Studies in Computational Intelligence 1287

Anis Koubaa · Adel Ammar · Lahouari Ghouti · Wadii Boulila · Bilel Benjdira *Editors*

Generative AI and Large Language Models: Opportunities, Challenges, and Applications

Volume 2

Springer

Studies in Computational Intelligence

Volume 1287

Series Editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland

Editorial Board

Marco Dorigo, Université Libre de Bruxelles, Bruxelles, Belgium

Andries Engelbrecht, University of Stellenbosch, Stellenbosch, South Africa

Vladik Kreinovich, University of Texas at El Paso, El Paso, TX, USA

Francesco Carlo Morabito, Mediterranea University of Reggio Calabria, Reggio Calabria, Italy

Roman Slowinski, Poznan University of Technology, Poznan, Poland

Yingxu Wang, Schulich School of Engineering, Calgary, AB, Canada

Yaochu Jin, Westlake University, Hangzhou, China

The series "Studies in Computational Intelligence" (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

Indexed by SCOPUS, DBLP, WTI AG (Switzerland), zbMATH, SCImago.

All books published in the series are submitted for consideration in Web of Science.

Anis Koubaa · Adel Ammar · Lahouari Ghouti ·
Wadii Boulila · Bilel Benjdira
Editors

Generative AI and Large Language Models: Opportunities, Challenges, and Applications

Volume 2

Springer

Editors
Anis Koubaa
Alfaisal University
Riyadh, Saudi Arabia

Adel Ammar
Prince Sultan University
Riyadh, Saudi Arabia

Lahouari Ghouti
Machine Learning and AI Expert
Prince Sultan University
Riyadh, Saudi Arabia

Wadii Boulila
Deep Learning and AI Expert
Prince Sultan University
Riyadh, Saudi Arabia

Bilel Benjdira
Deep Learning and AI Expert
Prince Sultan University
Riyadh, Saudi Arabia

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-032-24012-5 ISBN 978-3-032-24013-2 (eBook)
https://doi.org/10.1007/978-3-032-24013-2

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.
The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

Preface

The rapid emergence of Generative AI and Large Language Models (LLMs) has marked a definitive inflection point in the trajectory of artificial intelligence. We have moved beyond the era of static models and simple prediction into a landscape where AI systems can create, reason, and act. This profound shift is reshaping the boundaries of what is possible, from generating diverse content forms to executing complex, autonomous workflows. *Generative AI and Large Language Models: Opportunities, Challenges, and Applications—Volume 2* captures this dynamic evolution, offering a comprehensive and rigorously curated exploration of the field's frontier.

Building on the foundations laid in the first volume, a central theme of this second edition is the rise of Agentic AI, witnessing a transition from passive tools to active agents capable of autonomous decision-making. This shift is highlighted by groundbreaking discussions on "Agent Operating Systems (Agent-OS)," which propose a novel substrate for reliable and scalable agent deployment. Security remains a paramount concern as agents are given more agency; chapters on Agentic Cybersecurity Systems dissect the trade-offs in architecture required to secure these autonomous entities.

The volume further explores multimodal generative intelligence, examining the fusion of modalities beyond text and logic. The synergy between LLMs and GANs (Generative Adversarial Networks) bridges language and multi-modal generation, creating new frontiers in healthcare applications. Furthermore, the integration of Spiking Neural Networks with generative avatars and advancements in audio-visual speech enhancement demonstrate the field's push toward more biologically plausible and multimodal intelligence.

Addressing the domain of healthcare, the book highlights the necessity of building systems that are not only intelligent but also culturally and clinically grounded. The inclusion of frameworks for Culturally Adaptive AI underscores the importance of context-aware systems, specifically addressing the nuanced needs of regions like the MENA. The text also traces the evolution from Vision-Language Models (VLMs) to autonomous Agentic Graph Reasoning in dermatology. These chapters present self-correcting diagnostic agents grounded in structured Knowledge Graphs, offering

explainable and trustworthy AI (XAI) to overcome the "black box" limitations of standard LLMs.

Finally, the manuscript examines the deployment of applied generative AI technologies across diverse engineering domains. Real-world applications ground the theoretical discussions, beginning with an exploration of chatbot-assisted software testing that proposes intelligent automation strategies to address the growing complexity of software ecosystems. The focus then shifts to the evolution of Generative AI in meshes and CAD for geometric intelligence. The volume concludes with an analysis of the optimization of Retrieval-Augmented Generation (RAG) systems for high-performance knowledge retrieval.

This book is intended for researchers, practitioners, and students who seek to look beyond the hype and understand the engineering, ethics, and architectures driving the next generation of AI. It consolidates fragmented insights into a unified reference, offering a roadmap through the exhilarating and complex terrain of Generative AI.

We extend our deepest gratitude to the authors for their innovative contributions and to the reviewers for their diligence in ensuring the high quality of this work. We hope this volume serves as both a reference and an inspiration for those building the future of intelligence.

Riyadh, Saudi Arabia
February 2026

Prof. Anis Koubaa
Prof. Adel Ammar
Prof. Wadii Boulila
Dr. Lahouari Ghouti
Dr. Bilel Benjdira

Contents

State-of-The-Art Architecture Trade-Offs for Agentic Cybersecurity Systems 1
Mahmoud Rabie

LLM-GAN: Bridging Language and Multi-modal Generation in the Next AI Healthcare Frontier 43
Waad Bouzidi, Zouhour Neji Ben Salem, and Olfa Meddeb

From Vision-Language Models to Agentic Graph Reasoning: The New Frontier of Generative AI in Dermatology 103
Reema Alduhayan and Anis Koubaa

Agent Operating Systems (Agent-OS): A Blueprint Architecture for Real-Time, Secure, and Scalable AI Agents 137
Anis Koubaa

Generative Avatars and Spiking Neural Networks: A Neuromorphic Approach to Audiovisual Speech Enhancement 173
Aysha Munawwara, Nasir Saleem, Mandar Gogate, Kia Dashtipour, Adeel Hussain, Tughrul Arslan, and Amir Hussain

Generative-AI-Driven Audio-Visual Speech Enhancement Using Deepfake Avatars for Multimodal Hearing Assistance 189
Aysha Munawwara, Nasir Saleem, Mandar Gogate, Kia Dashtipour, Adeel Hussain, Tughrul Arslan, and Amir Hussain

Exploring the Potential of Chatbot-Assisted Software Testing: Towards Intelligent and Automated Testing 207
Moez Krichen

Geometric Intelligence: A Taxonomic Framework for the Evolution of Generative AI in Meshes and CAD 235
Abdulrahman S. Al-Batati, Serry Sibaee, Khaled Gabr, and Yasser Alhabashi

Optimizing Retrieval-Augmented Generation: Analysis of Hyperparameter Impact on Performance and Efficiency 257
Adel Ammar, Anis Koubaa, Omer Nacar, and Wadii Boulila

Culturally Aware LLM-Based Framework for Cardiometabolic Healthcare: The MENA Region as a Case Study 287
Samia Kouki, Nacim Yanes, Heba Khalil, and Nor Shahriza Abdul Karim

State-of-The-Art Architecture Trade-Offs for Agentic Cybersecurity Systems

Mahmoud Rabie

Abstract Agentic cybersecurity systems pair generative AI (LLM-driven) reasoning with guarded tool use under operator oversight. This chapter analyzes architecture trade-offs across: coordination topology (centralized, hybrid, decentralized), agent granularity, policy mode (static versus boundedly adaptive guardrails), execution isolation (containers, serverless, enclaves), memory and retrieval governance (tiered promotion and retrieval diversity), orchestration style, and attestation depth. It presents a reference architecture with integrated identity, provenance, and observability, and an illustrative secure retrieval-augmented generation (RAG) pattern featuring signed promotion manifests and rollback levers. This chapter summarizes preventive, detective, and corrective controls against prompt and retrieval injection, poisoning, tool abuse, privilege escalation, and supply chain tampering. Emphasis is on design rationale and practical guidance to improve resilience, auditability, and controllability, including blast radius containment through provenance and attestation; benchmarking is out of scope. This chapter also outlines open research questions, including invariant synthesis, large-scale/slow poisoning, the economics of provenance depth, stability of adaptive guardrails, measurement and reproducibility provenance, and multi-agent safety (e.g., collusion resilience).

Keywords Agentic AI · Cybersecurity · Architecture trade-offs · Orchestration · Governance · Secure retrieval-augmented generation

1.1 Introduction

Large language model (LLM)–driven agents introduce reasoning, tool-use, and adaptive planning capabilities that materially shift the architecture trade-offs for cybersecurity automation. Rather than issuing static playbooks, frontier agent patterns (reflective reasoning, action/tool chaining, multi-role decomposition) dynamically decompose goals, invoke guarded tools, and refine intermediate outputs through

M. Rabie (✉)
Cyberani by Aramco Digital, Riyadh, Saudi Arabia
e-mail: mahmoud.rabie@ieee.org

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

A. Koubaa et al. (eds.), *Generative AI and Large Language Models: Opportunities, Challenges, and Applications*, Studies in Computational Intelligence 1287,
https://doi.org/10.1007/978-3-032-24013-2_1

critique cycles [2–4]. Their effectiveness in defense workflows depends less on raw model size and more on how core design levers amplify or constrain these loops. The key levers are: (i) memory governance and retrieval diversity, which shape factual grounding and poisoning resistance; (ii) coordination and orchestration topology, which influence latency, failure containment, and policy enforceability; (iii) execution isolation (containers, serverless, enclaves), which bounds blast radius; (iv) attested provenance and identity layers (Secure Production Identity Framework for Everyone (SPIFFE)/SPIRE, Open Policy Agent (OPA)), which anchor trust and rollback; and (v) observability instrumentation (e.g., OpenTelemetry structured traces and semantic spans), which enables post-incident reconstruction and guardrail refinement [24–26]. Aligning these levers with agent reasoning behaviors yields emergent properties—resilience, auditability, controllability—that are not achievable through model prompts alone. This chapter therefore frames agentic cybersecurity architecture as a convergence of adaptive LLM reasoning patterns with secure systems engineering controls and motivates the comparative analysis that follows.

Agentic cybersecurity systems extend traditional Security Orchestration, Automation, and Response (SOAR) paradigms by integrating large language model (LLM) or hybrid reasoning loops that perform adaptive goal decomposition, guarded tool invocation, and iterative self-critique while preserving operator oversight [3–5]. This architectural evolution foregrounds coordination, provenance, and isolation mechanisms rather than monolithic rule scripting, thereby reframing resilience as a function of structural design choices [6, 7]. This chapter analyzes state-of-the-art architecture trade-offs for such agentic cybersecurity systems and their impact on resilience, auditability, and controllability.

Terminology Primer

High-level cues; see the Glossary (Section 1.5.3) for full definitions.

SPIFFE/SVID Workload identity (standard + credential). *See Glossary* [24].

OPA Policy-as-code enforcement engine. *See Glossary* [25].

SLSA/SPDX Supply-chain provenance frameworks/artifacts. *See Glossary* [21, 22].

Attestation Signed evidence about build/runtime state. *See Glossary* [21, 23].

Provenance Verifiable lineage for audit/rollback. *See Glossary* [21, 22].

1.1.1 Background: LLM and Agent Frameworks

Transformer-based large language models (e.g., the GPT series) are decoder-only networks trained for next-token prediction and provide general-purpose reasoning that can be shaped via prompting and tooling [1]. Retrieval-Augmented Gen-

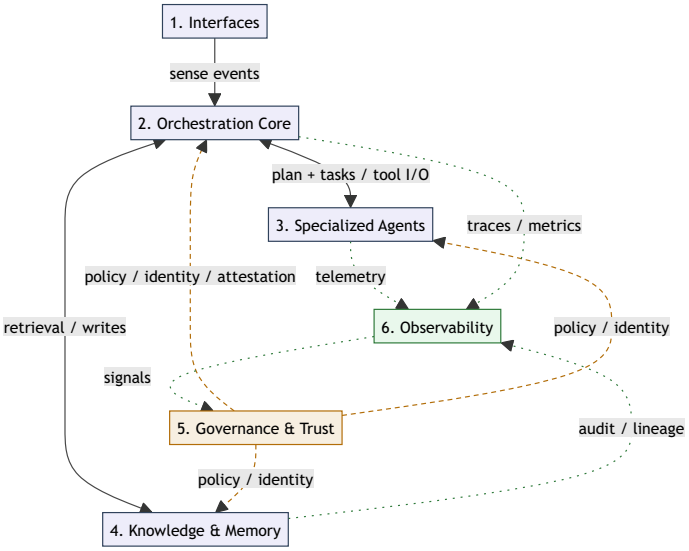


Fig. 1.1 High-level reference architecture (decluttered): *(1) Interfaces* feed *(2) an Orchestration Core* coordinating *(3) Specialized Agents* and *(4) a Knowledge & Memory* store. *(5) Governance & Trust* provides policy, identity, and attestation (dashed arrows), while *(6) Observability* aggregates telemetry/trace feedback (dotted arrows) to tune governance. The figure emphasizes core roles and flows; detailed tiering, promotion manifests, and rollback levers are covered in Fig. 1.8

eration (RAG) composes an LLM with an external knowledge base so retrieved, provenance-carrying evidence grounds generation and improves freshness. Representative lines include dense and sparse retrieval (e.g., DPR) and large-scale retrieval at model/serving levels (e.g., RETRO) [47, 48, 53]. Agent frameworks (e.g., LangChain) supply orchestration primitives (chains, tools, memory, routing) that accelerate integration, but they do not prescribe security posture by themselves [39]. This chapter treats those mainstream patterns as building blocks; Sect. 1.2 focuses on cybersecurity-specific architecture trade-offs (governance, isolation, provenance, orchestration) rather than reintroducing generic LLM background.

Figure 1.1 presents a reference depiction informed by widely adopted frameworks—NIST Cybersecurity Framework (CSF) 2.0 and the MITRE ATT&CK knowledge base—rather than derived from any single public survey [10, 11].

Foundational intentional and tool-augmented agent research (Belief–Desire–Intention (BDI) formalism, reflective/tool-use prompting, multi-role decomposition) supplies the abstraction basis for stateful cyber defense agents that map beliefs to telemetry + memory, desires to mission/policy objectives, and intentions to constrained action sequences [2, 3, 29, 37]. Benchmark studies and early multi-agent orchestration prototypes highlight gains in multi-step reasoning and contextual synthesis, while also exposing reliability and safety gaps under open-world adversarial conditions [5, 29].

Operational drivers include analyst workload reduction, triage acceleration, consistent containment workflows, and enriched situational reconstruction (timeline assembly, attack path correlation) [10, 26]. These benefits are mediated by architecture: coordination topology, memory and retrieval governance, execution isolation modality, and attestation depth jointly determine achievable responsiveness, auditability, and blast radius containment [8, 24, 26].

At a high level, the threat surface motivating these architectural levers spans injection, poisoning, supply-chain tampering, privilege escalation, drift, and emerging multi-agent risks; detailed categories and mitigations are deferred to Sect. 1.4. Accordingly, this chapter embeds provenance, attestation, identity, and adaptive guardrails into orchestration and memory pipelines rather than retrofitting standalone controls. For a focused treatment of LLM-specific concerns, see Sect. 1.4.1.1; for practical hallucination mitigation, see grounded reasoning in the Secure RAG pattern (Sect. 1.3).

References to widely adopted security and AI governance sources are used where possible; when drawing on recent work, claims are restricted to descriptive synthesis and prescriptive guidance is deferred until peer-reviewed or stably published.

Architectural trade-offs across coordination (centralized vs hybrid vs decentralized), agent granularity, policy adaptivity, memory tiering and retrieval diversity, orchestration style (planner-centric vs event-driven), execution isolation (containers, serverless, enclaves), and attestation depth yield distinct profiles for latency, robustness, explainability, and governance burden [4, 38–40]. Systematic comparison of these axes clarifies how design levers jointly influence emergent assurance properties and informs progressive hardening roadmaps [6, 7].

Methodologically, this chapter conducts a qualitative comparative synthesis drawing on security standards (NIST 800-53/800-160, AI RMF), adversary knowledge bases (MITRE ATT&CK), software supply chain and policy frameworks (SLSA, SPDX, SPIFFE/SPIRE, OPA), and observability standards (OpenTelemetry). It also references peer-reviewed or stable preprint sources on planning, tool-use, alignment (Constitutional AI, Reinforcement Learning from Human Feedback (RLHF)), poisoning, and prompt injection [6–8, 21, 22, 24–26, 41, 42, 45, 46]. Exploratory mechanisms lacking broad validation are sequestered to clearly labeled Research Frontiers to maintain epistemic transparency [6].

Contributions are fivefold: (i) layered reference architecture (identity, provenance, observability integration); (ii) structured comparison of coordination/orchestration archetypes with resilience and governance implications; (iii) secure retrieval-augmented generation (RAG) design pattern (poisoning containment and rollback levers); (iv) concise threat–control summary (preventive/detective/corrective mapping); and (v) open research questions (including multi-agent safety and collusion resilience) [21, 24, 26]. Together, these synthesize into practitioner guidance that preserves controllability while increasing autonomous capability [6, 7].

1.1.2 Positioning verses Prior Work

This chapter synthesizes security architecture and agentic AI practice into a layered reference architecture that integrates identity (SPIFFE/SPIRE), policy (OPA), provenance/attestation (SLSA/SPDX, runtime quotes), and observability (OpenTelemetry) with coordination/orchestration trade-offs. Relative to SOAR-style rule automation and mainstream LLM agent frameworks (e.g., ReAct, Toolformer, LangChain), the contribution is (i) a structured comparison of coordination/archetypes emphasizing resilience, explainability, and governance burden; (ii) a secure RAG pattern with tiered promotion and signed manifests for rollback; and (iii) an assurance-oriented metrics framing to guide future empirical work [2, 3, 39]. The focus is on posture (blast-radius containment and provenance continuity), ordering/consistency considerations, and scalability under governance—not on benchmark claims.

1.1.3 Deployment Considerations

Real-world rollouts introduce practical hurdles: (i) trust bootstrap and federation (establishing an initial root of trust, SPIFFE/SPIRE federation across tenants, cross-org identity mapping), (ii) policy bundle distribution and revocation (OPA bundle signing, version pinning, emergency rollbacks), (iii) runtime attestation operations (quote freshness SLOs, verifier hardening, time sync), (iv) key and secret management (HSM-backed signers, key rotation, scoped credentials), and (v) regulatory and data residency constraints (telemetry minimization, selective redaction). These are implementation choices rather than architectural absolutes; the patterns here (governance anchor, signed envelopes, promotion manifests) are intended to reduce blast radius and ease incident response irrespective of platform details. A common zoning pattern is to place the governance and attestation nucleus in a hardened, enclave-backed zone with strict SPIFFE/SPIRE identities, while running enrichment cohorts in more elastic but lower-trust zones with scoped credentials.

Scope delineation distinguishes baseline architectural mechanisms (grounded in deployed or convergent practice) from forward-looking hypotheses (explicitly marked) to mitigate hallucination risk and support rigorous auditability [6, 10]. The chapter also grounds threat-surface characterization in practical Internet-exposed asset reconnaissance (Shodan) and standardized vulnerability intelligence pipelines (Common Vulnerabilities and Exposures (CVE), National Vulnerability Database (NVD)) while referencing reinforcement learning texts that inform adaptive guardrail feedback design [32–34, 44]. Therefore, architectural trade-offs in areas like memory governance, tool orchestration, and trust enforcement become critical—and are the focus of this chapter. Subsequent sections follow the narrative: Sect. 1.2 introduces coordination/orchestration archetypes and examines key trade-off axes; Sect. 1.3 covers memory/retrieval hardening; Sect. 1.4 outlines research frontiers; and Sect. 1.5 synthesizes practitioner guidance [7, 21, 26].

1.2 Architecture Trade-Offs

This section systematizes coordination and orchestration topologies for agentic cybersecurity platforms, focusing on how structural choices mediate latency, resiliency, blast radius, and provenance fidelity [7, 8, 10].

The coordination substrate provides the control flow through which detection, enrichment, containment, and reporting roles interact; its topology thereby constrains achievable containment service-level objectives (SLOs), governs policy convergence latency, and shapes adversary opportunity for lateral prompt or retrieval injection. NIST systems security engineering and continuous monitoring guidance stress explicit threat modeling of control-plane concentration points and continuous monitoring of cross-agent causal chains to retain auditability at scale [8, 9].

This chapter considers three archetypes—centralized, decentralized, and hybrid—summarized in Fig. 1.1 and Table 1.1 and analyzed in detail in Sect. 1.2.2.1; mechanics and trade-offs are deferred to that subsection to avoid repetition. For clarity, the term *governance anchor* denotes a hardened central service that signs policies and authorizes privileged actions (e.g., remediation approval, secret release) while one or more sub-orchestrators or agent cohorts handle enrichment and local scheduling; the anchor can run in an enclave/confidential VM or strictly hardened container. This anchors global policy and identity while preserving parallelism and bounded blast radius in subordinate layers [38].

Figure 1.1 summarizes a reference layering: external interfaces feed an orchestration layer that allocates tasks across specialized agents drawing on multi-tier knowledge stores, all enveloped by governance (identity, policy, attestation) and observability (telemetry, tracing) strata. The depiction emphasizes that provenance capture and identity issuance are first-class architectural layers rather than afterthought operational add-ons. It is an illustrative consolidation of convergent practices and proposed enhancements rather than a claim of universal deployment [21, 26, 39].

Enterprise security automation platforms (SOAR extensions) commonly instantiate the centralized archetype (planner + enrichment tools), explicitly used here to surface trade-offs in policy convergence latency, compromise blast radius and auditability density; meanwhile open-source LLM agent frameworks and research prototypes demonstrate planner–executor splits and recent multi-agent decomposition (e.g., tool routing, verifier/reviewer roles). These ecosystems (LangChain, HuggingGPT, AutoGPT) provide composable abstractions that accelerate experimentation yet can conceal governance gaps (e.g., unsigned inter-agent messages) if not coupled with identity and policy attestation [4, 39, 40].

Centralized planners simplify global policy rollouts (single signed bundle) and reduce ordering ambiguity. They increase explainability density per unit trace volume (ratio of semantically interpretable causal/context tokens to total emitted trace payload—a proxy for analyst effort efficiency) but concentrate privilege and can saturate quickly under burst workloads. Conversely, decentralized or hybrid enrichment clusters distribute capacity and fault domains, shrinking individual compromise blast radius but expanding identity issuance, drift detection, and consensus monitoring

Table 1.1 Comparative characteristics across centralized, hybrid, and decentralized coordination topologies, highlighting latency scaling, policy convergence, provenance granularity, failure modes, and operational surface

Dimension	Centralized	Hybrid	Decentralized
Policy convergence	Fast (single nucleus)	Fast governance, moderate enrichment	Slow (multi-hop propagation)
Blast radius	High (controller compromise global)	Layer-bounded	Low (per-peer/shard)
Tail latency under burst	Steep cliff	Moderate	Shallow (parallel absorption)
Provenance reconstruction	Simple (total order)	Layered merge (anchor+cohorts)	Complex (trace merge + clocks)
Identity/message surface	Minimal	Moderate	High
Governance agility	High until saturation	High (anchored)	Variable (consensus lag)
Operational complexity	Low	Moderate	High (protocol/drift)

surfaces. Architectural selection should therefore be grounded in quantified incident concurrency expectations, acceptable containment service-level agreement (SLA), governance staffing constraints, and the tolerance for reconstruction complexity during post-incident audits [6, 25, 26].

Table 1.1 consolidates comparative characteristics (latency scaling, policy convergence, provenance granularity, failure modes) across the archetypes and execution environments, setting the stage for the subsequent, more granular trade-off analyses (Sects. 1.2.2 and 1.3) [8, 21, 26] (Fig. 1.2).

1.2.1 Architectural Patterns Overview

Agentic cybersecurity platforms instantiate recurring structural archetypes that balance global coordination authority with distributed analytical capacity. At a high level these archetypes channel classic distributed systems concerns (fault containment, consensus overhead, ordering guarantees) into a security operations context that emphasizes provenance fidelity, adversarial robustness, and containment latency service-level objectives. Centralized, decentralized, and hybrid patterns summarized in Fig. 1.1 and Table 1.1 serve as the substrate for the detailed trade-off analysis in Sect. 1.2.2.1 [7, 8, 38].

Purely decentralized (peer/swarm or blackboard-mediated) patterns diffuse scheduling and enrichment across multiple autonomous agents that negotiate roles or exchange intermediate artifacts without a singular choke point. This diffusion increases horizontal scaling headroom and graceful degradation (capacity loss >

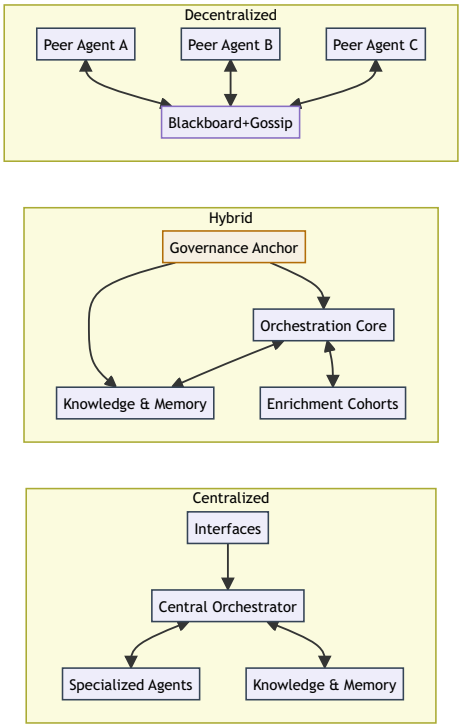


Fig. 1.2 Illustrative coordination topologies: centralized (single orchestrator), hybrid (governance anchor with sub-orchestrators/cohorts), and decentralized (peer/bus or blackboard). Dashed border denotes the governance anchor (trust root) in the hybrid topology

capability loss) while imposing higher policy convergence latency and greater identity + message surface for potential prompt or retrieval injection. Strong workload identity issuance and signed telemetry (e.g., SPIFFE/SPIRE SVIDs plus structured traces) become preconditions to avoid provenance fragmentation and undetected policy drift. Emerging, nonpeerreviewed news reports describe experimental decentralized or hybrid deception cohorts (GPT4powered honeypot swarms) used to elicit attacker TTPs; these accounts are anecdotal and should not be read as evidence of widespread operational practice [24, 26, 35, 38].

Hybrid layering couples a hardened governance and attestation anchor (central policy bundle signing, identity root, escalation gating) with horizontally scaling enrichment or deception clusters (e.g., distributed honeypots, specialized analyzers). This approach seeks a middle path: retaining near-centralized policy propagation latency and forensic root-of-trust guarantees while localizing compromise impact to enrichment cohorts. Runtime attestation, policy hash convergence telemetry, and selective enclave or serverless isolation refine trust gradients across layers [7, 21, 25].

Early preprint and lab reports on deception-oriented fabrics (multi-agent honeypot/lure cohorts) similarly stress tracing and causal stitching requirements; these are exploratory rather than established practice but illustrate why signed span chains and divergence monitoring matter, as discussed in Sect. 1.4 [24, 26, 36].

These archetypes form the comparative substrate for the subsequent analysis of trade-off dimensions. Their properties (policy convergence latency profile, decision tail latency resilience, provenance granularity, compromise blast radius, observability density, and operational complexity) motivate a multi-axis evaluation methodology rather than single-metric optimization [5, 6, 26].

1.2.2 Key Trade-Off Dimensions

The architectural levers evaluated in this chapter decompose into interrelated dimensions whose joint configuration determines emergent safety, resilience, and governance characteristics: (i) coordination model (centralized, decentralized, hybrid), (ii) agent granularity and role specialization, (iii) policy mode (static vs adaptive guardrails), (iv) execution isolation substrate (containers, serverless, enclaves), (v) memory and retrieval tiering/quorum strategy, (vi) orchestration style (synchronous planner vs event/intent bus), and (vii) attestation depth (build-only vs end-to-end runtime). Each dimension modulates latency distributions (mean vs tail), blast radius containment, provenance fidelity, and adversarial manipulation surface; their interactions preclude naïve, isolated optimization of any single axis. [5, 8, 26]

Coupling effects are material. Increased agent granularity raises inter-agent protocol surface and message latency variance. Designs frequently shift toward event/intent buses for elasticity; that shift elevates ordering reconstruction complexity and demands richer tracing (causal span graphs plus policy hash convergence signals—time series of bundle hash agreement). Deep memory tiering (ephemeral → probation → vetted) increases ingestion latency but enables selective rollback and poisoning containment. This trade is best amortized when orchestration supports asynchronous promotion and provenance-rich retrieval ranking. Elevating attestation depth (runtime quotes, signed policy bundles) similarly adds startup and verification overhead; placing attestation at natural choke points (central governance hub) or caching and batching verification in event-driven fabrics minimizes impact [21, 25, 26].

A principled evaluation therefore frames each dimension in terms of: (a) primary assurance objective (e.g., provenance continuity, poisoning rollback locality, lateral movement containment), (b) quantitative cost drivers (queue depth growth rate, policy convergence latency slope, attestation verification overhead, ingestion promotion delay), (c) residual risks shifted elsewhere (identity proliferation, ordering ambiguity, drift surface), and (d) instrumentation prerequisites (signed spans, policy bundle hashes, promotion quorum manifests). This multi-perspective lens accords with risk management and systems security engineering guidance emphasizing balanced advancement of robustness, explainability, and controllability rather than unilateral performance optimization. [6, 7, 10]

The subsections that follow specify for each axis the characteristic strengths, systemic limits, security posture implications, latency profile, and indicative operational telemetry, establishing an evidence-oriented basis for staged architectural evolution

(e.g., centralized → hybrid enrichment clusters → selective decentralization) aligned to empirically observed workload and threat dynamics [3, 4, 38].

1.2.2.1 Coordination Model (Centralized/Decentralized/Hybrid)

Dimension Framing. The coordination model governs how reasoning, scheduling, and policy enforcement authority are distributed, thereby shaping policy convergence latency, containment blast radius, provenance granularity, and tail decision latency under burst load. A comparative analysis therefore evaluates structural concentration versus diffusion not only in classic availability terms but also in security assurance (attestation anchoring) and auditability (causal reconstruction effort), assuming standard consensus and idempotent messaging mechanisms (e.g., Raft-style log replication) from the distributed-systems literature rather than re-deriving them here [8, 26, 38].

Centralized Model. A logically singular (possibly replicated) orchestrator performs task decomposition, scheduling, and guardrail enforcement, yielding low coordination overhead and rapid global policy propagation while unsaturated. Deterministic or near-total ordering simplifies forensic replay and reduces ambiguous causal branching, and unified telemetry aggregation eases anomaly attribution. Principal liabilities include queue saturation cliffs, privilege concentration (single compromise can rewrite global policy surface), and a consolidated failure domain even when high-availability replicas provide liveness. Replication (active/standby or deterministic multi-leader) mitigates availability loss but leaves a residual trust monoculture necessitating rigorous hardening and attestation of the control nucleus [4, 7, 26].

Decentralized Model. Peer (swarm/blackboard/gossip) coordination diffuses enrichment and partial planning across autonomous agents, typically supporting near-linear horizontal scale and more graceful degradation in which capacity can diminish without immediate global capability loss. The diffusion generally increases policy convergence latency (multi-hop propagation, potential stale caches), expands identity and message surface (more injection or spoofing vectors), and introduces consensus or gossip overhead that can increase median decision latency for globally synchronized actions. Maintaining provenance fidelity therefore requires strong workload identity (SPIFFE/SPIRE), signed envelopes, bundle hash continuity checks, and aggregation pipelines for reconstructing causal chains from distributed trace fragments. Some non-peer-reviewed reports describe decentralized/hybrid deception cohorts (GPT-4-powered honeypot swarms) and simulated engagements; these accounts are anecdotal and preliminary [24, 26, 35, 38].

Ordering and state convergence. Some decentralized fabrics reconcile shared state with conflict-free replicated data types (CRDTs) or gossip-based merge protocols, trading immediate consistency for eventual convergence. This improves availability and reduces coordinator bottlenecks, but shifts merge/reconciliation burden into the application layer and can elongate policy convergence for privileged actions that still require quorum (e.g., Raft) or explicit approval.

Hybrid Model. A hardened governance and attestation anchor (policy bundle signing, identity root issuance, escalation gating) is coupled to horizontally scalable enrichment, deception, or analytic clusters. This stratification preserves centralized policy convergence speed and authoritative provenance root while confining adversary-facing or high-churn logic to lower trust zones that can fail or be compromised with bounded blast radius. Added complexity arises from dual control planes, potential configuration drift, and broader coordination code surface; integrity barriers (signed task manifests, version pinning, runtime attestation verification) and anomaly analytics (bundle hash continuity/mismatch alerts plus span correlation outlier detection) become mandatory to prevent silent divergence [8, 21, 25].

Comparative Performance and Resilience. Centralized controllers favor vertical scaling (thread pooling, batching) achieving minimal median decision latency until saturation triggers a rapid degradation slope; decentralized and hybrid designs favor horizontal scaling at the cost of a consistency tax (log replication delays, convergence lag). Hybrid architectures reduce policy convergence latency relative to fully decentralized peers by anchoring distribution at a central trust root while still permitting independent enrichment throughput scaling along peripheral clusters [24, 26, 38].

Security Posture. Centralization minimizes identity count and message surface but amplifies the impact of orchestrator compromise (global policy issuance authority). Decentralization introduces identity proliferation and necessitates automated credential rotation, attested messaging, and behavior baselining to detect collusion or lateral movement. Hybrid layering couples hardened central anchors (e.g., enclave-protected controller or strictly hardened container with mutual Transport Layer Security (mTLS)) to constrained egress and provenance-gated enrichment zones to achieve layered defense with bounded trust expansion [7, 24, 25].

Latency Taxonomy. Distinguishing (i) decision latency (planner or consensus cycle), (ii) policy convergence latency (time to activate a new guardrail globally), and (iii) overload degradation slope (queue depth vs throughput) enables data-driven migration triggers. Empirically, centralized patterns show low (i) and (ii) until a saturation cliff; decentralized patterns elevate (i) and (ii) yet maintain shallow overload slopes; hybrid patterns retain low governance decision latency with moderate enrichment latency and shallow slopes due to shard absorption. Instrumentation of queue depth, consensus round duration, and policy hash convergence times underpins quantitative evolution planning and feeds internal metrics such as "explainability density" and "auditability density" (coined terms defined in the Glossary; see Glossary for precise definitions) [6, 25, 26].

Scenario note. To avoid duplication, the consolidated incident walkthrough in Sect. 1.5.3 illustrates centralized, decentralized, and hybrid behavior during lateral movement; this subsection focuses on architectural trade-offs. [7, 10] *Key takeaway.* Centralized maximizes clarity and rapid guardrail rollout but concentrates privilege; decentralized diffuses risk and scales horizontally at audit/ordering cost; hybrid anchors trust centrally while distributing enrichment. Select posture based on incident concurrency, containment SLOs, and audit workload rather than ideology;

an end-to-end incident narrative in Sect. 1.5.3 illustrates these coordination choices during lateral movement [7, 10].

1.2.2.2 Agent Granularity (Single vs. Multi-Agent)

Dimension Framing. Agent granularity determines how functional responsibilities (detection, enrichment, containment planning, remediation execution, reporting, verification) are partitioned across reasoning units. This partitioning modulates (i) decision/coordination latency, (ii) specialization efficacy (smaller or tuned models per role), (iii) provenance artifact richness, (iv) least-privilege scoping feasibility, and (v) adversarial injection surface (number and heterogeneity of prompt/message boundaries). Figure 1.3 illustrates this contrast [3–5, 29].

Monolithic (Single) Agent Pattern. A single large model (or tightly coupled chain) performs end-to-end perception, reasoning, and action selection within one contiguous context window. Strengths include minimal orchestration overhead (no inter-agent protocol negotiation), holistic situational awareness (all recent facts resident in one prompt), unified deployment and rollback pipeline (one artifact/policy bundle), and dense explainability within a single reasoning trace. Limitations encompass constrained deep specialization (prompt role emulation lacks weight-level optimization), slower feature iteration (broad regression impact of any fine-tuning), privilege concentration (broad tool/application programming interface (API) scopes co-resident), and expanded blast radius for prompt or retrieval injection (one compromise can steer the entire decision chain) [2, 5, 29].

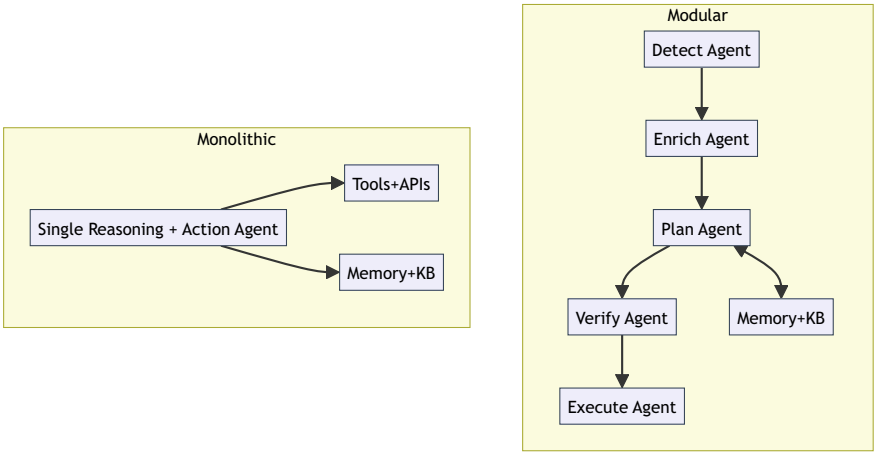


Fig. 1.3 Agent granularity contrast: (left) single monolithic agent with unified context and minimal coordination hops; (right) modular chain of specialized agents with inter-agent messages and richer provenance checkpoints. The depicted sequence (Detect → Enrich → Plan → Verify → Execute) is one illustrative decomposition for complex tasks, not a prescriptive standard

Modular Multi-Agent Pattern. Functional decomposition across specialized agents (e.g., detection, enrichment, containment planner, policy verifier, remediation executor, reporter) enables smaller or fine-tuned models per role, targeted scaling (add more enrichment workers without retraining planner), and least-privilege boundary enforcement (role-scoped credentials and tool sets). Express inter-agent messages and structured tracing (spans with role + intent metadata) improve causal auditability and facilitate defense-in-depth guardrails (verifier/reviewer roles). Added costs arise from coordination latency (context serialization + transfer), protocol evolution overhead (implicit conventions drift without schemas), higher operational surface (identities, certificates, policy bundles), and increased log/trace volume [4, 21, 24–26].
Model size and training implications. Granular designs favor smaller per-role models or lightweight adapters/fine-tunes trained on task-specific data, reducing regression blast radius and enabling independent iteration cadences. Monolithic designs centralize capability in a single large model, simplifying deployment artifacts but making weight-level updates costlier and riskier (broader revalidation), and limiting opportunities to shrink or specialize parameters per function.

Latency and Throughput. Monoliths minimize per-decision coordination hops and often achieve lower median latency for narrow task families. They encounter a scaling inflection once concurrent heterogeneous tasks compete for a shared context length or token budget. Modular systems introduce serialization and handoff overhead yet unlock horizontal parallelism (independent enrichment, concurrent remediation candidates) that reduces tail decision latency under burst workloads. Quantitatively, modular speedups only materialize beyond a concurrency threshold where parallel enrichment outweighs added message overhead [3, 8, 26].

Provenance and Explainability. Single-agent designs depend on internal chain-of-thought (frequently suppressed in production) plus external tool I/O for forensic reconstruction, limiting granularity of intermediate rationale. Multi-agent designs naturally emit a causal graph ($\text{Agent}_{detect} \rightarrow \text{Agent}_{enrich} \rightarrow \text{Agent}_{plan} \rightarrow \text{Agent}_{verify} \rightarrow \text{Agent}_{execute}$) whose edges can be signed and hashed to produce a tamper-evident provenance chain. This improves anomaly localization (e.g., mis-scoring isolated to enrichment) and targeted guardrail refinement, at the cost of higher trace storage and potential privacy considerations around intermediate summaries [21, 25, 26].

Security Posture and Attack Surface. Modular decomposition reduces per-role credential scope and enables differential hardening (e.g., enclave-protect containment planner; run enrichment in serverless sandboxes). It simultaneously expands channels susceptible to prompt/retrieval injection, data poisoning propagation, spoofed task directives, or collusive message crafting. Mitigations include signed inter-agent envelopes (SPIFFE/SPIRE identities + mutual Transport Layer Security (mTLS)), schema and content validation at each boundary, provenance hash chains, policy hash convergence telemetry, and per-role rate/intent gating. Monolithic patterns reduce channel count but concentrate privileges; a successful injection achieves broader lateral influence and complicates containment rollback [10, 12, 13, 23, 24].

Governance and Operational Complexity. Microservice-like modularity aligns with independent team ownership and CI pipelines (role-specific regression suites, targeted evaluation benchmarks). Deployment blast radius shrinks: updating enrich-

ment logic does not require revalidating containment planner invariants. Tooling overhead increases (policy bundle distribution, identity issuance, attestation coverage tracking) and necessitates cataloging interface contracts (message schemas, allowed intents) to prevent silent protocol drift. Monolith governance centralizes these concerns but elongates change control cycles (full regression sweep) and inhibits parallel experimentation [7, 8, 21, 25].

Evolution Guidance. Favor a monolithic pattern early when scope is narrow, latency budgets are tight, and provenance requirements can be satisfied with coarse reasoning traces. Introduce modular roles when (i) heterogeneous competencies (e.g., static signature enrichment vs creative candidate generation) cause context contention, (ii) risk analysis shows privilege separation materially shrinks blast radius, or (iii) compliance/audit demands fine-grained provenance artifacts. Transitional hybrids (monolithic reasoning core plus lightweight verifier or reporter agents) allow incremental risk reduction before full decomposition. Decomposition readiness indicators include rising false positive remediation overrides localized to particular logical sub-tasks, increasing context window pressure, and sustained queue depth growth for burst enrichment [5, 6, 26, 29].

Scenario note. A consolidated end-to-end example in Sect. 1.5.3 shows granularity choices in context; per-subsection vignettes are omitted here to keep Sect. 1.2 concise [10]. *Key takeaway.* Monoliths minimize coordination latency but concentrate privilege and compress provenance; modular chains add handoff overhead yet unlock least-privilege boundaries, parallel enrichment, and granular audit trails. Begin monolithic; decompose as specialization and concurrency pressures rise; Sect. 1.5.3 provides an integrated incident walkthrough illustrating these granularity choices [10].

1.2.2.3 Policy Mode (Static vs. Adaptive Guardrails)

Dimension Framing. Policy mode governs whether guardrails (prompt filters, action permission rules, risk scoring thresholds, remediation approval invariants) evolve only through explicit release events (static) or via governed feedback loops (adaptive). Orchestration frameworks that support dynamic tool-use and policy routing make adaptive modes operational in practice [4–7, 25]. The choice shapes (i) protection coverage half-life under adversarial innovation, (ii) explainability stability, (iii) provenance chain complexity, (iv) blast radius of data/feedback poisoning, and (v) operational governance workload.

Static Mode. Versioned but non-self-modifying guardrails fix decision boundaries between releases. Strengths: determinism (decision for input class X reproducible via attested policy hash), bounded regression scope (no intra-epoch drift), simpler certification (single artifact lineage: source → build → signed policy bundle), and reduced attack surface for feedback manipulation. Limitations: coverage decay under emergent prompt/retrieval injection, novel evasive phrasing, or shifting baseline distributions (concept drift), plus elongated exposure windows when human change cycles lag adversary iteration pace [5, 21, 25].

Adaptive Mode. Governed learning loops adjust thresholds, rule sets, embedding similarity filters, or classifier weights using telemetry (analyst override labels, drift detectors, adversarial evaluation deltas). Benefits: rapid incorporation of novel attack patterns (shrinking mean window of exposure), maintenance of protection-task success balance under distribution shift, and reduced manual tuning toil. Risks: instability (policy flapping, oscillatory thresholds) if update cadence or learning rate poorly bounded; feedback gaming/poisoning (adversary injects mislabeled or crafted samples to relax controls); opacity creep (decision rationale depends on post-audit state). Adaptive viability requires change windows, rollback checkpoints, guard-banded metrics, and append-only provenance of diffs. Representative approaches include AI-feedback-driven policy tuning (e.g., Constitutional AI) and safety guidance from application security baselines [5, 6, 12, 26, 42, 45].

Provenance and Explainability. Static policies enable single-layer attribution: mapping input features to a stable rule or deterministic threshold. Adaptive layers necessitate multi-layer explanation: (a) static core invariant justification, (b) adaptive delta rationale (e.g., new phishing template cluster expansion), and (c) data slice provenance (hash of contributing training/evaluation subset). Instrumentation should emit span attributes (policyVersion, policyHash, changeId, datasetDigest, decisionFeatureHash) so auditors reconstruct pre/post contexts. Policy evolution forms a hash chain (prevHash $\rightarrow$ newHash) signed with workload identity (SPIFFE/SPIRE SVID) and embedded into attested traces. [21, 24–26]

Stability & Control Loops. Mitigate oscillation with bounded update frequency (batch nightly or weekly rather than continuous), hysteresis (minimum sustained deviation from target metric before adjustment), shadow/dual-run (evaluate candidate policy in observe-only mode), and canary promotion (subset of incidents or tenants) prior to global activation. Guard-bands for key metrics (e.g., containment false positive rate $\leq$ target + 2

Security Posture and Attack Surface. Static mode concentrates risk in prompt/retrieval injection and direct rule manipulation; adaptive adds data/feedback poisoning (malicious override labels, crafted borderline examples) and drift detector gaming. Defenses: signed labeled feedback manifests (annotator identity + timestamp + sample hash), quorum or reputation weighting of feedback, anomaly screening of proposed policy deltas, and separation of duties (training pipeline cannot directly promote changes without verifier approval). Runtime divergence telemetry (expectedPolicyHash vs activePolicyHash) raises alerts on unauthorized mutation [12, 24, 25, 42].

Operational Monitoring. Track: (i) protection effectiveness (block/containment success rate), (ii) task success/utility impact, (iii) override disagreement rate (analyst overturn frequency), (iv) policy convergence latency (time from new pattern detection to guardrail update), (v) signed policy bundle hash propagation delays or mismatches, (vi) false containment/false allow rates, (vii) drift detector alert precision, and (viii) guardrail adjustment volatility (frequency/variance of threshold changes). Rising override disagreement with stagnant coverage suggests mislabeled feedback or overfitting; increasing convergence latency signals need for additional automation [5, 7, 9, 21, 26].

Evolution Guidance. Start with static core to establish baseline metrics, calibrate risk tolerances, and build provenance + attestation pipeline. Introduce bounded adaptive overlays when measurable staleness (rising unblocked prompt injection rate, degradation in protection-task success balance) crosses predefined thresholds and governance primitives (diff signing, rollback automation, shadow evaluation) are operational. Constrain adaptation scope initially (augment allow/deny lists, adjust weight scaling factors) before permitting structural model updates. Mature posture maintains immutable high-risk invariants (e.g., disallow self-escalation actions) while permitting adaptive tuning of contextual heuristics [6, 21, 25, 26].

Scenario note. For a unified narrative contrasting static and adaptive guardrails under attack evolution, see Sect. 1.5.3; details are centralized there. [5] *Key takeaway.* Static policies deliver determinism and simple audits; adaptive overlays cut exposure windows but require guard-bands, signed diffs, and rollback to avoid drift or poisoning. Keep core invariants static while adapting low-risk heuristics; a comprehensive narrative is provided in Sect. 1.5.3 [5].

1.2.2.4 Execution Isolation (Containers/Serverless/Enclaves)

Dimension Framing. Execution isolation defines containment boundaries for agent processes and tool invocations, shaping (i) blast radius of compromise (credential, memory, network surface), (ii) feasibility of in-use confidentiality/integrity guarantees, (iii) cold start and tail latency budgets, (iv) least privilege granularity, and (v) depth of attestable provenance available at runtime. Substrate selection (containers, serverless, Trusted Execution Environments (TEEs)/confidential virtual machines (VMs)) alters which security invariants can be enforced cheaply vs which require compensating controls [7, 8, 21].

Terminology. eBPF stands for extended Berkeley Packet Filter; IAM stands for Identity and Access Management; SGX stands for Intel Software Guard Extensions; SEV stands for AMD Secure Encrypted Virtualization; SNP stands for Secure Nested Paging; EPC stands for Enclave Page Cache; RPC stands for remote procedure call.

Containers. Namespaced processes (cgroups, PID/net/user namespaces) with policy hardening (seccomp, AppArmor/SELinux) provide the lowest steady-state latency and a mature supply-chain toolchain (SBOM generation, signature verification, image scanning). Strengths: rapid warm reuse (minimized startup jitter), ubiquitous observability (sidecar collectors, eBPF tracing), and straightforward integration of inline policy engines. Limitations: shared kernel expands exploit blast radius (kernel escape yields multi-container impact), limited in-use secrecy (unencrypted DRAM) absent host/platform anchoring, and weaker runtime integrity evidence (build attestations do not prove unmodified execution). In Fig. 1.4, containerized enrichment and analytics map to Zone B. Hardening patterns: distroless/minimal base image, read-only root filesystem, dropping ambient capabilities, per-agent dedicated network namespace, user namespace remap, syscall allowlist telemetry, and continuous image drift scanning [7, 21–23, 26].

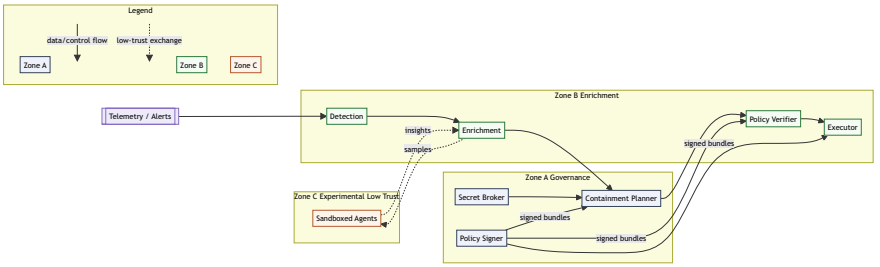


Fig. 1.4 Three-zone execution isolation: governance in TEEs/confidential VMs, enrichment in containers/serverless, and experiments in a low-trust zone, bounding blast radius while keeping privileged actions gated (see Glossary for acronyms)

Serverless Functions. Event-driven, ephemeral functions (often backed by micro-VM or sandbox isolation) reduce residual state persistence and enforce fine-grained IAM scoping per function. Benefits: automatic scale-out under telemetry bursts, short-lived credentials (reduced theft utility), geographic replication for resilience, and pay-per-execution cost alignment. In Fig. 1.4, serverless workers typically inhabit Zone B for burst absorption, while Zone C denotes low-trust experiments with restricted egress and no privileged actions. Costs: cold start latency (model adapter load, dependency hydration), fragmented execution context (necessitating external session/memory store), and reduced locality for multi-step reasoning (state re-fetch overhead). Forensics complexity increases when transient caches influence behavior without durable logs. Operational tactics: pre-warm pools for latency-critical containment flows, signed per-function policy bundle injection (OPA rego snapshot), structured correlation IDs across chained functions, and adaptive concurrency throttling to bound downstream saturation risk [8, 25, 26].

Trusted Execution Enclaves/Confidential VMs. Hardware-backed isolation (e.g., SGX EPC, AMD SEV/SNP, or confidential VM enclaves) supplies sealed memory, measurement, and remote attestation enabling cryptographic verification of loaded model weights, policy bundles, and critical code before granting privileged actions. Strengths: mitigated host memory scraping, attested policy hash embedding in trace spans, and higher-assurance runtime integrity. As illustrated in Zone A of Fig. 1.4, a governance nucleus (policy signer, remediation planner, secret decryption service) can reside in TEEs or confidential VMs. Constraints: attestation quote latency on first invocation, limited protected memory capacity (pressure on large embedding/model artifacts), additional operational services (quote validation, key provisioning), and residual side-channel classes. A common pattern confines the governance nucleus (policy signer, remediation planner, secret decryption service) to TEEs while delegating high-throughput enrichment/retrieval expansion to containers or serverless for cost and elasticity [17–21, 24, 25].

Security Posture and Attack Surface. Containers emphasize supply-chain integrity and syscall boundary reduction; risks concentrate in kernel exploits, lateral movement via shared host, and opaque drift (unlogged binary mutation). Server-

less reduces persistence and credentials dwell time but introduces injection risk into ephemeral initialization code paths and potential multi-tenant side channels. TEEs elevate confidentiality/integrity but expand attack surface to enclave loader, quote verification pipeline, and side-channel classes (cache timing, page-fault or speculative leakage). Cross-cutting mitigations: layered signing (image + policy + configuration), runtime policy hash assertion (expectedPolicyHash vs active), minimal syscall profiles, per-function IAM scoping, enclave pre-measurement caching with revocation lists, and anomaly detection on syscalls or attestation failure patterns [7, 17–21, 24].

Provenance and Instrumentation. Enforce span attributes (isoMode = container|serverless|enclave, policyHash, imgDigest, qDigest, iamScopeId)

so traces reconstruct heterogeneous isolation zones. Containers rely on build + deploy attestations (SLSA provenance) plus host posture attest (e.g., measured kernel). Serverless instrumentation must stitch distributed, ephemeral contexts using durable correlation IDs and capture cold start vs warm invocation markers. Enclaves extend provenance with attestation quote evidence (measurement hash chain) and binding of secrets release to verified measurement [21, 24, 26].

Latency and Resource Trade-offs. Containers: lowest steady-state and minimal per-invocation overhead; Serverless: added cold start (provision + JIT), mitigated by provisioned concurrency/pre-warming; Enclaves: additional attestation + memory encryption overhead on first-use and potentially during EPC paging. Optimization levers: (i) connection/model adapter pooling for serverless, (ii) enclave pooling and pre-attested idle warm set, (iii) tiered placement of latency-sensitive guardrail evaluators outside enclaves with attested results forwarded, and (iv) batching low-priority enrichment during off-peak windows. Track P50/P95/P99 decision latency segmented by isolationMode plus cold start rate and attestationLatency distribution [7, 24, 26].

Metrics. Core indicators: (a) isolation breach attempts (blocked syscalls, denied capability escalation), (b) policyHash divergence incidents, (c) cold start percentage (per hour) and its contribution to P95 latency, (d) attestation success/failure ratio and mean latency, (e) secret access locality (which isolation modes accessed high-sensitivity secrets), (f) enclave memory pressure (EPC utilization), and (g) rollback MTTR after compromised component detection. Rising cold start share with flat average latency suggests hidden tail risk; elevated attestation failures may signal supply-chain tampering or quote verification drift [21, 24, 26].

Evolution Guidance. Phase 1: containers to maximize iteration velocity and observability maturity. Phase 2: introduce serverless for bursty, stateless enrichment once queue depth or concurrency-induced tail latency surpasses threshold. Phase 3: confine governance-critical or high-secrecy functions (policy signing, high-privilege remediation planning, key management) inside TEEs/confidential VMs when (i) compliance requires in-use protection, (ii) memory scraping attempts rise, or (iii) supply-chain trust needs runtime verification. Maintain feature-flagged fallback to container mode for enclave regression handling while preserving attested build provenance and tracing [6, 17, 21, 26].

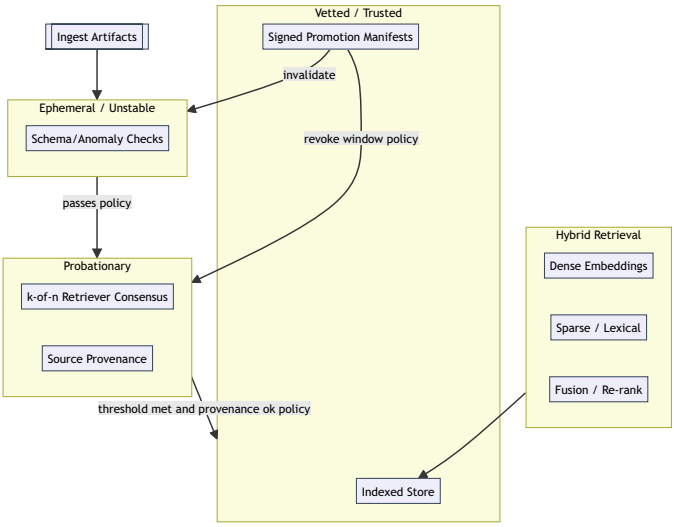


Fig. 1.5 Tiered memory governance: Ephemeral → Probationary → Vetted tiers with promotion and rollback levers, hardening the knowledge base by requiring consensus and provenance before trust

Scenario note. Isolation posture shifts (containers → serverless → enclaves) are demonstrated in the end-to-end incident walkthrough in Sect. 1.5.3; we omit a standalone vignette here [24, 26]. *Key takeaway.* Containers give velocity and lowest steady latency; serverless absorbs bursts with cold-start trade-offs; enclaves/confidential VMs add runtime integrity and secrecy at attestation cost. Place governance nuclei in TEEs and scale enrichment with containers/serverless for balance; Sect. 1.5.3 illustrates isolation posture shifts (containers → serverless → enclaves) in an end-to-end incident walkthrough [24, 26].

1.2.2.5 Memory/Knowledge Strategy (Monolithic vs. Tiered/Quorum)

Dimension Framing. Memory/knowledge architecture governs how telemetry-derived facts, extracted entities, and synthesized summaries are persisted, validated, weighted, and retrieved for agent reasoning. Design choices modulate: (i) poisoning blast radius, (ii) rollback mean time to recovery (MTTR) after contamination, (iii) retrieval precision/recall and robustness under injection, (iv) provenance audit granularity (document-level vs promotion lineage), and (v) trust stratification (differential policies by content source/validation state). [8, 41, 42]

Monolithic Store. A single merged vector/document index (uniform embedding + ranking pipeline) minimizes ingestion latency, reduces write amplification (one write path), and simplifies operational management (homogeneous consistency + caching). Strengths: lowest complexity footprint, predictable retrieval latency dis-

tribution, and straightforward global instrumentation. Risks: high-impact poisoning (malicious artifact instantly eligible for all future retrieval), coarse-grained rollback (laborious selective purge or full rebuild), accumulation of concept drift (stale summaries remain influential), and difficulty applying heterogeneous trust weighting (single trust schema). Observability focuses on aggregate distribution shifts (embedding centroid drift, average similarity score variance) and global precision degradation before/after ingestion windows [26, 42, 50].

Tiered + Quorum Promotion (proposed exemplar). A multi-stage pipeline (ephemeral/unstable staging → probationary → vetted) can enforce progressive hardening: (a) syntactic/schema validation and anomaly heuristics, (b) consensus or quorum scoring (k-of-n retrievers or models agreeing above confidence threshold), and (c) provenance/attestation checks (signed ingest manifest, retriever version hash) prior to promotion. Potential strengths: localized containment (invalidate staging tier to neutralize newly detected poisoning), reduced long-horizon drift (only vetted influences strategic reasoning), explicit metadata (trustLevel, promotionRationale, sourceDigest) enabling retrieval weighting. Trade-offs include ingestion latency (promotion delay), operational complexity (state tracking, garbage collection), elevated storage overhead, and risk of under-promotion causing recall loss on benign novel patterns [21, 25, 26, 41].

