorphic Integrated Sensing and Communications," *IEEE Wireless Communications Letters*, Vol. 12, No. 3, 2023. DOI: 10.1109/LWC.2022.3231388.
- [25] Isik, M., et al., *Accelerating Sensor Fusion in Neuromorphic Computing: A Case Study on Loihi-2*, 2024. Online: <https://arxiv.org/abs/2408.16096>.
- [26] Luo, Z., et al., "Radar Emitter Recognition Based on Spiking Neural Networks," *Remote Sensing*, Vol. 16, No. 14, 2024. DOI: 10.3390/rs16142680.
- [27] Harbour, S., et al., "Real-time Vision-based Control of SWaP-constrained Flight System with Intel Loihi 2," in *Digital Avionics Systems Conference*, 2023. DOI: 10.1109/DASC58513.2023.10311158.
- [28] Mack, J., et al., "RANC: Reconfigurable Architecture for Neuromorphic Computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 40, No. 11, 2021. DOI: 10.1109/TCAD.2020.3038151.
- [29] Isik, M., J. Naoukin, and I. C. Dikmen, "NeuroVM: Dynamic Neuromorphic Hardware Virtualization," in *IEEE Green and Sustainable Computing Conference*, IEEE, 2024.
- [30] Han, W., et al., *Photonic RF Channelization Based on Microcombs*, 2024. Online: <https://arxiv.org/abs/2401.09698>.
- [31] Zhang, Z., Y. Liu, and B. J. Eggleton, "Photonic Generation of Radar Signals with 30 GHz Bandwidth and Ultra-High Time-Frequency Linearity," *Lightwave Technology*, Vol. 40, No. 14, 2022. DOI: 10.1109/JLT.2022.3164637.
- [32] Zhang, W., et al., "A System-on-Chip Microwave Photonic Processor Solves Dynamic RF Interference in Real Time with Picosecond Latency," *Light Science & Applications*, Vol. 13, 2024. DOI: 10.1038/s41377-023-01362-5.
- [33] Krelina, M., *Quantum Warfare: Definitions, Overview and Challenges*, 2021. DOI: 10.48550/arXiv.2103.12548.
- [34] Congressional Research Service, *Defense Primer: Quantum Technology*, 2024. Online: <https://crsreports.congress.gov/product/pdf/IF/IF11836>.
- [35] Elliott, C., "The DARPA Quantum Network," in *Quantum Communications and Cryptography*, 2005. DOI: 10.48550/arXiv.quant-ph/0412029.
- [36] Liu, H.-Y., et al., "Optical-Relayed Entanglement Distribution Using Drones as Mobile Nodes," *Physics Review Letters*, Vol. 126, 2 2021. DOI: 10.1103/PhysRevLett.126.020503.
- [37] Chen, Y.-A., et al., "An Integrated Space-to-Ground Quantum Communication Network Over 4,600 Kilometres," *Nature*, Vol. 589, No. 7841, 2021. DOI: 10.1038/s41586-020-03093-8.

10

Test & Evaluation

AI has unique evaluation requirements, exacerbated when the system learns during a mission. This chapter describes how to test and evaluate performance so that customers and end users will trust the system, even when it encounters unexpected conditions. This chapter answers the question of how to test something that changes.

The goal of evaluation is to determine how much trust to place in the AI. Simply put, *is the model useful, and making valid decisions in the mission context?* Should you trust the AI enough to report observations? To fly the platform? To target an adversary? To learn rules of engagement?

*Trust is a function of risk:
The more risk for the trustor, and authority granted to the AI,
the greater the validation and assurance requirements.*

Trust must start with safety and security [1], and the degree of trust depends on the amount of risk the trustor can tolerate, as illustrated in Figure 10.1. The degree of trust determines the amount of authority they will grant to the AI.

The concern from a test and evaluation (T&E) standpoint is that cognitive systems may make changes to their own decision-making processes without human



Figure 10.1 Trust depends on safety, security, and the acceptable level of risk.

supervision, and therefore potentially do things that are inappropriate or dangerous. For example, in 2016, Microsoft's AI chatbot interacted with the environment (Twitter) to learn racist and misogynistic behaviors [2]. Microsoft initially used human supervision to attempt to "fix" the problem, but ultimately removed the bot.

Evaluating a cognitive system has many similarities to traditional validation and verification approaches. Tests include normal cases, edge cases, stress conditions, and adversarial situations. A *learning* system, however, requires adaptation to the T&E process. Notably, this process ensures and evaluates model robustness, particularly with respect to changes in the data. All of the steps of data management, model development, and metalearning affect the quality of the final system.

T&E must answer the question of *deployability*: Is the system ready for deployment to end users? Consider the analogy of obtaining a driver's license. You, as a cognitive system with a carbon-based brain, undergo a learning process under supervision. You take a written exam and a behind-the-wheel test. These tests do not guarantee that you will make no mistakes, or that you will have no accidents. Instead, they assess whether you are ready to be allowed to drive without supervision. Perhaps your region has a graduated license, for example limiting the number of passengers or types of roads. Your community expects you to continue learning as you gain experience. As you drive in various environments, you learn to assess other drivers and handle surprises. If you do have an accident, you will likely learn more in that moment than you would in weeks of safe driving. T&E for silicon-based brains needs to take the same approach. Are we ready to release a minimum viable product? Do we expect it to learn from feedback from the environment and end users? Will it improve itself with minimal outside supervision?

Many international organizations are developing guidelines on AI assurance. Table 10.1 lists a few that—when used within the iterative methods of Section 10.2—have approaches that can be applied to the testing of cognitive systems.

This chapter covers key concepts for testing cognitive systems, whether or not they are for EW. Although the chapter focuses more on *T&E to test AI*, rather than *AI to improve T&E*, there are many examples of using AI for test embedded in the sections below. Section 10.1 outlines the challenges that the T&E community must address when testing cognitive systems. The key steps to test and evaluation are:

- Validate the learning process, rather than the static final model (Section 10.2).
- Determine metrics by which to evaluate the learning system (Section 10.3).
- Understand the range of operational effectiveness (Section 10.4).
- Use a closed-loop mixed-fidelity testing framework (Section 10.5).
- Develop stimulus-response threat models (Section 10.6).
- Select the most effective set of experiments (Section 10.7).

- Evaluate models with appropriate mathematical concepts (Section 10.8).
- Use ablation tests to determine how much a priori data is required to trust the system (Section 10.9).
- Formally and empirically validate results (Section 10.10).

Section 10.11 concludes the chapter with an example of a complete implementation framework.

10.1 Paradigm Shifts

Cognitive capabilities require multiple paradigm shifts from legacy T&E approaches that will change the way we test our systems. For definition purposes, we view the system-under-test (SUT) as one of multiple agents in the T&E environment. Agents are independent components that process events and select actions in the environment, and may include other team members or adversaries, and can be automated or human. A SUT might be a single agent with independent goals, or multiple agents working towards the same goals, such as a drone swarm.

The first major challenge that T&E processes must address is *complexity*. Cognitive systems generally handle scenarios of much greater complexity than adaptive systems. Scenario variation cannot be evaluated comprehensively; we can no longer assume that we can perform a complete design-of-experiments for all interactions of metrics.

Objectives are harder to articulate: There are multiple stakeholders, multiple objectives, and complex trade-offs to manage. Moreover, a given system might be used operationally under different conditions, and therefore the importance of specific objectives may change according to setting.

Table 10.1
Resources for Assuring AI Systems

Source	Report Title
Cohen & Howe	How Evaluation Guides AI Research [3] (See also Callout 11.1)
Batareseh & Freeman	AI Assurance [4]
EASA	Concepts of Design Assurance for Neural Networks [5, 6] First Usable Guidance for Level 1 ML Algorithms [7] Formal Methods Use for Learning Assurance [8]
DEEL	Machine Learning in Certified Systems [9]
NIST	AI Risk Management Framework [10]
DTEA	Developmental Test & Evaluation of Artificial Intelligence Enabled Systems [11]

The core cognitive capability is in the software, not the hardware. Software systems are increasingly complex and can be rapidly updated. Therefore, determining the root cause for success or failure is more challenging: it could be in the data, the training process, the model, the algorithm, or the implementation.

In the CogEW environment, there is a blurring of functionality across the full EMS (radio, radar, infrared, visible, and ultraviolet), and is extending to include acoustic and cyber capabilities. With up to 1000 controllables per platform, in a large heterogeneous team with different sensors and different objectives, the number of experiments under traditional approaches is untenable.

The second major challenge that T&E must support is *interactions with the environment*. Cognitive systems interact with the environment. Even though we can control all agents in the test except the SUT, each action the SUT takes may require a different closed-loop response from the test resources. Moreover, if the test scenario contains multiple cognitive agents, human or unmanned, there can be an infinite number of interactions between the agents and the SUT.

We can also no longer assume that test assets have fixed, known behaviors. Therefore, to ensure that the test has validity in operational settings, the test scenario (and agents) must be sufficiently realistic and complex, without necessarily being “validated.” For example, adversarial agents may choose to hide actions or increase variability of their responses, as outlined in Table 1.2. Software-defined systems can rapidly update procedures to incorporate behaviors that are ephemeral, complex, and depend on the SUT’s activity.

In EW, scenario control has always assumed that the actions all happen in the EMS, and that the asset(s) will follow a preplanned path(s). In a CogEW system, however, this assumption may not be true. For example, the decision maker might choose to reroute the platform to get a better signal or to break the lock from a targeting radar.

The third, and most important, paradigm shift is due to the fact that cognitive systems *learn after deployment*. Cognitive systems must learn how to respond to surprises outside of the original training data and testing scenarios. The system may never be the same twice, not during a multiday test event, nor after the system has been deployed. We would not expect a human being to perform identically every day, especially when presented with the same scenario; we should therefore not expect identical performance from a cognitive system. The philosophical shift occurs because testing affects the learning cognitive system—the system after testing is not the same system the developer delivered—and it may be difficult or even impossible to generate test procedures that are deterministic and repeatable.

Because cognitive systems learn from experience, all experience must be collected and available to help train the system, from early concept design, through developmental test, operational test, and post-deployment operational settings.

Finally, testing a cognitive system will necessarily influence future training, both human and system. The cognitive system interacts and learns from the environment, which includes human users; likewise, humans learn how to interact with the system. As an example, consider how “prompt engineering” is changing the way we interact with ChatGPT. The relationship between (human) training and T&E will need to change as we build more systems with cognitive human-machine teaming.

10.2 Validating the Learning Process

Because the deployed system will continue learning after it has been deployed, we can no longer assume a “final exam” approach to testing, where the test validates a final learned model. Figure 3.12 describes the process generally used to train, test, and evaluate an ML model. The issue with the common process is that it only evaluates the *learned model* (solid line coming out of the design phase) and not the online steps that *update the model* (dotted line). Instead, we must validate both. We must validate the entire learning process, to include requirements, data, and model training [5]. Validating the learning process means that T&E must validate how the system learns, so that when it is learning without human supervision, we can trust that it will do the right thing.

The test must examine how well the SUT learns over time, against surprises, and for different operational settings. Retraining of the system becomes a key step in the test lifecycle, and the test must ensure that the SUT incorporates new information appropriately. Notably, test activity must evaluate how the number of samples needed for the system to accurately characterize and responds to the surprise. For CogEW systems or cybersecurity systems that must learn from only one or two observations, this evaluation step is particularly important. A critical step is to ensure that the model does not generate unintended effects [12]. Unintended effects in a learning-enabled system usually emerge from the data used to train the system. In EW, novel RF environments fall outside the domain-of-expertise of the model, and adversaries may manipulate the data. It is crucial to ensure that the system accomplish the intended objectives, without unacceptable behavior.

As part of this process, we must understand and control the effects of changes to the data, the models, and the training process, which means that we must adopt *continuous test* processes and feedback loops. The software-driven nature of cognitive systems means they can change and upgrade capabilities quickly, which requires smaller, more rapid test evolutions during the system lifecycle [13].

Since the learning system changes during test, extreme care must be taken to avoid introducing errors or discrepancies by using a different test strategy, different data fidelity, or scenario variations distinct from the developer intent. This raises the question: When a SUT performs less well in a T&E environment than was predicted in the development environment, is it because of deficiencies in the cognitive algorithms, or in the T&E environment itself? Different data fidelities not only creates problems with scenario evaluation (in the sense of apples-to-apples comparison), but can also produce evaluation errors. For example, models created from dirty data may have very different or incorrect behaviors across layers. Continuous T&E from development through operations helps mitigate this concern.

10.2.1 Iterative Design and Test

The design loop of Figure 10.2 ensures data quality, model accuracy, and model generalizability. The design phase is iterative, ensuring that all changes are verified: changes to the data formats, choices in model structure, and adaptations to hyperparameters. The first step in this loop is to collect a dataset. It doesn't have to be a great dataset. In fact, trying to collect a perfect dataset will waste a lot of time. Instead, make the initial dataset good enough that you can build a model.

The second step is to build a few models; experiment with different training approaches and hyperparameters. The mistakes these models make will highlight the failures in the data, such as missing features, too much missing data, or class bias. Improve the data, and build new models. Iterate at this level few times, until the model(s) seem to be performing reasonably well. Now look at what happens when data is sparse: Instead of testing with 80/20 train/test split, use a 20/80 split, 10/90, and 1/99.

The final step in the loop is to test the model(s) for generalizability: How well do they handle surprise? Callout 10.1 lists a few methods to evaluate how well the models handle surprise. If they do not perform well, you may need to change the data representation, remove or add features, transform the model, or change the training methods. Section 10.9 describes a process for comprehensively testing out-of-domain samples and determining how much a priori data is required. Iterating through these steps will help developers understand the domain better, understand factors that impact learning, and understand what matters to the end user.

10.2.2 Learning Assurance Process

Learning assurance is a *quality assurance* or *quality control* process for any ML-enabled system. Learning assurance aims to provide the (cognitive) EW stakeholders with the same trust levels that existing EW systems boast, having undergone the traditional “V”-shaped development assurance process.

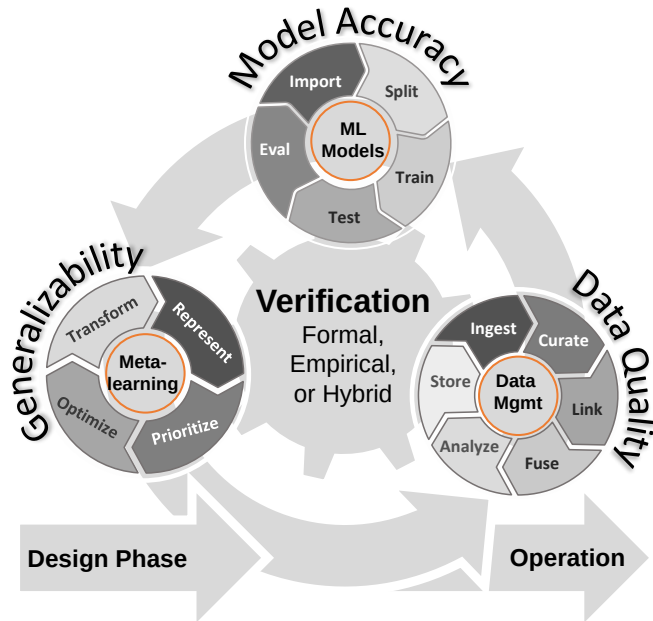


Figure 10.2 During the design phase, principles of verification apply to the iterative process of data management, model development, and metalearning. Good design verification leads to performance assurance during the operational phase.

To bring these design ideas into a formal certification methodology, the European Aviation Safety Agency (EASA) and Daedalean AG adapted the traditional “V”-shaped process to handle cognitive systems. The learning assurance “W” of Figure 10.3 incorporates steps for learning systems: data lifecycle management, and model training, and verification [5]:

- *Data management* identifies candidate datasets, data preparation methods, data quality requirements, and validation objectives with respect to the product/system requirements and concept-of-operations.
- *Learning process management* drives the selection and validation of elements such as the training algorithm, the initialization strategy, and hyperparameters. It also considers the hardware and software frameworks, and chooses the evaluation metrics. Section 10.3 describes evaluation metrics.

Callout 10.1 T&E must characterize the system's ability to handle surprise.

There are multiple ways to validate a CogEW system without developing cognitive threats. The key is to create environments that the system has not seen before the test. Some methods, in increasing complexity, include:

- Change parameters of a known emitter, such as modulation, pulse repetition interval, or center frequency.
- Augment the data with noise or channel effects (Section 8.3.3).
- Reserve a set of environments; for example, train on RF environments before the development of active electronically scanned array (AESA) emitters, and test on AESA radars.
- Form a CogEW red team that develops a set of formal adversarial tests.
- Analyze the system for robustness characteristics [14, 15].
- Comprehensively evaluate all combinations of possible a priori data to empirically show that the system can learn to handle any surprise (Section 10.9).
- Design a fully cognitive adversarial threat. GANs (Sections 3.1.2 and 8.3.3) can help automatically identify vulnerabilities.

Note that a cognitive threat is *not* required to test a cognitive SUT. In fact, in test settings the evaluator should be able to control the exact behavior of the test assets, and cognitive threats may not be fully controllable.

- *Model training* trains the model and evaluates model parameters using the validation dataset.
- *Learning process verification* evaluates the trained model against the test dataset, using the measurements identified for evaluation.
- *Model implementation* transforms the model into an executable model that can run on a target hardware. All optimizations and modifications for the embedded hard real-time environment must be validated.
- *Inference model verification* ensures that the embedded model continues to meet expected requirements.
- *Data verification* ensures that data assumptions have not changed.

If performance metrics require the system to learn without human supervision during an engagement on-board the platform, then these steps will help demonstrate the required capabilities.

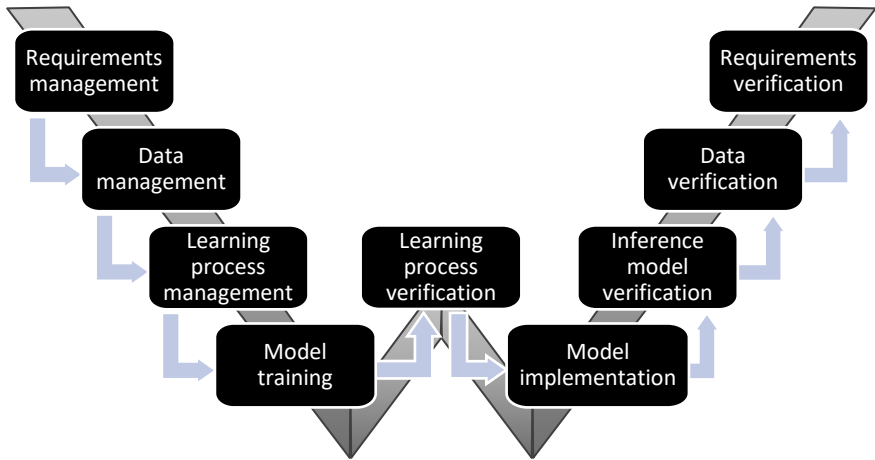


Figure 10.3 T&E for a learning system evaluates requirements, data, training process, and implementation in the embedded system. (Used with permission from [5].)

10.3 Evaluate Learning Goals

The most important missing concept when evaluating a cognitive system is to look at the way it learns. A teacher in a school evaluates children according to their abilities; some children are better at math, some at language, and some at sports. The students better at math might be asked to participate in math competitions; students stronger in language might participate in debate competitions; and athletic students might participate in competitive sports. The same should hold true for AI approaches and ML models: A model that is good at learning from a single example should not be expected to build complex lifelong patterns, and vice versa.

Traditionally, T&E approaches evaluate performance in a static way, giving one final score that combines the results equally across the scenario. For example, we evaluate most ML approaches using concepts such as F1 score (combination of precision and recall) or root-mean-squared error, which count the number or magnitude of mistakes, with equal weight given to all test examples.

This static approach does not consider the fact that a cognitive SUT learns and changes the way it makes decisions over time, and therefore the results at the start of the test will not be the same as those at the end. We must therefore define performance objectives in terms of their learning goals, such as how quickly the SUT can accurately handle a surprise, how much a priori training data is needed

to accomplish a certain learning goal, and transferability of the model(s) to new systems or environments. Callout 10.2 lists examples of metrics to consider when evaluating a self-modifying cognitive system.

10.4 Determine Range of Operational Effectiveness

Evaluation metrics include *measures of effectiveness (MOEs)*, which evaluate whether the system meets end-user goals (what the user wants), and *measures of performance (MOPs)*, which evaluate whether the system is performing as designed (what the developer created). MOEs and MOPs rarely capture necessary metrics for self-modifying systems.

A key concept often missing is that of *measures of suitability (MOSs)*, which determine whether a given component is suitable at a given time for a given objective. Because risk tolerance and mission context can change daily, a given MOE or MOP may be suitable at one point in time in one location for one mission, but not if any of those three factors change.

The reason we often do not consider MOSs is because many T&E settings use a “pass/fail” concept for determining whether a SUT is ready to be deployed. Given the complexity of modern software- and AI-defined systems, we need to move from this pass/fail approach to a model where we understand the limits and trade-offs of algorithm or system capability. We need to perform tests over a range of environmental settings to understand when the SUT achieves excellent, good, satisfactory, poor, or unacceptable performance.

Each contextual axis from the experimental design can become a filter or evaluation axis in a hypercube for presentation, for example as a receiver-operating-characteristic (ROC) curve. These MOSs allow operators to evaluate the readiness of the SUT for a specific operational environment. For example, an algorithm might perform less-and-less well as scenario complexity increases; it is the MOS that determines whether the algorithm meets the operational need in a particular setting. The end users may choose to use different algorithms for the same nominal task, depending on the context it will be used. Table 10.2 presents a few examples of where learning goals should be considered.

In CogEW, the number of emitters in the environment are an indicator of scenario complexity. An evaluation could show number of emitters vs desired performance. These trade-off analyses would allow users to select the right algorithm for the right context. For example, one algorithm might perform signal separation (aka deinterleaving) more effectively for aerial settings while another works better on ground-based systems.

Callout 10.2 Evaluation must focus on system deployability concerns such as learning goals, computation effort, data requirements, and usability.

- *Accuracy*: Successful characterization of the environment, behaviors, and causal event patterns, and high Adequacy of strategies; no unacceptable errors.
- *Data requirements*: Amount, type, completeness, correctness; see also Table 8.1.
- *Robustness / Stability*: Brittleness and sensitivity to variation of input values.
- *Computation effort*: Time to learn a model, time to make an inference, sample efficiency, memory usage.
- *Adaptability*: Time and number of examples required to update a model to novel conditions.
- *Generalizability*: Accuracy outside of the domain-of-expertise.
- *Transferability*: Effectiveness of model in slightly different contexts, for example trained on synthetic data and tested on real data.
- *Scalability*: Number of emitters, number of strategies, size of training data.
- *Complexity*: Mission/threat complexity.
- *Cumulative gains*: Impact of performance over time/experience, or in conjunction with additional AI/ML components.
- *Performance over time*: System performance over time; concept drift; stakeholder trust and willingness to rely on system decisions.
- *Uncertainty / Confidence*: Ability to self-determine confidence on the estimate.
- *Usability and explainability*: Ease for a human to understand the results, both in-mission, and for forensics.
- *Collaboration*: Ability to share information across models and with other participants.
- *Fairness*: Ethical compliance and social impact considerations.
- *Safety*: Ease of providing performance guarantees.
- *Security*: Resilience to attack.
- *Portability*: Hardware and operating system context; interoperability with legacy systems.

Table 10.2
Evaluation Metrics Differ by Context

Evaluation Metric	Context	Example Learning Goals
Measures of Suitability	Mission	Uncertainty tolerance; Generalizability
Measures of Effectiveness	Task	Scalability; Data needs
Measures of Performance	System	Accuracy; Learning rate; Computation time

Finally, in a self-modifying cognitive system, we need to establish which behaviors are *unacceptable* so that we can delineate the bounds of emergent activity. This process is the only reliable way to ensure that the system won’t overoptimize to the loophole in the optimization function, finding an unacceptable way to accomplish the specified objective.

10.5 Mixed-Fidelity Closed-Loop Testing

The test framework for a CogEW system requires several key capabilities. It must be closed-loop and support multiple vertically integrated resolutions. Ideally, it supports mixed-fidelity testing and can incorporate third-party components.

10.5.1 Closed-Loop Test Framework

The first requirement is a *closed-loop* test infrastructure. ML systems commonly use static datasets to learn how to classify objects. In EW, this approach is insufficient because it doesn’t show how the system handles novel examples, responds to dynamic situations, or operates against adversaries. Analogous to a static RF dataset, a video of you provides a snapshot of your behavior while, for example, playing soccer or cooking dinner. It is likely that the video doesn’t capture both playing soccer *and* cooking dinner, let alone activity in work settings (protected) or at the doctor’s office (private). These are static recordings and not interactive.

To validate a cognitive decision engine, a closed-loop testing framework is essential.

The framework needs to provide full closed-loop testing to capture the interaction between the cognitive SUT and the test assets. Closed-loop testing is crucial due to the interactive nature of cognitive systems, where the SUT must respond correctly to stimuli, and the test framework must respond correctly to SUT actions. Closed-loop testing is increasingly vital for modern RF systems, including sensors and EW, as it enables evaluation against dynamic scenarios with adversaries and realistic RF effects. Adversarial EW systems are becoming more sophisticated, while sensors are adopting agile methods to counter these threats.

In comparison, traditional EW testing has been *open-loop*, in that it can generate data for complex waveforms in complex propagation environments. Good software simulators and hardware signal generators generate extremely realistic data that capture enough of the fine-grained channel effects that are sufficient even for building SEI models. However, simulating systems in an open-loop lab setting often produces results that differ significantly from those observed in open-air ranges or real-world settings; open-loop approaches still do not support interactive testing that cognitive systems require. Lab-based closed-loop testing can achieve 80-90% accuracy compared to real-world conditions, making it a highly effective and reliable alternative. By incorporating realistic signals and physics-based accuracy in a closed-loop ecosystem, designers can identify and address issues earlier, train cognitive systems in near-real conditions, and quickly adapt to emerging scenarios.

10.5.2 Multifidelity Vertically Integrated Test Framework

The second requirement is a *multiresolution* test infrastructure. A complete T&E solution requires testing in multiple settings—low-fidelity modeling and simulation (M&S), high-fidelity M&S, hardware-in-the-loop (HWIL) labs, and live testing “in the wild,” as illustrated in Figure 10.4.

Each of these layers has its own strengths and weakness. For example, low-fidelity event-based simulations can run much faster than realtime, while live tests are the only setting with complete environmental realism. Software models support more comprehensive testing, wherein systems can be easily reconfigured, and many of the operational parameters can be explored. Event-based simulation tools are

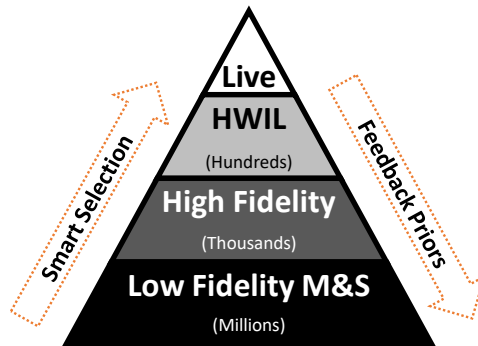


Figure 10.4 A complete CogEW testing solution requires closed-loop testing at all layers including low-fidelity M&S, high-fidelity M&S, HWIL emulated systems, and live tests.

scalable to very large, complex scenarios because they can operate faster than real time. Hardware testing in controlled environments is more realistic, but limited to (exactly) real time. Meanwhile, live tests are limited by scope, time, and cost.

For example, if real raw I/Q data is unavailable, RF signal classifiers can be initially trained/verified on synthetic I/Q data (e.g., generated via MATLAB). Although synthetic data introduces unwanted artifacts and lack realism (e.g., I/Q imbalance or phase noise of a given OFDM system), it can also provide data diversity and capture edge cases (Section 8.3.2).

In any CogEW project intended for deployment, all of these layers are crucial. Perform many millions of tests in a low-fidelity event-based M&S environment, thousands at high-fidelity M&S, hundreds with HWIL emulation, and tens of tests in a live environment.

*No one test approach can test everything;
no one test approach should test everything.*

The framework must decide which experiments to perform at higher resolutions, and ensure that data and lessons learned at higher fidelity are incorporated into lower fidelity tests. Scenarios should be modeled only at the level of fidelity necessary to assess the effects of actions.

The framework must be *vertically integrated*, in that results must be consistent across layers: Conclusions drawn at a higher fidelity must not contradict results at a lower fidelity. Scenario descriptions and results must be recorded using a consistent format across fidelities. There must be full traceability across layers. Results must be appropriately tagged with the fidelity of the original setting, and any lessons learned from higher-fidelity environments should be transferred to lower-fidelity environments. This process improves the quality of lower-fidelity models, and also mitigate problems caused by models that were created by poorly labeled data. When results from a lab-based closed-loop test are consistent with results in a real-world setting, the approach offers the practical benefits of reducing costs and saving time. Open-air range tests are not only expensive and limited in scope, but also difficult to schedule, given the scarcity of these high-demand facilities.

In contrast to frequent and dedicated Chinese or Russian experimentation, [USA] DoD does not pursue extensive EW or EMS experimentation due to concerns about operational security and access to ranges and other appropriate instrumented facilities... An increased reliance on virtual and constructive systems for EW and EMSO training could improve DoD's ability to experiment, as could a new approach to R&D that combines technical and operational innovation.

— *Center for Strategic and Budgetary Assessments, 2019 [16]*

10.5.3 Mixed-Fidelity Test Framework

Extending the multifidelity framework, *mixed-fidelity* testing supports more scale through greater flexibility. In the long term, the traditional T&E approach of testing the full system in independent settings will not prove practical given the complexity of cognitive systems. Mixed-fidelity T&E maximizes simulation speed while minimizing accuracy loss. A broker-based architecture, as described in Section 9.1, supports swapping of models and modules at different levels of fidelity.

In the CogEW setting, a mixed-resolution framework for air traffic control might choose a high-fidelity air traffic control radar, with low-fidelity weather radar and aircraft maneuvers, and event-based communications network to distribute Automatic Dependent Surveillance-Broadcast (ADS-B) beacons. A counter-drone system for the same airport might reduce fidelity on the air traffic control radar to enable high-fidelity simulation of drone communications and jamming.

10.5.4 Integrating Third-Party Tools for System Analysis

The final goal for a comprehensive T&E ecosystem is that the framework should aggregate models for multiple systems and software entities that are built and tested by multiple organizations. A key requirement of such a framework is the ability to seamlessly integrate third-party tools and components. This capability ensures flexibility in testing and evaluation, allowing organizations to leverage specialized tools, models, or algorithms developed externally. By supporting heterogeneous integration, the framework enables collaboration across teams and organizations, fostering innovation and reducing development redundancy.

The framework must provide robust mechanisms for tracking and evaluating performance at multiple levels. It should report mission-level success criteria, ensuring the overall goals of the system are met, while also capturing internal component-level performance metrics to identify areas for improvement. This dual-layered approach ensures that both high-level objectives and detailed subsystem performance are evaluated, enabling iterative development and refinement.

10.6 Behavior-Based Models with Closed-Loop Effects

The challenge with all closed-loop testing is that the test framework needs a *model* of how the waveform and platform would respond to various stimuli. Without a model capturing how systems respond, closed-loop testing is impossible. Imagine a boxing match, where you can encounter many different opponents who can do many different things; you choose the best place to punch them, *but they*

never punch you back. An opponent might punch you back, or duck, or tell you to punch again, or grab your wrist and use your own momentum to throw you over their shoulder “judo-style.” Key requirements for closed-loop models include: multilevel stimulus/response effects, high-level behaviors, fine-grained transceiver characteristics, and anticipated future capabilities.

10.6.1 Stimulus-Response Effects

Models must generate sufficiently realistic responses in the face of jamming or other stimuli. The framework uses these stimulus-response effects to support closed-loop testing. For example, a model of a threat radar must accurately capture how it handles situations when the SUT takes action to break the radar lock, such as jamming or maneuvering. We can estimate jamming effects using EW engineering expertise and available data describing threat equipment [17]. For example, a radar suffering interference would take anti-jamming measures such as changing frequencies or modulation. As an added benefit, these stimulus-response effects can also be used as “tells” by the EW BDA system (see also Figure 2.6).

Threat models must describe these effects in a concise format, at multiple levels of fidelity, to ensure consistent behavior in the test framework. More complex effects should be captured at higher levels of fidelity and provided to more abstract levels in which mission-level engagements occur. This multifidelity model abstracts the higher fidelity interactions in large scale exercises, where full fidelity is not computationally tractable within the required time frame.

10.6.2 High-Level Behaviors

Traditional EW test assets have fully deterministic, scripted behaviors that do not sufficiently challenge the capabilities of cognitive systems. A cognitive SUT can therefore adapt responses to the validated behaviors, possibly overfitting responses to the specific environment rather than generalizing to a more complex variation. As an example, if we test automated vehicles in an environment where every other vehicle performs exactly as expected, the SUT will not learn how to handle edge cases or surprises. Overfitting leads to reward hacking, where the decision maker finds unacceptable solutions by exploiting flaws in the objective function or scripted models (Section 5.3).

Likewise, capturing only the RF behaviors in a model obscures the variation caused by humans in the system. As cognitive entities, humans can utilize the platform in unexpected ways. For example, during the 9/11 terrorist attacks, there were approximately 800 civilian aircraft in the air over Washington, D.C. These all had identical civilian ADS-B beacons; the only feature indicating that one of these was a threat was a human-level behavior. A behavior-based approach to

describing test assets supports multiple settings and multiple use cases, which allow the cognitive system to develop robust models that are more resilient to attack.

Behavioral models should have a “dial” indicating how capable the test asset is. Table 1.2 shows levels of cognition that can be used as a gauge of SUT capability. The test report should indicate SUT effectiveness against each level as outlined in Section 10.4. Ideally, cognitive systems should be tested against adversarial systems with commensurate complexity and cognition. That is, the adversarial system model must match the cognitive complexity of the intended friendly EW system test case. This simply means, for example, a CogEW system developed to target level 4 threats should perform extremely well against a level 1 threat, sufficiently well against a level 4 threat, and likely not accomplish much against a level 8 threat. Full adversarial models, such as those developed with GANs (Section 3.1.2), are an effective way to develop adversarial examples and training models.

10.6.3 Detailed Transceiver Characteristics

One of the important challenges in CogEW testing is that many models are parametric, for example using radar pulse description words (PDWs) to define the system. This level of detail is not sufficient to capture many of the important characteristics of capable systems. For example, SEI models leverage features that are ambiguous in PDWs. Models rarely capture enough of the nonlinearities of hardware to begin developing SEI models, such as phase noise, filter distortions, and clock offsets. Models must therefore capture all features of the antenna array, RF electronics, and digital signal processing chain. Moreover, both transmitter and receiver components contribute to the quality of the model. To build receiver-independent SEI models, we also need detailed models of the receiver processing chain.

10.6.4 Possible Future Capabilities

In an adversarial setting, threat-validated test models and assets do not capture enough diversity in threat representations to highlight potential future capabilities of reprogrammable threats. There needs to be a clear structure to define “possible futures,” based on understanding of technology maturity. This structure is like a hurricane path that is represented by a cone of uncertainty. The current state is relatively well-understood, with a narrow cone and greater confidence in the capability. As the forecast extends further into the future, uncertainty increases and the cone widens.

A CogEW *red team* can help develop the information to populate these models. A red team’s goal is to build adversarial models that identify system vulnerabilities. By understanding current and emerging technological capabilities, the red team can develop models that capture a range of possible futures.

10.7 Smart Experimental Design

The number of scenarios in a complex cognitive setting quickly reaches infinity. Each factor—a cognitive SUT, cognitive test assets, and scenarios—are independently high-dimensional. Manual and traditional test designs cannot comprehensively evaluate the combinatorial explosion across contextual axes. Even randomized sampling approaches such as uncertainty estimation and Monte Carlo methods cannot assure statistically relevant results. Moreover, the required sample size to evaluate a SUT is typically not known a priori for any simulation model. We require smart sampling strategies to select efficient design points and rapidly explore the limits of complex systems [18, 19].

Since system requirements documents are often poorly specified with respect to learning, it is crucial to determine early which scenarios to test. As a starting point, identify which of the three axes illustrated in Figure 1.4 and Callout 10.2 are relevant to system performance—and customer needs—and test at multiple points along each axis.

Doing this process comprehensively is known as *design of experiments*, a pattern of systematic sampling of experiments that determines cause and effect relationships between inputs and outputs. Traditional experimental design is not adaptive; that is, decisions about which experiments to run are made and fixed in advance. It is impossible to test all ∞ configurations by ∞ scenarios. The test framework must therefore use modern experimental design approaches to manage the computational complexity caused by the combinatorial explosion. *Adaptive sampling* efficiently samples the high-dimensional decision space by selectively choosing experiments based on the previous experiments. Adaptive sampling can use heuristics about the domain, such as physics, progression of the engagement, and collection or denial of information, to focus test requirements. Cast as a sequential decision-making problem, the goal is to maximize the information gain produced by each experiment. The key aspects of adaptive sampling in complex environments are:

- *Iterative process:* Experiments are chosen based on information gathered from previous experiments.
- *Active learning:* The sampling strategy manages an explore-exploit trade-off to identify new patterns. (See Section 7.3.2 for more details.)
- *Model-based approach:* Statistical models and domain heuristics help guide the sampling.

Common approaches to adaptive sampling include multiarmed bandit (Section 7.3.3.2) [20, 21], Bayesian optimization [22–24], Latin hypercubes [25, 26],

and Kriging metamodeling [27–29]. The randomized search algorithms in Section 5.1.2 are also appropriate tools. For example, one could encode the scenario parameters as a chromosome for a genetic algorithm, and the function used to evaluate each chromosome would be a call to the M&S tool. Finally, some of the techniques to ensure diversity in a dataset are also relevant for selecting experiments (Section 8.3.2).

Products exist that combine these capabilities for complex systems where experiments are costly, hard-to-perform, or time-consuming, (e.g., [30, 31]). The combination of automated testing, distributed testing, statistics, and active learning enables efficient exploration and optimization of the design space and rapid characterization of key results.

10.8 Computing Accuracy and Adequacy

EW systems use both regression and classification learning approaches.

Regression models predict numerical values, i.e., $y = f(x)$ for $y \in \mathbb{R}$. The accuracy of a regression model is evaluated using normalized root mean square error (Section 10.8.1).

Classification models label discrete classes, i.e., $y = f(x)$ and $y \in \mathcal{S}$ for a discrete set of classes $\mathcal{S}$. Classification models are evaluated using confusion matrices and related statistics (Section 10.8.2).

In EW, we also capture cases when multiple EP/EA strategies yield different performance for different RF environments. For example, multiple EP strategies might protect against one jammer type with different efficacy. These models are evaluated with a modified confusion matrix (Section 10.8.3) that blends the idea of a confusion matrix with cost of errors.

Burnell et al. [32] highlights how current AI evaluation metrics are insufficient. Aggregate metrics obfuscate key information about where systems tend to succeed or fail, and introduce unwarranted assumptions about the evaluation. They state that reporting of results should be granular, and instance-by-instance results should be made available.

10.8.1 Regression and Normalized RMSE

Regression algorithms are usually evaluated with *root mean square error* (RMSE), namely the square root of the average squared difference between the observed values $\hat{\mathbf{y}}$ and the estimated values $\mathbf{y}$. *Normalized RMSE* (*nRMSE*) allows results from different data distributions to be compared fairly. nRMSE is the RMSE divided by the standard deviation σ of the values. For v instances with a mean value of $\mu = \frac{1}{v}(\sum_{i=1}^v \hat{y}_i)$:

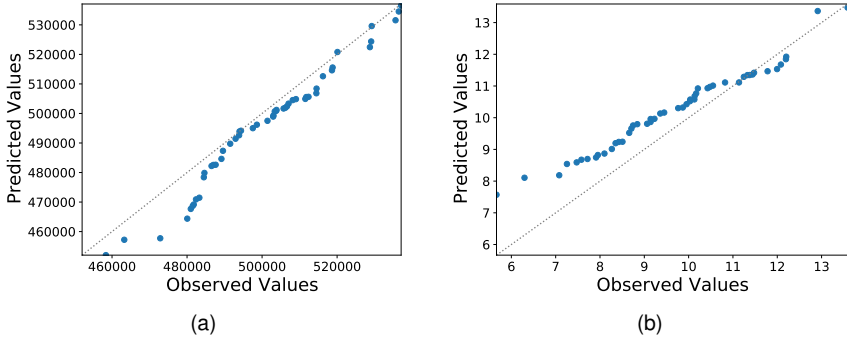


Figure 10.5 To compare models that predict different ranges, we must normalize the RMSE by the standard deviation. (a) Metric m_1 ranges from 458365 to 537068, with $\mu = 501041$ and $\sigma = 18061$; RMSE is 6328, yielding an nRMSE of 0.350. (b) Metric m_2 ranges from 5.7 to 13.6, with $\mu = 9.7$ and $\sigma = 1.7$; RMSE is 0.76, yielding an nRMSE of 0.442. (*nRMSE=0.0 occurs when all predictions are identical to the true values, and lie along the diagonal grey line.*)

$$\text{RMSE} = \sqrt{\frac{1}{v} \sum_{i=1}^v (\hat{y}_i - y_i)^2} \quad \text{nRMSE} = \frac{\text{RMSE}}{\sigma} = \frac{\sqrt{\frac{1}{v} \sum_{i=1}^v (\hat{y}_i - y_i)^2}}{\sqrt{\frac{1}{v} \sum_{i=1}^v (\hat{y}_i - \mu)^2}}$$

The standard deviation represents the performance of a learner that uses the mean as the prediction for all instances. Our goal is to achieve as low an nRMSE as possible; 0.0 indicates that every instance is predicted with no error, while > 1.0 indicates that no “fancy” model is needed because the mean value is better. Consider for example, using one model on multiple frequency ranges; nRMSE demonstrates clearly whether it performs equally well on each of the ranges even though their raw values are different. Figure 10.5 illustrates a simple example allowing us to compare the models for two metrics. The goal is to choose one model that accurately characterizes all of the underlying data patterns using a function that shows at-a-glance whether the results are reasonable.

A high nRMSE tells us that the model struggled to capture the performance surface. This result can help the system engineer understand and identify systematic problems, such as missing observables, faulty sensors, unreliable metrics, and temporal latencies.

Table 10.3
Confusion Matrix Showing How a Classification
Algorithm Identifies Objects

	Predicted Positive	Predicted Negative
Observed Positive	True positive	False negative
Observed Negative	False positive	True negative

Table 10.4
Notional Results for the Acoustic Signal Classes
Biological and Man-Made with 94.5% Accuracy

	Biological	Man-made
Biological	90	10
Man-made	1	99

10.8.2 Classification and Confusion Matrices

Classification algorithms are usually evaluated using accuracy values computed from *confusion matrices* using the structure illustrated in Table 10.3. The rows correspond to the known *observed* classes, while the columns correspond to the classes *predicted* by the model. A true positive or a true negative indicates that the model was correct, while a false positive or a false negative indicates that the model was wrong. Table 10.4 shows some notional results for the acoustic signal classes biological and man-made; this notional classifier yields 94.5% accuracy, with a propensity to label objects as man-made. Each cell in this confusion matrix counts the number of examples of each type, for two classes.

Depending on the problem, false negatives and false positives may have different consequences. For example, if an explosives detector in airport baggage scanner identifies a nonexplosive as an explosive, then the false positive creates some additional screening time. If the scanner misses an explosive, however, then the false negative could cost lives.

10.8.2.1 Accuracy, Precision, Recall, and Class Imbalance

The *accuracy* of the confusion matrix is the diagonal counts divided by the total number of test instances. For a confusion matrix $\mathcal{M}$ that counts labels across x classes, $\mathcal{M} \in \mathbb{J}^{x \times x}$:

$$\text{Accuracy} = \frac{\sum_{i=1}^x \mathcal{M}_{i,i}}{\sum_{i=1}^x \sum_{j=1}^x \mathcal{M}_{i,j}}$$

Accuracy can be a misleading metric for imbalanced datasets, for example when there are many fewer man-made objects than biological objects. *Balanced accuracy* normalizes predictions by the number of examples in each class:

$$\text{BalancedAccuracy} = \frac{1}{x} \sum_{i=1}^x \left(\frac{\mathcal{M}_{i,i}}{\sum_{j=1}^x \mathcal{M}_{i,j}} \right)$$

Recall corresponds to the number of samples in a class that were correctly labeled, by computing the diagonal value against the sum of the row i :

$$\text{Recall}_i = \frac{\mathcal{M}_{i,i}}{\sum_{j=1}^x \mathcal{M}_{i,j}}$$

Precision corresponds to how tightly the model predicts a class, by computing the diagonal value against the sum of the column j :

$$\text{Precision}_j = \frac{\mathcal{M}_{j,j}}{\sum_{i=1}^x \mathcal{M}_{i,j}}$$

Two other metrics are also commonly used. *Overkill* measures how many false positives there are (FP/nSamples). *Escape* measures how many false negatives there are (FN/nSamples). Figure 10.6 illustrates precision and accuracy.

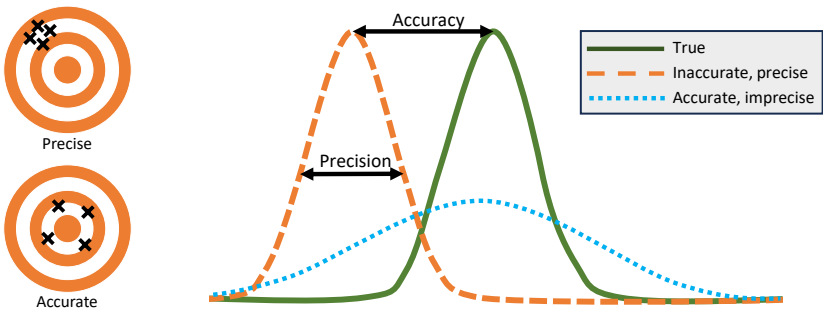


Figure 10.6 Precision and accuracy are two methods for evaluating classification algorithms.

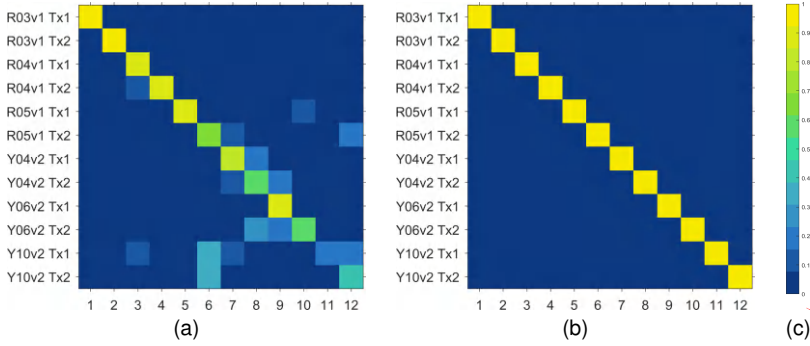


Figure 10.7 These confusion matrices represent the labels for specific emitter identification using two different classification algorithms [33]. (a) CNNs achieved 71.9% accuracy. (b) A multistage training algorithm achieved 98.7% accuracy. (c) Legend.

10.8.2.2 Multiple Classes

When there are more than two classes, we expect high match counts along the diagonal of the confusion matrix; off-diagonal results show misclassification errors. Rows and columns are usually sorted by some form of similarity, making it easier to see (and possibly ignore) misclassification errors between similar classes. For example, in a modulation classification evaluation, we would expect QAM16 and QAM64 to be adjacent, where misclassifications are unsurprising. Correctness is a function of the mission; these misclassifications may matter in certain contexts.

In the SEI example of Figure 10.7, each pair of rows/columns is two transmitters mounted on the same radio platform that share a power supply. Therefore, misclassifications within each “block” of four cells around the diagonal are “less” incorrect than misclassifications elsewhere.

10.8.2.3 Weighted Accuracy and Model Quality

When the reward for a classification differs by class, or the cost differs based on the type of error, we bias the accuracy evaluation using a reward matrix, $\mathcal{R} \in \mathbb{R}^{x \times x}$, and $0 \leq \mathcal{R}_{i,j} \leq 1$. In this case, we weight the confusion matrix $\mathcal{M}$ by the rewards $\mathcal{R}$, obtaining a weighted accuracy matrix $\mathcal{Q} \in \mathbb{R}^{x \times x}$; each cell i, j in $\mathcal{M}$ is weighted by its corresponding cell i, j in $\mathcal{R}$:

$$\mathcal{Q}_{i,j} = \mathcal{M}_{i,j} \mathcal{R}_{i,j}$$

Overall *quality* of the model is computed as a percentage of the total (label quality divided by total number of labels):

$$\text{Quality} = \frac{\sum_{i=1}^x \sum_{j=1}^x \mathcal{Q}_{i,j}}{\sum_{i=1}^x \sum_{j=1}^x \mathcal{M}_{i,j}}$$

Consider a notional set of rewards for six radar classes (two radars with three modes each) in Table 10.5. Diagonal cells capture the case when the model correctly labeled both the radar and its mode. It is partially correct to identify the correct radar (but the wrong mode), or the correct mode (but the wrong radar). Using these reward weightings $\mathcal{R}$ to weight the classification results $\mathcal{M}$ of Table 10.6, we obtain the quality matrix $\mathcal{Q}$ shown in Table 10.7. Overall quality is 90%.

10.8.3 Evaluating Strategy Performance

To evaluate the *effect* of the strategy on system performance, neither nRMSE nor confusion matrices capture an essential concept:

For a given RF environment, there may be multiple strategies that contribute to performance with a nonzero utility.

A *modified confusion matrix* measures performance of strategies against environments. Instead of the confusion matrix of environments by environments, we use environments by strategies. Each cell captures the performance of the strategy in that environment, recording “partial credit” for strategies with nonzero utility.

The modified confusion matrix is stored as a performance table $\mathcal{P}^{m_k} \in \mathbb{J}^{a \times b}$ for each metric m_k , where $\mathbb{J}$ is the set of positive integers, $a \ll \overline{o_n}$ is the number of environments, and b is the number of strategies $\Pi_{\forall c} v_c$. Each cell $\mathcal{P}_{i,j}^{m_k}$ corresponds to the observed performance of metric m_k using strategy $j = s_n$ in environment i associated with $\mathbf{o}_n$.[†] Section 2.1.1 describes how to compute an environment for collected observables. Table 10.8 shows the layout of this matrix.

The performance table $\mathcal{P}^{m_k}$ might be sparsely populated at the beginning of a mission, similar to Table 7.1. The table summarizes the data from a priori data, plus any experience collected in-mission. The data is a collection of examples, $\langle o, s, m \rangle$, while $\mathcal{P}^{m_k}$ is a table of size $a \times b$ for a given m_k . In theory, $\mathcal{P}^{m_k}$ could have a very large number of rows, but the test framework controls the number of environments a by setting the maximum number of environment clusters (Section 2.1.1).

[†]. It may be appropriate to maintain a list of the obtained performances, and then use a probability distribution or other statistical measure to determine goodness relative to optimal.

Table 10.5
Notional Radar Reward Matrix $\mathcal{R}$

	R1M1	R1M1	R1M3	R2M1	R2M2	R2M3
Radar 1, Mode 1	1.0	0.5	0.5	0.3		
Radar 1, Mode 2	0.5	1.0	0.5		0.3	
Radar 1, Mode 3	0.5	0.5	1.0			0.3
Radar 2, Mode 1	0.3			1.0	0.5	0.5
Radar 2, Mode 2		0.3		0.5	1.0	0.5
Radar 2, Mode 3			0.3	0.5	0.5	1.0

Table 10.6
Notional Model Classifies Radars and Modes, Yielding a
Confusion Matrix $\mathcal{M}$ With 85% Accuracy

	R1M1	R1M1	R1M3	R2M1	R2M2	R2M3	Recall
Radar 1, Mode 1	10			2			83%
Radar 1, Mode 2	1	11					92%
Radar 1, Mode 3		1	5		1	5	42%
Radar 2, Mode 1				11			100%
Radar 2, Mode 2				1	12		92%
Radar 2, Mode 3						12	100%
Precision	91%	92%	100%	79%	92%	71%	

Table 10.7

The Quality Matrix $\mathcal{Q}$ Obtained by Combining the Confusion Matrix $\mathcal{M}$ of Table 10.6 with Reward Weights $\mathcal{R}$ of Table 10.5 Has a Quality Score of 90%

	R1M1	R1M1	R1M3	R2M1	R2M2	R2M3
Radar 1, Mode 1	10			0.6		
Radar 1, Mode 2	0.5	11				
Radar 1, Mode 3		0.5	5		0	1.5
Radar 2, Mode 1				11		
Radar 2, Mode 2				0.5	12	
Radar 2, Mode 3						12

10.8.4 Adequacy: Strategy Performance over a Scenario

Adequacy is the performance of the system over a scenario. A simple way to compute Adequacy is to count the strategy choices for each environment and score

Table 10.8

Layout of a Modified Confusion Matrix Used as a Performance Table

Environment	s_1	s_2	...	s_n
Env1 observables	$m_k(o_1, s_1)$	$m_k(o_1, s_2)$	...	$m_k(o_1, s_n)$
Env2 observables	$m_k(o_2, s_1)$	$m_k(o_2, s_2)$	...	$m_k(o_2, s_n)$
Env3 observables	$m_k(o_3, s_1)$	$m_k(o_3, s_2)$	...	$m_k(o_3, s_n)$
Unknown	-	-	-	-

against the best strategy for that environment, similar to computing the quality matrix $\mathcal{Q}$. This approach, however, doesn't take into account cases when changing strategies is part of the utility function, i.e., when the best strategy at time t depends on which strategy was in place at time $t - \delta$.

To compute Adequacy during a mission, the test framework uses the observed performance $U_n(t)$ divided by the best possible performance $\hat{U}_n(t)$:

$$\mathcal{A}_n(t) = \frac{U_n(t)}{\hat{U}_n(t)}$$

$\hat{U}_n(t)$ corresponds to the strategy that could have achieved best performance at time t . The framework computes $\hat{U}_n(t)$ from the performance tables $\mathcal{P}^{m_k}$ using the utility function in place at time t at node n , and computes $U_n(t)$ from the performance feedback returned at time $t + \delta$. (It may not be the true optimal, but rather the optimal among previously collected ground truth.)

Adequacy for the complete scenario is the average over all timestamps:

$$\mathcal{A}_n = \frac{1}{T} \sum_{t=1}^T \mathcal{A}_n(t)$$

Algorithm 10.1 shows algorithmically how to compute Adequacy. As shown, the algorithm runs in exploit mode only; the model does not explore for unknown environments or unknown strategies. The evaluator knows the optimal strategy and its performance for each of the n environments. To modify the algorithm to support in-mission learning, ensure (1) that the scenario controller provides performance feedback at each timestep from real or simulated execution, and (2) that the decision maker must retrain the model when it computes the best strategy.

Figure 10.8 shows results from a scenario with six environments changing every five time units. Generally, the BBN SO chooses the best strategy for the

Algorithm 10.1 This simple control loop uses a model $\mathcal{M}$ to estimate strategy performance at each time step. Adequacy evaluates the performance over the scenario. *This outline is based on the exhaustive search shown in Algorithm 5.1.*

SCENARIO CONTROL

$\mathcal{T}$ Training data for all environments $e \in \mathcal{E}$
 $\mathcal{S}$ Test scenario, as sequence of observations $\mathcal{S}_t$ of environments $e \in \mathcal{E}$

Send data for subset $e \in \mathcal{E}$ of the training data $\mathcal{T}$ to decision maker
 for timestep $t = 1$ to $|\mathcal{S}|$ *// Start of test scenario*
 $\dot{s} = \text{Get strategy from decision maker for } \mathcal{S}_t$
 $a_t(\dot{s}) = \text{Compute performance of } \dot{s} \text{ (e.g., through simulation)}$
 $\mathcal{A}_t = a_t(\dot{s}) / \hat{a}_t$ *// Compute Adequacy for this timestep*
 $\mathcal{A} = \frac{1}{|\mathcal{S}|} \sum_{t=1}^{|\mathcal{S}|} \mathcal{A}_t$ *// Compute Adequacy over scenario*

COGNITIVE DECISION MAKER

$\mathcal{T}$ Training data for some subset of environments $e \in \mathcal{E}$
 $\mathcal{S}_t$ Observations of current environment e

TRAIN INITIAL MODEL ($\mathcal{T}$)

$\mathcal{M}.\text{fit}(\mathcal{T})$ *// Train the model*

COMPUTE BEST STRATEGY ($\mathcal{S}_t$)

for each candidate strategy s *// Test each strategy*
 $\mathcal{I}_s = \mathcal{S}_t + s$ *// Combine current obs with strategy*
 $p_t(s) = \mathcal{M}.\text{predict}(\mathcal{I}_s)$ *// Predict its performance*
 $\dot{s} = \text{argmax}_s p_t(s)$ *// Choose best strategy*

s Strategy; combination of settings for each controllable c
 $\mathcal{E}$ RF environments. Each environment $e \in \mathcal{E}$ represents a different set of core values for the observables (e.g., the centroid of the cluster)
 $\mathcal{I}_s$ Instance; combination of current observables $\mathcal{S}_t$ and strategy s
 $p_t(s)$ Prediction on the trained model $\mathcal{M}$ for instance $\mathcal{I}_s$ at time t

environment, yielding an overall Adequacy $\mathcal{A}_n = 0.86$. Most errors occur when the environment changes, in that the strategy in place in env_i has less utility against the next environment env_{i+1} . When the Adequacy stays at 1.0 even when the environment changes, that indicates that the chosen strategy is equally useful in both

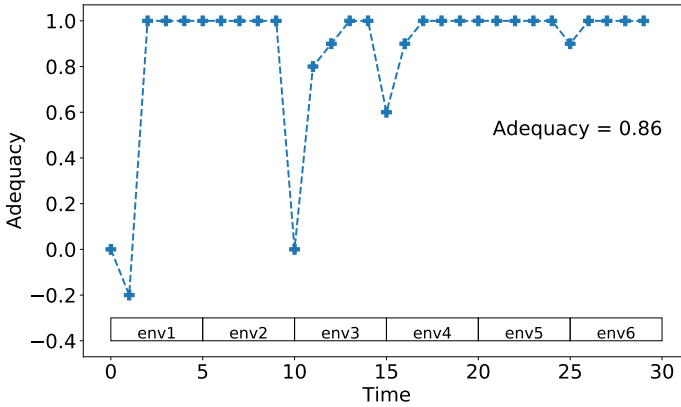


Figure 10.8 The test framework evaluates Adequacy at each time step using the decision maker's chosen strategy against the best-known strategy for that environment. (Example 1.1, [34].)

environments. Adequacy can be negative (e.g., at $t = 1$), because costs outweigh benefit. Adequacy can improve over multiple timesteps, e.g., $t = [11, \dots, 13]$, due to a ramp-up time of the strategy, or to incremental learning where the decision maker tries different strategies.

Figure 10.9 shows the impact of using incremental in-mission learning. The adaptive and cognitive systems start with a model trained on the same data. This initial model is logically equivalent to a mission data file; the adaptive system doesn't change the model, while the cognitive system updates it with experience. The closed-loop test framework exposes each system to the same sequence of novel environments. The adaptive system is not allowed to retrain and uses the learned-but-unchanging model throughout the scenario, yielding a final Adequacy score of 0.28. The cognitive system retrains three times, which enables it to choose better strategies, and achieves an Adequacy score of 0.88.

In-mission learning learns to handle the novel environments using only one example.

Human-generated labels or unsupervised clustering can determine the environments used for testing. Real-world observed performance determines the utility. The optimal value for a given environment may change during a mission, especially

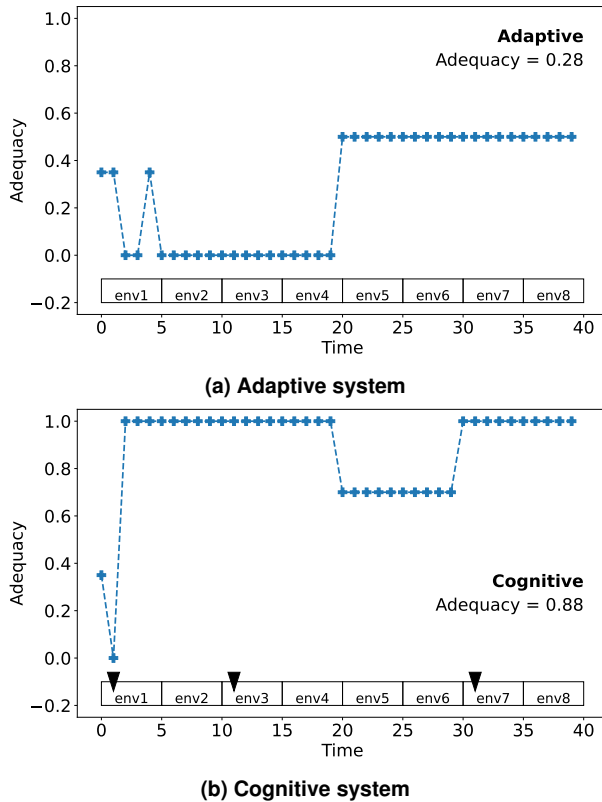


Figure 10.9 A cognitive system in (b) performs better than an adaptive system in (a) for the same test scenario. None of the environments in the training data were present in test scenario, therefore all environments in the scenario are novel. (*Triangles indicate when the decision maker triggered learning events. Example 1.1; redrawn from [34].*)

10.9 Ablation Testing

Ablation tests determine how much a priori training data is needed to get a desired accuracy. *n-choose-k ablation testing* proves that a cognitive system is capable of learning how to handle *new environments*. An ablation trial evaluates the impact of

specific training examples(s) on the generalization ability of the model.² The ground truth data has n known cases; we train the system on $k \subseteq n$ cases, and test on all n , for all values of k and all subsets $\binom{n}{k}$. Thus, during the test, $n - k$ environments are novel. For $n = 3$ environments known to the evaluator, there are a total of 8 ablation tests:

- $k = 0$. The test framework creates 1 test scenario $\binom{3}{0}$. The SUT receives no a priori training data; during the test, all $n = 3$ environments are novel.
- $k = 1$. The test framework creates 3 test scenarios $\binom{3}{1}$. In each, the SUT trains on 1 environment; during the test, 1 is known, and 2 are novel.
- $k = 2$. The test framework creates 3 test scenarios $\binom{3}{2}$. In each, the SUT trains on 2 environments; during the test, 2 are known, and 1 is novel.
- $k = 3$. The test framework creates 1 test scenario $\binom{3}{3}$. The SUT trains on all 3 environments; during the test, only the order of environments is unknown.

Ablation tests are similar to *leave-one-out testing*, which would train on $n - 1$ examples, and test on the remaining one example. Likewise, *k-fold cross validation* trains k models, and tests each on a different random $1/k^{\text{th}}$ of the data. Ablation tests are not random; they instead leave all subsets of classes out of the training data to demonstrate that the system can learn to handle new environments, regardless of what it was initially trained on. Ablation tests are a comprehensive way to evaluate model performance on data outside the domain-of-expertise.

To determine how much a priori training data is needed to get a desired accuracy, we run a scenario for all possible sets of a priori training data: for all $0 \leq k \leq n$, all combinations n -choose- k of k environments. This yields a total of 2^n scenarios, for n known environments. After predicting each new observation at time t in the test scenario S , the new observation is added to the training data before the next observation at time $t + 1$.

There may be many more environments than can be tested; for example, if there are 1000 known environments, running 2^{1000} scenarios is infeasible. Strategies to control the size of the full ablation test include:

- Test best case ($k = n$), then start with $k = 0$ and increase k until results show sufficient Adequacy.
- Rather than testing all n -choose- k subsets for a given k , test up to a maximum number of subsets to sample the distribution for each k .

2. In software engineering, ablation studies traditionally remove components to understand the contribution of the component to the system; ablation studies require that the system exhibit graceful degradation as components are removed [3].

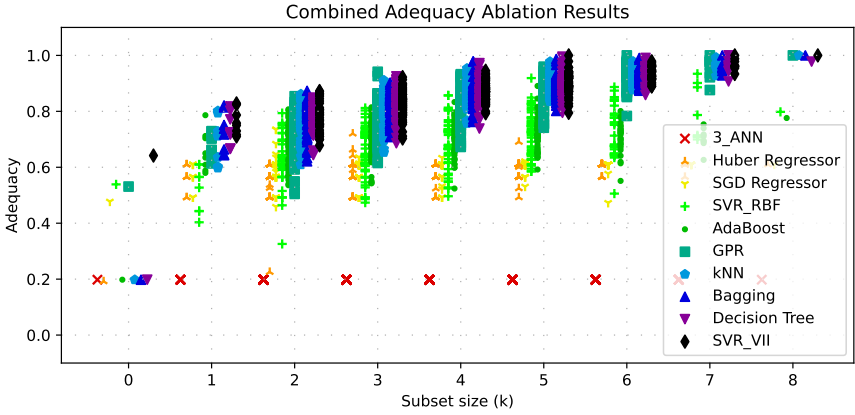


Figure 10.10 An n -choose- k ablation trial shows the impact of a priori data on training for 10 different ML models. SVM with the Pearson VII kernel achieves an Adequacy of 0.642 even with no a priori training data. (The x -axis is k showing how many conditions were present in the training dataset, and the y -axis is the Adequacy value per Section 10.8.4.)

- Cluster similar environments into m clusters, where $m \ll n$, and perform ablation testing of m -choose- k subsets.
- Select m canonical examples as the basis for the ablation tests.
- Use smart experimental design to select which scenarios to run (Section 10.7).

Figure 10.10 shows the ablation test results of 10 different ML models on the synthetic dataset from Table 4.1. SVMs with the Pearson VII kernel [35], decision trees, bagging, and k NN require the least a priori training data. The 3-layer ANN trained on 30 epochs does not perform well even when trained on everything; performance remains at the worst-case performance of 20% achieved by the default strategy, while incurring very high compute effort (Figure 11.4).

Figure 10.11 shows n -choose- k results collected when developing the BBN SO (Example 1.1, [34]). For each of $n = 9$ environments, the test framework trains the BBN SO on all subsets $k \subseteq n$, and tests on all 9 environments. This chart shows 512 individual tests, ranging from the case of $k = 0$ that has no a priori training data, to the case of $k = 9$ that trains on all $n = 9$ environments, but during the test the SO doesn't know their order.

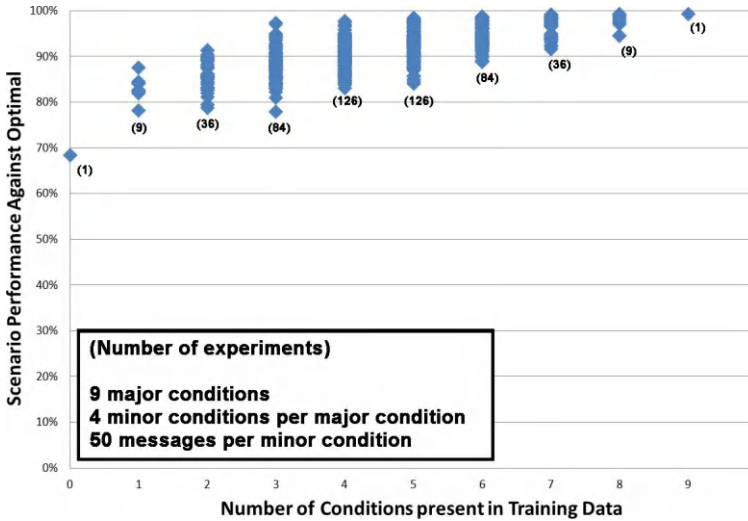


Figure 10.11 An n -choose- k ablation trial for the BBN SO shows that the performance gracefully degrades as the SO's a priori training data contains fewer conditions. (Each point corresponds to the Adequacy of one experiment such as Figure 10.8. Parenthetical "(x)" values show number of experiments performed for 9-choose- k .) (Example 1.1, [34].)

Even in the extreme case when the BBN SO receives no a priori training data, it achieves 70% of optimal performance due to in-mission learning.