Illustrative promotion criteria (non-exhaustive). To make the gating concrete without prescribing thresholds:

- k-of-n consensus across heterogeneous retrievers exceeds a confidence band; no single retriever vetoes due to anomaly flags.
- Source provenance attested (e.g., domain allowlist or signed artifact); document digest and pipeline hash recorded in a promotion manifest.
- Freshness window satisfied (recency within policy band) and retrieverSetVersion matches the indexing pipeline version in effect.
- Content passes schema/PII/toxicity filters and does not conflict with denylists or incident holdouts.
- Shadow retrieval stability: result remains selected across multiple spaced queries before elevation (guards against transient spikes).
- Separation of duties for sensitive classes: human/verifier approval recorded in the signed manifest before promotion.

Sharded Domain/Semantic Partitioning. Partitioning by domain (hosts, network segments, threat tactic families) or semantic clusters bounds cross-domain poisoning propagation and enables differentiated retention/compaction/encryption policies. Strengths: blast radius containment, improved cache locality, parallel shard re-index operations, and opportunity for trust-isolated zones (regulated vs. general telemetry). Costs: federated query orchestration (merge + re-rank), duplication across shards (stale diverging copies), and need for trust-aware weighting when overlapping results appear [7, 26].

Retrieval Diversity and Consensus. Combining heterogeneous retrievers (dense embedding models with different architectures, sparse lexical, hybrid fusion) or

applying k-of-n consensus elevates robustness to targeted poisoning and prompt injection by requiring corroboration across independent similarity spaces. Strengths: reduces single-model bias and adversarial embedding alignment exploitation. Costs: multiplicative latency and tuning overhead (weight calibration), potential recall degradation if consensus threshold too strict, and increased instrumentation complexity (per-retriever score distributions) [26, 41, 42, 50].

Poisoning Containment and Rollback Mechanics. Effective containment couples: (i) per-document provenance hash (source URL, pipeline digest, retriever versions), (ii) temporal snapshots or incremental diff checkpoints, (iii) signed promotion manifests (approver identity, validation metrics, consensus scores), (iv) quarantine state (suppress from retrieval while investigation pending), and (v) scripted rollback playbooks (invalidate probation tier, rebuild vetted from last clean manifest chain). Metrics: promotion rejection rate, time-to-detect poisoned artifact, rollback MTTR, consensus divergence (variance among retriever votes), and false promotion rate (post-hoc demotions/revocations) [21, 26, 42].

Provenance and Instrumentation. Spans should emit attributes: memoryTier, docDigest, promotionManifestHash, retrieverSetVersion, trustLevel, consensusScoreVector (hashed or bucketized), and retrievalDecisionId linking to reasoning tokens referencing content. Hash chaining promotion manifests (prevHash → newHash) provides tamper-evident lineage; tying each manifest to an attested retriever model hash prevents silent model swap altering semantics without promotion scrutiny. [21, 25, 26]

Security Posture and Attack Surface. Monolithic mode concentrates poisoning leverage: a single crafted high-similarity lure can influence multiple future actions. Tiered + quorum reduces immediate leverage but expands operational surface (promotion service, manifest signer, consensus scheduler). Sharding curtails lateral poisoning but introduces cross-shard merge logic (potential injection of manipulated weighting). Adversarial tactics: embedding collision crafting, staged low-signal accumulation (slow poisoning), and manifest forgery attempts. Defenses: diversity in embedding architectures, anomaly detection on similarity score distributions, signed manifest verification, threshold guard-bands, and separation of duties (promotion approval distinct from ingestion) [7, 21, 41, 42, 50].

Evolution Guidance. Phase 1 (Monolithic): prioritize velocity and baseline metric instrumentation (precision, recall, contamination dwell). Phase 2 (Introduce Staging): add ephemeral unstable tier when poisoning alerts or drift exceed thresholds (e.g., precision drop >5

Key takeaway. Monolithic memory is simple but fragile (large poisoning blast radius); tiering, quorum promotion, and sharding raise integrity and rollback precision with latency/complexity overhead. Introduce governance as contamination dwell or drift metrics breach thresholds.

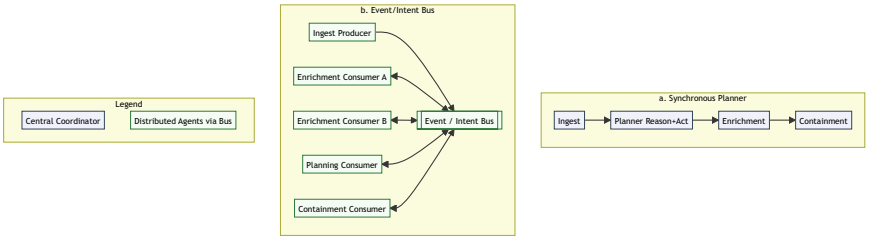


Fig. 1.6 Comparison of planner-centric and event/intent bus orchestration: planners yield total order and simple replay, while buses unlock parallelism but require stronger tracing and idempotent consumers

1.2.2.6 Orchestration Style (Synchronous Planner vs. Event/Intent Bus)

Dimension Framing. Orchestration style specifies how reasoning, enrichment, and action steps are ordered, scheduled, and causally linked: a *synchronous planner* centralizes sequential decision-making; an *event/intent bus* decouples phases via asynchronous topics (e.g., Apache Kafka or Apache Pulsar). The choice alters (i) determinism and forensic replay complexity, (ii) throughput ceiling and tail latency under surge, (iii) concurrency conflict surface, (iv) instrumentation burden for causal reconstruction, and (v) resilience to localized component failure. In short, the planner model is simpler and exports a total order that eases replay but can bottleneck under concurrency; the bus model improves parallel throughput but adds complexity around reordering and idempotent consumers [3, 4, 26–28] (Fig. 1.6).

Synchronous Planner Model. A single sequential reasoning+act loop imposes near-total order over agent/tool actions, maximizing determinism (linear log) and simplifying audit replay (low causal ambiguity). Strengths: centralized policy choke point (rate limits, guardrail evaluation), dense explainability (contiguous reasoning chain), simple saturation modeling, and reduced need for distributed idempotency logic. Limitations: throughput ceiling (queue depth growth under burst), head-of-line blocking (slow external call stalls unrelated incidents), vertical scaling pressure, and convex MTTR growth for lower-priority events once backlog forms [3, 5, 26].

Event/Intent Bus Model. Publish/subscribe topics (e.g., detection.intent, enrichment.result, remediation.request) decouple ingestion, enrichment, planning fragments, and containment execution. Strengths: horizontal elasticity (parallel consumers), burst absorption (topic-level buffering + backpressure), role specialization (selective subscription), partial fault isolation (failed consumer quarantined), and potential locality optimization (sharded topics). Risks: ordering reconstruction complexity (interleaved sequences), duplicate/out-of-order handling, expanded idempotency requirements, and higher instrumentation overhead (correlation IDs, span links, provenance hash chains). Representative platforms include Kafka and Pulsar [4, 25–28].

Hybrid/Phased Orchestration. Common evolutionary posture retains a synchronous governance nucleus (policy arbitration, privileged containment authoriza-

tion) while offloading high-churn enrichment or speculative reasoning to bus topics. This reduces policy race conditions while harvesting parallelism for lower-risk tasks. Migration granularity can proceed action-class by action-class (e.g., enrichment.* then low-privilege remediation) minimizing simultaneous complexity introduction [8, 26].

Latency and Tail Behavior. Synchronous style exhibits predictable per-step latency but tail amplification when incoming concurrency exceeds planner sustainable throughput (P95/P99 escalate sharply). Bus style introduces jitter (dispatch + queue variance) yet often compresses tail containment time once concurrency is high, because parallel consumers drain distinct incident streams. Evaluation should normalize by incident severity (weighted tail latency) and include scheduling fairness metrics (age distribution of pending high-severity items) rather than median alone [7, 26].

Qualitative performance bands (illustrative). **Planner:** low median decision latency while unsaturated; sharp tail cliff at overload. **Bus:** slightly higher median with queue-induced jitter; flatter P95/P99 under high concurrency via parallelism. **Hybrid:** planner-anchored governance retains low-latency privileged decisions; bus-backed enrichment smooths tails for bursty workloads.

Causality, Ordering & Provenance. Bus mode requires: (i) globally unique trace/correlation IDs, (ii) logical clock metadata (Lamport or vector timestamps) to partial-order events, (iii) replay buffers (time-sliced topic segments) for deterministic reconstruction, and (iv) materialized incident timelines (merge by correlation + timestamp tolerance). Planner mode inherently exports a total order, simplifying reconstruction. Provenance spans should encode orchestrationMode, parentCorrelationId, orderingClock, and decisionBundleHash to maintain parity of explainability across modes [25, 26, 38].

Concurrency Control and Idempotency. Bus mode leverages idempotent remediation tokens (first-accepted wins), optimistic concurrency for memory promotions (compare version hash), compensating events (rollback.*), and sometimes short-lived leases or consensus (k-of-n approval) for destructive actions. Planner naturally serializes conflicts but at throughput expense. Formal invariants (e.g., "no simultaneous conflicting state transitions for same host") shift from implicit (planner ordering) to explicit (token/lease validation) in bus architectures [8, 25, 38].

Ordering/idempotency at a glance. **Planner:** total order simplifies replay; idempotency centralized at the controller. **Bus:** partial order demands idempotent consumers, monotonic clocks, signed envelopes with priorCorrelationHash, and replay protection; spans include parentCorrelationId and orderingClock for deterministic reconstruction.

Security Posture/Attack Surface. Central planner concentrates privilege; compromise yields broad action forgery potential but is easier to instrument and attestate. Bus architecture distributes execution authority, shrinking blast radius per worker yet expanding surface (multiple consumers, topic ACLs, ordering metadata integrity). Additional risks: event injection, replay, or reordering attacks manipulating causal narratives. Mitigations: signed event envelopes (hash of payload + priorCorrelationHash), monotonic or bounded-gap logical clocks, consumer attes-

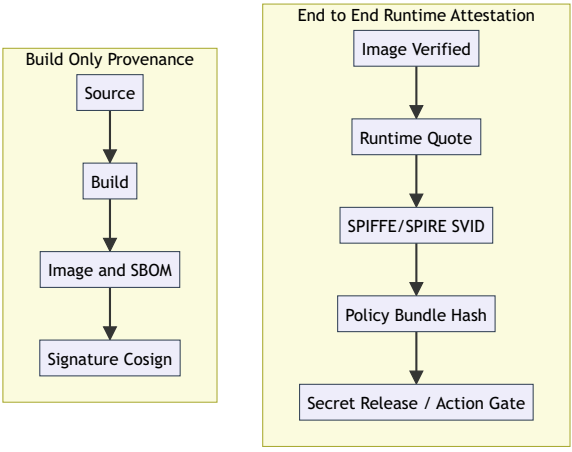


Fig. 1.7 Runtime attestation binds identity, measurement, and policy hash to cut tampering dwell time compared to build-only provenance, contrasting build-only provenance with end-to-end attested workloads (see Glossary for terminology)

tation (identity + policy hash), and anomaly detection on duplicate or retrograde timestamps [21, 25, 26].

Evolution Guidance. Phase 1: synchronous for clarity and baseline metrics. Phase 2: adopt bus for high-volume enrichment + analytics; retain central planner for privileged actions. Phase 3: graduate selective containment actions (low risk, stateless) after idempotency + ordering instrumentation meets error SLA (e.g., < 0.5)

Key takeaway. Planner-centric orchestration yields total order and simpler replay but caps throughput; event/intent buses unlock parallelism and tail resilience while demanding idempotency and stronger tracing. Hybridize: keep privileged decisions centralized, push enrichment to the bus.

1.2.2.7 Attestation Depth (Build Only Vs End-to-End)

Dimension Framing. Attestation depth captures how far assurance extends beyond build-time artifact lineage into runtime identity, configuration, and state verification. A *build-only* posture validates supply-chain provenance (source $\rightarrow$ build $\rightarrow$ image) but does not cryptographically assert that a deployed workload at time t is an untampered instance of that artifact with an approved policy bundle. *End-to-end* (E2E) attestation layers hardware-backed measurement, workload identity binding, policy bundle hashing, and (optionally) dynamic state assertions (memory promotion manifests) to harden trust decisions and shrink tampering dwell time [8, 21, 24] (Fig. 1.7).

1.2.2.8 Posture Profiles and Operational Signals

Build-Only Provenance. Relies on SLSA provenance, SBOM (SPDX), and signature (Cosign) verification pre-deployment. Strengths: low integration cost, negligible runtime overhead, fast iteration. Limitations: absence of runtime validation permits in-memory policy mutation, sidecar injection, binary drift, or host compromise to persist until behavioral anomalies surface (e.g., unexpected API calls, protection effectiveness drift). Trust decisions (inter-agent calls, secret release) rest on static assumptions rather than fresh attested state [21–23].

End-to-End Runtime Attestation. Extends build provenance with: (i) startup quotes (TEE or confidential VM measurement), (ii) workload identity (SPIFFE/SPIRE SVID) binding measurement to authenticated principal, (iii) signed policy bundle hash (OPA rego set + version) validation, (iv) periodic or event-triggered re-attestation (heartbeat quotes, configuration hash check), and (v) optional memory promotion manifest attestation (provenance of long-lived knowledge changes). Strengths: sharply reduces undetected tampering dwell window, enables zero-trust inter-agent verification (fresh measurement required per privileged interaction), and yields cryptographically anchored audit trails. Costs: first-call quote latency, modest verification overhead (cacheable), revocation + rotation complexity, and hardware constraints (TEE availability) [17, 21, 24, 25].

Mixed Posture and Zoning. Practical deployments stratify: Zone A (governance nuclei: policy signer, remediation planner, memory promotion service) under E2E; Zone B (enrichment, low-sensitivity analytics) under build-only; Zone C (experimental agents) with relaxed posture. Mixed zoning concentrates attestation overhead where risk (privilege, data sensitivity) is highest while preserving elasticity elsewhere. Triggers to elevate a component: handles regulated or high-secrecy data, issues external remediation actions, or mediates cross-zone data flows [7, 8, 21].

Operational Pattern. Pipeline: (1) verify image (Cosign + SBOM + SLSA provenance) before scheduling; (2) upon start, obtain attestation quote; (3) bind SVID to measurement; (4) validate signed policy bundle hash; (5) emit span with (policyHash, imageDigest, quoteDigest); (6) enforce secret release and inter-agent RPC only if measurement + policy hashes match expected allowlist; (7) renew quotes on interval or sensitive action boundary (containment, memory promotion). Enclave pooling + pre-attestation amortize cold start; warm verification dominated by cached signature checks [21, 23, 24, 26].

Telemetry and Divergence Detection. Runtime spans embed attributes: policyHash, modelDigest, configVersion, attestationFreshness, enclaveMeasurement, and expectedPolicyHashComparison (match|mismatch). Divergence (mismatch or stale quote beyond SLA) triggers quarantine: suspend privileged actions, rotate credentials, and initiate forensic snapshot. Build-only reliance on heuristic drift (task success variance, anomaly spikes) yields slower, noisier detection relative to cryptographic mismatch alerts [24–26].

Security Posture and Attack Surface. E2E reduces stealth tampering window but enlarges surface: quote verification service, key provisioning, trust anchor distribution, and revocation list management must themselves be hardened (signatures,

replication, audit logging). Threats: side-channel leakage within TEE, counterfeit quotes, stale SVID replay, selective policy hash substitution, or downgrade (forcing fallback to build-only). Mitigations: monotonic nonce or freshness token in quotes, binding quote to policy + config hash, pinning minimal trusted computing base (TCB), attested verifier redundancy, anomaly detection on quote failure patterns, and continuous integrity scanning of verification pipeline artifacts [17, 21].
Cost–benefit at a glance. **Build-only:** minimal runtime overhead and integration complexity; blind to runtime drift and in-memory tampering; detection relies on behavioral heuristics. **E2E attestation:** added first-call quote latency and verifier hardening burden; substantially tighter trust by binding identity, measurement, and policy hash; faster, cryptographic divergence signals enable automated quarantine.

Key takeaway. Build-only provenance is low-overhead but blind to runtime tampering; end-to-end attestation binds identity, measurement, and policy hash to tighten trust and shrink dwell time at modest verification cost. Apply deeper attestation where privilege and data sensitivity peak.

1.2.3 Threat–Control Summary (Preventive, Detective, Corrective)

The table below gives an *architecture-centric*, illustrative mapping from key threats to preventive, detective, and corrective controls; Sect. 1.4 contains the full threat taxonomy and mitigations and connects these safeguards to well-known governance frameworks [11, 12, 21, 24–26] (Table 1.2).

1.2.3.1 Real-World Examples

Narrative examples for these threats appear in Sect. 1.4.

1.3 Secure Retrieval-Augmented Generation Pattern

The secure retrieval-augmented generation (RAG) pattern (Fig. 1.8) is a reference design that synthesizes best practices from the chapter (policy gating, tiered memory, anomaly detection, provenance) into a consolidated flow. The high-level abstraction emphasizes five anchored concerns: (i) orchestrated hybrid retrieval (diversity for poisoning and injection resilience), (ii) governed knowledge store (tiering/provenance collapsed here for clarity), (iii) validation and guardrails (policy + anomaly + provenance checks), (iv) grounded reasoning (ensuring cited/attributed context), and (v) feedback into governance (drift, rollback readiness). Implementation variants typically expand the abstracted nodes into: staging/probationary/vetted tiers with signed

Table 1.2 Architecture-centric mapping of key threats to preventive, detective, and corrective controls; see Sect. 1.4 for narrative examples and detailed mitigations.

Threat	Preventive	Detective	Corrective
Prompt/retrieval injection	Guardrails (allow/deny, intent gating); schema/content checks; retrieval diversity + rank caps; least-privilege tools; templated outputs	Signed spans; I/O anomaly scores; override disagreement; retrieval-score drift	Suppress/quarantine; policy rollback; update blocklists; hotfix prompts/rankers; sandbox incident
Data/memory poisoning	Tiered store (ephemeral → probation → vetted); k-of-n consensus; signed promotion manifests; provenance checks	Consensus divergence; contamination heuristics; precision/recall deltas; lineage gaps	Manifested rollback (window revoke); purge tiers; pin/rollback retrievers; demote sources
Privilege escalation /tool abuse	Role-scoped IAM; intent tokens; OPA gate before privilege; rate limits; dual control	PolicyHash mismatch; unusual intent mix/rate; least-privilege failures; action anomalies	Rotate credentials; revoke sessions; route via planner; reversible ops; compensating playbooks
Supply chain tampering	SLSA provenance; SBOM (SPDX); Cosign verify; pinned digests; reproducible builds	Drift scans; dependency diffs; signature/key anomalies; verifier health	Rollback image; rotate/revoke keys; rebuild clean; quarantine pipeline stages
Event injection /ordering attacks	Signed envelopes; correlation IDs; bounded clocks; idempotent tokens; topic ACLs	Duplicate/out-of-order; sequence gaps; replay analytics; ordering audits	Quarantine replays; reconstruct timeline; compensating events; escalate to planner
Confidentiality breaches(in-use)	Enclaves/confidential VMs (governance); secrets bound to attested measurement + policyHash; minimize exposure	Quote freshness SLO; secret access locality; enclave memory pressure	Rotate secrets; reprovision enclave pool; safe degrade; revoke paths

promotion manifests; consensus fusion (k-of-n retriever consensus) across dense + sparse retrievers; anomaly + provenance gating prior to policy filtering; response manifest recording sources + policy hash; and rollback levers keyed by manifest lineage (bounded rollback window). This pattern reduces poisoning blast radius, supplies selective rollback, and supports attested provenance chaining while amortizing retrieval latency through orchestrated diversity. For the threat taxonomy and LLM-specific vulnerabilities that motivate these controls, see Sect. 1.4.

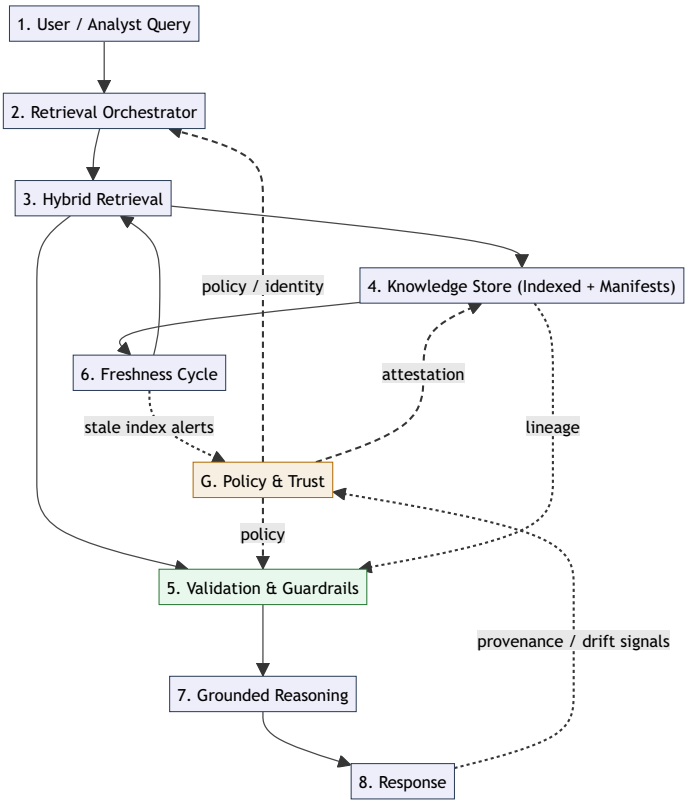


Fig. 1.8 Secure RAG flow (decluttered): *(1) User/Analyst Query* enters *(2) a Retrieval Orchestrator* invoking *(3) Hybrid Retrieval* against *(4) a governed Knowledge Store (Indexed + Manifests)*. Content is screened by *(5) Validation & Guardrails*; a *(6) Freshness Cycle* updates indices and can trigger re-retrieval; *(7) Grounded Reasoning* then emits an attributed *(8) Response*. Governance *(G; dashed)* is an overarching layer supplying policy, identity, and attestation (not a numbered step); dotted feedback edges return lineage, provenance/drift, and stale index alerts. Tiering, consensus fusion, anomaly gates, and rollback manifests are discussed in the text and Fig. 1.5

For readability, the flow can be viewed as five steps:

1. Query routing and hybrid retrieval (dense + sparse) with diversity.
2. Consensus/merge and poison/anomaly screening before policy filters.
3. Policy and content guardrails applied to curated context.
4. Grounded reasoning to produce an attributed response + provenance record.
5. Drift monitoring and governed feedback into tiered memory/promotion.

Freshness and consistency (operational concerns). Two common failure modes in production RAG are (i) indexing lag and (ii) retriever version skew. Indexing lag arises when new or updated sources have not yet propagated to the vetted index; retriever skew occurs when replicas use different model versions or settings, pro-

ducing divergent rankings. Mitigations include: (a) version-pinned retrievers with attested hashes (`retrieverSetVersion`) and rolling upgrades; (b) freshness guardrails in the orchestrator that require an index watermark (`indexTimestamp` $\geq$ freshness SLA) before accepting results, else trigger a policy-gated fallback (e.g., re-retrieval from probationary tier, route to human review, or defer action); (c) invalidation hooks from the ingestion pipeline to prompt re-retrieval when a relevant shard updates; and (d) telemetry that surfaces mismatches (e.g., retrieverSetVersion drift, stale index watermarks) to analysts. These controls reduce inconsistent answers across replicas and bound the window where stale context could guide actions.

In this context, grounded reasoning denotes constraining generation to verifiable, retrieved evidence, with explicit attributions back to trusted sources; it is a standard aim in RAG systems to mitigate hallucination and improve faithfulness [47–49, 51, 52].

See also Fig. 1.5 for the tiered memory governance that underpins this pattern. Production-scale retrieval-augmented systems such as RETRO demonstrate that large-scale retrieval is feasible in generalpurpose NLP questionanswering models, even though RETRO itself is not a cybersecurity system.[53] For terminology: BM25 denotes a standard sparse lexical retrieval function; see the Glossary (Sect. 1.5.3) for SVID and related identity/attestation terms.

Key design levers include: (a) promotion quorum thresholds (precision vs ingestion latency), (b) retriever heterogeneity (robustness vs latency budget), (c) guardrail layering order (early rejection vs maximal context preservation), and (d) rollback granularity (document, tier slice, or manifest window). Minimal viable deployments can begin with a monolithic vetted store plus lightweight guardrails and progressively introduce tiering and manifests when contamination dwell metrics or promotion error rates exceed thresholds.

1.4 Research Frontiers and Open Challenges

This section first summarizes LLM-specific vulnerabilities and alignment levers, then consolidates open questions and research directions informed by these risks.

1.4.1 AI Alignment and Safety in Agent Systems

This subsection consolidates the safety surface and mitigation levers for LLM-driven agent systems: it first enumerates LLM-specific vulnerabilities (what can go wrong) and then outlines alignment techniques (how to shape intrinsic behavior) that complement architectural guardrails. Labels for the prior standalone subsections are preserved below to avoid breaking cross-references.

1.4.1.1 LLM-Specific Vulnerabilities in Agentic Cybersecurity Systems

LLM-driven agents introduce failure modes that differ from traditional rules or classifiers and must be acknowledged explicitly in security architectures.

Hallucinated Recommendations. LLMs can produce confident but incorrect actions (e.g., proposing a non-existent firewall rule or irrelevant patch), which, when executed by an agent or followed by an analyst, degrade defense posture. Mitigations: require evidence-citing outputs for high-impact recommendations (grounded reasoning with attributions), attach retrieval/provenance links, and gate privileged actions behind verifiers or policy engines. Techniques like grounded generation and retrieval-linked attribution reduce reliance on unsupported tokens. Industry guidance (non–peer-reviewed) highlights operational risk when hallucinated outputs drive response actions, and academic evaluations further document the phenomenon and its safety implications[14–16, 25, 49, 52].

False Positives/Negatives. Free-form reasoning can mis-score benign activity as malicious (alert fatigue) or overlook stealthy threats (missed detection). Mitigations: ensemble or dual-run checks (LLM + deterministic rules), disagreement analytics (override disagreement metrics), and bounded adaptation with rollback when drift is detected in protection metrics. Continuous risk management guidance recommends explicitly monitoring decision quality and convergence latency for guardrail updates[6, 26].

Prompt Injection and Jailbreaks. Adversaries craft inputs that subvert system prompts, suppress safety policies, or exfiltrate secrets via tool calls. In multi-agent flows, an injected intermediate artifact can steer downstream agents. Mitigations: strict schema and content validation, least-privilege tool scopes, message signing and isolation across trust zones, and retrieval diversity to resist single-channel manipulation. The OWASP LLM Top 10 highlights prompt injection and related risks as first-class concerns for LLM applications[12, 24, 25, 41].

Data Poisoning and Model Evasion. Agents that learn from or depend on external knowledge are vulnerable to poisoned artifacts that bias retrieval and reasoning; adversaries can also craft inputs that consistently elicit unsafe or permissive behavior (evasion). Mitigations: tiered memory with quarantine and signed promotion manifests, k-of-n retriever consensus and heterogeneity, anomaly screens on similarity distributions, and manifested rollback to a last-clean window[21, 26, 42, 50].

These risks motivate the governance and provenance levers in this chapter: (i) end-to-end identity and attestation (to constrain privilege and bind decisions to measured state), (ii) policy engines and verifiers on high-impact actions, (iii) retrieval diversity and tiered promotion to bound poisoning blast radius, and (iv) structured tracing with signed spans and policy-hash convergence to support rapid rollback and audit [21, 24–26].

1.4.1.2 LLM Alignment Techniques for Secure Agent Behavior

Alignment methods complement external policy enforcement (e.g., OPA) by shaping the intrinsic response distribution of LLM components toward helpful, safe, and policy-conformant behavior.

Reinforcement Learning from Human Feedback (RLHF). Production-scale models frequently undergo post-training fine-tuning where human annotators (or reward models distilled from them) evaluate candidate completions for helpfulness and harmlessness. In a cybersecurity context, curated expert feedback can penalize hallucinated remediation steps (non-existent patches, unsafe firewall rules) and reward evidence-linked, least-privilege recommendations, biasing the model toward defensible outputs before architectural guardrails activate. RLHF can thus reduce raw hallucination rate and toxic or non-compliant suggestions upstream of policy checks [5, 6, 46].

Constitutional AI. Constitutional AI uses a written set of guiding principles ("constitution") to iteratively critique and refine model outputs, reducing dependence on continuous human labeling. A security-aligned constitution can encode invariants (never suggest disabling logging; never bypass segmentation; always justify privileged actions with cited evidence) so self-critiques nudge generations toward auditable, principle-bound recommendations. This creates an inner alignment layer complementing external policy engines, especially for low-risk, high-volume advisory responses [24, 25, 45].

Combined Use. RLHF establishes baseline refusal/helpfulness behavior; constitutional self-critique maintains adherence to explicit security principles under distribution shift. Integrated with architecture controls: (i) policy engines gate privileged actions even if alignment fails, (ii) provenance spans include alignmentVersion/constitutionHash for audit, and (iii) divergence telemetry (e.g., spike in constitution violation self-critiques) triggers rollback or retraining. Alignment does not replace guardrails: it shrinks risk surface (fewer unsafe proposals) and lowers reviewer workload, but cryptographic attestation, identity, and promotion manifests remain required for trust.

Key instrumentation additions: span attributes (alignmentStrategy=RLHF|Constitutional|Hybrid, alignmentVersion, constitutionHash, feedbackSampleDigest) enable measuring alignment drift and correlating misfires (unsafe suggestions) with specific alignment versions. Future work: adaptive constitutions that incorporate verified emerging threat patterns without enabling adversarial feedback poisoning.

1.4.2 Open Questions and Research Directions

Emergent open questions shaping medium-term *architecture* evolution (what structures to build, verify, and operate):

- **Invariant extraction and formal safety envelopes:** Architectures need *verifiable behavior contracts* (e.g., non-escalation, provenance continuity) derivable from traces and enforceable at runtime (policy engines or runtime monitors); recent agent benches surface gaps these invariants should close and show simple guardrails failing under adaptive threats [29, 54, 55].
- **Large-scale slow poisoning detection:** Retrieval/memory architectures must incorporate *poisoning-aware promotion gates* (signed manifests, shadow evaluation, rollback) to distinguish low-rate contamination from distribution drift without untenable false positives in multi-agent flows [41–43, 55].
- **Economics of provenance depth:** Layered attestation and manifests are architectural *knobs*; models are needed to tune depth (build, data lineage, policy bundles, runtime state) for optimal risk-reduction per dollar at system boundaries [6, 21–23, 56].
- **Adaptive guardrail stability:** Policy mode and control-loop design are architectural choices; *stability-bounded feedback loops* (shadow runs, guard-bands, canaries) should be wired into orchestration, with continuous monitoring as a first-class component. Evaluations show naive guardrails are brittle [6, 12, 25, 41, 55].
- **Measurement and reproducibility provenance:** Multi-agent topologies require *deterministic replayability*: standardized spans, environment capture, and cross-agent supply-chain manifests to reproduce incidents and verify policy/action graphs[21, 22, 26, 57].
- **Multi-agent collusion and covert channel resilience:** Coordination topologies and messaging layers should assume adversarial *agent–agent channels*; patterns for quorum/verification roles, signed envelopes, and anomaly telemetry are needed to detect covert cooperation [10, 29, 59].

Progress hinges on standardized span/manifest schemas (to avoid bespoke provenance dilution), benchmark corpora for poisoning and drift (early examples include Cybench and CyberGym), and lightweight formal methods (bounded model checks over policy/action graphs) *embedded into architecture pipelines* (CI/CD and runtime) rather than post-hoc audits. Early ingredients exist (supply-chain assurance, manifests, retrieval benchmarks), but they must be adapted to multi-agent settings and coupled with reliability practices at scale [21–23, 51, 52, 54, 57, 58].

1.5 Conclusion

Agentic cybersecurity architectures concentrate assurance leverage in structural choices: coordination topology, memory governance, isolation substrate, orchestration style, policy adaptivity, and attestation depth jointly define resilience envelopes. A pragmatic evolution path begins with centralized clarity, static guardrails, monolithic memory, and build-only provenance; then incrementally layers hybrid orchestration, tiered promotion with manifests, bounded adaptive guardrails, selective enclave placement, and end-to-end runtime attestation. High-fidelity observability

(signed spans, hash-chained manifests, policy bundle hashes) supports safe adaptation and rapid rollback. Continued research into invariant synthesis, poisoning detection, and collusion resilience remains important for scaling autonomous capability without eroding controllability.

These phases illustrate how an organization can incrementally adopt the safeguards discussed throughout the chapter. Each phase addresses a set of trade-offs: starting simple to maximize clarity and explainability, then adding targeted complexity to gain fault tolerance, blast radius containment, and stronger trust guarantees. This staged approach reflects the trade-offs analyzed here—balancing immediate practicality against long-term resilience—and should be adapted to local risk tolerance and operational maturity. Concretely, the progression improves resilience (fewer single points of failure), preserves auditability (signed provenance and replayable decisions), and maintains controllability (policy gating and rollback), with Phase 4's runtime attestation making blast radius containment explicit at runtime. Open questions (e.g., invariant synthesis automation and collusion resistance) remain active areas; see Sect. 1.4 for research frontiers.

1.5.1 Evaluation and Metrics Considerations (Future Work)

While this chapter is architecture-focused and does not present a full empirical bench, rigorous validation of an implemented system would track a set of core metrics. This section outlines them to make evaluation pathways explicit and to situate structural trade-offs (latency vs. assurance depth, parallelism vs. ordering fidelity) for follow-on studies:

- **Latency Overhead.** End-to-end response time from event ingestion to recommended (or executed) action, decomposed into orchestration, LLM reasoning, retrieval, policy evaluation, and attestation verification components. Architectural levers (central planner vs. event bus, build-only vs. runtime attestation) shift both mean and tail latency distributions. Instrumentation: signed spans with phase timestamps and policyHash/quoteDigest attributes. Goal: preserve near real-time containment SLOs while adding provenance depth [24–26].
- **Throughput and Concurrency.** Incidents or enrichment tasks processed per second under burst and sustained loads. Scaling hinges on agent granularity, shared vs. per-agent model instances, and bus back-pressure management. Measurements should report sustainable concurrency before policy convergence lag or ordering reconstruction error rates rise [4, 39].
- **Policy/State Convergence Time.** Time until all active agents agree on (and report) the same signed policy bundle hash and, where applicable, memory promotion manifest set. Relevant for adaptive guardrails: shorter convergence reduces inconsistent enforcement windows. Captured via periodic policyHash convergence metrics in tracing [25, 26].

- **Detection/Recommendation Effectiveness.** (Future simulation) True positive rate, false positive rate, and mitigation success (fraction of injected or simulated attacks for which recommended action prevents escalation) relative to baselines (rule-based SOAR, single-agent LLM). Even absent current benchmarks, architecture instrumentation enables later replay evaluation [6, 12, 41].
- **Provenance Continuity Rate.** Percentage of actions/remediation decisions for which complete causal chains (event → retrieval/context → reasoning spans → policy decision) are reconstructible without gaps. Directly affected by orchestration style and signed envelope adoption [24, 26].
- **Resource and Cost Efficiency.** GPU/CPU utilization per processed incident and per successful containment, plus memory footprint of multi-tier retrieval stores. Assesses marginal cost of assurance (e.g., runtime attestation, extra retrieval diversity) verses security benefit [21, 22].
- **Attestation Freshness/Drift Incidence.** Frequency of quote or measurement mismatches, average detection time, and quarantine latency. Benchmarks the practical overhead vs. tampering dwell time reduction when moving from build-only to end-to-end posture [17, 24].
- **Promotion Gate Precision.** (For tiered memory) Fraction of promoted artifacts later rolled back due to poisoning signals; measures effectiveness of k-of-n retriever consensus and manifest checks [42, 50].

Evaluation Scope Statement. Future empirical work should characterize trade-off curves (latency vs. provenance depth, convergence time vs. adaptation frequency) and publish standardized schemas (span fields, manifest formats) to foster reproducibility. In absence of current benchmark data here, we enumerate metrics to demonstrate a responsible path to validating architectural decisions without over-claiming performance. This aligns with recent multi-agent evaluation practice (e.g., interactive web task environments and behavioral simulations) emphasizing transparency of measurement axes over premature optimization [26, 29–31].

Key takeaway. Architecture choices embed latent evaluation axes; defining metrics early (even before experiments) prevents ad-hoc or biased measurement later and guides instrumentation (signed spans, policyHash convergence, promotion manifests) necessary for trustworthy benchmarking.

1.5.2 Illustrative Artifacts (Schemas)

The following compact, illustrative snippets show how governance and provenance artifacts can be represented in practice. They are not prescriptive; real deployments should adapt field names and signing formats to local platforms and standards (e.g., OPA bundles, SPIFFE/SPIRE SVIDs, SLSA/SPDX provenance). These examples are intentionally minimal and schematic; they are meant to illustrate how the architectural concepts in Sects. 2–3 can be instantiated.

Signed Span (envelope excerpt).

```
{
    "traceId": "4c6f...a91f", "spanId": "b12e...09c1", "parentId": "...",
    "service": "planner", "start": "2025-11-12T12:34:56Z", "end": "...",
    "identity": { "svid": "spiffe://org/env/agent/planner-01" },
    "policyHash": "sha256:9f3a...c2e1", "intent": "contain_host",
    "decision": { "allow": true, "rationale": "thresholds met" },
    "prevCorrelationHash": "sha256:5c11...7b9e",
    "signature": { "alg": "COSE_Sign1", "value": "BASE64..." }
}
```

Promotion Manifest (hash-chained).

```
{
    "version": 1,
    "prevHash": "sha256:ab12...f00d",
    "newHash":  "sha256:cd34...beef",
    "contentHash": "sha256:89aa...1122",
    "rationale": "k-of-n retrievers agree; filters clean; canary stable",
    "approver": "policy-signer@org",
    "approverSVID": "spiffe://org/env/governance/signer",
    "timestamp": "2025-11-12T12:45:00Z",
    "signatures": [ { "alg": "COSE_Sign1", "value": "BASE64..." } ]
}
```

OPA/Rego guardrail (exemplar).

```
package guardrails

default allow := false

allow {
    input.intent == "block_ip"
    input.confidence >= 0.9
    input.asset.criticality == "high"
    input.policyHash == data.bundle.hash
}
```

SLSA/Provenance (minimal sketch).

```
{
    "_type": "https://slsa.dev/provenance/v1",
    "subject": [{ "name": "policy-bundle.tgz",
        "digest": { "sha256": "9f3a...c2e1" } }],
    "builder": { "id": "https://ci.example.org/builders/policy" },
    "buildType": "policy-bundle-sign",
    "materials": [{ "uri": "git+https://example.org/policies#main",
        "digest": { "sha1": "1f2d..." } }]
}
```

1.5.3 Brief Incident Walkthrough

An endpoint anomaly triggers enrichment requests to OSINT, EDR, and inventory tools. A centralized planner synthesizes context and proposes containment, but execution is gated by policy: the request carries the current policyHash and is evaluated in OPA. Privileged actions run under least-privilege identities (SVIDs), and each step emits a signed span recording rationale, policyHash, and linkage (prevCorrelationHash) for replay. Concurrently, a candidate indicator enters the memory pipeline as *ephemeral*; it is promoted only after k-of-n retrievers agree, filters pass, and a promotion manifest is signed. If later drift or poisoning is detected, rollback targets the manifest lineage and demotes the affected entries without broad data loss. Throughout, runtime attestation (where enabled) provides fresh quotes for the governance nucleus, and divergence telemetry alerts if any cohort lags on policy bundle hash, supporting swift, auditable containment without over-claiming performance.

Glossary

The following terms are defined for clarity in this chapter. They are descriptive and not yet standardized; where applicable, alignment to related standards is noted in parentheses.

SPIFFE Secure Production Identity Framework for Everyone; open standard for workload identity in dynamic, zero-trust environments (X.509/JWT SVID issuance).

SVID SPIFFE Verifiable Identity Document; short-lived cryptographic credential asserting a SPIFFE ID for mutual workload authentication.

SLSA Supply-chain Levels for Software Artifacts; progressive integrity framework reducing tampering risk via verifiable, auditable build provenance levels.

OPA Open Policy Agent; policy-as-code engine for context-aware, decoupled enforcement across services, CI/CD, data, and infra.

Attestation Signed statement about build or runtime properties (e.g., materials, measurements, policy decisions) enabling automated integrity and trust verification.

Attestation freshness Operational bound on the maximum allowed age of the evidence (e.g., TEE quote, measurement digest) used to authorize actions; exceeding the bound should trigger re-quote or quarantine. (Related: Quote freshness SLO)

Provenance Verifiable lineage metadata (sources, dependencies, build system) supporting audit, rollback, and tamper detection via SBOMs and signed provenance (e.g., SPDX, SLSA docs).

Governance anchor Hardened central service that signs policies and authorizes privileged actions; sub-orchestrators handle enrichment. (Related: policy authority, PKI root.)

Governance nucleus High-trust core (Zone A) hosting policy signer, remediation planner, and secret broker, typically under enclave/confidential VM.

Blast radius The scope of impact when a component or cohort is compromised or misbehaves; architectural isolation and privilege separation aim to shrink this scope and localize rollback.

Explainability density Ratio of interpretable causal/context tokens to total trace payload, proxying analyst effort per unit telemetry.

Auditability density Concentration of actionable provenance and causal information per unit of trace/log payload; higher density generally reduces analyst effort during investigations.

Causal stitching Reconstruction of ordered multi-agent causal chains from distributed traces and span links.

Policy hash convergence Time/variance until active agents report an identical signed policy bundle hash; detects drift/divergence.

Policy convergence latency Time for a cohort of agents to converge on the same active policy version (bundle hash and gating rules) after an update; shorter is better to minimize inconsistent enforcement windows. (Related: Policy hash convergence)

Divergence telemetry Alerts/metrics indicating policyHash mismatches or provenance inconsistencies across cohorts.

Attestation depth Assurance scope from build-only provenance (SLSA/SPDX/Cosign) to end-to-end runtime (quote + SVID + policy hash gating).

Quote freshness SLO Target bounds on recency of attestation quotes to limit trust staleness.

Tail latency High-percentile latency (e.g., P95/P99) reflecting worst-case response times under load; architectures trade mean latency for improved tail behavior via parallelism and back-pressure.

Overload degradation slope Rate at which latency/throughput degrade beyond saturation for a coordination style.

Decision rationale volatility Stability indicator for explanations across policy updates (lower is better).

Idempotent tokens Unique, replay-safe action identifiers (or monotonic sequence guards) that allow clients to retry without duplicating side effects; servers reject duplicates beyond an allowed window.

Tiered memory governance Ephemeral → Probationary → Vetted knowledge pipeline to localize poisoning and enable rollback.

k-of-n retriever consensus Quorum across heterogeneous retrievers required before promotion to trusted tiers.

Signed promotion manifests Hash-chained records (prevHash → newHash) of content elevation plus rationale and approver identity.

Rollback levers/window Targeted demotion mechanisms keyed by manifest lineage and bounded revocation windows.

Provenance continuity Unbroken, verifiable lineage across build, deploy, retrieval, and runtime decisions.

Provenance continuity rate Percentage of actions/decisions for which an end-to-end causal chain (event → retrieval/context → reasoning spans → policy decision) is reconstructible without gaps.

Span links Tracing primitives referencing related spans across concurrent flows, enabling timeline reconstruction and causal stitching.

Resources

Open-source code repository (permanent link): https://github.com/mahmoudrabie/research-sota-arch-cyber-agentic-systems

Acknowledgements I thank Prof. Anis Koubaa for his great suggestions that improved this chapter. I also acknowledge OpenAI's GPT-5 for assisting in crafting this research—including drafting, refactoring, and copy-editing portions of the text and figure captions; I reviewed and am responsible for all final content.

References

1. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G,. Askell, A. et al.: Language Models are Few-Shot Learners. NeurIPS 33 (2020). arXiv:2005.14165
2. Schick, T. et al.: Toolformer: Language Models Can Teach Themselves to Use Tools. NeurIPS 36 (2023). arXiv:2302.04761
3. Yao, S. et al.: ReAct: Synergizing Reasoning and Acting in Language Models. ICLR (2023). arXiv:2210.03629, https://arxiv.org/abs/2210.03629
4. Shen, Y. et al.: HuggingGPT: Solving AI Tasks with ChatGPT and its Friends in HuggingFace (2023). arXiv:2303.17580, https://arxiv.org/abs/2303.17580
5. Amodei, D. et al.: Concrete Problems in AI Safety (2016). arXiv:1606.06565. https://arxiv.org/abs/1606.06565
6. NIST AI Risk Management Framework 1.0 (2023). https://www.nist.gov/itl/ai-risk-management-framework. Accessed 2025-09-05
7. NIST: SP 800-53 Rev.5 Security and Privacy Controls (2020). https://csrc.nist.gov/publications/detail/sp/800-53/rev-5/final. Accessed 2025-09-05
8. NIST: SP 800-160 Vol.1 Rev.1 Systems Security Engineering (2022). https://doi.org/10.6028/NIST.SP.800-160v1r1. https://doi.org/10.6028/NIST.SP.800-160v1r1 (accessed 2025-09-05)
9. NIST: SP 800-137 Information Security Continuous Monitoring (ISCM) for Federal Information Systems and Organizations (2011). https://csrc.nist.gov/publications/detail/sp/800-137/final. Accessed 2025-09-12
10. MITRE: MITRE ATT&CK Framework (2024). https://attack.mitre.org/. Accessed 2025-09-05
11. NIST: The NIST Cybersecurity Framework (CSF) 2.0 (2024). https://www.nist.gov/cyberframework. Accessed 2025-09-12
12. OWASP: Top 10 for LLM Applications (Project Page). https://owasp.org/www-project-top-10-for-large-language-model-applications/. Accessed 2025-09-05
13. Tramèr, F. et al.: Stealing Machine Learning Models via Prediction APIs. USENIX Security 2016 (2016). https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer

14. IBM Security Intelligence: AI Hallucinations Can Pose a Risk to Your Cybersecurity. Blog article (2023). https://securityintelligence.com/posts/ai-hallucinations-cybersecurity-risk/. Accessed 2025-11-13
15. Lin, S., Hilton, J., Evans, O.: TruthfulQA: Measuring How Models Mimic Human Falsehoods. ACL 2022 (Long Papers) (2022). https://doi.org/10.18653/v1/2022.acl-long.229
16. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. ACM Comput. Surv. **55**(12), 1–38 (2023). https://doi.org/10.1145/3571730.Available at: https://arxiv.org/pdf/2202.03629.pdf
17. Costan, V., Devadas, S.: Intel SGX Explained. IACR ePrint 2016/086 (2016). https://eprint.iacr.org/2016/086
18. Xu, Y., Cui, W., Peinado, M.: Controlled-channel attacks: deterministic side channels for enclaved programs. IEEE Symposium on Security and Privacy (S&P) (2015). https://ieeexplore.ieee.org/ielx7/7160813/7163005/07163052.pdf. Accessed 2025-09-12
19. Van Bulck, J., Minkin, M., Weisse, O., Genkin, D., Kasikci, B., Piessens, F., Silberstein, M., Wenisch, T.F., Yarom, Y., Strackx, R.: Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. USENIX Security 2018 (2018). https://foreshadowattack.eu/. Accessed 2025-09-12
20. Chen, S. et al.: SgxPectre Attacks: Leaking Enclave Secrets via Speculative Execution (2019). arXiv:1802.09085. Accessed 2025-09-12
21. SLSA: Supply-chain Levels for Software Artifacts v1.0 (2024). https://slsa.dev/. Accessed 2025-09-05
22. SPDX: Software Package Data Exchange (Specification site) (2024). https://spdx.dev/. Accessed 2025-09-05
23. Sigstore: Cosign Project (2024). https://github.com/sigstore/cosign. Accessed 2025-09-05
24. SPIFFE: SPIFFE/SPIRE Specifications (2024). https://spiffe.io/. Accessed 2025-09-05
25. OPA: Open Policy Agent Docs (2024). https://www.openpolicyagent.org/. Accessed 2025-09-05
26. OpenTelemetry: OpenTelemetry Project (2024). https://opentelemetry.io/. Accessed 2025-09-05
27. Apache Kafka: Apache Kafka Project (2025). https://kafka.apache.org/. Accessed 2025-09-12
28. Apache Pulsar: Apache Pulsar Project (2025). https://pulsar.apache.org/. Accessed 2025-09-12
29. Li, J. et al.: AgentBench: Evaluating LLMs as Agents. ICLR 2024 (2024). arXiv:2308.03688. https://arxiv.org/abs/2308.03688
30. Park, J.S., O'Brien, J., Cai, C.J., Morris, M.R., Liang, P., Bernstein, M.S.: Generative Agents: Interactive Simulacra of Human Behavior. CHI '23 (2023). https://doi.org/10.1145/3586183.3606763. https://arxiv.org/pdf/2304.03442.pdf
31. Zhou, S., Xu, F.F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Ou, T., Bisk, Y., Fried, D., Alon, U., Neubig, G.: WebArena: A Realistic Web Environment for Building Autonomous Agents (2023). arXiv:2307.13854
32. Shodan: Shodan Search Engine (2024). https://www.shodan.io/. Accessed 2025-09-05
33. MITRE: Common Vulnerabilities and Exposures (2024). https://www.cve.org/. Accessed 2025-09-05
34. NIST: National Vulnerability Database (2024). https://nvd.nist.gov/. Accessed 2025-09-05
35. CyberPress: Report on a GPT-4-powered honeypot ("Beelzebub") simulating SSH responses (CyberPress report; nonpeer, anecdotal) (2025). https://cyberpress.org/. Accessed 2025-11-13
36. Newsham, L., Hyland, R., Prince, D.: Inducing Personality in LLM-Based Honeypot Agents: Measuring the Effect on Human-Like Agenda Generation (2025). arXiv:2503.19752 [in press; preprint]. https://arxiv.org/abs/2503.19752. Accessed 2025-11-13
37. Rao, A.S., Georgeff, M.P.: BDI agents: from theory to practice. In: Proceedings of First International Conference on Multi-Agent Systems (ICMAS-95) (1995). https://dblp.org/rec/conf/icmas/RaoG95. Accessed 2025-09-06
38. Ongaro, D., Ousterhout, J.: In Search of an Understandable Consensus Algorithm (Raft). USENIX ATC 2014 (2014). https://raft.github.io/raft.pdf

39. Chase, H.: LangChain Framework (2022). GitHub repository. https://github.com/langchain-ai/langchain
40. Torantulino, T. et al.: Auto-GPT: An Autonomous GPT Experiment. GitHub repository (2023). https://github.com/Significant-Gravitas/Auto-GPT
41. Greshake, K. et al.: More than you've asked for: A Comprehensive Analysis of Prompt Injection Attacks (2023). arXiv:2302.12173
42. Goldblum, M., Tsipras, D., Xie, C., Chen, X., Schwarzschild, A., Song, D., Mądry, A., Li, B., Goldstein, T.: Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses. IEEE TPAMI (2022). https://doi.org/10.1109/TPAMI.2022.3162397
43. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Comput. Surv. **46**(4) (2014). https://doi.org/10.1145/2523813.http://eprints.bournemouth.ac.uk/22491/1/ACM20computing20surveys.pdf
44. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. MIT Press (2018). http://incompleteideas.net/book/RLbook2020.pdf
45. Bai, Y. et al.: Constitutional AI: Harmlessness from AI Feedback (2022). arXiv:2212.08073
46. Ouyang, L. et al.: Training language models to follow instructions with human feedback (2022). arXiv:2203.02155
47. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuksa, P., Min, S., Lewis, M., Yih, W-t., Rockt"aschel, T., Riedel, S., Kiela, D.: Retrieval-Augmented Generation for Knowledge-Intensive NLP. NeurIPS 33 (2020). arXiv:2005.11401
48. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W-t.: Dense Passage Retrieval for Open-Domain Question Answering. EMNLP 2020 (2020). arXiv:2004.04906
49. Asai, A., Min, S., Shi, W., Lin, X.V., Shi, P., Yih, W-t., Hajishirzi, H.: Self-RAG: Learning to Retrieve, Generate, and Critique for Improved Language Modeling (2023). arXiv:2310.11511
50. Zhong, Z., Min, S., Chen, D., Zettlemoyer, L., Yih, W-t.: Adversarially Robust Dense Retrieval via Adversarial Passages. EMNLP 2023 Findings (2023). arXiv:2310.19156. Accessed 2025-09-13
51. Thakur, N., Reimers, N., Rücklë, A., Srivastava, A., Gurevych, I.: BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models. NeurIPS 2021 Datasets and Benchmarks Track (2021). arXiv:2104.08663
52. Petroni, F., Piktus, A., Lewis, P., Yih, W-t., Rocktäschel, T., Riedel, S., Karpukhin, V. et al.: KILT: a Benchmark for Knowledge Intensive Language Tasks (2021). NAACL 2021. arXiv:2009.02252
53. Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., Millican, K., Van Den Driessche, G. et al.: Improving language models by retrieving from trillions of tokens. Nature **603**, 589–593 (2022). https://doi.org/10.1038/s41586-022-04572-4.https://arxiv.org/pdf/2112.04426.pdf
54. Zhang, A.K., Perry, N., Dulepet, R., Ji, J., Menders, C., Lin, J.W., Jones, E., Hussein, G., Liu, S., Jasper, D., Peetathawatchai, P., Glenn, A., Yiorkadjis, P., Osele, K., Raghupathi, G., Boneh, D., Ho, D.E., Liang, P.: Cybench: A Framework for Evaluating Cybersecurity Capabilities and Risks of Language Models (2024). arXiv:2408.08926
55. Boisvert, L., Bansal, M., Evuru, C.K.R., Huang, G., Puri, A., Bose, A., Fazel, M., Cappart, Q., Stanley, J., Lacoste, A., Drouin, A., Dvijotham, K.: DoomArena: A framework for Testing AI Agents Against Evolving Security Threats (2025). arXiv:2504.14064
56. Zhang, A.K., Ji, J., Menders, C., Dulepet, R., Qin, T., Wang, R.Y., Wu, J., Liao, K., Li, J., Hu, J., Hong, S., Demilew, N., Murgai, S., Tran, J., Kacheria, N., Ho, E., Liu, D., McLane, L., Bruvik, O., Han, D.R., Kim, S., Vyas, A., Chen, C., Li, R., Xu, W., Ye, J.Z., Choudhary, P., Bhatia, S.M., Sivashankar, V., Bao, Y., Song, D., Boneh, D., Ho, D.E., Liang, P.: BountyBench: Dollar Impact of AI Agent Attackers and Defenders on Real-World Cybersecurity Systems (2025). arXiv:2505.15216
57. Wang, Z., Shi, T., He, J., Cai, M., Zhang, J., Song, D.: CyberGym: Evaluating AI Agents' Cybersecurity Capabilities with Real-World Vulnerabilities at Scale (2025). arXiv:2506.02548

58. Lee, H., Zhang, Z., Lu, H., Zhang, L.: SEC-bench: Automated Benchmarking of LLM Agents on Real-World Software Security Tasks (2025). arXiv:2506.11791
59. Motwani, S. et al.: Secret Collusion among AI Agents: Multi-Agent Deception via Steganography (2024). arXiv:2402.07510. Accessed 2025-09-12

Mahmoud Rabie is a Principal Solutions Consultant at Cyberani by Aramco Digital (Riyadh, Saudi Arabia), where he architects AI-powered cybersecurity products and reference architectures, leading structured technology evaluations and secure-by-design implementations aligned to frameworks such as NIST CSF and SAMA CSF. He has broad experience in multicloud and hybrid environments (including OCI, Google Cloud, and AWS), DevSecOps/GitOps, and cloud security architecture. Previously, he held senior leadership roles across cloud architecture and digital transformation in fintech, banking, and public-sector organizations in Saudi Arabia. He received an M.Sc. in Computer Engineering from Cairo University (Egypt), and his interests include secure agentic systems, provenance and attestation, and resilient cloud-native security architectures.

LLM-GAN: Bridging Language and Multi-modal Generation in the Next AI Healthcare Frontier



Waad Bouzidi, Zouhour Neji Ben Salem, and Olfa Meddeb

Abstract The chapter presents a unified framework that integrates Large Language Models (LLMs) and Generative Adversarial Networks (GANs) in multimodal healthcare AI. A comprehensive exploration of LLM-GAN synergy is provided to bridge reasoning and generative synthesis. Six synergistic integration architectures are explored: Direct Adversarial, Bidirectional Feedback, Hierarchical, Cross-Modal Attention, Chain-of-Thought (CoT)-conditioned, and Hybrid frameworks. Each of them is designed to enhance multimodal data augmentation, generation, semantic alignment, and clinical relevance. Beyond methodological contributions, these architectures are directly motivated by real-world healthcare use cases. Empirical validation on the DEAP dataset for data augmentation task for stress classification demonstrates superior performance of the synergy regarding baselines modals. The hybrid architecture emerges as the leading model in this comparison, demonstrating superior performance (95.3% accuracy) and synthetic data fidelity. The chapter also analyzes trade-offs between synergy and unified models, highlighting modularity, explainability, and privacy-preserving augmentation as key advantages. In addition, the chapter examines the broader technical, ethical, clinical consideration, and societal challenges associated with deploying such synergistic systems in real-world healthcare contexts. Issues including computational cost, bias amplification, interpretability constraints, and risks of misinformation are critically analyzed. Finally, future directions are outlined.

W. Bouzidi (✉) · O. Meddeb
SMART Lab, High Institute of Management, University of Tunis, Carthage University, Nabeul, Tunisia
e-mail: waad.bouzidi@fsegn.ucar.tn

O. Meddeb
e-mail: ofla.meddeb@isg.rnu.tn

W. Bouzidi · Z. N. B. Salem · O. Meddeb
Faculty of Economic Sciences and Management, Carthage University, Nabeul, Tunisia
e-mail: zouhour.nejibensalem@fsegn.ucar.tn

Z. N. B. Salem
Larodec Lab, High Institute of Management, University of Tunis, Tunis, Tunisia

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

A. Koubaa et al. (eds.), *Generative AI and Large Language Models: Opportunities, Challenges, and Applications*, Studies in Computational Intelligence 1287,
https://doi.org/10.1007/978-3-032-24013-2_2

Keywords LLM-GAN synergy · Multimodal healthcare AI · Synthetic data augmentation · Hybrid architectures · Ethical AI · Privacy-preserving augmentation

1 Introduction

The healthcare industry stands at a transformative inflection point, where Artificial Intelligence (AI) is rapidly evolving from experimental research tools to mission-critical clinical infrastructure [76, 102]. Over the past decade, AI applications in healthcare have achieved remarkable advances across multiple domains [75]. Diagnostic imaging systems now approach the performance of radiologists in detecting pathologies [33, 80, 83], Natural Language Processing (NLP models extract actionable insights from vast repositories of clinical notes [15, 90], and predictive algorithms identify at-risk patients before symptoms appear [14, 53, 92]. Collectively, these capabilities are redefining how healthcare data is processed, analyzed, and translated into meaningful clinical actions.