Figures 10.8 and 10.9 show single scenarios corresponding to one point on Figure 10.11, and highlight how *in-mission learning* can learn how to handle a novel environment in only one or two examples. With about 30% of the training data, the BBN SO can achieve 98% Adequacy in the best case. In the worst case, it achieves 68% Adequacy.

This difference is explained by diversity; i.e., *which* environments were used in the size- k training data. The worst case uses three very similar environments, while the best case uses three very different environments. Section 2.1.1 describes how to compute environment similarity: Create unsupervised clusters of RF environments that can be visualized with a dendrogram. The dendrogram corresponds to observables for the n known environments, and the scenario determines which of these to use for training and testing. Because it is unsupervised, the EW decision maker can use this clustering to drive active learning (Section 7.3).

The framework can also use clustering to evaluate performance against optimal (Section 10.8.4); here, the clusters are named based on the ground-truth labels. Using Figure 2.3 as an example, a good test to determine performance on extremely different data would train the models on the 15 environments in the top cluster (16 to 23 vertically), and then test on 8 environments the bottom cluster (22 to 08).

Referring again to Figure 10.11, the best case might correspond to the three very different environments 16, 23, and 22 because the model generalizes effectively against all 20 of the other environments. Meanwhile, the worst case might correspond to training on the three similar environments 22, 02, and 07, causing the system to take longer to learn how to handle the novel situations, and thus receive a lower Adequacy score.

In Algorithm 4.1, during step 4 (classify new radars) the SUT does not know the ground-truth environment, but the framework can correctly evaluate the performance of an open-set classifier. The score of the classifier depends on the environments in question. For example, it may be acceptable to consider environments 05 and 06 as the same, especially if the environments represent jammers and the chosen mitigation techniques have the same performance. Likewise, an open-set classifier should be able to recognize a novel transmitter more quickly when it is really different than when it is only slightly different from known examples.

Ablation tests demonstrate empirically that the cognitive system can learn to generalize from its experience to handle novel environments.

10.10 Verification Approaches

Learning systems should be verified with both empirical and formal methods (or a combination thereof) [5]. Figure 10.12 briefly defines each category. Sections 10.10.1 and 10.10.2 present specific examples of formal and empirical verification methodologies, respectively.

The National Research Council describes how to assess the reliability of complex models [36].

Validation [of complex models] is a process involving measurements, computational modeling, and subject-matter expertise, for assessing how well a model represents reality for a specified quantity of interest and domain of applicability.

— National Research Council, 2012 [36]

Empirical Verification	Semiformal Verification	Formal Verification
Treat model as a white box	Combine mathematical and empirical concepts	Obtain worst-case robustness bounds using formal methods
•Conduct systematic testing of trained models •Examine both intrinsic and extrinsic features/behaviors	•Identify boundary cases for the trained models •Identify confidence of predictions •Adversarial testing	•Theoretically determine when the model changes greatly under the training dataset perturbations (e.g., when ML algorithm stability is compromised)

Figure 10.12 Learning verification approaches use empirical, formal, and hybrid methods.

Luckcuck et al. [37] provide a detailed overview of formal specification and verification for autonomous robotics. Several researchers examine methods for ANN verification for safety-critical systems (e.g., [38–42]). Example 10.1 presents an example of an assurance architecture for an aircraft taxiing application.

Pragmatically, useful outputs for verification methodologies include:

- Inputs that violate constraints (detecting out-of-distribution data);
- Amount that inputs can change before causing outputs to fail requirements;
- What outputs to expect for a given set of inputs;
- Confidence estimates that describe when a model is likely to fail; and
- Detecting unintended behavior.

10.10.1 Formal Verification Methods

Computational learning theory (CLT) and *statistical learning theory (SLT)* are subfields of AI that study the design and analysis of ML algorithms [45–47]. While CLT and STL both share the same theoretical framework, CLT attempts to determine what problems are “learnable,” and STL focuses on improving the accuracy of existing ML algorithms.

The *Vapnik-Chervonenkis (VC) dimension* determines the complexity of a given classifier [48]. A large VC dimension indicates that the classifier is more

Example 10.1 Multiple assurance approaches prove the safety of ML-based components within an aircraft taxiing application.

Cofer et al. [43] demonstrated a run-time assurance architecture for an aircraft taxiing application. The demonstration included a safety architecture based on the ASTM F3269-17 standard for bounded behavior of complex systems, diverse run-time monitors of system safety, and formal synthesis of critical high-assurance components. The developed architecture demonstrated the ability to maintain system safety in the presence of defects in the underlying ANN-enabled components.

Cofer et al. used the following techniques to assure the system: (a) modeling system architecture using the *architecture analysis and design language (AADL)*, (b) formally verifying system behaviors using the *assume guarantee reasoning environment (AGREE)*, (c) using an architecture-based assurance case for showing correct implementation with the Resolute language, (d) employing various run-time monitors for system safety, integrity, and availability, and (e) developing synthesis from formal specifications with proof of correctness for critical high-assurance components. AGREE uses k -induction as the underlying algorithm for model checking.

More recent work can be found online [15], and includes a discussion of how to demonstrate the absence of unintended behavior. Garcia and Cofer [44] present a discussion and tutorial on verifying AI, with a focus on safety-critical systems.

complex and vice versa. The VC dimension can, for example, predict a probabilistic upper bound on the test error of a classifier [49].

Probably approximately correct (PAC) learning is a theoretical framework proposed in 1984 by Leslie Valiant [50]. PAC learning is used for analyzing ML generalization errors in terms of their errors on training datasets, while also providing some measures of their complexity. Its typical objective is to demonstrate a high probability (hence the “probably” part) of an algorithm achieving a low generalization error (the “approximately correct” part). PAC learning has been extended to include *PAC-Bayesian inequalities* [51–53]. McAllester’s PAC-Bayesian analysis, for example, derives upper empirical bounds for Bayesian classifiers [54]. In a sense, this analysis could be considered semiformal.

Katz et al. [55, 56] develop a theoretical framework for verifying DeepNets. *Marabou* is based on *satisfiability modulo theories (SMT)* solvers and can answer queries about a DeepNet’s properties by transforming these queries into constraint

satisfaction problems. Marabou formally proved that DeepNets avoid midair collisions. SMT methods can also detect adversarial perturbations, that is, the minimal changes that cause the network to misclassify. The method can guarantee that adversarial examples, if they exist, are found for the given region and family of manipulations [57]. SMT solvers belong to a class of automated theorem provers that can deduce satisfiability and validity of first-order formulas, in particular logical theories. Balunovic et al. [58] show that SMT solvers are more general than satisfiability (SAT) solvers, and have been used for verification of ANNs, program synthesis, static analysis, and scheduling. SAT solvers essentially provide a generic combinatorial reasoning and search platform and have been used for software verification, planning, and scheduling [59].

Wang et al. [60] use *interval arithmetic* to compute rigorous bounds on the DeepNet outputs, and symbolic interval analysis to minimize overestimations of output bounds.

Maher and Orlando [61] use *ontology-based knowledge graphs* to enhance operational training of ML capabilities. An ontology model, through the creation of knowledge graphs, can address issues with bad data. This methodology allows for a formal and explicit expression of domain concepts by adding semantics to describe how domain data relate to each other. Knowledge graphs can support (i) structured knowledge validation to ensure conclusions are consistent with known facts, (ii) logic-based reasoning to ensure outputs maintain consistent and interpretable outputs, and (iii) incompleteness tolerance to eliminate false assumptions.

Formal verification methods are attractive due to their ability to theoretically determine algorithmic stability. However, the complexity of DeepNets and other ML algorithms makes it difficult to utilize formal software verification processes. Issues include computational complexity, scalability, applicability, lack of benchmarking, and trade-off between runtime and results completeness.

10.10.2 Empirical and Semiformal Verification Methods

As a contrast to formal methods, empirical verification methods typically use many execution runs to demonstrate system effectiveness. While this approach can provide practical insights, it has limitations such as data integrity and data bias that affect the reliability of the results. The most significant drawback to empirical verification is the size of the design space: We cannot determine statistically relevant results (Section 10.7). We therefore need to combine the strengths of both formal and empirical verification methods. By leveraging the theoretical rigor of formal techniques alongside the practical insights from empirical testing, we can

achieve a more robust and comprehensive evaluation. Approaches to empirical and semiformal verification include:

- *Ablation testing*: n -choose- k ablation testing (Section 10.9) demonstrates that the system can handle new environments because it trains on k cases, and tests all n known ground truth environments, thus evaluating performance on $n - k$ novel environments.
- *Adversarial testing*: This approach assesses ML algorithms' robustness against an adversary who does the worst possible thing (e.g., [14]).
- *White-box testing*: White-box verification methods assess the behavior of a model through testing. The white-box concept not only determines the final output behavior, but also its internal parameters such as neural coverage, activation patterns, etc. [62, 63].
- *Limitation studies*: To understand how well the system will perform in unexpected conditions, test it with known edge cases, such as increased noise, incomplete data, or errors. *Falsification*, for example, generates edge cases. While there are no guarantees of completeness or correctness with this method, test coverage is typically expanded in efficient ways that make sense [64, 65].
- *Mistake-bound model*: A mistake-bound model addresses the question of how many mistakes will an online learning algorithm make before it learns the intended concept or target function f [66, 67]. Learning occurs in rounds, where in each round the framework provides an unlabeled example x , and the learner must predict the value $f(x)$ of the unknown target function f . The SD then provides the performance feedback allowing the learner to retrain and update its hypothesis. The *mistake bound* of the learner is the worst-case number of mistakes it makes across all rounds.
- *Simplification*: Elboher et al. [68] propose a verification framework for simplifying neural nets by using over-approximation to reduce the size of the network. *Knowledge distillation* trains "teacher" models and then trains a simple "student" model to mimic the teacher [69–71]. Abstract interpretation approximates the behavior of the network, explicitly managing the trade-off between approximations and precisions [72].
- *Occam learning*: Occam learning is closely related to PAC learning but can produce tighter (empirical) bounds on the training dataset's complexity. The algorithm's objective is to output a *concise* representation of the received training dataset. Occam learning was named after *Occam's razor*, a principle

stating that, given two explanations for observed data, all other things being equal, the simpler explanation is preferred [73].

- *Uncertainty estimates*: Uncertainty can come from both data (*aleatoric* uncertainty) and from the prediction or model (*epistemic* uncertainty). Mucsányi et al. [74] discuss the need for disentangling these types of uncertainty. They show that there are no one-size-fits-all uncertainty estimators. Notably, the uncertainty estimators stay robust to perturbations longer than the models whose uncertainties they predict, thus enabling some level of safeguard for the underlying models. Calibration techniques [75], internal state-based confidence estimation [76], leveraging unlabeled data [77], and model explanations [78] further enhance ML trustworthiness. A recent survey for LLMs highlights these developments, emphasizing their growing role in AI/ML reliability [79].

Open-source tools exist to help compute confidence. Mucsányi [74] implement 19 models, covering supervised methods, distributional methods, and deterministic methods. The *model agnostic confidence estimator (MACEst)* is a Python library that integrates with scikit-learn, and handles different types of uncertainty [80]. IBM's *uncertainty quantification 360 (UQ360)* toolkit also provides a suite of tools to evaluate model confidence and reliability [81].

- *Sensitivity analysis/tuning studies*: Metalearning (Section 3.5) can estimate how sensitive the models are to changes in hyperparameters, yielding estimates of stability and robustness to change [82]. Output reachability analysis computes maximum sensitivity of ANNs, and then simulates the output reachable set [83].
- *Multiple-version dissimilar software*: Produce two or more components of software that provide the same function in a way that may avoid some sources of common errors between the components [44, 84]. Redundant solutions increase system resilience and improve confidence in learning approaches.
- *Direct assessment*: In some cases, an end user can evaluate the “plausibility” of system performance. While this approach doesn’t provide any “publishable” results, it builds stakeholder trust.

10.11 Example Implementation: Scenario Driver

Figure 10.13 illustrates the structure of a *scenario driver* (SD) that interacts with the cognitive engine, responding appropriately to stimuli in a fully closed-loop setting. The SD is a broker-based architecture, illustrated in Figure 9.5, supports data replay, data augmentation, data synthesis, simulation, emulation, and live components.

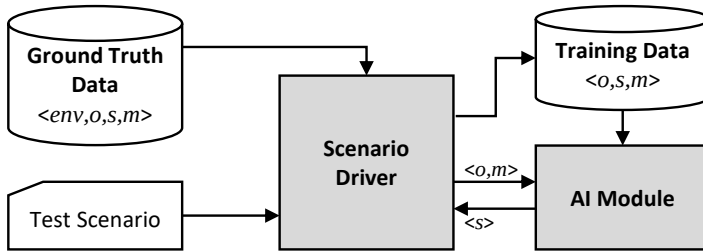


Figure 10.13 A closed-loop scenario driver ensures that a cognitive decision engine responds correctly to stimuli, sending observables o and metrics m to the AI module, which computes a strategy s .

A *ground-truth data file (GTDF)* contains all the known data, however it was generated; Table 10.9 shows an example from a real communications EP system. The data is normalized and anonymized to ensure that the ML model doesn't rely on the semantic meaning of features; even though some of these features may be system-specific, the learning approach should be platform-agnostic.

The GTDF is a record of all mission scenarios for which ground truth conditions are known: all combinations of tasks, nodes (friendly, neutral, and adversarial), and configurations. An *RF environment* (defined in Section 2.1.1) is one of the various test scenario conditions. In the simplest case, it might be free-and-clear communications under clear weather and low traffic requirements. Other environments might add mobility, jammers, weather, and topography. During controlled experiments, each environment is tagged with its known environment, and recorded in the GTDF.

The GTDF contain any observables, including raw I/Q samples, emitter descriptions such as pulse descriptor words (PDWs) or features from Figure 2.1, or inferred features. It also contains a record of all actions taken in each environment.

If the data is recorded in a comma-separated value (CSV) file, each row captures a single example of the observables, controllables, and metrics $\langle o, s, m \rangle$, describing how the strategy performed against that environment for node n . The observables correspond to a single observation of the environment.

In theory, the GTDF could have a very large number of rows for each tested RF environment, and an exponential (or even infinite) number of columns corresponding to every possible strategy. In practice, a smaller, sparse, dataset with high diversity is more valuable (Section 8.3.2). As the mission proceeds, the system collects the data, adds experience, and updates the table. A typical approach might

Table 10.9
A GTDF Contains All Known Data (Simulated, Emulated, and Real-World)

ENV	SHo1	SHo2	e1o1	e1o2	e1o3	e1o4	e1o5	e1o6	e2o1	e2o2	e2o3	e2o4	e2o5	e2o6	MT1	MT2	MT3	MT4	MT5	m1	m2
env1	-64	-10	123	127	-99	NaN	-127	-127	126	8	127	-127	-43	127	0	0	1	0	0	120	112
env1	-80	-10	123	127	-113	NaN	-127	-127	126	54	127	-127	-24	83	1	0	0	0	0	114	56
env1	-80	-10	126	-20	127	-127	-33	127	126	34	127	-127	-46	127	0	0	0	1	0	127	112
env1	-80	-10	126	23	127	-127	-47	127	123	127	-64	NaN	-127	-127	0	0	1	0	0	120	112
env2	127	-12	126	125	127	118	-127	77	NaN	NaN	NaN	NaN	NaN	NaN	0	0	0	1	0	127	127
env2	15	-56	126	124	123	119	-127	-127	73	127	-118	NaN	-127	-127	0	0	0	1	0	127	127
env2	15	-56	126	125	124	119	-127	-59	87	127	-127	NaN	-127	-127	0	0	0	0	0	88	0
env3	-127	NaN	126	NaN	NaN	NaN	-127	-127	NaN	NaN	NaN	NaN	NaN	NaN	0	0	0	0	0	0	0
env3	-127	NaN	126	NaN	NaN	NaN	-127	-127	NaN	NaN	NaN	NaN	NaN	NaN	0	1	0	0	0	0	70
env4	-64	-14	126	-53	93	-67	60	127	126	-39	88	-113	42	127	0	0	0	1	0	127	127
env4	-64	-14	126	-53	93	-67	60	127	126	-39	88	-113	42	127	0	1	0	0	0	101	70
env4	-64	-14	126	-54	107	-85	64	36	126	63	-64	NaN	-127	-113	0	0	0	1	0	127	127
env4	-64	-16	126	-53	96	-76	72	127	126	-41	111	-42	62	127	0	0	0	1	0	127	127
env5	-48	-14	126	-59	102	104	55	83	126	-54	109	51	40	107	0	0	0	1	0	127	127
env5	-48	-14	126	-66	115	95	68	95	126	-34	46	7	85	127	0	0	0	0	0	88	0
env5	-48	-14	126	-66	115	95	68	95	126	-34	46	7	85	127	0	0	0	0	0	88	0
env5	-48	-14	126	-66	115	95	68	95	126	-34	46	7	85	127	0	0	0	0	0	88	0
env5	-64	-16	126	-37	95	77	60	10	126	11	81	13	46	54	0	0	1	0	0	76	0
env5	-64	-16	126	-37	95	77	60	10	126	11	81	13	46	54	0	1	0	0	0	101	70
env6	47	-10	126	-127	127	127	100	127	NaN	NaN	NaN	NaN	NaN	NaN	1	0	0	0	0	63	62
env6	47	-10	126	-127	127	127	104	127	NaN	NaN	NaN	NaN	NaN	NaN	0	0	1	0	0	101	93
env6	47	-10	126	-127	127	127	76	127	126	-127	3	-127	127	127	0	0	1	0	0	101	93
env6	47	-10	126	-127	127	127	82	127	126	-127	4	-127	127	127	0	0	0	0	1	76	0
env6	47	-10	126	-127	127	127	82	127	126	-127	4	-127	127	127	1	0	0	0	0	63	62

This dataset is normalized to `int8` representation, and represents inferred features for (a) two shared observables that describe the entire RF spectrum, (b) six observables for each of two emitters, (c) five techniques that can be combined, and (d) two metrics.

be to collect data for each independent controllable (varying one controllable while the others are set to default values), and a few pairs (e.g., Table 7.1).

A *test scenario* describes which environments to use as training data and which to use as test data. The SD selects those examples of the GTDF that match the training environments and generates a training data file for the AI module to use premission. (Figure 9.6 shows the premission steps for the BBN SO.) The closed-loop test contains the following steps:

- SD generates one example observation of the test data (observables o and metrics m).
- AI characterizes the signal or chooses a strategy, or both, and returns the result to the SD.
- SD evaluates that response, and computes the metric m at the next iteration and the optimal performance per Section 10.8.3.

The SD plays sequences that interact appropriately with the decisions made by the AI. Using the simple state machine of Figure 10.14, for example, if the AI observes the request-to-send (RTS) and then successfully chooses a jamming technique to block the clear-to-send (CTS), then the SD would replay another RTS. If the AI did not choose to jam the CTS, then the SD would start sending the data. In the radar world, these labels might correspond to radar modes. Each arc indicates the trigger that causes the state to transition.

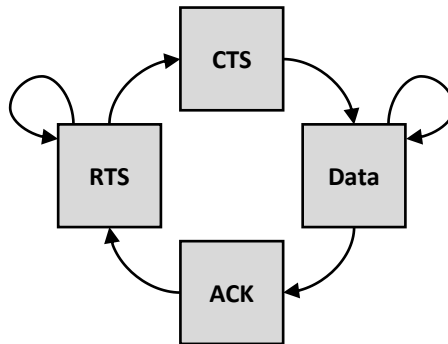


Figure 10.14 The SD can replay data from the GTDF using state machines such as this simple comms sequence. The “environments” in the GTDF would correspond to the RTS, CTS, Data, and ACK labels. In a radar, environments might correspond to radar modes.

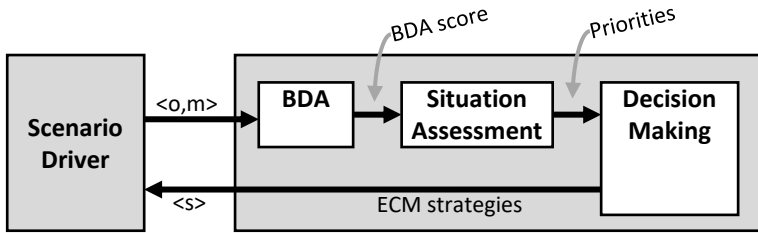


Figure 10.15 The SD can drive testing of multiple AI modules.

Structure the SD so that it can replay any kind of data, using any underlying state machine. This approach allows testing of multiple AI modules, for example one that generates I/Q samples to test a deinterleaver, one that generates PDWs to test a classifier, and one that generates PDWs with a state machine to test countermeasure choices. Figure 10.15 illustrates a notional example for developing and testing multiple AI modules. The SD generates data using the same format as it would arrive in the fielded system.

Each module in the SUT should be independently unit tested before testing the sequence. When bringing modules into the sequence, replace other modules with “omniscient” components that are always correct. For example, to test the situation assessment module, the decision maker can be a simple set of hand written rules, and BDA knows exactly how well the chosen countermeasure worked.

The SD evaluates the BDA and situation assessment modules using a weighted accuracy (Section 10.8.1 or Section 10.8.2.3), and the decision maker using Adequacy (Section 10.8.4).

For open-set classification (Section 3.4), this interactive process supports evaluation of how many new examples the SUT requires before correctly identifying a novel class. For example, if the training data contains 10 known emitters, the test data can generate observations describing an 11th unknown emitter. For each observation, the open-set classifier reports the probability that the new emitter is a new class, and the SD records how many observations are required before the classifier is correct. For ablation trials (Section 10.9), the SD evaluates how quickly the system learns how to handle a new RF environment.

10.12 Conclusion

AI capabilities are perhaps the most transformative technology in the history of humanity [85]. We are not currently prepared to sufficiently assure the behavior of

cognitive systems that can self-learn, and trust that the system operates consistently and safely within our expectations.

Evaluating a learning system requires thinking outside the traditional T&E methods. Traditional approaches tend to validate system performance one (or several) fixed, known scenarios, and do not portray novel situations. The learning assurance process must validate data quality, model accuracy, and model generalizability, as illustrated in Figure 10.2. System requirements generally handle pure decision-making (optimization) approaches, but have not yet caught up to the needs of learning-based AI. Data quality, data storage, model security, and validation objectives are lacking.

Due to the interactive and adversarial nature of EW, cognitive systems must be tested in a closed-loop environment. System requirements highlight which of the three axes of cognition (Figure 1.4) matter to the mission, ensuring that test scenarios and measured characteristics meet goals.

References

- [1] McLeod, S., “Maslow’s Hierarchy of Needs,” *SimplyPsychology*, 2024. Online: <https://www.simplypsychology.org/maslow.html>.
- [2] Lee, P., *Learning from Tay’s Introduction*, 2016. Online: <https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/>. Accessed 2025-01-15.
- [3] Cohen, P., and A. Howe, “How Evaluation Guides AI Research: The Message Still Counts More Than the Medium,” *AI Magazine*, Vol. 9, No. 4, 1988. DOI: 10.1609/aimag.v9i4.952.
- [4] Batarseh, F., and L. Freeman, *AI Assurance: Towards Trustworthy, Explainable, Safe, and Ethical AI*. Amsterdam, Netherlands: Elsevier, 2023. Online: <https://www.sciencedirect.com/book/9780323919197/ai-assurance>.
- [5] Cluzeau, J., et al., *Concepts of Design Assurance for Neural Networks (CoDANN)*, 2020. Online: <https://tinyurl.com/CoDANN-2020>. European Union Aviation Safety Agency.
- [6] EASA, and Daedalean, “Concepts of Design Assurance for Neural Networks (CoDANN) II,” European Aviation Safety Agency (EASA), Tech. Rep., 2021. Online: <https://tinyurl.com/CoDANN-2021>.
- [7] Soudain, G., *EASA Concept Paper: First Usable Guidance for Level 1 Machine Learning Applications*, 2021. Online: <https://tinyurl.com/easa-2021>.
- [8] EASA, and Collins Aerospace, “Formal Methods Use for Learning Assurance (ForMuLA),” European Aviation Safety Agency (EASA), Tech. Rep., 2023. Online: <https://tinyurl.com/easa-collins-formula>.
- [9] Delseny, H., et al., “Machine Learning in Certified Systems, Dependable and Explainable Learning (DEEL),” Tech. Rep., 2021. DOI: 10.48550/arXiv.2103.10529.
- [10] National Institute of Standards and Technology (NIST), *AI Risk Management Framework*, 2023. Online: <https://www.nist.gov/itl/ai-risk-management-framework>.
- [11] Office of Director for Developmental Test, Evaluation, and Assessments (DTE&A), *Developmental Test, Evaluation and Assessment*, 2025. Online: <https://www.cto.mil/dtea/te.aies/>. Accessed 2025-05-27.

- [12] Cofer, D., et al., "Flight Test of a Collision Avoidance Neural Network with Run-Time Assurance," in *Digital Avionics Systems Conference*, 2022.
- [13] Gendron, J., and R. Maduro, "The Role of Inference in AI: Start S.M.A.L.L. with Mindful Modeling," in *AI Assurance*, 2023. DOI: 10.1016/B978-0-32-391919-7.00019-6.
- [14] Katz, G., et al., "Towards Proving the Adversarial Robustness of Deep Neural Networks," in *Formal Verification of Autonomous Vehicles*, 2017. DOI: 10.4204/EPTCS.257.3.
- [15] AAHAA: *Architecture and Analysis for High-Assurance Autonomy*. Online: <https://loonwerks.com/projects/aaahaa.html>. Accessed 2025-01-23.
- [16] Clark, B., W. McNamara, and T. Walton, "Winning the Invisible War: Gaining an Enduring U.S. Advantage in the Electromagnetic Spectrum," Center for Strategic, and Budgetary Assessments, Tech. Rep., 2019. Online: <https://tinyurl.com/csba-ems-2019>.
- [17] Wang, S., et al., *Principle and Technology of Cognitive Electronic Warfare*. Beijing, China: National Defense Industry Press, 2018.
- [18] Sheikholeslami, R., and S. Razavi, "Progressive Latin Hypercube Sampling: An Efficient Approach for Robust Sampling-based Analysis of Environmental Models," *Environmental Modelling & Software*, Vol. 93, 2017. DOI: 10.1016/j.envsoft.2017.03.010.
- [19] Liu, C., and J. Ou, "Use Machine Learning Based Smart Sampling to Improve System Level Testing Efficiency," in *IEEE International Test Conference in Asia*, 2021. DOI: 10.1109/ITC-Asia53059.2021.9808718.
- [20] Meka, R., et al., "Multi-Armed Bandit Regularized Expected Improvement for Efficient Global Optimization of Expensive Computer Experiments With Low Noise," *IEEE Access*, Vol. 9, 2021. DOI: 10.1109/ACCESS.2021.3095755.
- [21] Schreiber, S., et al., "Enhancing Experimental Design through Bayes Factor Design Analysis: Insights from Multi-Armed Bandit Tasks," *Wellcome Open Research*, Vol. 9, 2024. Version 1; peer review: 3 approved with reservations. DOI: 10.12688/wellcomeopenres.22288.1.
- [22] Riis, C., et al., "Bayesian Active Learning with Fully Bayesian Gaussian Processes," in *NeurIPS*, Koyejo, S., et al., (Eds.), Vol. 35, Curran Associates, Inc., 2022. Online: <https://tinyurl.com/bayesian-smart-sample>.
- [23] Rainforth, T., et al., "Modern Bayesian Experimental Design," *Statistical Science*, Vol. 39, No. 1, 2024. DOI: 10.1214/23-STS915.
- [24] Cardoso Coelho, R., et al., "A Composite Bayesian Optimisation Framework for Material and Structural Design," *Computer Methods in Applied Mechanics and Engineering*, Vol. 434, 2025. DOI: 10.1016/j.cma.2024.117516.
- [25] Iordanis, I., C. Koukouvinos, and I. Silou, "On the Efficacy of Conditioned and Progressive Latin Hypercube Sampling in Supervised Machine Learning," *Applied Numerical Mathematics*, Vol. 208, 2025. DOI: 10.1016/j.apnum.2023.12.016.
- [26] Bogoclu, C., D. Roos, and T. Nestorović, "Local Latin Hypercube Refinement for Multi-objective Design Uncertainty Optimization," *Applied Soft Computing*, Vol. 112, 2021. DOI: 10.1016/j.asoc.2021.107807.
- [27] Kleijnen, J., "Kriging Metamodeling in Simulation: A Review," *European Journal of Operational Research*, Vol. 192, 2007. DOI: 10.1016/j.ejor.2007.10.013.
- [28] Fuhg, J., A. Fau, and U. Nackenhorst, "State-of-the-Art and Comparative Review of Adaptive Sampling Methods for Kriging," *Archives of Computational Methods in Engineering*, Vol. 28, No. 4, 2021. DOI: 10.1007/s11831-020-09474-6.

-
- [29] Rojas-Gonzalez, S., and I. Van Nieuwenhuyse, "A Survey on Kriging-based Infill Algorithms for Multiobjective Simulation Optimization," *Computers & Operations Research*, Vol. 116, 2020. DOI: 10.1016/j.cor.2019.104869.
 - [30] LinQuest, *Harness for Adaptive Learning*. Online: <https://tinyurl.com/linQuest-HAL>. Accessed 2025-01-23.
 - [31] SmartUQ, *Design of Experiments Overview*. Online: <https://smartuq.com/software/design-of-experiments/>. Accessed 2025-01-23.
 - [32] Burnell, R., et al., "Rethink Reporting of Evaluation Results in AI," *Science*, Vol. 380, No. 6641, 2023. DOI: 10.1126/science.adf6369.
 - [33] Youssef, K., et al., "Machine Learning Approach to RF Transmitter Identification," *IEEE Journal of Radio Frequency Identification*, Vol. 2, No. 4, 2018. DOI: 10.1109/JRFID.2018.2880457.
 - [34] Haigh, K. Z., et al., "Parallel Learning and Decision Making for a Smart Embedded Communications Platform," BBN Technologies, Tech. Rep. BBN-REPORT-8579, 2015. Online: <http://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-SO.sm.pdf>.
 - [35] Üstün, B., W. Melssen, and L. Buydens, "Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel," *Chemometrics and Intelligent Laboratory Systems*, Vol. 81, No. 1, 2006. DOI: 10.1016/j.chemolab.2005.09.003.
 - [36] National Research Council, *Assessing the Reliability of Complex Models*. National Academies Press, 2012. Online: <https://www.nap.edu/read/13395/chapter/1>.
 - [37] Luckcuck, M., et al., "Formal Specification and Verification of Autonomous Robotic Systems: A Survey," *ACM Computing Surveys*, Vol. 52, No. 5, 2019. DOI: 10.1145/3342355.
 - [38] Jacklin, S., et al., "Verification, Validation, and Certification Challenges for Adaptive Flight-Critical Control System Software," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004. DOI: 10.2514/6.2004-5258.
 - [39] Sun, X., H. Khedr, and Y. Shoukry, "Formal Verification of Neural Network Controlled Autonomous Systems," in *Hybrid Systems: Computation and Control*, 2019. DOI: 10.1145/3302504.3311802.
 - [40] Lahiri, S., and C. Wang, *Computer Aided Verification*. New York, NY: Springer Nature, 2020, vol. 12224. DOI: 10.1007/978-3-030-53288-8.
 - [41] Manzanas Lopez, D., et al., "Evaluation of Neural Network Verification Methods for Air-to-Air Collision Avoidance," *Journal of Air Transportation*, Vol. 31, No. 1, 2023. DOI: 10.2514/1.D0255.
 - [42] Manzanas Lopez, D., "Learning and Verification of Dynamical Systems with Neural Network Components," Ph.D. dissertation, Vanderbilt University, Nashville, TN, 2022. Online: <http://hdl.handle.net/1803/17754>.
 - [43] Cofer, D., et al., "Run-Time Assurance for Learning-Enabled Systems," in *NASA Formal Methods Symposium*. Springer, New York, NY, 2020. DOI: 10.1007/978-3-030-55754-6_21.
 - [44] Garcia, L., and D. Cofer, *AI Verification and Validation: Trends, Applications, and Challenges*, 2024. Online: <https://neurips.cc/Expo/Conferences/2024/talk%20panel/100351>. Accessed 2025-01-23. 90 minute video with slides.
 - [45] Bousquet, O., S. Boucheron, and G. Lugosi, "Introduction to Statistical Learning Theory," in *Summer School on Machine Learning*. Springer, New York, NY, 2003. DOI: 10.1007/978-3-540-28650-9_8.

- [46] Kearns, M., U. Vazirani, and U. Vazirani, *An Introduction to Computational Learning Theory*. Cambridge, MA: MIT Press, 1994. DOI: 10.7551/mitpress/3897.001.0001.
- [47] Russell, S., and P. Norvig, *Artificial Intelligence: A Modern Approach*. London, UK: Pearson Education, 2015.
- [48] Vapnik, V., and A. Chervonenkis, "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities," *Theory of Probability & Its Applications*, Vol. 16, No. 2, 1971. DOI: 10.1137/1116025.
- [49] Vapnik, V., *The Nature of Statistical Learning Theory*, 2nd Edition. New York, NY: Springer-Verlag, 2000. DOI: 10.1007/978-1-4757-3264-1.
- [50] Valiant, L., "A Theory of the Learnable," *Communications of the ACM*, Vol. 27, No. 11, 1984. DOI: 10.1145/1968.1972.
- [51] Shawe-Taylor, J., and R. Williamson, "A PAC Analysis of a Bayesian Estimator," in *Computational Learning Theory*, 1997. DOI: 10.1145/267460.267466.
- [52] McAllester, D., "Some PAC-Bayesian Theorems," *Machine Learning*, Vol. 37, No. 3, 1999. DOI: 10.1023/A:1007618624809.
- [53] McAllester, D., "PAC-Bayesian Model Averaging," in *Computational Learning Theory*, 1999. DOI: 10.1145/307400.307435.
- [54] Guedj, B., "A Primer on PAC-Bayesian Learning," in *International Conference on Machine Learning (ICML)*, 2019. Online: <https://arxiv.org/abs/1901.05353>.
- [55] Katz, G., et al., "The Marabou Framework for Verification and Analysis of Deep Neural Networks," in *Computer Aided Verification*. Springer, New York, NY, 2019. DOI: 10.1007/978-3-030-25540-4_26.
- [56] Katz, G., et al., "Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks," in *Computer Aided Verification*. Springer, New York, NY, 2017. DOI: 10.1007/978-3-319-63387-9_5.
- [57] Huang, X., et al., "Safety Verification of Deep Neural Networks," *Lecture Notes in Computer Science*, 2017. DOI: 10.1007/978-3-319-63387-9_1.
- [58] Balunovic, M., P. Bielik, and M. Vechev, "Learning to Solve SMT Formulas," in *NeurIPS*, 2018. Online: <https://papers.nips.cc/paper/8233-learning-to-solve-smt-formulas.pdf>.
- [59] Gomes, C., et al., "Satisfiability Solvers," *Foundations of Artificial Intelligence*, Vol. 3, 2008. DOI: 10.1016/S1574-6526(07)03002-7.
- [60] Wang, S., et al., "Formal Security Analysis of Neural Networks Using Symbolic Intervals," in *Conference on Security Symposium*, USENIX Association, 2018. Online: <https://arxiv.org/abs/1804.10829>.
- [61] Maher, M., and R. Orlando, "Solving the "Garbage in-Garbage out" Data Issue through Ontological Knowledge Graphs," Processus Group, and Kord Technologies, Tech. Rep., 2019.
- [62] Pei, K., et al., "DeepXplore: Automated Whitebox Testing of Deep Learning Systems," in *Symposium on Operating Systems Principles*, 2017. DOI: 10.1145/3132747.3132785.
- [63] Tian, Y., et al., "DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars," in *Software Engineering*, 2018. DOI: 10.1145/3180155.3180220.
- [64] Dreossi, T., A. Donzé, and S. Seshia, "Compositional Falsification of Cyber-Physical Systems with Machine Learning Components," in *NASA Formal Methods Symposium*. Springer, New York, NY, 2017. DOI: 10.1007/978-3-319-57288-8_26.

- [65] Zhang, Z., et al., “Two-Layered Falsification of Hybrid Systems Guided by Monte Carlo Tree Search,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 37, No. 11, 2018. DOI: 10.1109/TCAD.2018.2858463.
- [66] Buhrman, H., D. García-Soriano, and A. Matsliah, “Learning Parities in the Mistake-Bound Model,” *Information Processing Letters*, Vol. 111, No. 1, 2010. DOI: 10.1016/j.ipl.2010.10.009.
- [67] Littlestone, N., “Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm,” *Machine learning*, Vol. 2, No. 4, 1988. DOI: 10.1023/A:1022869011914.
- [68] Elboher, Y., J. Gottschlich, and G. Katz, “An Abstraction-Based Framework for Neural Network Verification,” in *Computer Aided Verification*. Springer, New York, NY, 2020. DOI: 10.1007/978-3-030-53288-8_3.
- [69] Hinton, G., O. Vinyals, and J. Dean, *Distilling the Knowledge in a Neural Network*, 2015. Online: <https://arxiv.org/abs/1503.02531>.
- [70] Papernot, N., et al., “Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data,” in *ICLR*, 2017. Online: <https://arxiv.org/abs/1610.05755>.
- [71] Mishra, A., and D. Marr, *Apprentice: Using Knowledge Distillation Techniques to Improve Low-Precision Network Accuracy*, 2017. Online: <https://arxiv.org/abs/1711.05852>.
- [72] Singh, G., et al., “Fast and Effective Robustness Certification,” in *NeurIPS*, 2018. DOI: <https://dl.acm.org/doi/10.5555/3327546.3327739>.
- [73] Blumer, A., et al., “Occam’s Razor,” *Information Processing Letters*, Vol. 24, No. 6, 1987. DOI: 10.1016/0020-0190(87)90114-1.
- [74] Mucsányi, B., M. Kirchhof, and S. J. Oh, “Benchmarking Uncertainty Disentanglement: Specialized Uncertainties for Specialized Tasks,” 2024. DOI: 10.48550/arXiv.2402.19460.
- [75] Guo, C., et al., *On Calibration of Modern Neural Networks*, 2017. DOI: 10.48550/arXiv.1706.04599.
- [76] Beigi, M., et al., *InternalInspector I²: Robust Confidence Estimation in LLMs through Internal States*, 2024. DOI: 10.48550/arXiv.2406.12053.
- [77] Li, C., X. Hu, and C. Chen, *Confidence Estimation Using Unlabeled Data*, 2023. Online: <https://arxiv.org/abs/2307.10440>.
- [78] Wu, K., C. Huang, and X. Yao, “Confidence Estimation Based on the Explanations of Model’s Predictions,” in *Neuromorphic Computing*, 2023. DOI: 10.1109/ICNC59488.2023.10462812.
- [79] Geng, J., et al., “A Survey of Confidence Estimation and Calibration in Large Language Models,” in *Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2024. DOI: 10.18653/v1/2024.naacl-long.366.
- [80] Rowe, M., *The Model Agnostic Confidence Estimator Helps Over-Confident Models*, TotalEnergies Digital Factory, 2021. Online: <https://blogs.oracle.com/ai-and-datascience/post/macest-release>. Accessed 2025-02-11.
- [81] IBM, *UQ360*. Online: <https://github.com/IBM/UQ360>. Accessed 2025-02-23.
- [82] Adkins, J., M. Bowling, and A. White, “A Method for Evaluating Hyperparameter Sensitivity in Reinforcement Learning,” in *NeurIPS*, 2024. Online: <https://openreview.net/forum?id=4OJdZhwBb>.
- [83] Xiang, W., H.-D. Tran, and T. Johnson, “Output Reachable Set Estimation and Verification for Multilayer Neural Networks,” *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 29, No. 11, 2018. DOI: 10.1109/tnnls.2018.2808470.

- [84] *Software Considerations in Airborne Systems and Equipment Certification*, Radio Technical Commission for Aeronautics (RTCA), 2011. RTCA DO-178C / EUROCAE ED-12C Standard.
- [85] Lee, K.-F., *How AI Will Completely Change the Way We Live in the Next 20 Years*, 2021. Online: <https://time.com/6097625/kai-fu-lee-book-ai-2041/>.

11

Getting Started: First Steps

Creating a CogEW system is not the hurdle that many believe.

It's easy to start small and grow.

1. Choose a bite-sized task.
2. Choose an ML toolkit and prototype some ML models.
3. Evaluate the models with representative data.
4. Implement on representative hardware.

As with all things, the devil's in the details, but starting small develops (human) expertise and awareness of which details will affect the final product.

Step 1: The first step in adopting cognitive solutions in an existing system is to identify a bite-sized task that is easy to accomplish. Figure 1.4 suggests a way of decomposing the problem to identify good options. Typically, this new component would replace or augment existing traditional approaches. Section 11.6 has several good candidates, structured as “summer intern” projects that could then grow into larger deployable systems.

Step 2: Choose one of the ML toolkits of Section 11.4. Implement an initial prototype with the process outlined in Algorithm 4.1. The goal is to *develop the pipeline*, not to test the accuracy. Test the pipeline on any available machine.

The main factor driving rapid prototyping is training time, which in turn is driven by amount of data and number of training iterations. Use a very small sample of data (e.g., < 100 samples); the data should have approximately similar ranges and numerical representations (preferably normalized), but doesn't have to be realistic or representative as outlined in Table 11.1. (That said, the more that the characteristics of this initial data represent the characteristics of the expected data, the better.) Train on 10% of the data, test on 90%, and choose hyperparameters to make the run as fast as possible, for example increase error limits and reduce number of training epochs.

Table 11.1
Development of the Initial Logic Flow Does Not Need Real or Realistic Data

Class	Duty	Freq	PRF	PW
Pulse	2.8	9200	5048	0.0005546
Pulse	2.9	7430	6373	0.0004550
Pulse	2.5	9790	1396	0.0017908
Pulse	1.5	4640	4609	0.0003254
PulseDoppler	38	1900	7799	0.0048724
PulseDoppler	49	4600	5148	0.0095182
PulseDoppler	34	8590	1619	0.0210006
PulseDoppler	26	5730	5116	0.0050820
CW	100	8830	7559	0.0132292
CW	100	610	3810	0.0262467
CW	100	7240	6772	0.0147666
CW	100	1570	9280	0.0107758

Step 3: Once the logic flow is established, switch to a representative dataset. Start with synthetic data and move to emulated and real data wherever possible. Focus on maintaining diversity (Section 8.3.2). Evaluate candidate models, and vary their hyperparameters over acceptable ranges. Tune the hyperparameters to get good results. Use the evaluation approaches in Chapter 10 to choose an ML algorithm.

Step 4: Implement on representative hardware. SDR platforms are a good stand-in for many EW platforms: they are relatively inexpensive and provide an RF front end (Section 11.5.2). Expect translation from prototype-model to embedded-code to be a significant effort (Section 9.3). When stable, transition to the target platform.

11.1 Engineering Resilient Systems

Defining a system's boundary influences how engineers approach risk and trust [1]. The difference between a "cognitive EW system" and a "system with cognitive EW" has important implications for assessing attributes like risk, safety, and security. Because the challenges and novelty associated with CogEW are generally very technical, the team will likely be composed of individuals with highly technical backgrounds and training. They will address the challenge using tools, processes, and frameworks from their domain of knowledge. However, focusing solely on the technical perspective makes it difficult to assess risk associated with the non-deterministic nature of AI.

We can expand the tools available to evaluate trust and risk effectively by broadening the system definition to a “system with CogEW” that includes operators, networks, and governance. This broader view is sociotechnical and brings to bear a wider variety of disciplines and employs a systems engineering perspective. The following approaches have slightly different complementary attributes that support engineering resilient systems:

- *Systems engineering*: Holistic lifecycle management, interdisciplinary focus, scalability and complexity management, system-level attribute assessment.
- *Agile development*: Iterative development, incremental risk assessment, flexibility to evolving requirements, stakeholder collaboration.
- *MLOps*: Continuous integration and deployment, monitoring for model-specific risks, automation of controls.
- *NIST’s AI risk management framework*: Structured risk management, continuous monitoring and adjustment, scalable to system complexity, aligns security with operational objectives.

Systems engineering is an interdisciplinary field focused on designing, integrating, and managing the lifecycle of complex systems, ensuring all components work harmoniously to meet user needs and objectives. It focuses on the design, analysis, and optimization of both hardware and software to meet user needs and operational requirements. Systems engineering provides a holistic approach that connects different engineering disciplines, addressing system reliability, scalability, and performance. Systems security engineering is a specialized discipline within systems engineering focused on integrating security, risk, and trust into system design, development, and operations to manage risks and ensure mission resilience [2].

One example of the application of the systems-based approach to safety is system-theoretic process analysis (STPA), developed by Dr. Nancy Leveson at MIT in the early 2000s as part of her work on system-theoretic accident model and processes (STAMP), a framework designed to address safety and risk management in complex systems [3, 4]. CMU’s Software Engineering Institute and the DoD’s Chief Data and AI Office recommend that STPA be used to ensure safety across all AI-enabled safety-critical systems.

Agile development focuses on iterative progress through short cycles (e.g., sprints in Scrum). It prioritizes flexibility and collaboration, allowing teams to adapt to changing requirements throughout the development process. By delivering functional pieces of software frequently and gathering feedback early, Agile helps ensure that the final product aligns closely with user needs. Agile also fosters a close relationship between developers, stakeholders, and end users, which is essential for rapid problem-solving and refining solutions.

In the agile approach, one creates *user stories* to capture each user's requirements and desired functionality at the highest level. The story's purpose is to communicate how a feature will provide value to various stakeholders. These user stories should encapsulate all the necessary interfaces/interaction points for each end user. A user story is an informal, general description of a feature written from the perspective of the end user, and can be as simple as [5]:

As a type of user, I want feature for benefit.

User stories are not system requirements, but a way to ensure that these various requirements are not “lost in translation” down the line. A good user story will state what is needed and why, and leaves the mechanism of how to achieve this up to the development team.

A key component of agile software development is putting people first, and a user story puts end users at the center of the conversation. These stories use non-technical language to provide context for the development team and their efforts. After reading a user story, the team knows why they are building, what they're building, and what value it creates.

— Max Rehkopf, 2020 [6]

At the highest level, the CogEW system's users are the stakeholders: designers, commanders, mission planners, and EWOs. The agile approach supports good human factors design, wherein stakeholders are regularly consulted from the beginning. MITRE's HMT engineering guide suggests approaches for this process [7].

ML operations (MLOps) extends the principles of development operations (DevOps) into the domain of ML. It focuses on automating and streamlining the deployment, monitoring, and management of ML models in production environments. MLOps ensures that ML models are continuously integrated, tested, and updated, maintaining high performance and relevance over time. This practice addresses challenges such as data drift, model decay, and the complexities of scaling ML solutions across various applications, ensuring that ML systems can evolve effectively within an operational framework.

NIST's *AI risk management framework (AI RMF)* [8] provides a structured, flexible process for managing security and privacy risks in systems. It encompasses six steps: (1) *Prepare*, to establish a risk context; (2) *Categorize*, to determine the system's security impact; (3) *Select*, to choose appropriate controls; (4) *Implement*, to apply those controls; (5) *Assess*, to verify their effectiveness; (5) *Authorize*, to approve the system for operation; and (6) *Monitor*, to ensure continuous compliance and improvement. AI RMF promotes a lifecycle approach, integrating security into system design, development, and operations. It supports scalability, aligns with

Callout 11.1 The five steps of an AI project guide and constrain development effort to encompass only necessary and sufficient actions.

In 1988, Cohen and Howe [9] presented an AI research five-stage model that describes evaluation guidelines per stage. These guidelines are provided in the form of detailed evaluation criteria and techniques, describing how to conduct an evaluation, and are still valid today:

- *Refine the topic to a task.* Is the task significant? Is it representative of the class of tasks?
- *Design the method.* Is the method an improvement over existing approaches? What is the scope of the method? What alternative approaches exist?
- *Build a program.* How demonstrative is the program? Is it tuned for an example? Are the results predictable?
- *Design experiments.* How many examples can be demonstrated? What benchmarks should be used to compare results? How does each component impact the result?
- *Analyze the results.* What is the performance? How efficient is the algorithm? What are its limitations?

The paper goes into more detail about each task, and how to evaluate steps.

organizational missions, and emphasizes ongoing collaboration among stakeholders to address evolving threats.

NIST's AI RMF provides a robust framework that supports the structured planning of systems engineering, the iterative progress of agile development, and the operational focus of MLOps. Agile development emphasizes iterative progress and flexibility, which complements MLOps by ensuring the continuous integration and deployment of ML models. Systems engineering bridges the gap, providing a structured framework to manage the entire lifecycle of complex systems, ensuring that both software and ML components work cohesively. Together, these practices foster a dynamic and efficient approach to building and maintaining robust, scalable systems that align with operational objectives.

11.2 Development Considerations

Callout 11.1 presents a list of tasks necessary for developing an effective AI system. EW efforts should factor in the following items:

- *Define a scenario.* This step is too often neglected. A scenario ensures collecting the right data and choosing the right algorithms for the platform

and mission. It is easy to collect the wrong data, develop “too much” ontology, choose the wrong hardware, or solve the wrong problem. Create a set of requirements, asking questions such as those in Table 3.3, Figure 6.2, and Chapter 8, and determine where and how the adversary plays a part. Determine the MOSSs, MOEs, and MOPs for system evaluation.

- *Ensure the team incorporates end users from the start.* End-user concerns are often neglected, whether it is the choice of the problem to solve, the metrics by which to evaluate the solution, the interface the users need when operating the system, or building trust in the end-user community. Use an agile development process to incorporate early feedback, and often to mitigate these problems (Section 11.1).

*Researchers like to develop the Rolls-Royce of systems,
when the end user only needs a wheelbarrow.*

- *Ensure the team incorporates RF engineers from the start.* RF engineers’ domain expertise is invaluable, in terms of identifying which parts of the system most require an AI-based solution, and identifying which domain features can be leveraged for a better assessment of the environment and actionable inferences (Section 4.2.1 and Section 11.3). Also, “upskilling” RF engineers with AI techniques can be an effective way to improve overall team performance.
- *Bring in data engineers from the start.* There are too many cases where RF engineers collect good data that can’t be reused later. The initial data framework (e.g., structure, ontology) should capture the initial use cases [10], and provide hooks for extensibility. Clearly define the metadata (Section 8.2.2).
- *Build the system in small increments.* Start with a small, well-scoped task, and grow the capability as you learn what works and what doesn’t. We learn more from failure than we do from success, so do not be disappointed when something doesn’t work. Instead, understand what caused the failure, and change the system design as needed. Ensure that no unacceptable behaviors appear (Section 5.3). Anything might change: requirements, data, models, algorithms, hardware, or test infrastructure. Use a flexible process for engineering resilient systems (Section 11.1).
- *Identify the required data early.* Too many solutions expect to have a good dataset, but in EW there will be no good existing dataset, it will be small, and there may be no ground truth. Knowing the characteristics of the data will drive the solution choice more than any other factor (Section 11.5.2).

- *Expect that most of your time will be cleaning data.* Common problems include correcting sensor errors, deconflicting with legacy capabilities, and only receiving the “scraps” of data that the legacy system will share (Section 8.1.1).
- *Prototype and evaluate a variety of candidate solutions.* Choose a tool that allows rapid development and comparison of algorithms (Section 11.4). When choosing a final solution, consider not only the accuracy, but other requirements (Section 3.9 and Section 10.3), including data availability, training time, inference time, memory usage, explainability and security.
- *Test for generalizability to surprise.* Validate the learning process (Section 10.2). Assume that your offline data is only somewhat indicative of online data. The solution *must* handle novel conditions. Use ablation trials to evaluate how much a priori data is required to reach acceptable performance and to ensure that there are no unacceptable effects (Section 10.9). While ablation trials establish confidence that the system will work on unknown conditions, events will occur outside *all* expected boundaries.
- *Design for a fully decentralized system.* Create a system that supports informed improvements rather than expecting graceful degradation (Section 5.5). Leverage connectivity for collaboration and coordination when available, rather than expecting safe and reliable communications. Only use a centralized node for advice that does not need to happen rapidly, and as an unreliable information archive.
- *Ensure you have a scalable software architecture.* Interoperability, modularity, and scalability are hard to retroengineer. Expect conflicts with legacy systems that will need to be resolved (Section 9.1).
- *Ensure system security from the start.* Adding security infrastructure later will not achieve objectives. Protect the data and the model (Section 8.3.5).
- *Prepare a closed-loop test environment.* EW systems operate against an adversary; testing using a static training set will not fulfill the goal. There is generally a lack of closed-loop test systems on which to test decision making (Section 10.5, Section 11.5.2).
- *Expect to manually translate the prototype code for the embedded platform.* No existing vendor toolchain is sufficient for deployment or memory and timing management on embedded hard real-time platforms (Section 9.3).

11.3 Choices: AI or Traditional?

AI, despite the hype in the media, is not applicable to every problem. AI solutions are less brittle, more transferrable to different problems, and more scalable than traditional approaches. AI is better at making “intelligent guesses:” handling missing values and finding a reasonable solution within some error bounds.

One does not need large computers to achieve impressive results. (That said, most AI practitioners would choose the largest compute and memory possible within the SWaP constraints.) You can always “right-size” the AI algorithm.

AI-based approaches scale to available hardware, including smaller hard real-time embedded systems.

For example, the BBN SO (Example 1.1, [11]) learns how to maintain communications in a variety of novel interference conditions. The SO uses SVMs [12], which enables it to achieve real-time in-mission learning on an ARMv7 processor in under a second.

The question is therefore how to choose between a traditional approach or an AI-based approach. There are times that a direct computation will be faster and more efficient. Broadly speaking, if an available model is accurate, then the traditional approach is likely to be more efficient than an AI-based approach. For example, if physics captures all relevant interactions, and empirical data doesn’t change the function significantly, then the physics model is a better choice. Likewise, if the physics is “close,” as is the case with path loss ($1/R^2$ for comms or $1/R^4$ for radar; Callout 3.1), then the physics model can guide and constrain the learner and make it easier to find a good solution from empirical data. See the *EW Handbook* [13] for additional examples.

In Chapter 3, we discussed how many traditional approaches can offer closed-form solutions, which are precise formulas that provide exact answers without requiring iterative approximation. Some techniques, such as linear regression and Fourier transforms, often have closed-form solutions, making them computationally efficient and straightforward to implement. Other examples include least squares estimation, which provides an exact solution for fitting linear models, and PCA, where eigenvalue decomposition yields principal components in a closed form [14]. Similarly, quadratic programming in certain convex optimization problems [15] and Kalman filters in linear dynamical systems allow for analytical solutions under specific conditions [16].

Recall that these closed-form methods ensure accurate, interpretable results when their underlying assumptions hold. However, they become impractical when

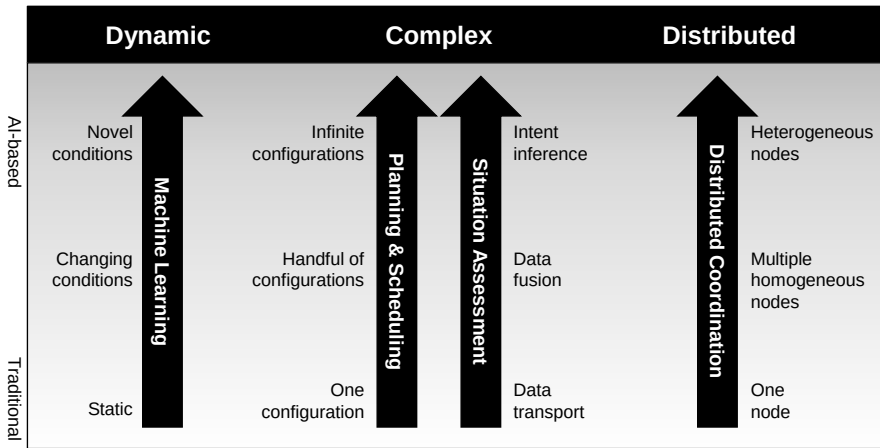


Figure 11.1 Domain characteristics determine whether AI is useful or necessary. Figure 1.4 places the AI functions in 3D to show levels of cognition.

dealing with complex, high-dimensional, or nonlinear problems, where explicit solutions are often unavailable. In such cases, ML methods rely on iterative optimization, enabling greater flexibility and scalability by adapting to complex patterns in data rather than requiring exact mathematical formulations.

AI approaches and traditional approaches should coexist on platforms, performing tasks appropriate for their strengths. Depending on measurements, conditions, and uncertainty, either could be the final arbiter for given tasks. Likewise, the two approaches could operate in parallel, and the results combined to generate the final outcome.

We can evaluate a problem domain according to any of the characteristics in Section 1.9. The most informative of these are how *distributed*, *complex*, or *dynamic* the problem is. Figure 11.1 illustrates these axes, indicating attributes that determine when a more traditional approach is likely to be sufficient, and when an AI-based approach becomes necessary.

When the problem domain is highly *distributed*, distributed coordination (or multiagent techniques) become increasingly important. If a single node can accomplish the task, the system does not need to merge potentially disparate observations of the same object, negotiate task assignments, or coordinate actions. With a team of multiple homogeneous nodes, each node can make assumptions about its teammates, minimizing communications and facilitating coordination. Heterogeneous

nodes must allocate resources and tasks across the nodes and dynamically update those allocations during a mission, while minimizing communications overhead. Most AI-based distributed coordination approaches generate more resilient plans than conventional control systems. Networked EW systems have loosely coupled task coordination (e.g., deciding which nodes should do stand-in jamming or becoming a communications relay), and they also have tightly coupled task coordination (e.g., performing distributed coherent transmissions).

When the problem is highly *complex*, situation assessment techniques become relevant for data understanding. When data simply needs to be moved from one node to another, no knowledge is extracted from the data. When a traditional model can extract useful information efficiently and accurately, then AI is unlikely to improve results. Data fusion brings together disparate data sources, possibly different types of data collected, or the same type of data from multiple nodes. The more disparate the types of data, and the more varied the temporal characteristics, the more that data fusion concepts need to be employed. Plan recognition and intent inference techniques enable the system to understand the intent of actors in the domain, whether they are users or adversaries. To maintain system performance, EP systems need to recognize the intent of the users. EA systems are more effective when they understand the intent of the adversary.

Planning and scheduling techniques address decision making as domain complexity increases. If the system does exactly one thing (i.e., has one configuration), then traditional control approaches are appropriate. Simple rules can manage a handful of possible configurations. Planning, scheduling, and optimization approaches become necessary when there are many possible configurations. In most EW systems, the configuration parameters can be combined in many ways, yielding a possibly infinite number of configurations. At the same time, the EW system may face an infinite number of scenarios. AI-based decision making is critical.

The third key domain characteristic is *dynamism*. When the domain is static, a preconfigured system is sufficient for the task. When the domain is dynamic, and the system encounters changing conditions, an adaptive approach becomes necessary, for example selecting EP countermeasures appropriate for the observed jammers. These responses can be designed a priori by hand or by a (non-real-time) ML approach, and then uploaded to the system via EW reprogramming. When the system will encounter novel conditions, such as novel transmitters, then a real-time in-mission ML approach is crucial.

Each of these technical areas (distributed coordination, situation assessment, decision making, and ML) support a vibrant research community in AI, including dedicated technical conferences and publications. In each case, the research

community is developing multiple approaches; the CogEW community can choose approaches appropriate for available hardware. Each of the chapters in this book discusses relevant approaches, and highlights the tradespace that system architects should consider.

The key lesson is to not select the solution (i.e., cool AI technology) without fully understanding the problem.

What desired outcome will the AI enable? What data exists upon which to base a ML solution? Do the characteristics of the problem warrant an AI-based solution?

11.4 ML Toolkits

Popular ML libraries include scikit-learn [17], TensorFlow [18], MATLAB Machine Learning Toolbox [19], R [20], and WEKA [21]. These kits have good online documentation. The book *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow* [22] is an excellent foundation for the Python environment. The book *Hands-On Machine Learning with R* covers similar concepts for R [23].

These toolkits offer a wide variety of ML models, including clustering, classification, and regression. You can find methods to compare ML models on the same dataset, for example, Python's package `lazypredict`, MATLAB's classification (or regression) learner app, or R's `caret`. `lazypredict` computes accuracy and time, while the classification/regression learner apps only compute accuracy, and you will have to write code to evaluate the models for other metrics. If you need help generating code to do the comparison, one of the popular LLMs like OpenAI Codex [24] or CursorAI [25] is a good assistant. Use a prompt like *Write a function in MATLAB to compare and contrast 3 regression models according to accuracy, training time, and inference time. Use a single function call if possible.* Both Python and MATLAB have other associated packages like statistics, data visualization, and RF data generation.

These libraries are effective for prototyping, but will need significant modification for embedded hard real-time operation (Section 9.3).

11.5 RF Datasets and RF Data-Generation Tools

Existing datasets and simulation or emulation frameworks can accelerate the prototype and development process. Given the complexity of the RF environment, exacerbated by the adversarial nature of EW, synthetic and emulated data generation capabilities are key, both for basic data generation and interactive scenarios.

11.5.1 RF Datasets for ML

Datasets are a good way to bootstrap a project, develop algorithms, and explore initial ideas. The development of robust ML models is directly correlated to the quality and quantity of data used to train those models. In image recognition and natural language processing, large curated benchmark datasets are available to train models and assess performance of architectures and training regimes, thereby yielding more objective and unbiased comparisons. The success of these datasets is due to the diversity of samples, the precision with which they have been labeled, and the ability for anyone to easily download them from the internet.

In EW, a large, curated, accessible dataset does not exist, but several starting points are available. NIST Science Data Portal [26], IEEE DataPort [27], the RF Data Factory [28], Kaggle [29], and Google Dataset [30] provide links to many public datasets. These are not always curated. Kaggle's datasets are used for competitions, and often come with associated code. NIST's *Advanced Communications* category yields about 100 results for "IQ data" in January 2025. Lang et al. survey synthetic aperture radar (SAR) datasets [31].

Good, comprehensive RF-specific datasets are desperately needed [32, 33]. Even if the data *exists*, few of the datasets are tagged well enough for reuse (Section 8.2.2). Starting points include:

- USRP RF Fingerprinting from the GENESYS Lab [34]. Recordings of I/Q samples collected from over-the-air transmissions of 16 USRP X310 transmitter radios; one dataset is raw samples, and the other dataset has been demodulated and equalized.
- WiSig RF Fingerprinting from the University of California (Los Angeles) [35]. 10 million packets captured from 174 off-the-shelf WiFi transmitters and 41 USRP receivers over 4 captures spanning a month.
- Single- and Multi-Channel Signal Separation Challenge from the Massachusetts Institute of Technology (MIT) [36]. Code, simulation engine, and data. 14 GB of data with three types of signals (interference and two communication signals) collected over-the-air for signal separation and demodulation tasks.
- I/Q data associated with ADS-B beacons [37]. Section 8.3.3 uses this data in an example. Labeled, real-world I/Q data, collected over one month. 200-600 samples per category. The 1.5 GB dataset has 198 categories, each sample has 3000 I/Q samples. The 9 GB dataset has 530 categories; each sample has 6000 I/Q samples.

- Radar Challenge 2024 from IEEE Aerospace and Electronic Systems Society [38]. Code, simulation engine, and data. Teams worked with the PlutoSDR and the Phaser 10 GHz Phased Array Radar platform from Analog Devices. This is a low-cost, simplified phased array radar that allows real beamforming hardware to be used for education, research activities, and even software development in MATLAB. Winning teams showed (1) ISAR imaging for navigation surveillance; (2) radar-based tomography; and (3) small drone surveillance and classification.
- WiFi and Bluetooth I/Q data from NIST [39]. Includes baseband, interleaved RF recordings of WiFi and Bluetooth emissions in the 2.4 GHz and 5 GHz unlicensed bands, collected using low-cost SDRs. 900 one-second captures, with a sampling rate of 30 MS/s.
- ADS-B feature data from OpenSky [40]. Global ADS-B data since 2013 with 30 trillion samples from more than 7000 sensors. 16 features: time, icao24, lat, lon, velocity, heading, vertrate, callsign, onground, alert, spi, squawk, baroaltitude, geoaltitude, lastposupdate, lastcontact.
- Global Navigation Satellite System (GNSS) Combined Vehicular Measurements Dataset from University of Bristol [41]. GNSS I/Q recordings with lidar, inertial measurement unit (IMU), and computer vision data.
- Multichannel airborne ground moving target indicator (GMTI) radar dataset from DARPA KASSPER [42]. Feature data from a high-fidelity physics-based multichannel simulated radar with terrain, leakage, and ground traffic.
- GPS Spoofing Test Battery from the University of Texas at Austin [43]. Six high-fidelity recordings of over-the-air civilian GPS spoofing tests. The spoofer reads authentic GPS and generates a counterfeit signal.
- I/Q signal data collected on an SDR for signal classification, interference detection, device optimization, weather analysis, and geolocation [44]. 164k of data collected on an SDR over a month. Includes metadata and I/Q. Sample code is for anomaly detection.
- OpenRAN Gym contains data from LTE, 5G, and mmWave data [45]. It pairs real-world cellular traffic traces with emulated 5G and LTE networks.
- RF data from radar and comms systems from SRC [46]. Synthetic I/Q data from realistic settings (9 GB or larger), with varying SNR and background interference, along with parsing utilities in Python.
- Wideband radar detection dataset for low probability of intercept signal detection from Queensland University of Technology [47]. Long sequences

with 1 million baseband I/Q samples, stored as 40,000 frames and up to 9 radars per frame; shared as spectrograms.

The communications EP feature dataset shown in Table 10.9 was collected on a communications node in over-the-air conditions (Example 1.1). This dataset is unique in that it captures the actions that the system took, and the resulting performance. Section 4.3 shows how the system used the data to estimate performance of available actions, and shows an example of a similar synthetic dataset.

The limitation of recorded datasets is that they are fully static, and therefore can only be used for systems with *no decision making capability of any kind*. For example, a recording of a day-in-the-life of your hometown is an excellent resource for developing initial signal understanding capabilities. However, *as soon as your system can make decisions*, you need a closed-loop test environment. For example, you can use a recorded video to develop initial object-recognition capabilities, but if you want to develop a smart camera, and test its ability to change the pan-tilt-zoom, you need a closed-loop setting.

11.5.2 RF Data-Generation Tools

RF data-generation tools can be software or hardware, low-fidelity or high-fidelity, and open-loop or closed-loop. All of these capabilities are required in a complete testing infrastructure; Section 10.5 discusses these issues in more detail.

11.5.2.1 Software Simulation Tools

Software approaches have the advantage that they can be run anywhere, at any level of fidelity, from fine-grained physics-based simulation through faster-than-real-time event-based models.

MATLAB offers an RF Toolbox [48], and other plugins exist such as SEWES [49] and Skydel [50]. Python has several libraries, including TorchSig [51], Scappy [52], PyLayers [53], and RFML in PyTorch [54].

Communications network simulators include Riverbed [55], EMANE [56], NS3 [57], OMNeT++ [58], and QualNet [59].

RFView [60, 61] and SynDICAte [46] perform signal generation and compute propagation effects for realistic RF environments. If there is a model of how the system would respond to stimuli, they can be used within a closed-loop system. SynDICAte, for example, selects radar or radar parameters such as interpulse modulation, intrapulse modulation, and scan patterns to generate RF data, augmenting it with scenario-specific physics and geometry-based priors, as well as correlated and uncorrelated stochastic effects such as frequency selective fading,

multipath, Doppler effects, time/delay spread, thermal noise, and receive system transfer functions. SynDICAte labels the synthetic RF data and detection data with high-precision truth, varying from high-level parametric summaries to precise labels for each pulse of a simulated signal.

Spirent's GNSS simulator [62] supports closed-loop simulation testing for position, navigation, and timing. Simics [63] and SystemVue [64] simulate low-level hardware.

11.5.2.2 Hardware Test Environments

Hardware platforms include low-cost SDRs through very expensive exquisite signal generators, covering a range of applications, from vehicle communication and spectrum monitoring to GNSS and radar emulation. Some support full comprehensive, closed-loop testing and emulation under real-world conditions.

Many commercial signal generators exist, including NEWEG [65], CEESIM [66], and vector signal generators [67, 68]. Some of the software tools above can also operate with hardware-in-the-loop (e.g., [60, 62]). The majority of signal generators are *open-loop*, in that they generate data without supporting the fully interactive nature of EW. To use them within a closed-loop setting, you would need a model of how the system would respond to stimuli (Section 10.6).