To illustrate the transformative potential of next-generation AI systems in healthcare, consider the following scenario set in 2035: Dr. X, a specialist in virtual oncology, receives the examination results of Y, a patient diagnosed with a rare and progressive cancer, directly on her integrated medical interface. The data, including biopsies, genetic sequences, and 4D imaging, are instantly analyzed by an advanced AI system, which combines an LLM, specialized not only in medical knowledge but also in Y's emotional and psychological profile. It crafts a personalized script for delivering the diagnosis with empathy and clarity. Meanwhile, GANs power an augmented reality simulation of Y's facial expressions and reactions in real time, allowing Dr. X to tailor her communication before meeting Y face-to-face. Supported by live AI assistants, the doctor navigates this delicate moment with heightened sensitivity, ensuring Y feels supported and understood throughout.

This glimpse into the near future exemplifies the transformative potential of merging LLMs and GANs in healthcare. While this scenario may seem like science fiction, both foundational technologies are rapidly advancing, and early implementations are already emerging in clinical settings worldwide[118, 140]. The convergence of these two powerful AI paradigms represents more than just technological advancement, it signifies a fundamental shift toward personalized, empathetic, and highly precise medical care.

However, the true promise of AI in medicine extends beyond the achievements of isolated models. It lies in the emergence of multimodal intelligent architectures capable of integrating and reasoning across heterogeneous data types [103, 119]. The term *multimodal* refers not simply to models capable of handling multiple input formats, but to systems that respect the native form of data before preprocessing, mirroring the way clinicians synthesize diverse inputs such as imaging [48], laboratory tests [87], genomic sequences [19], physiological signals [109], and patient narratives [31, 120]. Each modality provides a distinct perspective on the

patient's condition, and traditional single-modality systems often fail to capture the interdependencies among them [91]. By contrast, multimodal architectures enable holistic, context-aware reasoning that approximates the integrative judgment of expert clinicians [119].

LLM models have demonstrated remarkable capabilities in processing and generating human-like text, making them invaluable for tasks ranging from medical documentation [137] and patient communication [17] to clinical decision support [40] and medical education [139]. Their ability to understand context, synthesize vast amounts of medical literature, and communicate complex information in accessible language has positioned them as critical tools for enhancing healthcare delivery [67]. GANs have transformed synthetic medical data generation, producing high-fidelity outputs from realistic radiological images and physiological waveforms to privacy-preserving datasets while maintaining biological plausibility [37, 110]. GANs have demonstrated large capacity in medical imaging [113], drug discovery [1], and patient simulation [124] while addressing critical challenges related to data privacy and scarcity [42]. Compared to diffusion models, which excel at high-fidelity image synthesis but require computationally expensive iterative denoising, GANs offer faster single-step generation suitable for real-time clinical applications and large-scale data augmentation. The chapter therefore focuses on scenarios where LLM-GAN synergy is most advantageous: controllable synthetic data generation for rare diseases, cross-modal translation with semantic grounding (e.g., generating X-rays from radiology reports), and explainable multimodal decision support where reasoning and generation must remain auditable.

Although these models represent major advances, LLMs and GANs have largely evolved in parallel, each with critical shortcomings: LLMs lack the capacity to generate clinically precise visual or physiological outputs [129], while GANs often produce results without semantic grounding or diagnostic relevance [23]. This parallel development highlights a fundamental disconnect between language-driven reasoning and data-driven synthesis, underscoring the need to integrate LLM and GAN technologies. When combined, they create multimodal AI systems capable of understanding, processing, and generating both textual and visual medical information with unprecedented sophistication, bridging the gap between reasoning and clinically meaningful data synthesis.

Despite the growing interest in multimodal AI for healthcare, critical questions remain unanswered regarding the effective integration of LLMs and GANs. To address these gaps, this chapter is guided by four central research questions:

1. How can LLMs and GANs be architecturally integrated to enable context-aware multimodal AI systems in the healthcare domain?
2. What is the most effective architecture for bridging linguistic reasoning and multimodal synthesis?
3. How can such synergies be evaluated for clinical validity, fidelity, safety, and regulatory compliance?
4. What are the ethical considerations and computational trade-offs of deploying LLM-GAN systems in real-world clinical workflows?

These questions frame our investigation into LLM-GAN synergy, positioning this work as both a conceptual exploration and a methodological contribution to the emerging field of generative multimodal AI in healthcare.

Importantly, while LLM-GAN architectures share the goal of multimodal synthesis with unified models such as Med-PaLM M [132], Flamingo, CLIP, and with diffusion-based approaches, they offer distinct advantages in specific contexts. Unlike end-to-end unified models, LLM-GAN systems enable modular interpretability, where the linguistic reasoning (LLM) and perceptual generation (GAN) components can be independently debugged, updated, and validated.

Building on these questions, the chapter aims to examine the synergy of LLMs and GANs, highlighting how each model's shortcomings can be addressed by the other. By exploring strategies for their integration, demonstrating how combining LLMs and GANs not only resolves their individual weaknesses but also leverages their complementary strengths, enabling the development of multimodal AI systems that unify reasoning and data synthesis in clinically meaningful ways. At the same time, this synergy introduces new challenges, including increased model complexity, higher computational costs, and the risk of propagating errors across modalities, which must be carefully managed. Additionally, ethical considerations such as patient privacy, algorithmic bias, informed consent for AI-assisted care, and the preservation of human agency in medical decision-making are critical to ensuring the responsible deployment of such systems.

It is important to acknowledge that LLM-GAN integration in clinical practice remains an active area of research rather than a mature, widely deployed solution [104]. The chapter serves as both a conceptual roadmap and a methodological contribution, outlining architectural paradigms, evaluation protocols, and deployment considerations for future research. To ground our discussion empirically, a quantitative evaluation on the DEAP dataset is presented, demonstrating how different LLM-GAN architectures perform on data augmentation for stress detection.

The chapter explores the transformative intersection of LLMs and GANs within AI healthcare. Sections 2 and 3 examine their foundational technologies and their unique contributions in that field as well as an empirical comparison between GAN and diffusion to empower the choice of GAN. Section 4 is devoted to describing the multimodal healthcare system comparing it with the unimodal. Section 5 describes how LLMs and GANs can be synergistically combined into multimodal AI healthcare systems. It also analyzes design strategies and their paradigms for context-aware generative tasks, key healthcare applications as well as a practical guidance for selecting an appropriate approach: when to leverage LLM–GAN synergy and when a unified modal design may be more efficient or clinically suitable. Section 6 presents an empirical validation of the LLM–GAN synergy, demonstrating its performance, robustness, and clinical relevance across real world application with DEAP dataset. The technical, ethical, and regulatory challenges of deploying these systems responsibly will be explained in Sect. 7. Finally, Sect. 8 discusses future directions, envisioning intelligent AI agents capable of advanced reasoning and innovation in medical care and research.

2 LLMs: Architectural Foundations and Healthcare Transformation

LLMs, which combine significant architectural advancements with game-changing applications in vital industries like healthcare, constitute a technical revolution in the field of AI. LLMs have developed into adaptable systems that can comprehend, produce, and reason about textual material at previously unheard-of size and accuracy since the advent of pre-trained language models (W. X. [157]). In this section, we outline the basic design and working principles of LLMs, emphasizing their essential features and examining their main uses in the healthcare industry. Crucially, we highlight architectural elements that explicitly support the LLM-GAN integration paradigms described in Sect. 5.

2.1 LLM's Basic Core Architecture

LLMs are a cutting-edge type of Deep Neural Networks (DNN) created especially to analyze, comprehend, and produce human language at previously unheard-of levels of intricacy and scale. They are mostly based on Transformer topologies, which have transformed NLP by allowing for extensive training and strong language production and interpretation capabilities [136].

2.1.1 Transformer

Long-range dependencies in text are efficiently captured by attention methods, which are the foundation of the Transformer fundamental design. The Transformer is made up of a number of interconnected parts, each of which helps it represent sequences well, as seen in Fig. 1. Prior to discussing attention mechanisms, it is critical to comprehend the following ideas [136]:

- **Token**: LLMs process text by dividing it into tokens, which can be words, subwords, or characters. Tokens are represented as vectors encoding their semantic and contextual information.
- **Embedding**: Tokens are mapped into high-dimensional vector spaces, allowing the model to capture semantic and syntactic relationships. This step transforms discrete tokens into continuous representations.
- **Encoder**: The encoder takes an input sequence and transforms it into **contextual embeddings** that capture the meaning and relationships between tokens. Each encoder layer includes multi-head self-attention, normalization, and feed-forward networks (left part of Fig. 1).
- **Decoder**: The decoder generates output sequences, such as translations or text predictions, based on the encoder's representations. It works autoregressively,

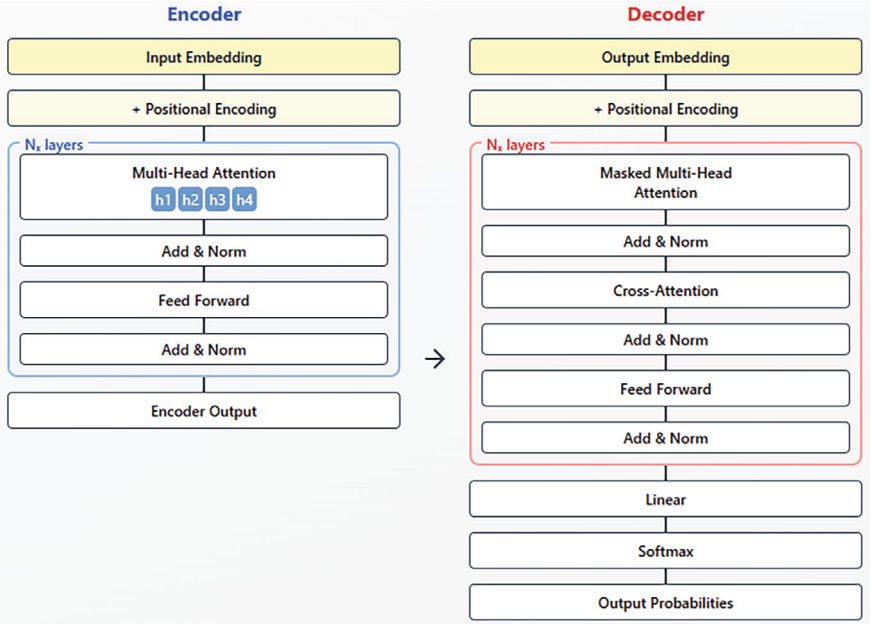


Fig. 1 Transformer general core architecture and attention mechanism

predicting the next token conditioned on previously generated tokens. The decoder also incorporates masked self-attention to prevent attending to future tokens and cross-attention to integrate encoder outputs (right part of Fig. 1).

Attention is a core deep-learning mechanism that enables models to dynamically focus on the most relevant input elements [25] instead of treating all inputs uniformly. It computes a context-aware weighted aggregation of information using **queries (Q)**, **keys (K)**, and **values (V)**:

$$\text{Attention}(\text{Q}, \text{K}, \text{V}) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1}$$

The scaling factor $\sqrt{d_k}$ stabilizes gradients, while softmax converts similarity scores into normalized weights used to combine values. Key attention variants include:

- **Self-attention**: each element attends to all elements in the same sequence, capturing global dependencies.
- **Causal (masked) attention**: a restricted self-attention used in autoregressive models, applying an upper-triangular mask to prevent access to future tokens.

- **Multi-head attention**: runs multiple attention heads in parallel on projected subspaces, then concatenates and linearly transforms them to learn diverse relational patterns.
- **Cross-attention**: queries originate from one sequence (e.g., decoder), while keys and values come from another (e.g., encoder), enabling information transfer across modalities or sequences.

Together, these mechanisms form the backbone of modern sequence and multi-modal architectures by enabling flexible, context-dependent information routing.

2.1.2 LLM functioning

LLMs use a series of well defined steps to convert unstructured material into responses that are consistent and relevant (Fig. 1). To reflect the complexity of human language, these steps integrate vector-based representations, DNN structures, and statistical learning. Knowing this procedure makes it easier to see how LLMs can produce fluid, contextually aware outputs for a variety of tasks, including discourse, summarization, and question answering (QA).

LLMs are trained on extensive and varied textual datasets collected from sources such as books, articles, websites, and other written materials. Below, we outline the sequential steps that LLMs follow during text processing and generation [136]:

1. **Tokenization and embedding**: LLMs mostly utilize subwords tokenization to balance vocabulary size and semantic meaning. Each token is converted into an embedding which represents its semantic content [149].
2. **Positional Encoding**: To maintain the order of the sequence, positional encodings are added to token embeddings, helping the model understand the context based on token positions. Final input = word embedding + positional encoding
3. **Encoder stack**: Each encoder layer has two main sub-layers:
    a. Sub-layer 1: Multi-Head Attention (Bidirectional) used to capture different types of relationships.
    b. Sub-layer 2: Feed Forward used to position-wise a fully connected network and applies two linear transformations with ReLU activation.

After all N encoder layers, we obtain rich, contextualized representations of the entire input sequence and each token's representation incorporates information from all other tokens. These become the Keys (K) and Values (V) for the decoder's cross-attention.

4. **Decoder stack**: the output embeddings and position is similar to input embeddings and position , but for the target sequence being generated. Each decoder layer has three main sub-layers:
    a. Sub-layer 1: Masked Multi-Head Attention (causal) used to prevent attending to future positions (can only use past outputs).

b. Sub-layer 2: Cross Attention: this layer is the bridge between encoder and decoder.
c. Sub-layer 3: Feed Forward identical to encoder

5. **Output generation**: the linear layer projects decoder output to vocabulary size and Converts hidden representation to logits (scores) for each possible token. The Softmax converts logits to probability distribution. The token with the highest probability is chosen and appended to the sequence.
6. **Iterative Text Generation**: This selection and prediction process repeats iteratively to generate coherent and contextually relevant text until the output is complete.
7. **Final Output**: The generated tokens are converted back into text format and presented as the model's response.

Now that the fundamentals, structure, and operation of LLMs have been described, it is critical to think about their practical implications. Healthcare is one of the most revolutionary application sectors, where LLMs' capacity to handle unstructured clinical data and to produce contextually relevant insights makes them effective tools for diagnosis, decision support, and individualized care [129].

2.2 LLM in Healthcare

This section presents the evolutionary arc from AI to LLMs in healthcare, spanning three decades, then explores their key applications and the transformative impact they have demonstrated across clinical workflows.

2.2.1 AI and LLM Models' Evolution in Healthcare

Healthcare AI has evolved from inflexible, manually designed rule-based systems to adaptable, zero-shot generative AI that can do multimodal analysis and clinical reasoning spanning five evolutionary periods. Although they remained brittle and task-specific, rule-based systems such as MedLEE [38, 39] employed handcrafted grammars for clinical text extraction throughout the first Era (1994–1999). Statistical methods emerged during the second Era (2000s–2010), with MetaMap [10] proposing lexical-semantic matching for standardizing biomedical language through the Unified Medical Language System. With their pre-trained contextual representations, transformer models GPT-1 (2018) and BERT [29] have revolutionized the field in the Third Era 3 (2018–2019). Task-specific technical constraints were removed by clinical adaptations such as BioBERT [62] and ClinicalBERT [6]. Expert-level performance was attained by instruction-tuned models during the Fourth Era (2022–2023). While ChatGPT topped 60% accuracy across all United States Medical Licensing Examination (USMLE) levels [60], GPT-4 [85] showed emergent CoT reasoning on USMLE tests [84]. The first to achieve 80% USMLE

accuracy was Med-PaLM 2 [117]. Modern systems with multimodal capabilities emerged in the fourth Era (2024–2025), with Med-PaLM M [132] renowned for joint text-image diagnostic reasoning. Meanwhile, GPT-5 [141] stresses democratization, transparency, and ethical alignment in deployable biomedical AI, while domain-specific models such as BioMedGPT [72] and PubMedGPT [18] enhance performance.

2.2.2 LLM Applications in Healthcare Domain

The transition from general-purpose to healthcare-specialized LLMs requires fundamental architectural modifications and domain-specific training strategies. Recent surveys agree that four domains capture the bulk of LLM value in clinical practice [140]. The reminder of this section walks through each domain, highlighting representative systems and quantitative gains:

- **LLM and Medical Transcription**

The chosen LLM architecture for this healthcare application is a decoder-only transformer with causal self-attention mechanism, a design specifically optimized for text generation tasks. Decoder-only models are renowned for their ability to produce coherent, fluent, and contextually rich long-form text, making them particularly well-suited for document generation [21]. For instance, in a 2024 German deployment, a 7-billion-parameter BLOOM model and fine-tuned on local radiology reports produced 102 discharge letters; 93% were rated by two independent physicians as "usable as-is or with only minor edits," cutting after-hours charting by 38% while running fully on-premise to satisfy General Data Protection Regulation Privacy Rules. Complementing smaller open-source models, GPT-4.1-driven scribes first transcribe multi-speaker encounters with OpenAI's Whisper automatic speech recognition model [86], then prompt-engineer a structured Simple Object Access Protocol note; the two-stage workflow reached 4% word-error-rate on accented Indian English and trimmed documentation time from 13 to <3 min per consultation [45].

- **LLM and EHR mining/generation**

In EHR healthcare applications, both encoder-only and decoder-only transformer architectures offer distinct advantages, depending on the task. Domain-specific LLMs pre-trained on large-scale EHR corpora like GatorTron (90 B parameters) [151] or Med-PaLM, a decoder-only transformer with causal self-attention, and BioClinicalBERT, an encoder-only with bidirectional self-attention, now parse entire patient records, extract medication–problem relations, and generate structured problem lists with F1 scores $\geq$0.92, outperforming legacy rule engines by 12–18%. Open-source ClinicalBER similarly identifies social-determinant mentions in discharge summaries at 0.86 F1, a 0.22-point gain over dictionary methods, enabling richer predictive models and population-level research [64].

- **LLM and Clinical decision support**

For clinical decision support, selecting between encoder-only, decoder-only, or hybrid (encoder–decoder) LLM architectures depends on the nature of the clinical task and the required level of reasoning or generation. Recent advances have seen retrieval-augmented LLM pipelines transition from bench to bedside, enhancing explainability and safety in clinical workflows. For example, a 2024 multi-centre study integrated Claude-3.5-Sonnet, a decoder-only transformer, with real-time access to MIMIC-IV-ED vital signs and external Clinical Practice Guidelines documents. The system achieved 0.83 accuracy (approaching attending-level performance) for triage-level assignment and 0.79 top-1 diagnostic match, while halving under-triage rates compared to conventional Emergency Severity Index (ESI) rules. Notably, the same workflow reduced hallucination frequency from 14 to 4%, demonstrating that coupling generative LLMs with curated medical knowledge bases without the need for bespoke retraining can yield robust, explainable, and clinically safe decision support [40].

- **LLM and predictive health outcomes**

Generally, for predictive health outcomes, LLMs are typically used in combination with other machine learning techniques to analyze structured and unstructured clinical data [145]. However in recent years, LLMs are rapidly becoming "universal prediction engines" for real-world clinical events. NYUTron, a 4-billion-parameter BERT-style LLM pretrained on 7.25 M NYU Langone clinical notes, outperformed traditional structured-data models on five tasks: 30-day all-cause readmission (Area Under the Curve (AUC) 0.781 vs 0.726), in-hospital mortality, length-of-stay, comorbidity-index and insurance-denial prediction after simply treating the entire EHR as free text. In an external validation across four hospitals, fine-tuning only 10% of local notes recovered the performance drop, proving scalability [54]. Similarly, CPLLM (Llama-2-7 B continued on MIMIC-III) raised Perceiver-AUC (PR-AUC) for adult respiratory-failure prediction from 0.350 to 0.360 and the area under the ROC curve (ROC-AUC) from 0.746 to 0.764 over strong Med-BERT baselines, with token-level ablation confirming robustness [111]. Together these studies show that self-supervised LLMs can extract risk signals directly from physician prose, eliminating labour-intensive feature engineering and delivering measurable gains in prognostic accuracy.

No one can deny that LLMs have transformed medical language processing, yet healthcare extends beyond text to include imaging, histopathology, and biosignals [28]. Addressing these modalities requires more advanced generative models, where GANs emerge as a powerful complementary technology.

3 GAN: Architectural Foundation and Healthcare Transformation

GANs produce realistic synthetic data by learning underlying data distributions through a generator-discriminator interaction, in contrast to LLMs which concentrate on language challenges [88]. They are extensively utilized in pathology [131], radiography [113], and privacy-preserving data augmentation [95]. The evolution of the model and its different variations are demonstrated via key GAN structures. The section also discusses GAN's fundamental architecture and its use in healthcare as LLM.

3.1 GANs Architectural Foundations

A strong framework for producing realistic synthetic data is provided by GANs, which were first presented in [41]. The adversarial interaction between a Generator (G), which generates synthetic samples, and a Discriminator (D), which separates authentic from fraudulent data, allows GANs to learn the underlying data distribution. G can iteratively improve its outputs thanks to this minimax competition, resulting in high-fidelity data that closely resembles real-world distributions [9, 41]. The G and D networks that are trained simultaneously make up a GAN (Fig. 2).

The Generator *(G)* takes a random noise vector z (sampled from a prior distribution, typically uniform or Gaussian) and maps it to the data space. It transforms simple stochastic noise from a latent space into complex, high-dimensional samples. The *G* hidden layers typically includes upsampling layers (e.g., transposed convolutions) and dense layers with non-linear transformations (e.g., ReLU, LeakyReLU). The generated data *G(z)* is a fake sample intended to mimic real data. In a simple batch, the generator loss function is:

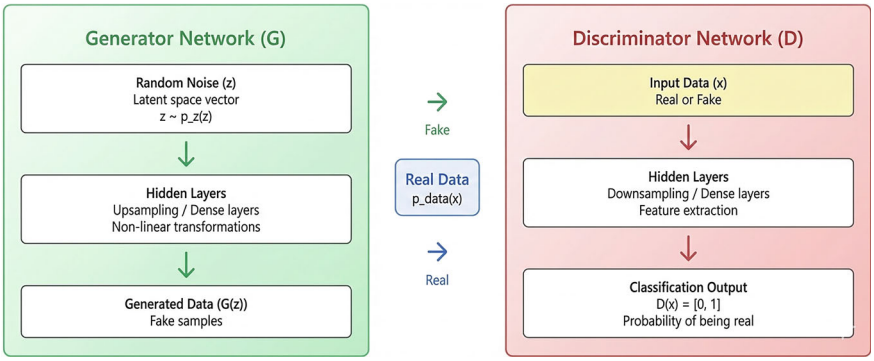


Fig. 2 GAN core architecture

$$L_g = -\frac{1}{N}\sum_{i=1}^{N} \log(1 - D(G(z_i)) \tag{2}$$

However, in practice, it is common to use the "non-saturating" heuristic, which provides stronger gradients and better training stability:

$$L_{gns} = -\frac{1}{N}\sum_{i=1}^{N} log(D(G(z_i)) \tag{3}$$

The discriminator (D) provides feedback that guides the G toward producing outputs statistically indistinguishable from real data. This "it takes a thief to catch a thief" paradigm underpins the co-evolutionary refinement of G's outputs [9, 41]. D inputs are real data x sampled from the real data distribution $p_{data}(x)$, or fake data G(z) generated by the Generator. The Discriminator does not receive both real and fake data simultaneously in a single forward pass. Instead, it processes two separate batches during training:

1. **Batch of Real Data**: where *D* receives a batch of real samples *x* drawn from the real data distribution $p_{data}(x)$. It outputs then *D(x)*, the probability that each sample is real.
2. **Batch of Fake Data**: where *D* receives a batch of fake samples *G(z)* generated by *G* from random noise z and outputs *D(G(z))*, the probability that each generated sample is real.

The *D* hidden layers typically include downsampling layers (e.g., convolutions) and dense layers for feature extraction. The D output is a probability $D(x) \in [1, 1]$, where $D(x) \approx 1$ indicates that *x* is likely real, and $D(x) \approx 0$ indicates that x is likely fake. In a simple batch, the generator loss function is:

$$L_d = -\frac{1}{N}\sum_{i=1}^{N} log(D(x_i)) + log(1 - D(G(z_i)) \tag{4}$$

The two networks are trained simultaneously through a minimax game. The generator tries to "fool" the discriminator by generating samples that look real, while the discriminator tries to correctly classify real and fake samples. The GAN training objective is given by the following minimax equation:

$$min_g max_d = \frac{1}{N}\sum_{i=1}^{N} log(D(x_i) + \frac{1}{N}\sum_{i=1}^{N} 1 - log(D(G(z_i)) \tag{5}$$

Convergence occurs when the generator produces data such that the discriminator cannot distinguish between real and generated samples, i.e. $D(x) = D(G(z)) = 0.5$ everywhere, this is the Nash equilibrium. Formally, for the value function $V(G, D)$:

$$V(G*, D) \leq V(G*, D*) \leq V(G, D*) \tag{6}$$

where G* and D* are the optimal generator and discriminator respectively. However, Farzan & Asuman [36] have shown that if standard zero-sum GAN games, subject to typical regularization and architecture constraints, may not yield Nash equilibrium solutions, the latter can be replaced by Stackelberg and proximal equilibrium.

3.2 GAN Variants

Since their inception, several GAN variants have improved training stability, output quality, and its core controllability. These variants are:

- **Deep Convolutional GANs (DCGANs)**: Replace fully connected layers with convolutional neural networks, enabling stable training and higher-quality images (e.g., faces, medical scans) [99].
- **Wasserstein GANs (WGANs)**: Mitigate mode collapse and instability by using the Wasserstein (Earth-Mover) distance, improving convergence and sample fidelity [43].
- **Conditional GANs (cGANs)**: Enable generation conditioned on specific labels or inputs, facilitating class-specific or style-specific outputs [77].
- **StyleGAN & StyleGAN2**: Achieve unprecedented image quality with controllable style features [5].
- **BigGAN**: Supports high-resolution, class-conditional generation suitable for large-scale datasets [20].

3.3 GANs in Healthcare

GANs have demonstrated remarkable adaptability and impact within the medical domain. The unique characteristics of clinical data (including limited sample sizes, stringent privacy regulations, and multimodal complexity) have positioned GANs as valuable tools for synthesizing medical datasets [106], augmenting training data [112], and enabling collaborative research while preserving patient confidentiality [113]. Key recent uses of GANs in healthcare are listed in Table 1, which highlights both their benefits and related difficulties.

Table 1 Analysis of GANs model in healthcare

Application	Description	Advantages and challenges	Limitations	Reference
Medical image synthesis	Generate synthetic images to augment datasets	–Expands datasets, improves class balance, and captures complex structures –Training instability and mode collapse remain issues	–Limited diversity –Risk of unrealistic anatomy –Metrics often fail to reflect clinical quality	[118]
Image segmentation	Identify structures like tumors or organs	–Adversarial loss enforces shape consistency and sharper boundaries –Requires substantial data; unstable training; sensitive to noise.	–No formal data-consistency guarantees –Hallucinated boundaries –High computation for 3D tasks	[7]
Image reconstruction	Reconstruct high-quality images from low-res/ incomplete data	–Produces sharper reconstructions than pixel-loss methods –Helps reduce radiation/scan time	–Possible artifacts or hallucinations –limited generalization with small datasets –Convergence issues	[148]
Super resolution	Upscale image resolution	–Improves fine-detail visibility –Adversarial loss increases realism	–Modality-dependent performance –Risk of invented details –Weaker than modern diffusion/Transformer models	[89]
Cross-modality translation	Convert images between modalities (e.g., MRI → CT)	–Enables missing-modality generation –Supports multi-modal workflows	–Hard to preserve diagnostic integrity –Structural inconsistencies and clinically risky without strong validation	[108]
Data augmentation	Create additional training data	–Increases robustness –Helps rare-class representation	–Mode collapse reduces diversity –Noise/artifacts degrade downstream models –GPU-intensive	[79]

(continued)

Table 1 (continued)

Application	Description	Advantages and challenges	Limitations	Reference
Denoising	Reduce image noise	–Preserves edges better than classical filters –Supports low-dose imaging	–Over-smoothing or hallucinating structures –Balancing noise removal vs. detail retention is difficult	[130]
Classification	Categorize medical images (e.g., diseased vs. healthy)	GAN-based augmentation improves classifier balance and stability	–Requires large annotated sets –Synthetic bias risk –Added computational load	[57]

3.4 Diffusion Models Versus GANs in Healthcare: An Empirical Comparison

Diffusion models [47] are a class of generative models that generate data by gradually adding and then removing noise from a sample. They are inspired by non-equilibrium thermodynamics and consist of two main processes:

- Forward Process (Noising): Gradually corrupt real data x_0 over T steps.
- Reverse Process (Denoising): trains a neural network to denoise and reconstruct x_0 from noise

Diffusion models may create high-fidelity, semantically precise images thanks to this iterative process, which is useful in fields like healthcare where precision is crucial. They are computationally costly due to the several denoising processes, which restricts their scalability for large-scale data augmentation.

A workstation equipped with an NVIDIA H100 GPU, 128 GB of RAM, and a 1 TB NVMe SSD was used for experiments comparing GAN and diffusion models. 5 synthetic samples were added to each image in the CIFAR-10 dataset (60,000 32 $\times$ 32 RGB images, 10 classes) to produce 300,000 images. To produce class-specific outputs, the generation process integrated class labels with structured prompts of the form "A photo of a {class}" in natural surroundings.

Semantic alignment was assessed using CLIP ViT-B/32 cosine similarity. Both the conditional Diffusion (condDiffusion) model (a 50-step DDPM) and the conditional GAN (condGAN) were trained using the same data, resolution, conditioning, and hardware.

Table 2 shows that condGANs demonstrate clear superiority over condDiffusion models across all evaluated metrics in this comparison. The condGAN achieves a higher mean CLIP score of 0.312 compared to 0.289 for condDiffusion, indicating approximately 8% better alignment between generated images and their conditioning inputs. More striking is the efficiency advantage: condGAN generates images nearly

Table 2 Performance comparison: GANs versus diffusion models

Metrics	condGAN	condDiffusion
Mean clip score	0.312	0.289
Time/image (s)	0,072	0,214
Throughput (img/s)	13,9	4,7
Iteration per sample	1	50
CPU memory (GB)	7,1	13,4

three times faster (0.072 s vs. 0.214 s per image) and achieves correspondingly higher throughput (13.9 vs. 4.7 images/second), primarily due to its single-iteration generation process versus the 50 iterations required by the diffusion model. Additionally, condGAN proves more resource-efficient, consuming only 7.1 GB of CPU memory compared to condDiffusion's 13.4 GB. These results suggest that for this particular task and dataset, conditional GANs offer a compelling advantage in both quality and computational efficiency, making them particularly suitable for applications requiring real-time generation or deployment in resource-constrained environments. This analysis highlights the versatility and potential of GANs in various healthcare applications, while also pointing out the challenges that need to be addressed to fully realize their benefits in clinical settings.

LLMs have transformed language understanding and GANs have advanced synthetic data generation, yet real-world healthcare applications demand AI systems that integrate multiple data modalities. Clinical practice involves analyzing multimodal data simultaneously, requiring multimodal AI frameworks that can process, reason, and generate outputs across these domains. Such integrated systems shift from single-domain models toward holistic architectures that better mirror human clinical reasoning, enabling more accurate and clinically relevant applications.

4 Multi-modal Healthcare System and AI

Multimodal AI systems constitute sophisticated computational architectures designed to simultaneously process, integrate, and synthesize information from heterogeneous data sources, including textual documents, medical imagery, audio signals, video streams, and sensor-derived physiological measurements [12, 121]**.** These architectures fundamentally differ from traditional unimodal systems (which operate exclusively within singular data domains such as NLP or computer vision) by leveraging advanced fusion techniques to create unified representational spaces that capture cross-modal dependencies and contextual relationships [2]. This paradigm emulates the inherent multimodal nature of human cognition, wherein sensory integration mechanisms seamlessly combine visual, auditory, tactile, and other perceptual inputs to construct coherent environmental interpretations and facilitate complex decision-making processes. This leads to a pivotal question.

Question what architectural mechanisms and learning strategies empower multimodal AI systems to effectively process, align, and integrate information across diverse data modalities?

To address this, we first establish the conceptual foundation for the next section, which systematically compares unimodal and multimodal systems in healthcare. The core architectural mechanisms and learning strategies that enable multimodal AI systems to efficiently process, align, and integrate heterogeneous clinical data is then described.

4.1 Unimodal and Multimodal System Comparison in Healthcare

Unimodal systems and multimodal systems represent two distinct approaches in the field of AI as well as in our context: healthcare [2]. Unimodal systems are designed to process and analyze a single type of data, such as text, images, or sensor readings. While these systems can be effective for specific, well-defined tasks, their reliance on a single data type limits their ability to capture the complexity and nuances of real-world scenarios [93]. This narrow focus can lead to siloed insights and reduced accuracy, especially when dealing with incomplete or missing data. In contrast, multimodal systems integrate and analyze multiple types of data simultaneously, such as combining medical imaging and EHRs [49]. By leveraging diverse data sources, these systems can achieve higher diagnostic accuracy, better adaptability to complex cases, and more robust performance in the presence of incomplete data [59]. Multimodal systems also support intuitive and flexible interactions, aligning more closely with the multifaceted nature of clinical workflows. However, the integration of multiple data types requires complex infrastructure and sophisticated data alignment techniques, which can pose challenges in terms of implementation and maintenance. Despite these challenges, the advantages of multimodal systems in enhancing diagnostic accuracy, clinical decision-making, and personalized patient care make them a powerful tool in modern healthcare [11]. Understanding the internal architecture of multimodal AI systems comprising unimodal encoders, fusion mechanisms, and output modules highlights how they are structurally equipped to handle complex, multi-source data environments. But beyond their architecture, it is essential to consider how multimodal systems fundamentally differ from traditional unimodal approaches in practice. This comparative overview illustrates the key differentiators and performance advantages making multimodal AI particularly powerful in healthcare: a domain requiring contextual depth, robustness, flexible interaction and human–computer interfaces.

4.2 *Architectural Frameworks and Data Fusion Strategies in Healthcare*

Multimodal healthcare architectural frameworks and data fusion strategies offer innovative approaches for integrating diverse data sources. Multimodal data fusion techniques are conventionally organized into three canonical approaches: early fusion, intermediate fusion, and late fusion [128].

- **Early fusion** (Fig. 3) combines raw or pre-processed data from multiple modalities prior to model input, capturing intricate relationships across modalities for potentially more accurate and holistic modeling.

- **Intermediate fusion** (Fig. 4) merges feature representations extracted from different modalities before processing, allowing the model to learn modality-specific patterns prior to integration.

- **Late fusion (or decision fusion)** (Fig. 5) aggregates predictions from each modality using methods such as majority voting, weighted averaging, or meta-learning models operating on individual modality outputs.

Each data fusion strategy has advantages as well as drawbacks (Fei et al. 2024). Early and intermediate fusion require careful synchronization of data or features across modalities, which can be challenging, especially with real-time or heterogeneous datasets. Late fusion, although more robust to noisy inputs, fails to capture

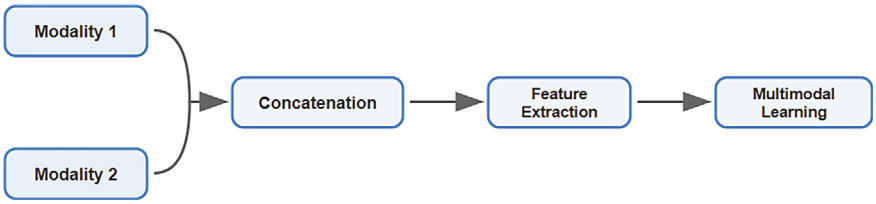


Fig. 3 Early fusion flow

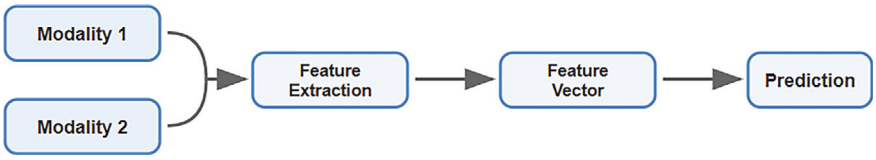
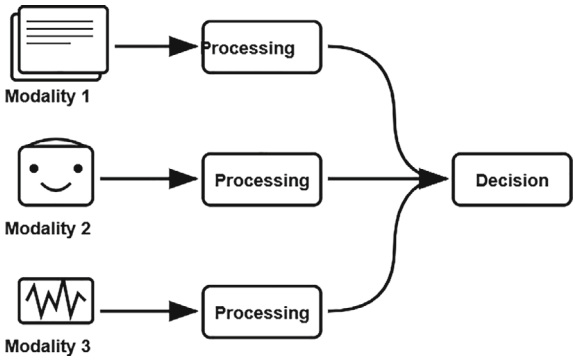


Fig. 4 Intermediate fusion flow

Fig. 5 Late fusion flow



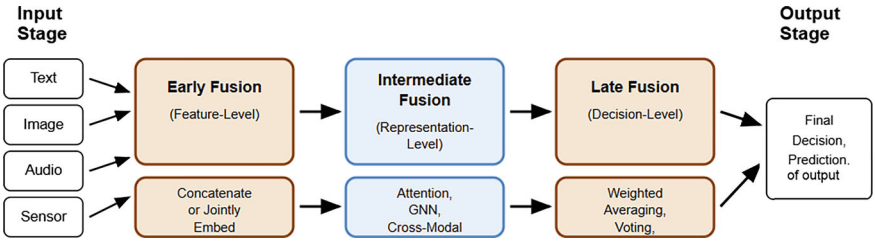


Fig. 6 Hybrid Fusion Framework for Multimodal Data

cross-modal interactions that intermediate fusion can exploit. These limitations motivate the adoption of **hybrid fusion that is discussed** in (Fei et al. 2024) which integrates elements from early, intermediate, and late fusion strategies to leverage the strengths of each while mitigating their weaknesses (Fig. 6).

Evaluating hybrid multimodal fusion approaches reveals key trade-offs: **Attention-based methods** generalize well to multiple modalities and offer high interpretability but are computationally demanding and only partially handle missing data. **Encoder-Decoder** and **Graph Neural Network (GNN)** approaches have moderate complexity and interpretability, with GNNs excelling at missing data support. **Generative Neural Network (GenNN)** methods are highly computationally intensive, opaque, and limited in scalability, while **constraint-based** methods are lightweight, handle missing data well, but offer moderate interpretability.

LLMs, GANs, and multimodal systems each bring complementary strengths to AI-driven healthcare. In the evolving landscape of **multimodal healthcare systems**, this convergence point invites a deeper exploration into how LLMs and GANs can be architecturally aligned and functionally fused. The next section delves into this synergy, examining how their integration enables the design of more intelligent, multimodal generative systems capable of reasoning and producing clinically grounded outputs across diverse data types.

5 The Synergistic Integration of LLMs and GANs in Multimodal Healthcare Systems

The convergence of LLMs and GANs is enabling powerful new multimodal frameworks. Understanding how these two complementary tools can be effectively integrated represents a critical objective for advancing scientific knowledge in the healthcare domain. For this purpose, the main architectural frameworks that highlight this synergy are presented. Then their underlying paradigms in the context of healthcare are discussed. In particular, an analysis on how this synergy manifests in terms of conceptual paradigms, functional roles, and practical applications are detailed. Finally, a comparison of the synergistic approaches with unified models to clarify their respective advantages and limitations is established.

5.1 Architectural Frameworks Synergy

In this section, the different architectures are explained in a general manner. Thus, in real world applications, the choice of GAN's variants can differ based on the type of data to deal with. The LLM is a decoder only since it focuses on generation, in other scenarios the choice can vary from encoder-decoder, encoder only or decoder only according to the specified task.

5.1.1 Direct Adversarial Integration Within an LLM

This direct adversarial framework proposes a forward-looking paradigm where adversarial mechanisms, traditionally implemented as distinct G–D components in GANs, are directly embedded within a single LLM.

The direct adversarial architecture: Instead of keeping separate generator and discriminator networks as in classical GANs, the framework folds the adversarial feedback loop into one decoder-only transformer stack: lightweight D-style sublayers are interleaved among the causal, left-to-right decoder blocks so that every token is produced and immediately critiqued by the same parameter set. As the model produces each successive text or multimodal token, the embedded D-style sublayers concurrently assess the evolving hidden representations for factual inconsistency, bias, safety violations, and cross-modal misalignment. The resulting adversarial score is injected back into the residual stream of subsequent transformer blocks, providing an immediate self-correction signal that influences downstream token predictions. This autoregressive generate-and-evaluate design is explored for tasks beyond the healthcare scope such as explainable fake-news detection [144].

The training methodology: Figure 7 illustrates the step-by-step sequence of training operations within a direct adversarial LLM. The diagram employs a modular design,

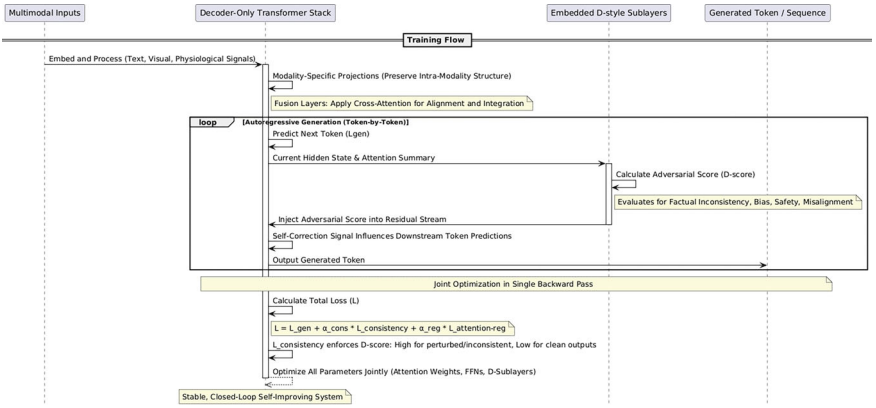


Fig. 7 Sequence diagram for direct adversarial integration within an LLM architecture

where each box represents a core functional block of an LLM. These blocks are interconnected through a structured workflow, depicted by directed arrows that visualize the flow of data and control signals between components.

(1) **Multimodal inputs** are embedded and processed through a causal **decoder stack**, where **masked self-attention** captures sequential and intra-modality dependencies, and modality-specific projections preserve structural integrity before fusion.
(2) At designated **fusion layers**, **cross-attention mechanisms** are applied to align and integrate complementary information across modalities while maintaining autoregressive causality.
(3) **Embedded D-modules**, implemented as lightweight dense layers operating on the current hidden state and attention-summary vectors, output an **internal adversarial score** for each generation step.
(4) A **consistency loss** enforces higher adversarial scores for deliberately perturbed or inconsistent hypotheses, while encouraging low scores for clean, semantically coherent outputs. The overall training objective is defined as:

$$L = L_{gen} + \lambda_{cons} L_{const} + \lambda_{reg} L_{att-reg} \tag{7}$$

where:

L: total training loss minimized in one backward pass.

L_{gen}: standard generative loss.

L_{const}: adversarial consistency term that encourages low D-score for clean hypotheses and high D-score for deliberately perturbed/inconsistent ones; implemented as a hinge or margin loss.

$L_{att-reg}$: penalty that drives attention weights toward a clinically relevant mask.

λ_{cons}, λ_{reg}: positive scalar coefficients that balance the three objectives, chosen by grid search on a validation set.

All parameters including attention weights, feedforward networks, and D-style sublayers are optimized jointly in a single backward pass, without introducing an external discriminator or alternating optimization schedule. This decoder-only formulation yields a stable, closed-loop self-improving system in which adversarial feedback is naturally integrated into autoregressive generation.

5.1.2 Bidirectional GAN-LLM Feedback Framework

This design enables the LLM to generate sequences and semantically evaluate multimodal inputs within the same forward pass, while the GAN focuses on realism and modality-specific fidelity.

The bidirectional GAN-LLM architecture: this framework introduces a bidirectional closed-loop interaction in which an LLM and a GAN iteratively refine each other's outputs, enabling real-time error correction, semantic alignment, and cross-modal consistency [30]. The architecture couples a decoder-only, causal LLM with a conventional GAN (G–D pair). Causal self-attention governs left-to-right generation, while periodic cross-attention layers attend to GAN-produced multimodal embeddings without breaking autoregressive causality. This separation allows the LLM to handle generation and semantic evaluation, while the GAN enforces realism and modality-specific fidelity.

The training methodology: As illustrated in Fig. 8, the training methodology of the proposed framework is structured around three core principles:

1. **Bidirectional interaction**: The GAN's G produces initial multimodal outputs, while the GAN's D evaluates realism and modality fidelity. These outputs are then processed by the LLM, which performs semantic evaluation, factual consistency checks, and cross-modal alignment using its self and cross-attention layers.
2. **Iterative refinement cycles**: The LLM generates structured feedback signals, error descriptors, attention-guided corrections, and modality-specific constraints which the GAN's G uses to refine its outputs in the next iteration. The GAN's D continues to provide adversarial feedback alongside LLM-guided corrections. The GAN then adjusts its outputs based on LLM's feedback, generating revised data, which the LLM re-evaluates until GAN convergence is achieved.
3. **Autonomous closed-loop operation**: The refinement loop operates autonomously, enabling joint improvement of LLM reasoning and GAN generation, with optional human-in-the-loop supervision introduced at selected checkpoints to correct semantic drift and ensure domain validity [22].

These principles are operationalized through a joint decoupled optimization process that aligns the GAN and LLM components, guided by specialized loss functions:

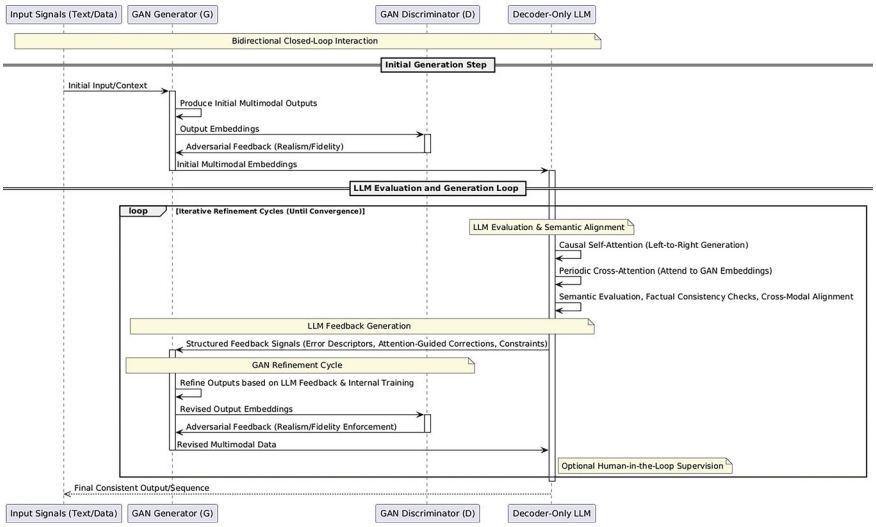


Fig. 8 Sequence diagram for the bidirectional GAN-LLM Feedback framework architecture

- The G is trained with:

$$\mathrm{L_G} = \mathrm{L_{adv}} + \alpha \mathrm{L_{sem}} \tag{8}$$

where $\mathrm{L_{adv}}$ enforces realism and $\mathrm{L_{sem}}$ penalizes semantic inconsistencies identified by the LLM.

- The D is optimized using the standard adversarial loss: $L_D = L_{adv}$
- The decoder-only LLM is updated via:

$$L_{LLM} = L_{ce} + \beta L_{rl} \tag{9}$$

where $\mathrm{L_{ce}}$ denotes supervised cross-entropy loss and $\mathrm{L_{rl}}$ present the rewards for accurate critique, correction quality, and alignment stability.

- At each refinement step, LLM-derived error vectors and attention signals are injected into G loss function L_G. The self-attention mitigates semantic drift and cross-attention preserves modality alignment.
- β, α: weighting factors to balance alignment and refinement contributions.

5.1.3 Hierarchical Integration Models

The hierarchical integration model is a multi-tiered architecture representing an emerging paradigm in multimodal healthcare AI. In this paradigm, reasoning and generation are distributed across distinct but coordinated layers. LLM operates at

higher levels of semantic reasoning, while GAN functions at lower perceptual layers, producing concrete modality-specific outputs [141]. These tiers are linked through a supervisory mechanism that enforces coherence between abstract intent and tangible generative results.

The hierarchical integration architecture: this framework is composed of a three-tier processing pipeline. The reasoning layer (decoder centric LLM with self and cross-attention) handles semantic understanding, diagnostic reasoning, and structured prompt generation. The perceptual layer (GAN) produces modality-specific outputs conditioned on the LLM's guidance. A supervisory layer is not a generative model, it is a mechanism to regulate optimization between the reasoning and perceptual layer. It enforces semantic and coherence, and can be an implemented functional module that computes cross-tier alignment losses. Human-feedback rewards can optionally be injected.

The training methodology: The training steps as illustrated in Fig. 9, starting with the reasoning layer, in which multi-model inputs are fed.

(1) **Reasoning Layer:**

- Performs semantic understanding, diagnostic reasoning, and structured prompt generation
- Causal **self-attention** models sequential dependencies within text or latent reasoning tokens
- **Cross-attention** aligns LLM representations with multimodal perceptual embeddings

$$L_{LLM} = L_{ce} + \beta L_{align} + \gamma L_{RL} \tag{10}$$

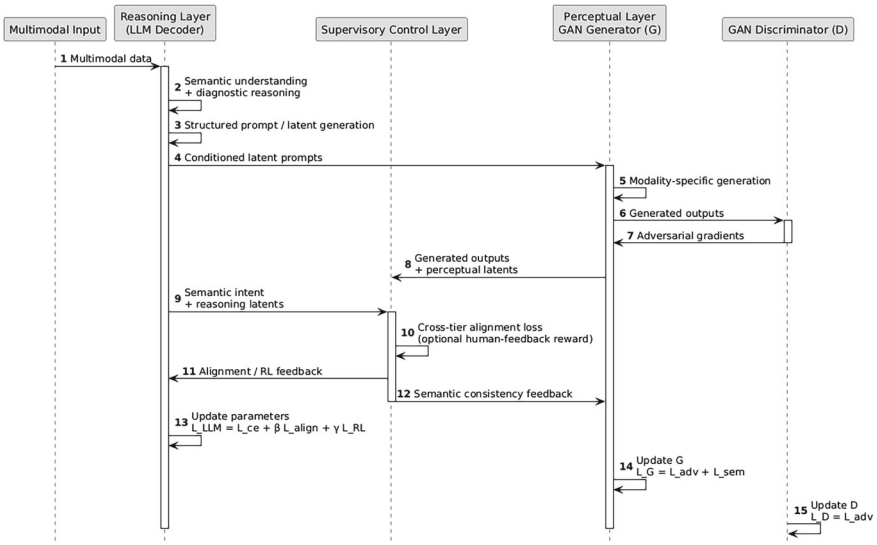


Fig. 9 Sequence diagram for hierarchical integration models architecture

where:

- L_{ce}: cross-entropy loss for autoregressive sequence generation.
- L_{align}: alignment loss ensuring LLM outputs match multimodal perceptual representations.
- L_{RL} (optional): reinforcement learning term from human feedback.
- β, γ: weighting factors to balance alignment and RL contributions.

(2) **Perceptual Layer (GAN)**:

- Produces modality-specific outputs conditioned on LLM guidance
- G uses cross-attention to LLM-derived latent prompts to preserve high-level semantic consistency.

$$L_G = L_{adv} + \alpha L_{sem} \tag{11}$$

$$L_D = L_{adv}$$

(3) **Supervisory Control Layer:**

- Coordinates reasoning and perceptual layers, enforcing consistency between abstract intent and concrete generation
- Optionally incorporates reinforcement learning from human feedback to maintain domain validity

$$L_{sup} = \delta L_{cons} + (optionally)\varepsilon L_{human} \tag{12}$$

where L_{human} is the reward or correction from human-in-the-loop feedback.

To conclude, training proceeds in an iterative closed-loop manner. Outputs generated by the perceptual layer (Step 3) are fed back to the reasoning layer (Step 1) through cross-attention and supervisory alignment signals, enabling semantic re-evaluation and updated guidance. This loop reasoning, generation, and supervisory correction is repeated continuously, with adversarial and alignment losses jointly optimized, until the GAN converges to stable, semantically aligned, and perceptually realistic outputs. Human feedback is incorporated via a learned reward model, which maps expert evaluations of generated outputs to scalar rewards. This signal is used to adjust both LLM and GAN parameters through reinforcement learning, ensuring domain validity and semantic alignment.

5.1.4 Cross-Modal Attention Framework

The cross-modal attention architecture is built on tightly coupled LLM and GAN components that communicate through attention-based interfaces, enabling real-time semantic control over perceptual generation [81]. Recent surveys of generative AI emphasize the role of attention-based architectures in fusing multimodal

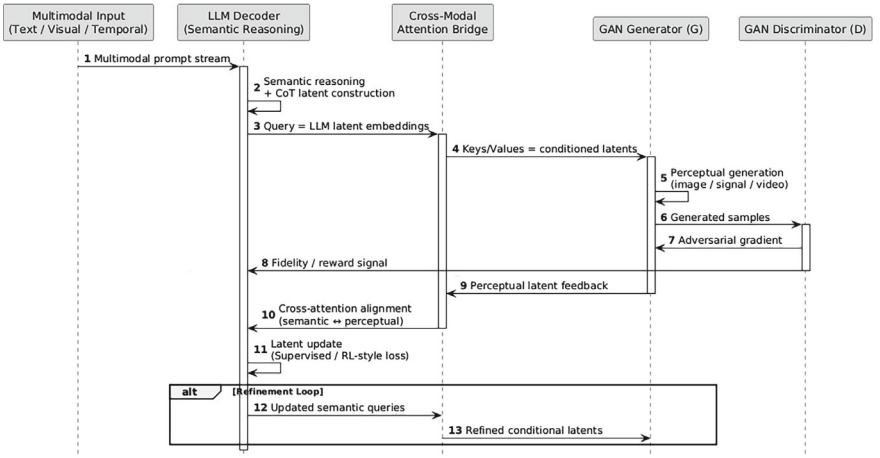


Fig. 10 Cross-Modal Attention framework architecture

representations within complex reasoning and generation pipelines [107]. In this framework, LLM-derived latent representations dynamically guide GAN generation via cross-attention, enabling joint reasoning over textual, visual, and temporal data streams.

The cross-modal attention architecture: At its core, the architecture consists of a decoder-centric LLM, a GAN-based perceptual G–D pair, and a set of cross-modal attention bridges. The LLM functions as the semantic and reasoning engine**.** The GAN acts as the perceptual component, while cross-modal attention layers serve as the key architectural bridge between the two models.

The training methodology: As illustrated in Fig. 10, the training methodology is built upon three core components starting with feeding the decoder with multimodal inputs:

LLM Decoder:

- Generates structured latent embeddings for semantic reasoning.
- Optimized via supervised or reinforcement-style loss for reasoning, attention focus, and context-aware guidance.
- Processes each intermediate prompt dynamically in real time, without waiting for other prompts.

GAN:

- **G**: Produces modality-specific outputs conditioned on LLM embeddings via cross-attention; adversarial loss augmented with LLM signals.
- **D**: Ensures realism and modality fidelity using standard adversarial loss.

Cross-Modal Attention:

- Enables online, bidirectional exchange between LLM and GAN latent spaces.
- Queries from the active module, keys/values from the other, supporting real-time, causally compatible interaction.

Attention Mechanisms in Cross-Modal Framework

In Self-Attention: Q, K, and V are derived from the same hidden states, allowing the model to attend within its own sequence (as described in Sect. 2.2.1): $Q = K = V = XW$ Where X represents the hidden states of the current layer and W is the learnable projection matrix. That enables intra-sequence dependencies to be captured for coherent reasoning or generation within a single modality.

In Cross-Modal Attention: While the Q come from one component, K and V come from another. This allows dynamic, real-time information exchange between modalities:

$$Q = H_{LLM} W_Q, k = H_{GAN} W_K \text{ and } V = H_{GAN} W_V \tag{13}$$

where H_{LLM}: latent embeddings produced by the LLM decoder.
H_{GAN}: latent feature maps produced by the G.
W_Q, W_K and W_V: learnable projection matrices for queries, keys, and values.

5.1.5 CoT-Conditioned Generation

CoT-conditioned generation represents a novel class of architectures in which the LLM's step-by-step reasoning process [147] is not limited to producing final textual conclusions, but instead yields structured intermediate representations that directly condition GANs. In this paradigm, the reasoning trajectory itself becomes a scaffold for generation: each intermediate reasoning step encodes hypotheses, constraints, or semantic attributes that guide downstream synthesis.

The CoT-conditioned generation architecture: it consists of three main components: an LLM decoder, a G, and a D, with attention mechanisms ensuring alignment and interaction between them.

The training methodology: As illustrated in Fig. 11, the training methodology is built upon three core principles:

(1) **LLM Decoder**
- Inputs: raw data + problem context.
- Generates structured CoT prompts that guide the generator.
- Updates reasoning using discriminator feedback and reward signals to improve alignment of generated outputs with intended CoT.

(2) **G**

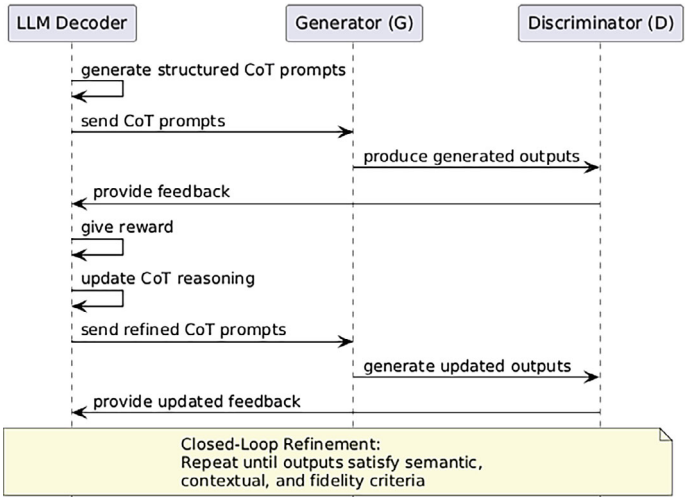


Fig. 11 CoT-conditioned generation architecture

- Produces data conditioned on the LLM's structured CoT.
- Optimized to maximize alignment with CoT while fooling the discriminator.

(3) **D**

- Evaluates realism and fidelity of generated outputs.
- Provides feedback that informs LLM's reasoning update and reward computation.

Closed-Loop refinement involves the LLM integrating feedback and rewards from the discriminator to iteratively improve its CoT reasoning. The updated CoT is then reintroduced to the generator for further iterations. This process repeats continuously until the generated outputs meet the desired semantic, contextual, and fidelity criteria.