RFnest [69] computes complex RF propagation effects; DARPA's DRBE program [70] aims to build a massively many-to-many version. ARES [71] is a radar environment simulator that computes target reflections through a complex propagation environment.

FABRIC [72] (formerly NSF's GENI project) incorporates a variety of wireless network emulators. FABRIC is a distributed set of equipment for research in networking, 5G, cybersecurity, ML, and science. The OpenRAN Gym [45] is research platform for AI experimentation on wireless networks. ORBIT [73] and POWDER [74] have several 5G and SDRs testbeds with levels increasing realism, from controlled testbenches going up to full real-world city-size mobile platforms.

You can also create your own purpose-built testbed. Tutorials exist to show you how to generate your own data or set up a lab, (e.g., [28, 45, 74]). A small testbed using Raspberry Pi or USRP radios supports inexpensive over-the-air signal collection and testing for initial prototypes to develop and test the decision logic. SDRs, illustrated in Figure 11.2, can generate both radio and radar waveforms, and can be integrated with other systems such as GPS and inertial navigation units. A GNU radio module exists for PDW generation [75]. Moreover, both military and civilian organizations are both deploying SDRs in the field, making it sensible to use these affordable devices. When designing approaches for more expensive

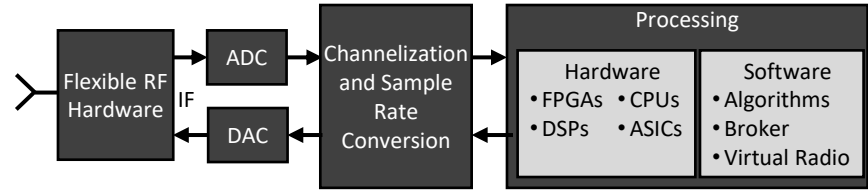


Figure 11.2 SDR platforms provide RF front end, CPU, FPGA, and in some cases ASICs. (See also Figure 8.5.)

operational systems, SDRs may not have the same signal purity or power output, so further testing on these devices becomes necessary.

11.6 Projects

Table 11.2 outlines a few possible “summer intern” projects that could grow into larger, deployable concepts. NATO’s *Cognitive Radar Working Group* released a cognitive radar report that also compiles good examples [76]. NATO’s panel on *ES for Cognitive EW* studied relevant ES techniques [77].

Many of these projects begin with a dataset or the ability to collect one. Section 11.5.1 lists several existing datasets, or you can create your own using RF data-generation tools from Section 11.5.2. You will also need appropriate models from one of the ML toolkits in Section 11.4. Always build your code so that it can

Table 11.2
Project Ideas

Situation Assessment	Decision Making	Other
Classify signals (Section 4.2.2)	Route planning	Data augmentation
RF fingerprinting (Project 11.6.1)	(Project 11.6.8)	(Project 11.6.11)
Open-set recognition (Project 11.6.2)	Decision control and learn	Evaluate ML
Detect causal relationships	action performance	(Project 11.6.12)
(Project 11.6.3)	(Project 11.6.9)	DeepNet on FPGA
Describe emitters (Project 11.6.4)	5G (Project 11.6.10)	(Project 11.6.13)
Classify interference in WiFi and		Data compression
Bluetooth (Project 11.6.5)		(Project 11.6.14)
Detect GNSS anomalies (Project 11.6.6)		
Fuse GNSS sensors (Project 11.6.7)		
EW BDA (Section 7.1.1)		

Algorithm 11.1 Simple ANNs can perform RF fingerprinting tasks.

```
# Assuming 3000 I/Q samples
model = models.Sequential([
    layers.Dense(3000, activation="relu"),
    layers.Dense(1000, activation="relu"),
    layers.Dense(198)
])
conv_model = models.Sequential([
    layers.Conv1D(10, 1000, activation="relu",
                  input_shape=(3000,2)),
    layers.Conv1D(16, 1000, activation="relu"),
    layers.GlobalAveragePooling1D(),
    layers.Dense(198)
])
```

compare multiple models against each other, for example by creating a list of all available models and then iterating on it, similar to Algorithm 7.1.

11.6.1 Project Idea: RF Fingerprinting

Objective. Uniquely identify transmitters based on their intrinsic properties, even when they are notionally identical.

Approach. Construct a classifier and evaluate it.

Starting Points. (1) Obtain or collect an I/Q dataset for SEI or RF fingerprinting. (2) Construct a model, such as the ones in Algorithm 11.1, or one you find elsewhere (e.g., [54, 78–81]). (3) Evaluate it using a confusion matrix (Section 10.8.2). (4) Try several different models, both classic and ANN-based, to see which works best.

Example Results. Figure 8.7 for ADS-B I/Q data, and Figure 10.7 for USRPs.

Future Directions. (1) Evaluate model performance according to multiple metrics (see Project 11.6.12). (2) Limited training data: Instead of the usual 80/20 training/test split, use 10/90 or 1/99. (3) Rapid identification: Instead of 1000 IQ samples, use 100, 50, or 10. (4) Augmentation: Augment limited training data with both domain-independent and domain-dependent techniques. (See Project 11.6.11 for more discussion.) (5) Complex transmitters: Add mobility and waveform agility. (6) Complex RF propagation: Add complex multipath environments, especially low SNR. (7) Receiver invariance: Transfer models to multiple receivers.

11.6.2 Project Idea: Open-Set Recognition

Objective. Perform open-set recognition of new objects.

Approach. Create new classes on the fly for cases when the training data does not cover all known cases.

Starting Points. (1) Obtain or collect a feature-based or I/Q-based dataset of emitters. (2) Extract a list of the classes; select some classes to be in the training data, say 50%, and some to be in the open-set test data, say 5 classes. (3) Select one or more multiclass classification algorithms so that you get prediction probabilities. Read Section 3.4 for algorithmic approaches. (4) Run in “online” mode: For each novel class, for each example i of the novel class, predict what class i belongs to. If the most likely class is very low probability, add the new class to the classifier. This step is the tricky one, because i might simply be an anomaly, and not a new class. (5) Plot a graph of i vs probability that it belongs to the new class; each observation of the class should increase the probability that it is a new class and not an anomaly. (6) Evaluate the approaches for accuracy and speed to identify new classes.

Future Directions. Determine the impact of training set characteristics on the probability of a new class, such as fewer a priori classes, fewer instances per class, complex emitters, and complex RF propagation.

11.6.3 Project Idea: Detect Causal Connections

Objective. Understand network connectivity patterns and identify key nodes, without decoding the packets.

Approach. Capture statistics about transmissions and evaluate the probability that nodes are talking to each other. For example, terrorist cell networks may all use the same Iridium satellite phone, but the independent cells do not talk to each other.

Starting Points. (1) Obtain or collect a dataset of emitters that includes at least the node identifier, the start time of a transmission, and the duration of the transmission. (2) Build a conversation probability matrix based on timing or Granger causality [82, 83]. (3) Identify network sources and sinks, and key nodes such as the hub of a hub-and-spoke network.

Future Directions. (1) Explore other techniques, per Section 4.6, and determine which is most accurate for your context. Multiple Python and MATLAB libraries exist for causal inference [84, 85]. (2) Estimate the confidence in the statistics. (3) Estimate routes for a packet through a multihop network. (4) Estimate the wireless protocol, and determine if and how the protocol being used is different from a standard known protocol.

11.6.4 Project Idea: Describe Emitters

Objective. Create human-legible emitter descriptions.

Approach. Cluster similar emitters and create a generalized description.

Starting Points. (1) Obtain or collect a feature-based dataset of emitters. The first dataset should not include ambiguous emitters, i.e., ones with overlapping descriptions. (2) Select one or more clustering approaches. (3) Look at emitter similarity using a dendrogram (Section 2.1.1). (4) Create generalized description using a human-friendly model such as a decision tree. Search for references on explainable or interpretable clustering, for example [86–89].

Future Directions. (1) Increase the complexity of the dataset to include ambiguous emitters. (2) Expand to using I/Q data. (3) Examine self-supervised learning approaches for clustering and feature extraction, for example [90, 91]. (4) Test different responses for each description; present available responses in a table similar to Section 4.3. (5) Examine whether contextual information can be extracted from EMS data in a way similar to video data, and then used to improve tracking.

11.6.5 Project Idea: Classify WiFi and Bluetooth Interference

Objective. Detect and classify interference types within WiFi and Bluetooth bands, enhancing threat detection in contested RF environments.

Approach. Extract features from WiFi and Bluetooth I/Q data to train ML models that classify interference types such as jamming and spoofing. Use online learning to adapt to new interference behaviors.

Starting Points. (1) Analyze the spectral and temporal characteristics of WiFi and Bluetooth IQ data, focusing on power spectral density and modulation types. Extract features like frequency occupancy, kurtosis, and skewness to establish a baseline for normal signal environments and distinguish interference types. (2) Simulate real-world interference by introducing narrowband jamming at specific frequencies, wideband noise, pulsed jamming bursts, network congestion using multiple devices on similar channels, or replaying I/Q samples of legitimate signals for spoofing. Use synthetic data or over-the-air data from SDRs for creating and validating these interference types. (3) Implement initial classifiers for signal differentiation, then incorporate online learning (e.g., passive-aggressive or RL) to continuously adapt the model to evolving interference patterns.

Future Directions. (1) Use imbalanced datasets. (2) Perform anomaly detection for novel interference types. (3) Require detection in low SNR conditions. (4) Incorporate overlapping interference signals for complex environments. (5) Introduce real-time processing constraints for low-latency classification. (6) Validate the system's performance in live environments using SDRs.

11.6.6 Project Idea: GNSS Signal Integrity and Spoofing Detection

Objective. Identify anomalies to detect GNSS spoofing or jamming incidents.

Approach. Analyze signal characteristics in GNSS I/Q data [41], and use ML models to classify normal vs anomalous data patterns.

Starting Points. (1) Explore the GNSS dataset to understand its structure and attributes, including bandwidths, GNSS coordinates, and automatic gain control settings. (2) Preprocess the I/Q data to ensure consistency across recordings and extract key features such as SNR, Doppler shifts, and phase changes. (3) Perform statistical analysis to establish benchmarks for normal signal behavior. (4) Implement a simple anomaly detection model (e.g., using isolation forests) to classify normal versus anomalous patterns.

Example Results. Barodzka [92] shows example results for detecting GNSS spoofing attacks, using a variety of detectable features.

Future Directions. (1) Implement real-time anomaly detection to handle continuous data streams, while striving for low-latency processing. (2) Utilize more complex models to enhance anomaly detection (e.g., [93]).

11.6.7 Project Idea: Multisensor Fusion for Localization

Objective. Develop a model that combines GNSS, inertial measuring unit (IMU), and lidar data to improve vehicle localization accuracy.

Approach. Using GNSS data from the University of Bristol's Combined Vehicular Measurements Dataset [41], develop a multisensor fusion model that integrates GNSS, IMU, and lidar data to enhance vehicle positioning accuracy, particularly in urban environments where GNSS signals may be weak or obstructed.

Starting Points. (1) Clean and synchronize the GNSS, IMU, and lidar data from the GNSS dataset. (2) Develop a multisensor fusion model (e.g., using Kalman or particle filtering) to enhance vehicle positioning in urban environments. (3) Evaluate the model's performance against ground truth data.

Future Directions. (1) Optimize the fusion algorithm for real-time, low-latency processing in dynamic environments. (2) Integrate DeepNets to enhance sensor fusion, improving performance in complex conditions and adapting to varying environments, such as weather and GPS-denied areas.

11.6.8 Project Idea: Multiplatform Route Planning

Objective. Assign drones to target locations.

Approach. Use a randomized search algorithm to solve the travelling salesman problem (TSP), wherein each salesman must visit every city with an efficient route. This problem is equivalent to assigning a drone to photograph target locations, or assigning game park rangers to patrol locations to track wildlife and poachers.

Starting Points. (1) Search for and implement a randomized search engine that solves the TSP effectively (e.g., ACO [94]). (2) Ensure that the code handles geographic coordinates. (3) Update the code to handle two or more salesmen. One approach might be to break the cities into two groups and then do TSP for each group; iterate to find the best groupings of cities [95–97].

Future Directions. (1) Consider resource usage, such as battery power and drone lifespan, to ensure that every plan is operationally feasible. (2) Ensure the generated plan allows for flexibility to support task reallocation and changing mission requirements. (3) Update to handle dynamic targets [98]. (4) Update the code so that if drones are lost or targets vanish, tasks can be reallocated among remaining drones. (5) Update the code so that some locations need to be visited by more than one drone. (6) Update the code to support time windows and deadlines, similar to the IJCAI TSP competition [99].

11.6.9 Project Idea: Decision Loop and Online Learning

Objective. Demonstrate agile decision making with a simple decision-control loop.

Approach. Create a decision maker that chooses appropriate responses for the current RF environment. Perform online learning for novel RF environments.

Starting Points. (1) Choose a simple scenario, for example four RF environments and four techniques, where each technique is designed to mitigate one of the environments. For example, these environments could be (i) free and clear comms, (ii) tone jammer, (iii) blinking jammer, and (iv) co-channel interference. The corresponding techniques could be (i) default operations, (ii) notch filter, (iii) redundant packets, and (iv) directional antenna. (2) Collect training data for those 16 points (all four techniques for all four environments). Section 4.3 presents examples of performance data for different environments. (3) Run a scenario as if you were controlling a platform, using Algorithm 10.1 to choose an action at each timestep.

Future Directions. (1) Extend to a more complex scenario, where environments overlap, and multiple techniques are applicable and may be combined. (2) Extend to doing ES, EP, and EA actions. (3) Extend to performing multiobjective decisions. (4) Incorporate a delayed reward, such as mission survivability, in the utility

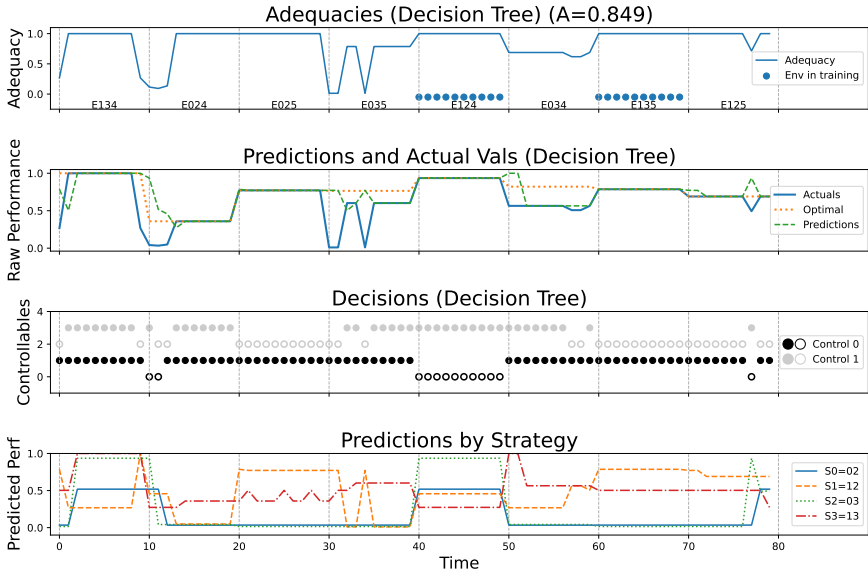


Figure 11.3 A decision tree achieves an Adequacy score of 0.849 by incrementally learning new environments. The model was pretrained with $k = 2$ environments (E124 and E135). Each of the eight environments is present for ten time units; the decision maker can choose one of four strategies. (Compare to Figure 7.8 for SVMs and $k = 0$.) *Top row:* Adequacy at each time step, and what environments were used for a priori training. *Second row:* Predicted performance of the chosen strategy, actual performance of the chosen strategy, and optimal performance of the best strategy. *Third row:* Control decisions. *Bottom row:* Predictions of how well each of the four strategies will perform.

function. (5) Do not pretrain on all of the data. Instead, update the model at every timestep within the loop to incrementally learn during execution. Section 4.3 describes the concept of learning action performance. (6) Add exploration to the action choices: Do not always choose the strategy with the highest-expected performance. (7) Only retrain when necessary. (8) Evaluate using ablation tests to determine how much, and which, prior training data is necessary (Section 10.8.4).

Example Results. Figure 11.3 shows example results for a decision tree that must learn how to handle novel environments. The model was pretrained on two of the eight environments it encounters.

11.6.10 Project Idea: EW Decision Making with 5G Technologies

Objective. Enhance decision making in EW by applying 5G technologies, such as network slicing and dynamic spectrum access, for resource optimization in contested environments.

Approach. Apply RL to optimize spectrum allocation and improve decision making in contested RF environments, utilizing real or synthesized datasets. Focus on dynamically allocating resources, adapting to changes in the network environment, and enhancing spectrum efficiency while minimizing interference.

Starting Points. (1) Simulate interference and resource allocation in 5G environments using a closed-loop framework (Section 11.5.2). Apply baseline strategies for handling interference such as power control and frequency hopping. (2) Use RL to optimize real-time spectrum allocation [100–102]. Gradually introduce dynamic spectrum access and network slicing. (3) Enhance adaptability by introducing dynamic factors like mobility, interference, and network changes. Focus on improving real-time decision making to minimize interference and boost spectrum efficiency.

Future Directions. Develop some of the full-sized 5G projects outlined in Call-out 11.2. (1) Investigate self-organizing networks for autonomous resource management in EW systems. (2) Explore game-theoretic approaches for multinode interference mitigation in 5G-based EW networks. (3) Leverage 5G beamforming to enhance resource allocation models in high-interference environments.

11.6.11 Project Idea: Augment Existing Data

Objective. Augment data to increase the diversity of the training data set.

Approach. Use both domain-independent and domain-dependent approaches to augment a very small dataset, similar to Section 8.3.3. Determine the impact of types of augmentation and the amount of augmentation on the results.

Starting Points. (1) Obtain an I/Q dataset for SEI or RF fingerprinting. (2) Import the data, and downselect to a handful of classes. (3) Select a few examples from each class to become the training set. (4) Train and evaluate ML model, as shown in Project 11.6.1. (5) Develop a few domain-independent augmentation techniques, retrain the model, and re-evaluate. (6) Develop a few domain-dependent augmentation techniques, retrain the model, and re-evaluate.

Example Results. Figure 8.7 shows the impact of three domain-independent augmentation techniques for ADS-B data.

Future Directions. (1) Reduce the number of I/Q samples on the results. What is the minimum number of I/Q samples required to get a reasonable result, for the number of classes you chose? Does it matter whether those I/Q samples come from signal

Callout 11.2 5G use cases and solutions have direct application to EW.

5G is the fifth generation of commercial cellular networks. Developed by the 3rd Generation Partnership Project (3GPP) standardization body, it is a new global wireless standard that could be transformative as it promises significantly higher data rates (multi-Gbps peak data speeds), lower latency (in milliseconds), ubiquitous connectivity, and higher reliability than 3G and 4G technologies. 3GPP defines three primary 5G “New Radio” use cases:

- *Enhanced mobile broadband (eMBB)*. Large amounts of data will be transmitted at much higher throughputs compared to 3G and 4G. eMBB will address bandwidth-hungry applications, such as massive video streaming and virtual/augmented reality [103].
- *Ultra-reliable & low-latency communications (URLLC)*. Also known as mission-critical communications, URLLC will provide a stable network and the lowest possible latency (in milliseconds) to initiate connectivity (e.g., tactile internet, autonomous vehicles (cars and drones), and a low latency of 1 millisecond or less for collision avoidance).
- *Massive machine-type communications (mMTC)*. Sets the foundation for the IoT. mMTC will allow machines (up to one million devices per 1 square km) to communicate with one another while requiring only minimal human involvement (e.g., industrial applications, metering, or massive sensor networks).

Moreover, new 5G features, such as the dynamic air interface, network function virtualization, and network slicing, introduce additional system design complexity and optimization requirements to address challenges related to network operation and maintenance. Consequently, ML has recently come back into focus in the field of communications for its potential to address these challenges that cannot be resolved with traditional methodologies. For example, 3GPP and International Telecommunications Union (ITU) have both proposed 5G research projects involving various AI/ML techniques. You et al. [104] discuss four problems in 5G whose solutions have direct application to EW:

- *Network resource allocation*: The 5G new radio orthogonal frequency-division multiplexing (OFDM) resource block allocation is much more complex than that of 4G long-term evolution (LTE) because of the required support for the three aforementioned use cases. RL performs resource block allocation [104] and 5G network slicing [105].

You et al. [104] discuss how to achieve energy-efficient beamforming in a massive multiple-input multiple-output (MIMO) system by finding the beamforming matrix, which introduces minimum power amplifier nonlinearities within a large solution space. The RNN recursively learns about the nonlinearities of the power amplifiers and finds suitable neuron weights that satisfy two requirements: (1) zero forcing (ZF) beamforming, meaning minimal multiuser interference, and (2) minimum overall nonlinear distortion. The RNN models the nonlinearity of the PA array, which then optimized to minimal transmitted power while providing a ZF solution. The RNN then notifies the 5G system on how to set its ZF beamforming weights [103].

Other ML techniques incorporate context-awareness for resource allocation [106], game theory to optimize power control for users from multiple cells that are assigned the same resource block [107], and managing and orchestrating cellular network resources [108].

- *Self-organizing networks (SONs)*: SON is a new way of network management that provides intelligence in the operation and maintenance of the network. 3GPP introduced SON as a key component of 4G LTE networks. 3GPP divides SON solutions into three categories: self-configuration, self-optimization, and self-healing. With 5G, SON functions will have to improve due to network ultra-densification, dynamic resource allocation, and increased network complexity.

AI and ML techniques such as ACO, ANN, and genetic algorithms solve various SON functions such as new cell and spectrum deployment, automatic base station configuration, coverage and capacity optimization, cell outage detection and compensation [104, 109–111].

Gomez et al. [112] developed a root-cause analysis system that combines supervised and unsupervised learning techniques in three steps: (1) unsupervised self-organizing maps training, (2) unsupervised clustering, and (3) labeling by experts.

- *Uniform 5G baseband acceleration*: The 5G baseband signal processing comprises a series of signal processing blocks, including massive MIMO detection and polar codes for channel decoding. The increased number of baseband blocks results in more complex hardware designs and implementations. To speed up baseband signal processing, a uniform accelerator can be designed with factor-graphs-based belief propagation algorithms (applied to all the blocks), aided by DeepNets (e.g., [113, 114]).

- *Optimization of end-to-end physical layer communication:* O'Shea and Hoydis [115] propose an autoencoder-based end-to-end PHY optimization method. By interpreting a communications system as an autoencoder, the authors propose a novel approach of thinking about communications system design as an end-to-end reconstruction task that seeks to jointly optimize transmitter and receiver components in a single process.

The demonstrated solutions to the 5G use cases map to problems in the communications, radar, and CogEW across many operational domains.

onset, or from the signal once it is stable? (2) Determine the impact of using edge cases vs common cases in the initial training set. Can you create edge cases? (3) Develop autoencoders or GANs to create fake data (Section 3.1.2).

11.6.12 Project Idea: Evaluate ML Models

Objective. Compare performance of ML models.

Approach. Every model has advantages and disadvantages; evaluate a set of models based on the criteria that matter to your mission. Section 3.9 describes some of the trade-offs, and Callout 10.2 lists some of the metrics.

Starting Points. (1) Select a learning problem and associated dataset. (2) In your ML toolkit of choice, compare the performance of each model. MATLAB has apps to assist with classification and regression [116, 117], and scikit-learn examples are easy to find, (e.g., [118, 119]). Online examples of anomaly detection and clustering are also common. (3) Extend your evaluator to look at other metrics, particularly time and memory. If different types of errors matter, then be sure to examine the detailed results (Section 10.8.2.3).

Example Results. Figure 10.10 shows the ablation results for 10 different ML models. Figure 11.4 shows timing results for various ML models using scikit-learn and TensorFlow, including a 3-layer ANN trained on 100 epochs. Figure 11.4 (a) shows the time to train the model against nRSME (80/20 train/test data); good results have low nRMSE¹ and low time. Figure 11.4 (b) shows the time it takes to train the model against Adequacy in an incremental learning scenario; good results have high Adequacy and low time (see Section 10.8.4). Time in this figure shows the cumulative learning time for the entire 80-sample scenario corresponding to Figure 11.3 when $k = 4$. The results show that both decision trees and SVMs with the Pearson VII kernel [12] are good choices. Extra trees and random forest

1. nRMSE = 0.0 is a perfect model, nRMSE > 1.0 is extremely bad. See Section 10.8.1.

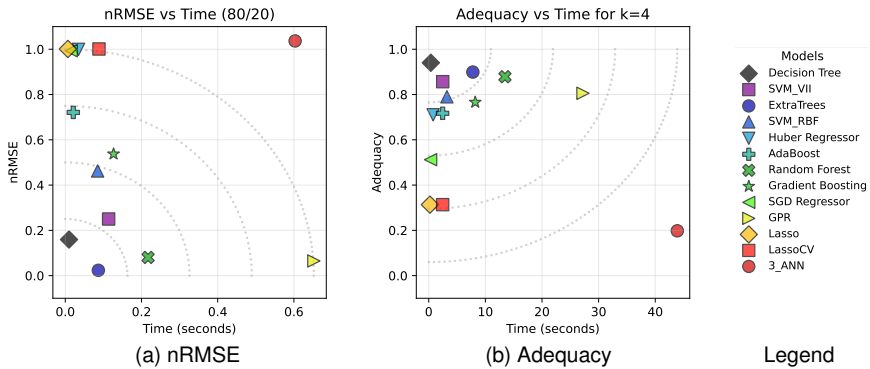


Figure 11.4 ML models must be evaluated with at least accuracy and training time. (a) nRMSE Accuracy vs Time for a static model trained with an 80/20 train/test split. $nRMSE=0.0$ is a perfect fit. (b) Adequacy vs Time for a dynamic model that learns in mission starting with $k = 4$ pretrained environments out of $n = 8$ total environments. $Adequacy=1.0$ is optimal.

perform well for both nRMSE and Adequacy, but training time is significant. The 3-layer ANN does not perform well for any of the three metrics because there is not enough data either during pretraining or during the run.

Future Directions. (1) Test your methods within an online setting, where the model must learn incrementally using RL. (2) For the algorithms that perform better, conduct ablation tests to determine impact of the amount of prior training data.

11.6.13 Project Idea: Implement a DeepNet on an FPGA

Objective. Implement a DeepNet on an FPGA.

Approach. Train a DeepNet that can be used for inference at hard real-time on a small embedded system.

Starting Points. Various tutorials exist online with more details (e.g., [120–123]), as do examples with RF data (e.g., [124–126]). (1) Build a simple DeepNet on some representative data. (2) Determine the throughput, latency, and memory constraints of the FPGA. (3) Optimize the model to fit within the FPGA constraints. This step will likely involve pruning, quantization, compression, or distillation (Section 3.7). (4) Compile the model into an FPGA-compatible format. (5) Implement and test.

Future Directions. (1) Explore other ML techniques on the FPGA (e.g., [127, 128]). (2) Make the DeepNet more complex, so that multiple optimizations are required to get the model to fit. If the model still doesn't fit, consider layer-wise partitioning, pipeline parallelism on multiple FPGAs, or offloading parts of the computation to

the CPU. (3) Use the results of the model to perform decision making. (4) Partition the model so that the last layer resides in the CPU, and do online learning of the weights, as in Project 11.6.9.

11.6.14 Project Idea: Data Compression

Objective. Compress I/Q data using ML models.

Approach. Compress I/Q data for on-board storage and later replay or post-test analysis, and optionally near real-time over-the-air transmission.

Starting Points. (1) Select an I/Q dataset. (2) Explore ML-based compression techniques such as dimensionality reduction, autoencoders, or Kohonen networks (e.g., [129–132]).

Future Directions. (1) Implement a compression “dial” that dynamically adjusts the level of data compression to achieve the desired balance between file size and quality, similar to the way JPEG compression varies from level 0 (maximum compression, lower quality) to level 9 (minimum compression, higher quality). (2) Dynamically change the compression level according to the importance of the observed data. Importance will change according to the mission, for example, a novel emitter may be more important than a known emitter in an unusual location. (3) Augment the compressor to identify key events for near real-time reporting. Report the event. If there is available bandwidth, compress the relevant I/Q data to fit within the bandwidth. (4) Examine additional concepts such as compressive sensing and functional compression (e.g., [133–136]).

11.7 Conclusion

Creating a CogEW system is conceptually easy to grasp. Taking small incremental steps to incorporate cognitive concepts in an existing system will ensure success of the final product. Don’t aim for a fully cognitive system (Figure 1.4) in a single step; choose the most brittle piece that needs to be replaced with a more robust empirical or heuristic method. Ask yourself:

- Does your system need better situation assessment? Deeper understanding of the RF environment, the anomalies, and the intent of the emitters?
- Does your system need better decision making? Something that can adapt to changing conditions and surprises?
- Does your system need to learn from its mistakes?

Each small step is a step in the right direction.

We don't need to wait for a 'boil the ocean' 'big bang' 'all in' new network of the future to get software defined networking (#SDN) into tactical systems. We can layer it on top of what we have today.

— Tim Grayson (DARPA), 18 Dec 2020, referring to #DyNAMO [137]

References

- [1] Kasser, J., "Guidelines for Creating Systems," *Journal of Systems Engineering*, Vol. 38, No. 4, 2015. DOI: 10.1002/j.2334-5837.2015.00088.x.
- [2] Ross, R., M. Winstead, and M. McEvilly, "Engineering Trustworthy Secure Systems," National Institute of Standards and Technology (NIST), Tech. Rep. NIST SP 800-160 Vol.1 Rev.1, 2022. DOI: 10.6028/NIST.SP.800-160v1r1.
- [3] Leveson, N., *Engineering a Safer World: Systems Thinking Applied to Safety (Engineering Systems)*. Cambridge, MA: MIT Press, 2016.
- [4] Leveson, N., *An Introduction to System Safety Engineering*. Cambridge, MA: MIT Press, 2023. Online introductory references at <https://psas.scripts.mit.edu/>.
- [5] Mountain Goat Software, *User Stories*. Online: <https://www.mountaingoatsoftware.com/agile/user-stories>. Accessed 2020-09-28.
- [6] Rehkopf, M., *User Stories with Examples and Template*. Online: <https://www.atlassian.com/agile/project-management/user-stories>. Accessed 2020-09-28.
- [7] McDermott, P., et al., "Human-Machine Teaming Systems Engineering Guide," MITRE, Tech. Rep. AD1108020, 2018. Online: <https://apps.dtic.mil/sti/citations/AD1108020>.
- [8] National Institute of Standards and Technology (NIST), *Risk Management Framework*. Online: <https://csrc.nist.gov/projects/risk-management/about-rmf>. Accessed 2025-01-23.
- [9] Cohen, P., and A. Howe, "How Evaluation Guides AI Research: The Message Still Counts More Than the Medium," *AI Magazine*, Vol. 9, No. 4, 1988. DOI: 10.1609/aimag.v9i4.952.
- [10] Haigh, K. Z., et al., "Rethinking Networking Architectures for Cognitive Control," in *Microsoft Research Cognitive Wireless Networking Summit*, 2008. Online: <http://www.cs.cmu.edu/~khaigh/papers/haigh-msr08.pdf>.
- [11] Haigh, K. Z., et al., "Parallel Learning and Decision Making for a Smart Embedded Communications Platform," BBN Technologies, Tech. Rep. BBN-REPORT-8579, 2015. Online: <http://www.cs.cmu.edu/~khaigh/papers/2015-HaighTechReport-SO.sm.pdf>.
- [12] Üstün, B., W. Melssen, and L. Buydens, "Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel," *Chemometrics and Intelligent Laboratory Systems*, Vol. 81, No. 1, 2006. DOI: 10.1016/j.chemolab.2005.09.003.
- [13] Avionics Department, "Electronic Warfare and Radar Systems Engineering Handbook," Naval Air Warfare Center Weapons Division, Tech. Rep. NAWCWD TP 8347, 2013. Online: <https://tinyurl.com/navy-ew-2013>.
- [14] Jolliffe, I. T., *Principal Component Analysis*, 2nd Edition. New York, NY: Springer, 2002. DOI: 10.1007/b98835.
- [15] *Convex Optimization*. Cambridge University Press, 2004. Online: <https://web.stanford.edu/~boyd/cvxbook/>.
- [16] Kalman, R., "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, Vol. 82, No. 1, 1960. DOI: 10.1115/1.3662552.

- [17] *scikit-learn: Machine Learning in Python*. Online: <https://scikit-learn.org/stable/>. Accessed 2025-02-26.
- [18] Abadi, M., et al., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, 2015. Online: <https://arxiv.org/abs/1603.04467>.
- [19] MathWorks, *Statistics and Machine Learning Toolbox*. Online: <https://www.mathworks.com/products/statistics.html>. Accessed 2025-03-03.
- [20] R Core Team, *R: A Language and Environment for Statistical Computing*, Accessed 2025-03-03, R Foundation for Statistical Computing, Vienna, Austria, 2025. Online: <http://www.R-project.org/>.
- [21] University of Waikato, NZ, *WEKA: The Workbench for Machine Learning*. Online: <https://www.cs.waikato.ac.nz/ml/weka/>. Accessed 2025-03-03.
- [22] Géron, A., *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*. Sebastopol, CA: O'Reilly, 2019.
- [23] Boehmke, B., and B. Greenwell, *Hands-On Machine Learning with R*. Boca Raton, FL: Chapman Hall, 2019. Associated website <https://bradleyboehmke.github.io/HOML/>.
- [24] OpenAI, *Codex*. Online: <https://openai.com/index/openai-codex/>. Accessed 2025-02-17.
- [25] Cursor, *The AI Code Editor*. Online: <https://www.cursor.com/>. Accessed 2025-02-17.
- [26] National Institute of Standards and Technology (NIST) Science Data Portal, 2025. Online: <https://data.nist.gov/>. Accessed 2025-01-08.
- [27] *IEEE DataPort*, 2025. Online: <https://ieee-dataport.org/>. Accessed 2025-03-03.
- [28] Northeastern and Rice Universities, *RF Data Factory*, 2025. Online: <https://www.rfdatafactory.com/>. Accessed 2025-01-12.
- [29] *Kaggle*, 2025. Online: <https://www.kaggle.com/>. Accessed 2025-03-03.
- [30] *Google dataset*, 2025. Online: <https://datasetsearch.research.google.com/>. Accessed 2025-03-03.
- [31] Lang, P., et al., *A Comprehensive Survey of Machine Learning Applied to Radar Signal Processing*, 2020. Online: <https://arxiv.org/abs/2009.13702>.
- [32] Hall, T., et al., "Reference Datasets for Training and Evaluating RF Signal Detection and Classification Models," in *IEEE Globecom Workshops*, 2019. DOI: 10.1109/GCWkshps45667.2019.9024532.
- [33] O'Shea, T., and N. West, "Radio Machine Learning Dataset Generation with GNU Radio," *GNU Radio Conference*, Vol. 1, No. 1, 2016. Online: <https://pubs.gnuradio.org/index.php/grcon/article/view/11>.
- [34] Sankhe, K., et al., "ORACLE: Optimized Radio Classification through Convolutional neural Networks," in *INFOCOM*, 2019. DOI: 10.1109/INFOCOM.2019.8737463, Dataset available at <https://genesys-lab.org/oracle>.
- [35] Hanna, S., S. Karunaratne, and D. Cabric, "WiSig: A Large-Scale WiFi Signal Dataset for Receiver and Channel Agnostic RF Fingerprinting," *IEEE Access*, Vol. 10, 2022. DOI: 10.1109/ACCESS.2022.3154790.
- [36] MIT, *MIT Challenge Data*, 2024. Online: <https://rfchallenge.mit.edu/>. A collection of recorded Comms I/Q data on a busy channel used to develop and test machine learning algorithms. Accessed 2024-10-24.

- [37] Tu, Y., et al., “Large-scale Real-world Radio Signal Recognition with Deep Learning,” *Chinese Journal of Aeronautics*, Vol. 35, No. 9, 2022. Data accessed 2022-08-10 from <https://gitee.com/heu-linyun/Codes>. DOI: 10.1016/j.cja.2021.08.016.
- [38] *Radar Challenge 2024*, 2024. Online: <https://ieee-aess.org/radar-challenge/radar-challenge-2024>. Accessed 2025-01-08.
- [39] Tschimben, S., et al., *Wi-Fi and Bluetooth I/Q Recordings in the 2.4 GHz and 5 GHz Bands with Low-Cost Software Defined Radios*, National Institute of Standards and Technology, 2022. DOI: 10.18434/mds2-2731. Accessed 2024-11-11.
- [40] *The OpenSky Network*. Online: <https://opensky-network.org/>. Accessed 2025-03-03.
- [41] Pelham, T., *GNSS Combined Vehicular Measurements*, 2024. DOI: 10.5523/bris.33bzd4wbwhhg2cznap5gl3hns.
- [42] Bergin, J., and P. Techau, *High-Fidelity Site-Specific Radar Data Set*, 2002. DOI: 10.21227/H27Q0R. Accessed 2025-01-10.
- [43] Humphreys, T., et al., “The Texas Spoofing Test Battery: Toward a Standard for Evaluating GPS Signal Authentication Techniques,” in *ION GNSS*, 2012. Online: <https://radionavlab.ae.utexas.edu/texbat/>.
- [44] Suraj, *RF Signal Data*, 2023. Online: <https://www.kaggle.com/datasets/suraj520/rf-signal-data/data>.
- [45] *OpenRAN Gym*. Online: <https://openrangym.com/>. Accessed 2025-01-12.
- [46] SRC, Inc., *Datasets from the Synthetic Data Incubator for Cognitive and Tactical ELINT (SynDICAte)*. Online: <https://datasets.srcinc.com/>. Accessed 2025-02-12.
- [47] Huang, Z., et al., *RadDet: A Wideband Dataset for Real-Time Radar Spectrum Detection*, 2025. Online: <https://arxiv.org/abs/2501.10407>.
- [48] MathWorks, *RF Toolbox*. Online: <https://www.mathworks.com/products/rftoolbox.html>. Accessed 2020-12-13.
- [49] Reddy, R., et al., “Simulation Architecture for Network Centric Sensors and Electronic Warfare Engagements,” in *Interservice/Industry Training, Simulation, and Education Conference*, 2018. Online: <http://researchspace.csisr.co.za/dspace/handle/10204/10886>, Software available as a MATLAB module: <https://tinyurl.com/csisr-sewes>.
- [50] *Skydel GNSS Simulation Software*. Online: <https://tinyurl.com/skydel>. Accessed 2025-01-11.
- [51] *TorchSig*. Online: <https://torchsig.com/>. Accessed 2025-01-12.
- [52] *Scapy*. Online: <https://scapy.net/>. Accessed 2025-01-12.
- [53] *PyLayers (github)*. Online: <https://pylayers.github.io/pylayers/>. Accessed 2025-01-12.
- [54] Flowers, B., *Radio Frequency Machine Learning (RFML) in PyTorch: A Tutorial (github)*. Online: <https://github.com/brysef/rfml>. Accessed 2025-01-12.
- [55] *Riverbed Modeler*. Online: <https://www.riverbed.com/products/riverbed-modeler/>. Accessed 2025-01-12.
- [56] U.S. Naval Research Laboratory, *Extendable Mobile Ad-hoc Network Emulator (EMANE)(github)*. Online: <https://github.com/adjacentlink/emane/wiki>. Accessed 2025-01-12.
- [57] *NS3*. Online: <https://www.nsnam.org/>. Accessed 2025-01-12.
- [58] *OMNeT++*. Online: <https://omnetpp.org/>. Accessed 2025-01-12.
- [59] *QualNet Network Simulation Software*. Online: <https://tinyurl.com/qualnetsim>. Accessed 2025-01-12.

- [60] Information Systems Laboratories, *RFView: High-fidelity RF Signals and System Modeling*. Online: <https://rfview.islinc.com>. Accessed 2020-12-13.
- [61] Nguyen, H., et al., "Cognitive Radar Scheduler Architecture," in *IEEE Radar Conference*, 2024. DOI: 10.1109/RadarConf2458775.2024.10548605.
- [62] Spirent, *GSS7000 GNSS Simulator*. Online: <https://www.spirent.com/products/gnss-simulator-gss7000>. Accessed 2025-01-12.
- [63] Intel Simics: *Full System Simulation*. Online: <https://www.windriver.com/products/simics>. Accessed 2025-01-12.
- [64] *PathWave System Design (SystemVue)*. Online: <https://tinyurl.com/pathwave-systemvue>. Accessed 2025-01-12.
- [65] *Next-Generation Electronic Warfare Environment Generator (NEWEG)*, 2020. Accessed 2020-12-13. Wayback machine contains snapshot 2021-11-20 <https://web.archive.org/web/20211120073813/https://www.navair.navy.mil/nawctsd/sites/g/files/jejdrs596/files/2019-07/2016-neweg.pdf>.
- [66] *Combat Electromagnetic Environment Simulator (CEESIM)*, 2020. Online: <https://tinyurl.com/ems-ceesim>. Accessed 2020-12-13.
- [67] *Vector Signal Generators*. Online: <https://tinyurl.com/rohdeschwarz-vsg>. Accessed 2025-01-12.
- [68] *PXI Vector Signal Transceiver*. Online: <https://tinyurl.com/ni-pxi-vst>. Accessed 2025-01-12.
- [69] BlueHalo, *RFnest*. Online: <https://tinyurl.com/rfnest>. Accessed 2025-01-12.
- [70] *Digital RF Battlespace Emulator (DRBE)*. Online: <https://www.darpa.mil/research/programs/digital-rf-battlespace-emulator>. Accessed 2025-01-12.
- [71] *ARES Advanced Radar Environment Simulators*, 2025. Online: <https://tinyurl.com/mrcy-res>. Accessed 2025-01-12.
- [72] *FABRIC infrastructure*. Online: <https://portal.fabric-testbed.net/>. Accessed 2025-01-12.
- [73] *Open-Access Research Testbed for Next-Generation Wireless Networks (ORBIT)*. Online: <https://www.orbit-lab.org/>. Accessed 2025-01-12.
- [74] *Platform for Open Wireless Data-driven Experimental Research*. Online: <https://powderwireless.net/>. Accessed 2025-01-12.
- [75] Humphries, J., et al., "gr-pdw: An OOT Module for Pulse Descriptor Word (PDW) Generation," *GNU Radio Conference*, Vol. 9, No. 1, 2024. Online: <https://pubs.gnuradio.org/index.php/grcon/article/view/151>.
- [76] Task Group SET-227, "Cognitive Radar," NATO Science, and Technology, Tech. Rep. TR-SET-227, 2020. Online: <https://tinyurl.com/nato-set-227-cr>.
- [77] Research Task Group Systems, Concepts and Integration (SCI) 326 Panel, "Electronic Support (ES) Techniques Enabling Cognitive Electronic Warfare," NATO Science, and Technology, Tech. Rep. STO-TR-SCI-326, 2024.
- [78] Čabrić, D., *ML for Transmitter Authentication*, 2020. Online: <https://cores.ee.ucla.edu/research/rf-transmitter-fingerprinting-using-deep-learning/>. Accessed 2025-01-10.
- [79] Youssef, K., et al., *Machine Learning Approach to RF Transmitter Identification*, 2017. Online: <https://arxiv.org/abs/1711.01559>.
- [80] Youssef, K., et al., "Machine Learning Approach to RF Transmitter Identification," *IEEE Journal of Radio Frequency Identification*, Vol. 2, No. 4, 2018. DOI: 10.1109/JRFID.2018.2880457.

- [81] Kuzdeba, S., *Radio Frequency Machine Learning: A Practical Deep Learning Perspective*. Norwood, MA: Artech House, 2025.
- [82] Cousins, D., et al., "Understanding Encrypted Networks through Signal and Systems Analysis of Traffic Timing," in *IEEE Aerospace*, Vol. 6, 2003. DOI: 10.1109/AERO.2003.1235227.
- [83] Tilghman, P., and D. Rosenbluth, "Inferring Wireless Communications Links and Network Topology from Externals Using Granger Causality," in *MILCOM*, 2013. DOI: 10.1109/MILCOM.2013.219.
- [84] Ahmed, A., *4 Python Packages to Start Causal Inference and Causal Discovery*, 2024. Online: <https://awadrahman.medium.com/recommended-python-libraries-for-practical-causal-ai-5642d718059d>. Accessed 2025-02-24.
- [85] Barnett, L., *The Multivariate Granger Causality (MVGC) Toolbox*. Online: <https://www.mathworks.com/matlabcentral/fileexchange/78727-the-multivariate-granger-causality-mvgc-toolbox>. Accessed 2025-02-24.
- [86] Alvarez-Garcia, M., R. Ibar-Alonso, and M. Arenas-Parra, "A Comprehensive Framework for Explainable Cluster Analysis," *Information Sciences*, Vol. 663, 2024. DOI: 10.1016/j.ins.2024.120282.
- [87] Schmidt, S., et al., "RiftNeXt: Explainable Deep Neural RF Scene Classification," in *Workshop on Wireless Security and Machine Learning*, New York, NY, USA: ACM, 2021. DOI: 10.1145/3468218.3469045.
- [88] Wang, S., and R. Ferrús, "Extracting Cell Patterns From High-Dimensional Radio Network Performance Datasets Using Self-Organizing Maps and K-Means Clustering," *IEEE Access*, Vol. 9, 2021. DOI: 10.1109/ACCESS.2021.3065820.
- [89] Guilbert, M., C. Vrain, and T.-B.-H. Dao, *Towards Explainable Clustering: A Constrained Declarative-based Approach*, 2024. Online: <https://arxiv.org/abs/2403.18101>.
- [90] Gidariss, S., et al., *Tutorial on Leave Those Nets Alone: Advances in Self-Supervised Learning*, 2021. Online: <https://gidariss.github.io/self-supervised-learning-cvpr2021/>.
- [91] Milosheski, L., et al., "Self-supervised Learning for Clustering of Wireless Spectrum Activity," *Computer Communications*, Vol. 212, 2023. DOI: 10.1016/j.comcom.2023.10.009.
- [92] Barodzka, M., *Anomaly Detection in Raw GNSS Data for Time Server Protection Against Spoofing*, 2023. Online: <https://tinyurl.com/gnss-anomalies>. Accessed 2025-01-13.
- [93] Savolainen, O., et al., "GNSS Anomaly Detection with Complex-Valued LSTM Networks," in *Localization and GNSS*, 2024. DOI: 10.1109/ICL-GNSS60721.2024.10578405.
- [94] McGuigan, J., *Ant Colony Optimization Algorithm*. Online: <https://www.kaggle.com/code/jamesmcguigan/ant-colony-optimization-algorithm>. Accessed 2025-01-12.
- [95] Gao, S., J. Wu, and J. Ai, "Multi-UAV Reconnaissance Task Allocation for Heterogeneous Targets Using Grouping Ant Colony Optimization Algorithm," *Soft Computing*, Vol. 25, 2021. DOI: 10.1007/s00500-021-05675-8.
- [96] Zhang, L., Y. Zhu, and X. Shi, "A Hierarchical Decision-Making Method with a Fuzzy Ant Colony Algorithm for Mission Planning of Multiple UAVs," *Information*, Vol. 11, No. 4, 2020. DOI: 10.3390/info11040226.
- [97] Shayboub, M., et al., "An Efficient Framework for Sensor Data Collection by UAV Based on Clustering, Two-Fold Ant Colony Optimization and Node Grouping," *Information Technology and Control*, Vol. 53, No. 3, 2024.

- [98] Liu, S., et al., "Current Development and Future Prospects of Multi-target Assignment Problem: A Bibliometric Analysis Review," *Defence Technology*, Vol. 43, 2025. DOI: 10.1016/j.dt.2024.09.006.
- [99] Blik, L., et al., "AI for TSP Competition," in *IJCAI Workshop: Data Science Meets Optimization*, 2021. Online: <https://tspcompetition.com/TSP.Competition.pdf>.
- [100] Kim, H., and S. Ankireddy, *Safe RL Deployment for 5G (github)*. Online: <https://tinyurl.com/rl-for-5g>. Accessed 2025-01-26.
- [101] Gopal, S., *AdapShare: An RL-Based Dynamic Spectrum Sharing Solution for O-RAN (github)*. Online: <https://tinyurl.com/rl-for-oran>. Accessed 2025-01-26.
- [102] Mismar, F., *Deep Reinforcement Learning for 5G Networks (github)*. Online: <https://tinyurl.com/deep-rl-for-5g>. Accessed 2025-01-26.
- [103] Yao, M., et al., "Artificial Intelligence Defined 5G Radio Access Networks," *IEEE Communications Magazine*, Vol. 57, No. 3, 2019. DOI: 10.1109/MCOM.2019.1800629.
- [104] You, X., et al., "AI for 5G: Research Directions and Paradigms," *Science China Information Sciences*, Vol. 62, No. 2, 2019. DOI: 10.1007/s11432-018-9596-5.
- [105] Li, R., et al., "Deep Reinforcement Learning for Resource Management in Network Slicing," *IEEE Access*, Vol. 6, 2018. DOI: 10.1109/ACCESS.2018.2881964.
- [106] Bogale, T., X. Wang, and L. Le, *Machine Intelligence Techniques for Next-Generation Context-Aware Wireless Networks*, 2018. Online: <https://arxiv.org/abs/1801.04223>.
- [107] Wang, J., et al., "Distributed Optimization of Hierarchical Small Cell Networks: A GNEP Framework," *IEEE Journal on Selected Areas in Communications*, Vol. 35, No. 2, 2017. DOI: 10.1109/JSAC.2017.2658999.
- [108] Li, R., et al., "Intelligent 5G: When Cellular Networks Meet Artificial Intelligence," *IEEE Wireless Communications*, Vol. 24, No. 5, 2017. DOI: 10.1109/MWC.2017.1600304WC.
- [109] Wang, X., X. Li, and V. Leung, "Artificial Intelligence-Based Techniques for Emerging Heterogeneous Network: State of the Arts, Opportunities, and Challenges," *IEEE Access*, Vol. 3, 2015. DOI: 10.1109/ACCESS.2015.2467174.
- [110] Klaine, P., et al., "A Survey of Machine Learning Techniques Applied to Self-Organizing Cellular Networks," *IEEE Communications Surveys and Tutorials*, Vol. 19, No. 4, 2017. DOI: 10.1109/COMST.2017.2727878.
- [111] Pérez-Romero, J., et al., "Knowledge-based 5G Radio Access Network Planning and Optimization," in *International Symposium on Wireless Communication Systems*, 2016. DOI: 10.1109/ISWCS.2016.7600929.
- [112] Gómez-Andrades, A., et al., "Automatic Root Cause Analysis for LTE Networks Based on Unsupervised Techniques," *IEEE Transactions on Vehicular Technology*, Vol. 65, No. 4, 2016. DOI: 10.1109/TVT.2015.2431742.
- [113] Tan, X., et al., *Improving Massive MIMO Belief Propagation Detector with Deep Neural Network*, 2018. Online: <https://arxiv.org/abs/1804.01002>.
- [114] Liang, F., C. Shen, and F. Wu, "An Iterative BP-CNN Architecture for Channel Decoding," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 12, No. 1, 2018. DOI: 10.1109/JSTSP.2018.2794062.
- [115] O'Shea, T., and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, Vol. 3, No. 4, 2017. DOI: 10.1109/TCCN.2017.2758370.

- [116] MathWorks, *Classification Learner App*. Online: <https://tinyurl.com/matlab-classification>. Accessed 2025-01-14.
- [117] MathWorks, *Train Regression Models in Regression Learner App*. Online: <https://tinyurl.com/matlab-regression>. Accessed 2025-01-14.
- [118] *Classifier Comparison in scikit-learn*. Online: <https://tinyurl.com/scikit-learn-classification>. Accessed 2025-01-14.
- [119] *Comparing Regression Models*. Online: <https://tinyurl.com/sl-regression>. Accessed 2025-01-14.
- [120] MathWorks, *Prototype Deep Learning Networks on FPGA*. Online: <https://tinyurl.com/matlab-DN-FPGA>. Accessed 2025-01-15. One of the tutorials is for Modulation Classification.
- [121] *Vitis-AI Tutorials (github)*, 2023. Online: <https://github.com/Xilinx/Vitis-AI-Tutorials>. Accessed 2025-01-15. One of the tutorials is for RF Modulation Recognition.
- [122] Mahpour, A., *Building Neural Networks on FPGAs*, 2024. Online: <https://resources.altium.com/p/building-neural-networks-fpgas>. Accessed 2025-01-15.
- [123] Er1cZ, *A Getting Started tutorial on FPGA implement of CNN using OpenCL (github)*, 2018. Online: <https://tinyurl.com/DNtoFPGA-openCL>. Accessed 2025-01-15.
- [124] Soltani, S., et al., “Real-Time and Embedded Deep Learning on FPGA for RF Signal Classification,” in *MILCOM*, 2019. DOI: 10.1109/MILCOM47813.2019.9021098.
- [125] Jian, T., et al., “Radio Frequency Fingerprinting on the Edge,” *IEEE Transactions on Mobile Computing*, Vol. 21, No. 11, 2022. DOI: 10.1109/TMC.2021.3064466.
- [126] Keller, T., “On the Design of Reconfigurable Edge Devices for RF Fingerprint Identification (RED-RFFI) for IoT Systems,” M.S. thesis, Rice University, 2023. Online: <https://tinyurl.com/keller-2023-rffi>.
- [127] *hls4ml: Python for ML Using HLS*, 2025. Online: <https://fastmachinelearning.org/hls4ml/>. Accessed 2025-01-15.
- [128] Goswami, P., and D. Bhatia, “Application of Machine Learning in FPGA EDA Tool Development,” *IEEE Access*, Vol. 11, 2023. DOI: 10.1109/ACCESS.2023.3322358.
- [129] MathWorks, *Autoencoders for Wireless Communications*. Online: <https://tinyurl.com/matlab-rf-compression>. Accessed 2025-01-17.
- [130] Kaasaragadda, Y., A. Rodriguez, and S. Kokalj-Filipovic, “Can We Learn to Compress RF Signals?” In *Balkan Conference on Communications and Networking*, 2024. DOI: 10.1109/BalkanCom61808.2024.10557214.
- [131] Armstrong, J., and E. Fallon, “Dimensionality Reduction for Optimization of Radio Base Station Transmission Based on Energy Efficiency,” in *Conference on Information Technology and Management Science of Riga Technical University*, 2023. DOI: 10.1109/ITMS59786.2023.10317744.
- [132] Kärkkäinen, T., and J. Hänninen, “Additive Autoencoder for Dimension Estimation,” *Neurocomputing*, Vol. 551, 2023. Code at <https://github.com/TommiKark/AdditiveAutoencoder>. DOI: 10.1016/j.neucom.2023.126520.
- [133] Doshi, V., “Functional Compression: Theory and Application,” M.S. thesis, 2008. Online: <https://dspace.mit.edu/handle/1721.1/43038>.
- [134] Feizi, S., and M. Médard, “On Network Functional Compression,” *IEEE Transactions on Information Theory*, Vol. 60, No. 9, 2014. DOI: 10.1109/TIT.2014.2332464.

- [135] Sharma, S., et al., "Application of Compressive Sensing in Cognitive Radio Communications: A Survey," *IEEE Communications Surveys & Tutorials*, Vol. 18, No. 3, 2016. DOI: 10.1109/COMST.2016.2524443.
- [136] Saidutta, Y., A. Abdi, and F. Fekri, "Analog Joint Source-Channel Coding for Distributed Functional Compression using Deep Neural Networks," in *IEEE International Symposium on Information Theory*, 2021. DOI: 10.1109/ISIT45174.2021.9517797.
- [137] Grayson, T., *LinkedIn Post: #DyNAMO*. Accessed 2020-12-19; link no longer available.

Acronyms

Acronym	Definition
ACO	Ant colony optimization
ADS-B	Automatic dependent surveillance-broadcast
AI	Artificial intelligence
ANN	Artificial neural network
API	Application programming interface
ASIC	Application-specific integrated circuit
BBN SO	BBN Strategy Optimizer
BDA	Battle damage assessment
BER	Bit error rate
CNN	Convolutional neural network
CPM	Critical path method
CPU	Central processing unit
CRN	Cognitive radio network
DSA	Dynamic spectrum access
DoD	Department of Defense (USA)
EA	Electronic attack
EBM	Electronic battle management
ECM	Electronic countermeasures
EMS	Electromagnetic spectrum
EP	Electronic protect
ES	Electronic support
EW	Electronic/electromagnetic warfare
EWO	Electronic warfare officer
FPGA	Field-programmable gate array
GA	Genetic algorithm
GAN	Generative adversarial network
GNSS	Global navigation satellite system
GPU	Graphics processing unit
HTN	Hierarchical task network

Acronym	Definition
HWIL	Hardware in the loop
I/Q	In-phase and quadrature
IoT	Internet of Things
kNN	k-nearest neighbor
LLM	Large language model
MANET	Mobile ad-hoc network
MCTS	Monte Carlo tree search
MDP	Markov decision process
ML	Machine learning
MOE	Measure of effectiveness
MOP	Measure of performance
MOS	Measure of suitability
nRMSE	Normalized root mean square error
PCA	Principal component analysis
Pd	Probability of detection
Pfa	Probability of false alarms
POI	Probability of intercept
RF	Radio frequency
RL	Reinforcement learning
RMSE	Root mean square error
RNN	Recurrent neural network
SDR	Software-defined radio
SEI	Specific emitter identification
SNR	Signal-to-noise
SUT	System-under-test
SWaP	Size, weight, and power
SVM	Support vector machines
T&E	Test and evaluation

About the Authors

Dr. Karen Zita Haigh is a consultant and evangelist for AI and ML for embodied embedded systems. Dr. Haigh regularly presents to academic and military communities worldwide. Formerly at Honeywell, BBN, L3, and Mercury Systems, Dr. Haigh has worked on safety-critical complex systems requiring rapid real-time decision making and learning, including cognitive RF systems, smart homes, cybersecurity, oil refineries, and space systems. Dr. Haigh is a Fellow of the Institute of Electrical and Electronics Engineers (IEEE) for contributions to closed-loop control of embedded systems, and a Fellow of the Asia-Pacific AI Association (AAIA) for achievements in smart homes. She is a life member of the Association of Old Crows. She earned her undergraduate degree in Honours Computer Science from the University of Ottawa, and her Ph.D. in Computer Science from Carnegie Mellon University focusing on AI and robotics. Dr. Haigh is a pioneer in three different fields that are now common across the globe:

- *Autonomous vehicles: Closed-loop planning and ML on autonomous robots.* Her Ph.D. work at CMU was the first to implement AI-style symbolic planning on real robot hardware and use ML to update planning models, forming the first complete closed-loop system of an autonomous robot. This capability is crucial to all autonomous vehicle work conducted globally.
- *Smart homes for elder care: Passive behavior monitoring to assist the elderly.* As principal investigator for Honeywell's Independent LifeStyle Assistant™ (ILSA), she developed an intelligent, adaptive home automation system integrating sensors, medical devices, and smart appliances for safe independent living. ILSA's multiagent system incorporated a unified sensing model, probabilistic situation awareness, intent recognition, HTN response planning, real-time action control, ML, and human factors. It was the first system of its kind, and led to a strategic Honeywell business acquisition.
- *Cognitive RF systems: ML to control complex multiobjective communication systems.* Dr. Haigh has worked in cognitive RF systems for over 20 years. She designed the cognitive controller for the DARPA's Adaptive Dynamic Radio Open-source Intelligent Team (ADROIT), the first known real-world

system (not simulation) to use ML to dynamically control a mobile ad hoc network (MANET). She designed the cognitive engine for DARPA's CommEx program, optimizing network performance amid previously unknown interference and jamming. CommEx was the first system with real-time in-mission learning for embedded EP.

This is Dr. Haigh's fourth book, following *The Dinner Co-op Recipe Collection* (1997), *Scripting Your World: The Official Guide to Scripting in Second Life* (2008), and the first edition of this book *Cognitive EW* (2021). She can be reached through her website, <https://sites.google.com/site/kzhaigh>.

Julia Andrusenko is a senior wireless communications engineer at Rampart Communications. Previously, she was Chief Engineer of the Mission Critical Communications group at the Johns Hopkins University Applied Physics Laboratory (JHU/APL). She has bachelor's and master's degrees in Electrical Engineering from Drexel University.

Ms. Andrusenko has over 20 years of experience in communications theory, wireless networking, satellite communications, RF propagation prediction, communications system vulnerability, simulation of communications systems, evolutionary computation, genetic algorithms, MIMO, and mmWave technologies. She also has extensive experience developing EW methodologies for various advanced commercial communications systems and military data links. She has authored several technical papers and coauthored two books: *Wireless Internetworking: Understanding Internetworking Challenges* (2013) and the first edition of this book, *Cognitive Electronic Warfare: An Artificial Intelligence Approach* (2021).

Ms. Andrusenko is a senior member of IEEE, a member of the IEEE Communications Society, and a voting member of the IEEE 1900.5 Working Group on Policy Language and Architectures for Managing Cognitive Radio for Dynamic Spectrum Access Applications. She has served as session chair, organizer, technical reviewer, invited speaker, and panelist at conferences. She has also contributed to the steering committee for the annual IEEE/APL 5G Technologies for First Responder and Tactical Networks Workshops, and co-led tutorials on mmWave technologies, AI applications to military networks, and cognitive EW at numerous Military Communications conferences.

Index

3GPP, 17, 113, 358
5G, 10, 17–19, 99, 118, 152, 221, 347, 349, 357–360
 Key performance indicators, 113
 Localization, 41, 99, 112, 113
 Network resource allocation, 358
 Use cases, 358–360
5G (application), 17
5G (example), 357–360
5G base station deployment (application), 359
5G baseband acceleration (application), 359
5G cell outage detection (application), 359
5G coverage optimization (application), 359
5G for asset tracking (application), 113
5G for first responders (application), 113
5G for traffic monitoring (application), 113
5G localization (application), 19, 112, 113
5G localization metrics (example), 113
5G massive machine-type communications (application), 358
5G mobile broadband (application), 358
5G physical layer communication (application), 360

5G ultra-reliable communications (application), 358
5G vulnerabilities (application), 118
6G, 10, 17, 149
6G (application), 17

A

A2/AD (application), 216
AADL, 321
Ablation testing, 78, 106, 260, 289, 292, 294, 315–319, 323, 328, 341, 356, 360, 361
Accelerate RL (application), 146
Accuracy, *see* Evaluation
ACO, *see* Optimization technique
Acoustic, 114, 290, 307
Action preconditions (application), 217
Action selection, 229
Action selection (application), 251
Action uncertainty, 180
Actionable features, 103
Actions, *see* Controllables
Active communication, *see* Planning
Active decoys (application), 144
Active learning, *see* RL concept
Active sensing, *see* Planning
Activity recognition (application), 119
Ada, 278, 280
Adaptability, 56, 71, 297, *see also* Evaluation
Adaptive jamming (application), 74
Adaptive vs cognitive, 4, 314, 315

- Adequacy, *see* Evaluation
- Admission control (application), 189
- ADROIT, 23, 27, 273–276
- ADS-B data, 263, 264, 346, 347, 351
- Adversarial attack generation (application), 155
- Adversarial attacks, 62, 208, 209, 218, 259, 263
- Adversarial data generation (application), 73, 74
- Adversarial plan recognition (application), 119, 121
- Adversarial sensor gain (application), 212
- Adversarial SINR (application), 212
- Adversarial testing, 242, 303, 323, *see also* ML technique: GANs
- Adversarial tracking ability (application), 212
- Adversary monitoring (application), 190
- Aerial mapping (application), 113
- AESA antenna scheduling (application), 152
- Agent, 4, 8, 9, 15, 18, 114, 119, 121, 149, 154, 158, 182, 191, 192, 197, 225, 228, 229, 289, 290, 343
- Agile development, 337, 338, 340
- AGREE, 321
- AI, 1, 2, 6–9, 13–15, 23, 32, 55, 75, 77, 81, 83, 84, 135, 138, 141, 142, 149, 191, 192, 242, 254, 281, 296, 336, 338–340, 342, *see also* Decision making; HMI; ML concept; ML technique
- enabled, 25, 241, 246, 337
- and EW terms, 14
- Domain agnostic, 2, 26, 27, 32, 121, 244, 266, 325
- AI, *cont.*
- Nonsymbolic, 66
- Symbolic, 57, 66, 67, 229
- vs cognitive, 4, 5
- vs traditional, 56, 57, 66, 74, 121, 141, 340, 342, 343, *see also* Hybrid approaches
- Air campaign (application), 179
- Air support mission scheduling (application), 174
- Air tasking orders (application), 174
- Air traffic control (application), 112
- Air traffic control (example), 301
- Air traffic safety (application), 117
- Air traffic safety metadata (example), 252, 253
- Air-air refueling (application), 174
- Airborne GMTI dataset, 347
- Aircraft carrier scheduling (application), 179
- Aircraft taxiing (application), 321
- Airline safety (application), 68
- Aleatoric uncertainty, 324
- Algorithm selection (application), 72
- AlphaGo, 120
- Ambiguity, 21, 23, 51, 112, 115, 119, 184, 185, 210, 303, 353
- Amplifier design (application), 145
- Analytic hierarchy process, 211, 212
- Anderson, Martin quote, 82
- ANNs, *see* ML technique
- Anomaly detection (application), 61, 73, 74, 76, 81, 97, 112, 115, 116, 354
- Ant colony optimization, *see* Optimization technique
- Antenna design (application), 144, 146

- Antenna fault detection (application), 146
 - Antenna placement (application), 145
 - Anti-jamming (application), 17, 19, 74, 75, 144, 155, 176, 190, 225, 228, 344
 - Antipoaching (application), 16
 - Antispoofing (application), 17
 - Antiterrorism (application), 16
 - Anytime algorithms, 120, 154, 155, 159, 215, 225
 - API, 254, 273, 274, 277
 - Applicability domain (application), 40
 - Architecture
 - Broker, 273–277
 - Centralized, 155
 - Decentralized, 155
 - Distributed, 155
 - Firmware, 252, 255, 281, 282
 - Hardware, 20, 252, 254, 255, 281, 282, 293
 - Multiple timescales, 188
 - Platform agnostic, 26, 27
 - Publish/subscribe, 277
 - Security, 341
 - Separation of concerns, 275
 - Shared memory, 278, 279
 - Software, 20, 174, 188, 252, 254, 255, 273, 275, 278, 282, 293, 324, 341
 - Temporal requirements, 188, 189
 - Threads, 278–280
 - Architecture adaption (application), 71
 - Argumentation, 184, 196
 - ASIC, 281–283, 350
 - ASPI, 83
 - Asset tracking in 5G (application), 113
 - Assurance, *see* Evaluation
 - Augmented reality in 5G (application), 113
 - Australia, 83
 - Authority, 287
 - Autoencoders, *see* ML technique
 - Autonomous vehicle (example), 7
 - Autonomous vehicles (application), 17
 - Autonomy, 4, 7, 8, 42, 44, 75, 84, 99, 180, 320, 358
 - Adjustable, 196
 - Availability (application), 18
 - Azimuth resolution (application), 40
- ## B
- Bandwidth waste (application), 41
 - Baseband acceleration (application), 359
 - Battery lifetime (application), 41
 - Battle damage assessment, *see* BDA
 - Battle management, 180
 - Bayesian methods, *see* ML technique
 - BBN, 284
 - BBN Strategy Optimizer, 25–27, 64, 76, 109, 158, 217–219, 222, 223, 258, 278–280, 314, 315, 317, 318, 327, 342, 348
 - BBN Strategy Optimizer (example), 25–27, 64, 76, 109, 158, 217–219, 222, 223, 258, 278–280, 314, 315, 317, 318, 327, 342, 348
 - BDA, 11, 14, 18, 20, 21, 25, 40, 43, 47, 51, 63, 68, 103, 110, 119, 138, 207–215, 227, 302, 328
 - BDA (application), 18, 40, 63, 211, 302
 - BDA classification (example), 213
 - Beamforming (application), 74, 145
 - Bearing accuracy (application), 113
 - Behavior prediction (application), 112
 - Belval, Edouard quote, 280
 - Bias, *see* Data

Bit error rate, 47
 Bit error rate (application), 40, 41, 43, 44, 49, 50, 109, 210, 275
 Black-box attacks, 263
 Border patrol (application), 16
 Bounded rationality, 142
 Box, George quote, 34