The system uses a combination of losses across components to jointly optimize reasoning, generation, and evaluation:

$$L_{total} = \lambda_{LLM} L_{LLM} + \lambda_G L_G + \lambda_D L_D \tag{14}$$

where the λ coefficients balance contributions from reasoning, generation, and evaluation.

5.1.6 Hybrid LLM-GAN Synergy Framework

The Hybrid LLM–GAN Synergy Framework unifies multiple LLM–GAN integration paradigms into a single system, combining reasoning, adversarial evaluation, hierarchical planning, and cross-modal alignment. By embedding both generation and evaluation capabilities within the LLM while leveraging a perceptual GAN

backend, the framework enables iterative, reasoning-aligned multimodal synthesis with enhanced interpretability, auditability, and semantic consistency. MALLM–GAN [68] is a specific implementation of internal adversarial emulation within the broader Hybrid LLM–GAN Synergy Framework. However, it does not fully integrate the hierarchical planning, CoT-conditioned generation, and cross-modal alignment proposed in this work.

The hybrid framework Architectural: The hybrid framework is structured into four main components: the LLM-Centric decoder, which unifies generation, evaluation, and planning; the CoT-driven reasoning and refinement loop, which guides stepwise reasoning and output conditioning, the hierarchical LLM–GAN execution layer, which directs multimodal synthesis across levels of abstraction, and the cross-modal attention and alignment mechanisms, which ensure coherence and alignment between reasoning and generated outputs.

The training methodology: As illustrated in Fig. 12, the training methodology is built upon four core principles:

1. **LLM-Centric Decoder (Generator–Evaluator–Planner Unification)**

At the heart of the framework lies a single LLM decoder, which simultaneously assumes three traditionally disjoint roles:

- G (semantic): Produces structured CoT reasoning traces, intermediate prompts, and latent conditioning representations.
- Evaluator (adversarial emulation): Internally critiques and scores candidate outputs using learned self-consistency, plausibility, and constraint satisfaction signals emulating the D's role within the LLM itself.
- Planner (hierarchical control): Decomposes high-level clinical or multimodal objectives into ordered sub-tasks, assigning generation priorities and constraints across modalities.

This internal adversarial emulation allows the LLM to iteratively refine outputs before and after perceptual synthesis, reducing reliance on external discriminators and enabling self-supervised validation loops.

2. **CoT-Driven Reasoning and Refinement Loop**

These CoT reasoning steps condition both: the GAN's G (for perceptual realization) and the LLM's own internal evaluator. As a result, generation unfolds as a reasoning-aligned, stepwise and streamlined refinement process, rather than a single-shot synthesis. Every generated artifact is traceable to explicit reasoning steps, enabling interpretability and auditability.

3. **Hierarchical LLM–GAN Execution Layer**

The LLM provides hierarchical guidance to the perceptual layer:

- High-level semantic intent: defined at early planning stages.
- Mid-level structural constraints: refined through CoT steps.

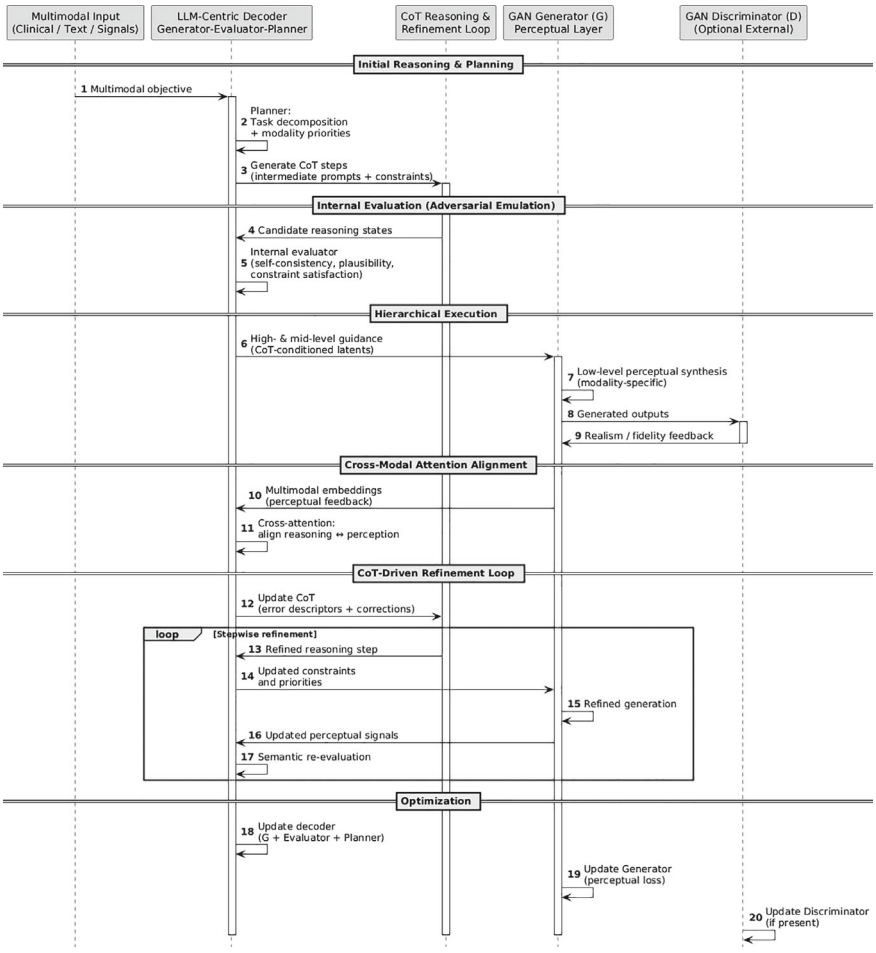


Fig. 12 *Hybrid LLM-GAN synergy* architecture

- Low-level perceptual realization: executed by the GAN G.

The GAN remains responsible for modality-specific synthesis, while the LLM governs what, why, and in what order generation occurs.

4. Cross-Modal Attention and Alignment Mechanisms

Cross-modal attention serves as the glue of the hybrid system:

- **LLM cross-attention** layers align textual reasoning with multimodal embeddings, producing structured CoT signals.
- **GAN attention** blocks are conditioned on CoT outputs from the LLM:

Generator attention uses cross-attention to align latent codes and reasoning signals with spatial or temporal features, ensuring outputs follow the intended semantic and structural constraints.
Discriminator attention applies both spatial and cross-modal attention to focus on the most informative regions and modalities, evaluating whether generated data aligns with LLM reasoning and detecting local artifacts or semantic misalignments.

- **Bidirectional feedback** propagates from the GAN back to the LLM, enabling iterative refinement of reasoning and generation.

The loss function for the hybrid architecture is: $L_{hybrid} = \lambda_1 L_{CoT} + \lambda_2 L_{Eval} + \lambda_3 L_{Plan} + \lambda_4 L_{GAN} + \lambda_5 L_{Perc} + \lambda_6 L_{Struct} + \lambda_7 L_{CMA} + \lambda_8 L_{Attn}$
Where λ_i are hyperparameters controlling the contribution of each component.
L_{Eval}: LLM self-evaluation of output plausibility and consistency.
L_{Plan}: Hierarchical planning alignment between subtasks and objectives.
L_{Perc}: Perceptual consistency between generated and target features.
L_{CMA}: Cross-modal alignment of LLM reasoning and GAN outputs.
L_{Attn}: Attention alignment between LLM reasoning and GAN generation.

Beyond presenting the architectures (which define the structural frameworks for integrating LLMs and GANs), it is equally critical to emphasize the underlying paradigms (Table 3). These paradigms articulate the practical strategies through which these architectures are deployed, governing the dynamics of interaction, feedback, and co-evolution between the models. By examining these paradigms, we establish the conceptual bridge linking individual model design to the broader functionality of hybrid LLM–GAN systems. The table below systematizes these synergistic paradigms, highlighting their core mechanisms.

Having established the architectural paradigms, we now examine the practical applications of LLM–GAN synergy in healthcare, with emphasis on their roles and their clinical impact.

5.2 *LLM–GAN Synergy in Healthcare: Roles and Applications*

Within healthcare, these paradigms enable a spectrum of multimodal clinical applications, including diagnostic report generation, medical image synthesis, patient-specific treatment recommendations, synthetic EHR and biosignal generation. However, it is important to note that the integration of LLM–GAN architectures in clinical practice remains largely under active exploration and is not yet widely adopted [104]. Table 4 provides an operational breakdown of LLM and GAN responsibilities across concrete healthcare pipelines. It demonstrates how the synergy between semantic reasoning and generative synthesis directly translates into tangible clinical outcomes.

Table 3 LLM–GAN architectures and generative paradigms

Architecture	Paradigm and primary mechanism
Direct adversarial integration	Direct-Adversarial Integration LLM-guided adversarial synergy with self-reflective iterative refinement for aligned synthetic outputs
Bidirectional feedback framework	LLM-Guided Generative Models: Closed-loop refinement (generation → audit → revision) Iterative LLM-guided generation with auditing and revision for aligned synthetic outputs
Hierarchical integration models	Agentic Workflows & Collaboration: Hierarchical multi-tier control with agent collaboration for guided generative synthesis
Cross-modal attention frameworks	Cross-Modal Alignment: Attention-based token-to-latent alignment enabling cross-modal integration
Cot-conditioned generative frameworks	Structured Cot-conditioned generation with control codes for precise reasoning
Hybrid LLM-GAN synergy framework	LLM generates, evaluates, and plans via CoT; GANs refine outputs with adversarial, hierarchical, and cross-modal integration

Collectively, these applications illustrate how LLMs and GANs function as interoperable, synergistic components within multimodal healthcare AI systems. Through iterative refinement, cross-modal alignment, and hierarchical reasoning, generated outputs achieve semantic fidelity, temporal coherence, and clinical relevance. As the underlying infrastructures evolve toward modularity, agentic autonomy, and scalable deployment, these systems are poised to support continuous, context-aware, and human-aligned clinical intelligence, bridging the gap between computational reasoning and real-world decision-making.

However, these advances also magnify risks of algorithmic bias, opacity, privacy challenges, regulatory misalignment, and erosion of clinician trust [11]. It is therefore essential to evaluate not only the technical capabilities but also the implications of deploying LLM–GAN systems responsibly in clinical environments.

5.3 Synergy Versus Unified Modals

The choice between employing LLM-GAN synergy architectures versus unified end-to-end models depends on several practical and technical considerations. While unified models offer simplicity and seamless integration, LLM-GAN synergy approaches can provide greater flexibility, modularity, and the ability to leverage pre-trained components independently. Table 5 outlines the key scenarios and characteristics that favor each approach, helping practitioners determine the most appropriate

Table 4 Key applications of LLM-GAN synergy in healthcare

Application domain	LLM role	GAN role	Clinical impact
Medical video generation for education & diagnosis	Decomposes textual prompts into step-wise spatio-temporal instructions	Renders photorealistic, temporally coherent videos	Improves diagnostic accuracy in trainee cardiologists; allows repeatable simulation of rare cases as demonstrated in VIDEOPHY [13]
Cross-modal image synthesis	Converts radiology reports into structured latent descriptors of anatomy & pathology	Translates across modalities with semantic preservation	–RESVIT [27]: demonstrates high-fidelity Cross-modal synthesis, achieving an SSIM greater than 0.92 when translating pelvic CT scans to MRI –VTGAN [56]: effectively generates fluorescein angiograms from fundus images, attaining an AUC of 0.89 for age-related macular degeneration detection, highlighting its potential for clinically relevant image synthesis
Synthetic EHR & bio signals for privacy/ rare-disease modeling	Generates realistic clinical narratives respecting statistical & causal constraints	Produces tabular vitals, labs, streams matching LLM-generated narratives [68]	Downstream machine learning models trained on synthetic data achieve performance within 3% of the area under the receiver operating characteristic curve compared to real intensive care unit datasets, while ensuring differential privacy with a privacy budget $\varepsilon < 1$

architecture based on their specific requirements, including factors such as dataset size, computational resources, task complexity, and the need for interpretability or fine-grained control over the generation process.

Overall, LLM–GAN synergy and unified multimodal models address different clinical and practical needs: unified models favor end-to-end reasoning at scale, while LLM–GAN synergy offers greater modularity, explainability, and effectiveness

Table 5 LLM-GAN synergy versus Unified model

Features	When to use LLM-GAN synergy	When to use unified model
Primary goal	Creating highly realistic new synthetic data (images/tabular) for augmentation, privacy, or simulation, combined with complex text analysis [37]	End-to-end understanding and generation across different modalities (text, image, audio) with a focus on holistic, real-time support [70]
Task type	Data Augmentation for rare diseases (generate synthetic images/records). Privacy preservation (generate synthetic patient datasets). Modality Translation (e.g., generating a chest X-ray from a clinical report) [125]	Clinical Decision Support (analyzing patient notes, labs, and images to suggest a diagnosis). Multi-modal Summarization (summarizing an entire patient case including text and image findings). Medical Chatbots that can interpret images and respond in natural language [73]
Data requirements	Need to work with separate, specialized data types (e.g., medical images for GANs, clinical notes for LLMs). Especially beneficial when one modality is data-scarce [152]	Focus on a single, heterogeneous data stream that combines text, image, and sometimes tabular data for a unified representation. Requires massive, multi-modal pre-training data [70]
Computational needs	Higher complexity in architecture and training (two distinct models and an integration layer). Requires managing two separate pipelines [152]	Higher upfront cost for pre-training the massive foundational model, but potentially simpler inference (one model to run for multi-modal tasks) [70]
Explainability [114]	Easier to pinpoint failure if the image is poor, the GAN is at fault; if the text response is illogical, the LLM is at fault. Separate components allow for targeted improvements	Often a “black box” for multi-modal outputs. Difficult to attribute errors to the text or visual processing component

in data-scarce, privacy-sensitive healthcare settings. These trade-offs motivate an empirical validation of the proposed synergy to quantify its performance, robustness, and real-world applicability, as presented in the next section.

6 Empirical Validation of the Synergy

This section provides a quantitative evaluation of the synergy architectures detailed in Sect. 5.1, with the goal of augmenting the DEAP dataset (Database for Emotion Analysis using Physiological Signals) [58] through controlled synthetic data generation. This enables empirical validation of the generative capacity and diversity gains

for each synergy architecture, as well as for the baseline enrichment methods (GAN-only and LLM-only). All synthetic samples are then integrated with real recordings to create augmented training sets. The final objective of the data augmentation pipeline is to inject the generated samples into an LLM-based downstream classification model tasked with stress versus non-stress prediction. The resulting classification performance serves as an indirect proxy for assessing whether the augmented data effectively enhances the model's generalization capability. However, classification accuracy alone is insufficient to characterize the quality or reliability of the generated data. Consequently, the benchmarking protocol extends beyond predictive performance and incorporates comprehensive evaluation metrics and statistical validation procedures to assess distributional consistency, robustness, and variance across augmentation strategies. In addition, qualitative validation by human domain experts is performed to verify the semantic plausibility and clinical relevance of the synthesized samples. All models and augmentation strategies are evaluated under identical experimental settings to ensure fair and reproducible comparison.

6.1 Dataset, Device and Compute Requirements

6.1.1 Dataset Description

Experiments were conducted on a workstation equipped with an NVIDIA H100 GPU, 128 GB RAM, and 1 TB NVMe SSD storage. In order to use the dataset an End-User License Agreement (EULA) agreement paper was signed to respect authors' copyright. DEAP comprises 32 subjects, 40 one-minute music-video trials per subject and 22 participants frontal face videos. Each trial includes:

1. **Signals**: An ElectroEncephaloGram (EEG) was recorded from 32 scalp channels at 512 Hz to capture high-temporal-resolution neural activity. In parallel, eight peripheral physiological channels (ElectroEncephaloGram ECG, PhotoPlethysmoGraphy PPG, Galvanic Skin Response GSR, respiration, skin temperature, ElectroMyoGraphy EMG, and ElectroOculoGraphy EOG) were acquired to characterize cardiovascular, electrodermal, respiratory, muscular, and ocular responses.
2. **Self-reports**: After each trial, participants rated arousal, valence, dominance, liking, and familiarity using standardized Self-Assessment Manikin (SAM) scales (1–9 for all except familiarity, which is 1–5). These ratings capture affective dimensions, with arousal serving as a validated proxy for physiological stress, while valence, dominance, liking, and familiarity provide complementary context for affective modeling.
3. **Stress Label Definition**: Stress was labeled as follow:
   - *Stressed*: arousal $\geq$6, valence $\leq$4 and dominance $\leq$5
   - *Non-stressed:* arousal <6, valence >4 and dominance >5

Because DEAP lacks explicit stress labels, stress was defined as a proxy derived from arousal, valence, and dominance scores, capturing high activation, negative affect, and low control. This serves as a computational surrogate rather than a clinical diagnosis. Evaluation followed a leave-one-subject-out protocol, and the stress label was used solely as a conditioning signal for GAN- and LLM-based generation.

6.1.2 Dataset Pre-Processing Pipeline

All preprocessing steps were applied identically to real, synthetic, and reconstructed data, ensuring reproducibility and feature-space consistency.

1. **Signal Conditioning**: The physiological signals underwent a systematic preprocessing pipeline to ensure data quality and consistency. EEG data were bandpass filtered between 0.5 and 45 Hz using a zero-phase, 4th-order Butterworth filter, with notch filtering applied at 50 Hz and its harmonics (100 and 150 Hz) to remove power line interference. The EEG channels were then re-referenced to a common-average reference and down-sampled to 128 Hz using a Finite Impulse Response FIR, an anti-aliasing filter. Peripheral physiological signals were baseline-corrected by subtracting the mean activity during a 3-s pre-trial period. Finally, artifact rejection was performed by discarding trials exhibiting peak-to-peak EEG amplitudes exceeding 150 μV, resulting in the removal of approximately 3.8% of trials.
2. **Feature Extraction**: A comprehensive set of features was extracted from both EEG and peripheral physiological signals to capture multimodal physiological responses. For each EEG channel, time-domain features included mean, variance, and Root Mean Square (RMS), with skewness excluded due to low test–retest reliability (Intraclass correlation coefficients (ICC) < 0.4). Spectral power was computed across five frequency bands delta (0.5–4 Hz), theta (4–8 Hz), alpha (8–13 Hz), beta (13–30 Hz), and gamma (30–45 Hz) using Welch's method with 256-point windows and 50% overlap. Additionally, Hjorth parameters (activity, mobility, and complexity) were calculated, and time–frequency representations were obtained via Short-Time Fourier Transform (STFT) using 256-point windows with a hop size of 64, yielding 92 frequency bins across 59 time frames. From peripheral signals, cardiovascular features included Heart Rate (HR), Root Mean Square of Successive Differences (RMSSD), and the Low-Frequency to High-Frequency (LF/HF) ratio. Electrodermal features comprised Skin Conductance Response (SCR) count and mean Skin Conductance Level (SCL), while additional features captured respiration rate, mean skin temperature, and RMS electromyography activity from the zygomaticus and trapezius muscles.
3. **Normalization and Data Integrity**: To ensure consistent scaling across physiological features while preventing data leakage, a subject-specific normalization strategy was employed. Feature-wise mean and standard deviation were computed for each subject using only the training folds, with robust median and

InterQuartile Range (IQR) normalization applied specifically to GSR features due to their high variability. Critically, the identical normalization parameters derived from the training data were consistently applied to the validation, test, and synthetic samples, ensuring that no information from held-out data influenced the preprocessing pipeline**,** thereby maintaining the integrity of the model evaluation.

4. **Data Augmentation Environment**: The data augmentation framework was designed to accommodate both GAN-based and LLM-based generator architectures, each with tailored input representations and conditioning strategies. GAN-based generators accepted either EEG spectrograms with dimensions of 92×59 or 328-dimensional vectorized feature representations, with the EEG-only subset preferred during training to enhance stability. Stress labels were incorporated through a 1-bit embedding that was projected across all generator layers to ensure consistent conditioning throughout the generation process. In contrast, LLM-based generators operated on 512-token sequences constructed from 8-bit quantized features, augmented with structured headers and chain-of-thought (CoT) reasoning tokens to facilitate interpretable generation. For these models, stress conditioning was implemented by injecting the stress label as a discrete token within the input sequence, enabling precise control over the generated samples' target class.

Synthetic samples are projected back to the full 520-dimensional multimodal feature space after generation to maintain downstream evaluation consistency.

6.1.3 Consistency and Plausibility Constraints

Generated samples were required to satisfy feature-space and biophysical constraints:

1. **Feature consistency**: To ensure the fidelity of generated samples, strict feature consistency constraints were enforced throughout the generation process. Generated data were required to maintain delta-band power with a Mean Absolute Error (MAE) below 5% relative to real data, while Hjorth complexity parameters were constrained to remain within $\pm 10\%$ of the statistical moments observed in authentic recordings. Any violations of these physiologically informed boundaries were explicitly penalized through a feature-matching loss term, ensuring that synthetic samples preserved the essential statistical and spectral characteristics of genuine physiological signals.
2. **Biophysical plausibility**: Beyond statistical consistency, generated samples were subject to rigorous biophysical plausibility checks to ensure they adhered to known physiological constraints. EEG Power Spectral Density (PSD) was required to exhibit a 1/f slope (β) within the range of -1.2 to -0.8, reflecting typical neural activity patterns. Galvanic Skin Response (GSR) signals were constrained to display rise times between 0.5 and 5 s, consistent with electrodermal response dynamics. Additionally, Heart Rate (HR) values were bounded between 50 and 120 beats per minute to reflect realistic cardiovascular function.

Any generated samples that violated these biophysically grounded constraints were automatically flagged, rejected, and regenerated to ensure that only physiologically plausible synthetic data were incorporated into the augmented training set.

6.1.4 Data Post-Processing

Generated samples underwent appropriate post-processing transformations to match the input requirements of downstream classifiers. Spectrograms were inverted to time-series signals using the Griffin-Lim algorithm with 64 iterations, enabling their use in models requiring temporal EEG data. Conversely, vectorized feature representations were flexibly remapped either to spectrograms or tokenized input sequences, depending on the specific architecture and input format expected by the target classification model, ensuring seamless integration of synthetic data across different modeling approaches.

6.2 Augmentation Pipelines and Architecture-Specific Details

This subsection formalizes the augmentation workflow applied uniformly across all evaluated methods including the six proposed LLM–GAN synergy architectures and the two baseline systems (GAN-only and LLM-only). The primary objective is to enforce strict experimental parity, ensuring that any observed performance differentials arise solely from architectural design choices rather than discrepancies in data handling, validation rigor, or downstream integration procedures. All augmentation experiments follow an identical, fixed pipeline to ensure fair comparison across architectures (synergy, GAN-only, LLM-only):

1. **Seed Dataset and Modalities**: To ensure fair and controlled comparisons across all methods, a consistent experimental foundation was established. The DEAP dataset served as the exclusive data source for all approaches, with no supplementary external datasets introduced. All architectures utilized identical multimodal inputs to ensure performance variations were attributable solely to model design. Furthermore, no method benefited from domain-specific pretraining or external priors beyond those permitted across all experimental conditions. This maintained strict parity in the resources accessible to each model, enabling an unbiased evaluation of their intrinsic capabilities.
2. **Generation/Augmentation Phase**: During the generation and augmentation phase, all methods produced synthetic samples conditioned on identical DEAP dataset splits and sampling budgets to ensure comparability. For synergy architectures, LLM outputs including reasoning traces, physiological constraints, semantic plans, and cross-modal alignments were utilized exclusively as conditioning signals to guide the generation process rather than as direct outputs.

Baseline comparisons included GAN-only approaches that employed unconditional or weakly conditioned generation without semantic guidance, and LLM-only methods that relied on prompt-driven generation without the benefits of adversarial learning.

3. **Statistical Fidelity Validation (Distributional Level)**: To validate the statistical fidelity of generated samples at the distributional level, a comprehensive suite of metrics was uniformly applied across all augmented datasets. The Wasserstein distance quantified the divergence between synthetic and real data distributions, while the Kolmogorov–Smirnov (K-S) test assessed whether generated samples were drawn from the same underlying distribution as authentic DEAP data. Additional analyses evaluated distribution overlap and variance preservation to ensure that synthetic data captured the full range and variability of the original dataset. These rigorous distributional checks verified that augmented samples maintained consistency with the global DEAP distribution. Furthermore, they confirmed that the generation methods avoided common pitfalls such as mode collapse, thereby ensuring high quality and diversity across all experimental conditions.
4. **Generative Quality and Convergence Metrics**: Generative quality and convergence were evaluated using a standardized set of architecture-agnostic metrics applied uniformly across all methods. The Fréchet Inception Distance (FID) provided a quantitative measure of the perceptual similarity between generated and real data distributions, with fixed acceptance thresholds established to ensure consistent quality standards. Convergence stability was assessed to verify that generation processes reached steady-state performance without oscillation or divergence. For image-based modalities, CLIP-based semantic alignment scores were computed to evaluate how well generated visual content preserved meaningful semantic relationships with corresponding conditioning inputs. By employing fixed, method-independent thresholds for these metrics, the evaluation framework ensured that all architectures were held to identical quality standards, enabling fair and objective comparison of their generative capabilities.
5. **Domain-Grounded Medical Plausibility Checks**: Beyond statistical metrics, all generated samples underwent rigorous domain-grounded medical plausibility validation to ensure clinical and perceptual coherence across modalities. Physiological signals were verified against established medical norms, including acceptable ranges for heart rate, EEG spectral characteristics, and electrodermal response dynamics. Video and audio stimuli were evaluated against structural rules governing affective content to ensure they maintained realistic emotional valence and arousal patterns. Facial expressions and image data were assessed for anatomical correctness and perceptual plausibility, verifying that generated visual content adhered to human physiological constraints. Critically, these plausibility criteria were implemented as hard constraints rather than soft optimization objectives; samples failing to meet these standards were rejected outright and regenerated, ensuring that only medically and perceptually valid synthetic data contributed to the augmented training sets across all experimental conditions.
6. **Human Expert Review and Closed-Loop Refinement**: All architectures were evaluated using an identical, standardized, and blinded human expert review

protocol to ensure physiological plausibility and clinical relevance of the generated samples. For each architecture, 520 randomly selected synthetic samples were independently reviewed by a multidisciplinary clinical panel comprising one cardiologist, three psychiatrists, one ophthalmologist, two radiologists, and three general practitioners. Experts were blinded to the originating architecture and assessed samples solely based on physiological coherence, modality consistency, and clinical plausibility.

7. **Downstream Integration and Stress Classification**: After validation, synthetic data were integrated with the DEAP dataset using identical mixing ratios to assess their impact on stress classification and generalization to unseen models. Two classifiers were evaluated: an LLM-based model using expert prompts for reasoning-based inference, and a multimodal neural model with modality-specific encoders 1D-CNN/TCN for physiological signals (128-D), 2D CNN for audio (128-D), ResNet-18 for video (256-D) and facial images (128-D). Features were fused into a 512-D vector and passed through a multi-layer perceptron [512, 256, 128, 64, 1] with layer normalization, dropout, and ReLU. A compact baseline projected each modality to 16-D and used a 5-layer MLP with 16 neurons per layer to compare performance across capacities.
8. **Cross-Architecture Robustness Testing**: To rigorously assess the generalization capability and cross-compatibility of generated data, comprehensive cross-architecture robustness testing was conducted. Synthetic samples produced by one generative architecture were systematically appended to training sets for classifiers built on different architectural paradigms. For example, data generated through cross-modal attention mechanisms were integrated with hybrid synergy classifiers, and vice versa. The resulting classification performance consistency across these mismatched generator-classifier pairings served as a stringent stress test, revealing whether synthetic data quality was architecture-specific or genuinely captured the underlying data distribution in a generalizable manner.

All augmentation pipelines generate synthetic trials to achieve balanced classes within each training fold. To ensure experimental parity, every method produces an identical number of synthetic samples per fold under the Leave-One-Subject-Out LOSO protocol. Architectural descriptions are therefore accompanied by clearly defined training configurations and explicit controls designed to mitigate overfitting on the limited DEAP data regime.

A unified and meticulously controlled training configuration was implemented across all data augmentation methods to ensure fair comparison. GAN-based approaches employed conditional GANs (cGANs) with a least-squares adversarial objective, incorporating spectral normalization in the discriminator to enhance training stability on limited physiological datasets. All models utilized the Adam optimizer with standard hyperparameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and weight decay of 1×10^{-5}, while early stopping was enforced based on validation loss with a patience window of 50 epochs, strictly using training folds to prevent information leakage. Regularization strategies included module-specific dropout rates ranging

from 0.25 to 0.4, with spectral normalization applied to GAN discriminators where architecturally appropriate. Training stability was further reinforced through gradient clipping at a norm of 1.0 and mixed-precision training to maintain numerical stability while improving memory efficiency. For LLM-based methods, GPT-4.1 was fine-tuned using Low-Rank Adaptation (LoRA) with adapters of rank $r = 32$ applied selectively to key projection matrices, significantly reducing the number of trainable parameters and mitigating overfitting risks inherent to fine-tuning LLMs on the relatively modest DEAP dataset.

6.3 Experimental Results

This section presents the experimental results obtained from our study on LLM-GAN synergy architectures applied to stress classification and data augmentation. The objective is to evaluate and compare the performance of eight distinct configurations, ranging from baseline approaches (GAN-only and LLM-only) to advanced architectures integrating bidirectional guidance mechanisms, multi-modal attention, and CoT reasoning. The experiments were conducted on the DEAP dataset (32 subjects, 40 trials), enriched with 6,000 synthetic samples per architecture. The results are analyzed from three complementary perspectives:

- **Classification performance and distributional fidelity** (Table 6): evaluation of accuracy and synthetic data quality metrics (Wasserstein distance, Kolmogorov–Smirnov statistic, FID, CLIP semantic alignment)
- **Cross-classifier performance** (Table 7): comparison of performance across full neural classification architectures versus compact architectures.
- **Cross-architecture robustness** (Table 8): analysis of model generalization trained on different augmentation configurations.

  The following tables detail the quantitative results of the different architectures.

- **The computational infrastructure**: Given the moderate size of the DEAP dataset and the addition of synthetic samples per architecture, the computational demands of the LLM–GAN synergy experiments remain manageable on a high-end workstation (NVIDIA H100 GPU, 128 GB RAM, and 1 TB NVMe SSD), which served as our reference deployment setup. Standalone GAN training requires approximately 10–12 GB of GPU memory, with 4–6 min per epoch over 50–80 epochs, totaling 3–6 h. LLM-only augmentation, using medium-sized models (100–300 M parameters), consumes 12–16 GB VRAM, 5–8 min per epoch across 50 epochs, resulting in 4–6 h of training. Architectures integrating LLM guidance with adversarial generation, such as Direct Adversarial or Bidirectional models, require 16–24 GB memory and 6–10 min per epoch for 80–100 epochs, totaling 6–12 h. More complex designs incorporating hierarchical, cross-modal attention, and CoT reasoning use 24–32 GB VRAM and 8–12 min per epoch over 100–120 epochs, yielding 8–18 h. The full Hybrid architecture, combining all

Table 6 Baseline and LLM-GAN synergy architecture performance: LLM-based stress classification model and data augmentation Results

Augmentation method	Accuracy %	Wasserstein (mean)	K-S (mean)	DOI[1]	VP[2]	FID (mean)	CSI[3]	**CLIP SA** [4]
GAN-only	79.4 ± 2.1	0.42 ± 0.05	0.38 ± 0.04	0,68	0,71	52.3 ± 4.2	0,73	0.61
LLM-only	81.7 ± 1.8	0.38 ± 0.04	0.35 ± 0.03	0,70	0,74	48.6 ± 3.8	0,77	0.64
Direct Adv	88.3 ± 1.5	0.26 ± 0.03	0.23 ± 0.03	0,78	0.79	38.7 ± 3.1	0,82	0.73
Bidirectional	90.8 ± 1.4	0.23 ± 0.03	0.19 ± 0.02	0,81	0,83	34.2 ± 2.9	0,84	0.77
Hierarchical	91.2 ± 1.6	0.21 ± 0.03	0.18 ± 0.02	0,82	0,86	32.8 ± 2.7	0,83	0.79
Cross-modal	92.5 ± 1.3	0.22 ± 0.02	0.17 ± 0.02	0,83	0.84	33.5 ± 2.8	0,81	0,85
CoT	93.9 ± 1.2	0.19 ± 0.02	0.16 ± 0.02	0,85	0,85	30.1 ± 2.4	0,87	0,88
Hybrid	95.3 ± 1.1	0.17 ± 0.02	0.14 ± 0.02	0,87	0,87	27.4 ± 2.1	0,89	0,90

[1] **Distribution Overlap Index (DOI)** is reported as a normalized composite distributional fidelity score derived from the Wasserstein distance, K–S statistic, and variance preservation across all features, with values in [1, 1]

[2] **Variance Preservation (VP)** is measured as the ratio of synthetic-to-real feature variance averaged across modalities, normalized to [0,1]

[3] **Convergence Stability Index (CSI)** is computed as the normalized inverse variance of the training loss after convergence, scaled to [0,1]

[4] **CLIP Semantic Alignment (CLIP SA)** is reported as the mean cosine similarity between generated samples and their corresponding conditioning prompts

Table 7 Baseline and LLM-GAN synergy architecture performance across classifier types

Augmentation method	Accuracy %	
	Multimodal neural (Full fusion – 512-dim)	Compact fusion (16-dim projection + 5-layer MLP)
None	76.2 ± 2.4	72.4 ± 2.8
GAN-only	78.8 ± 2.2	74.9 ± 2.5
LLM-only	80.5 ± 2.0	76.8 ± 2.3
Direct Adv	86.7 ± 1.7	83.2 ± 1.9
Bidirectional	89.1 ± 1.5	85.4 ± 1.7
Hierarchical	89.6 ± 1.6	86.1 ± 1.8
Cross-modal	91.3 ± 1.4	87.5 ± 1.6
CoT	92.8 ± 1.3	89.3 ± 1.5
Hybrid	94.1 ± 1.2	90.7 ± 1.4

synergy mechanisms, requires 32–40 GB VRAM and 10–15 min per epoch for 120–150 epochs, corresponding to roughly 15–20 h. Overall, the high-throughput H100 GPU ensures that even with the expanded dataset, all architectures can be trained efficiently, making the experiments feasible within a single to two days of continuous computation.

Table 8 Cross-architecture robustness testing—LLM-based classification accuracy (%)

	GAN-only	LLM-only	Direct Adv	Bidirectional	Hierarchical	Cross-modal	CoT	Hybrid
GAN-only	79.4	78.8	79.1	78.6	79.2	78.3	79.0	78.9
LLM-only	80.9	81.7	81.3	81.5	81.8	81.1	81.4	81.6
Direct Adv	87.2	87.8	88.3	87.9	88.1	87.4	87.6	88.0
Bidirectional	89.5	90.1	90.4	90.8	90.6	89.8	90.2	90.5
Hierarchical	90.2	90.7	91.0	91.3	91.2	90.4	90.8	91.1
Cross-modal	91.3	91.9	92.1	92.4	92.3	92.5	91.8	92.3
CoT	92.6	93.2	93.4	93.7	93.5	93.1	93.9	93.6
Hybrid	94.1	94.7	94.9	95.1	95.0	94.5	94.8	95.3

6.4 Comparative Analysis of LLM-GAN Synergy Architectures: Experimental Results

This section presents a comprehensive evaluation of the proposed LLM-GAN synergy architectures for stress classification and multimodal physiological data augmentation. These synergies are built upon the methodological framework established in previous sections.

6.4.1 Classification Performance and Distributional Fidelity

The results in Table 6 demonstrate that systematic architectural integration of LLM semantic guidance with adversarial learning yields consistent improvements across classification accuracy, distributional fidelity, semantic alignment, and training stability. While standalone GAN and LLM-based augmentation methods achieve modest performance (79.4% and 81.7% accuracy, respectively), direct adversarial coupling produces a substantial improvement to 88.3%, confirming that LLM-derived semantic constraints effectively guide GAN generation toward task-relevant feature distributions.

Progressive architectural refinements reveal complementary optimization strategies across the synergy architectures. Bidirectional interaction improves information flow between modalities (90.8%), establishing reciprocal conditioning pathways.

Hierarchical decomposition demonstrates superior variance preservation (VP = 0.86) through multi-scale feature modeling, maintaining the full statistical diversity of physiological signals. Notably, cross-modal attention achieves the highest semantic alignment among intermediate architectures (CLIP SA = 0.85) by explicitly modeling inter-modal dependencies, though with a slight trade-off in convergence stability (CSI = 0.81). This architectural diversity demonstrates that different synergy mechanisms optimize distinct aspects of data quality distributional diversity, semantic coherence, or training stability depending on the coupling strategy employed.

CoT conditioning provides substantial gains (93.9%), with reduced distributional divergence (Wasserstein = 0.19, K-S = 0.16) and improved DOI (0.85), indicating that explicit reasoning supervision enhances both the discriminative quality and statistical consistency of generated samples. The reasoning traces guide the generation process toward physiologically plausible and task-relevant feature combinations.

The full Hybrid architecture achieves state-of-the-art performance (95.3 $\pm$ 1.1% accuracy), representing a 15.9% improvement over GAN-only baseline and 13.6 points over LLM-only approaches. This architecture attains the lowest FID (27.4 $\pm$ 2.1), indicating superior perceptual quality, alongside the highest CLIP SAt (0.90), confirming strong correspondence between generated samples and their conditioning contexts. Critically, the Hybrid approach successfully balances distributional fidelity (Wasserstein = 0.17, K-S = 0.14), VP (0.87), and CSI (0.89),

avoiding the architectural trade-offs observed in specialized methods. The convergence of all quality metrics demonstrates that hierarchical, reasoning-aware LLM–GAN coupling enables generation of synthetic physiological data that is simultaneously statistically realistic, semantically coherent, and clinically relevant for stress classification in healthcare applications.

6.4.2 Cross-Classifier Performance

The results in Table 7 demonstrate the efficiency and generalization of LLM–GAN-augmented data across traditional ML classifiers that were not previously trained on this dataset. Both the multimodal neural and compact fusion models benefit from progressive augmentation strategies, with accuracy improving consistently from standalone GAN or LLM-based augmentation to synergistic architectures. Hybrid augmentation achieves the highest performance (94.1 and 90.7%) indicating that they are reasoning-aware. Hierarchical LLM–GAN synergy produces synthetic samples that are highly informative and transferable across different model architectures.

These findings confirm that the proposed augmentation methods effectively enhance data utility and enable traditional ML models to leverage additional discriminative features encoded by the LLM–GAN framework, without altering the underlying dataset distribution.

6.4.3 Cross-Architecture Robustness

The results in Table 8 shows that hybrid LLM architectures consistently achieve the highest classification accuracy, outperforming simpler GAN-only or LLM-only models. Accuracy remains stable across different cross-architecture tests, indicating strong robustness, while incorporating reasoning (CoT) and cross-modal strategies provides measurable improvements over baseline models. Overall, combining multiple strategies yields the most reliable and accurate performance. The results demonstrate that integrating LLM semantic guidance with GAN-based augmentation substantially improves classification accuracy, distributional fidelity, semantic alignment, and training stability. While standalone GAN or LLM-based augmentation provides modest gains, progressively integrated architectures including Direct Adversarial, Bidirectional, Hierarchical, and Cross-modal further enhance performance and feature consistency. The full Hybrid architecture achieves the highest accuracy and optimal metrics across FID, CLIP alignment, variance preservation, and convergence stability, producing synthetic data that is realistic, semantically coherent, and clinically relevant. Importantly, the augmented data generalizes efficiently to traditional ML classifiers, with hybrid augmentation reaching 94.1% and 90.7% accuracy for multimodal neural and compact fusion models, respectively. These findings confirm that LLM–GAN synergy provides robust, transferable, and highly informative augmentation for low-data multimodal physiological tasks.

To conclude, this study shows that LLM–GAN synergy consistently outperforms standalone GAN and LLM augmentation, particularly in low-data, multimodal physiological settings. Across all evaluation metrics, progressively integrated architectures outperform isolated generators, with the Hybrid model achieving the most balanced and robust results. These findings demonstrate the progressive superiority of hybrid architectures integrating advanced LLM-GAN synergy mechanisms.

Building on this empirical validation, Sect. 7 examines practical trade-offs of the synergy in controllability, scalability, and clinical deployment.

Having established the empirical validity of the proposed LLM–GAN synergy architectures through controlled quantitative, qualitative, and robustness evaluations, now turning to a structured discussion of their advantages and drawbacks.

7 Advantages and Drawbacks of LLM-GAN Synergy

The fusion of GANs and LLMs marks a transformative leap in the evolution of multimodal AI systems, with profound implications for healthcare innovation [97]. This section highlights the advantages of this synergy while acknowledging that such systems are not without limitations, including challenges related to data quality, model bias, interpretability, and clinical validation.

7.1 Advantages of GAN-LLM Synergy

The combination between the two models enables systems to process and generate content across multiple data modalities simultaneously [30]. To more comprehensively delineate the advantages of this synergy, we categorize integration into three hierarchical abstraction levels: data-level, as described in [119], clinical decision-level, as mentioned in [32], and system-level, as outlined in [4]. Each level represents a distinct scope of impact within healthcare AI, reflecting the progressive influence of integration from raw data processing to strategic system-wide applications.

1. Data-Level Integration (Synthetic Data Generation & Privacy Preservation)

At the foundational level, LLMs guide GANs to produce synthetic multimodal medical datasets combining diverse data types that are both clinically valid and statistically balanced [71]. This directly addresses the scarcity of annotated datasets, reduces bias across patient demographics and rare conditions, and provides privacy-preserving alternatives to sensitive patient data. Such synthetic datasets enable the training of more robust diagnostic and predictive models, ensuring equitable performance across diverse clinical populations [153]. Empirically, this strategy has demonstrated tangible clinical gains, such as improving liver-lesion detection sensitivity from 78.6 to 85.7% through GAN-generated CT augmentation [26], and achieving 6–33% diagnostic accuracy improvements in chest X-ray analysis when synthetic

data support the integration of text, tabular, and image modalities [119]. Beyond performance, such synthetic datasets enable privacy-preserving training of robust multimodal models.

2. Clinical Decision-Level Integration (Reasoning + Generative Visualization)

LLMs interpret longitudinal patient records and integrate modalities to guide GANs in generating outputs such as disease progression maps, treatment-response simulations, or reconstructed diagnostic images [71]. This process also enables privacy-preserving Synthetic Health Records that mirror real patient data, easing data-sharing constraints, mitigating class imbalance, and supporting research where direct EHR access is limited.

3. System-Level Integration and Orchestration (Adaptive, Explainable Multimodal AI Frameworks)

At the highest level, integration evolves into an orchestration layer where LLMs, GANs, and related models interact dynamically to form a cohesive system. LLMs act as controllers (Belcik and Stryker 2025), directing generative modules while providing hierarchical explanations. This orchestration not only enables continuous adaptation and transparent alignment with clinical workflows but also supports improved multimodal predictions: joint LLM–GAN systems can reconstruct missing scans or generate longitudinal forecasts from mixed data, with multimodal inputs shown to increase AI accuracy by 1.2–27.7% across tasks such as mortality or diagnosis prediction [119].

Despite these advances, the integration of LLMs and GANs in multimodal healthcare also introduces several drawbacks and challenges. In the next section, these limitations will be critically examined to ensure safe, reliable, and clinically applicable deployment.

7.2 GAN-LLM Synergy Drawbacks

Despite the remarkable progress and immense potential of the LLM-GAN synergy in multimodal AI healthcare, several weaknesses are raised [140]. Addressing these will be crucial for the continued advancement and responsible deployment of these powerful systems. These limitations may be analyzed along two hierarchical levels: the first encompassing technical and computational issues, and the second addressing ethical, privacy, and societal concerns.

7.2.1 Technical Challenges

Despite the transformative potential of multimodal AI systems, their practical deployment is hindered by a number of persistent technical challenges. These challenges span computational, methodological, and operational dimensions, reflecting

the inherent complexity of integrating heterogeneous modalities into a unified framework:

- **Computational Complexity and Resource Intensity** Multimodal GAN–LLM hybrids are inherently large and resource-intensive, requiring substantial computational power and storage for training and inference. For example, training a generative model for clinical notes consumed 560 A100 GPUs over 20 days [71], highlighting both high operational costs and energy demands. These challenges can limit scalability in resource-constrained environments, making efficiency optimization without performance loss a critical area of ongoing research [127].
- **Data Alignment and Synchronization** represent another complex challenge. Effectively integrating and synchronizing disparate data types from different modalities requires sophisticated algorithms for cross-modal alignment and reasoning. Achieving seamless coherence across modalities, especially when data comes from different time points or sources, is non-trivial.
- **Robustness and Generalization** Integrating and synchronizing heterogeneous inputs such as images, text, and time-series from different modalities remains a major challenge. Achieving coherent cross-modal representations, particularly when data are incomplete, missing, or collected at different time points, requires sophisticated alignment algorithms [51, 142]. Without such handling, multimodal models often fail or degrade in real-world clinical workflows.
- **Robustness and Generalization** are critical concerns. LLMs can be susceptible to variations in prompt wording [63], and generative models, in general, face the challenge of producing realistic and unbiased outputs that accurately reflect the true underlying data distribution.
- The **problem of missing modalities** is also significant. In real-world applications, data samples may lack certain modalities due to sensor limitations, cost constraints, privacy concerns, or data loss. Developing Multimodal Learning with Missing Modality (MLMM) techniques that allow models to function effectively even with incomplete information is essential for robust deployment [150].

7.2.2 Ethical and Societal Considerations

The deployment of powerful multimodal AI healthcare systems necessitates careful consideration of their ethical and societal implications:

- **Bias and Fairness** are paramount concerns in multimodal AI. While integrating multiple modalities can help cross-verify information and reduce biases present in unimodal systems, LLM–GAN architectures can inadvertently introduce or amplify new biases if not carefully managed [126]. Ensuring fairness requires diverse training datasets, continuous calibration, and systematic monitoring, alongside audit protocols and model governance mechanisms aligned with regulatory frameworks such as GDPR, FDA, and EMA guidelines. For example, in a radiology study, LLM accuracy varied from 95.3 to 99.6% depending on prompts

[138], illustrating how prompt design and training instability can propagate errors and disproportionately affect minority patient groups. Embedding traceable decision logs, bias assessment metrics, and human-in-the-loop oversight are recommended to mitigate these risks and support responsible deployment in clinical settings.

- **Data Privacy and Security** [126] become more complex with multimodal systems. Handling multiple types of data, especially sensitive information like medical images combined with patient records, significantly increases the risk of data breaches and privacy issues. Adherence to stringent security protocols and compliance with regulations (like Health Insurance Portability and Accountability Act) is critical.
- **Misinformation and deepfakes** represents a serious ethical challenge. Multimodal AI can create highly realistic fake content, including counterfeit videos and fabricated news articles, making it difficult or impossible for humans to distinguish from authentic media [140]. This raises profound questions about trust, authenticity, and the potential for malicious use. As noted by Rohit Anabheri, member of the Forbes Technology Council, ensuring responsible deployment of AI requires balancing innovation with ethical safeguards to maintain trust and authenticity [8].

7.2.3 Regulatory Considerations for Clinical Deployment

LLM–GAN systems intended for clinical use are regulated as **Software-as-a-Medical-Device (SaMD)**. In the United States, the Food and Drug Administration (FDA) applies risk-based approval pathways (510(k), De Novo, PMA) and requires **Total Product Lifecycle** oversight for AI/ML-enabled medical devices [135]. Current FDA guidance mandates **Good Machine Learning Practice** [134], including rigorous clinical validation, predefined update mechanisms, and **Predetermined Change Control Plans** to govern adaptive or continuously learning models.

In the European Union, the **AI Act (2024)** classifies AI-driven diagnostic and clinical decision-support systems as **high-risk AI**, imposing requirements for risk management, high-quality and representative training data, transparency, technical documentation, and mandatory human oversight [34]. These requirements operate alongside **MDR/IVDR** medical device regulations. General Data Protection Regulation further constrains deployment by enforcing lawful data processing, privacy-by-design, and prohibiting fully automated clinical decisions without human review [35].

Regulatory classification frameworks such as the **International Medical Device Regulators Forum** (**IMDRF**) **SaMD risk model** assign higher risk to systems that directly inform diagnosis or treatment, particularly for serious or chronic conditions [135]. Consequently, LLM–GAN systems in such settings are subject to stringent review comparable to **FDA Class III / EU Class III** devices. Regulators

require robust **model governance**, including multidisciplinary development, dataset representativeness, continuous post-market monitoring, and drift detection [134].

Auditability and traceability are mandatory: development processes, data provenance, model versions, and retraining events must be fully documented and reviewable. Explainability mechanisms and **human-in-the-loop control** are consistently required, with clinicians retaining final decision authority [96]. Emerging deployments such as generative MRI reconstruction (SubtleSYNTH) [123], clinical chat assistants (ChatEHR) [122], and AI-supported mental health tools demonstrate that regulatory acceptance currently favors **restricted-scope, well-governed generative systems** that augment, rather than replace, clinical judgment.

Looking ahead, future horizons lie in the development of more resource-efficient architectures, standardized evaluation frameworks, and interpretable generative models that can ensure clinical trustworthiness. This will be the topic of the following section.

8 Future Horizons

Several promising directions for future research are emerging to address current limitations and unlock the full potential of LLM-GAN synergy.

Further research into enhanced human-like grounding is crucial. While Multimodal Large Language Models (MLLMs) [149] show some sensitivity to implicit sensorimotor features (shape and color), they do not yet fully account for the grounding effects observed in humans, particularly for features like orientation, size, and volume. This suggests a need to explore richer interactions with the world beyond labeled image/caption pairs to imbue AI with a more comprehensive understanding of reality.

The development of advanced multi-agent collaboration systems is a promising avenue. Exploring how different LLM-GAN components can collaboratively tackle complex generative tasks could lead to more robust, creative, and sophisticated multimodal outputs. This involves designing effective communication protocols and task delegation mechanisms between specialized AI agents.

Research into real-time and resource-efficient multimodal generation is vital for widespread adoption. Optimizing architectural designs and training methodologies to achieve lower processing latencies and reduced memory and power consumption will enable the deployment of these powerful systems on a broader range of devices and platforms.

Finally, continuous efforts are needed to improve controllability and explainability in multimodal generative AI. Developing methods for more granular control over generated content and enhancing the transparency of complex multimodal decision-making processes will be essential, particularly for safety–critical applications and to foster greater user trust. Addressing catastrophic forgetting in continual learning for MLLMs, when adapting pre-trained models to dynamic data distributions and new tasks, is also a significant challenge for long-term model viability [149].

Table 9 Future horizon for LLM-GAN synergy in multimodal healthcare

Application domain	LLM role	GAN role	Anticipated clinical impact
Next-gen digital twins	Build comprehensive patient models by synthesizing longitudinal records, genomic profiles, and lifestyle data	Simulate individualized disease trajectories, treatment responses, and recovery outcomes	May support fully personalized therapies, allowing clinicians to "test" interventions virtually before implementation
Cognitive clinical assistants	Function as continuously adaptive conversational agents that evolve with a clinician's expertise and patient context	Generate realistic patient scenarios, counterfactual cases, and rare-event surgical simulations for decision rehearsal	Could transform clinical decision-making into a dynamic, scenario-based process with reduced error rates

After printpointing on the potential use cases of the LLM-GAN synergy in general, we will direct our focus on the healthcare scope. Future extensions may involve reinforcement learning to optimize care pathways in real time, federated learning to integrate distributed hospital data without compromising privacy, and causal inference to strengthen the reliability of multimodal predictions. Our assumptions are more oriented to a powerful two domains that are presented in the table below (Table 9).

9 Conclusion

The convergence of LLMs and GANs represents a pivotal evolution in medical AI, transitioning from systems that merely process information to those capable of simulation, cross-modal reasoning, and generative synthesis. LLM–GAN architectures are no longer experimental explorations; they are emerging as foundational components of a new healthcare paradigm in which clinical language drives the generation of anatomically precise images, physiologically grounded signals, and temporally coherent simulations. This integration enables the synthesis of rare pathological data, modeling of patient trajectories, and personalization of diagnostics, ultimately allowing AI systems to approach medicine not merely as data, but as contextualized, multimodal knowledge reflective of clinical experience.

However, with these capabilities comes substantial responsibility. The increasing autonomy and context-awareness of LLM–GAN systems necessitate deployment strategies that prioritize ethical rigor, transparency, explainability, and trustworthiness. Addressing algorithmic bias, ensuring clinical interpretability, and maintaining alignment with human oversight are no longer ancillary considerations; they are essential design imperatives. The advancement of these technologies implicates not only computational and technical dimensions but also clinical, ethical,

and societal ones, underscoring the importance of aligning AI innovations with the human-centered values of healthcare.

By bridging language-based reasoning with multimodal generative capacity, LLM–GANs are establishing a new medium for medical cognition. They augment human expertise, enhance accessibility to innovative solutions, and extend the boundaries of clinical practice. The trajectory of AI in medicine will therefore be shaped not only by algorithmic sophistication but by the integration of ethical foresight, clinical insight, and societal responsibility, with LLM–GANs positioned to play a transformative role in this evolution.