C

C-EOB, 255
 C/C++, 254, 278, 280, 281
 CADET (example), 180
 Cap'n Proto, 257
 Categorization (application), 81
 CEESIM testbed, 349
 Cellular traffic trace dataset, 347
 CEM, *see* Optimization technique
 Center for Strategic and Budgetary Assessments quote, 300
 Centralized architecture, 155
 CESMO, 255
 Chairman of the Joint Chiefs of Staff quote, 9
 Challenge
 Adversarial setting, 21
 Adversary goals, 190
 Ambiguous, 21
 Cognitive modeling, 192
 Complexity, 2, 3, 21, 57, 289, 343, 344
 Conflicting data, 110
 Constrained, 24
 Credit assignment, 228
 Data association, 111
 Data fusion, 192
 Data quality, 110, 241
 Distributed, 22, 343

Challenge, *cont.*
 Diverse, 22
 Dynamic, 20, 22, 343, 344
 Dynamic properties, 101
 Embedded code, 280, 336, 341, 345
 Heterogeneity, 23
 High-dimensional data, 57, 111
 Human assessment, 192
 Human-machine interaction, 192
 Integration with legacy systems, 277
 Interactions, 290
 Lack of training data, 21
 Learn after deployment, 290
 Macroscale, 23
 Multiobjective, 2, 24
 Mutual training, 192
 Non-linear problems, 57
 Non-stationarity, 229
 Novel emitters, 2, 3
 Operational timing, 111
 Partially observable, 21
 Policy, 24
 Resource constrained, 22
 Scalability, 228
 Scale, 22
 Spoofing, 101
 Temporal feedback loops, 21
 Timeline, 2, 3, 24
 Understanding limitations, 254
 Unknown emitters, 101
 User requirements, 24, 289
 Channel assignment (application), 144
 Channel selection (application), 225
 Characterization, 306
 Characterization, emitter (application), 97, 101
 Characterization, waveform (application), 104

- Characterize the environment (application), 15
- ChatGPT, 291
- China, 83–85, 284, 300
- Choose responses (application), 15
- CISA, 17
- Clarke, Arthur C. quote, 148
- Classification, *see* ML concept
- Classification, drones (application), 347
- Classification, emitter (application), 97, 100, 101
- Classification, pulse trains (application), 100
- Classification, radar (application), 77, 104, 105, 249, 319
- Classification, radar emitters (application), 104
- Classification, radio emitters (application), 104
- Classification, signal (application), 100, 228, 347
- Classification, specific emitter (application), 106
- Classification, waveform (application), 70, 104, 245
- Climate forecasting (application), 68
- Closed-form solutions, 57
- Closed-loop control, 149
- Closed-loop testing, 290, 298, 301, 324, 325, 327, 341, 348, 349, 357, *see also* Scenario driver
 - vs open-loop testing, 299
- CLT, 320
- Clustering, *see* ML concept, *see* ML technique
- Clustering RF environments (example), 34–36
- Clustering signals (application), 106
- Clutter suppression (application), 146
- Clutter-to-noise ratio (application), 40
- CMU, 85, 337
- CNNs, *see* ML technique
- Coalition battle planning (application), 179, 180
- Coast guard (application), 16
- Codex, 345
- Coexistence (application), 228
- Cognition, 8
 - Definition, 3
 - Deliberative, 3, 4
 - Interactive, 3–5
 - Iterative, 4
 - Learn after deployment, 15
 - Levels, 15
 - Loop, 5, 13, 273, 274
 - Spectrum, 5, 6, 304, 335
 - Transformative, 3, 4
- Cognitive
 - vs adaptive, 4, 314, 315
 - vs AI-enabled, 4
- Cognitive EW, 13, 14, 19, 23, 136, 249, 251, 273
 - Definition, 3
 - Scenario, 3
- Cognitive radar, 1, 18, 19, 186, 249, 252, 350, 360
- Cognitive radar identification (application), 197
- Cognitive radars (application), 143
- Cognitive radio, 1, 18, 23, 33, 38, 249, 360
- Cognitive radio networks, 18, 23, 41, 49, 50, 139, 143, 144, 146, 152, 190, 225, 228, *see also* MANET
- Cognitive radio networks (example), 50

- Cognitive system, 1, 3–5, 20, 192, 273, 287–291
- Collaboration, 4, 9, 11, 114, 144, 152, 154, 159, 191, 196, 197, 215, 245, 250, 297, 301, 337, 339, 341
- Collaborative radar surveillance (application), 152
- Command and control, 255
- Commander's intent, 180, 194
- Common electronic order of battle, 255
- Common operating picture (application), 98
- Comms
 - EA, 12, 13
 - EP, 12
 - ES, 12
 - vs Radar, 12, 44
- Comms and radar co-design (application), 139
- Comms efficiency (application), 49
- Comms EP (application), 26, 27, 50
- Comms EP data (example), 326
- Comms jamming (application), 72
- Comms link stability (application), 50
- Comms link stability (example), 50
- Comms performance (application), 19, 74
- Comms quality (application), 49
- Comms-aware radars (application), 152
- Communication
 - free learning, 158
 - Implicit, 159
- Communications planning (application), 186
- Competition
 - IEEE AESS, 347
 - Kaggle, 346
 - Optimization, 145, 147
- Competition, cont.
 - Planning, 176, 179
 - Signal classification, 104
 - Travelling salesman problem, 355
- Computational learning theory, 320
- Concept drift, *see* ML concept
- Confidence, 324
- Configuration (application), 31
- Confusion matrices, 307, 309–312
- Confusion matrix (example), 263, 264
- Consensus propagation, 157
- Consistency, *see* Data
- Constraints, 20, 22, 31, 41, 44, 47, 48, 76–80, 114, 137, 140, 147, 150, 151, 173, 175, 177–180, 186, 187, 189, 193, 196, 209, 221, 226, 249, 257, 258, 281, 320, 321, 342, 353, 361
- Context awareness (application), 112
- Controllables, 31, 32, 38, 39, 45, 49, 325
 - Examples, 10, 11, 13, 39, 49–51, 136, 141, 185, 210
- Cooperative communication (application), 146
- Cooperative electronic support measure operations, 255
- Cooperative games, 189
- Cooperative jamming (application), 144
- Cooperative localization (application), 159
- Cooperative operation (application), 114
- Coordinated transmissions (application), 188
- Coordination, 4, 9, 11, 12, 15, 16, 22–24, 39, 114, 118, 154–159, 173, 174, 180, 183, 185, 186, 188, 191, 195, 216, 341, 343, 344

Coordination time (application), 141
 Coreset, 259
 Cost (application), 18, 40, 41
 Counter-drone (application), 17
 Course of action planning (application), 180, 195
 Coverage (application), 41
 Covert communications (application), 152
 CPU, 25, 76, 224, 253, 277, 281–283, 350
 CPython, 278
 Creating predistortion filters (application), 145
 Credit assignment, 228
 Crisis management (application), 179
 Critical infrastructure (application), 17
 Critical infrastructure inspection (application), 113
 CRN, *see* Cognitive radio networks
 CRN collision management (application), 112
 CRN control overhead (application), 40
 CRN energy consumption (application), 112
 CRN management (application), 225
 CRN message traffic (application), 112
 CRN optimization (application), 50
 CRN performance (application), 144
 CRN routing (application), 352
 CRNs (application), 23, 62, 143
 Cross-entropy method, *see* Optimization technique
 CSBA quote, 300
 Cursor.AI, 83, 345
 Cyber operations planning (application), 195

Cyber red-teaming planning (application), 177
 Cyber red-teaming planning (example), 177
 Cybersecurity, 2, 9, 17, 117, 177, 190, 211, 249, 267, 290, 349
 Cybersecurity (application), 117, 190
 Cybersecurity and infrastructure security agency, 17
 Cycle
 Cognition, 5, 13, 274
 Data, 241, 242, 293
 FFTTEA, 13
 Learning assurance, 292
 OODA, 5, 13
 OOPDAL, 5
 Perception-action, 5, 229
 Radar processing, 106
 Sensing, acting, and execution monitoring, 208, 209, 230
 Testing, 291
 Train, test, evaluate for ML, 77, 78, 104, 105, 291, 293

D

DARPA, 180, 347, 349, 363

Data

a priori training data, 15, 16, 21, 26, 27, 55, 57, 58, 61, 64, 67, 69, 74, 77–81, 97, 101, 102, 104–108, 221–224, 226, 228, 247, 248, 253, 258–261, 263, 265, 266, 278, 290–292, 295, 297, 313, 315–318, 321, 323, 327, 328, 335, 341, 346, 351, 352, 355, 357, 361, *see also* ML concept: Training approach
 Accessibility, 242–244, 246

Data, cont.

- Adding noise, 261
- Anonymization, 266
- Augmentation, 61, 66, 69, 71, 74, 77, 80, 104, 248, 257, 259, 261–265, 294, 324, 357
- Authenticity, 112
- Availability, 112, 241–244
- Bias, 7, 61, 71, 74, 77, 110, 219, 243, 245–248, 251, 261, 278, 322
- Collected in-mission, 310
- Completeness, 242, 244, 245
- Consistency, 244, 246, 257
- Credibility, 115, 184, 242–245, 248, 249
- Curation, 110, 241–243, 245, 246, 249, 341
- Degradation, 245, 252
- Destruction, 265
- Diversity, 7, 27, 66, 80, 104, 144, 219, 225, 242, 247, 248, 253, 257–259, 261, 263, 265, 300, 305, 318, 325, 336, 346, 357
- Diversity technique, 260, 261
- DoD data strategy, 243
- Example, NEU RF fingerprinting meta-data, 253
- FAIR principles, 243
- Forgetting, 27, 66, 70, 257, 258, 265
- Fusion, 19, 98, 99, 102, 110–115, 159, 184, 211, 243, 249, 254, 344, 354
- Generation, 348
- Ground truth, 245, 316, 325, 326
- in embedded system, 257
- Integrity, 49, 243, 245, 248, 250, 322
- Interoperability, *see* Interoperability
- Invariance, 261

Data, cont.

- Lifecycle, 241, 242, 246, 293
- Limited, 8, 58, 64, 69, 70, 79, 80, 97, 221, 259, 263, 351
- Management, 25, 242, 257, 293
- Metadata, *see* Metadata
- Missing, 7, 70, 71, 79, 101, 246, 247, 265, 292, 306, 342
- Modeling, 242
- Ontology, *see also* Ontology
- Out of distribution, 69, 71, 100, 261, 263, 290, 292, 320
- Overfitting, 248
- Oversampling, 261
- Perturbation, 322
- Privacy, 74–76, 79, 257, 265–267
- Provenance, 115, 184, 242, 245, 246, 248
- Quality, 77, 241–246, 251, 306
- Regularization, 248, 260
- Reliability, 97, 112
- Requirements, 15, 244, 297, 340
- Safety, 242, 297
- Schema, *see also* Schema
- Security, 243, 248, 257, 265, 341
- Semantics, 242, 249, 250
- Size, 259
- Synthetic, 73, 251, 259, 261, 263, 297, 326, 336
- Timeliness, 244
- Traceability, 110, 243, 244, 248, 249, 300
- Understanding the problem, 241
- Verification, 292
- Data analysis (application), 195
- Data augmentation (example), 357
- Data compression (example), 362
- Data decoding (application), 283

- Data distribution service, 277
- Data fusion (application), 97, 283
- Data fusion (example), 354
- Data fusion information group, 111
- Data integrity (application), 49
- Datasets, 256, 263, 264, 278, 346–348
- DDS, 277
- Decentralized architecture, 155, 341
- Decision confidence (application), 40
- Decision making, 6–9, 14, 19, 22, 25, 26, 31, 33, 38, 40, 46, 51, 72, 135–138, 140, 181, 185, 187–190, 194, 207, 219, 223, 229, 243, 244, 266, 273, 274, 290, 302, 312–315, 325, 328, 341, 344, 348, 355–357, 362, *see also* Game theory; Planning; Reward hacking; Scheduling; Optimization
 - Bounded rationality, 142
 - Computation capability, 142
 - Game theory, 190
 - in humans, 142
 - Information available, 142
 - Time available, 142, 154, 215
- Decision making (example), 223, 355, 356
- Decision theory, 137
- Decision trees, *see* ML technique
- Decluttering (application), 3
- Deep RL in Python (example), 221
- DeepNet on FPGA (example), 361
- DeepNets, 104, *see* ML technique
- DeepSeek, 76, 83
- Defeating adversary tracking (application), 43
- Defensive counter air planning (application), 174
- Deinterleaving, 99, 296
- Deinterleaving (application), 74, 99, 118
- Demodulation dataset, 346
- Dempster-Shafer theory, 183
- Dendrogram, 36, 318, 319
- Deployability, 26, 75, 79, 120, 221, 246, 280, 288, 290, 291, 296, 297, 300, 335, 337–339, 341, 349, 350, *see also* Evaluation
- Design decision (application), 81
- Design of experiments, 304
- Design question, 19, 20, 186, 339–341
- Detect concept drift (application), 71
- Detecting attacks (application), 228
- Detection, 98, 99
- Detection (application), 98
- Detection accuracy (application), 41
- Development, 189, 340
 - Agile, 337, 338
 - Iterative, 153, 288, 292, 293, 301, 337, 339
 - Key steps, 335, 362
 - MLOps, 338, 339
 - of data pipeline, 336
 - Systems engineering, 337
- Development considerations, 339
- Device connectivity (application), 149
- Device optimization (application), 347
- Diagnosis (application), 68
- Differential privacy, 266
- Disrupt adversary communication (application), 74
- Disrupt adversary decision-making (application), 43
- Disrupt adversary radar (application), 50, 74
- Distributed architecture, 155, 341
- Distributed beam forming (application), 159

- Distributed computing fallacies, 23, 24
- Distributed jamming (application), 145
- Distributed scheduling (application), 159
- DistrO, 216
- DistrO (example), 216
- Diversity, 22, 38, 39, 222, 303, *see also*
 - Data
- DoD, 300
 - Army, 104, 180, 195
 - Chief Data and AI Office, 337
 - DARPA, 180, 347, 349, 363
 - Data strategy, 243, 244
 - DTEA, 289
 - Navy, 140
 - Planning doctrine, 174
- Domain of applicability, *see* ML concept: Domain of expertise
- Doppler resolution (application), 40
- DRBE testbed, 349
- Drone communications (application), 301
- Drone jamming (application), 301
- Drone surveillance (application), 249, 347
- Drone tracking (application), 144
- Dynamic spectrum access (application), 117, 141, 151
- E**
- EA, 1, 11–14, 18, 19, 22, 24, 25, 32, 39, 44, 98, 101, 110, 135, 136, 138, 141, 154, 184, 185, 207, 208, 210, 211, 215, 245, 305, 344, 355
 - Comms, 12, 13
 - Radar, 12, 13, 49–51, 139
 - vs EP, 2, 12, 32, 135
- EA effectiveness (application), 40, 50, 51, 211, 212
- Earth observation (application), 176
- EarthRanger, 16
- EASA, 289, 293
- EBM, 1, 11, 14, 19, 22, 25, 138, 173, 192, 208, 245
 - Effective detection range (application), 51
 - Effectiveness (application), 40, 211
 - Effects-based operations (application), 194
- Electric power (application), 158
- Electromagnetic order of battle generation (application), 195
- Electromagnetic spectrum, *see* EMS
- Electromagnetic warfare, 9, *see also* EW
- Electronic attack, *see* EA
- Electronic battle management, *see* EBM
- Electronic countermeasures (application), 152
- Electronic protect, *see* EA
- Electronic support, *see* ES
- Electronic vs Electromagnetic warfare, 9
- Electronic warfare, *see* EW
- ELEXIR, 120
- EMANE, 348
- Embedded code, 38, 75, 280, 341
- EMBM, *see* EBM
- Emitter avoidance (application), 147
- Emitter descriptions (example), 353
- Emitter descriptor words, 104
- Emitter recognition (application), 283
- Emitter sophistication (application), 103
- Emitter strategy (application), 103
- EMS, 3, 9–11, 13, 15, 17, 18, 21, 23, 34, 39, 174, 255, 283, 290, 300, 353
- EMSO, 9, 11, 18, 23, 255, 300

- Encryption, 266
- Energy per fix (application), 113
- Energy-efficient beamforming (application), 359
- Ensemble methods, *see* ML technique
- Ensemble methods (example), 213, 214
- Environment similarity (application), 34, 37
- Environments, 107, 108, 260, 265
 - Clustering, 35, 219, 310, 314, 318
 - Comms EP Example, 325, 327
 - Managing, 220
- EP, 1, 11–14, 19, 22, 24, 25, 32, 44, 103, 135, 136, 138, 141, 152, 154, 184, 185, 190, 207, 208, 210, 245, 305, 344, 355
 - Comms, 12, 18, 25–27, 49, 141, 142, 158, 222, 325, 348
 - Radar, 12
 - vs EA, 2, 12, 32, 135
- EP countermeasures (application), 344
- Epistemic uncertainty, 324
- ES, 1, 5, 9, 11–14, 20, 24, 25, 32, 33, 43, 44, 47, 97, 99, 101, 135, 136, 138, 141, 152, 182, 184, 185, 193, 196, 207–210, 215, 245, 249, 350, 355
 - and execution monitoring, 208
- Escape, 308
- European Aviation Safety Agency, 293
- Evaluation, 1, 5, 25, 77, 79, 145, 177, 209, 211, 251–253, 261, 292, 296, 300, 301, 304, 305, 315, 316, 318, 319, 323, 327, 328, 335–337, 339, 340, 351, 352, 356, 357, 360, 361
 - Ablation testing, *see* Ablation testing
 - Adaptability, *see* Adaptability
 - Adaptive sampling, 65, 304
 - Evaluation, *cont.*
 - Adversarial perturbations, 322
 - Adversarial testing, 72
 - Architecture, 277
 - Assurance, 2, 57, 78, 243, 244, 287–289, 304, 320, 321, 337
 - Classifiers, 305, 307, 309
 - Closed-loop testing, 288, 298, 341, *see also* Scenario driver
 - Confidence, 341
 - Confusion matrices, 105, *see* Confusion matrices
 - Cross-validation, 212, 214, 248, 316
 - Data, 322
 - Data management, 293, 294
 - Deployability, *see* Deployability
 - Design of experiments, 289, 304
 - Explainability, *see* Explainability
 - Fairness, *see* Fairness
 - Interoperability, *see* Interoperability
 - Key steps, 319
 - Learning goals, 80, 295, 341, *see also* Goals; Trade-offs
 - Learning process, 292–295, 319
 - Metric, 71, 80, 221, 293, 296–298, 341, *see also* Goals; Metrics; Trade-offs
 - Accuracy, 10, 19, 20, 24, 27, 43, 61, 62, 70, 73, 76, 77, 80, 81, 83, 97, 99, 101, 102, 107, 110, 114, 115, 193, 212–214, 228, 244–246, 252, 253, 259, 261–263, 284, 292, 297, 299, 301, 305, 307–309, 316, 320, 328, 335, 341, 345, 352, 354, 361
 - Adequacy, 223, 297, 311–314, 316–319, 328, 356, 360, 361
 - Balanced accuracy, 308

Evaluation, cont.

Metric, cont.

- Confidence, 40, 65, 78, 85, 110, 211, 222, 226, 254, 297, 320, 352
- Cost of errors, 307, 309
- Escape, 308
- F1 score, 295
- Flexibility to surprise, 40, 215
- Kullback-Leibler divergence, 226
- nRMSE, 295, 305, 306, 310, 328, 360, 361
- Overkill, 308
- Performance, 310
- Performance guarantees, 68
- Precision, 295, 308
- Predictability, 147
- Recall, 295, 308
- Robustness, 58, 62, 64, 70, 71, 77, 99, 114, 115, 146, 147, 229, 262, 288, 294, 297, 301, 323, 324, 339
- Scalability, 7, 56, 57, 79, 112, 113, 120, 148, 181, 228, 297, 300, 322, 337–339, 341–343
- Sensitivity, 72, 284, 324
- Stability, 76, 147, 222, 227, 297, 322, 324
- Surprise, 1, 40, 78, 288, 290–292, 294, 295, 302, 341
- Mixed-fidelity testing, 288, 298, 301
- Multifidelity testing, 299, 301
- of ANNs, 320, 324
- of complex models, 319
- of safety-critical software, 320
- Over time, 312
- Regression, 305
- Risk, 4, 74, 118, 180, 192, 194, 196, 242, 248, 287, 296, 336, 337
- Robustness, 139, 303, 346

Evaluation, cont.

- Safety, *see* Safety
- Scenario driver, *see* Scenario driver
- Security, *see* Security
- Smart sampling, 304
- Test coverage, 72, 304
- Test framework, *see* Scenario driver
- Third-party tools, 301
- Transferability, *see* Transferability
- Trust, 2, 25, 57, 67, 153, 178, 191, 192, 196, 208, 241, 252, 287, 291, 292, 297, 324, 336, 337, 340
- Uncertainty, *see* Uncertainty
- Validation, 2, 25, 77, 78, 250, 287–289, 291, 293–295, 319, 341
- Verification, 25, 57, 76, 80, 248, 288, 292–295, 300, 319–323, 338
- Vertically-integrated testing, 300
- with Metalearning, 72
- Evaluation of aircraft taxiing (example), 321
- EW, 9, 83, 84
 - and 5G, 19, 360
 - and AI terms, 14
 - and cognitive radar, 18, 360
 - and cognitive radio, 18, 360
 - and cybersecurity, 2
 - and distributed robotics, 15
 - and information warfare, 2
 - and ISR, 2
 - and PNT, 2
- Battle damage assessment, *see* BDA
- Civilian, 9, 15, 17
- Definition, 9
- Domains, 9
- Electronic attack, *see* EA
- Electronic battle management, *see* EBM
- Electronic protect, *see* EP

EW, cont.

Electronic support, *see* ES

EW performance (application), 40

EWPMPT playbook interface (example), 195

Execution monitoring, 7, 208, 209, 215, 321

Experimental design (application), 225

Explainability, 66, 79, 192, 193, 229, 297, 341, 353, *see also* Evaluation

Exploration guidance (application), 72

Exposure (application), 41

F

FABRIC testbed, 349

Fairness, 150, 246, 297, *see also* Evaluation

False target generation (application), 51

Falsification, 323

Feature construction, 193

Feature construction (application), 57

Feature engineering, 102

Feature estimation (application), 97

Feature extraction, 56

Feature extraction (application), 64, 69, 70, 99, 100, 353

Feature interactions, 56

Feature selection (application), 71, 103

Feature transformation (application), 71, 103

Features, 32, 33, 45, 80, 102–104, 109, *see also* Observables

Latent, 59, 68, 73, 79, 81, 97, 102, 104, 224, 245, 247

Few-shot learning, *see* ML technique

Find-fix-track-target-engage-assess, 13

Fire control (application), 43

FlatBuffers, 257

Flexibility, 57, 343

Flexible computation, 154

Flight plans and jamming (application), 144

Fortran, 280

Fourier transforms, *see* ML technique

FPGA, 38, 40, 224, 252, 254, 281–283, 350, 361

Frame error rate (application), 50

Fratricide minimization (application), 186

Free-space path loss, 66, 67, 102

Future value prediction (application), 55

Fuzzy logic, 184

G

Gürbüz et al. quote, 252

GA, *see* Optimization technique

Game theory, 47, 189, 190, 357

Action costs, 190

Adversarial, 121, 189

Adversarial situations, 190

Algorithmic, 190

and Multiobjective optimization, 190

and Optimization, 140, 152

Computational cost, 190

Zero-sum games, 190

GANs, *see* ML technique

Gaussian processes, *see* ML technique

Generalization, *see* ML concept

Generative AI, *see* ML concept

Genetic algorithms, *see* Optimization technique

GENI testbed, 349

Geolocation (application), 347

GMMs, *see* ML technique

GNSS, 17, 114, 347, 349, 354

GNSS simulator, 349
 GNSS vehicular measurements dataset, 347
 GNU radio, 254, 255, 349
 Goal recognition (application), 119
 Goals, *see also* Evaluation; Trade-offs
 Mission, 31, 178, 179, 188, 208, 209, 214, 215, 293, 295, 301
 Solution, 40, 71, 78, 292, 297, 340, 341
 Google dataset, 346
 GPS, 251, 347, 354
 GPS spoofing dataset, 347
 GPT, 74, 83
 GPU, 76, 253, 281–283
 Graceful degradation, 156
 Grayson, Tim quote, 363
 Grey-box attacks, 263
 Grey theory, 211, 212
 Ground moving target indicator (application), 112
 Ground truth data (example), 325, 326

H

Hanlon, Robert quote, 249
 Hardware, 242, *see* Architecture
 Heuristic learning (application), 152
 Heuristic Scheduling Testbed System, 175
 Heuristics, 32, 57, 66, 69, 72, 146, 147, 178, 183, 194, 229, 304
 Engagement, 19, 23, 147
 Expertise, 43, 57, 68, 102, 121, 174, 193, 211, 302, 319, 340
 Physics, 19, 23, 33, 67, 68, 102, 147, 209, 211, 214, 304, 342
 Science, 68

Hierarchical Task Networks, *see* Planning
 HMI, 25, 157, 174, 178, 191, 192, 243, 274, 340, *see also* Humans; User
 Adjustable autonomy, 196
 and Training, 291
 Argumentation, 196
 Commanders, 194
 EWOs, 194
 HMT engineering guide, 338
 Mission planners, 194
 Mixed-initiative planning, 196
 Playbook interfaces, 194, 195
 Roles, 192, 193
 System Designers, 193
 System designers, 193
 Tools, 194
 Understanding the humans, 196
 HMMs, *see* ML technique
 HMT engineering guide, 338
 Horne cognition mapping, 6
 HTN planning, *see* Planning
 HTN planning and scheduling (example), 180
 HTN planning for SEAD (example), 179
 Hubble Space Telescope, 175
 Hubble Space Telescope planning (example), 175
 Human-machine interface, *see* HMI
 Human-machine teaming, *see* HMI
 Humans, 2, 25, 178, 190–195, 216, 290, 291, 297, 335, *see also* User
 as sensors, 196
 Bounded rationality, 142
 Commanders, 194, 338
 Direct assessment, 324

- Humans, cont.
 EWOs, 11, 25, 119, 136, 157, 194, 209, 214, 338
 Learning, 219
 Mission planners, 138, 194, 338
 Requirements imposed by, 24
 Stakeholders, 25, 44, 192, 252, 289, 292, 297, 324, 337–339
 Supervision, 3
 System designers, 193, 267, 337, 338, 340
 HWIL testing, 299, 300
 Hybrid approaches, 57, 59, 64–69, 97, 99, 100, 102–104, 107, 117, 147, 193, 196, 212, 229
 Benefits, 67
 Design patterns, 69
 Hyperparameter tuning (application), 71, 72
 Hypersonics, 84
 Hypothesis formation (application), 250
- I**
 Identify novel emitters (application), 3
 IEEE datasets, 346, 347
 Image generation (application), 73
 Image processing (application), 76
 Image recognition (application), 71
 Image rotation (application), 76
 Implicit communication, 159
 IMU, 347, 354
 Incremental learning, *see* RL concept
 Incremental learning (example), 217, 361
 Information fusion (application), 159
 Information theory, 116
 Information uncertainty, 183, 219
 Information utility, 184
 Information warfare, 2
 Information warfare (application), 190
 Informed improvements, 156
 Infrared, 9, 10, 114, 290
 Integrated system design (application), 145
 Integrating sensing and comms (application), 62
 Integrity, 74, 321, *see also* Data
 Intent recognition, 118, 119, 196
 with game theory, 121
 Intent recognition (application), 97, 119
 Intent recognition (example), 120
 Interference (application), 41, 50
 Interference classification (application), 353
 Interference classification (example), 353
 Interference control (application), 228
 Interference detection (application), 74, 347
 Interference estimation (application), 197
 Interference mitigation, 26
 Interference mitigation (application), 17, 27, 75, 141, 355, 357
 Interference prediction (application), 228
 International Planning Competition, 176
 Internet of Things (example), 358
 Interoperability, 242–245, 249, 257, 297, 341, *see also* Evaluation
 Interval analysis, 322
 IoT, 16, 17, 75, 358
 IP stack, 33, 38, 39
 ISAR, 99, 347
 ISAR image generation (application), 99
 ISAR imaging for navigation dataset, 347
 ISIF, 112
 ISO, 244

ISR, 2
ITU, 358

J

JADC2, 173
Jammer localization (application), 146
Jammer mitigation controllables (example), 141
Jammer suppression (application), 145, 146
Jammer-to-signal ratio (application), 40, 41
Jamming (application), 144, 152, 225, 228
Jamming detection (application), 245, 354
Jamming effectiveness (application), 43, 107, 110, 210
Jamming networked radars (application), 152
Jamming, distributed (application), 228
Java, 280, 281
JDL, 111
Jitter (application), 47, 48
Joint comms and radar (application), 24
Joint ES, EP, and EA (application), 24
Joint optimization (application), 44
Joint optimization of transmitter and receiver (application), 360
JSON, 257

K

Kafka, 277
Kaggle, 346
Kalman filters, *see* ML technique
Klemm, Richard quote, 186
kNN, *see* ML technique
Knowledge distillation, 323

Knowledge management, 7, 8, 14
Knowledge transfer, 56
Knowledge-based ML, *see* Hybrid approaches
Kriging metamodeling, 305
Kullback-Leibler divergence, 226

L

Label data (application), 71
Latency (application), 18, 40, 41, 43, 44, 47, 48, 75, 113, 141
Latin hypercubes, 304
Learn effects (application), 15
Learn how to collaborate (application), 197
Learn how to schedule (application), 197
Learn novel emitters (application), 80
Learn reward functions (application), 72
Learning human actions (application), 197
Learning, in-mission, *see* RL concept
Lee, Kai-Fu quote, 84
Legal responsibility, 244
Lehrer, Tom quote, 267
LEXrec, 120
Lidar, 10, 112, 347, 354
Lifecycle
 Data, 246, 293
 Hardware, 281–283
 System, 293, 337–339
 Test, 291
Limitation studies, 323
Link stability (application), 49
Live testing, 299
LLMs, *see* ML technique
Load balancing (application), 144
Locality (application), 43

Localization (application), 98, 99
 Logic reasoning, 250, 322
 Logistics (application), 179
 Low-shot learning, *see* ML technique
 LPI, 44, 99, 347
 LTE, 347, 358, 359
 Lua, 280
 LǎoZi quote, 28

M

MAB, *see* RL technique
 MACEst, 324
 Machine learning, 7, 8, *see also* ML
 concept; ML technique
 MANET, 23, 27, 84, 152, 193, *see also*
 Cognitive radio networks
 MANET configuration (application), 23
 MANET control (application), 27
 MANET routing (application), 152
 MANET transmission scheduling (ap-
 plication), 152
 Map quality (application), 27
 Marabou, 321
 Markov decision processes, *see* Plan-
 ning
 MATLAB, 63, 300, 345, 347, 348, 352,
 360
 McRae, Thomas quote, 267
 MCTS, *see* Optimization technique
 MDO, 173
 MDPs, *see* Planning
 Measure, *see also* Evaluation
 of Effectiveness, 42, 296, 298, 340
 of Performance, 42, 296, 298, 340
 of Suitability, 296, 298, 340
 Medical causality (application), 117
 Medicine (application), 179
 Message error rate (application), 40

Metacontrol, 155
 Metadata, 102, 111, 115, 241, 242, 245,
 247, 249–252, 254, 255, 265, 340,
 346
 Air traffic safety example, 252, 253
 NEU RF fingerprinting example, 252
 Metaheuristics, 142, 155
 Metalearning, 64, 66, 70–73, 103, 117,
 139, 147, 155, 225, 288, 293, 324
 Benefits, 71
 Hyperparameter optimization, 72
 Search prioritization, 72
 Metamaterials, 83, 84, 148
 Metasurfaces, 148
 Metasurfaces (application), 148
 Metrics, 31, 32, 40, 41, 43, 44, 47, 49,
 150, 325, *see also* Evaluation
 Computed, 41
 Cost, 40
 Cost of changing plan, 216
 Effectiveness, 40
 Inferred, 43, 210
 Latency, 43
 Learned, 41
 Locality, 43
 Measurability, 43, 210
 Properties, 42, 211
 vs Observables, 44
 Metrics for RF environments and strate-
 gies (example), 107, 108
 MicroPython, 278, 280
 Microwave, 10
 MIMO, 146, 359
 MIMO sensor array design (applica-
 tion), 146
 Missing features, 56
 Mission planning (application), 174, 195
 Mission success (application), 40, 41

- Mission-critical communications (example), 358
- Mistake-bound model, 323
- MIT, 346
- ML concept, 55, 56, *see also* RL concept
 - Adaptive learning, 70, 117, 304
 - Adversarial training, 80, 259, 261, 263, 303
 - Anomaly detection, 1, 61, 63, 69, 70, 73, 74, 115–117, 153, 347, 353, 360
 - Causal inference, 117, 184
 - Causal reasoning, 1
 - Characterization, 1, 101
 - Classification, 1, 55, 56, 58, 59, 63, 70, 80, 99–101, 105, 114, 116, 212, 213, 222, 248, 260, 262, 265, 305, 307–309, 320, 328, 345, 347, 353, 360
 - Clustering, 34–36, 56, 58, 63, 64, 99, 100, 106, 116, 145, 219, 310, 318, 319, 345, 353, 359, 360
 - Concept drift, 79, 117, 209, 247
 - Data augmentation, *see* Data
 - Data fusion, *see* Data
 - Dendrogram, 34, 37
 - Dimensionality reduction, 60, 61, 63, 64, 71, 103, 111, 114, 362
 - Domain of expertise, 15, 65, 69, 71, 209, 291, 297, 316, 319
 - Embedded, 75
 - Foundational models, 83
 - Generalization, 27, 58, 64, 65, 67, 69–71, 73, 77, 78, 109, 137, 154, 219, 258–260, 262, 266, 292, 297, 298, 302, 316, 319, 321, 341, *see also* Evaluation
 - ML concept, cont.
 - Generative AI, 62, 73, 74, 83
 - Hybrid ML, *see* Hybrid approaches
 - Hyperparameters, 65, 71–73, 292, 293, 324, 335, 336
 - Instance-based methods, 57
 - Intent recognition, 1, 119, 121, 344
 - Interpretability, 57, 67, 74
 - Knowledge distillation, 266
 - Knowledge-based ML, *see* Hybrid approaches
 - Latent embedding, 69
 - Learning from encrypted data, 266
 - Model calibration, 65
 - Model quality, 310
 - Model-based methods, 58
 - Neurosymbolic ML, *see* Hybrid approaches
 - Open-set classification, 70, 80, 102, 106, 117, 328, 352
 - Outlier detection, 56, 58, 64
 - Overfitting, 56, 61, 64, 65, 261, 302
 - Regression, 27, 56–58, 63, 80, 107, 114, 212, 222, 262, 305, 342, 345, 360
 - Sample efficiency, 57
 - Scientific ML, *see* Hybrid approaches
 - Self-supervised, 56, 353, *see also* RL concept
 - Semi-supervised, 56
 - Supervised, 55, 216, 359
 - Time-series analysis, 63, 64, 100
 - Train, test, evaluate loop, 77, 105, 291

ML concept, cont.