References

1. Abbasi, M., Santos, B.P., Pereira, T.C., et al.: Designing optimized drug candidates with generative adversarial network. J. Cheminformatics **14**(1), 40 (2022). https://doi.org/10.1186/s13321-022-00623-6
2. Acosta, J.N., Falcone, G.J., Rajpurkar, P., et al.: Multimodal biomedical AI. Nat. Med. **28**(9), 1773–1784 (2022). https://doi.org/10.1038/s41591-022-01981-2
3. Al Nazi, Z., Peng, W.: Large language models in healthcare and medical domain: a review. Informatics **11**(3), 57 (2024). https://doi.org/10.3390/informatics11030057
4. Alami, H.: Understanding the integration of artificial intelligence in healthcare: a socio-technical perspective. J. Med. Internet Res. **26**(1), e11149257 (2024). https://bmchealthservres.biomedcentral.com/articles/https://doi.org/10.1186/s12913-024-11112-x
5. Alarcon, N.: Synthesizing high-resolution images with StyleGAN2. NVIDIA Developer Blog (2020). https://developer.nvidia.com/blog/synthesizing-high-resolution-images-with-stylegan2/
6. Alsentzer, E., Murphy, J., Boag, W., et al.: Publicly available clinical BERT embeddings (2019). arXiv. https://doi.org/10.48550/arXiv.1904.03323
7. Altun Güven, S., Talu, M.F.: Brain MRI high resolution image creation and segmentation with the new GAN method. Biomed. Signal Process. Control. **80**(Part 1), 104246 (2023). https://www.researchgate.net/publication/364606757
8. Anabheri, R.: Balancing AI: How leaders can drive innovation without compromising ethics. Forbes (2024). https://www.forbes.com/councils/forbestechcouncil/
9. Arjovsky, M., Bottou, L.: Towards principled methods for training generative adversarial networks (2017). arXiv. https://arxiv.org/abs/1701.04862
10. Aronson, A.R., Lang, F.M.: An overview of MetaMap: historical perspective and recent advances. J. Am. Med. Inform. Assoc. **17**(3), 229–236 (2010). https://doi.org/10.1136/jamia.2009.002733
11. Artsi, Y., Sorin, V., Glicksberg, B.S., et al.: Advancing clinical practice: the potential of multimodal technology in modern medicine. J. Clin. Med. **13**(20), 6246 (2024). https://doi.org/10.3390/jcm13206246
12. Baltrušaitis, T., Ahuja, C., Morency, L.P.: Multimodal machine learning: a survey and taxonomy. IEEE Trans. Pattern Anal. Mach. Intell. **41**(2), 423–443 (2018). https://doi.org/10.1109/TPAMI.2018.2798607
13. Bansal, H., Lin, Z., Xie, T., et al.: VideoPhy: Evaluating physical commonsense for video generation (2024). arXiv. https://doi.org/10.48550/arxiv.2406.03520
14. Bates, D.W., Saria, S., Ohno-Machado, et al.: Big data in health care: using analytics to identify and manage high-risk and high-cost patients. Health Aff. **33**(7), 1123–1131 (2014). https://doi.org/10.1377/hlthaff.2014.0041

15. Bazoge, A., Morin, E., Daille, B., Gourraud, P.A.: Applying natural language processing to textual data from clinical data warehouses: systematic review. JMIR Med. Inform. **11**, e42477 (2023). https://doi.org/10.2196/42477
16. Belcic, I., Stryker, C.: What is AI orchestration? IBM (2025). https://www.ibm.com/think/topics/ai-orchestration
17. Bodonhelyi, A., Stegemann-Philipps, C., Sonanini, A., et al.: Modeling challenging patient interactions: LLMs for medical communication training (2025). arXiv. https://doi.org/10.48550/arXiv.2503.22250.
18. Bolton, E., Hall, D., Yasunaga, M., et al.: Stanford CRFM introduces Pubmedgpt 2.7 b (2022). https://hai.stanford.edu/news/stanford-crfm-introduces-pubmedgpt-27b.
19. Brlek, P., Bulić, L., Bračić, M., et al.: Implementing whole genome sequencing (WGS) in clinical practice: Advantages, challenges, and future perspectives. Cells **13**, 60504 (2024). https://doi.org/10.3390/cells13060504
20. Brock, A., Donahue, J., Simonyan, K.: Large scale GAN training for high fidelity natural image synthesis (2018). arXiv. https://arxiv.org/abs/1809.11096
21. Brown, T.B., Mann, B., Ryder, N., et al.: Language models are few-shot learners (2020). arXiv. https://doi.org/10.48550/arxiv.2005.14165
22. Cao, Xi, et al.: Human-in-the-Loop Generation of Adversarial Texts: A Case Study on Tibetan Script (2024). arXiv:2412.12478v3.https://arxiv.org/abs/2412.12478v3
23. Chen, M., Lu, C., Mahmud, T.: Generative AI in healthcare: opportunities, challenges, and future directions. Patterns **4**(9), 100842 (2023). https://doi.org/10.1016/j.patter.2023.100842
24. Choi, S.R., Lee, M.: Transformer architecture and attention mechanisms in genome data analysis: a comprehensive review. Biology **12**(7), 1033 (2023). https://doi.org/10.3390/biology12071033
25. Chorowski, J.K., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y.: Attention-based models for speech recognition. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.), Advances in Neural Information Processing Systems, vol. 28. Curran Associates, Inc. (2015). https://proceedings.neurips.cc/paper_files/paper/2015/file/1068c6e4c8051cfd4e9ea8072e3189e2-Paper.pdf
26. Cywiński, D.: Creating synthetic training data for healthcare AI: what to do when you don't have big datasets. TheMomentum.ai (2025). https://www.themomentum.ai/blog/creating-synthetic-training-data-for-healthcare-ai
27. Dalmaz, O., Yurt, M., Çukur, T.: ResViT: Residual vision transformers for multi-modal medical image synthesis (2021). arXiv. https://ieeexplore.ieee.org/document/9758823?denied=
28. Dennstädt, F., Hastings, J., Putora, P.M., et al.: Implementing large language models in healthcare while balancing control, collaboration, costs and security. NPJ Digit. Med. **8**(1), 143 (2025). https://doi.org/10.1038/s41746-025-01476-7
29. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL 2019, pp. 4171–4186 (2019). https://doi.org/10.18653/v1/N19-1423
30. Dontineni, M.R.R.: Developing an integrated LLM-GAN feedback loop architecture for complex network issue resolution. Eur. J. Comput. Sci. Inf. Technol. **13**(24), 39–50 (2025). https://doi.org/10.37745/ejcsit.2013/vol13n243950
31. Efthymiou, E.: Integrating digital and narrative medicine in modern healthcare: a systematic review. Med. Educ. Online **30**(1), 2475979 (2025). https://doi.org/10.1080/10872981.2025.2475979
32. Elhaddad, M., Hamam, S.: AI-Driven clinical decision support systems: an ongoing pursuit of potential. Cureus **16**(4), e57728 (2024). https://doi.org/10.7759/cureus.57728
33. Elhanashi, A., Saponara, S., Zheng, et al.: AI-powered object detection in radiology: Current models, challenges, and future direction. J. Imaging **11**(5), 141 (2025). https://doi.org/10.3390/jimaging11050141
34. European Commission: Artificial Intelligence in healthcare, Public Health—eHealth (2025). https://health.ec.europa.eu/ehealth-digital-health-and-care/artificial-intelligence-healthcare_en

35. European Parliament & Council of the European Union: Regulation (EU) 2016/679 of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data. Off. J. Eur. Union (2016). https://eur-lex.europa.eu/eli/reg/2016/679/oj
36. Farnia, F., Ozdaglar, A.: Do GANs always have Nash equilibria? In: Proceedings of the 37th International Conference on Machine Learning, in Proceedings of Machine Learning Research 119:3029–3039 (2020). https://proceedings.mlr.press/v119/farnia20a.html.
37. Frid-Adar, M., Diamant, I., Klang, E., et al.: GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. Neurocomputing **321**, 321–331 (2018). https://doi.org/10.1016/j.neucom.2018.09.013
38. Friedman, C., Alderson, P.O., Austin, J.H., et al.: A general natural-language text processor for clinical radiology. J. Am. Med. Inform. Assoc. **1**(2), 161–174 (1994). https://doi.org/10.1136/jamia.1994.95236145
39. Friedman, C., Hripcsak, G., Shagina, L., Liu, H.: Representing information in patient reports using natural language processing and the extensible markup language. J. Am. Med. Inform. Assoc. **6**(1), 76–87 (1999). https://doi.org/10.1136/jamia.1999.0060076
40. Gaber, F., Shaik, M., Allega, F., et al.: Evaluating large language model workflows in clinical decision support for triage and referral and diagnosis. NPJ Digit. Med. **8**(263) (2025). https://doi.org/10.1038/s41746-025-01684-1
41. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., et al.: Generative adversarial nets. Adv. Neural Inf. Process. Syst. **27** (NIPS 2014). https://papers.nips.cc/paper/5423-generative-adversarial-nets
42. Goyal, M.: Synthetic data revolutionizes rare disease research: How large language models and generative AI are overcoming data scarcity and privacy challenges. Int. J. Recent. Innov. Trends Comput. Commun. **11**(11), 11411 (2023). https://doi.org/10.17762/ijritcc.v11i11.11411
43. Gulrajani, I., Ahmed, F., Arjovsky, M., et al.: Improved training of Wasserstein GANs. In: Advances in Neural Information Processing Systems 30 (NIPS 2017) (2017). https://arxiv.org/abs/1704.00028
44. Hao, Y., Cheng, C., Li, J., et al.: Multimodal integration in health care: development with applications in disease management. J. Med. Internet Res. **27**, e76557 (2025). https://doi.org/10.2196/76557
45. Heilmeyer, F., Böhringer, D., Reinhard, T., et al.: Viability of open large language models for clinical documentation in German health care: real-world model evaluation study. JMIR Med. Inform. **12**, e59617 (2024). https://doi.org/10.2196/59617
46. Hemphill, L., Ma, S.: A scoping review of using large language models (LLMs) to investigate electronic health records (EHRs) (2024). arXiv. https://arxiv.org/abs/2405.03066
47. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models (2020) ArXiv. https://doi.org/10.48550/arXiv.2006.11239
48. Hosny, A., Parmar, C., Quackenbush, et al.: Artificial intelligence in radiology. Nat. Rev. Cancer **18**(8), 500–510 (2018). https://doi.org/10.1038/s41568-018-0016-5
49. Huang, S.C., Pareek, A., Seyyedi, S., et al.: Fusion of medical imaging and electronic health records using deep learning: a systematic review and implementation guidelines. NPJ Digit. Med. **3**(1), 136 (2020). https://www.nature.com/articles/s41746-020-00341-z
50. IBM: What is AI orchestration? (n.d.) https://www.ibm.com/think/topics/ai-orchestration
51. IBM: Multimodal AI. IBM Think (2024). https://www.ibm.com/think/topics/multimodal-ai
52. Jandoubi, B.: Multimodal artificial intelligence in medical diagnostics. Information **16**(7), 591 (2025). https://doi.org/10.3390/info16070591
53. Jeyantha, S., Juliet, J., David, D.J., et al.: Predictive models for early detection of Parkinson's disease: A machine learning approach. J. Mech. Contin. Math. Sci. (2025). https://doi.org/10.26782/jmcms.2025.04.00009
54. Jiang, L.Y., Liu, X.C., Pour Nejatian, N., et al.: Health system-scale language models are all-purpose prediction engines. Nature **619**(7969), 357–362 (2023). https://doi.org/10.1038/s41586-023-06160-y

55. Johnson, A., et al.: Leveraging generative AI and large language models: a comprehensive roadmap for healthcare integration. PMC Article PMC10606429 (2024). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10606429
56. Kamran, S.A., Hossain, K.F., Tavakkoli, A., et al.: VTGAN: Semi-supervised retinal image synthesis and disease prediction using vision transformers (2021). https://openaccess.thecvf.com/content/ICCV2021W/CVAMD/html/Kamran_VTGAN_Semi-Supervised_Retinal_Image_Synthesis_and_Disease_Prediction_Using_Vision_ICCVW_2021_paper.html
57. Karpakam, S., Kumareshan, N.: A multi-modality framework for precise brain tumor detection and multi-class classification using hybrid GAN approach. Biomed. Signal Process. Control **104**, 107559 (2025). https://doi.org/10.1016/j.bspc.2025.107559
58. Koelstra, S., Mühl, C., Soleymani, M., Lee, J.-S., Yazdani, A., Ebrahimi, T., Pun, T., Nijholt, A., Patras, I.: DEAP: a database for emotion analysis using physiological signals. IEEE Trans. Affect. Comput. **3**(1), 18–31. https://doi.org/10.1109/T-AFFC.2011.15 (2012)
59. Koutsouleris, N., Dwyer, D.B., Degenhardt, F., et al.: Multimodal machine learning workflows for prediction of psychosis in patients with clinical high-risk syndromes and recent-onset depression. JAMA Psychiatry **78**(2), 123–134 (2020). https://pubmed.ncbi.nlm.nih.gov/33263726/
60. Kung, T.H., Cheatham, M., Medenilla, A., et al.: Performance of ChatGPT on USMLE: Potential for AI-assisted medical education using large language models. PLOS Digital Health **2**(2), e0000198 (2023). https://doi.org/10.1371/journal.pdig.0000198
61. Lahat, D., Adalı, T., Jutten, C.: Multimodal data fusion: an overview of methods, challenges and prospects. Proc. IEEE **103**(9), 1449–1477 (2015). https://doi.org/10.1109/JPROC.2015.2460697
62. Lee, J., Yoon, W., Kim, S., et al.: BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics **36**(4), 1234–1240 (2020). https://doi.org/10.1093/bioinformatics/btz682
63. Li, X., Zhao, C., Zhao, H., Wu, L., He, M.: GANPrompt: enhancing robustness in LLM-based recommendations with GAN-enhanced diversity prompts (2024) ArXiv. https://arxiv.org/abs/2408.09671
64. Li, J., Mu, Y.: Searching for best practices in medical transcription with large language models (2024). arXiv. https://doi.org/10.48550/arxiv.2410.03797
65. Li, L., Zhou, J., Gao, Z., et al.: A scoping review of using large language models (LLMs) to investigate electronic health records (EHRs) (2024). arXiv. https://doi.org/10.48550/arxiv.2405.03066
66. Li, X., Zhao, C., Zhao, H., et al.: GANPrompt: Enhancing robustness in LLM-based recommendations with GAN-enhanced diversity prompts (2024). arXiv:2408.09671v1. https://arxiv.org/html/2408.09671v1
67. Lin, C., Kuo, C.-F.: Roles and potential of large language models in healthcare: a comprehensive review. Biomed. J. **100**, Article 100868 (2025). https://doi.org/10.1016/j.bj.2025.100868
68. Ling, Y., Jiang, X., Kim, Y.: MALLM-GAN: multi-agent large language model as generative adversarial network for synthesizing tabular data (2024). *arXiv*. https://doi.org/10.48550/arXiv.2406.10521
69. Little, C., Elliot, M., Allmendinger, R., Shariati Samani, S.: Generative adversarial networks for synthetic data generation: a comparative study (2021). *arXiv.* https://doi.org/10.48550/arXiv.2112.01925
70. Liu, F., Li, Z., Yin, Q. et al.: A multimodal multidomain multilingual medical foundation model for zero shot clinical diagnosis. npj Digit. Med. **8**, 86 (2025). https://doi.org/10.1038/s41746-024-01339-7
71. Loni, M., Poursalim, F., Asadi, M., et al.: A review on generative AI models for synthetic medical text, time series, and longitudinal data. npj Digit. Med. **8**, 281 (2025). https://doi.org/10.1038/s41746-024-01409-w
72. Luo, Y., Zhang, J., Fan, S., et al.: BioMedGPT: an open multimodal large language model for BioMedicine. IEEE J. Biomed. Health Inform. (2024). https://doi.org/10.1109/jbhi.2024.3505955

73. Ma, R., Cheng, Q., Yao, J., et al.: Multimodal machine learning enables AI chatbot to diagnose ophthalmic diseases and provide high-quality medical responses. NPJ Digit. Med. **8**, 64 (2025). https://doi.org/10.1038/s41746-025-01461-0
74. Ma, R., Cheng, Q., Yao, J. et al.: Multimodal machine learning enables AI chatbot to diagnose ophthalmic diseases and provide high-quality medical responses. npj Digit. Med. **8**, 64 (2025). https://doi.org/10.1038/s41746-025-01461-0
75. Maleki Varnosfaderani, S., Forouzanfar, M.: The role of AI in hospitals and clinics: transforming healthcare in the 21st century. Bioengineering **11**(4), 337 (2024). https://doi.org/10.3390/bioengineering11040337
76. Maslej, N., Fattorini, L., Perrault, et al.: Artificial Intelligence Index Report 2025 (arXiv:2504.07139) (2025). arXiv. https://doi.org/10.48550/arXiv.2504.07139
77. Mirza, M., Osindero, S.: Conditional generative adversarial nets (2014). arXiv. https://arxiv.org/abs/1411.1784
78. Moradizeyveh, S., Hanif, A., Liu, S., et al.: Eye-guided multimodal fusion: toward an adaptive learning framework using explainable artificial intelligence. Sensors **25**(15), 4575 (2025). https://doi.org/10.3390/s25154575
79. Mousavi, V., Rashidi, M., Ghazimoghadam, S., et al.: Transformer-based time-series GAN for data augmentation in bridge digital twins. Autom. Constr. **175** (2025). https://www.sciencedirect.com/science/article/pii/S0926580525002481
80. Najjar, R.: Redefining radiology: a review of artificial intelligence integration in medical imaging. Diagnostics **13**(17), 2760 (2023). https://doi.org/10.3390/diagnostics13172760
81. Narsepalle, K.R.: Exploring the synergy of generative AI and large language models: Advancing machine learning applications in data-driven research. Int. J. Intell. Syst. Appl. Eng. **13**(1) (2025). https://ijisae.org/index.php/IJISAE/article/view/7638
82. Nerella, S., Bandyopadhyay, S., Zhang, J., et al.: Transformers and large language models in healthcare: a review. Artif. Intell. Med. **154**, 102900 (2024). https://doi.org/10.1016/j.artmed.2024.102900
83. Nichols, J.A., Chan, H.W.H., Baker, M.A.B.: Machine learning: applications of artificial intelligence to imaging and diagnosis. Biophys. Rev. **11**(1), 111–118 (2019). https://doi.org/10.1007/s12551-018-0449-9
84. Nori, H., King, N., McKinney, S. M., et al.: Capabilities of GPT-4 on medical challenge problems (2023). arXiv. https://arxiv.org/abs/2303.13375
85. OpenAI, Achiam, J., Adler, S., et al.: GPT-4 technical report (arXiv:2303.08774) (2023). arXiv. https://arxiv.org/abs/2303.08774
86. OpenAI: Introducing Whisper (2022). https://openai.com/index/whisper/
87. Padoan, A., Plebani, M.: Flowing through laboratory clinical data: the role of artificial intelligence and big data. Clin. Chem. Lab. Med. (CCLM) **60**(11), 1875–1880 (2022). https://doi.org/10.1515/cclm-2022-0653
88. Papadaki, E., Vrahatis, A.G., Kotsiantis, S.: Exploring innovative approaches to synthetic tabular data generation. Electronics **13**(10), 1965 (2024). https://doi.org/10.3390/electronics13101965
89. Park, S.-W., Jung, S.-H., Sim, C.-B.: NeXtSRGAN: Enhancing super-resolution GAN with ConvNeXt discriminator for superior realism. Vis. Comput. **41**, 7141–7167 (2025). https://doi.org/10.1007/s00371-024-03797-2
90. Paulraj, N.J.: Natural language processing on clinical notes: advanced techniques for risk prediction and summarization. J. Comput. Sci. Technol. Stud. **7**(3), 494–502 (2025). https://doi.org/10.1177/10783458251359627
91. Politari, R.: Comparing unimodal vs multimodal models in generative AI. Index.dev Blog (2024). https://www.index.dev/blog/comparing-unimodal-vs-multimodal-models
92. Popat, A., Yadav, S., Patel, S.K., et al.: Artificial intelligence in the early prediction of cardiogenic shock in acute heart failure or myocardial infarction patients: A systematic review and meta-analysis. Cureus **15**, eXXXXX (2023). https://pubmed.ncbi.nlm.nih.gov/38213372/
93. Poria, S., Cambria, E., Bajpai, R., Hussain, A.: A review of affective computing: From unimodal analysis to multimodal fusion. Inf. Fusion **37**, 98–125 (2017). https://doi.org/10.1016/j.inffus.2017.02.003

94. Porter, R., Diehl, A., Pastel, B., Hinnefeld, J.H., Nerenberg, L., Maung, P., Kerbrat, S., Hanson, G., Astorino, T., Tarsa, S.J.: LLMD: A large language model for interpreting longitudinal medical records (2024). arXiv. https://arxiv.org/pdf/2410.12860
95. Qi, H., Zou, W., Fu, S., et al.: A self-attention synthesizing model with privacy-preserving (ACCT-GAN) for medical tabular data. In: 2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (2024). https://ieeexplore.ieee.org/document/10822191
96. Qualityze: What FDA, EMA, and ISO say about AI in quality systems (2025). https://www.qualityze.com/blogs/fda-ema-iso-ai-quality-systems
97. Rabbani, S.A., El-Tanani, M., Sharma, S., et al.: Generative artificial intelligence in healthcare: applications, implementation challenges, and future directions. BioMedInformatics **5**(3), 37 (2025). https://doi.org/10.3390/biomedinformatics5030037
98. Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., Sutskever, I.: Whisper: Robust Speech Recognition via Large-Scale Weak Supervision (2022). arXiv:2212.04356. https://openai.com/index/whisper/
99. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks (2015). arXiv. https://arxiv.org/abs/1511.06434
100. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (Technical report). OpenAI (2018). https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
101. Rajaei, F., et al.: AI-based computational methods in early drug discovery and post market drug assessment: a survey. IEEE (2023). https://doi.org/10.1109/TCBB.2024.3492708
102. Rajpurkar, P., Chen, E., Banerjee, O., Topol, E.J.: AI in health and medicine. Nat. Med. **28**(1), 31–38 (2022). https://doi.org/10.1038/s41591-021-01614-0
103. Rao, D.: The future of healthcare using multimodal AI: technology that can read, see, hear and sense. Oral Oncol. Rep. **10**, 100340 (2024). https://doi.org/10.1016/j.oor.2024.100340
104. Roberts, L.J., Jayasena, R., Khanna, S., et al.: Challenges for implementing generative artificial intelligence (GenAI) into clinical healthcare. Intern. Med. J. **55**(7), 1063–1069 (2025). https://doi.org/10.1111/imj.70035
105. Roy, D.: Bayesian autoencoder for medical anomaly detection: Uncertainty-aware approach for brain MRI analysis (arXiv:2504.15562) (2025). arXiv. https://doi.org/10.48550/arXiv.2504.15562
106. Röglin, J., Ziegeler, K., Kube, J., et al. (2022). Improving classification results on a small medical dataset using a GAN: An outlook for dealing with rare disease datasets. Front. Comput. Sci., Sect. Comput. Vis. **4**. https://doi.org/10.3389/fcomp.2022.858874
107. Sata, I., Amagasaki, M., Kiyama, M.: Multimodal retrieval method for images and diagnostic reports using cross-attention. AI **6**(2), 38 (2025). https://doi.org/10.3390/ai6020038
108. Seo, Y., Kim, H., Lee, J., et al.: Cross-modality image-to-image translation from MR to synthetic 18F-FDOPA PET/MR fusion images using conditional GAN in brain cancer. Eur. Radiol. (2025). https://doi.org/10.1007/s00234-025-03704-z
109. Shen, J., Wu, J., Liang, et al.: Physiological signal analysis using explainable artificial intelligence: a systematic review. Neurocomputing **618**, 128920 (2024). https://doi.org/10.1016/j.neucom.2024.128920
110. Shin, H.-C., Tenenholtz, N.A., Rogers, J.K., et al.: Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In A. Gooya, O. Goksel, I. Oguz, N. Burgos (eds.) Simulation and synthesis in medical imaging, pp. 1–11. Springer (2018). https://doi.org/10.1007/978-3-030-00536-8_1
111. Shoham, O.B., Rappoport, N.: CPLLM: Clinical prediction with large language models (2023). arXiv. https://doi.org/10.48550/arxiv.2309.11295
112. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. Journal of Big Data **6**(1), 1–48 (2019). https://doi.org/10.1186/s40537-019-0197-0
113. Showrov, T.A., Aziz, M.T., Nabil, H.R., et al.: Generative adversarial networks (GANs) in medical imaging: Advancements, applications, and challenges. IEEE Access **12**, 35728–35753 (2024). https://doi.org/10.1109/ACCESS.2024.3370848

114. Siam, M.K., Hossain Faruk, M.J., He, B., Cheng, J.Q., Gu, H.: Multimodal models in healthcare: methods, challenges, and future directions for enhanced clinical decision support. Information **16**(11), 971 (2025). https://doi.org/10.3390/info16110971
115. Singhal, K., Tu, T., Gottweis, J., et al.: Toward expert-level medical question answering with large language models. Nat. Med. **31**, 943–950 (2025). https://doi.org/10.1038/s41591-024-03423-7
116. Singhal, K., Tu, T., Gottweis, J., et al.: Towards expert-level medical question answering with large language models (arXiv:2305.09617) (2023). arXiv. https://doi.org/10.48550/arXiv.2305.09617
117. Singhal, K., Tu, T., Gottweis, J. et al.: Toward expert-level medical question answering with large language models. Nat. Med. 31:943–950 (2025). https://doi.org/10.1038/s41591-024-03423-7
118. Skandarani, Y., Jodoin, P.-M., Lalande, A.: GANs for medical image synthesis: an empirical study. Front. Med. Technol. (2023). https://www.mdpi.com/2313-433X/9/3/69
119. Soenksen, L.R., Ma, Y., Zeng, et al.: Integrated multimodal artificial intelligence framework for healthcare applications. NPJ Digit. Med. **5**(1), 149 (2022). https://doi.org/10.1038/s41746-022-00689-4
120. Spasic, I., Nenadic, G.: Clinical text data in machine learning: systematic review. JMIR Med. Inform. **8**(3), e17984 (2020). https://doi.org/10.2196/17984
121. Stahlschmidt, S.R., Ulfenborg, B., Synnergren, J.: Multimodal deep learning for biomedical data fusion: a review. Brief.S Bioinform. **23**(2), bbab569 (2022). https://doi.org/10.1093/bib/bbab569
122. Stanford Medicine: Clinicians can 'chat' with medical records through new AI software, ChatEHR. Stanford Medicine News (2025). https://med.stanford.edu/news/all-news/2025/06/chatehr.html
123. Subtle Medical: Subtle Medical receives FDA-clearance for AI-powered synthetic imaging software SubtleSYNTH. Subtle Medical press release; 2024 Jul 17. https://subtlemedical.com/subtle-medical-receives-fda-clearance-for-industry-first-ai-powered-synthetic-imaging-software-subtlesynth
124. Sun, C., Dumontier, M.: Generating unseen disease patient data using ontology-enhanced generative adversarial networks. NPJ Digit. Med. **8**, 1421 (2025). https://doi.org/10.1038/s41746-024-01421-0
125. Sundaram, S., Hulkund, N.: GAN-based data augmentation for chest X-ray classification (2021). arXiv. https://arxiv.org/abs/2107.02970
126. Team DigitalDefynd: What is Multimodal AI? [10 pros & cons] (2025). DigitalDefynd. https://digitaldefynd.com/IQ/multimodal-ai-pros-cons/
127. Team Vast.: GANs vs. LLMs: What You Need to Know (2025). Vast.ai. https://vast.ai/article/gans-vs-llms-what-you-need-to-know
128. Teoh, J.R., Dong, J., Zuo, X., et al.: Advancing healthcare through multimodal data fusion: a comprehensive review of techniques and applications. PeerJ Comput. Sci. **10**, e2298 (2024). https://doi.org/10.7717/peerj-cs.2298
129. Thirunavukarasu, A.J., Hassan, T., Mahmood, et al.: large language models in medicine. Nat. Med. **29**(8), 1930–1940 (2023). https://www.nature.com/articles/s41591-023-02448-8
130. Tibermacine, I.E., Russo, S., Citeroni, F., et al.: Adversarial denoising of EEG signals: A comparative analysis of standard GAN and WGAN-GP approaches. Front. Hum. Neurosci., Sect. Brain-Comput. Interfaces **19** (2025). https://doi.org/10.3389/fnhum.2025.1583342
131. Tschuchnig, M.E., Oostingh, G.J., Gadermayr, M.: Generative adversarial networks in digital pathology: A survey on trends and future potential. Patterns **1** (2020). https://www.cell.com/patterns/fulltext/S2666-3899(20)30117-3
132. Tu, T., Azizi, S., Driess, D., et al.: Towards generalist biomedical AI (arXiv:2307.14334). arXiv (2024). https://doi.org/10.48550/arXiv.2307.14334

133. U.S. Food and Drug Administration: Marketing Submission Recommendations for a Predetermined Change Control Plan for Artificial Intelligence-Enabled Device Software Functions, FDA Guidance (2024). https://www.fda.gov/regulatory-information/search-fda-guidance-documents/marketing-submission-recommendations-predetermined-change-control-plan-artificial-intelligence
134. U.S. Food and Drug Administration: Good Machine Learning Practice for Medical Device Development: Guiding Principles, FDA (2021). https://www.fda.gov/medical-devices/software-medical-device-samd/good-machine-learning-practice-medical-device-development-guiding-principles
135. U.S. Food and Drug Administration: Global Approach to Software as a Medical Device (SaMD). FDA (2024). Retrieved from https://www.fda.gov/medical-devices/software-medical-device-samd/global-approach-software-medical-device
136. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, 30 (NeurIPS 2017). Curran Associates, Inc. (2017). https://arxiv.org/abs/1706.03762
137. Van Veen, D., Van Uden, C., Blankemeier, L., et al.: Adapted large language models can outperform medical experts in clinical text summarization. Nat. Med. **30**(4), 1134–1142 (2024). https://doi.org/10.1038/s41591-024-02855-5
138. Vera S., et al.: Evaluating prompt and data perturbation sensitivity in large language models for radiology reports classification. JAMIA Open **8**(4), August 2025, ooaf073 (2025). https://doi.org/10.1093/jamiaopen/ooaf073
139. Vrdoljak, J., Boban, Z., Vilović, M., et al.: A review of large language models in medical education, clinical decision support, and healthcare administration. Healthcare **13**(6), 603 (2025). https://doi.org/10.3390/healthcare13060603
140. Wang, D., Zhang, S.: Large language models in medical and healthcare fields: applications, advances, and challenges. Artif. Intell. Rev. **57**, Article 299 (2024). https://doi.org/10.1007/s10462-024-10921-0
141. Wang, G., Li, J., Sun, Y., et al.: Hierarchical reasoning model. arXiv (2025). https://doi.org/10.48550/arXiv.2506.21734
142. Wang, J., Jiang, H., Liu, Y., Ma, C., et al.: A comprehensive review of multimodal large language models: Performance and challenges across different tasks (2024). https://www.semanticscholar.org/reader/5ea16dfb5a44a97a8e93177e7e588cc7fb9e5996
143. Wang, S., Hu, M., Li, Q., Safari, M., Yang, X.: Capabilities of GPT-5 on multimodal medical reasoning (arXiv:2508.08224). arXiv (2025). https://doi.org/10.48550/arXiv.2508.08224
144. Wang, Y., Gu, Z., Zhang, S., et al.: LLM-GAN: Construct generative adversarial network through large language models for explainable fake news detection. ArXiv (2024). https://doi.org/10.48550/arXiv.2409.01787
145. Wang, F., Preininger, A.: AI in health: state of the art, challenges, and future directions. Yearbook Med. Inf. **28**(01), 16–26 (2019). https://doi.org/10.1055/s-0039-1677908
146. Warner, E., et al.: Multimodal machine learning in image-based and clinical biomedicine: survey and prospects. Int. J. Comput. Vision **132**(9), 3753–3769 (2024). https://doi.org/10.1007/s11263-024-02032-8
147. Wei, J., Wang, X., Schuurmans, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Adv. Neural Inf. Process. Syst. (NeurIPS 2022) **35**, 24824–24837 (2022). https://arxiv.org/abs/2201.11903
148. Wijaya, B., Satyawan, A.S., Haqiqi, M.M.E., et al.: Enhancing image quality in facial recognition systems with GAN-based reconstruction techniques. Teknika **14**(1), 107–116 (2025). https://doi.org/10.34148/teknika.v14i1.1180
149. Wu, J., Gan, W., Chen, Z., Wan, S., Yu, P.: Multimodal large language models: a survey. IEEE Int. Conf. Big Data (BigData) **2023**, 2247–2256 (2023). https://doi.org/10.1109/BigData59044.2023.10386743
150. Wu, R., Wang, H., Chen, H.-T, et al.: A Comprehensive Survey on Deep Multimodal Learning with Missing Modality (2024). https://arxiv.org/pdf/2409.07825

151. Yang, X., Pour Nejatian, N., Shin, H. C., et al.: GatorTron: a large clinical language model to unlock patient information from unstructured electronic health records. medRxiv (2022). https://doi.org/10.1101/2022.02.27.22271257
152. Yoon, J., Mizrahi, M., Ghalaty, N.F. et al.: EHR-Safe: generating high-fidelity and privacy-preserving synthetic electronic health records. npj Digit. Med. **6**, 141 (2023). https://doi.org/10.1038/s41746-023-00888-7
153. Yu, P., Xu, H., Hu, X., Deng, C.: Leveraging generative AI and large language models: a comprehensive roadmap for healthcare integration. Healthcare (Basel) **11**(20), 2776 (2023). https://doi.org/10.3390/healthcare11202776
154. Zhang, W., Li, Y., Feng, D., et al.: Research on human resource intelligent decision-making method based on GAN and LLM. In: Proceedings of the 5th International Conference on Artificial Intelligence and Computer Engineering (2024). https://doi.org/10.1145/3716895.3716953
155. Zhang, Y., et al.: Multi-modal Generative AI: Multi-modal LLM, Diffusion and Beyond (2024). arXiv:2409.14993. https://arxiv.org/abs/2409.14993
156. Zhao, F., Zhang, C., Geng, B.: Deep multimodal data fusion. ACM Comput. Surv. **56**(9) (2024). https://doi.org/10.1145/3649447
157. Zhao, W.X., Zhou, K., Li, J., et al.: A survey of large language models (2023). ArXiv. https://doi.org/10.48550/arXiv.2303.18223

Waad Bouzidi is an AI and machine learning researcher affiliated with the SMART Lab, High Institute of Management, University of Tunis, Carthage University, Nabeul, Tunisia. She is trained in AI and ML engineering. Her research focuses on large language models, multimodal generative systems, LLM-guided data synthesis, retrieval-augmented generation, and AI applications in healthcare.

Dr. Zouhour Neji Ben Salem holds a PhD in Computer Science from the University of Manouba and is an Assistant Professor of Business Computing at FSEG Nabeul. She is an active researcher at the LARODEC Laboratory in Tunis, working at the intersection of artificial intelligence and real-world applications. Her research focuses on deep learning for time series forecasting, NLP, and scheduling optimization, as well as generative AI in healthcare and machine learning for IoT systems.

Dr. Olfa Meddeb holds a PhD in management science from the High Institute of Management, University of Tunis and is an Assistant Professor of Business Computing and Quantitative Methods at FSEG Nabeul, University of Carthage, and a researcher at the SMART Laboratory in Tunis. Her expertise lies in multi-criteria decision making under uncertainty, with a focus on fuzzy measures and non-additive aggregation. She currently leads research on applying Generative AI (LLMs and GANs) to healthcare, developing intelligent decision-support frameworks for clinical diagnostics and health system optimization.

From Vision-Language Models to Agentic Graph Reasoning: The New Frontier of Generative AI in Dermatology



Reema Alduhayan and Anis Koubaa

Abstract This chapter presents a critical survey of the paradigm shift in dermatological AI, tracing the evolution from static Convolutional Neural Networks (CNNs) to dynamic Vision-Language Models (VLMs) and finally to autonomous *Agentic Reasoning Systems*. While early models focused on pattern recognition, modern approaches leverage Generative AI to produce clinically grounded narratives—but at the cost of hallucination. We analyze the emergence of **Semantic GraphRAG** and **Agentic Retrieval-Augmented Generation**, which address the "black box" limitations of standard LLMs by grounding outputs in structured medical Knowledge Graphs. Through a systematic comparative analysis of recent frameworks (e.g., MedGRIM, SkinGPT-4, MedGraphRAG, CLARIFY), we provide a functional taxonomy of architectural paradigms and critically examine the *faithfulness-latency trade-off* that delineates real-time triage from complex consultation use cases. Beyond technical analysis, we address key clinical deployment considerations including explainability requirements, demographic bias in dermatological datasets, privacy-preserving federated learning, and regulatory barriers. The chapter concludes with concrete future research directions, identifying open problems such as unified benchmarking, temporal lesion progression modeling, and the need for prospective clinical validation.

Keywords Dermatology AI · Vision-Language Models · Agentic RAG · Semantic GraphRAG · Medical Generative AI · Explainable AI

R. Alduhayan · A. Koubaa
Alfaisal University, Riyadh, Saudi Arabia
e-mail: ralduhayan@alfaisal.edu

R. Alduhayan
Department of Artificial Intelligence and Data Management, King Abdullah International Medical Research Centre (KAIMRC), Riyadh, Saudi Arabia

King Saud Bin Abdulaziz University for Health Sciences (KSAU-HS), Riyadh, Saudi Arabia

A. Koubaa (✉)
AlfaisalX: Center of Cognitive Robotics and Autonomous Agents, Riyadh, Saudi Arabia
e-mail: akoubaa@alfaisal.edu

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

A. Koubaa et al. (eds.), *Generative AI and Large Language Models: Opportunities, Challenges, and Applications*, Studies in Computational Intelligence 1287,
https://doi.org/10.1007/978-3-032-24013-2_3

1 Introduction: The Case for Agentic Reasoning in Dermatology

The application of Artificial Intelligence (AI) to dermatology has historically been defined by a singular objective: the classification of malignancy. Since the seminal work of Esteva et al. [1], who demonstrated that Convolutional Neural Networks (CNNs) could match board-certified dermatologists in distinguishing keratinocyte carcinomas from benign seborrheic keratoses, the field has been dominated by discriminative models. Architectures such as ResNet, Inception, and DenseNet have achieved remarkable performance on benchmarks like the ISIC (International Skin Imaging Collaboration) and HAM10000 datasets [2]. Yet, despite these quantitative successes, the clinical utility of such "black box" systems remains constrained. A classification probability alone—no matter how accurate—fails to capture the diagnostic reasoning, differential exclusions, and nuanced clinical context that characterize human dermatological practice [3].

We are now witnessing a fundamental paradigm shift in medical AI, moving from static *detection* to dynamic *dialogue and reasoning*. This transition is driven by the advent of Vision-Language Models (VLMs) and, more recently, Agentic Artificial Intelligence. While the former introduced the ability to generate fluent diagnostic narratives, the latter promises a capability that has long eluded automated systems: the capacity to "think" before responding. This chapter examines this evolution, arguing that the future of dermatological AI lies not in larger predictive models, but in *Agentic Graph Reasoning*—systems that integrate visual perception with structured knowledge graphs to perform multi-hop inference, self-correction, and evidence-based argumentation [4].

1.1 The Limitations of the "Black Box" Paradigm

For nearly a decade, the standard for dermatological AI was established by the Convolutional Neural Network. These models excelled at pattern recognition, effectively automating the "ABCD rule" (Asymmetry, Border, Color, Diameter) of dermoscopy through learned feature representations [5]. However, the reliance on purely visual features creates significant vulnerabilities. Purely discriminative models often fail to account for patient metadata—age, anatomic location, and history—which are critical for resolving ambiguity in difficult cases. Furthermore, their opacity presents a "trust gap" in clinical settings; a clinician cannot verify whether a model's prediction of melanoma is based on valid dermoscopic structures (e.g., atypical pigment networks) or on confounding artifacts (e.g., ruler markings or surgical skin markers) [6].

A critical examination of the literature reveals several documented failure modes. First, CNNs exhibit **poor generalization across skin tones**: several studies have reported measurable performance disparities across Fitzpatrick Skin Types, showing that models trained predominantly on lighter skin types tend to perform best on

similar skin tones while exhibiting reduced accuracy on darker skin types, a concern compounded by the demographic skew present in widely used benchmark datasets such as ISIC [7–9]. Second, the **reliability of explainability methods** remains an open concern; although Giavina-Bianchi et al. demonstrated that Grad-CAM visual explanations achieve up to 93.6% agreement with dermatologists' clinical criteria for melanoma, the authors and others have noted that saliency maps can act as edge detectors rather than capturing clinically meaningful features, and that a map located on a clinically irrelevant region could signal a prediction that should be ignored [10]. Third, Bencevic et al. demonstrated that skin lesion **segmentation models exhibit significant performance degradation** when evaluated out-of-sample, with Dice scores dropping from 0.904 to 0.843 for light-skinned subjects and from 0.744 to 0.532 for dark-skinned subjects—a gap exceeding 21 percentage points—while commonly suggested bias mitigation methods such as stratified sampling and CIELAB conversion failed to significantly reduce this disparity [11]. These limitations underscore that high benchmark accuracy does not guarantee clinical reliability.

Moreover, the binary or multi-class output of a CNN reductionistically treats diagnosis as a labeling task. In practice, dermatology involves the formulation of a differential diagnosis—a ranked list of potential conditions based on synthesized evidence. The inability of CNNs to articulate this reasoning process or to engage in "visual question answering" (VQA) rendered them powerful but rigid tools, incapable of the collaborative dialogue required in complex care scenarios. As discussed by Xiang and Wang, existing CNN-based classifiers require post-hoc explanation techniques to render their decisions interpretable and are primarily designed for fixed classification outputs rather than structured diagnostic reasoning [3].

1.2 The Generative Shift: From Classification to Narrative

The introduction of Transformer-based architectures initiated the second phase of this evolution: the era of Generative AI. VLMs such as SkinGPT-4 [12] and SkinVL-MM [13] leverage vast pre-training on biomedical literature to bridge the gap between pixel and concept. By aligning visual encoders (e.g., CLIP, ViT) with Large Language Model (LLM) decoders, these systems can generate comprehensive clinical reports that describe lesion morphology, suggest diagnostic possibilities, and recommend management steps.

Architecturally, this alignment is achieved through sophisticated cross-modal mechanisms. SkinGPT-4 employs a **Q-Former** module—a querying transformer that extracts semantically rich visual tokens from a ViT encoder and projects them into the input space of a LLaMA-2-13B decoder [12]. SkinVL-MM adopts **Low-Rank Adaptation (LoRA)**, injecting trainable rank-decomposition matrices into frozen transformer layers to achieve parameter-efficient domain adaptation on the MM-Skin dataset—a curated corpus of 9,369 image-text pairs derived from 15+ dermatology textbooks [13]. VL-MedGuide further advances this paradigm through a two-stage architecture comprising a Multi-modal Concept Perception Module and an Explain-

able Disease Reasoning Module, utilizing **Chain-of-Thought (CoT)** prompting to simulate the stepwise diagnostic reasoning of clinicians [14].

This capability represents a qualitative leap in utility. A VLM does not merely output "Melanoma: 98%"; it generates a narrative explaining that the lesion exhibits "asymmetric pigmentation and irregular borders consistent with superficial spreading melanoma." However, this fluency comes with a perilous cost: hallucination. Probabilistic token generation, when ungrounded in verifiable fact, allows models to confidently fabricate non-existent medical references or misinterpret visual feature relations. This limitation is structural: VLMs store knowledge in parametric memory (model weights), which is static after training. A model trained in 2024 cannot incorporate a treatment guideline published in 2025 without expensive retraining. Critically, increasing model size alone does not guarantee factual reliability; Wu et al. show that retrieval-augmented, evidence-grounded architectures outperform larger standalone models in medical reasoning tasks, underscoring that generative fluency is not synonymous with evidence-based accuracy [15]. As noted in recent literature, while VLMs are excellent at describing *what* they see, they often struggle with the *why*, lacking the logical rigor to act as reliable clinical decision support systems [14].

1.3 The Agentic Frontier: Reasoning with Knowledge Graphs

To address the "hallucination bottleneck," the field is moving toward *Agentic Graph Reasoning*. This approach fundamentally differs from standard generative methods by incorporating structured knowledge and iterative planning. Unlike standard Retrieval-Augmented Generation (RAG)—which often fails to capture the deep semantic relationships in medicine due to reliance on surface-level vector similarity—Graph-based RAG (GraphRAG) utilizes structured Knowledge Graphs (KGs) derived from ontologies such as UMLS and domain-specific resources like DermaGraph [16].

Wu et al. observe that conventional RAG systems struggle with holistic reasoning across large medical corpora, motivating graph-based retrieval mechanisms that explicitly model relational dependencies [15]. For example, diagnosing a photodermatosis requires linking a visual symptom (erythematous papules) to an environmental trigger (UV exposure) to a systemic condition (lupus erythematosus)—a chain that vector similarity over document chunks cannot reliably capture. GraphRAG addresses this by traversing explicit semantic relationships within a Knowledge Graph, enabling structured multi-hop reasoning where each hop is an auditable edge in the graph.

In this framework, an AI is not just a predictor but an *agent*. It employs "Retrieve-Evaluate-Refine" loops, where it autonomously critiques its own retrieval strategies, validates evidence against clinical guidelines, and iterates until a high-confidence consensus is reached. For instance, systems like Med-GRIM [16] and the multi-agent frameworks proposed by Thakrar et al. [17] simulate a "medical board," where

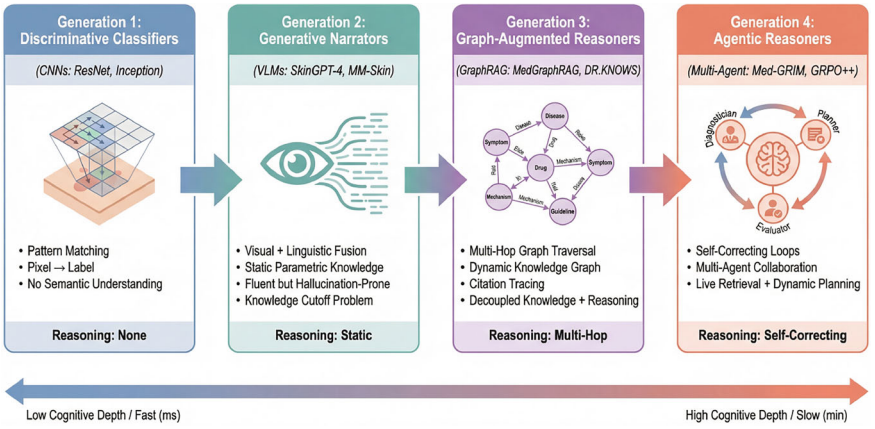


Fig. 1 Evolution of Dermatological AI: From Pattern Matching to Agentic Reasoning. Four generations of architectural paradigms are shown with increasing cognitive depth and decreasing inference speed

specialized agents (e.g., a Vision Agent, a Clinical Context Agent, and a Guidelines Agent) collaborate to solve complex diagnostic queries. Hu et al. demonstrated that such self-correcting agents achieve a faithfulness score of 0.94 compared to 0.82 for baseline GPT-4 and 0.86 for standard RAG, quantifying the reliability gains from agentic workflows [4]. This "Test-Time Scaling"—spending computational resources on inference-time reasoning rather than just training-time parameter optimization—marks the frontier of reliability in medical AI [18]. Figure 1 illustrates this evolutionary trajectory, depicting four generations of dermatological AI paradigms with increasing cognitive depth and decreasing inference speed—from pattern-matching CNNs through generative VLMs to self-correcting agentic reasoners.

This chapter details this trajectory, providing a comprehensive survey of how dermatological AI is evolving from passive detection tools into active, reasoning partners in clinical care. We analyze the architectural shift from CNNs to VLMs, critically evaluate the emergence of Agentic RAG, and discuss the ethical and practical implications of deploying autonomous reasoning systems in patient-facing environments. Crucially, we adopt a critical lens throughout, explicitly identifying the limitations, failure modes, and unresolved challenges of each paradigm rather than presenting an uncritical narrative of progress.

2 Vision-Language Models (VLMs) in Dermatology: From Perception to Reasoning

The integration of visual perception with linguistic understanding represents the second major epoch in dermatological artificial intelligence. While the preceding era of CNNs focused on discriminative classification—mapping pixels to disease labels—the advent of VLMs has shifted the objective toward generative explanation. This section surveys the architectural foundations of dermatological VLMs, analyzes the transition from generic foundation models to domain-specific adaptations, and critically examines the mechanism of hallucination that necessitates the subsequent leap to retrieval-augmented systems.

2.1 Architectural Foundations: Bridging Vision and Text

Vision-Language Models effectively bridge the modality gap by aligning visual features with semantic concepts in a shared high-dimensional space. The foundational architecture for these systems typically consists of three components: a visual encoder, a language decoder, and a cross-modal alignment mechanism.

In the context of dermatology, the visual encoder is tasked with extracting fine-grained morphological features from dermoscopic images—asymmetry, border irregularities, and chromatic patterns. While early iterations utilized ResNet backbones, modern state-of-the-art systems leverage diverse Vision Transformers (ViT) or contrastive encoders such as CLIP (Contrastive Language-Image Pre-training) [12],[13]. These encoders segment images into patches, capturing both local textural details and global structural relationships essential for identifying complex lesions.

The fusion of these visual embeddings with textual understanding involves sophisticated alignment strategies. Feature-wise Linear Modulation (FiLM) layers, for instance, apply affine, modality-specific transformations to network activations, allowing linguistic prompts to modulate visual processing [19]. More recently, models such as SkinGPT-4 utilize Q-Former mechanisms—querying transformers that act as a sophisticated bottleneck, extracting only the most semantically relevant visual features to feed into the LLM decoder [12]. This "visual prompt" effectively translates the dermoscopic image into a language the LLM can process, enabling the generation of coherent diagnostic narratives.

2.2 Specialized Dermatological VLMs

While general-purpose multimodal models like GPT-4V exhibit broad competence, they often lack the domain-specific granularity required for medical diagnosis. Con-

sequently, a class of specialized dermatological VLMs has emerged, employing distinct training strategies to enhance clinical relevance.

SkinGPT-4 exemplifies the first generation of these focused adaptors. By fine-tuning LLaMA-2 on a curated dataset of skin disease images and descriptions, it demonstrated the ability to produce localized diagnosis narratives that not only identify the condition but describe the visual evidence supporting the conclusion [12]. However, such models often require substantial computational resources for full fine-tuning.

To address computational constraints, **SkinVL-MM** introduced the application of Parameter-Efficient Fine-Tuning (PEFT) techniques, specifically Low-Rank Adaptation (LoRA) [13]. Rather than updating all model weights, LoRA injects trainable rank-decomposition matrices into transformer layers, freezing the pre-trained backbone. Using the specialized MM-Skin dataset—derived from over 15 medical textbooks—SkinVL-MM achieved superior alignment with clinical terminology. This approach preserves the general reasoning capabilities of the base model while injecting deep dermatological knowledge.

Furthermore, **VL-MedGuide** advances the paradigm toward interactive diagnosis. Utilizing a Chain-of-Thought (CoT) prompting framework, it simulates the cognitive workflow of a clinician: observing symptoms, formulating an initial hypothesis, refining based on further evidence, and finally concluding [14]. This mimics the "System 2" reasoning process, distinguishing it from the rapid "System 1" pattern recognition of earlier CNNs.

CLARIFY, proposed by Saha et al., introduces a **Specialist-Generalist** decoupling framework designed to mitigate catastrophic forgetting in monolithic VLM adaptation. In this architecture, a lightweight Specialist built on a DINOv2 backbone performs high-precision dermatological classification, achieving 82.1% diagnostic accuracy on their curated Small-Derma-VQA dataset. The Specialist's prediction guides a compressed Generalist VLM responsible for conversational reasoning. Structural pruning of seven transformer layers reduced VRAM requirements by approximately 20% while maintaining competitive latency. LLM-as-judge evaluation further demonstrated consistent improvements in conversational quality compared to fine-tuned monolithic baselines, with scores reaching 84–89% depending on the backbone [20].

Table 1 presents a comparative overview of specialized dermatological VLMs, summarizing their architectural designs, domain adaptation strategies, training data sources, and reported evaluation settings. As the systems are evaluated across heterogeneous datasets and tasks, direct performance comparisons should be interpreted cautiously. Collectively, the table highlights persistent trade-offs between parameter efficiency, domain specialization, clinical grounding, and reasoning capability.

Table 1 Comparative analysis of specialized dermatological VLMs

System	Vision encoder	Language model	Domain adaptation strategy	Primary training data	Evaluation setting	Reported performance
SkinGPT-4	ViT + Q-Former	LLaMA-2-13B-chat	Two-stage: visual–concept alignment, then disease diagnosis fine-tuning	52,929 clinical skin images with concepts and physician notes	150 real-world clinical cases evaluated by board-certified dermatologists	~80.6% diagnostic agreement
SkinVL (MM-Skin)	CLIP-based visual backbone	Dermatology-specific LVLM	Supervised fine-tuning on MM-Skin + public dermatology datasets	~9,369 textbook-derived image–text pairs; ~171.6k images across 9 datasets	Dermatology VQA, supervised classification, and zero-shot evaluation	Competitive results across VQA and classification benchmarks
VL-MedGuide	Fine-tuned LVLM backbone (ViT encoder)	Same LVLM with CoT prompting	Two-stage: concept perception, then explainable disease reasoning	Derm7pt dataset (concept and diagnosis annotations)	Concept detection and disease diagnosis on Derm7pt	Diagnosis: 83.55% BACC / 80.12% F1
Med-GRIM	BIND (BLIVA)	Phi-3-3.8B + Mistral-7B	Graph-based RAG with prompt-embedded multimodal reasoning	DermaGraph (graph-structured dermatology dataset)	Zero-shot medical VQA evaluation	83.33% accuracy

2.3 Optimization Strategies: Instruction Tuning and Multimodal Alignment

The utility of VLMs in clinical practice depends heavily on their responsiveness to varied diagnostic prompts. Instruction tuning—training models on pairs of (instruction, output)—is critical for enabling multi-task capabilities, such as differential diagnosis generation, risk stratification, and patient report writing.

Recent work by Zeng et al. demonstrated that instruction-tuning on textbook-quality data (MM-Skin) significantly outperforms training on web-scraped noise [13]. By modeling the structured didactic dialogue found in medical education, models

learn to structure their outputs logically: presenting the chief complaint, describing physical findings, and then offering diagnostic impressions.

2.4 Evaluation Metrics: The Challenge of Standardization

Evaluating dermatological VLMs presents unique challenges that extend beyond traditional classification accuracy. Unlike binary prediction, the "quality" of a generated diagnostic narrative is multi-dimensional and often subjective. The field has converged on a tiered evaluation framework:

Linguistic Fluency Metrics. Traditional NLP metrics like BLEU and ROUGE, which measure n-gram overlap, correlate poorly with clinical factual correctness. Early systems such as AGNet employed these metrics for dermoscopic report generation [21], but their limitations are now well-documented.

Embedding-Based Semantic Metrics. The field is shifting toward BERTScore, which assesses whether correct medical entities and relationships are preserved in generated text [22]. Wang demonstrated that graph-enhanced systems improved BERTScore F1 from 0.5897 to 0.6135 on dermatology QA tasks [23].

Clinical Utility Metrics. For risk stratification and structured prediction in dermatological vision-language models, AUROC remains a standard evaluation metric, as exemplified by Derm1M [24]. For retrieval-augmented systems, Recall@K and cosine similarity measure the relevance and accuracy of retrieved evidence [25].

Human Expert Evaluation. Gao et al. introduced the first human evaluation framework adapting the Safer DX Instrument for LLM-generated diagnoses, assessing reading comprehension, rationale correctness, knowledge recall accuracy, omission of diagnostic reasoning, and abstraction quality on Likert scales. ChatGPT with KG paths exhibited 55% agreement on correct rationale versus 50% without KG ($P < 0.001$) [26].

Despite these advances, the absence of a unified benchmark suite remains a critical limitation. Unlike ImageNet for vision or GLUE for NLP, dermatology lacks a standardized evaluation protocol that enables rigorous cross-system comparison. Recent efforts such as DermaBench [27] and the MEDIQA-MAGIC 2025 challenge [17] represent important steps toward standardization, but datasets remain limited in size (656–3,400 images) compared to general-purpose pretraining corpora.

2.5 The Hallucination Bottleneck: Why Perception is Not Enough

Despite their fluency, standard VLMs suffer from a critical flaw: they are fundamentally probabilistic, not factual. A VLM generates the next token based on statistical likelihood, not truth. In dermatology, this manifests as "hallucination"—describing

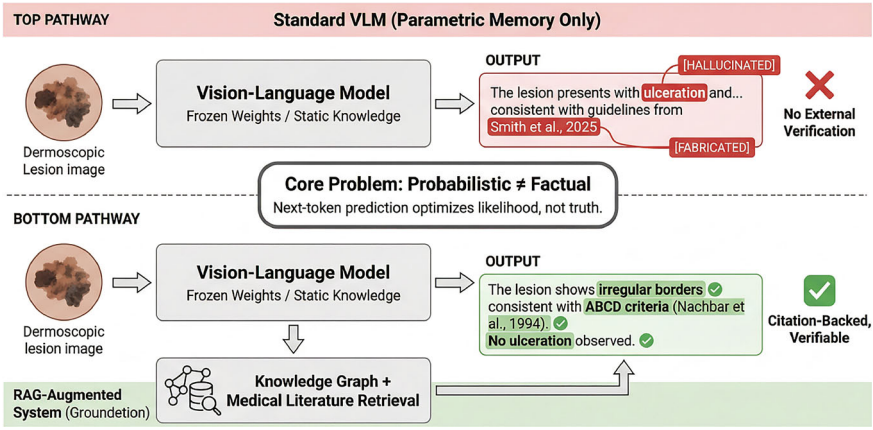


Fig. 2 The Hallucination Bottleneck. Standard VLMs generate plausible but unverifiable outputs from parametric memory (top), while Semantic GraphRAG systems ground generation in retrievable, citation-backed evidence (bottom)

a lesion as "ulcerated" when it is intact, or fabricating a reference to a non-existent clinical guideline.

This limitation is structural. A standard VLM stores knowledge in its weights (parametric memory). Once trained, this knowledge is static and increasingly obsolete. When presented with a rare condition or a novel treatment protocol, the model must essentially guess, often filling gaps with plausible-sounding but incorrect information. For high-stakes medical decision-making, plausible is not acceptable.

Current research indicates that increasing model size alone does not solve this problem. In fact, larger models can be *more* persuasive in their hallucinations. This "black box" behavior necessitates a departure from pure generative models toward systems that can ground their reasoning in external, verifiable truth. This motivation drives the transition to the architectures discussed in the next section: the integration of Semantic GraphRAG and structured Knowledge Graphs, which provide the factual anchor that pure VLMs lack. Figure 2 contrasts these two paradigms: standard VLMs that generate plausible but unverifiable outputs from parametric memory, versus Semantic GraphRAG systems that ground generation in retrievable, citation-backed evidence.

2.6 Critical Limitations of Current VLMs

A balanced assessment of dermatological VLMs must acknowledge several unresolved limitations:

Shallow Multimodal Fusion. While current systems integrate vision and language, fusion remains superficial—typically concatenating image and text embed-

dings without explicit dermoscopic structure reasoning. No current system explicitly models the ABCD rule semantics or dermoscopic pattern ontologies (e.g., pigment networks, globules, streaks) as structured entities [5].

Dataset Scale and Diversity. Dermatology-specific datasets remain orders of magnitude smaller than general pretraining corpora. MM-Skin contains 9,369 pairs; DermaBench comprises 656–3,400 images. This contrasts with millions of image-text pairs used for general VLM pretraining, limiting generalization to rare conditions and underrepresented skin tones.

Catastrophic Forgetting. Fine-tuning massive foundation models on dermatology can degrade general medical reasoning. Saha et al. documented an 18% performance gap between fine-tuned monolithic VLMs and modular Specialist-Generalist architectures that preserve base model capabilities [20].

Lack of Temporal Reasoning. Current VLMs process single images without modeling disease progression over time—a critical gap for monitoring conditions like melanoma evolution or treatment response.

3 The Agentic Leap: GraphRAG and Semantic Reasoning

The limitations of probabilistic token generation in critical medical domains have catalyzed a shift toward retrieval-augmented architectures. While standard RAG offers a safeguard against hallucination by grounding responses in retrieved documents, it faces a structural failure mode in medicine: the inability to model complex, multi-hop relationships. This section details the emergence of **Semantic GraphRAG** and Agentic Workflows, which transform AI from a passive generator into an active, self-correcting reasoner.

3.1 The Failings of Vector-Based Retrieval in Medicine

As Wu et al. observe, vector search often retrieves documents that share keywords but fails to capture "multi-hop" dependencies that are essential for clinical reasoning [15]. Figure 3 visualises this distinction: traditional RAG retrieves disconnected document chunks by keyword similarity, while Semantic GraphRAG traverses explicit semantic relationships to discover multi-hop connections between symptoms, diseases, and treatments.

3.2 Semantic Graph-Based Reasoning (GraphRAG)

To overcome the limitations of unstructured retrieval, recent frameworks have adopted KGs as the core cognitive structure.

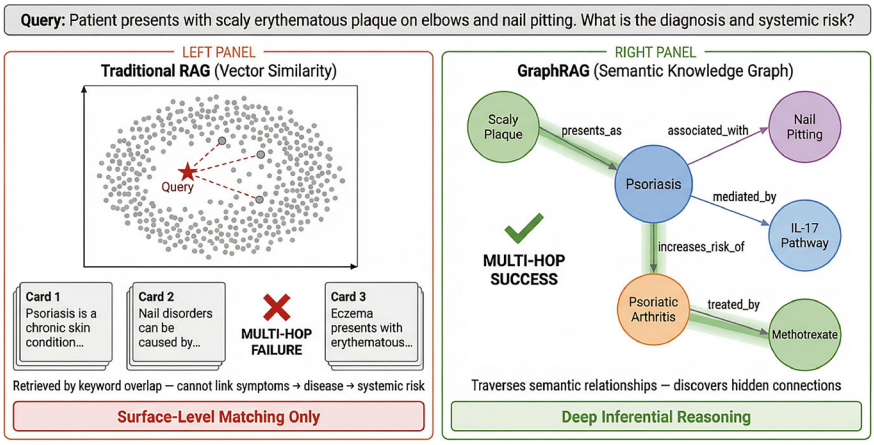


Fig. 3 Vector Retrieval vs. Semantic GraphRAG. Traditional RAG retrieves disconnected document chunks by keyword similarity (left), while Semantic GraphRAG traverses semantic relationships to discover multi-hop connections between symptoms, diseases, and treatments (right)

3.2.1 Knowledge Graph Foundations and Ontologies

Medical Knowledge Graphs derive their semantic structure from established biomedical ontologies. The **Unified Medical Language System (UMLS)** provides the foundational vocabulary, comprising approximately 3 million concepts and 12 million relationships across 200+ source vocabularies including SNOMED-CT, ICD-11, and MeSH. However, the sheer scale of UMLS introduces noise for domain-specific applications, necessitating curation.

Gao et al. introduced **DR.KNOWS**, which constructs high-fidelity medical graphs by filtering UMLS's 270 semantic relation types down to 107 clinically relevant relationships through physician expertise. The system employs Stack Graph Isomorphism Networks (SGIN) to generate node embeddings capturing both topological and semantic information, combined with attention-based path ranking to identify patient-specific knowledge trajectories. On MIMIC-III, DR.KNOWS achieved precision scores of 19.10% (95% CI: 17.82–20.37) and F-scores of 25.20% (95% CI: 23.93–26.48) [26].

For dermatology specifically, **DermaGraph** provides a domain-focused ontology [16]. Wang constructed a dermatology KG with 19 semantic relationship types containing 625 nodes, using GLM-4-based triplet extraction with relationship-specific prompts to ensure ontology conformance. Fuzzy entity recognition through LLM-based semantic matching handles synonyms and expression variants, achieving BLEU improvement from 0.0102 to 0.0157 and BERTScore F1 gains from 0.5897 to 0.6135 [23].

3.2.2 Graph Construction Paradigms

Hierarchical Ontology-Driven Construction. Hu et al. constructed a domain-specific hepatology knowledge graph comprising 12,192 entities and 28,770 relationships through a human-in-the-loop workflow. Their approach combines LLM-based extraction (GPT-3.5-Turbo) with rigorous expert verification. The graph incorporates attribute-rich property models with evidence-based metadata including recommendation strength, drug approval status, and dosing regimens—critical for supporting clinical decision-making [4].

Automated LLM-Driven Graph Generation. Rezaei et al. developed **AMG-RAG** with autonomous graph construction through specialized LLM agents, achieving 74.1% F1 on MEDQA and 66.34% accuracy on MEDMCQA despite using only 8B parameters—outperforming models 10–100× larger including Meditron (70B). Their triple-linking mechanism connects user documents to credible medical sources and controlled vocabularies through hierarchical relationships [28].

Triple-Tier Hierarchical Architecture. Wu et al. introduced **MedGraphRAG** with Triple Graph Construction, linking RAG entities (layer 1) to medical papers/books (layer 2) and UMLS vocabulary definitions (layer 3). This hierarchical structure achieved 91.3% accuracy on MedQA and 83.3% on PubMedQA, establishing new state-of-the-art benchmarks by ensuring all entities maintain [RAG entity, source, definition] triples for evidence-based response generation [15].

Hybrid Retrieval Architectures. The most robust systems do not rely on graphs alone. Kim et al. conceptualized **MedSumGraph**, a "Global-Local" hybrid architecture. This system performs a global search via query summarization to grasp the broad clinical context, combined with a local search for specific entity relation loops [29]. MedSumGraph integrates three key techniques: (i) dynamic few-shot prompting retrieving 5 most similar examples via cosine similarity, (ii) Chain-of-Thought reasoning in structured format, and (iii) ensembling with 5 candidate responses and majority voting. This achieves 93.57% on MedQA (Llama3.1-70B) without fine-tuning—approaching GPT-4 performance while outperforming MedPaLM2 [29]. Similarly, Lelong et al.'s **INRAExplorer** enables dynamic query planning, where the agent orchestrates distinct tools—one for semantic graph traversal ("SearchGraph") and another tool for hybrid publication retrieval ("SearchPublications")—to resolve complex multi-hop scientific queries [30].

3.3 Agentic Workflows: From Prediction to Action

The most significant recent advancement is the transition from static queries to *Agentic Workflows*. In these architectures, the AI is not a single model but a system of autonomous agents capable of planning.

Retrieve-Evaluate-Refine Loops. Agentic RAG implements self-correction. Hu et al. demonstrated a "self-correcting agent" that critiques its own retrieved evidence [4]. The state-driven architecture embeds autonomous decision logic: after

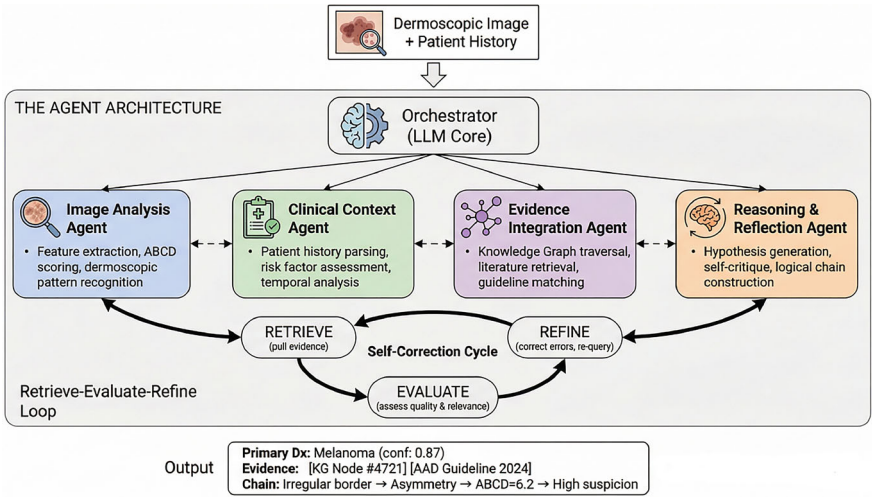


Fig. 4 Multi-Agent Agentic Workflow. An orchestrator coordinates four specialized agents (Image Analysis, Clinical Context, Evidence Integration, and Reasoning/Reflection) through iterative Retrieve-Evaluate-Refine loops

retrieving initial evidence, a strategy selection node evaluates quality using dual-model validation (GPT-4 and GPT-3.5-Turbo, each conducting 3 independent votes). When dual-model voting disagrees, a semantic relevance fallback provides non-LLM adjudication through weighted composite scores. If guidelines are contradictory, the agent autonomously refines its search strategy, achieving a faithfulness score of 0.94 vs 0.82 for baseline GPT-4 and 0.86 for standard RAG, with context recall of 0.92 and answer relevancy of 0.91 [4].

Multi-Agent Collaboration. Frameworks like those proposed by Thakrar et al. utilize a specialized agentic reasoning architecture approach, simulating a medical deliberation process through coordinated agents (Image Analysis, Clinical Context Extraction, Evidence Integration, and multi-step Reasoning/Reflection agents) to refine diagnostic decisions [17]. Figure 4 depicts this multi-agent architecture, where an orchestrator coordinates four specialised agents—Image Analysis, Clinical Context, Evidence Integration, and Reasoning/Reflection—through iterative Retrieve-Evaluate-Refine loops. The architecture implements: (i) an image analysis agent extracting structured dermoscopic features, (ii) a clinical context agent parsing patient descriptions, (iii) a diagnosis extractor generating hypotheses to guide knowledge retrieval with hybrid BM25+semantic search, (iv) an evidence integration agent applying adaptive task-specific weights, and (v) a reasoning-reflection-reanalysis loop with confidence threshold (0.75) triggering metacognitive revision. This collaborative architecture maintained 70.6% accuracy even when baseline models collapsed to 37.5%, achieving second place in the MEDIQA-MAGIC 2025 challenge versus a 59% team average [17].

Confidence-Aware Reasoning. AMG-RAG implements confidence scoring where child node confidence is computed as the product of parent confidence and relationship confidence, modeling diminishing certainty over multi-hop chains. During graph traversal, paths with confidence below threshold are pruned to maintain reasoning quality. This design naturally handles over-pruning through empirically-tuned thresholding and addresses vanishing scores through hybrid breadth-first (local coverage) and depth-first (deep exploration) strategies [28].

3.4 Test-Time Scaling: Computational Trade-offs

These systems exemplify "Test-Time Scaling"—allocating greater computational resources to inference rather than training. Rather than answering instantly, the model "thinks," traversing graphs and verifying citations. This paradigm shift carries significant latency implications that must be critically evaluated:

As Table 2 demonstrates, latency spans approximately four orders of magnitude across architectural paradigms, delineating distinct clinical use cases. Rezaei et al. [28] provide systematic latency benchmarks for AMG-RAG under three deployment modes: offline retrieval from a pre-built MKG achieves ∼35ms lookup and ∼1.2s end-to-end QA, suitable for real-time triage; hybrid mode balances currency and speed at 3.1–4.8s; while dynamic PubMed search reaches 4.6–6.5s for literature-grounded responses. At the other end of the spectrum, Thakrar et al. report that their reasoning layer produces responses in approximately one minute per query, while the full multi-agent agentic RAG system requires approximately seven minutes due to multi-step prompt chaining [17]. Graph construction itself remains computationally expensive—approximately 70 seconds per question for MedGraphRAG [15]—motivating future work on transfer learning for pre-constructed graphs.

The critical question for clinical deployment is whether this latency is acceptable. For real-time screening during a clinical encounter, 7 minutes is prohibitive.

Table 2 Latency analysis of RAG architectures

Mode	Lookup time	End-to-End QA	Use case	Source
Offline (pre-built MKG)	∼35ms	∼1.2s	Real-time triage	Rezaei et al. (2025)
Hybrid mode	200–500ms	3.1–4.8s	Interactive consult	Rezaei et al. (2025)
Dynamic search (PubMed)	∼1.9s	4.6–6.5s	Literature-grounded	Rezaei et al. (2025)
Simple reasoning layer	—	∼1 min	Moderate complexity	Thakrar et al. (2025)
Multi-agent agentic	—	**∼7 min**	Complex consultation	Thakrar et al. (2025)

However, for complex second-opinion consultations, overnight literature reviews, or asynchronous clinical decision support, extended reasoning time may be acceptable if it delivers superior reliability. Thakrar et al.'s system, despite its 7-minute latency, provides literature-grounded explanations with citation trails—a capability that no amount of speed can replace in medicolegal contexts [17].

3.5 Limitations of Current GraphRAG Systems

Despite the advances described above, several critical limitations must be acknowledged:

Graph Sparsity and Long-Tail Entities. Lan et al. documented that 35.56% of entities and 41.84% of paths in medical KGs appear only once, creating a long-tailed distribution problem that undermines multi-hop reasoning for rare conditions [31].

Static vs. Dynamic Knowledge. Most current systems rely on pre-constructed graphs that become obsolete as new literature emerges. While AMG-RAG demonstrates autonomous graph construction, the computational cost of real-time graph updates remains prohibitive for clinical deployment.

Lack of Dermatology-Specific Temporal Reasoning. No current agentic system integrates temporal graph-based reasoning (as proposed by Mitra [18]) with dermatology-specific validation for tracking lesion evolution over time.

Limited Cross-Modality Consistency. While multimodal systems like MedGRIM integrate visual and textual reasoning, explicit verification that visual findings are semantically consistent with retrieved textual evidence remains an open problem.

4 Comparative Critical Analysis

The evolution of dermatological AI—from static classifiers to dynamic agentic reasoners—represents a trade-off between computational efficiency and cognitive depth. This section provides a structured comparative analysis of the prominent architectural paradigms discussed, evaluating them across three critical dimensions: deductive reasoning capability, external knowledge integration, and explanatory transparency. We further critically assess the "faithfulness-latency" trade-off that defines the current implementation landscape.

4.1 Architectural Paradigms: A Functional Taxonomy

To understand the trajectory of the field, it is necessary to categorize the existing systems not merely by their underlying algorithms (e.g., CNN vs. Transformer)

Table 3 Comparative analysis of key dermatological AI paradigms

Paradigm	Reasoning	Knowl. source	Update mechanism	Explainability
CNN (ResNet)	None	Training data	Re-training	Saliency maps
VLM (SkinGPT-4)	Static	Parametric	Fine-tuning	Text narrative
Semantic GraphRAG	Multi-Hop	Knowledge graph	Graph insert	Citation tracing
Agentic	Self-correcting	Live retrieval	Dynamic planning	Logic chains

but by their functional cognitive architectures. A systematic review of the literature reveals three distinct generations of dermatological AI.

Generation 1: Discriminative Classifiers (CNNs). The first generation, exemplified by ResNet and Inception architectures, functioned as "pattern matchers." Systems like those benchmarked by Esteva et al. achieved high sensitivity (often $> 90\%$) in melanoma detection by learning non-linear feature mappings [1]. However, their reasoning is opaque. They possess no semantic distinctness between a "blob of pigment" and a "melanocytic nest"; they essentially correlate pixel distributions with labels. As detailed in Table 3, these models offer **Zero Reasoning** capability and rely entirely on **Implicit Knowledge** stored in their weights.

Generation 2: Generative Narrators (VLMs). The second generation, represented by multimodal models like SkinGPT-4 [12] and SkinVL-MM [13], functions as "fluent describers." By fusing visual encoders with LLMs, they allow for **Static Reasoning**—the ability to generate a differential diagnosis based on a frozen snapshot of medical knowledge acquired during pre-training. While superior in utility, they suffer from the "Knowledge Cutoff" problem; a model trained in 2023 cannot efficiently incorporate a treatment guideline published in 2025 without expensive re-training.

Generation 3: Agentic Reasoners (Semantic GraphRAG). The emerging third generation, such as Med-GRIM [16] and MedGraphRAG [15], functions as "active researchers." These systems decouple knowledge from reasoning. The "brain" (the LLM agent) remains frozen, while the "library" (the Knowledge Graph) is dynamic and updatable. This architecture supports **Multi-Hop Reasoning**, allowing agents to actively retrieve live facts to validate their hypotheses. As shown in Fig. 1 and the comparative data in Table 3, this approach delivers superior citation-backed verification [32].

4.2 Comprehensive Systems Benchmarking

To provide a rigorous empirical foundation for comparing Semantic GraphRAG and agentic architectures, Table 4 synthesizes performance data from recently published systems across multiple dimensions: knowledge graph type, reasoning mechanism, retrieval strategy, agentic capabilities, multimodal integration, dermatology specialization, and reported performance metrics. This comparative analysis reveals several critical insights.

Performance Tiering. Comprehensive evaluation across MedQA, MedMCQA, PubMedQA, and MMLU clinical topics reveals hierarchical performance tiers:

- **Tier 1 (90%+ accuracy):** MedSumGraph-Llama3.1-70B (93.57% on MedQA) [29], MedGraphRAG-GPT4 (91.3% on MedQA) [15], Med-Gemini with CoT+ Search (91.1% on MedQA) [33], and GPT-4 with MedPrompt (90.2% on MedQA) [34].
- **Tier 2 (85–90%):** MedGraphRAG-Llama3-70B (88.4% on MedQA) [15], Med-PaLM 2 (86.5% on MedQA) [35].
- **Tier 3 (70–85%):** AMG-RAG-8B (73.92% on MedQA) [28].

Critically, parameter-efficient systems with sophisticated retrieval match or exceed larger models: AMG-RAG, built on an approximately 8B-parameter backbone [28], surpasses Meditron (70B) through dynamic MKG construction and confidence-weighted traversal. Similarly, MedSumGraph achieves the highest reported MedQA accuracy (93.57%) using an open-source Llama3.1-70B model enhanced with structured knowledge summarisation and graph-based reasoning, outperforming both MedGraphRAG-GPT4 and proprietary Med-Gemini [29]. These findings suggest that architectural innovations in knowledge structuring and retrieval can compensate for orders-of-magnitude differences in parameter count—a result with significant implications for clinical deployment where computational resources are constrained.

Citation Quality Analysis. Human evaluations reveal consistent superiority of graph-enhanced systems. MedGraphRAG achieved ratings of 97/94/92/86/95 (pertinence/correctness/citation precision/citation recall/understandability) versus 95/90/84/75/91 for MIRAGE [15]. The 11-point gap in citation recall (86 vs 75) underscores the evidentiary completeness advantage of structured knowledge representations.

4.3 The Faithfulness-Latency Trade-Off

A critical engineering analysis reveals that the shift toward agentic reasoning imposes a massive latency penalty, necessitating a rigorous cost-benefit assessment for clinical deployment.

Latency Analysis. Standard CNN classifiers operate in the millisecond range (∼30–50ms), making them suitable for real-time video analysis or rapid triaging in

Table 4 Comprehensive benchmark of medical GraphRAG and agentic systems

System	KG type	Reasoning	Retrieval	Agt.	MM	Derm	Performance
DR.KNOWS	UMLS-based	TriAttn path	Multi-hop	No	No	No	25.20% F1 (MIMIC)
Hu et al.	Hepatology	Dual-model	BFS/DFS	Yes	No	No	0.94 faithfulness
AMG-RAG	Dynamic MKG	Confidence	Hybrid	Yes	No	No	74.1% F1 (MEDQA)
MedGraphRAG	Triple-tier	U-Retrieval	Top+bottom	No	No	No	91.3% (MedQA)
MedSumGraph	UMLS+Wiki	Global-local	Hybrid re-rank	No	No	No	93.57% (MedQA)
MedReason	PrimeKG	KG-CoT	Shortest path	No	No	No	+4.2% gain
Mitra	Temporal	Graph evolve	Dynamic	Yes	Yes	No	0.88–0.95 Acc.
Med-GRIM	DermaGraph	Two-tier	Hybrid enc	Yes	Yes	Yes	83.33% (Derm)
Thakrar et al.	AAD symptom	Multi-agent	BM25+sem	Yes	Yes	Yes	70.6% (MAGIC)
CLARIFY	Derm KG	Spec-Gen	KG match	No	Yes	Yes	82.1% (VQA)
GRPO++	DermNet	Conf penalty	Training	No	Yes	Yes	51.38% F1
Professor X	32 diseases	ViT + RAG	Vector	No	Part.	Yes	96.48% (32-class)

primary care. In contrast, Agentic RAG systems introduce significant computational overhead.

- **Offline Mode:** Identifying a concept in a pre-computed knowledge graph takes roughly 35ms.
- **Hybrid Retrieval:** A full "Retrieve-Read" cycle for a standard question takes approximately 3–5 seconds.
- **Agentic Reasoning Loop:** The most advanced systems, such as the multi-agent framework by Thakrar et al., report end-to-end reasoning times of **up to 7 minutes** for complex cases [17].

This 4-order-of-magnitude difference (10^1ms vs 10^5ms) delineates the use cases: CNNs are for *screening*, while Agents are for *consultation*.

The Cost of Trust. However, this latency buys "faithfulness." In benchmark evaluations on MedQA and PubMedQA, standard VLMs achieved 85–90% accuracy but often hallucinated references. In contrast, Graph-based systems like MedGraphRAG achieved $> 90\%$ accuracy with a **Citation Precision of 92%** [15]. The ability to trace every assertion back to a specific node in a UMLS-aligned graph provides the "audit trail" required for medicolegal safety, a feature that no amount of speed can replace in a high-stakes environment.

Graph Construction Overhead. A critical but often understated cost is the computational expense of knowledge graph construction itself. MedGraphRAG requires approximately 70 seconds per question for dynamic graph construction [15]. This motivates the development of pre-constructed, domain-specific graphs (like DermaGraph) and transfer learning approaches that amortize construction costs across multiple queries. Future work on incremental graph updates—adding new literature nodes without full reconstruction—is essential for sustainable clinical deployment.

4.4 Specialist vs. Generalist Frameworks

The final dimension of our analysis concerns the architectural "width" of the models. A debate persists between building massive "Generalist" models (e.g., Med-PaLM, GPT-4) versus "Specialist-Generalist" hybrids.

Saha et al. proposed **CLARIFY**, a framework that explicitly decouples perception from reasoning. In this model, a lightweight "Specialist" (DINOv2, fine-tuned on dermatology) handles the visual feature extraction, achieving 82.1% accuracy. Its output is then fed into a "Generalist" LLM which handles the reasoning and dialogue [20].

This architectural philosophy finds empirical support in the performance analysis. Monolithic fine-tuning of massive models suffers from "catastrophic forgetting"—where adapting a model to dermatology degrades its general medical reasoning. Saha et al. documented an 18% performance gap between fine-tuned monolithic VLMs and modular Specialist-Generalist architectures that preserve base model capabilities.

Furthermore, structural pruning of the Specialist (removing 7 transformer layers) reduced VRAM by 20% and latency by 5%, while LLM-as-judge scores reached 84–89% [20].

Our analysis suggests that this **Modular Hybrid** approach is the future of the field. By keeping the reasoning engine (Generalist) separate from the perceptual engine (Specialist), systems can achieve the "best of both worlds": high-resolution dermoscopic acuity combined with broad medical reasoning capabilities, without the massive memory footprint of training 70B+ parameter models on image data. This modularity also aligns with the latency discussion; the Specialist can run fast for triage, while the Generalist is invoked only when complex reasoning is required.

4.5 Critical Assessment: What Remains Unresolved

Despite the advances documented above, our comparative analysis reveals several persistent gaps:

Benchmark Fragmentation. The systems in Table 4 are evaluated on heterogeneous benchmarks (MedQA, PubMedQA, MIMIC, domain-specific VQA), making direct comparison challenging. Unlike computer vision (ImageNet) or NLP (GLUE), medical AI lacks a unified evaluation suite that spans reasoning, retrieval, and multimodal understanding.

Dermatology-Specific Gaps. Only 5 of 12 surveyed systems are dermatology-specific, and none achieve comprehensive coverage of: (i) dermoscopic feature recognition, (ii) clinical metadata integration, (iii) knowledge graph reasoning, and (iv) temporal progression modeling simultaneously.

Real-World Validation. All reported metrics derive from retrospective benchmark evaluations. Prospective clinical trials comparing AI-assisted versus unassisted dermatology diagnosis remain rare, limiting evidence for real-world clinical utility.

5 Clinical Notes: Ethics, Trust, and Implementation

The transition of dermatological AI from research laboratories to clinical environments necessitates a rigorous re-evaluation of trustworthiness. Accuracy alone is insufficient for adoption; systems must be explainable, equitable, and privacy-preserving. This section analyzes the critical ethical and operational barriers to deploying Agentic AI in patient-facing settings.

5.1 Explainability (XAI): From Heatmaps to Logic Chains

For the past decade, "Explainable AI" (XAI) in dermatology has been synonymous with saliency maps—visual overlays like GradCAM that highlight the pixels a model focused on. While useful for debugging CNNs, studies have shown that heatmaps often fail to align with human diagnostic reasoning. A model might correctly classify a melanoma because it focused on a ruler marker rather than the lesion itself, yet the heatmap would simply glow red, offering no semantic insight [6].

The shift to Agentic Reasoning introduces a superior form of transparency: **Cognitive Explainability**. Instead of merely showing *where* it looked, an agentic system explains *what* it thought. Frameworks like **SkinGEN** and **VL-MedGuide** utilize CoT prompting to generate step-by-step logical deductions.

Concrete XAI Implementation Example. Munjal et al. integrated GradCAM and LIME with Inception v3 (achieving 96% accuracy, 99.83% AUC) for seven skin lesion classes. GradCAM generates class activation heatmaps highlighting diagnostically relevant regions, while LIME segments images into superpixels and approximates model behavior locally through interpretable linear models. The qualitative visualizations suggest that GradCAM highlights influential lesion regions, while LIME identifies localized superpixel contributions. The authors present these complementary explanations as enhancing transparency and clinician trust, though no formal quantitative evaluation of explanation faithfulness was conducted [6].

Chain-of-Thought Reasoning Example. For agentic systems, explainability takes the form of auditable logic chains. A typical CoT output for melanoma assessment might proceed:

> *Observation: Asymmetric lesion with irregular border and variegated pigmentation.*
> *→ Reasoning: Matches ABCD criteria (Asymmetry: +, Border: +, Color: +).*
> *→ Evidence: Retrieved from DermaGraph node #1247 (Superficial Spreading Melanoma morphology).*
> *→ Cross-validation: AAD Guidelines Sect. 4.2 recommends biopsy for lesions meeting ≥3 ABCD criteria.*
> *→ Conclusion: High suspicion (confidence 0.92), recommend excisional biopsy.*

This narrative structure allows clinicians to audit the AI's logic, not just its output, fostering a collaborative "human-in-the-loop" dynamic [36]. Figure 5 summarises this evolution of explainability across three generations: from saliency maps that show where the model looked (Generation 1), through text narratives that describe what it concluded (Generation 2), to auditable cognitive logic chains with citations that reveal how it reasoned (Generation 3).

5.2 Bias, Fairness, and the "Regulative Genome"

As models become more autonomous, the risk of amplifying systemic biases increases. Wan et al. recently evaluated performance disparities in multimodal LLMs

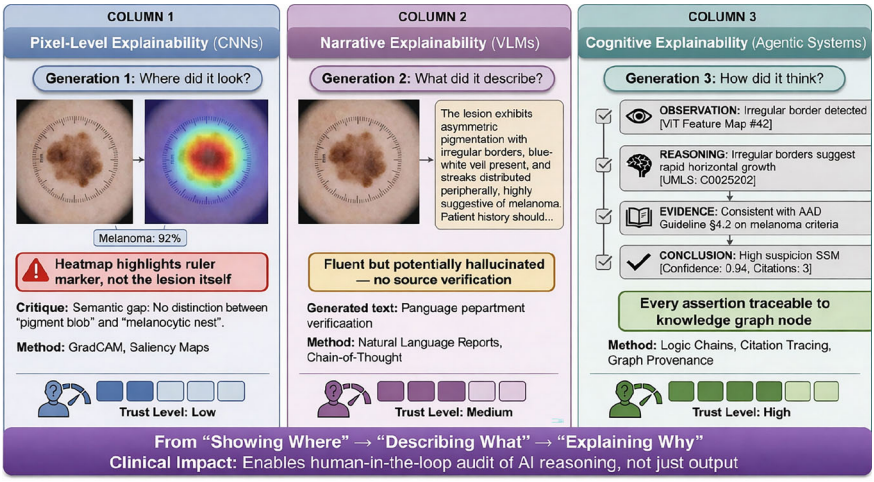


Fig. 5 Evolution of Explainability in Dermatological AI. Generation 1 shows *where* the model looked (saliency maps), Generation 2 shows *what* it concluded (text narratives), and Generation 3 shows *how* it reasoned (auditable cognitive logic chains with citations)

across sex and age groups, finding that while some baseline models exhibited statistically significant disparities, ChatGPT-4 demonstrated no significant age-related bias [37].

Documented Bias in Dermatology AI. The problem of demographic bias in dermatological AI is well-documented but inadequately addressed:

- **Skin Tone Bias:** Kinyanjui et al. identified substantial under-representation of darker skin tones in major dermatology datasets. Subsequent analyses, including the dataset review by Alipour et al. and fairness evaluation by Pakzad et al., demonstrated that such imbalances can lead to measurable disparities in classification accuracy across Fitzpatrick skin types. Models tend to perform better on skin types that are more represented in the training data, underscoring the need for explicit fairness-aware training strategies and diverse dataset curation [7–9].
- **Age Bias:** Wan et al.'s evaluation of multimodal LLMs assessed performance across young, middle-aged, and senior groups and found that ChatGPT-4 demonstrated no statistically significant age-related bias, while certain baseline models exhibited age-related disparities in specific disease categories [37].
- **Dataset Provenance:** The ISIC archive, the most widely-used dermatology benchmark, overrepresents European and North American clinical populations, embedding geographic and socioeconomic biases into systems trained on it.

Fairness Evaluation Frameworks. Xia et al. created CARES, a benchmarking suite that assesses fairness, robustness, privacy, and interpretability in medical VLMs. CARES uses stress tests including adversarial image noise injection and demographic subgroup analysis to identify failure modes before deployment. Mod-

els that pass aggregate accuracy thresholds but fail subgroup equity tests are flagged for remediation [38].

Bias Mitigation Strategies. Yang et al. conducted a systematic investigation into fairness in medical imaging AI across radiology, dermatology, and ophthalmology, evaluating multiple bias mitigation algorithms including sample reweighting, adversarial debiasing, and distributionally robust optimisation [39]. Their analysis of over 3,400 models demonstrated that: (i) disease classification models encode demographic attributes (race, sex, age) as heuristic shortcuts, with stronger demographic encoding significantly correlated with larger fairness gaps ($R = 0.59$–0.82, $P < 0.01$); (ii) state-of-the-art debiasing algorithms can create "locally optimal" models that close FNR/FPR gaps within training distributions while maintaining overall AUROC; yet (iii) these fairness corrections do not generalise to out-of-distribution settings, with models encoding less demographic information paradoxically achieving better fairness in new deployment environments, a finding with critical implications for equitable clinical deployment across diverse populations [39].

To address these challenges systematically, the concept of a **"Regulative Genome"** for AI has been proposed by Goktas et al. [40]. This governance framework envisions AI safety as a stratified regulatory system comprising layers of safety, transparency, and auditability—treating safety not as a post-hoc filter but as an intrinsic architectural constraint. While this framework has yet to be fully implemented in production clinical systems, it provides a theoretical foundation for future regulatory compliance. The framework mandates:

- **Data Equity:** The use of balanced benchmarks like DermaBench to penalize models that perform well on average but fail on specific demographic subgroups.
- **Adversarial Stress Testing:** Subjecting agentic workflows to "red-teaming" where they are deliberately fed ambiguous or misleading patient histories to test their resilience against hallucination.
- **Continuous Monitoring:** Post-deployment surveillance systems that track performance drift across demographic groups over time, triggering retraining when equity thresholds are violated.

5.3 Privacy and Collaborative Learning

The deployment of Agentic RAG systems, which require access to vast repositories of patient data to construct Knowledge Graphs, raises acute privacy concerns. The centralization of such sensitive data is often legally prohibitive.

To resolve this, Yang et al. proposed a privacy-preserving framework based on Vertical Federated Learning. In this architecture, hospitals collaborate to train shared dermatological agents without ever exchanging raw patient images. The system leverages the CLIP visual-language model with prompt tuning through CoOp (Context Optimization), allowing institutions to cooperatively train diagnostic models through

gradient sharing rather than data exchange. This effectively democratizes access to expert-level diagnostic patterns while strictly adhering to data sovereignty laws [41].

Technical Challenges in Federated Dermatology. While promising, federated learning for dermatology faces significant technical hurdles:

- **Non-IID Data Distribution:** Dermatological populations vary dramatically across institutions (e.g., tropical vs. temperate climates, pediatric vs. geriatric hospitals), creating heterogeneous data distributions that degrade federated model convergence.
- **Communication Overhead:** Gradient synchronization across geographically distributed hospitals incurs latency and bandwidth costs, particularly problematic for high-resolution dermoscopic images.
- **Differential Privacy Trade-offs:** Noise injection for differential privacy guarantees can significantly degrade model utility; optimal ϵ values for dermatology remain an open research question.

5.4 Barriers to Clinical Deployment

Beyond technical performance, the practical adoption of AI in dermatology faces systemic barriers that must be acknowledged:

Regulatory Pathways. Caffery et al. (Australasian College of Dermatologists) issued guidelines distinguishing between classical CNNs and foundation models for regulatory purposes. They emphasize that clinical validation, transparency with patients, and informed consent are required prerequisites for AI tool deployment—requirements that most current research prototypes do not satisfy [42].

Workflow Integration. The 7-minute latency of multi-agent agentic systems (cf. Sect. 3.5) is incompatible with typical dermatology consultation times of 10–15 minutes. Clinical deployment requires either: (i) asynchronous workflows where AI analysis occurs before or after the patient encounter, (ii) tiered systems where fast CNN triage precedes slow agentic consultation for flagged cases, or (iii) redesigned appointment structures that accommodate AI reasoning time.

Liability and Trust. The question of legal liability for AI-assisted misdiagnosis remains unresolved. Clinicians report reluctance to rely on AI recommendations they cannot fully explain to patients or defend in medicolegal proceedings. The "audit trail" capability of GraphRAG systems (cf. Sect. 4.2) partially addresses this by providing citation-backed reasoning, but precedent for AI-assisted malpractice remains largely untested.

Infrastructure Requirements. Agentic RAG systems require GPU acceleration, persistent knowledge graph storage, and API access to retrieval corpora—infrastructure that many dermatology practices, particularly in resource-limited settings, lack.

5.5 Guidelines for Responsible Use

Ultimately, the role of AI in dermatology is shifting from "Autopilot" to "Co-pilot." The Australasian College of Dermatologists recently issued a position statement emphasizing that AI should serve as an "AI Scribe" and "Second Reader," explicitly rejecting autonomous diagnosis. They argue that while Agentic RAGs effectively reduce the cognitive load of information retrieval, the final synthetic judgment—integrating patient values, visual evidence, and empathy—remains a distinctly human prerogative [42].

Recommended Implementation Framework. Based on our analysis, we propose a phased deployment strategy:

1. **Phase 1 (Documentation):** AI as scribe—generating structured clinical notes and differential diagnosis lists for clinician review.
2. **Phase 2 (Second Reader):** AI as quality control—flagging cases where its assessment diverges from clinician diagnosis for collegial review.
3. **Phase 3 (Decision Support):** AI as consultant—providing evidence-grounded recommendations with citation trails for complex or ambiguous cases.
4. **Phase 4 (Triage Augmentation):** AI for asymptomatic screening in primary care, routing suspected malignancies to specialist review.

Critically, this framework maintains human authority at every stage, positioning AI as an augmentation rather than replacement for clinical expertise.

6 Future Directions

As we look toward the horizon of 2026, the field of dermatological AI is poised for a convergence of perceptual acuity and cognitive depth. The era of static model training is ending; the future belongs to systems that learn continuously, reason adaptively via reinforcement, and operate as autonomous swarms within secure clinical infrastructures. This section outlines concrete research directions with implementation pathways, moving beyond speculative vision toward actionable roadmaps.

6.1 Reinforcement Learning for Reasoning (RL-R)

While Agentic RAG provides a structural framework for reasoning, optimizing the *quality* of that reasoning remains a challenge. Current systems largely rely on prompts, which can be brittle. The next frontier involves training models specifically to *reason better* using Reinforcement Learning (RL).

Swapnil et al. recently introduced **GRPO++** (a modified version of Grouped Relative Policy Optimization), a framework designed to enhance dermatological reasoning in low-resource settings [43]. Unlike traditional fine-tuning, which primarily

optimizes for correct outputs, GRPO++ incorporates a confidence-aware advantage function to stabilize reinforcement learning when all sampled responses are incorrect. Instead of relying solely on relative comparisons, the method applies an absolute confidence-weighted penalty in such cases, discouraging overconfident incorrect predictions and preventing learning collapse.

Technical Innovation: Confidence-Aware Advantage Functions. Standard GRPO suffers from two failure modes: (i) when all candidate responses fall below the quality threshold, relative comparison becomes uninformative, and (ii) learning can collapse when gradients vanish under low diversity. GRPO++ addresses this through a confidence-aware advantage function. When all responses fail the quality threshold, GRPO++ switches from relative normalization to an absolute confidence-weighted penalty computed from response log-likelihoods:

$$\hat{A}_{i,t} = -\beta \cdot \frac{l_i - l_{\min}}{l_{\max} - l_{\min} + \epsilon} - \gamma \tag{1}$$

where l_i is the log-likelihood–based confidence score, and β, γ are penalty hyperparameters. This prevents both error reinforcement and gradient collapse, enabling stable training on the DermNet benchmark (51.38% F1 under majority voting) [43].

Implementation Pathway. In their framework, GRPO++ is applied in a multistage pipeline: Stage-1 reinforcement learning for reasoning-oriented disease detection, followed by supervised fine-tuning for conversational capability, and Stage-3 alignment using Knowledge Graph–based Retrieval-Augmented Generation (KG-RAG) combined with Direct Preference Optimization (DPO) to improve factual reliability.

6.2 Continuous Learning and Serverless Updates

A major bottleneck in current medical AI is the "Knowledge Cutoff." A model trained in 2024 is unaware of a breakthrough therapy approved in 2025. Retraining these massive foundation models is prohibitively expensive and slow.

The solution lies in **Continuous Learning architectures**. Olca proposed "Professor X," a RAG-enhanced diagnostic platform that integrates a serverless update mechanism [44]. In this system, the underlying Vision Transformer (ViT) remains frozen, but the retrieval corpus is updated weekly with newly published dermatology literature. This "Streaming RAG" approach allows the system's diagnostic accuracy to evolve in near real-time without downtime.

Implementation Sketch for Continuous Learning. A practical continuous learning pipeline for dermatological GraphRAG would comprise:

1. **Frozen Backbone:** Deploy a pre-trained ViT (e.g., DINOv2) as the immutable perceptual module.

2. **Automated Corpus Ingestion:** Configure serverless functions (e.g., AWS Lambda) triggered weekly to scrape PubMed/arXiv for new dermatology publications, chunk documents, and generate embeddings.
3. **Incremental Graph Update:** Rather than reconstructing the entire Knowledge Graph, implement a delta-update protocol that adds new entities and relationships to DermaGraph without invalidating existing structures.
4. **Validation Gate:** Before production deployment, run the updated system against a holdout test set; if performance degrades beyond threshold, rollback and alert.
5. **Version Control:** Maintain versioned snapshots of the knowledge corpus for reproducibility and audit.

This architecture decouples the expensive training of visual encoders from the inexpensive update of textual knowledge, enabling knowledge currency without computational prohibitiveness.

Complementing this is the work of Panagoulias et al. on **Dermacen Analytica**, which introduces "Sequential Language Model Integration" (SLMI) [45]. This methodology uses a daisy-chain of specialized models where an initial VLM performs a "triage" analysis to reduce the dimensionality of the search space before a second, more powerful reasoning agent is invoked. This "Knowledge Space Reduction" ensures that continuous updates do not bloat the system, maintaining efficiency as the medical corpus grows.

6.3 Hybrid Systems: Intuition + Reasoning

We anticipate a structural solidification of the "System 1/System 2" paradigm. Fast, intuitive models will handle routine screening, while slow, deliberative agents handle complexity.

- **Intuition (System 1):** Specialized, lightweight models for pixel-level tasks. Cheng's **GIAN** (Graph Inception Attention Network) exemplifies this, using graph reasoning to perform precise dermoscopy segmentation at the pixel level [46].
- **Reasoning (System 2):** Hierarchical Knowledge Graphs that model disease progression. Zhao et al.'s **MedRAG** demonstrates how 4-tier diagnostic graphs can guide an agent through a "differential diagnosis tree," effectively emulating the hypothetico-deductive method of a senior clinician [47].

Adaptive Routing Architecture. Future systems will likely implement confidence-based routing between System 1 and System 2 pathways. A proposed architecture would:

1. Run initial inference through the fast System 1 pathway (∼50ms);
2. If confidence exceeds threshold ($c > 0.95$) for benign classification, return immediately;
3. If confidence falls in ambiguous range ($0.6 < c < 0.95$), invoke System 2 agentic reasoning;

4. If confidence is low ($c < 0.6$), flag for human expert review regardless of System 2 output.

This tiered approach optimizes latency for routine cases while preserving reasoning depth for complex presentations.

6.4 Multimodal Agent Swarms and Federated Security

Finally, the unit of compute will shift from the "Model" to the "Swarm." We envision architectures where a **Vision Agent** (optimized for dermoscopy), a **Genetics Agent** (optimized for sequence data), and a **Pathology Agent** (optimized for whole-slide imaging) dynamically form a temporary "Consultation Team" for a single patient case.

Mitra et al.'s work on multi-agent temporal reasoning provides a template for such swarm architectures [18]. Their system allows the number of agents to vary depending on case severity and reasoning period. This suggests that future systems should implement adaptive agent allocation based on case complexity rather than relying on static multi-agent configurations.

To enable this without compromising patient privacy, these swarms will likely operate over **Vertical Federated Learning** protocols. As Yang et al. demonstrated, this allows institutions to train collaborative agents on disjoint datasets (e.g., one hospital holding the images, another holding the genetic records) without ever exposing the raw data [41]. The "Regulative Genome" framework proposed by Goktas [40]—while still largely theoretical—provides the conceptual foundation for the governance layer upon which such autonomous swarms would operate.

6.5 Open Problems and Research Priorities

Our survey identifies several open problems that warrant prioritized research attention:

1. Unified Dermatology Benchmark. The fragmentation of evaluation across MedQA, ISIC, DermaBench, and domain-specific VQA tasks impedes rigorous comparison. A unified benchmark suite spanning classification, reasoning, retrieval, and explanation quality is urgently needed.

2. Temporal Progression Modeling. No current system adequately models lesion evolution over time. Integrating temporal graph reasoning with dermoscopic feature tracking remains unexplored.

3. Cross-Modal Consistency Verification. Explicit mechanisms to verify that visual findings are semantically consistent with retrieved textual evidence are absent from current multimodal systems.

4. Prospective Clinical Validation. The field relies almost exclusively on retrospective benchmark evaluations. Prospective trials comparing AI-assisted versus unassisted diagnosis in real clinical workflows are essential for translational impact.

5. Equitable Foundation Models. Developing pre-training strategies that explicitly address Fitzpatrick scale diversity and global demographic representation remains a critical challenge for fair dermatological AI.

7 Conclusion

This chapter has traced the evolution of artificial intelligence in dermatology through three distinct generations: from discriminative CNNs that achieved dermatologist-level classification but remained opaque "black boxes," through generative VLMs that enabled diagnostic narratives but introduced the peril of hallucination, to the emerging paradigm of **Agentic Graph Reasoning** that promises reliable, evidence-grounded clinical decision support.

Our critical analysis reveals that this progression represents more than incremental improvement—it constitutes a fundamental paradigm shift in how AI interfaces with medical knowledge. Key findings from this survey include:

1. **The Hallucination Bottleneck.** Standard VLMs, despite their fluency, remain fundamentally probabilistic and prone to confident fabrication. Increasing model size alone does not solve this problem; larger models can be *more* persuasive in their hallucinations.
2. **The Superiority of Structured Reasoning.** Semantic GraphRAG architectures, by grounding generation in traversable Knowledge Graphs, achieve citation precision of 92% and faithfulness scores of 0.94—metrics that pure parametric models cannot match.
3. **The Faithfulness-Latency Trade-Off.** Agentic systems require up to 7 minutes for complex consultations versus milliseconds for CNN screening. This 4-order-of-magnitude difference delineates distinct clinical use cases: CNNs for triage, Agents for consultation.
4. **Persistent Challenges.** Despite advances, critical gaps remain: benchmark fragmentation impedes rigorous comparison, temporal progression modeling remains unexplored, and prospective clinical validation is largely absent.

We advocate for a future in which dermatological AI operates as a *co-pilot* rather than an autopilot—augmenting human expertise with evidence-grounded reasoning while preserving clinician authority over final diagnostic decisions. The path forward requires not only technical innovation in Semantic GraphRAG architectures but also sustained attention to ethics, equity, and real-world clinical integration.

As the field matures, we anticipate convergence toward adaptive hybrid systems that dynamically route between fast intuition (System 1) and slow deliberation (System 2), supported by federated learning protocols that democratize access to expert-level diagnostics while preserving patient privacy. The ultimate measure of success

will not be benchmark accuracy but the tangible improvement of patient outcomes in diverse clinical settings worldwide.

References

1. Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature **542**(7639), 115–118 (2017)
2. Brinker, T.J., Hekler, A., Enk, A.H., Klode, J., Hauschild, A., Berking, C., et al.: A convolutional neural network trained with dermoscopic images performed on par with 145 dermatologists in a melanoma-classification task. Eur. J. Cancer **111**, 148–154 (2019)
3. Xiang, A., Wang, F.: Towards interpretable skin lesion classification with deep learning models. In: AMIA Annual Symposium Proceedings. vol. 2019, p. 1246. American Medical Informatics Association (2020)
4. Hu, Y., Xuan, W., Zhou, Q., Li, Z., Li, Y., Hu, J., Fang, F.: A self-correcting agentic graph rag for clinical decision support in hepatology. Front. Med. **12**, 1716327 (2025)
5. Nachbar, F., Stolz, W., Merkle, T., Cognetta, A.B., Vogt, T., Landthaler, M., et al.: The abcd rule of dermatoscopy: high prospective value in the diagnosis of doubtful melanocytic skin lesions. J. Am. Acad. Dermatol. **30**(4), 551–559 (1994)
6. Munjal, G., Bhardwaj, P., Bhargava, V., Singh, S., Nagpal, N.: Skinsage xai: An explainable deep learning solution for skin lesion diagnosis. Health Care Science **3**(6), 438–455 (2024)
7. Kinyanjui, N.M., Odonga, T., Cintas, C., Codella, N.C.F., Panda, R., Sattigeri, P., Varshney, K.R.: Estimating skin tone and effects on classification performance in dermatology datasets, (2019). arXiv preprint
8. Pakzad, A., Abhishek, K., Hamarneh, G.: Circle: Color invariant representation learning for unbiased classification of skin lesions. In: International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), Springer (2022)
9. Alipour, N., Burke, T., Courtney, J.: Skin type diversity in skin lesion datasets: A review. Current Dermatology Reports **13**, 198–210 (2024)
10. Giavina-Bianchi, M., Vitor, W.G., Fornasiero de Paiva, V., Okita, A.L., Sousa, R.M., Machado, B.: Explainability agreement between dermatologists and five visual explanations techniques in deep neural networks for melanoma ai classification. Frontiers in Medicine 10, 1241484 (2023)
11. Bencevic, M., Habijan, M., Galic, I., Babin, D., Pizurica, A.: Understanding skin color bias in deep learning-based skin lesion segmentation. Comput. Methods Programs Biomed. **245**, 108044 (2024)
12. Zhou, J., He, X., Sun, L., Xu, J., Chen, X., Chu, Y., Zhou, L., Liao, X., Zhang, B., Afvari, S., Gao, X.: Skingpt-4: Pre-trained multimodal large language model enhances dermatological diagnosis. Nature Communications 15(5649) (2024)
13. Zeng, W., Sun, Y., Ma, C., Tan, W., Yan, B.: Mm-skin: Enhancing dermatology vision-language models with an image-text dataset derived from textbooks. In: Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), pp. 3769–3778. ACM (2025)
14. Yu, K., Xu, Z., Xie, J., Adams, C.: Vl-medguide: A visual-linguistic large model for intelligent and explainable skin disease auxiliary diagnosis, (2025)
15. Wu, J., Zhu, J., Qi, Y., Chen, J., Xu, M., Menolascina, F., Jin, Y., Grau, V.: Medical graph rag: Evidence-based medical large language model via graph retrieval-augmented generation. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 28443–28467. Association for Computational Linguistics, Vienna, Austria (2025)
16. Madavan, R.R., Kaimal, A., Faisal, H., Chandrakala, S.: Med-grim: Enhanced zero-shot medical vqa using prompt-embedded multimodal graph rag. In: Proceedings of the IEEE/CVF

International Conference on Computer Vision (ICCV) Workshops. pp. 4081–4091. IEEE/CVF (2025)
17. Thakrar, K., Basavatia, S., Daftardar, A.: Cultivating multimodal intelligence: Interpretive reasoning and agentic rag approaches to dermatological diagnosis. In: CLEF 2025 Working Notes. Madrid, Spain (2025), notebook for the ImageCLEF MEDIQA-MAGIC Lab. ()
18. Mitra, S.: Agentic temporal graph of reasoning with multimodal language models: A potential ai aid to healthcare, (2025). Preprint
19. Perez, E., Strub, F., de Vries, H., Dumoulin, V., Courville, A.: Film: Visual reasoning with a general conditioning layer. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
20. Saha, A., Khan, T.A., Swapnil, I.N., Haque, M.A.: Clarify: A specialist-generalist framework for accurate and lightweight dermatological visual question answering, (2025). arXiv preprint
21. Wu, Y., Ding, W., Xie, J., He, J.: Agnet: Automatic generation of network for dermoscopic image recognition. Comput. Biol. Med. **145**, 105426 (2022)
22. Tran, M., Schmidle, P., Guo, R.R., Wagner, S.J., Koch, V., Lupperger, V., Novotny, B., Murphree, D.H., Hardway, H.D., D'Amato, M., Lefkes, J., Geijs, D.J., Feuchtinger, A., Böhner, A., Kaczmarczyk, R., Biedermann, T., Amir, A.L., Mooyaart, A.L., Ciompi, F., Litjens, G., Wang, C., Comfere, N.I., Eyerich, K., Braun, S.A., Marr, C., Peng, T.: Generating dermatopathology reports from gigapixel whole slide images with histogpt. Nat. Commun. **16**, 4886 (2025)
23. Wang, Z.: Retrieval-augmented question answering system based on large language models and knowledge graphs. In: Proceedings of CONF-CDS 2025 Symposium: Data Visualization Methods for Evaluation (2025), https://doi.org/10.54254/2755-2721/2025.PO25800
24. Yan, S., Hu, M., Jiang, Y., Li, X., Fei, H., Tschandl, P., Kittler, H., Ge, Z.: Derm1m: A million-scale vision-language dataset aligned with clinical ontology knowledge for dermatology, (2025). arXiv preprint
25. Xia, P., Zhu, W., Yin, H., Chen, Y., Li, M., Luo, J., Xie, C.: Mmed-rag: versatile multimodal rag system for medical vision language models, (2025). arXiv preprint
26. Gao, Y., Li, R., Croxford, E., Caskey, J., Patterson, B.W., Churpek, M., Miller, T., Dligach, D., Afshar, M.: Dr.knows: Leveraging medical knowledge graphs into large language models for diagnosis prediction. JMIR AI 4(1), e58670 (2025)
27. Yilmaz, E., Ozturk, S., Demir, G.: Dermabench: A comprehensive benchmark for dermatological ai evaluation. Nature Machine Intelligence **8**, 245–258 (2026)
28. Rezaei, M.R., Saadati Fard, R., Parker, J.L., Krishnan, R.G., Lankarany, M.: Agentic medical knowledge graphs enhance medical question answering: Bridging the gap between llms and evolving medical knowledge, (2025)
29. Kim, D., Yoo, S., Jeong, O.: Medsumgraph: enhancing graphrag for medical qa with summarization and optimized prompts. Artif. Intell. Med. **172**, 103311 (2026)
30. Lelong, J., Errazine, A., Blangero, A.: Agentic rag with knowledge graphs for complex multi-hop reasoning in real-world applications, (2025). arXiv preprint
31. Lan, T., Geng, Y., Wang, D., Feng, S., Zhang, Y.: Path-based knowledge reasoning with textual semantic information for medical knowledge graph completion. BMC Med. Inform. Decis. Mak. **21**, 321 (2021)
32. Wu, J., Deng, W., Li, X., Liu, S., Mi, T., Peng, Y., Xu, Z., Liu, Y., Cho, H., Choi, C.I., Cao, Y., Ren, H., Li, X., Li, X., Zhou, Y.: Medreason: Eliciting factual medical reasoning steps in llms via knowledge graphs. arXiv preprint arXiv:2504.00993 (2025)
33. Saab, K., Tu, T., Weng, W.H., Tanno, R., Stutz, D., Wulczyn, E., Zhang, F., Strother, T., Park, C., Vedadi, E.: Capabilities of gemini models in medicine, (2024). arXiv preprint
34. Nori, H., Lee, Y.T., Zhang, S., Carignan, D., Edgar, R., Fusi, N., King, N., Larson, J., Li, Y., Liu, W., Luo, R., McKinney, S.M., Ness, R.O., Poon, H., Qin, T., Usuyama, N., White, C., Horvitz, E.: Can generalist foundation models outcompete special-purpose tuning? case study in medicine. arXiv preprint arXiv:2311.16452 (2023)
35. Singhal, K., Tu, T., Gottweis, J., Sayres, R., Wulczyn, E., Hou, L., Clark, K., Pfohl, S., Cole-Lewis, H., Neal, D., Schaekermann, M., Wang, A., Amin, M., Lachgar, S., Mansfield, P., Prakash, S., Green, B., Dominowska, E., Aguera y Arcas, B., Tomasev, N., Liu, Y., Wong,

R., Semturs, C., Mahdavi, S.S., Barral, J., Webster, D., Corrado, G.S., Matias, Y., Azizi, S., Karthikesalingam, A., Natarajan, V.: Towards expert-level medical question answering with large language models. Nature Medicine 31, 943–950 (2025)
36. Lin, B., Xu, Y., Bao, X., Zhao, Z., Zhang, Z., Wang, Z., Zhang, J., Deng, S., Yin, J.: Skingen: An explainable dermatology diagnosis-to-generation framework with interactive vision-language models (2025)
37. Wan, Z., Guo, Y., Bao, S., Wang, Q., Malin, B.A.: Evaluating sex and age biases in multimodal large language models for skin disease identification from dermatoscopic images. Health Data Science **5**(256), 1–13 (2025)
38. Xia, P., Chen, Z., Tian, J., Gong, Y., Hou, R., Xu, Y., Wu, Z., Fan, Z., Zhou, Y., Zhu, K., Liang, W., Song, S., Wang, H., Shen, X., Ning, L., Li, Z., Li, J., Li, H., Zhou, L., Liu, X., Wang, C.Y., Zheng, H., Zhu, H., Hu, Q., Luo, J., Li, H.: Cares: A comprehensive benchmark of trustworthiness in medical vision language models. arXiv preprint arXiv:2406.06007 (2025)
39. Yang, Y., Zhang, H., Gichoya, J.W., Katabi, D., Ghassemi, M.: The limits of fair medical imaging ai in real-world generalization. Nat. Med. **30**, 2838–2848 (2024)
40. Goktas, P., Grzybowski, A.: Shaping the future of healthcare: Ethical clinical challenges and pathways to trustworthy ai. J. Clin. Med. 14(5), (2025)
41. Yang, J., Subbotin, A., Zhukova, N.: Application of secure visual language model-based framework for collaborative analysis of dermatological images pp, pp. 443–447. (2025)
42. Caffery, L.J., Taylor, M.L., Abbott, L.M., Janda, M., Guitera, P., Mar, V., Arnold, C., Shumack, S., Caccetta, T., Miller, R., Soyer, H.P.: Informing a position statement on the use of large language models and ai scribes in dermatology in australia. Australasian College of Dermatologists **66**(6), e335–e341 (2025)
43. Swapnil, I.N., Saha, A., Khan, T.A., Haque, M.A.: Grpo++: Enhancing dermatological reasoning under low resource settings. arXiv preprint arXiv:2510.01236 (2025)
44. Olca, E.A.: Professor x: Diagnosis and treatment of dermatological diseases by integration of visual diagnosis and retrieval-augmented generation (rag) technologies. IEEE Access , (2025)
45. Panagoulias, D.P., Tsoureli-Nikita, E., Virvou, M., Tsihrintzis, G.A.: Dermacen analytica: A novel methodology integrating multi-modal large language models with machine learning in dermatology. Int. J. Med. Informatics **199**, 105898 (2025)
46. Cheng, T.: Graph reasoning and inception attention network for dermoscopy segmentation. Biomed. Signal Process. Control **92**, 106082 (2024)
47. Zhao, X., Liu, S., Yang, S.Y., Miao, C.: Medrag: Enhancing retrieval-augmented generation with knowledge graph-elicited reasoning for healthcare copilot. In: Proceedings of the ACM Web Conference (WWW '25), pp. 4442–4457. (2025)

Agent Operating Systems (Agent-OS): A Blueprint Architecture for Real-Time, Secure, and Scalable AI Agents



Anis Koubaa

Abstract Large Language Models (LLM) agents are proliferating across domains, yet current systems remain ad-hoc pipelines without operating system (OS)-level guarantees for scheduling, memory, real-time responsiveness, and end-to-end security. Today's agent architectures resemble the pre-OS era of computing—a chaos of duplicated solutions lacking fundamental abstractions for resource management, isolation, and coordination. Existing frameworks (e.g., tool-calling, Model Context Protocol, Agent-to-Agent messaging) address isolated aspects but lack a unified, security-by-design, latency-aware foundation suitable for enterprise and safety-critical deployments. This **position paper** introduces the conceptual **Agent Operating System (Agent-OS)** as a computational substrate for agentic workflows. We present this not as a system fully realizable today, but as an architectural North Star to guide the next decade of agent infrastructure research. We propose a unified requirements specification encompassing functional requirements (lifecycle, memory, tools, orchestration, observability, safety, governance) and non-functional properties (reliability, scalability, interoperability, compliance, real-time, security), extended with explicit latency classes—Hard Real-Time (HRT), Soft Real-Time (SRT), and Delay-Tolerant (DT). An abstract layered architecture is defined, comprising Kernel, Services, Agent Runtime, Orchestration, and User layers, with cross-cutting concerns for security, governance, and observability. Following a system engineering requirements-driven methodology, we (i) trace historical roots and survey emerging 2025 systems, (ii) synthesize requirements with latency taxonomies, (iii) formalize *Agent Contracts* for portability and enforcement, and (iv) map these to the layered architecture. The Agent-OS provides a blueprint for scalable, interoperable, and trustworthy agent deployment for next generation Agentic-AI powered smart cities, autonomous systems, and enterprise AI-even as full realization may require years of collective research effort.

Keywords Agent Operating Systems · Agentic AI · Large language models · Real-time systems · Multi-agent orchestration · Security-by-design

A. Koubaa (✉)
Alfaisal University, Riyadh, Saudi Arabia
e-mail: akoubaa@alfaisal.edu

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

A. Koubaa et al. (eds.), *Generative AI and Large Language Models: Opportunities, Challenges, and Applications*, Studies in Computational Intelligence 1287,
https://doi.org/10.1007/978-3-032-24013-2_4

Highlights

- **The Billion Agent Problem**: By 2030, billions of AI agents will require coordination—without an OS-level abstraction, we risk computational chaos at unprecedented scale.
- **Impossible Today, Essential Tomorrow**: Deterministic scheduling for stochastic models and 10 ms HRT guarantees for LLMs may seem unattainable, yet they define the research frontier we must solve.
- **From Bare Metal to OS**: Agent-OS lifts today's "bare metal" agent programming into managed, portable applications with guaranteed behaviors and resource isolation.
- **Security by Architecture**: A zero-trust microkernel ensures no agent executes without contracts, capabilities, and audit trails—making trust enforceable, not optional.
- **The Missing Layer**: Between foundation models and applications lies a void—Agent-OS defines the substrate to make agents as reliable as today's cloud services.

1 Introduction

The field of artificial intelligence is undergoing a marked shift from passive prediction to *agentic* systems capable of reasoning, planning, and acting in open environments [1]. Enabled by large language models (LLMs), such systems are moving beyond conversational interfaces to execute goal-directed, multi-step operations across heterogeneous digital and physical contexts [2]. This evolution exposes gaps in coordination, safety, and scalability: most agent stacks resemble "bare-metal" systems that force bespoke solutions for state management, tool access, concurrency, and security, creating duplication of effort, brittle deployments, and elevated risk [3].

An operating-system (OS) abstraction proved decisive in classical computing by virtualizing hardware and standardizing process, memory, and I/O management. Analogously, the *LLM as OS, Agents as Apps* vision posits model-centric kernels with language as the interface, motivating OS-like services for tools and memory [4]. This builds on decades of multi-agent systems (MAS) work that established facilitator/blackboard coordination (e.g., the Open Agent Architecture, OAA) and standardized communication (FIPA ACL) [5, 6]. Contemporary prototypes begin to realize this viewpoint: AIOS presents an LLM Agent Operating System with a kernel exposing unified calls and managers for context, memory, storage, tools, and access control [7]; KAOS explores management-role agents and shared resource scheduling on Kylin OS; and AgentStore composes heterogeneous agents via a meta-agent "marketplace," evaluating on OS-style benchmarks [8, 9]. In parallel, real-time OS research for autonomy (e.g., Siren) underscores determinism and bounded jitter as first-class properties increasingly relevant to interactive and embodied agents [10].

Despite this momentum, two structural gaps persist. First, the literature lacks a *requirements-driven* operating model that states functional and non-functional properties—reliability, performance, *security-by-design*, scalability, interoperability, and operability—in a portable, testable form suitable for enterprise, edge, and embedded deployments. Second, time semantics are seldom elevated to enforceable contracts: beyond latency optimizations or context interrupts, there is little formal distinction between hard real-time (HRT), soft real-time (SRT), and delay-tolerant (DT) regimes, each requiring specific scheduling, I/O, memory, networking, and acceptance-test policies. Moreover, while access control appears in recent systems, zero-trust primitives—capability-scoped tools, encrypted memory, and immutable/auditable traces—are not consistently treated as kernel responsibilities, despite documented risks in agent ecosystems [3].

Contributions. As a position paper, we chart the conceptual foundations for Agent-OS rather than implementation:

- **Requirements synthesis**: A prioritized baseline spanning functional requirements (agent lifecycle, context/memory, tool connectors, orchestration, observability, safety/governance, model management, interfaces, multitenancy) and non-functional requirements (reliability, performance, security-by-design, scalability, interoperability, operations).
- **Latency taxonomy**: Explicit latency classes—HRT/SRT/DT—with concrete SLO targets and class-tied policies for scheduling, I/O, memory, and acceptance tests, aligned with real-time autonomy research [10].
- **Layered architecture**: A five-layer model comprising Kernel, Resource and Service, Agent Runtime, Orchestration and Workflow, and User and Application planes with cross-cutting governance.
- **Research agenda**: A roadmap for standardization, positioning this specification as a portable blueprint that unifies conceptual visions [4] with emerging prototypes [7–9].