- Training approach, 60, 61, 63, 65, 70, 76–78, 80, 82, 104, 105, 153, 209, 210, 212, 226–228, 251–253, 260–264, 266, 290–295, 299, 309, 314, 315, 317–319, 322, 323, 335, 346, 356, 359–361, *see also* Data: A priori training data
- Transparency, 57
- Underfitting, 64, 65
- Unsupervised, 34, 55, 114, 318, 359

ML evaluation (example), 351, 352, 356, 357, 360

ML in Python (example), 345

ML in R (example), 345

ML technique

- Adaptive binning, 101
- Analogical networks, 69
- Analogical reasoning, 64
- ANN challenges, 228
- ANNs, 27, 55, 58, 59, 63, 69, 71, 107, 116, 211, 225, 228, 253, 317, 322, 359–361
- Apprenticeship learning, 197
- Attention, 67, 74, 81
- Autoencoders, 60, 61, 69, 70, 100, 116, 360, 362
- AutoRegression, 63
- Bagging, 63, 317
- Bayesian methods, 63, 64, 101, 107, 116, 196, 321
- Belief propagation, 359
- Boosting, 63
- Burst detection, 101
- Case-based reasoning, 64, 146
- Causal models, 81, 209, 352
- Change detection, 212
- Clustering, 314

ML technique, cont.

- CNNs, 60, 70, 80, 82, 99, 104, 106, 309
- Combinatory categorical grammars, 120
- Compression, 75
- Curve fitting, 55
- Data association, 114
- DBScan, 63, 64, 100
- Decision fusion, 114
- Decision trees, 63–66, 75, 81, 82, 193, 212, 213, 224, 317, 353, 356, 360
- Decomposition, 75, 76
- DeepNet breakthroughs, 59
- DeepNets, 57, 59, 65–69, 71, 73, 75, 76, 79–81, 83, 97, 100, 102–104, 106, 116, 145, 193, 212, 222, 224, 226, 228, 262, 282, 321, 322, 354, 359, 361
- Diffusion models, 74
- Distance-based classification, 69
- Diversified minibatch stochastic gradient descent, 260
- Embedded GANs, 76
- Ensemble methods, 58, 62, 65, 75, 104, 212–214, 266
- Extra trees, 212, 213, 360
- Federated learning, 266
- Few-shot learning, 64, 71
- Finite input response, 106
- Fourier transforms, 39, 57, 101, 103, 342
- Function approximation, 55
- Fuzzy expert system, 212
- GANs, 61, 62, 73, 74, 76, 104, 263, 294, 303, 360

ML technique, cont.

- Gaussian processes, 63, 64, 101, 145
- Generative models, 69, 70, 73
- GMMs, 63, 100, 106
- Granger causality, 118, 352
- Graph neural networks, 100, 118
- Greibach normal form grammars, 120
- Hashing, 57
- Hawkes processes, 100, 118
- Hierarchical clustering, 38, 63
- Hilbert-Huang transform, 101
- HMMs, 63, 64, 118
- Hysteresis, 118
- Imitation learning, 197
- Independent component analysis, 99
- Information-based localization, 114
- Instance-based learning, 278
- Interpulse interval analysis, 101
- Inverse RL, 197
- Isolation forest, 63, 354
- Junction tree algorithms, 115
- Kalman filters, 63, 221, 342
- Kernel smoothing, 101
- Kleinberg's algorithm, 101
- kMeans, 38, 63
- kNN, 57, 63, 64, 69, 70, 80, 82, 116, 212, 213, 224, 317
- Knowledge distillation, 75, 76
- Kohonen networks, 61, 362
- Lightweight model design, 75, 76
- Linear regression, 57, 116, 342
- LLMs, 62, 74, 75, 324, 345
- Logistic regression, 63, 81, 82
- Low-shot learning, 69, 70, 228
- LSTM, 100
- Majority voting, 63
- Markov random fields, 106

ML technique, cont.

- Memory-augmented ANNs, 69
- Metric learning, 69, 70
- Mixture-of-experts, 75, 76
- Multiattribute utility theory, 117
- Multistage training, 309
- Multitask learning, 70
- Multitask-prompted learning, 70
- n-gram analysis, 118
- Naïve Bayes, 63, 81, 82, 104, 105, 212, 213, *see also* ML technique: Bayesian methods
- Neural ODEs, 100
- Neural point processes, 100
- Open-set classification, 69, 319
- PCA, 63, 64, 99, 342
- Poisson processes, 100
- Precursor detection, 118
- Probabilistic graphical models, 73
- Probabilistic structural causal models, 117
- Pruning, 75
- Pseudoclass methods, 70, 260
- Quantization, 75
- Random forest, 63, 82, 116, 360
- Renewal processes, 100
- Resampling, 101
- Restricted Boltzmann machines, 116
- RNNs, 60, 100, 104, 116, 359
- Rules, 116, 212
- Self-organizing maps, 61, 100, 359
- Self-supervised learning, 70, 216
- Siamese neural nets, 61, 104
- Simulated annealing, 146
- SLIDE, 76
- SNNs, 61, 283
- State estimation, 114

- ML technique, *cont.*
 - Structural analogy, 70
 - Structural causal models, 106, 118
 - SVMs, 27, 38, 58, 59, 63–65, 69, 70, 75, 79, 81, 82, 100, 104, 107, 109, 116, 211, 218, 219, 221–224, 278, 280, 317, 342, 356, 360
 - Symbolic deduction, 67
 - Table lookup, 57, 64
 - Teacher-student models, 76, 266
 - Temporal CNNs, 60
 - Temporal LSTMs, 100
 - Temporal point processes, 100
 - Temporal RNNs, 100
 - Thresholding, 69
 - Time-frequency analysis, 99
 - Transfer learning, 8, 56, 64, 70, 72, 229
 - Transformers, 62, 73, 74, 99, 100
 - Trend lines, 55
 - Uncertainty-aware interpolation, 101
 - VAEs, 73, 74, 116
 - Wavelet transforms, 99, 101
 - Wavelets, 106
 - Zero-shot learning, 69, 70
- MLOps, 337
- mmWave, 9, 112, 347
- Mobile Broadband (example), 358
- MobileGPT, 76
- Model selection (application), 71
- Model the spectrum (application), 55
- Modeling and simulation, 299, 348
- Models, 33, 245, 303
 - of adversary, 182, 301–303
- Modulation classification (application), 55
- Modulation clustering (application), 56
- Modulation recognition (application), 81, 104, 254, 262, 265
- Monitor regulatory compliance (application), 68
- Monitoring, *see* Execution monitoring
- Monte Carlo tree search, *see* Optimization technique
- Multiarmed bandit, *see* RL technique
- Multidomain operations, 173
- Multifunction radar grammars (application), 120
- Multifunction radar scheduling (application), 152
- Multifunction task scheduling (application), 152
- Multiobjective optimization, 139, 187, 190, 289
- Multiobjective optimization (application), 2
- Multiobjective optimization (example), 355
- Multitarget tracking (application), 146, 152
- Myth
 - AI is ML, 55
 - AI solves all problems, 342
 - Bandwidth is infinite, 24
 - CogEW is a big effort, 2, 335, 363
 - Cognition can't be evaluated, 1
 - Cognition is undefined, 3
 - Communications is safe and plentiful, 156, 185
 - Coordination requires communication, 158
 - EW is a big-data environment, 283
 - EW is hardware-only, 25
 - EW is military-only, 15
 - Latency is zero, 24

Myth, cont.

Must have large dataset, 27, 79, 259, 267

Network is homogeneous, 24

Network is reliable, 24

Network is secure, 24

of Distributed computing, 24

Python belongs on an embedded system, 278

Topology doesn't change, 24

Transport cost is zero, 24

N

n-choose-*k* tests, 316–318, 323

National Research Council quote, 319

NATO, 83, 84, 143, 255, 350

Naval command and control (application), 179

Naval engagement (application), 176

Navigation surveillance (application), 347

Navy mission planning (example), 140

Network connectivity (application), 352

Network connectivity (example), 352, 357

Network coordination (application), 22

Network embedded systems (application), 152

Network intrusion detection (application), 116

Network management (application), 189

Network operation (application), 114

Network paths (application), 146

Network performance (application), 26

Network reliability (application), 146

Network robustness (application), 115

Network task scheduling (application), 176

Network throughput (application), 40

Network topology estimation (application), 118

NEU, 252, 253, 256

Neuromorphic computing, 283

Neurosymbolic ML, *see* Hybrid approaches

NEWEG testbed, 349

NIST, 289, 337, 338, 347

NIST datasets, 346

No free lunch, 159

Node activation (application), 114

Node deployment (application), 114

Node prioritization (application), 114

Noise, 11, 33, 56, 64, 73, 77, 115, 119, 248, 259, 261–263, 266, 294, 323, 349

Noise elimination (application), 60

Noise reduction (application), 64

Non-stationarity, 229

Non-terrestrial networks, 17

Non-terrestrial networks (application), 17

Noncooperative games, 189

Nonstationary policy, 182

Normalized RMSE, *see* Evaluation

Novelty, 1, 2, 15, 21, 24, 26, 27, 55, 69, 70, 77, 78, 80, 83, 97, 102, 107, 207, 216, 217, 219, 223, 224, 248, 253, 259, 261, 265, 291, 297, 298, 314–316, 318, 319, 323, 328, 341, 342, 344, 352, 353, 355, 356, 362, *see also* Surprise

nRMSE, *see* Evaluation

NS3, 348

NSF, 349

Nulls placement (application), 146

O

- Object recognition (application), 352
- Objective function, 31, 44, 135, 151, 153, 177, 194, *see also* Utility
 - as Reward function, 221
- Controllables, 31, 38, *see also* Controllables
- Creation of, 44
- Example 5G, 113
- Example for EW, 41
- Formal definition, 31, 45
- in Optimization, 138, 140
- Learning from human, 196
- Maintenance, 221
- Metrics, *see* Metrics
- Multinode, 49
- Objectives, 31, *see* Metrics
- Observables, 32, *see also* Observables
- Properties, 31, 44
- Requirements, 44
- Structure, 48
- Unobservable context, 33
- Utility, 45
- Weights, 41
- Observables, 31–34, 43–45, 49, 102, 107–109, 325, 327, *see also* Features; Unobservables
 - Examples, 34, 50, 51
 - for comms link stability, 50
 - Normalized values, 33
 - vs Metrics, 44
- Observables, controllables, and metrics (example), 108
- Observation scheduling (application), 175
- Occam learning, 323
- OMNeT, 348
- On-line shopping (application), 68
- Online learning, *see* RL concept
- Ontology, 186, 193, 249, 250, 322, 340
- Ontology-based knowledge graphs, 322
- OODA, 5, 13
- OOPDAL, 5
- Open-loop testing, 299, 348, 349
- Open-set classification, *see* ML technique
- Open-set recognition (example), 352
- OpenAI, 82, 83, 345
- OpenRAN Gym testbed, 349
- Operational insights (application), 76
- Optimality, 23, 31, 142
- Optimization, 138, *see also* Planning; Scheduling
 - Optimization concept
 - and Game theory, 140
 - Anytime algorithms, 154, 159
 - Centralized, 156, 157
 - Decentralized, 156–158
 - Distributed, 155–157, 184
 - Heuristics, 138
 - Human-guided, 140
 - Limits of centralized, 158
 - Multiobjective, 139
 - Objective function, 140
 - Randomized search, 225
 - Requirements, 147
 - Reward hacking, 152
 - Satisficing, 155
 - Scheduling, 147
 - Optimization technique
 - ACO, 143, 158, 178, 355, 359
 - Bayesian, 145, 304
 - Constraints, 140, 221
 - Cross-entropy methods, 146
 - Evolutionary computation, 152

Optimization technique, *cont.*

GAs, 64, 71, 144–146, 178, 225, 305, 359

Gradient descent, 146

Heuristic scheduling, 152

Iterative approximation, 57, 64, 145, 178, 226, 304, 342, 343

Knowledge-based engineering, 152

MCTS, 120, 142, 181, 221, 222, 226, 304

Policy-gradient, 222

PSO, 144, 145

Randomized search, 64, 138, 143–146, 222, 304, 305, 355

Risk-based scheduling, 152

Simulated annealing, 145

Tabu search, 144

Task scheduling, 152

Temporal reasoning, 152

ORBIT testbed, 349

Outlier detection, *see* ML concept: Anomaly detection

Overfitting, *see* ML concept

Overkill, 308

OWL, 250

Own-system stealth (application), 51

P

PAC learning, 321, 323

PAC-Bayesian inequalities, 321

Packet error rate (application), 50

Pareto-optimal frontier, 139, 140

Particle swarm optimization, *see* Optimization technique

Pascal, 280

Path loss prediction (application), 112

Pattern classification (application), 76

Patterns across missions (application), 265

PCA, *see* ML technique

Pd vs Pfa balance (application), 24, 139

PDDL, 177, 180

Perception-action cycle, 5, 229

Perceptual attention, 229

Performance estimation, 26, 27, 107

Performance estimation (application), 55, 107, 109, 118, 217, 218, 228

Performance prediction (application), 155

Phased array radar dataset, 347

Phased array task scheduling (application), 152

PHATT intent recognition, 120

Photonic computing, 284

Plan generation (application), 55

Plan recognition (application), 119

Planning, 14, 25, 138, 155, 156, 173–178, 180–183, 186, 188, 195, 322, 344, *see also* Search; Optimization; Scheduling; Replanning

Action uncertainty, 180

Active communication, 183–185, 215, 219

Active sensing, 25, 183–185, 216, 219

Analogical, 216

and game theory, 188

Anytime, 120

Belief space planning, 181

Classical, 175, 177

Communication, 183

Communications resources, 186

Conditional, 175, 180, 181, 215

Conflicts, 186, 195

Contingent, 175, 176, 216

Evaluation criteria, 177

Expected outcomes, 215

- Planning, cont.
 First-order decision diagrams, 181
 Graph, 216
 HTN, 178–180, 193, 195, 216
 Hubble Space Telescope, 175
 In-mission, 19, *see* Replanning
 Information uncertainty, 183
 Inputs and outputs, 173
 Interleaving with execution, 216
 MDPs, 107, 180, 181, 183, 187, 215, 220, 225, 227–229
 Mixed-initiative, 194–196
 PDDL, 177
 Plan stability, 215, 216
 Policy, 227
 POMDP, 181, 183
 Probabilistic, 175, 216
 Resource, 175, 186–188
 Satisfiability solvers, 180, 181, 322
 Task allocation, 216
 Temporal, 152, 175, 180, 186
 Unexpected outcomes, 215
Planning (application), 177
Planning for SEAD (example), 156
Platform survival (application), 40
Platform wear-and-tear (application), 40
Playbook interfaces, 194, 195
PlutoSDR testbed, 347
PNT, 2, 349
Police jamming (application), 15
POMDP, *see* Planning
Position accuracy (application), 113
POWDER testbed, 349
Power (application), 40, 41, 141
Power allocation (application), 146
Power consumption (application), 18, 50, 113
Power control (application), 152, 189, 359
Power management (application), 145
Precision, 308
Predict EMS (application), 3
Predicting future activities (application), 119
Preference (application), 40
Preference learning (application), 196
Probability of detection (application), 24, 40, 41, 44, 48, 139, 210
Probability of event detection (application), 40
Probability of false alarms (application), 24, 40, 48, 139, 210
Probability of interception (application), 41, 44, 47, 210, 347
Probability of mission kill (application), 40
Process failure detection (application), 81
Production (application), 179
Program synthesis (application), 322
Protégé, 250
Protobuf, 257
PSO, *see* Optimization technique
Pulse compression (application), 144
Pulse train separation, 99
Pulse-repetition intervals (application), 70
PyLayers, 348
PyMite, 278
Python, 63, 221, 254, 255, 278, 280, 324, 345, 347, 348, 352

Q

Quality, 310
 Quality of service (application), 41, 48, 146
 QualNet, 348
 Quantum communication, 84, 284
 Quantum computing, 83, 284
 Quantum key distribution (application), 284
 Quantum radars, 284
 Queensland University of Technology, 347
 Queue scheduling (application), 152
 Quote
 Anderson, Martin, 82
 Belval, Edouard, 280
 Box, George, 34
 Center for Strategic and Budgetary Assessments, 300
 Chairman of the Joint Chiefs of Staff, 9
 Clarke, Arthur C., 148
 Gürbüz, Sevgi, 252
 Grayson, Tim, 363
 Hanlon, Robert, 249
 Klemm, Richard, 186
 Lee, Kai-Fu, 84
 Lehrer, Tom, 267
 LǎoZi, 28
 McRae, Thomas, 267
 National Research Council, 319
 Rao, Naveen, 281
 Rehkopf, Max, 338
 Sutton, Richard, 221
 von Moltke, Helmuth, 214
 Zilberstein, Shlomo, 155
 Qwen2.5-Max, 83

R

R, 345
 Radar
 EA, 12, 13
 EP, 12
 ES, 12
 vs Comms, 12, 44
 Radar and comms co-design (application), 139
 Radar and map data integration, 112
 Radar class detector (example), 77, 78, 104
 Radar cross section minimization (application), 145
 Radar cross-section estimation (application), 146
 Radar design (application), 145
 Radar detection (application), 19
 Radar EA (example), 51
 Radar imaging (application), 146
 Radar mode (application), 210
 Radar mode evaluation (example), 310
 Radar object tracking (application), 155
 Radar placement (application), 145
 Radar scan type (application), 104
 Radar scheduling (application), 145
 Radar signal identification (application), 70, 104
 Radar situation awareness (application), 51
 Radar surveillance (application), 116
 Radar target assignment (application), 155
 Radar task scheduling (application), 152
 Radar time-to-engage (application), 51
 Radar tracking (application), 19
 Radio Access Technology, 114

- Radio Knowledge Representation Language, 194
- Range ambiguity (application), 51
- Range resolution (application), 40
- Rao, Naveen quote, 281
- Raspberry Pi testbed, 349
- Real-time in-mission ML, *see* RL concept
- Recall, 308
- Receive scheduling (application), 147
- Reconfigurable intelligent surfaces (application), 147
- Red team, 294, 303
- Reducing adversary operator effectiveness (application), 43
- Refine heuristics (application), 72
- Regression, *see* ML concept
- Rehkopf, Max quote, 338
- Relationship mapping (application), 81
- Remote sensing (application), 112
- Replanning, 25, 174, 207, 209, 214, 215, *see also* RL
 - Plan repair, 215
- Reprogramming, 11
- Requirements
 - Data, 242, *see* Data
 - for system evaluation, 297
 - Hardware, 281
 - Learning from human, 196
 - System, 19
 - Test scenario, 327, 339
 - User, 24
- Resilience to countermeasures (application), 284
- Resistance to eavesdropping (application), 284
- Resolute, 321
- Resource allocation (application), 19, 74, 145, 188–190, 358
- Resource efficiency (application), 51
- Resource management, 25, 174, 186–188
- Resource management (application), 19, 139, 144, 152, 159, 175, 186, 228, 357, 359
- Resource optimization (application), 186, 357
- Resource planning (application), 173
- Resource use (application), 18, 24, 41, 107
- Response planning (application), 194
- Response time (application), 41
- Reward, 46, 153, 182, 183, 187, 197, 219, 221, 224–227, 229, 309, 310, 355, *see also* Utility
- Reward hacking, 152, 154, 302
- Rewards, 76
- RF data factory, 346
- RF environments (example), 107, 108
- RF fingerprinting (application), 101, 106, 228, 252, 253, 262–264, 346, 351, 357
- RF fingerprinting (example), 106, 263, 264, 351
- RF fingerprinting dataset, 346
- RF fingerprinting metadata (example), 252, 253
- RF quality (application), 81
- RF sensing (application), 283
- RF signal dataset, 347
- RFML, 348
- RFnest testbed, 349
- RFNoC, 254
- RFView, 348

- RiftNet, 106
 - Risk, *see* Evaluation
 - Risk (application), 40, 81
 - Risk management, 196, 337, 338
 - Risk-aware planning (application), 180
 - Riverbed, 348
 - RKRL, 194
 - RL concept, 56, 216, 247
 - Action selection, 218, 219
 - Active learning, 16, 147, 219, 220, 247, 258, 261, 304, 305, 318
 - and adaptive control, 216
 - and model-predictive control, 216
 - Challenges, 228
 - Explore-exploit, 145, 147, 153, 181, 186, 219, 221–223, 225–229, 304, 305, 312, 315, 356
 - for novel environments, 26, 207
 - In-mission learning, 26, 27, 55, 56, 208, 209, 216, 217, 312, 314
 - Incremental learning, 26, 70, 109, 217, 223, 314, 315, 353, 356, 360, 361
 - is not MDP, 183
 - Learning opportunities, 218
 - Retraining, 69, 80, 209, 218, 219, 222, 223, 312, 314, 323, 356, 357
 - When to update models, 218
 - RL technique, *see also* ML technique; Optimization technique; Planning
 - Actor-critic, 221, 222, 227
 - Advantage actor-critic, 227
 - Anytime MAB, 225
 - Asynchronous advantage actor-critic, 227
 - Bayesian, 221, 222, 225, 229
 - Cognitive architectures, 222, 229
 - Deep deterministic policy gradient, 226
 - RL technique, *cont.*
 - Deep Q-learning, 221, 222
 - Deep Q-Networks, 228
 - Deep RL, 225, 227
 - Deterministic policy gradients, 225
 - Expected model change, 219
 - Group relative policy optimization, 226
 - Linear function approximators, 227
 - MAB, 183, 188, 221, 222, 224, 225, 304
 - MDPs, 183
 - Other ML models, 183
 - Policy-gradient, 225
 - Proximal policy optimization, 226
 - Q-learning, 183, 221, 222, 227
 - SARSA, 227
 - Tabular actor-critic, 227
 - Trust region policy optimization, 226
 - RMSE, *see* Evaluation
 - RNNs, *see* ML technique
 - Root-cause analysis (application), 359
 - Root-mean-squared error, *see* Evaluation
 - Route planning (application), 178
 - Routing trees (application), 144
 - Russia, 300
 - Rust, 278, 280, 281
- ## S
- Safety, 17, 142, 196, 228, 252, 287, 297, 320, 321, 336, 337, *see also* Evaluation
 - SAR, 98, 99, 347
 - SAR datasets, 346
 - SAR image generation (application), 99
 - Satellite broadcast scheduling (application), 152

- Satellite self-defense (application), 39
- Satisfiability modulo theories, 321
- Satisficing, 155
- Scappy, 348
- Scenario driver, 324, 325, 327, 328
- Scenario generation (application), 72, 74
- Scene segmentation (application), 69
- Scheduler verification (application), 322
- Scheduling, 138, 147, 155, 175, 180, 322, 344, *see also* Planning; Optimization
 - Critical path methods, 147
 - Cyclical schedules, 150
 - Emergency mode, 151
 - Flexibility, 151
 - Hubble Space Telescope, 175
 - Linear schedules, 150
 - Mission environment, 151
 - Resource, 180
- Scheduling (application), 152
- Scheduling for 5G (application), 152
- Scheduling for CRNs (application), 152
- Scheduling in sensor networks (application), 152
- Schema, 249, 250, 255–257
- scikit-learn, 35, 36, 212, 213, 324, 345, 360
- scikit-learn classifiers (example), 213
- scikit-learn clustering (example), 35
- SDR, 250, 273
- SDR radio test system (example), 254
- SDR testbed, 254, 336, 347, 349, 350, 353
- SEAD planning (application), 179
- Search, 141, 313, *see also* Planning
 - Complexity, 147
 - Gradient-based methods, 142
 - Metaheuristic methods, 142, 178
 - Randomized methods, 142, 147, 178
 - Stochastic algorithms, 142
- Security, 20, 282, 287, 297, 336, 337, 341, *see also* Evaluation
- SEI, 81, 118
- SEI (application), 101, 106, 245, 254, 299, 303, 309, 351, 357
- Selecting tasks (application), 228
- Self-configuration (application), 18
- Self-organizing maps, *see* ML technique
- Self-organizing networks, 359
- Self-organizing networks (application), 189, 357, 359
- Self-regulating network (application), 18
- Self-supervised learning, *see* ML technique
- Sensing, 19
- Sensitivity (application), 40
- Sensitivity analysis, 324
- Sensor fusion, *see* Data fusion
- Sensor network clustering (application), 145
- Sensor networks (application), 112
- Sensor scheduling (application), 147, 152
- Sentiment analysis (application), 81
- SEWES, 348
- Shared spectrum (application), 145
- SigMF dataset, 256
- SigMF metadata (example), 255, 256

- Signal characterization (application), 81, 104
- Signal classification (application), 12, 104, 106
- Signal classification challenge, 104
- Signal detection (application), 75, 155, 347
- Signal generation (application), 74
- Signal interception (application), 149
- Signal patterns (application), 74
- Signal processing, 59, 102
- Signal processing (application), 116, 283
- Signal recognition (application), 81
- Signal separation, 99
- Signal separation (application), 346
- Signal separation dataset, 346
- Signal spoofing (application), 265
- Signal tracking (application), 100
- Signal-to-clutter (application), 49
- Signal-to-interference plus noise (application), 49
- Simics, 349
- Simon, Herbert A., 142, 155
- Simplification, 323
- Simulate adversarial RF (application), 76
- Situation assessment, 6–8, 14, 19, 25, 207, 244, 273, 274
 - vs Situation awareness, 5, 7
- Situation awareness, 5, 97, 98, 110, 111, 191, 250
- Situation awareness (application), 117
- Skydel, 348
- SLT, 320
- Small drone classification dataset, 347
- Social relationships (application), 117
- Software, 242
- Software architecture, 20
- Software Engineering Institute, 337
- Software languages, 63, 221, 254, 255, 278, 280, 281, 324, 345, 347, 348
- Solution stability (application), 40, 215
- Source separation (application), 118
- Space surveillance radar (application), 225
- Spam filtering (application), 81
- Specific emitter identification (example), 309
- Spectral efficiency (application), 50
- Spectrograms, 104
- Spectrum agility (application), 41
- Spectrum deployment (application), 359
- Spectrum learning (application), 57
- Spectrum management (application), 18, 74, 195
- Spectrum monitoring (application), 70, 75
- Spectrum scanning (application), 76
- Spectrum sensing (application), 152
- Spectrum sharing (application), 19, 139, 152, 190, 194, 228
- Spectrum understanding (application), 78
- Speech recognition (application), 64
- Speed (application), 18
- Speed accuracy (application), 113
- Spoofing (example), 354
- Spoofing detection (application), 74, 347, 354
- Stable Diffusion, 82
- STAMP, 337
- Static analysis (application), 322
- Stationary policy, 182

- Statistical approaches, 40, 56, 57, 102, 114–118, 142, 145, 159, 226, 304, 305, 345, 352, 354
- Statistical learning theory, 320
- STPA, 337
- Strategy, 11, 22, 31, 38, 46, 50, 107, 108, 135–138, 141, 142, 207, 219–221, 223, 260, 265, 297, 311, 312, 314, 315, 325, 327, 356, *see also* Controllables
- Search, 142, 143, 178
- Strategy search (application), 72
- Strategy selection (application), 186
- Surprise, 1, 40, 64, 71, 78, 214, 223, 288, 292, 294, *see also* Novelty
- Sutton, Richard quote, 221
- SVMs, *see* ML technique
- SynDICAte, 348
- Synthesizing deceptive signals (application), 76
- Synthesizing new modulations (application), 265
- Synthetic data (application), 61, 71, 73, 83
- Synthetic signals (application), 74
- System availability (application), 40
- System configuration (application), 2
- System scalability (application), 113
- Systems engineering, 337, 339
- SystemVue, 349
- T**
- Tülu 3, 83
- Target assignment (application), 155, 195, 355
- Target classification (application), 116
- Target concealment (application), 51
- Target discrimination (application), 19
- Target identification (application), 49, 145
- Target locations (example), 355
- Target recognition (application), 57, 265
- Target surveillance (application), 144
- Target threat assessment (application), 152
- Target tracking (application), 43, 81, 112, 115, 190
- Task allocation (application), 176
- Task assignment (application), 186
- Task assignment in heterogeneous teams (example), 186
- Task coordination (application), 186
- Task management (application), 152
- Task scheduling (application), 180, 228
- Telescope schedules (application), 175
- TEMPLAR planning (example), 174
- Temporal patterns (application), 104
- TensorFlow, 345, 360
- Terrorist activity (application), 112, 118
- Test coverage (application), 72
- Test framework, *see* Scenario driver
- Test scenario, 327, 339
- Test scenarios (application), 74
- Testbed, 254, 336, 347, 349, 350
- Threat analysis (application), 155
- Threat evaluation (application), 194, 196
- Threat geolocation (application), 195
- Threat identification (application), 99, 195
- Threat prioritization (application), 251
- Throughput (application), 43, 44, 50, 103, 141, 193, 210
- Time (application), 40, 41
- Time to first fix (application), 113
- Time-domain coding (application), 146
- Timeliness (application), 18

- Timing synchronization (application), 159
 - TinyML, 75, 76
 - Tomography (application), 347
 - Toolkit, 35, 36, 63, 83, 212, 213, 250, 280, 300, 324, 341, 345, 347–350, 352, 360
 - TorchSig, 348
 - Track quality (application), 43
 - Tracking accuracy (application), 51, 107
 - Trade-offs
 - AI algorithms, 27, 56–58, 60–62, 67, 68, 74–76, 78–81, 83, 102, 107, 117, 120, 177, 183, 186, 190, 193, 215, 218, 221, 227, 281, 282, 297, 341–343
 - Hardware, 26, 27, 38, 75, 76, 79, 186, 257, 258, 265, 281, 282, 341
 - Mission, 24
 - Verification approaches, 322
 - Traffic monitoring (application), 113
 - Train, test, evaluate for ML, 77, 78, 104, 105
 - Training data, *see* Data
 - Training data management (application), 71
 - Training strategy management (application), 71
 - Trajectory optimization (application), 228
 - Transfer learning, *see* ML technique
 - Transferability, 297, *see also* Evaluation
 - Transition probabilities (application), 216
 - Transmission parameters (application), 144, 145
 - Transmission performance (application), 145
 - Transmit energy (application), 152
 - Transmit scheduling (application), 147
 - Trust, 287, *see* Evaluation
- ## U
- UAV performance (application), 113
 - Uncertainty, 4, 19, 25, 40, 57, 69, 99, 101, 110, 118, 145, 159, 174, 180, 183–186, 189, 190, 219, 221, 225, 229, 249, 254, 261, 297, 303, 304, 324, 343, *see also* Evaluation
 - Uncertainty (application), 40
 - Uncertainty generation (application), 190
 - Underfitting, *see* ML concept
 - Understand EMS (application), 3
 - Understand participants (application), 55
 - Understanding human decisions (application), 197
 - University of Amsterdam, 71
 - Unmanned combat robot teams (application), 179
 - Unobservables, 33, 45, 81, 102
 - Unsupervised ML, *see* ML concept
 - Update rate (application), 113
 - Uplink timing (application), 152
 - UQ360, 324
 - USA, 83, 180, 195, 300
 - User, 18, 24, 31, 39, 40, 46, 82, 174, 193–196, 211, 244, 254, 291, 292, 296, 324, 338, 340, 344, 359
 - Access control, 243
 - Behavior modeling, 228
 - Expectations, 153
 - Goals, 44, 119, 196, 296
 - Intent, 7, 119
 - Localization, 113
 - Needs, 196, 337

User, cont.
 Preferences, 153, 196
 Requirements, 20, 22, 24, 289
 Stories, 338
User behavior prediction (application), 228
User intent recognition (application), 119
User interface, *see* HMI
User localization (application), 113
User preferences (application), 196
User scheduling (application), 152
USRP, 254–256, 346, 349, 351
USRP (example), 254, 255
USRP testbed, 349
Utility, 27, 31, 32, 44–47, 49, 73, 137, 138, 141–143, 155, 157, 181, 182, 190, 194, 196, 219, 221, 310, 312–314, 356
 Local vs global, 158
 MetaUtility, 72
 of adversary, 190, 212
 of Information, 184, 219
 vs computation time, 154
Utility accrual (application), 152
Utility function, 31, *see also* Objective function
UV, 10, 290

V

Value of information (application), 40
VC dimension, 320
Vector signal generators testbed, 349
Vehicle localization (application), 354
Velocity ambiguity (application), 51
Verification of AI (example), 321

Versatility to data, 56
Version control, 242, 252
Video generation (application), 73
Visible light, 10, 112, 148, 290
Visualization, 33
 Box-and-whisker plot, 212, 214
 Dendrogram, 34, 35, 318
Visualization of data (application), 112
VITA-49, 256
von Moltke, Helmuth quote, 214
Vulnerability to saturation (application), 141

W

Wave propagation (application), 149
Waveform design (application), 19, 73, 144, 145, 189
Waveform selection (application), 145
Weapons control (application), 190
Weather analysis (application), 347
Weather-control radars (application), 155
Web Ontology Language, 250
Weights, 41, 49
WEKA, 280, 345
White-box attacks, 263
White-box testing, 323
Wideband radar dataset, 347
WiFi and Bluetooth dataset, 347
Wildlife poaching (application), 176

X

XML, 27, 257

Z

Zilberstein, Shlomo quote, 155