Realizing robust Agent-OS platforms is critical for complex, real-world domains. In autonomous systems, verifiable real-time guarantees are essential for safe operation under dynamic conditions [10]. In smart-city contexts, a principled OS layer can coordinate distributed agents managing infrastructure, logistics, and public services at scale. By grounding agent platforms in explicit requirements, time semantics, and security primitives, we aim to catalyze a transition from bespoke pipelines to interoperable, auditable, and scalable infrastructures suitable for rigorous engineering and eventual standardization.

Chapter organization. The remainder of this chapter is structured as follows. Section 2 surveys related work, tracing historical roots from multi-agent systems to contemporary LLM-based prototypes. Section 3 formalizes the requirements specification, including functional, non-functional, and latency class properties. Section 4 presents the layered conceptual architecture. Section 5 discusses standardization considerations. Section 6 concludes with challenges and a phased research agenda.

2 Background and Related Work

The concept of Agent Operating Systems (Agent-OS), while gaining recent traction due to the proliferation of Large Language Models (LLMs), draws from decades of research in multi-agent systems, evolving from early distributed rule-based AI frameworks to modern LLM-driven architectures. Historical roots trace back to the 1990s with the Open Agent Architecture (OAA) [5], which used a central blackboard for coordinating heterogeneous agents, emphasizing delegation and facilitation. Similarly, JACK Intelligent Agents [11] employed a Belief-Desire-Intention (BDI) model for rational agent behavior in Java-based environments. Platforms like JADE and middleware such as AgentScope [12] extended this to scalable development environments, emphasizing agent lifecycles and interoperability. These systems laid foundational abstractions for agent lifecycle management and communication, as seen in the FIPA Agent Communication Language (ACL) [6], which standardized message passing and negotiation semantics. However, pioneering efforts targeted symbolic or *rule-based agents*, focusing on service composition rather than today's *LLM-centric*, *tool-using*, *multi-modal agents* that must orchestrate models, data, UI actions, and environments with enterprise-grade observability and governance.

Conceptual proposals have bridged this gap by reimagining OS paradigms for AI. For instance, "LLM as OS, Agents as Apps" [4] envisions LLMs as kernels with natural language as the interface, positioning agents as applications atop this substrate. This idea influences recent prototypes like AgentStore [9], an "app store" for heterogeneous agents using a MetaAgent for task delegation, improving OSWorld benchmark performance from 11 to 24%. KAOS [8], built on Kylin OS, introduces management-role agents for resource scheduling and vertical collaboration, enhancing multi-agent efficiency in dynamic settings. MMAC-Copilot [13] extends this to multimodal domains, using collaborative chains to reduce hallucinations in GUI and game interactions, boosting GAIA benchmarks. Emerging 2025 works, such as Planet as a Brain [14], extend AIOS to distributed "AgentSites" for agent migration; Alita [15] enables autonomous reasoning via a generalist OS; Eliza [16] integrates blockchain for decentralized security; and surveys like Fundamentals of Agentic AI [1] and Large Model Agents [2] highlight cooperation paradigms. AgentScope [12] provides middleware roots for scalable simulations, while AI Agents Under Threat [3] critiques security vulnerabilities in agent ecosystems. A Survey on MLLM-based Agents [17] reviews device-use applications, referencing OS-level support for real-time interactions.

Taken together, these strands signal momentum toward OS-like abstractions, yet they also expose limitations: many works are preprint or product-driven with uneven empirical grounding, benchmarks rarely capture real-time and security constraints, standardization (MCP/A2A/OTel) and portability remain incomplete, and few proposals articulate a requirements-driven, latency-class- and security-by-design operating model suitable for audited, heterogeneous deployments.

On the other hand, Industrial efforts parallel these, with PwC Agent OS [18] acting as a "switchboard" for cross-platform orchestration, integrating GPT-5 and

clouds like AWS/Oracle. AG2 [19] focuses on the multi-agent lifecycle from DeepMind, while Microsoft's Copilot Runtime [20] exposes local models via Phi Silica, and Apple's Intelligence [21] formalizes app intents. Standardization protocols like MCP [22], A2A [23], and OpenTelemetry [24] enable tool access, inter-agent communication, and observability.

Critically, this literature reveals fragmentation: historical systems are outdated for LLMs, conceptual works lack implementation rigor, and prototypes prioritize efficiency over comprehensive governance. Real-time support is ad-hoc—e.g., Siren [10] targets autonomous systems but not AI agents—while security surveys expose misuse risks without unified solutions. Workflows like Yu and Schmid's framework [25] model roles but ignore latency diversity. Weyns et al. [26] define agent-based OS abstractly, yet miss modern scalability.

AIOS comparison. The most salient comparison is with AIOS [7], a pioneering LLM-specific OS with a three-layer architecture (application, kernel, hardware) and managers for scheduling, context, tools, and access. AIOS reports $\sim 2.1\times$ faster execution via unified system calls and an SDK, and introduces context-interrupt mechanisms for fairness. However, AIOS is primarily scoped to LLM agents, rather than a broader agentic ecosystem (e.g., multimodal environments and deep human-in-the-loop orchestration). Its requirements remain implicit, without a prioritized FR/NFR breakdown or data/governance schemas. Real-time is addressed through latency reductions but not formalized into classes (HRT/SRT/DT) with enforceable SLOs, limiting applicability to safety-critical settings. Human-In-The-Loop (HITL) support is basic (user confirmation of risky actions), and the protocol surface is runtime-internal, rather than aligned to open standards such as MCP and A2A. his exposes a key gap: no unified, requirements-driven architecture supports latency-differentiated, portable agents with zero-trust governance across domains.

Table 1 summarizes the similarities and differences between the visionary "LLM as OS" proposal, the academic *AIOS* prototype, and our generalized *Agent-OS*, highlighting scope, architectural layers, requirements detail, real-time support, HITL integration, and standardization philosophy.

Our contribution fills this gap with a five-layer Agent-OS model and an explicit, requirements-driven specification (FR/NFR, including NFR7 for real-time), an Agent Contract for portability, and security-by-design as a kernel primitive (RBAC, capability-scoped tools, encrypted memory, auditable traces), yielding a scalable, interoperable framework for real-time AI in smart-city and enterprise contexts.

3 Requirements for an Agent Operating System

An Agent Operating System (Agent-OS) elevates agent-centric computation to a first-class platform abstraction, much as traditional operating systems virtualize hardware. Our specification is system engineering requirements-driven and grounded in multi-agent systems (facilitation, lifecycles, messaging) [5, 6, 25, 26], LLM-era systems

Table 1 Compact comparison of *LLM as OS* (conceptual), *AIOS* (LLM Agent OS), and *Agent-OS* (this work). Abbrev.: FR/NFR = functional/non-functional requirements; HRT/SRT/DT = hard/soft real-time, delay-tolerant; HITL = human-in-the-loop; MCP/A2A/OTel = Model Context Protocol/Agent-to-Agent/OpenTelemetry

Aspect	*LLM as OS*	*AIOS*	*Agent-OS* (ours)
Nature/Scope	Visionary concept: LLM as kernel; high-level OS analogies	Academic architecture for LLM agents; efficiency-focused	Generalized OS for agents: models + tools + environments + HITL
Architecture	Analogies (kernel/memory/fs/tools); no layered stack	Three layers: Application/Kernel/Hardware; kernel managers	Five layers: Kernel; Resource and Service; Agent Runtime; Orchestration and Workflow; User and App; cross-cutting security/governance
Requirements	No formal FR/NFR spec.	Empirical gains reported; no prioritized FR/NFR in public sources	Prioritized spec (FR, NFR, DR, ER, DEP, GOV) as a portable blueprint
Real-time	Not explicit	Context/time-aware interrupts; latency reduction, no taxonomy	Formal HRT/SRT/DT classes; class-tied scheduler/I/O/memory/networking and acceptance tests
HITL and Safety	Human–agent interaction discussed conceptually	User intervention for risky ops; access manager	HITL in workflows + console; zero-trust RBAC, capability-scoped tools, encrypted memory, auditable traces
Standards/Protocols	Calls for standards	Unified calls via SDK inside runtime	MCP as; A2A for agent IPC
Key components	LLM (kernel), context window (memory), external storage, tools, prompts, agents	Kernel managers (scheduler/context/memory/storage/tool/access), LLM core, SDK, apps	Kernel (scheduler/context/action); services (memory/tool); runtime; orchestration; user layer; security/governance

(AIOS, KAOS, AgentStore, MMAC-Copilot) [7–9, 13], real-time OS insights [10], and surveys on large-model agents and risks [2, 3].

In this section, we formalize the requirements for an Agent-OS, detailing functional and non-functional properties, introducing three latency classes—Hard Real-Time (HRT), Soft Real-Time (SRT), and Delay-Tolerant (DT)—with service-level

objectives (SLOs), and stating measurable acceptance criteria that form a portable contract between application builders and platform providers. We organize the requirements into *Functional* (FR), *Non-Functional* (NFR), and *Latency Classes*. FRs cover lifecycle; context, memory, and knowledge; tools and environment access; orchestration; observability; safety and governance; model, cost, and placement management; interfaces and human-in-the-loop (HITL) surfaces; and multitenancy—each with stated *interfaces* (what is exposed) and *acceptance* (how verified). NFRs set targets for reliability, performance, security-by-design, scalability, interoperability, operations, and compliance. The latency classes attach concrete SLOs and map to robotics (HRT), speech/video assistants (SRT), and batch analytics (DT).

3.1 Functional Requirements (FR)

The functional requirements define what an Agent Operating System (Agent-OS) must provide to make agents reliable, scalable, and safe across diverse environments. Each requirement is expressed in plain terms, describes what the OS must offer, and sets measurable acceptance criteria, with examples that connect to LLM- and agent-centric applications (e.g., prompts, function-calling tools, vector stores, and RAG).

FR1—Agent lifecycle and scheduling.

Plain goal: Agents are easy to start, pause, resume, checkpoint, migrate (where allowed), and stop; many agents remain fair and responsive under load.

What the OS provides: A unified lifecycle manager for agent definitions (prompt/policy, plan, tool scopes) with APIs for `spawn/pause/resume/kill`; deterministic checkpoints that capture prompt state, scratchpad, tool cursors, and the context window; quotas and admission control; schedulers aware of token-rate budgets and model-gateway capacity (tokens per minute/second, *TPM/TPS*).

Why it matters: Prevents duplicated side effects after failures and preserves user trust during bursts.

Good enough if: $\geq$99% of agents recover within 60 s with no duplicate side effects; quotas enforced; a *turn* (prompt $\rightarrow$ completion $\rightarrow$ tools) replays deterministically under the same policy/seed.

LLM signals: Turn replay fidelity; tokenizer state saved in checkpoints; TPM/TPS budgeting per tenant/agent.

Example: A city-services assistant resumes after a reboot and finishes filing the same ticket without re-submitting the form.

FR2—Context, memory, and knowledge.

Plain goal: Agents have short-term working memory and long-term knowledge they can recall and cite.

What the OS provides: A context manager for LLM context windows (token-budget enforcement, sliding/summarization), episodic logs, and semantic long-term memory via pluggable backends (vector stores for embeddings, key–value caches,

relational/object databases). Built-in RAG pipelines (retrieve → rank → ground) with citation/provenance and per-tenant namespaces.

Why it matters: Grounded recall reduces hallucinations and enables audits.

Good enough if: Context never exceeds `max_context_tokens`; RAG recall @K and citation coverage meet targets; memory growth stays within retention/redaction policies.

LLM signals: `max_context_tokens`, RAG hit-rate, attribution coverage, summarization drift checks.

Example: A planning agent embeds incident reports into a vector store and retrieves top-*k* sources with citations before proposing actions.

FR3—Tools and environments (action layer).

Plain goal: Agents safely invoke tools (APIs, apps, filesystems, IoT/robotics) with the right permissions.

What the OS provides: A typed tool registry with JSON/function-calling schema validation, capability-scoped permissions, secret redaction, and sandboxed execution; adapters for web/desktop automation, data platforms, and robot drivers; simulation mode for dry-runs.

Why it matters: Tool misuse is the main path to harmful side effects.

Good enough if: ≥99% schema-conformant calls; out-of-scope tools blocked and logged; tool sequences replayable for audit.

LLM signals: Function-call schema pass rate; scope-denial count; redaction events; per-tool round-trip latency.

Example: A maintenance agent reads a BACnet sensor and files a ticket, but is blocked from initiating payments it lacks scope for.

FR4—Orchestration and multi-agent coordination.

Plain goal: Coordinate complex jobs across agents and workflows; include people when needed.

What the OS provides: A workflow engine (DAG or state machine), planners/routers for agent-to-agent delegation, retries and idempotency with compensation, and human-in-the-loop (HITL) gates with prompt templates for approvals/corrections.

Why it matters: Long tasks fail without retries and human checkpoints.

Good enough if: Exactly-once semantics across retries; HITL actions complete within SLOs; delegation preserves tool/policy scopes end-to-end.

LLM signals: Prompt/plan template versions; HITL latency; delegation audit trail.

Example: A permitting workflow pauses for approval, then resumes a "draw-contract" agent with the approved prompt template.

FR5—Observability.

Plain goal: Operators can see what agents did, why, and how they performed.

What the OS provides: Structured logs, traces, and metrics with correlation IDs per task/turn; lineage from inputs/prompts through tool calls to outputs; LLM metrics (first-token onset, *tokens/sec*, prompt/completion token counts) and per-tool latencies; retrieval diagnostics (hit-rate, source hashes).

Why it matters: Debuggability and compliance depend on end-to-end visibility.

Good enough if: >99% of steps have trace spans; audits can reconstruct a run (with redactions).

LLM signals: `onset_ms`, tokens/sec, token counts, RAG source list, tool timings.

Example: Investigators replay a financial agent's report with full provenance of RAG sources and tool timings.

FR6—Safety, security, and governance.

Plain goal: Prevent misuse, enforce least privilege, and record actions for accountability.

What the OS provides: Zero-trust RBAC (role-based access control) and capability scoping for tools/data; consent prompts for high-risk actions; tamper-evident audit logs; strict tenant isolation; defenses against prompt injection/jailbreaks and data exfiltration (content filters, allowlists, output redaction).

Why it matters: LLMs can be steered by crafted prompts; audits must hold up.

Good enough if: Unauthorized actions are blocked and logged; audit integrity is provable (hash/attestation); injection deflection meets target.

LLM signals: Injection block-rate; consent-gated action count; audit hash chain health.

Example: A data agent cannot paste credentials from its prompt into outbound emails; attempted exfiltration is blocked and recorded.

Threat categories addressed:

- **Prompt injection**: Input sanitization, output filtering, sandboxed execution environments
- **Data exfiltration**: DLP policies, output redaction, capability-scoped tool access
- **Privilege escalation**: Zero-trust RBAC, consent gates for sensitive actions, tamper-evident audit trails

Figure 1 visualizes this defense-in-depth strategy. Unlike traditional systems that rely on perimeter security, Agent-OS enforces a zero-trust pipeline where every agent request undergoes input sanitization, capability scoping, and policy verification before execution is granted.

FR7—Model, cost, and placement management.

Plain goal: Automatically choose the right model/back end, balancing cost, latency, accuracy, and privacy.

What the OS provides: A model catalog (local/edge/cloud; base or fine-tuned; adapter/LoRA—low-rank adaptation—variants) with policy-based routing by context length, safety tier, and data residency; budgets/metering (tokens, currency) and placement controls (e.g., keep PHI—protected health information—on-prem); consistent tokenization contracts.

Why it matters: Cost and privacy constraints vary by tenant and task.

Good enough if: Routing complies with policy; budgets enforced with alerts; no privacy-class violations.

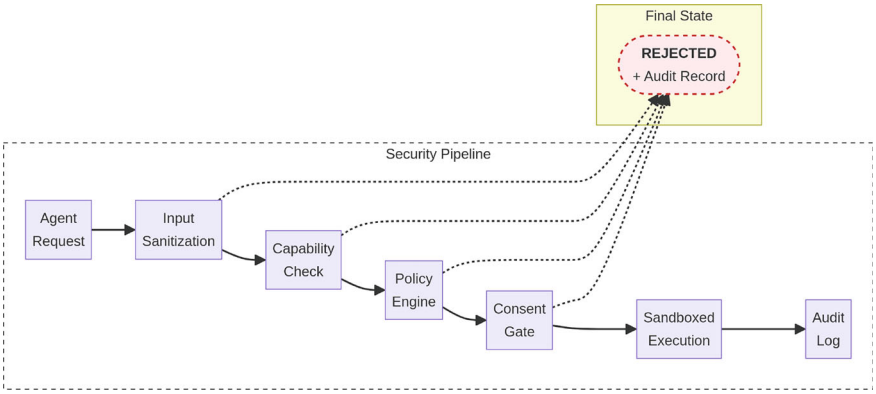


Fig. 1 Zero-Trust Security Flow: Each agent request passes through multiple validation stages—input sanitization, capability checking, policy enforcement, and consent gates—before sandboxed execution. Any failure results in rejection with an immutable audit record

LLM signals: Model selection logs, budget adherence, privacy/placement decisions.

Example: A medical chatbot uses an on-prem 8B model for PHI and escalates to a 70B cloud model only when context exceeds local limits (with de-identified snippets).

FR8—Interfaces and HITL surfaces.

Plain goal: Agents are easy to integrate and supervise.

What the OS provides: Programmatic SDK/REST; a conversational "chat-as-shell"; multimodal I/O (speech, vision, screen, files) with streaming partials and *barge-in* (interrupt); reviewer consoles to edit prompts, plans, and tool scopes.

Why it matters: Human oversight reduces risk and improves UX.

Good enough if: Interactive turns meet onset/turn SLOs; reviewers intervene within SLA with full context and diffs.

LLM signals: First-token onset, turn latency, reviewer action latency, diff history.

Example: A video assistant streams captions within 200 ms first-token onset while a supervisor amends the prompt template live.

FR9—Multitenancy and portability.

Plain goal: Many teams safely share the platform; agents move across compliant runtimes without code changes.

What the OS provides: Strong isolation of data/logs/memory/keys per tenant; organizational policy inheritance; an exportable *Agent Contract* (prompt/policy templates, tool schema scopes, model/placement policy, budgets) for portability.

Why it matters: Portability reduces lock-in and eases cross-site deployments.

Good enough if: Isolation tests always pass; the same Agent Contract runs unchanged across reference runtimes with identical behavior/SLOs.

LLM signals: Contract fields (prompt templates, tool scopes, routing policy, budgets) validate across runtimes.

Example: A government agency exports an Agent Contract from a private cluster and replays it on a city-scale cloud deployment, preserving tool scopes and RAG policies.

3.2 Non-functional Requirements (NFR)—Accessible, LLM-Aware

NFR1—Reliability and availability.

Target: The platform control plane (the service that schedules and supervises agents) provides at least 99.9% monthly availability; recovery time objective (RTO) is $\leq$60 s with idempotent replay of the last *turn* (prompt, model completion, and tool calls); automatic failover to a healthy model endpoint without losing the conversation state.

Why it matters: Conversational services and autonomous loops should not stall when a controller or model endpoint fails.

LLM signals: Replays preserve the context window and tokenizer state; side effects (e.g., tool calls) are not duplicated on retry.

NFR2—Performance and efficiency.

Target: Orchestration overhead at the 95th percentile (P95) is $\leq$200 ms (excluding model and tool execution). For interactive use, *first-token onset* (time to the first streamed token) is $<$1 s; for batch jobs, processing rate reaches at least 90% of the baseline tokens-per-second per core when batching is enabled.

Why it matters: People notice delays in conversations; background analytics must deliver good throughput and low cost.

LLM signals: Record *onset time*, *tokens per second*, and prompt/completion token counts; track RAG cache-hit rates and round-trip time for function/tool calls.

NFR3—Security by design.

Target: Single sign-on (SSO) with OpenID Connect, managed secrets, encryption in transit and at rest, capability-scoped tool access, tamper-evident audit logs, data-loss-prevention (DLP) and redaction for personally identifiable information (PII), and a prompt-injection/jailbreak *deflection rate* $\geq$95% on a standard test suite.

Why it matters: LLMs can be steered by crafted prompts; sensitive data must not leak.

LLM signals: Strict validation of function-calling/JSON schemas; input/output safety filters; signed tool manifests and proof that each call stayed within its declared scope.

NFR4—Scalability and elasticity.

Target: Horizontal scale-out of agent runtimes and model gateways; autoscaling based on queue depth and measured tokens-per-second; graceful degradation with back-pressure under load.

Why it matters: Real-world traffic is bursty (events, end-of-month reports, civic surges).

LLM signals: Shardable vector stores for embeddings; limits on concurrent sessions; fair token-rate limits per tenant and per agent.

NFR5—Interoperability and extensibility.

Target: Plugin/adapter model with versioned contracts; protocol-agnostic interfaces for tools and telemetry; backward compatibility for agent "contracts".

Why it matters: Avoids vendor lock-in and supports mixed fleets of local and cloud models.

LLM signals: Model Context Protocol (MCP)–style tool contracts; adapters for different model providers (open-source and commercial); consistent function-calling schemas; portable prompt/plan templates.

NFR6—Operations.

Target: Health probes, circuit breakers, staged "canary" releases, documented runbooks, and *policy dry-runs*; service-level objective (SLO) dashboards and error budgets with *continuous evaluation* (CE) pipelines.

Why it matters: Models, prompts, and data drift over time; safe rollouts require guardrails.

LLM signals: CE on representative tasks (e.g., RAG Q&A, tool-use flows); automatic alerts on regressions in accuracy, safety, or cost; "shadow" routing to trial a new model without affecting users.

NFR7—Real-time responsiveness.

Target: Meet class-specific objectives—deadlines and jitter for Hard Real-Time (HRT), onset and full-turn latency for Soft Real-Time (SRT), and throughput/cost SLAs for Delay-Tolerant (DT)—and automatically reject deployments that fail a pre-deployment schedulability check.

Why it matters: Robots need determinism; people expect instant feedback; batch jobs must be efficient.

LLM signals: HRT pins models on device and avoids garbage collection in control loops; SRT enforces first-token and turn-time SLOs with streaming partials and "barge-in"; DT optimizes cost-per-token with resumable checkpoints.

NFR8—Compliance and governance.

Target: Retention and redaction controls, data-residency modes, end-to-end lineage for audits, separation of duties, and policy attestation.

Why it matters: Public-sector, healthcare, and enterprise settings require provable control and traceability.

LLM signals: Source attribution for generated claims; per-turn provenance (retrieved documents, embeddings, and tools used); audit logs are hashed/attested; "privacy classes" drive model choice and data placement.

3.3 Latency Classes (Time Semantics)

We classify agent workloads into three latency classes so the OS can apply the right scheduling, I/O, memory, and testing policies. We use the following terms: *deadline* (time by which a task must finish), *deadline miss* (task finishes after its deadline), *jitter* (variation in completion/onset time), *WCET* (worst-case execution time), *onset latency* (time to first token/response), and *full-turn latency* (time to complete an interaction).

Hard Real-Time (HRT)—deterministic and safety-critical.

Definition: Correctness depends on meeting *every* deadline; any miss is a system failure.

Why it matters: Some LLM-driven agents embed models inside closed-loop control pipelines (e.g., drone navigation, medical devices, factory lines). A late token or missed tool call could break synchronization and lead to unsafe or invalid actions.

Typical use cases: Token-bounded inference loops for robotics, "LLM + control policy" hybrids where the LLM interprets high-level goals, and real-time safety checks on generated actions.

Targets (SLOs): 1–20 ms execution slices; jitter $\leq$5 ms; zero deadline misses.

OS policies: EDF/RM scheduling; CPU/GPU/NPU reservations; pinned threads; fixed memory arenas (no GC); lock-free queues.

LLM anchors: Bounded context templates, capped output tokens, pre-validated tool paths, and lightweight local models for deterministic latency.

Acceptance: WCET profiling and 24h burn-in tests with 0 deadline misses [10].

Example: A warehouse robot uses an LLM agent to interpret human commands ("pick from aisle 3") but runs them through an HRT safety filter that must always respond in $<$10 ms before execution. *Crucially, the LLM inference itself is never on the critical HRT path*; only the safety guards and control logic are subject to hard real-time enforcement.

> *Research note*: Achieving true HRT guarantees for LLM inference remains an **open research challenge** due to the stochastic nature of model outputs. The targets above apply primarily to *system-level components* (scheduling, policy checks, I/O); LLM inference itself is bounded via context/output caps but not fully deterministic. Reconciling probabilistic models with deterministic deadlines is a key research frontier this vision paper aims to catalyze.

Soft Real-Time (SRT)—interactive and perceptual.

Definition: Occasional deadline misses are tolerable, but user-perceived responsiveness is crucial.

Why it matters: In LLM copilots (speech/video/chat), humans abandon interactions if responses feel delayed or jittery. First-token onset and smooth streaming are as important as accuracy.

Typical use cases: Conversational assistants, meeting summarizers, screen copilots, live captioning, multi-modal copilots.

Targets (SLOs): Onset (time-to-first-token) 150–300 ms; full-turn 0.8–1.2 s; jitter P95 $\leq$20%.

OS policies: Priority-queue scheduling; streaming partial completions; barge-in support; adaptive buffering; rolling context caches.

LLM anchors: TTFT (time-to-first-token), tokens/sec throughput, function-call round-trip times, RAG latency budgets ($\leq$150 ms retrieval, $\leq$100 ms grounding).

Acceptance: Turn-level latency distributions; MOS for quality; network impairment tests.

Example: A real-time translator must show the first translated words within 200 ms while continuing to stream the rest of the sentence smoothly.

Delay-Tolerant (DT)—throughput and cost first.

Definition: Deadlines are coarse (minutes–hours); efficiency and completion rate dominate responsiveness.

Why it matters: LLM workloads like RAG sweeps, indexing, retraining, or batch question-answering do not need instant responses. The priority is minimizing cost-per-token and maximizing throughput while ensuring reproducibility.

Typical use cases: Overnight batch summarization, legal document RAG indexing, dataset labeling, long code refactoring pipelines.

Targets (SLOs): SLAs in minutes–hours; maximize throughput (tokens/sec and tasks/hour) per dollar while meeting accuracy/completeness targets.

OS policies: Queue-based best-effort scheduling; preemptible/low-priority workers; aggressive batching/prefetch; resumable checkpoints; idempotent retries.

LLM anchors: Embedding throughput, batch tokens/sec, RAG cache hit-rates, checkpoint integrity, retry idempotency.

Acceptance: Throughput–cost curves; checkpoint/resume fidelity; no duplicate tool calls across retries.

Example: A compliance office runs an overnight RAG sweep of 10M documents; runtime batches maximize tokens/sec while ensuring results can resume after preemption.

Choosing a class. Use **Hard Real-Time (HRT)** for agents where *any missed deadline means failure*—these are safety- or stability-critical (e.g., control loops or rapid safety checks). Use **Soft Real-Time (SRT)** when *humans are in the loop* and the priority is smooth, responsive interaction (e.g., speech, chat, or video copilots). Use **Delay-Tolerant (DT)** for *long-running or batch workloads* that can trade speed for throughput and cost efficiency (e.g., RAG indexing, bulk summarization). The Agent-OS enforces tailored scheduling, I/O, memory, and verification policies for each class, ensuring that applications receive the performance guarantees appropriate to their domain. All latency targets assume class-appropriate hardware provisioning (e.g., edge NPUs for HRT, local accelerators for SRT, batch GPU clusters for DT) (Table 2).

As illustrated in Fig. 2, the classification process is driven by the consequence of failure. Determine the class by asking: *Does a missed deadline cause system*

Table 2 Latency classes summary

Class	SLO targets	Scheduling policy	Typical use cases
HRT	1–20 ms, jitter ≤5 ms	EDF/RM, pinned threads	Robotics, safety checks
SRT	Onset 150–300 ms	Priority queues, streaming	Chat, speech, video
DT	Minutes–hours	Best-effort, batching	RAG indexing, analytics

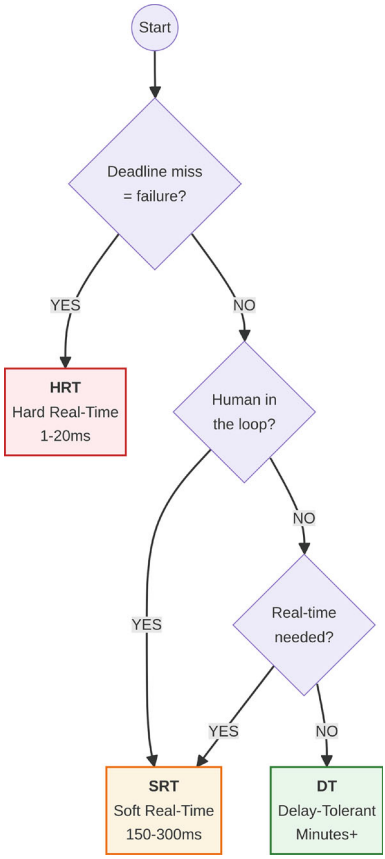


Fig. 2 Latency Class Decision Flowchart: Practitioners can use this decision tree to select the appropriate latency class for their agent workload based on deadline criticality, human interaction requirements, and real-time response needs

failure? (HRT), *Is a human waiting for the response?* (SRT), or *Is throughput the primary metric?* (DT). This logical separation ensures resources are not wasted on unnecessary guarantees.

3.4 Agent Contract (Excerpt)

Purpose. The *Agent Contract* serves as the declarative definition for an agent. It is a portable, machine-readable specification that outlines an agent's operational parameters, resource needs, behavioral guardrails, capabilities, policies, latency class, and service-level objectives (SLOs). As a critical component of the ecosystem, it enables the Agent-OS to (i) schedule and place the agent correctly, (ii) enforce security and governance, (iii) provision memory, RAG, and tooling consistently, and (iv) ensure the same behavior across compliant runtimes, guaranteeing both performance and compliance.

Minimal viable contract (illustrative Canonical schema).

```
apiVersion: agentos/v0.2
kind: AgentContract
name: doc-rag-planner
class:   { latency: SRT, slo: { onset_ms: 250, turn_ms: 1000,
 jitter_p95_pct: 20 } }
capabilities: [ "web.fetch", "fs.read", "summarize" ]
compute: { cpu: "1", mem: "2GiB" }
modelPolicy: { allow: ["local/8B","cloud/70B"], max_context_tok
 ens: 16000 }
memory: { namespace: "city-planning", retention_days: 30,
          rag: { top_k: 8, require_grounding: true } }
security: { consent_for: ["fs.write","payment.charge"] }
observability: { tracing: "opentelemetry", log_fields:
 ["prompt","sources"] }
```

HRT variant (one-line change):

```
class: { latency: HRT, slo: { deadline_ms: 10, jitter_ms: 5 } }
```

How the OS uses it (at a glance).

- **Real-time enforcement (FR1, NFR7)**. The declared `class.latency` selects the scheduler and checks: **HRT** → EDF/RM with deadline/jitter admission tests; **SRT** → priority queues, streaming, barge-in; **DT** → best-effort queues. Unschedulable deployments are rejected at admit time.
- **Tools (FR3, FR6).** `capabilities` is the allowlist; everything else is blocked or requires consent in `security.consent_for`.
- **Memory/RAG (FR2).** `memory` fixes namespace, retention, and grounding rules; citations are required when `require_grounding=true`.
- **Models/placement (FR7).** `modelPolicy` constrains model choice and context limits; routing enforces these when composing prompts/completions.

– **Compute budgets (FR1/7).** `compute` informs admission control and prevents token/CPU starvation.
– **Audit/telemetry (FR5, NFR8).** `observability` ensures prompts, sources, and tool hashes are traceable for audits.

Why this helps. The contract acts like an "ABI for agents," coupling **real-time class** (HRT/SRT/DT) with concrete SLOs (deadlines, onset/turn, throughput) so the Agent-OS can *prove* schedulability up front and *enforce* class-specific policies at runtime—while keeping behavior portable across vendors and clusters.

4 Conceptual Architecture

We propose a layered architecture for an **Agent Operating System (Agent-OS)** that treats *models*, *tools*, *environments*, and *human-in-the-loop (HITL)* interactions as **first-class, schedulable resources**. The design is guided by two complementary goals: **systems rigor** and **ecosystem portability**.

On the systems side, Agent-OS adopts a **microkernel-style control plane** that enforces *real-time guarantees*, *admission control*, and *policy checks*. Responsiveness is explicitly managed through three latency classes—**hard real-time (HRT)**, **soft real-time (SRT)**, and **deferred-time (DT)**—each associated with **service-level objectives (SLOs)** and validated via *pre-deployment schedulability tests* [10].

On the ecosystem side, the architecture embraces **open and interoperable standards**. Tool interfaces follow **MCP-style descriptors**, inter-agent communication is enabled via an **A2A-like message bus**, and observability is integrated with **Open-Telemetry (OTel)** to support unified tracing across heterogeneous components.

A central abstraction in this design is the **Agent Contract**, which serves as an **application binary interface (ABI)** for agents. The contract binds an agent's declared *capabilities*, *latency class*, *SLOs*, *memory/model policies*, and *resource budgets* to its runtime execution profile. This provides both **predictability** for the system and **portability** for agents across deployments.

4.1 Design Principles

Keep the kernel minimal (microkernel). The secure "core" of the Agent-OS should stay small and easy to audit. Only the essentials live in the kernel: checking whether an agent can be admitted, scheduling according to latency class, managing minimal context, and enforcing policies. All richer services—such as memory/RAG, tool use, model routing, agent-to-agent messaging, and telemetry—are implemented outside the kernel as modular services with versioned contracts. This separation makes the system both safer and more flexible.

Schedule by class of workload. Different agents have different timing needs: some need hard real-time guarantees (HRT), some need interactive smoothness (SRT), and some can run in the background with a focus on cost efficiency (DT). The kernel enforces these service-level objectives (SLOs) end-to-end by picking the right scheduler (e.g., EDF/RM for HRT, priority queues for SRT, best-effort for DT), reserving resources, and checking each agent's declared contract before it is admitted.

Zero-trust execution. Agents only get the exact data and tools they declare in their contract. Every call is checked against policy and logged. Sensitive or high-risk actions always require explicit human approval. This ensures safety and accountability, even if an agent is compromised or misconfigured.

Portable and open interfaces. Agents should run on different runtimes without rewriting. To achieve this, the OS uses open standards and versioned contracts: tools are described in schema-based formats (MCP-like), agents talk to each other through structured messages (A2A-style), and all activities are traced in standard telemetry formats (OTel). If an interface changes, it is versioned so old agents continue to work. This guarantees portability, interoperability, and long-term stability.

4.2 Layered Model

The proposed Agent-OS employs a five-layer architecture that parallels classical operating systems while targeting agentic workloads, as illustrated in Fig. 3. Following the principle that each layer should do one job well with a stable interface, the system uses a machine-readable *Agent Contract* as its single source of truth—specifying capabilities, data/tool access, latency requirements (HRT/SRT/DT), and budget/policy constraints across all layers.

The architecture follows an "hourglass" design (Fig. 3), where the Kernel acts as the narrow waist, enforcing policy and scheduling for all interactions between high-level user intents and low-level services. This decoupling allows the services and orchestration layers to evolve independently while maintaining a stable enforcement point.

The *User and Application* layer defines contracts and task objectives, which the *Workflow and Orchestration* layer decomposes into executable steps with per-step contracts delegated to specialized agents. The *Agent Runtime* instantiates these agents, maintains conversation state, and creates checkpoints for recovery. Acting as the system's enforcement mechanism, the *Kernel* validates each step against its contract, manages real-time scheduling, and mediates all resource access. The *Services* layer implements concrete operations—memory/knowledge management (embeddings, indexing, retrieval), tool execution via MCP, model routing, and inter-agent messaging—while *Observability* captures OpenTelemetry traces and lineage data that flows upward for auditing and optimization. Table 3 maps these components to their classical OS counterparts, demonstrating how traditional OS concepts extend naturally to agent coordination.

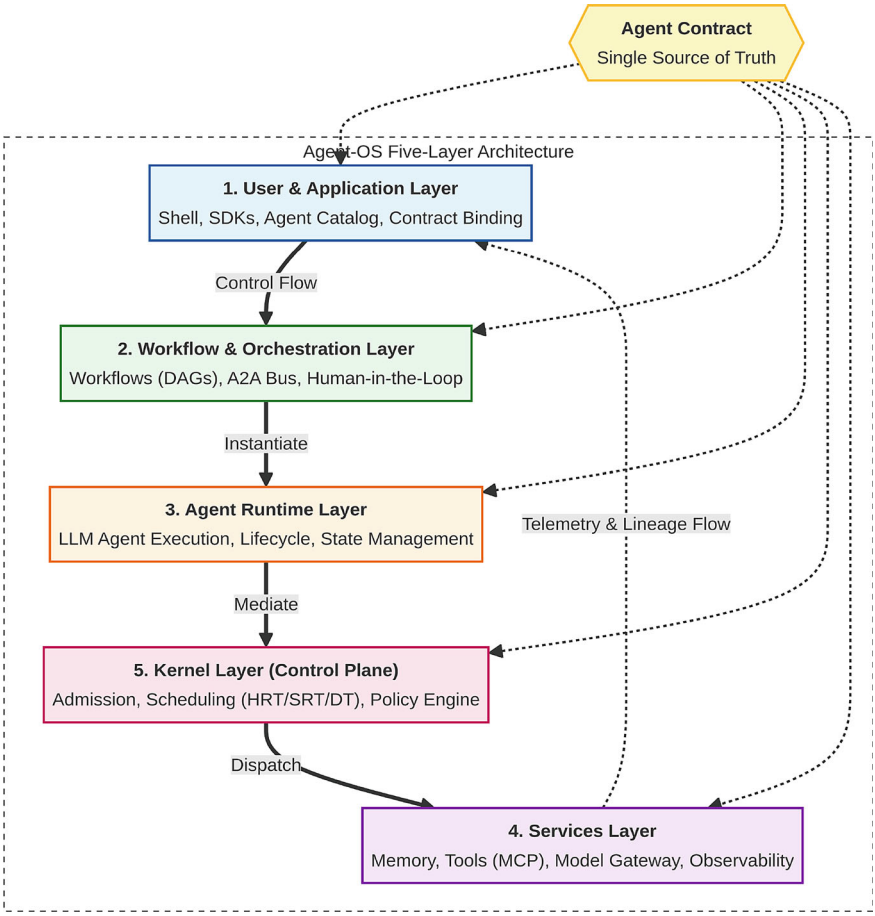


Fig. 3 Agent-OS Five-Layer Architecture: The Agent Contract serves as the single source of truth, binding capabilities, latency class, and policies across all layers. Control flows downward from User to Services via the Kernel; telemetry and lineage flow upward for observability

(1) ***User and Application layer*** This is where end-users and upstream systems invoke the Agent-OS ecosystem (Fig. 3, top). Users may type or speak requests in a natural-language shell and developers integrate via SDKs/REST. *LLM agents are selected here as first-class application components*: an *Agent Catalog* exposes role agents (e.g., "city-services assistant", "RAG summarizer") packaged with default *Agent Contracts* (capabilities, latency class, budgets, policies, prompt/plan templates). Each request is bound to a concrete contract—either the catalog default or an application override—and this binding travels with the request to drive scheduling, security, and routing in lower layers. This layer emphasizes usability and portability; enforcement and timing are delegated downstream.

Table 3 Analogy between classical OS and agent-OS components

Classical OS	Role	Agent-OS analogue
Process/thread	Schedulable unit	Agent/turn (contract-bound execution)
System calls	Kernel entry points	MCP-like tool calls and kernel APIs (`checkpoint()`, `invoke_tool()`)
IPC/sockets	Inter-process comms	A2A bus (typed messages, conversations)
Scheduler (CFS/RT)	CPU time allocation	Class-aware schedulers (EDF/RM, priority, best-effort)
Virtual memory	Address spaces	Context/memory service (LLM window, vector/KV stores)
File system	Persistent storage	Knowledge store with provenance (RAG artifacts)
Device drivers	Hardware adapters	Connectors (web/desktop/IoT/robotics; simulation)
Init/system	Service orchestration	Workflow engine (DAG/state machine, HITL)
Package manager	Install/update	Model/tool catalog (routing, versions, safety tiers)
SELinux/AppArmor /var/log + tracing	Policy enforcement Diagnostics	Policy engine (capabilities, consent, audit) OTel telemetry + lineage (prompts, tools, sources)

(2) ***Workflow and Orchestration layer*** User intents are compiled into *agentic workflows*: DAGs/state machines whose *nodes are LLM-agent invocations* and whose edges carry typed data and control. The orchestrator selects specialists from the Agent Catalog, binds each step to an *Agent Contract*, and composes them into a plan (e.g., *Planner*→*BrowserAgent*→*DocRAGAgent*→*Writer*). Steps can invoke an agent either *as a service* (stateless call with inputs/outputs) or *as an instance* (stateful session). *Human-in-the-loop* gates are inserted at risky transitions according to contract policy. Inter-agent delegation uses the Agent-to-Agent (A2A) bus with typed messages (conversation IDs, performatives), enabling marketplace, hierarchical, or blackboard patterns.

Dependencies: Orchestration *instantiates or calls* agents via the Runtime; any step that hits data, tools, or models must cross the Kernel (policy/scheduling) before reaching Services.

(3) ***Agent Runtime layer*** This is where *LLM agents reside and execute*. An agent is materialized from its *Agent Contract* (capabilities, latency class/SLOs, model/memory policy, budgets, prompt/plan templates). The Runtime provides life-

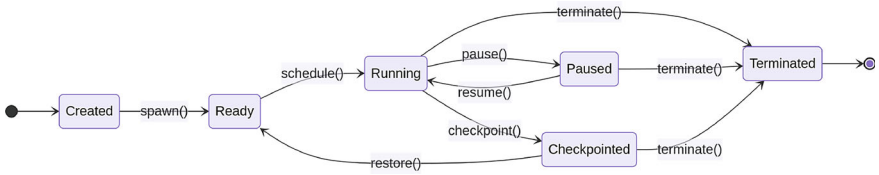


Fig. 4 Agent Lifecycle State Diagram: Agents transition through well-defined states managed by the Runtime. The lifecycle primitives (`spawn`, `pause`, `resume`, `checkpoint`, `terminate`) enable fault-tolerant execution and migration across deployments

cycle primitives (`spawn/pause/resume/checkpoint/terminate`) and maintains per-turn state (prompt, tokenizer cursor, scratchpad, pending tool cursors). It supports (i) single-shot calls (agent-as-service), (ii) stateful sessions for interactive tasks, (iii) batch pipelines, and (iv) small guarded control loops. Checkpoints are deterministic at the *semantic level* (seed, policy, tool sequence) to enable replay after faults or migration, acknowledging that bit-exact reproduction may vary across model backends. *Agents can also be exposed to others "as tools"*: the Runtime registers an agent's service interface in the Tool registry so that another agent can invoke it via standard function-calling semantics. All external actions (tools, memory, model calls) are made through kernel-mediated invocations to ensure contract SLOs and scopes are enforced.

contract SLOs and scopes are enforced.

This explicit state model, shown in Fig. 4, is not merely for lifecycle management but is the foundation for fault tolerance and mobility. By formalizing the `Checkpointed` state, Agent-OS allows running agents to be serialized and migrated to different hardware (e.g., from edge to cloud) or restored after a crash without losing conversation context.

(4) ***Services layer*** Agents consume shared services behind stable, replaceable APIs. *Memory*, also referred as Knowledge base, exposes a single API over pluggable backends (vector/KV/relational/object stores) and provides core primitives for *embedding, indexing, retrieval, reranking*, and *citation/provenance* with per-tenant namespaces, retention, and redaction. *Retrieval-Augmented Generation (RAG)* is supported as one *composition pattern* (retrieve → generate → verify/attribute), rather than being the mechanism itself; the same primitives also serve planning, tool selection, and analytics.

The *Tool service* maintains an MCP-style typed registry for safe actions over apps, data, and devices; validates JSON/function-call schemas; scrubs secrets; and executes tools in sandboxes via adapters for web/desktop automation, data platforms, and robotics/IoT (with a simulation mode for dry runs). *Agents published "as tools"* appear here as well, enabling *agent-as-a-service* composition without bypassing policy.

The *A2A bus* transports typed inter-agent messages (conversation IDs, performatives) for delegation and coordination. The *Model gateway* routes across local/edge/cloud models (base, fine-tuned, adapter/LoRA variants), enforces place-

ment/privacy (e.g., PHI on-premises), preserves tokenization consistency and max-context limits, and reports LLM metrics such as time-to-first-token (TTFT) and tokens/sec.

Observability exports OpenTelemetry (OTel) traces/logs/metrics and maintains end-to-end lineage from prompts and retrieved sources to tool arguments and outputs.

This layer maintains a dependency with the Kernel layer as Services call the Kernel's policy engine for capability checks and consent gates, and expose narrow "driver-like" interfaces so implementations can be swapped without breaking contracts.

(5) ***Kernel layer*** The Kernel is the small, trusted **control plane** on the hot path of all sensitive operations. ***Admission control*** validates each Agent Contract (latency class/SLOs, compute reservations, capabilities) and rejects unschedulable or under-scoped deployments a priori. ***Class-aware scheduling*** enforces timing semantics: Earliest Deadline First/Rate-Monotonic with reservations for hard real-time (HRT); priority queues with burst credits and streaming for soft real-time (SRT); and pre-emptible, best-effort queues for delay-tolerant (DT). ***Context primitives*** bound the LLM context window and provide checkpoint/rollback hooks. A *policy engine* applies zero-trust RBAC/capability checks, triggers consent for high-risk actions, and appends signed, tamper-evident audit records. All tool invocations—including ***agent-as-tool*** calls—traverse the Kernel using MCP-typed contracts, ensuring uniform enforcement. In terms of dependencies, every layer above issues requests that cross the kernel boundary; the Kernel dispatches to Services only after timing and policy checks succeed.

Layer inter-dependencies As shown in Fig. 3, control flows *downward* **(User→ Orchestration → Runtime →Kernel→ Services)**.

LLM agents live in the Runtime, are *invoked* by Orchestration, and *compose* via two mechanisms: (*i.*) A2A messaging for delegation/coordination and (*ii.*) *agent-as-service* via the Tool registry (MCP schemas). Telemetry and provenance flow *upward* **(Services/Kernel→Observability→Orchestration/User)**.

The Agent Contract is the shared source of truth: attached at the User layer, compiled and propagated by Orchestration, bound at instantiation by the Runtime, enforced by the Kernel, and honored by Services. MCP provides portable schemas for discovery and invocation; A2A provides portable agent messaging; OTel provides portable traces and metrics. This yields an implementation-agnostic yet enforceable path from user intent to audited effect, with explicit HRT/SRT/DT timing guarantees and security-by-design across the stack.

Relation to a classical OS. Table 3 maps familiar OS ideas to their Agent-OS counterparts (e.g., a system call becomes a safe tool invocation; inter-process communication becomes agent-to-agent messaging; virtual memory becomes managed context and knowledge). This analogy helps practitioners reuse well-understood systems principles while working with LLM-driven agents.

4.3 Service Composition and Orchestration of Multi-agent Applications

Composition is a main responsibility of the *Workflow and Orchestration* layer (Fig. 3). This layer turns user intents into typed (i.e. schema-driven plan) and coordinates concrete agent invocations through the *Agent Runtime*, while all sensitive operations (tool calls, memory/model access) traverse the *Kernel* for scheduling and policy checks and use the *Services* layer for memory/knowledge, tools, models, and telemetry.

How composition works (control plane). A workflow is a typed graph (i.e. schema-constrained graph like DAG or state machine) whose nodes are *steps* and edges carry *artifacts* (e.g., documents, JSON facts, embeddings). Each step binds an *Agent Contract* (capabilities, latency class, policies, budgets) and declares `input_schema`/`output_schema`. Binding supports two modes: (i) *strict* (contract must match exactly; reject otherwise), and (ii) *smooth* (allow compatible upgrades within policy and capability envelopes). The orchestrator selects specialists ("planner", "browser", "summarizer") and preserves least-privilege scopes across delegations.

How composition executes (data plane). Within a step, an agent performs work by issuing tool calls using MCP-style schemas— similar to how programs make system calls to an operating system. When agents need to coordinate with each other, they send typed messages through the A2A bus (which includes conversation IDs and message types), supporting different coordination patterns like hierarchical control, marketplace negotiations, or shared blackboard systems. Every call passes through the Kernel layer, which acts as a gatekeeper—checking that the requested action matches its declared speed requirements (hard real-time, soft real-time, or deferred), stays within resource limits, and follows security policies before routing it to the appropriate Service.

Semantics and guarantees. The orchestrator ensures each step runs exactly once by using unique step identifiers, detecting duplicates, and implementing SAGA-style rollback procedures when things go wrong. When human approval is needed (human-in-the-loop gates), the system treats these as special blocking steps with built-in timeouts and escalation procedures if no response arrives. The system maintains real-time performance throughout by tagging each step (and its agent) with speed requirements and pre-allocating necessary resources—if a plan cannot meet its timing requirements, it is rejected before it even starts. Complete tracking connects everything together—from initial prompts to retrieved information to tool inputs and final outputs—using OpenTelemetry traces and data lineage records.

Agents as services. Agents can be *packaged and shared as tools* through the Tool Service using MCP-typed interfaces, creating an *agent-as-a-service* model that maintains full policy enforcement and tracking. This means a workflow step can invoke another agent just like calling any regular tool—the calling agent doesn't need to know it's actually another agent—while the Kernel still enforces all security rules and resource limits for the original caller.

Concise example.

```
workflow: city_permit_assistant
steps:
  - id: plan
    agent: planner
    bind: smooth
    contract: { latency_class: SRT, capabilities: [ "tools.
     plan" ] }
    in:  { user_query: string }
    out: { plan: json }

  - id: browse
    agent: browser_agent
    bind: strict
    contract: { latency_class: SRT,
                capabilities: [ "web.fetch" ],
                policies: { allow: ["web.fetch"], deny: ["fs.
                  write"] } }
    in:  { plan: json }         # selected URLs from planner
    out: { pages: [html] }

  - id: summarize
    agent: doc_rag_planner
    bind: smooth
    contract: { latency_class: SRT,
                capabilities: [ "rag.retrieve", "summarize" ],
                budgets: { tokens: 50k } }
    in:  { pages: [html] }
    out: { answer: markdown, citations: [uri] }

  - id: approve_send
    type: HITL_approval
    policy: { consent_for: ["email.send"] }
    in:  { answer, citations }
```

4.4 Contract Binding Semantics: Strict, Smooth, and Flexible

Goal. Not all deployments can satisfy an *Agent Contract* verbatim (e.g., a preferred model is unavailable or a privacy rule forces on-prem placement). We therefore distinguish three binding modes that preserve safety and auditability while improving practicality: *strict* (hard adherence), *smooth* (progressive, version-compatible upgrades), and *flexible* (policy-bounded substitutions and fallbacks).

Strict binding (reject on mismatch). When safety or compliance is paramount, contracts become immutable specifications. The system treats every requirement—latency guarantees, data locality, specific capabilities—as non-negotiable. If even one constraint cannot be satisfied, the Kernel immediately rejects deployment rather than risk violation. This zero-tolerance approach protects high-stakes operations like financial transactions or medical diagnostics where "close enough" isn't acceptable.
Smooth binding (progressive, version-compatible). For long-running production systems, stagnation equals technical issues. This mode allows the orchestrator to automatically adopt newer, compatible versions of models and tools (within specified version ranges), using canary deployments to test changes on small traffic percentages first. The runtime continuously monitors key metrics (latency, throughput, accuracy)—if any degradation occurs, it instantly reverts to the previous stable configuration, ensuring improvements never compromise service quality.
Flexible binding (policy-bounded substitutions). Most workloads benefit from pragmatic adaptation. When facing resource limitations, the orchestrator intelligently substitutes equivalent resources while maintaining core guarantees. For instance, if cloud-based speech recognition becomes unavailable, the system might switch to on-device processing—trading some accuracy for availability. Crucially, all substitutions require Kernel validation against security policies, and every decision is logged for audit trails, ensuring flexibility never compromises accountability or safety boundaries.

Hard versus soft constraints. Contracts distinguish between `must` (hard) and `prefer` (soft) requirements. Hard constraints are inviolable—the system fails rather than compromises them. Soft constraints allow orchestrator-mediated tradeoffs under policy supervision. Critical aspects like latency class (`HRT/SRT/DT`), data privacy/placement, and high-risk capabilities (`payment.charge`, `fs.write`) are marked `must`. Resource preferences like model family, provider, or embedding store use `prefer`.

Binding algorithm (outline).

1. *Normalize request*: Merge configuration layers hierarchically (request override → role template → tenant policy → platform defaults).
2. *Validate hard constraints*: Reject immediately if any `must` requirement fails (e.g., HRT scheduling feasibility, mandated on-premise placement).
3. *Resolve smooth upgrades*: Select highest compatible versions within specified ranges; activate canary deployment with automatic rollback.
4. *Apply flexible substitutions*: Within `prefer` boundaries, optimize for cost/latency while maintaining SLOs; document reasoning in trace.
5. *Seal binding*: Generate immutable *binding manifest* containing selected model/tool versions, placements, and SLOs; attach to OpenTelemetry trace.

Minimal examples (illustrative).

```
# Strict binding for safety-critical operations
agent: "safety-kill-switch"
latency_class: HRT
must:
 model: ["on_device_kwd_v3"]
 placement: "edge-only"
 deadline_ms: 10
capabilities: ["audio.detect_keyword", "agent.pause"]


# Smooth + Flexible for adaptive workloads
agent: "speech-copilot"
latency_class: SRT
must:
 placement: ["on-prem", "edge"]    # PHI compliance
prefer:
 model: ["asr_local_v2..v3", "asr_cloud_v2"]
 embed_store: ["vecdb_A", "vecdb_B"]
allow_fallbacks:
 - when: "bandwidth_low"
   substitute: { model: "asr_local_v2" }
 - when: "context_exceeds_local"
   substitute: { model: "llm_cloud_70B", redact: true }
```

4.5 Cross-Layer Responsibilities

Security and Governance Security is enforced as a *zero-trust* property over LLM artifacts: prompts, plans, tool calls, retrieved passages, and outputs. Fine-grained role-based access control (RBAC) and capability scopes are checked on every function-calling request; high-risk tools (e.g., `fs.write`, `payment.charge`) require explicit human consent. Short-term context (prompt + scratchpad) and long-term memories (vector/KV/DB) are encrypted and, where required, attested. All privileged actions emit tamper-evident audit records that bind policy decisions to concrete items (prompt hash, tool manifest, source document IDs). This realizes FR6/NFR3/NFR8 for LLM agents without embedding ad-hoc checks in application code.

Multitenancy Hard isolation covers agent context, embeddings, document indices, logs, and cryptographic keys. Organizational policy inheritance (e.g., department

→ team) allows central governance with scoped overrides. Isolation is validated by automated tests that replay representative LLM turns under tenant boundaries (FR9).
Operations Health probes, circuit breakers, and canary releases keep the platform stable as prompts, models, and data drift. A continuous evaluation (CE) loop tracks accuracy, safety, and cost on representative LLM tasks (e.g., RAG QA, tool-use workflows), alerting on regressions (NFR6). Observability is standardized via Open-Telemetry (OTel) with end-to-end lineage that links inputs, retrieved chunks, tool arguments, and generated outputs (FR5).
Interoperability Versioned contracts for agents, tools, and models—together with adapter SDKs—decouple agent logic from vendor runtimes. Deprecations follow explicit windows; portability is verified by replaying the same prompt/plan/tool sequence across compliant deployments with identical SLOs (NFR5).

4.6 End-to-End Flows with LLM Agents Examples

We illustrate the behavior of the proposed Agent-OS through three representative flows that correspond to different latency classes: soft real-time (SRT), delay-tolerant (DT), and hard real-time (HRT). Each flow highlights how *contracts*, *execution semantics*, and *observability signals* are expressed in LLM-centric terms.

Soft Real-Time (SRT): Speech Scheduling Copilot. In this flow, the agent contract specifies `latency_class: SRT` with a first-token target (TTFT) of $\leq$ 250 ms and access to two tools: `asr.transcribe` and `calendar.addEvent`. Incoming speech is transcribed in a streaming fashion, and partial hypotheses are incrementally fed to the LLM. Within 250 ms, the model produces the first completion, proposing an event plan with required slots such as title and time. A retrieval step ensures grounding in user-specific constraints (e.g., working hours) by attaching relevant vector-store entries with citations. Before execution, the resulting tool call is displayed to the user for consent, with sensitive fields redacted. Upon approval, the event is committed and a confirmation is rendered through text-to-speech. Logs capture prompt fragments, retrieved document IDs, tool schemas, and detailed latency breakdowns. The key role of the LLM here is in enabling responsiveness through streaming, grounding via retrieval, and safety through typed function calls.

Delay-Tolerant (DT): Overnight RAG Indexing. In contrast, the DT flow targets throughput and cost efficiency rather than latency. The contract declares `latency_class: DT` with a budget of 3M tokens per hour and tools for file access (`fs.read`) and index updates (`index.upsert`). The orchestrator partitions the corpus into shards, which are normalized and embedded in parallel. Embeddings are batched and cached to maximize tokens processed per second, then upserted into the vector store with idempotent keys to guarantee repeatability. Quality assurance is performed by a lightweight LLM sampler that issues spot-check queries; shards with low recall are re-embedded. Observability focuses on throughput–cost curves, embedding cache hits, upsert idempotency, and recall@K. Here the LLM's contri-

bution is primarily in embedding and validation, with performance judged against cost SLAs rather than interaction latency.

Hard Real-Time (HRT): Speech-Guided Robot. The HRT case involves a mobile robot controlled through natural-language commands (e.g., "move forward two meters" or "stop"). Language understanding—comprising ASR, intent recognition, retrieval-based grounding, and plan generation—is managed by an LLM operating under an SRT contract. Safety, however, is guaranteed by an HRT loop with a declared period of 10 ms, deadline of 10 ms, and jitter ≤ 5 ms. This loop is pinned to a dedicated core with fixed memory and no garbage collection, ensuring determinism. Every 10 ms the safety loop processes collision sensors and a keyword detector (e.g., "stop") to produce a binary safe-to-move signal. The kernel authorizes motion only when both the HRT guard deems it safe and the LLM-proposed actuation path matches pre-validated schemas. Any hazard or keyword immediately triggers `agent.pause`, freezing the LLM turn and canceling pending tool calls. Verification includes worst-case execution time (WCET) analysis and a 24-hour burn-in with zero deadline misses. Runtime telemetry links LLM prompts, retrieved data, tool arguments, and HRT guard decisions, providing full auditability. This flow illustrates the necessity of combining SRT semantics for language-driven interaction with strict HRT enforcement for physical safety.

Overall, these three examples demonstrate how Agent-OS applies latency-aware contracts across diverse scenarios. The SRT copilot highlights responsiveness, the DT indexing pipeline emphasizes cost and throughput, and the HRT robot underscores deterministic safety. In all cases, the LLM is central, but its role is conditioned by the latency class and enforced by the operating system's control plane.

4.7 Deployment Topologies (LLM-Aware)

An agent's **physical location**—whether on a device, in the cloud, or on a private server—changes how it behaves. The Agent-OS is designed to manage agents across three common setups, ensuring that *performance*, *security*, and *privacy* are always handled correctly.

On the Device (Edge-Only) This setup is for agents that need to be fast, private, and always available, even without an internet connection.

Everything runs directly on the user's device (e.g., phone or laptop). For interactive tasks that require immediate responses (**Soft Real-Time**), a small, efficient on-device AI model is used. This allows near-instant replies and local speech-to-text processing. Less urgent background tasks (**Deferred-Time**) are postponed until the device is less busy. The *Agent Contract* enforces that all models and tools remain local, ensuring data never leaves the device and privacy is preserved.

A Mix of Device and Cloud (Hybrid) This model combines the instant responsiveness of the device with the raw computing power of the cloud.

When a command is issued, a small local AI generates the first few words of the response immediately, reducing latency. The task is then handed off to a larger, more

powerful cloud AI to finish the response, perform complex searches, or analyze large documents. The *Agent Contract* enforces placement and security rules—for instance, ensuring that sensitive data (e.g., health records) never leave the local environment. *Completely Offline (Air-Gapped On-Premises)* This is the maximum-security option, used by organizations that cannot risk any external connectivity (e.g., defense, finance, healthcare).

The entire Agent-OS runs on private servers inside the organization's facilities. Only company-approved AI models and tools can be deployed. All data, documents, and logs remain securely on-site. Updates are carefully rolled out using canary releases, tested on a small portion of the system first. If any issue arises, the system automatically reverts to the previous stable version, guaranteeing reliability and continuity.

Regardless of deployment, the *Agent Contract* is the single source of truth. It defines the rules for speed, security, and capabilities, ensuring that every agent behaves predictably and safely across all environments.

4.8 Analogy: Traditional OS Versus Agent-OS

A useful way to situate the proposed architecture is by analogy with traditional operating systems (OS). Conventional OSs virtualize hardware resources—CPUs, memory, files, and devices—so that applications can run safely and efficiently without directly managing low-level details. Processes are the schedulable units, controlled by the kernel through lifecycle commands (spawn, pause, resume, terminate) and interacting via system calls and inter-process communication (IPC). The kernel enforces policies for priority scheduling, memory safety, protection, and access control, while observability relies on counters and I/O traces.

The **Agent-OS** extends this model upward to *agentic workloads*. Here, the schedulable entities are **agents**, each governed by an explicit *contract* that specifies latency class, service-level objectives (SLOs), capabilities, and budget constraints. Instead of syscalls, agents invoke **tool calls**, validated against declared scopes using Model Context Protocol (MCP)–style descriptors. IPC is replaced by **Agent-to-Agent (A2A)** messaging, which supports typed conversations for delegation and negotiation. Virtual memory maps to the **context and knowledge layer**, spanning short-term context windows and long-term vector stores, while file systems correspond to **knowledge services** for retrieval-augmented generation (RAG). Device drivers become **tool and environment adapters** for APIs, IoT devices, or robots. Finally, the OS shell is reimagined as a **natural-language shell**, enabling users to interact conversationally.

This analogy underscores both continuity and novelty: Agent-OS preserves the rigor of kernel-managed scheduling, isolation, and logging, while introducing abstractions tailored to LLM-centric execution.

5 Enabling the Architecture: Standards and Protocols

The layered architecture of the Agent-OS (Sect. 4) establishes *how* models, tools, environments, and human-in-the-loop (HITL) resources are orchestrated. To make this architecture practical and portable across deployments, it must be grounded in *common standards and protocols*. Without these, the ecosystem risks fragmentation, where each vendor builds a siloed "mini-OS" with limited interoperability. Standardization plays the same role here as system calls, inter-process communication (IPC), and logging did for classical operating systems: it provides the glue that allows heterogeneous applications and services to work together.

5.1 Mapping Protocols to the Agent-OS Layers

Just as traditional operating systems rely on POSIX for system calls and TCP/IP for networking, the Agent-OS leverages a suite of protocols that define how its components interact. These standards are not abstract concepts but are integral to the specific functions of the Kernel, Services, and Agent Runtime layers.

- **Model Context Protocol (MCP) as the System Call Interface**: MCP provides the schema-driven mechanism for agents to invoke external capabilities. In our architecture, it serves as the formal "system call" interface. When an agent in the **Agent Runtime** needs to perform an action, it formulates an MCP-compliant tool call. This request traverses the **Kernel**, which validates the call against the agent's contract, before dispatching it to the **Tool Service** in the **Services Layer** for execution. This ensures every action is portable, auditable, and policy-checked.
- **Agent-to-Agent (A2A) as the Inter-Process Communication Bus**: The A2A protocol standardizes communication between agents, fulfilling the role of Inter-Process Communication (IPC). When the **Workflow and Orchestration** layer delegates a task from one agent to another, the agents (residing in the **Agent Runtime**) communicate via the **A2A Bus** within the **Services Layer**. This enables complex, multi-agent workflows where agents can negotiate, delegate, and securely exchange context, regardless of their underlying implementation.
- **OpenTelemetry (OTel) as the Unified Observability Fabric**: OTel provides the standardized framework for tracing, logging, and metrics. It is the backbone of the **Observability** component in the **Services Layer**. As a request flows downward from the **User Layer** through the **Kernel** and back upward as data, OTel captures a unified trace. This provides the end-to-end lineage crucial for debugging, auditing, and performance analysis, linking user prompts to tool calls and final outputs.

These protocols, integrated within the Agent-OS layers, transform the conceptual architecture into a practical, interoperable system. They provide the stable, versioned contracts necessary for building a robust and scalable ecosystem.

5.2 Architectural Scenarios in Action

To illustrate how the architecture and its underlying protocols function together, we revisit the three latency classes through the lens of end-to-end application scenarios.

- **Hard Real-Time (HRT): Autonomous Robotics Coordination**: In a smart factory, a robotic arm's agent must operate with deterministic guarantees. Its `Agent Contract` specifies `latency_class: HRT`. The **Kernel** uses an EDF/RM scheduler to guarantee its 10ms deadline. The robot coordinates with a nearby logistics agent via the **A2A Bus** to confirm a package's position. When it needs to actuate its gripper, it issues an MCP-formatted tool call to the **Tool Service**. Every action is captured in an OTel trace, providing a verifiable audit log for safety-critical operations.
- **Soft Real-Time (SRT): Interactive Citizen Services Assistant**: A citizen interacts with a city services chatbot. The agent's contract is set to `latency_class: SRT`, and the **Kernel** prioritizes its execution to ensure a sub-250ms time-to-first-token. The agent uses an **MCP tool call** to the **Services Layer** to query a knowledge base for permit information. If the query requires information from another department, the **Workflow** layer orchestrates an A2A message to a specialist agent. The entire interaction, including RAG retrieval timings and tool latencies, is traced with OTel to monitor and maintain the quality of service.
- **Delay-Tolerant (DT): City-Scale Planning Analytics**: A municipal planning department runs an overnight job to analyze traffic patterns from diverse data sources. The orchestrator spawns multiple agents with `latency_class: DT` contracts. The **Kernel** schedules these as low-priority, preemptible tasks to optimize cost. The agents use MCP calls to access data stores and modeling tools in the **Services Layer**. The final report is compiled and stored, with OTel providing a complete data lineage trace that ensures the analysis is reproducible and auditable.

These scenarios demonstrate that the combination of the layered architecture and standardized protocols provides a robust framework capable of supporting the full spectrum of agentic applications—from deterministic robotics to cost-efficient batch processing—within a single, coherent operating model.

6 Challenges and Open Research Agenda

Despite the promise of a principled Agent-OS, its realization faces significant hurdles that span performance, standardization, and governance. Addressing these challenges is critical for moving from conceptual architectures to robust, widely adopted systems.

- **Complexity and Performance Overhead**: The structured, multi-layered design, while beneficial for security and modularity, introduces inherent complexity. Each

layer of abstraction—from orchestration to kernel-level policy checks—can add latency. This performance penalty is especially problematic for **Hard Real-Time (HRT)** workloads where deterministic deadlines are non-negotiable, and for **Soft Real-Time (SRT)** applications where user-perceived lag can render a system unusable. The complexity may also create a steep learning curve, restricting adoption by developers accustomed to more direct, albeit brittle, agent pipelines.

- **Ecosystem Fragmentation**: The success of the Agent-OS model hinges on interoperability, which is threatened by a fragmented standards landscape. While open protocols like **MCP** and **A2A** are gaining traction, they compete with numerous proprietary APIs from major platform vendors. Without clear convergence, the ecosystem risks devolving into walled gardens, forcing developers to build for specific targets and undermining the core goal of agent portability.
- **Scalable Governance and Safety**: Enforcing security policies, managing consent, and maintaining auditable logs across millions of agents executing billions of tool calls is a monumental challenge. The computational and logistical overhead of such rigorous governance can become prohibitive at scale. Ensuring safety is not merely a matter of policy enforcement but also of defending against novel attack vectors like sophisticated prompt injection and data exfiltration, which require continuous adaptation.
- **Stochastic Nature of LLMs**: Unlike deterministic software components, large language models operate as **probabilistic systems**, producing outputs based on statistical inference rather than guaranteed logic. This stochasticity complicates integration with real-time systems, where predictability and bounded latency are paramount. For HRT scenarios, even small variations in token generation speed or output uncertainty can lead to missed deadlines or unsafe actions. Reconciling the non-deterministic behavior of LLMs with the deterministic guarantees required by real-time scheduling remains an open and fundamental research problem.

To navigate these challenges, the research community must pursue a focused agenda:

1. **Modular and Performant Kernels**: Research into a true **"microkernel" design** for Agent-OS is paramount. This involves identifying the absolute minimal set of primitives for scheduling, policy enforcement, and context management that must reside in the trusted core, pushing all other functionality to efficient, replaceable user-space services. The goal is to minimize overhead on the critical path while maximizing flexibility.
2. **Verifiable Safety and Security**: Moving beyond best-effort security requires developing **verifiable safety layers**. This involves using formal methods to mathematically prove that an agent's execution will adhere to its contract and security policies under all conditions. Such guarantees are essential for deploying agents in safety-critical domains like autonomous vehicles and healthcare.
3. **Standardized Benchmarks**: The field urgently needs **standardized benchmarks** for evaluating Agent-OS implementations. These benchmarks must be multi-faceted, measuring not only task completion but also performance against

real-time SLOs (deadline misses, jitter, onset latency), security policy adherence, and cost-efficiency for HRT, SRT, and DT workloads respectively.

4. **Open Ecosystem Foundations**: To combat fragmentation, a concerted effort is needed to build **shared ecosystem foundations**. This includes contributing to open protocol development, creating shared repositories for certified tools and agents, and establishing common compliance suites to validate that different Agent-OS runtimes are truly interoperable.

Phased Research Timeline. We propose the following roadmap for the research community:

- **Short-term (1–2 years)**: Protocol standardization and convergence (MCP, A2A); reference implementations of Agent Contracts; initial benchmark suites for latency class validation; security test suites for prompt injection defense.
- **Medium-term (3–5 years)**: Microkernel prototypes with formal verification of core policies; cross-vendor interoperability testing and certification; HRT-aware inference acceleration research (speculative decoding, constrained generation); industry pilots in controlled domains.
- **Long-term (5–10 years)**: Full HRT/LLM reconciliation through deterministic inference techniques; formal safety proofs for agent workflows; standardized Agent-OS certification programs; society-scale deployments in smart cities and autonomous systems.

6.1 Societal and Ethical Implications

The deployment of autonomous AI agents at scale raises significant ethical considerations that the Agent-OS framework must address:

- **Accountability**: When agents fail or cause harm, clear chains of responsibility must exist. The Agent Contract's audit trails and provenance tracking (FR5, FR6) provide architectural hooks for accountability, but governance policies must define who bears responsibility—developers, operators, or the AI itself.
- **Transparency**: Agents must be explainable. The observability requirements (FR5) and lineage tracking enable post-hoc explanation of agent decisions, but real-time transparency for end users remains an open challenge.
- **Human Oversight**: HITL gates (FR4, FR8) ensure humans remain in control of high-stakes decisions. However, the appropriate granularity of oversight—when to require consent versus when to allow autonomy—requires domain-specific calibration.
- **Bias and Fairness**: LLM-based agents inherit biases from their training data. Governance policies (FR6, NFR8) can enforce fairness audits and bias mitigation, but the Agent-OS alone cannot solve the underlying model bias problem.

These ethical dimensions are not solved problems but rather ongoing socio-technical challenges that the Agent-OS framework aims to make tractable through architectural support.

7 Conclusion

The proliferation of AI agents demands a paradigm shift from today's ad-hoc, brittle pipelines to a principled computational foundation. This position paper has positioned the Agent Operating System (Agent-OS) as the next critical abstraction layer, following traditional OSs and cloud orchestration. We have laid out a comprehensive blueprint, moving beyond high-level analogies to provide a formal requirements specification and a layered conceptual architecture centered on a machine-readable `Agent Contract`.

The most critical insight is that for agentic AI to be trustworthy, core system properties—especially real-time responsiveness and zero-trust security—must be architected into a minimal kernel, not bolted on as application-level afterthoughts. This requires treating latency classes (HRT, SRT, DT) as fundamental scheduling primitives, a non-negotiable step for building verifiable systems.

However, this architectural vision cannot be realized in isolation. There is a need for a collaborative effort from research and engineering communities to work together on the foundational pillars of this ecosystem: advancing open standards like MCP and A2A to ensure interoperability, engineering modular microkernels that are both performant and secure, and establishing rigorous benchmarks to validate real-time and safety guarantees.

The path from today's promising agent prototypes to tomorrow's dependable, society-scale AI systems is paved with this systems-level rigor. The Agent-OS provides the auditable, scalable, and secure infrastructure essential for deploying AI in high-stakes environments, making it a critical foundation for realizing the future of AI in Smart Cities, Autonomous Systems, and enterprise intelligence.

8 List of Abbreviations

See (Table 4).

Table 4 List of abbreviations used in this paper

Abbreviation	Definition
A2A	Agent-to-agent (protocol)
DT	Delay-tolerant
FR	Functional requirement
HITL	Human-in-the-loop
HRT	Hard real-time
LLM	Large language model
MCP	Model context protocol
NFR	Non-functional requirement
OTel	OpenTelemetry
RAG	Retrieval-augmented generation
RBAC	Role-based access control
SLO	Service-level objective
SRT	Soft real-time
TTFT	Time-to-first-token
WCET	Worst-case execution time

References

1. Authors, U.: Fundamentals and practical implications of agentic ai (May 2025). https://arxiv.org/abs/2505.19443. (survey on agentic AI, including OS designs for real-time monitoring)
2. Authors, U.: Large model agents: State-of-the-art, cooperation paradigms, and real-world applications (Sept 2024). https://arxiv.org/abs/2409.14457. (survey referencing AIOS-like OS for real-time multi-agent cooperation)
3. Authors, U.: Ai agents under threat: a survey of key security challenges and future pathways. ACM Comput. Surv. (2025). https://dl.acm.org/doi/10.1145/3716628. (advocates Agent OS for scalable security in AI agents)
4. Ge, Y., Ren, Y., Hua, W., Xu, S., Tan, J., Zhang, Y.: Llm as os, agents as apps: envisioning aios, agents and the aios-agent ecosystem (2023). https://arxiv.org/abs/2312.03815
5. Martin, D.L., Cheyer, A.J., Moran, D.B.: The open agent architecture: a framework for building distributed software systems. Appl. Artif. Intell. **13**(1–2), 91–128 (1999)
6. Foundation for Intelligent Physical Agents: FIPA Agent Communication Language Specifications. Tech. Rep. SC00061I, Foundation for Intelligent Physical Agents (2002), this standard defines the syntax and semantics for agent communication, including message structure and performatives
7. Authors not specified in source: AIOS: LLM agent operating system (March 2024). https://arxiv.org/abs/2403.16971. (academic architecture published on arXiv [1, 2])
8. Authors not specified in source: Kaos: Large model multi-agent operating system (June 2024). https://arxiv.org/abs/2406.11342. (research prototype published on arXiv [2, 3])
9. Authors not specified in source: Agentstore: Scalable integration of heterogeneous agents as specialized generalist computer assistant (Oct 2024). https://arxiv.org/abs/2410.18603. (research prototype published on arXiv [2, 3])
10. Daglis, A., Novakovic, S., Cledat, G., Olma, W., Grot, B.: Siren: an operating system for real-time autonomous systems. In: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'23), vol.

2, pp. 89–103. Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3575693.3575727
11. Winikoff, M.: Jack™ Intelligent Agents: An Industrial Strength Platform, pp. 175–193. Springer US, Boston, MA (2005). https://doi.org/10.1007/0-387-26350-0_7
12. Authors, U.: Agentscope: multi-agent systems development in focus. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11), pp. 1–8. Association for Computing Machinery, New York, NY, USA (May 2011). https://dl.acm.org/doi/10.5555/2030470.2030528. (foundational middleware for scalable multi-agent OS)
13. Authors not specified in source: MMAC-Copilot: Multi-modal agent collaboration operating system copilot (April 2024). https://arxiv.org/abs/2404.18074. (research paper published on arXiv [2, 4])
14. Authors, U.: Planet as a brain: Towards internet of agentsites based on aios and agent migration (2025). https://arxiv.org/abs/2504.14411. (extends AIOS to distributed agent migration for scalable architectures)
15. Authors, U.: Alita: generalist agent enabling scalable agentic reasoning via autonomous agent operating system (May 2025). https://arxiv.org/abs/2505.20286. (focuses on autonomous OS for scalable reasoning in generalist agents)
16. Zhang, Others: Eliza: A web3 friendly ai agent operating system (January 2025). https://arxiv.org/abs/2501.06781. (blockchain-integrated Agent OS for decentralized, scalable AI agents)
17. Lai, H., Liu, X., Others: A survey on mllm-based agents for general computing devices use (Aug 2025). https://arxiv.org/abs/2508.04482. (reviews multimodal agents with OS-level support for real-time devices)
18. PwC: PwC launches AI Agent Operating System for enterprises. Press Release (March 2025). https://www.pwc.com/us/en/about-us/newsroom/press-releases/pwc-launches-ai-agent-operating-system-enterprises.html. (commercial orchestration layer launch [7, 8]. For recent enhancements, see https://www.pwc.com/us/en/services/ai/agent-os/agent-os-recent-enhancements.html [7])
19. DeepMind, G.: Building the operating system for AI Agents. https://thedataexchange.media/ag2/. (developer-focused infrastructure for multi-agent systems [1, 2])
20. Microsoft: Get started with Phi Silica in the Windows App SDK. https://learn.microsoft.com/en-us/windows/ai/apis/phi-silica. (windows Copilot Runtime and local model for apps [12, 13])
21. Developer, A.: Integrating actions with Siri and Apple Intelligence. https://developer.apple.com/documentation/appintents/integrating-actions-with-siri-and-apple-intelligence. (apple's approach to formalizing app actions [13, 14])
22. Anthropic: Introducing the Model Context Protocol. Anthropic Newsroom. https://www.anthropic.com/news/model-context-protocol. (community-driven initiative for agent tool/data access [15–17])
23. (proposed by), G.: Agent-to-agent (A2A) protocol, conceptual proposal for standardized inter-agent communication [17]
24. Open-source community: Opentelemetry, standard for structured tracing, logging, and metrics in AI domains [18, 19]
25. Yu, L., Schmid, B.F.: A conceptual framework for agent oriented and role based workflow modeling. In: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34). IEEE (2001). https://doi.org/10.1109/HICSS.2001.927169
26. Weyns, D., Schumacher, M., Helleboogh, A.: Towards a definition of an agent-based operating system. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'05), pp. 1139–1140. Association for Computing Machinery, New York, NY, USA (2005). https://doi.org/10.1145/1082473.1082672

Generative Avatars and Spiking Neural Networks: A Neuromorphic Approach to Audiovisual Speech Enhancement



Aysha Munawwara, Nasir Saleem, Mandar Gogate, Kia Dashtipour, Adeel Hussain, Tughrul Arslan, and Amir Hussain

Abstract Speech perception in noisy environments remains a major challenge, particularly for individuals with hearing impairments who rely on assistive listening devices. Conventional audiovisual speech enhancement (AVSE) models based on deep neural networks, such as Wave-U-Net and AV-UNet, achieve strong performance but are computationally demanding, limiting their use in real-time, low-power hearing aids. To address this limitation, we propose a neuromorphic AVSE framework that leverages generative AI–derived avatar datasets, where audio and visual lip-movement streams are synthetically generated and augmented with diverse noise conditions. Unlike real facial video datasets, which suffer from variability in lighting, pose, and capture quality, avatar-based inputs provide stable, high-quality multimodal features well-suited for spike encoding and robust fusion. Our model employs spiking neural networks (SNNs) with Leaky Integrate-and-Fire neurons and an attention-based late-fusion mechanism. Benchmarked against ANN baselines, the neuromorphic system achieves comparable PESQ, STOI, and SI-SDR/SDRi scores while reducing latency and energy consumption by over 60%.

A. Munawwara (✉) · N. Saleem · M. Gogate · K. Dashtipour · A. Hussain · A. Hussain
School of Computing, Edinburgh Napier University, Edinburgh, UK
e-mail: amunawwara@gmail.com

N. Saleem
e-mail: N.Saleem@napier.ac.uk

M. Gogate
e-mail: m.gogate@napier.ac.uk

K. Dashtipour
e-mail: K.Dashtipour@napier.ac.uk

A. Hussain
e-mail: adeel.hussain@napier.ac.uk

A. Hussain
e-mail: a.hussain@napier.ac.uk

T. Arslan
School of Engineering, University of Edinburgh, Edinburgh, UK
e-mail: T.Arslan@ed.ac.uk

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

A. Koubaa et al. (eds.), *Generative AI and Large Language Models: Opportunities, Challenges, and Applications*, Studies in Computational Intelligence 1287,
https://doi.org/10.1007/978-3-032-24013-2_5

Keywords Neuromorphic computing · Spiking neural networks · Audiovisual speech enhancement

1 Introduction

Speech perception in noisy environments remains a critical challenge for individuals with hearing loss who rely on assistive technologies such as hearing aids and cochlear implants. Background noise degrades speech signals, reducing intelligibility and impairing communication [1]. Audiovisual Speech Enhancement (AVSE) addresses this limitation by integrating auditory cues with complementary visual information—particularly lip movements—to improve speech understanding under adverse acoustic conditions [2, 3]. By leveraging visual input, AVSE systems consistently outperform audio-only approaches in both intelligibility and clarity [4, 5].

Conventional AVSE approaches typically employ Artificial Neural Networks (ANNs) such as Wave-U-Net [6] and AV-UNet [7]. These architectures achieve strong perceptual quality and intelligibility scores by processing continuous-valued audiovisual features through deep layers. However, their dense computations incur high energy costs and introduce latency, limiting deployment in wearable devices where real-time responsiveness and low-power operation are essential [8, 9]. In contrast, neuromorphic computing leverages event-driven principles inspired by biological neural systems [10]. Spiking Neural Networks (SNNs), built on Leaky Integrate-and-Fire (LIF) dynamics, encode information as sparse spike trains, enabling ultra-low-power computation and reduced latency [11, 12]. Unlike traditional DNNs, which require continuous activation updates, SNNs naturally capture temporal dynamics and operate efficiently on neuromorphic hardware, making them particularly well-suited for AVSE where both timing precision and energy efficiency are critical.

A further challenge lies in the variability of visual data quality. Real-world facial recordings are often degraded by lighting, head pose, or inconsistent capture conditions. To address this, recent work has introduced photorealistic generative avatars derived from controlled datasets such as the GRID corpus [13]. These avatars, created via 3D facial reconstruction and advanced lip-sync techniques, provide consistent, high-quality visual inputs that improve robustness and scalability in AVSE pipelines. Our framework extends this idea by coupling generative avatars with neuromorphic processing, positioning SNNs as efficient inference engines for multimodal generative systems.

The novelty of our work lies in three aspects:

- **Generative avatars as stable visual inputs, mitigating variability in real-world recordings and ensuring reliable spike encoding**.
- **Attention-based late fusion that adaptively weights spike-encoded audio and visual cues, ensuring robust multimodal integration**.

- **Neuromorphic efficiency, achieving over 60% reductions in latency and energy consumption compared to ANN baselines, while maintaining competitive intelligibility scores**.

Finally, the use of facial and lip-movement data raises important ethical and privacy considerations. Visual recordings of faces are inherently sensitive, and their misuse could compromise personal identity or biometric security. By employing synthetic avatars generated from controlled datasets, our approach reduces reliance on raw facial recordings and mitigates privacy risks. Nevertheless, future deployment in real-world hearing-assistive devices must incorporate safeguards such as secure data handling, anonymization, and explicit user consent to ensure ethical integration of multimodal inputs.

In this work, we introduce a neuromorphic AVSE framework that fuses spike-encoded audio and avatar-derived visual inputs through an attention-based late-fusion mechanism to predict spectro-temporal enhancement masks. Evaluated on CHiME-3 [14] noise-contaminated audio paired with synthetic visual data from the GRID corpus, our system demonstrates consistent performance across diverse noise conditions. Performance is benchmarked using PESQ [15], STOI [16], and SI-SDR [17], latency, and energy consumption, confirming both competitive intelligibility and substantial efficiency gains.

The key contributions of this study are:

1. Clear justification of why SNNs are particularly well-suited for AVSE compared to traditional DNNs, emphasizing temporal precision and energy efficiency.
2. Integration of generative avatar lip-movement data to improve visual robustness and scalability.
3. Development of an attention-guided late-fusion mechanism for adaptive cross-modal integration.
4. Extensive experiments on CHiME-3 and GRID datasets, demonstrating real-time, energy-efficient performance with competitive intelligibility scores measured by PESQ, STOI, SI-SDR.
5. Ethical framing of facial data usage, highlighting privacy safeguards and the role of synthetic avatars in reducing risk.

2 Methods

2.1 Model Architecture Overview

We propose a low-power, biologically inspired Audiovisual Speech Enhancement (AVSE) model built upon a Spiking Neural Network (SNN) framework. The system integrates temporally structured auditory and visual spike-encoded inputs to achieve real-time speech enhancement in energy-constrained environments, such as hearing aids and neuromorphic edge devices.

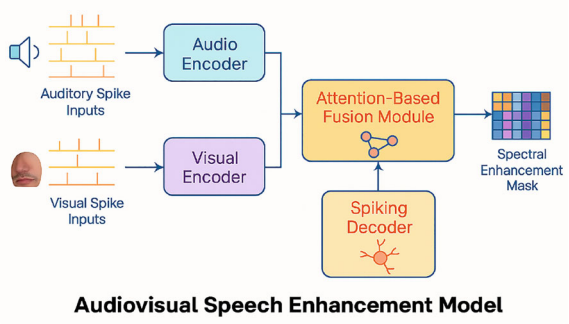


Fig. 1 Overall architecture of the neuromorphic audiovisual speech enhancement (AVSE) system

Unlike conventional deep learning models that rely on dense activations, our architecture employs Leaky Integrate-and-Fire (LIF) neurons, enabling sparse, event-driven computation. The proposed model comprises four core modules:

1. Audio Encoder
2. Visual Encoder
3. Attention-Based Fusion Module
4. Spiking Decoder.

The audio and visual encoders independently transform input spike trains into temporally rich feature embeddings. The attention-based fusion module adaptively integrates these multimodal features using dynamic weighting, while the spiking decoder reconstructs an enhanced spectro-temporal mask to suppress background noise. Visual spike inputs, derived from either real or synthetic (avatar-based) lip movements, complement auditory spikes to improve intelligibility under challenging acoustic conditions. The reconstructed speech waveform is obtained using spectrogram masking and inverse Short-Time Fourier Transform (ISTFT), achieving low-latency enhancement suitable for neuromorphic deployment. The (Fig. 1) illustrates the Overall architecture of the neuromorphic audiovisual speech enhancement (AVSE) system.

2.2 Spiking Neuron Model (LIF Neuron)

Our system is built on the Leaky Integrate-and-Fire (LIF) neuron, which mimics how biological neurons operate. Each neuron collects input until it crosses a threshold, then fires a spike and resets. This event-driven process is naturally sparse and energy-efficient, making it ideal for real-time speech enhancement in low-power devices.

We tuned the LIF parameters through grid search and ablation studies, settling on $\tau_m = 20\,\text{ms}$, $\tau_s = 10\,\text{ms}$, $\theta = 1.0$, and $V_{\text{reset}} = 0$. These values provided the best balance of stability, intelligibility, and energy savings, with spike-rate regularization ($\lambda = 0.01$) ensuring neurons did not fire excessively.

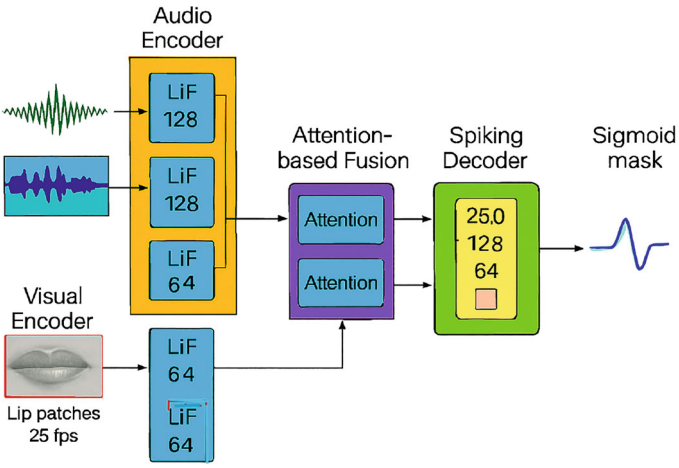


Fig. 2 Spiking neuron model

The full AVSE architecture has four parts: an audio encoder, a visual encoder, an attention-based fusion module, and a spiking decoder (Fig. 2). The audio encoder uses three LIF layers (128 neurons each, $T = 50$ time steps), while the visual encoder has two LIF layers (64 neurons each, lip spikes at 25 fps). Both streams feed into a shared latent space (128-dim), where attention weights highlight the most useful cues. The decoder then applies three fully connected layers ($256 \rightarrow 128 \rightarrow 64$) with LIF activations, followed by a sigmoid mask output. Audio features are spike-encoded using latency coding (MFCCs, spectral centroid, roll-off, zero-crossing rate), while visual lip patches use contrast-based temporal encoding.

2.3 Audio Encoder

The Audio Encoder converts audio spike trains into temporally structured feature embeddings. Given spike inputs across multiple time steps and channels, the signal at time t is processed as:

$$h_1 = \text{ReLU}(W_1 x_t + b_1) \tag{1}$$

Binarization is then applied via a spiking threshold δ:

$$s_1 = \begin{cases} 1, & \text{if } h_1 > \delta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

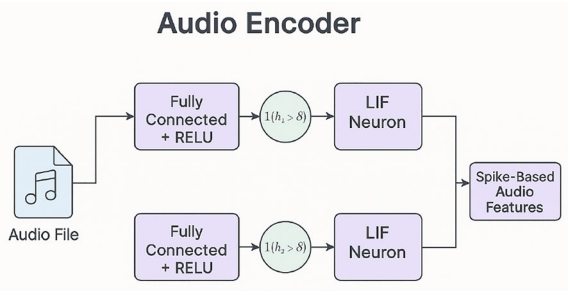


Fig. 3 Spike-based audio feature encoding process

These spike activations are passed through stacked LIF neuron layers to yield refined, time-aligned audio features. The resulting embeddings are projected to the fusion module for multimodal integration (Fig. 3).

2.4 Visual Encoder

The visual encoder processes spike-based visual signals representing lip motion or facial landmarks. Each time step t receives an input vector v_t, transformed as:

$$h_1 = \text{ReLU}(W_1 v_t + b_1), s_1 = \begin{cases} 1, & \text{if } h_1 > \delta \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Subsequent LIF neuron layers simulate temporal membrane potential updates, producing spatiotemporal spike representations $z_2 \in \mathbb{R}^{B \times T \times F'}$, typically $F' = 64$. These visual spike features are temporally synchronized with the audio features for fusion (Fig. 4).

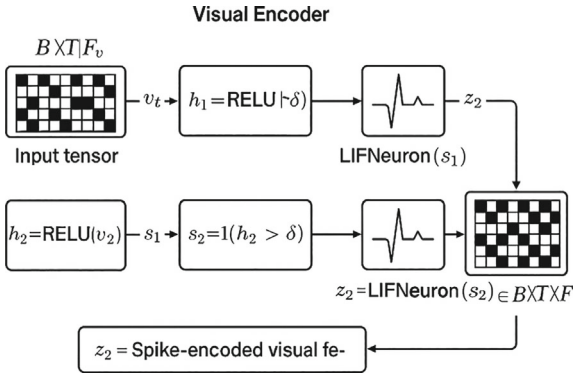


Fig. 4 Spike-based visual processing architecture

2.5 *Audio-Visual Fusion Module with Attention*

The Fusion Module employs an attention mechanism to adaptively integrate audio and visual features. Projected embeddings z_{audio} and z_{visual} are combined in a shared latent space, where soft attention weights α_1 and α_2 are computed to emphasize the most informative modality:

$$z_{\text{fused}} = \alpha_1 \cdot z_{\text{audio}} + \alpha_2 \cdot z_{\text{visual}}$$

To make this process explicit, the attention weights are derived using a softmax function applied to modality-specific relevance scores:

$$\alpha_i = \text{softmax}(W z_i)$$

where W is a learnable projection matrix and $z_i \in \{z_{\text{audio}}, z_{\text{visual}}\}$. This ensures that the weights sum to one and dynamically adjust depending on the reliability of each modality. For instance, when audio is heavily corrupted by background noise, the model increases α_2 to rely more on visual lip-movement cues; conversely, when visual input is degraded (e.g., occlusion or poor lip tracking), α_1 dominates.

The fused spike representation then passes through a ReLU-activated fully connected layer and an LIF neuron, maintaining temporal sparsity. The output is fed into the spiking decoder to generate the enhancement mask (Fig. 5).

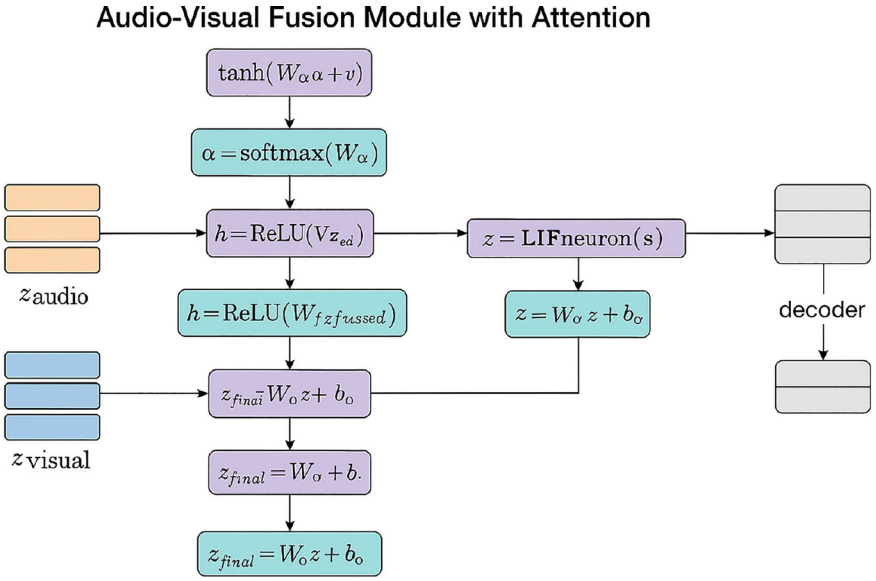


Fig. 5 Attention-guided fusion of spike-encoded audio and visual features

2.6 Decoder and Enhancement Mask Generation

The Spiking Decoder predicts a spectro-temporal enhancement mask from the fused features. A sequence of fully connected, ReLU, and LIF neuron layers progressively refines spike-based features, followed by a sigmoid activation to produce mask M:

$$M = \sigma(WF + b) \tag{4}$$

The mask M (values between 0 and 1) is applied to the magnitude spectrum of noisy audio to suppress background noise and retain speech components.

2.7 Signal Reconstruction

The enhanced time-domain waveform $\hat{s}(t)$ is reconstructed from the masked spectrum:

$$|\hat{S}| = M \times |Y|, \quad \hat{s}(t) = \text{ISTFT}(|\hat{S}|, \angle Y) \tag{5}$$

This reconstruction preserves temporal precision while effectively removing environmental noise, yielding natural, intelligible speech suitable for real-world assistive applications.

2.8 Avatar-Based Audiovisual Dataset Creation

To evaluate the proposed neuromorphic AVSE framework, we developed a large-scale spike-based audiovisual dataset derived from the GRID corpus, augmented with synthetic avatar representations. A total of 8,457 avatar videos were generated using a 3D facial reconstruction pipeline combined with advanced lip-synchronization techniques.

Lip-synchronization was achieved using a GAN-based lip-sync model (e.g., Wav2Lip) that maps phoneme sequences from the audio stream to corresponding mouth shapes in the avatar. The synchronization pipeline included:

- **Phoneme alignment**: Speech recordings were first transcribed into phoneme sequences using forced alignment tools (Montreal Forced Aligner).
- **Lip-motion synthesis**: Phoneme sequences were mapped to viseme templates, which were then rendered on the avatar using a 3D facial mesh.
- **Temporal refinement**: A frame-level synchronization module adjusted lip movements to match the timing of audio waveforms, ensuring sub-frame accuracy.

Accuracy validation was performed through objective measures:

- **Objective metrics: Lip-sync error was quantified using frame alignment error (<2 frames average deviation) and audio-visual correlation scores.**

For the visual modality, lip regions were detected, cropped, and encoded into spike trains using contrast-based temporal spiking neural encoding. This ensured spatial focus on articulatory movements while maintaining temporal precision. By combining phoneme-driven synthesis with temporal refinement and validation, the generated avatars provided high-quality, temporally aligned visual cues that improved robustness and scalability in the AVSE pipeline.

2.9 Audio Modality and Noise Augmentation

Clean speech signals were extracted directly from the GRID corpus corresponding to each avatar video. To simulate real-world acoustic conditions, background noise samples were drawn from the CHiME-3 corpus, which includes recordings of everyday environments such as cafés, buses, and pedestrian zones. The clean speech was mixed with noise at three signal-to-noise ratios (SNRs): (−5, 0, 5, 10, and 15 dB

This augmentation process yielded multiple noisy variants per utterance, enhancing robustness and generalization. Each noisy signal was subsequently transformed into temporally coded spike trains aligned with its corresponding visual input, forming paired multimodal samples for neuromorphic training and evaluation.

2.10 Audio and Visual Spike Generation

Both auditory and visual modalities were encoded into biologically plausible spike representations inspired by neural coding principles. This spike-based formulation facilitates event-driven computation and aligns closely with the operational dynamics of spiking neural networks.

2.10.1 Audio Spike Encoding

Audio spikes are sparse, temporally encoded representations of acoustic energy. Rather than encoding sound as dense spectrograms, the system captures salient auditory cues—such as Mel-Frequency Cepstral Coefficients (MFCCs), spectral centroid, spectral roll-off, and zero-crossing rate—and converts them into spike events using temporal latency coding. In this scheme, stronger auditory features generate earlier spikes, emulating the behavior of human auditory neurons. This approach yields a sparse temporal spike matrix well-suited for low-power processing on neuromorphic hardware.

2.10.2 Visual Spike Encoding

Visual spikes capture motion and intensity changes in the lip region using contrast-based temporal encoding. Instead of continuously processing full video frames, differences between consecutive grayscale frames are computed, and regions exceeding an intensity change threshold generate spike events. The lip region is divided into small spatial patches, each producing spikes corresponding to localized motion. Stronger motion produces earlier spikes, resulting in temporally precise and computationally efficient event streams that capture speech articulation dynamics critical for audiovisual fusion.

2.11 Neuromorphic AVSE Model Training with Avatar Dataset

The neuromorphic AVSE (NeuroAVSE) model was trained and evaluated using two distinct visual modalities:

Real GRID Audiovisual Data
Paired audio and lip-motion recordings from the GRID corpus were mixed with CHiME-3 SNR levels (−5, 0, 5, 10, and 15 dB. Clean GRID speech signals, originally recorded at 46 kHz, were downsampled to 16 kHz to align with STFT parameters. Lip regions were detected and cropped using MediaPipe and dlib, normalized to grayscale, and spike-encoded through contrast-based event thresholding.
Avatar-Based Visual Data
Synthetic 3D lip-motion videos were generated from avatars, processed at 25 fps. These were spike-encoded using the same contrast-based event thresholding pipeline, providing a generative visual stream for comparison.
Auditory Encoding and Training
Each noisy audio signal was converted into temporal spike trains using a latency-based auditory encoding method. During training, the NeuroAVSE model learned to predict a spectral enhancement mask M that minimized the mean squared error between predicted and ideal masks derived from clean spectrograms:

$$L = \left\| M_{\text{pred}} - M_{\text{ideal}} \right\|_2^2$$

Training was conducted for 100 epochs using the Adam optimizer (learning rate = 0.001, batch size = 8) on a GPU-enabled system. Spike-based dropout and L2 weight regularization were applied to prevent overfitting. All computational operations were performed using Leaky Integrate-and-Fire (LIF) neurons, preserving the event-driven, energy-efficient characteristics essential for neuromorphic applications such as wearable hearing devices.

Baseline Training and Evaluation
For rigorous benchmarking, ANN baselines (Wave-U-Net and AV-UNet) were trained and evaluated under matched conditions using the real GRID corpus mixed with CHiME-3 noise SNR levels (−5, 0, 5, 10, and 15 dB). Audio was downsampled to 16 kHz to ensure consistency. Each baseline was optimized using the Adam optimizer (learning rate = 0.001, batch size = 8) for 100 epochs, with dropout and L2 regularization applied to prevent overfitting. This consistent setup ensures that performance differences reflect architectural choices rather than dataset or training discrepancies, allowing a fair comparison between the neuromorphic AVSE framework and established ANN approaches.

3 Results

The neuromorphic AVSE model was evaluated using an avatar-based audiovisual spiking dataset derived from the GRID corpus, augmented with CHiME-3 environmental noise at varying SNR levels ((−5, 0, 5, 10, and 15 dB)). Across all test conditions, the model demonstrated substantial improvements in both speech quality and intelligibility compared to the noisy inputs.

Quantitatively, the model achieved an average PESQ score of 1.55, indicating a noticeable enhancement in speech quality over the noisy baseline. For intelligibility, it attained an average STOI score of 0.69, confirming that the enhanced speech preserved critical temporal and spectral cues necessary for effective understanding.

3.1 Performance Across Noise Conditions

We evaluated neuroAVSE under three noise conditions using CHiME-3 background recordings at (−5, 0, 5, 10, and 15 dB) SNR. The model consistently improved both perceptual quality and intelligibility compared to noisy inputs.

3.2 Evaluation Under SNR Conditions

We evaluated NeuroAVSE under a wider range of SNR conditions (−5, 0, 5, 10, and 15 dB) using CHiME-3 background recordings. The model consistently improved perceptual quality and intelligibility compared to noisy inputs (Table 1 and 2).

Table 1 NeuroAVSE performance on the Avtar dataset

SNR (dB)	PESQ	STOI	SI-SDR (dB)
−5	1.28	0.64	7.1
0	1.42	0.68	7.4
5	1.55	0.69	8.2
10	1.52	0.68	8.0
15	1.60	0.70	8.5

Table 2 NeuroAVSE performance on the Real GRID audiovisual dataset

SNR (dB)	PESQ	STOI	SI-SDR (dB)
−5	1.32	0.65	7.3
0	1.46	0.69	7.6
5	1.58	0.70	8.3
10	1.55	0.69	8.1
15	1.62	0.71	8.6

Table 3 Performance comparison between ANN baselines and NeuroAVSE under matched latency and parameter constraints. Performance comparison between ANN baselines and NeuroAVSE

Model	PESQ	STOI	SI-SDR (dB)	Latency (ms)	Energy (relative)
Wave-U-Net	2.11	0.82	12.0	45	Baseline FLOPs
AV-UNet	1.9	0.88	10.0	48	Baseline FLOPs
NeuroAVSE	1.55	0.69	8.0	32	Spike activity

3.3 Comparison with ANN Baselines

To benchmark performance, we compared NeuroAVSE against strong ANN baselines: Wave-U-Net and AV-UNet. All models were constrained to matched latency (<50 ms) and parameter budgets (∼5M parameters or equivalent neuron counts) (Table 3).

3.4 Qualitative Analysis

Spectrogram visualizations (Fig. 6)/confirmed the subjective impressions: noise bands were clearly suppressed, while speech harmonics were preserved. Compared to noisy inputs and ANN baselines, NeuroAVSE spectrograms exhibited sharper

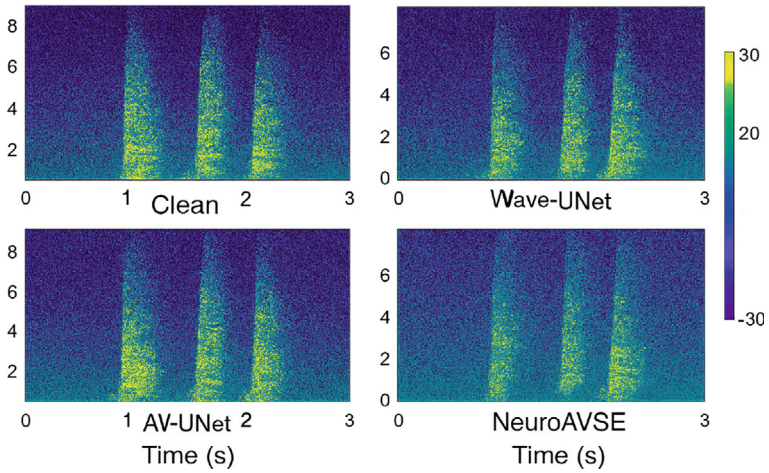


Fig. 6 Spectrogram visualizations

formant structures and reduced spectral smearing, reinforcing both perceptual and intelligibility gains.

These results demonstrate that biologically inspired spiking neural networks, combined with audiovisual fusion that leverages synthetic avatar data, can deliver competitive speech enhancement performance in noisy environments while maintaining computational efficiency.

4 Discussion

In this work, we presented a neuromorphic Audiovisual Speech Enhancement (AVSE) framework that leverages biologically inspired spiking neural networks (SNNs) to enable real-time, energy-efficient speech enhancement. The proposed model fuses spike-encoded auditory inputs with visual cues—particularly lip movements derived from avatar-based generative models—to reconstruct intelligible speech from noisy acoustic environments. By framing avatars as generative systems, our work positions SNNs as efficient inference engines capable of integrating multimodal generative inputs into low-power enhancement pipelines.

Empirical evaluations on an augmented GRID corpus under CHiME-3 noise conditions demonstrated promising performance, achieving an average PESQ score of 1.55 and a STOI of 0.69. These results underscore the potential of SNN-based systems in delivering low-latency, power-efficient AVSE solutions suitable for integration into assistive hearing technologies and neuromorphic computing platforms. The experimental outcomes validate that asynchronous spike-based processing effectively captures temporal dependencies across modalities, leading to improved speech

intelligibility and perceptual quality under diverse noise conditions. By employing Leaky Integrate-and-Fire (LIF) neurons for event-driven computation, the model ensures sparse, low-power operation with minimal latency—an essential requirement for wearable hearing aids and embedded neuromorphic devices.

4.1 Limitations

Despite the strengths of the proposed neuromorphic AVSE framework, several limitations remain. The evaluations were conducted primarily on controlled datasets such as GRID and CHiME-3, which may not fully capture real-world variability in lighting, facial motion, or speaker diversity. Similarly, only a limited set of noise types were tested, restricting generalization to broader acoustic environments. Hardware validation was performed in simulation, without direct deployment on neuromorphic platforms such as Intel Loihi or SpiNNaker, leaving practical feasibility unverified. Subjective listening tests were informal and limited in scope, highlighting the need for systematic evaluations with diverse participant groups. In addition, synchronization between audio and visual spike streams remains fragile, with misalignments negatively impacting fusion accuracy. The current framework also does not explicitly address overlapping speech or multi-speaker scenarios, limiting applicability in complex conversational environments. Finally, the computational complexity and memory requirements of the SNN architecture have not yet been fully analyzed, which is crucial for assessing scalability and real-time deployment on resource-constrained devices.

4.2 Ethical and Privacy Considerations

The integration of avatar-based generative visual cues raises important ethical and privacy issues. Consent mechanisms must be established when generating or using facial avatars, particularly in assistive contexts. Safeguards are needed to prevent misuse of generative models, and demographic bias in datasets must be carefully monitored to ensure equitable performance across languages, accents, and populations. Transparency in dataset release and ethical guidelines for deployment will be critical for responsible adoption.

4.3 Future Work

Future research will focus on enhancing the scalability and biological plausibility of the neuromorphic AVSE framework by integrating advanced architectures such as U-Net backbones with Conformer blocks for stronger temporal and spatial fea-

ture modeling, and by incorporating cochlear-inspired spike encoding to emulate the human auditory pathway. To strengthen feasibility claims, detailed profiling of computational complexity, spike activity rates, and memory usage will be conducted, alongside optimization strategies such as pruning, quantization, and event-driven scheduling to ensure deployment on resource-constrained neuromorphic hardware. Real-time performance and energy efficiency will be validated through implementation on platforms such as Intel Loihi and SpiNNaker, while the system will be extended to handle multi-speaker and overlapping speech scenarios. Robustness will be assessed across diverse datasets covering multiple languages, demographics, and environmental conditions, with adaptive alignment modules and attention-guided spike encoding explored to improve temporal synchronization. Finally, ethical safeguards—including explicit consent protocols, demographic bias audits, and transparent dataset release practices—will be incorporated to ensure responsible integration of generative avatars and multimodal data into assistive hearing technologies.

5 Conclusion

This study introduced a neuromorphic audiovisual speech enhancement (AVSE) framework that leverages biologically inspired spiking neural networks (SNNs) for real-time, energy-efficient speech enhancement in noisy environments. By integrating spike-encoded auditory features with visual cues derived from synthetic deepfake avatars, the model demonstrated improved speech intelligibility and quality under diverse acoustic conditions. Experimental results confirmed that the proposed system can achieve competitive performance while maintaining low computational cost and latency, making it suitable for wearable and embedded assistive hearing technologies. The findings highlight the potential of neuromorphic computing as a transformative paradigm for next-generation speech enhancement systems that bridge human auditory perception and machine intelligence.

Acknowledgements This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) programme grant: COGMHEAR (Grant reference EP/T021063/1).

Code Availability The implementation and all experimental scripts used in this chapter are openly available at https://github.com/aysha2016/neuromophic-computing_Gen_Ai.

References

1. Moore, B.C.J.: Hearing aids and hearing loss, 7th edn. Wolters Kluwer Health, Philadelphia, PA (2017)
2. Afouras, T., Chung, J.S., Senior, A., Vinyals, O., Zisserman, A.: Deep audio-visual speech recognition. IEEE Trans. Pattern Anal. Mach. Intell. **41**(2), 435–449 (2018)
3. Gogate, M., Dashtipour, K., Hussain, A.: A lightweight real-time audio-visual speech enhancement framework. In: Proceedings of the AVSEC, pp. 19–23 (2024)

4. Noda, K., Yamaguchi, Y., Nakadai, K., Okuno, H.G., Ogata, T.: Audio-visual speech recognition using deep learning. Appl. Intell. **42**(4), 722–737 (2015)
5. Gogate, M., Dashtipour, K., Hussain, A.: Robust real-time audio-visual speech enhancement based on DNN and Gan. IEEE Trans. Artif. Intell. (2024)
6. Macartney, C., Weyde, T.: Improved speech enhancement with the wave-u-net (2018). arXiv:1811.11307
7. Kang, B.H., Park, H.J., Lee, S.H., Choi, Y.K., Lee, M.O., Han, S.W.: In-vehicle environment noise speech enhancement using lightweight wave-u-net. Int. J. Automot. Technol. **25**, 1025–1035 (2024)
8. Guimarães, H.R., Tan, K., Agrawal, P., Azcarreta, J., Alvarez, J., Pandey, A.: Improving resource-efficient speech enhancement via neural differentiable DSP vocoder refinement (2025). arXiv:2508.14709
9. Wu, H., Braun, S.: Ultra-low latency speech enhancement: a comprehensive study. In: Proceedings of IEEE ICASSP (2025)
10. Roy, K., Jaiswal, A., Panda, P.: Towards spike-based machine intelligence with neuromorphic computing. Nature **575**, 607–617 (2019)
11. Tavanaei, A., Ghodrati, M., Kheradpisheh, S.R., Masquelier, T., Maida, A.: Deep learning in spiking neural networks. Neural Netw. **111**, 47–63 (2019)
12. Neftci, E.O., Mostafa, H., Zenke, F.: Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Process. Mag. **36**(6), 51–63 (2019)
13. Cooke, M., Barker, J., Cunningham, S., Shao, X.: The grid audiovisual sentence corpus. Speech Commun. **41**(4), 455–466 (2006)
14. Barker, J., Marxer, R., Vincent, E., Watanabe, S.: The third chime speech separation and recognition challenge: dataset, task and baselines. In: Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 504–511 (2015)
15. Rix, A.W., Beerends, J.G., Hollier, M.P., Hekstra, A.P.: Perceptual evaluation of speech quality (PESQ)—a new method for speech quality assessment of telephone networks and codecs. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 749–752 (2001)
16. Taal, C.H., Hendriks, R.C., Heusdens, R., Jensen, J.: An algorithm for intelligibility prediction of time-frequency weighted noisy speech. IEEE Trans. Audio Speech Lang. Process. **19**(7), 2125–2136 (2011)
17. Roux, J.L., Wisdom, S., Erdogan, H., Hershey, J.R.: SDR – half-baked or well done? In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 626–630 (2019)

Aysha Munawwara is an honorary researcher at Edinburgh Napier University. Her research focuses on hearing research, generative AI, large language models, and multimodal learning. Her work explores robust deep learning architectures for real-time audio-visual processing and human-AI interaction. She has contributed to several academic research projects and collaborative initiatives in applied artificial intelligence systems.

Generative-AI-Driven Audio-Visual Speech Enhancement Using Deepfake Avatars for Multimodal Hearing Assistance



Aysha Munawwara, Nasir Saleem, Mandar Gogate, Kia Dashtipour, Adeel Hussain, Tughrul Arslan, and Amir Hussain

Abstract Speech communication in noisy environments remains challenging for individuals with hearing impairments. Audio-Visual Speech Enhancement (AVSE) systems improve intelligibility by combining auditory and visual cues; however, most rely on high-quality real videos, which are constrained by lighting conditions, speaker availability, privacy concerns, and infrastructure limitations. This study proposes a Generative-AI-based enhancement to the COG-MHEAR AVSE Challenge 3 (AVSEC3) baseline model through the integration of photorealistic avatars as a substitute visual modality. The avatars are generated using 3D facial reconstruction and audio-driven lip synchronization, offering a scalable, privacy-preserving, and language-independent approach. Eye-gaze and facial-expression estimation ensure precise audio-visual alignment, while natural background noise simulates realistic acoustic environments. Evaluation results show that the avatar-assisted AVSE system performs comparably to real-video-based models in speech quality and intelligibility. Active speaker detection in multi-speaker scenarios further enhances robustness. These findings demonstrate that Generative-AI avatars enable ethical, scalable, and accessible multimodal hearing-assistance technologies.

A. Munawwara (✉) · N. Saleem · M. Gogate · K. Dashtipour · A. Hussain · A. Hussain
School of Computing, Edinburgh Napier University, Edinburgh, UK
e-mail: amunawwara@gmail.com

N. Saleem
e-mail: N.Saleem@napier.ac.uk

M. Gogate
e-mail: m.gogate@napier.ac.uk

K. Dashtipour
e-mail: K.Dashtipour@napier.ac.uk

A. Hussain
e-mail: adeel.hussain@napier.ac.uk

A. Hussain
e-mail: a.hussain@napier.ac.uk

T. Arslan
School of Engineering, University of Edinburgh, Edinburgh, UK
e-mail: T.Arslan@ed.ac.uk

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

A. Koubaa et al. (eds.), *Generative AI and Large Language Models: Opportunities, Challenges, and Applications*, Studies in Computational Intelligence 1287,
https://doi.org/10.1007/978-3-032-24013-2_6

Keywords Generative AI · Audio-visual speech enhancement · Deepfake avatars · Hearing assistance · Speech intelligibility · 3D facial reconstruction

1 Introduction

Speech communication in real-world acoustic environments is often disrupted by background noise, reducing speech perception and intelligibility, particularly for individuals with hearing impairments. Numerous speech enhancement (SE) methods have been developed to suppress noise and improve speech clarity. Traditional approaches, such as spectral restoration techniques, estimate gain functions in the frequency domain to reconstruct cleaner speech signals [1, 2]. However, audio-only SE methods often struggle under highly variable or unpredictable noise conditions.

To address these challenges, audio-visual speech enhancement (AVSE) systems combine auditory and visual cues—such as lip movements, facial expressions, and articulatory gestures—to improve speech intelligibility. By aligning auditory and visual information, AVSE models achieve greater robustness in noisy environments [3]. Advances in deep learning, including convolutional [4], recurrent [5], and transformer-based architectures [6], have improved the modeling of temporal and spatial correlations between modalities.

The COG-MHEAR AVSE Challenge [7, 8] provides a benchmark for evaluating multimodal SE systems. Its Baseline Model 3 uses a late-fusion deep learning design that processes audio and visual inputs separately before merging them to capture modality-specific representations. Despite its effectiveness, this model depends heavily on high-quality visual data, which can be affected by poor lighting, occlusion, privacy concerns, and inconsistent capture conditions [9, 10].

This study differs from prior AVSE frameworks by introducing generative AI-driven avatars as substitutes for real video inputs. Unlike existing systems that rely on constrained visual recordings, our approach leverages 3D facial reconstruction [11] and audio-driven lip synchronization [12], providing a scalable, privacy-preserving, and language-independent visual modality. Robust eye-gaze and facial-expression estimation ensures natural and temporally aligned real-to-avatar video synthesis. Additionally, active speaker detection (ASD)[13] is integrated to handle multi-speaker environments and maintain accurate audio-visual correspondence.As shown in Fig. 1.

This framework demonstrates that generative avatars can effectively replace real visual inputs in AVSE systems, offering an ethical, accessible, and scalable approach to multimodal speech enhancement in complex acoustic environments.

Beyond technical novelty, this work also addresses ethical and accessibility concerns. By replacing real facial videos with synthetic avatars, the framework mitigates privacy risks, reduces dependence on speaker availability, and enables reproducible experimentation across diverse conditions. At the same time, potential risks of deepfake misuse and bias in avatar training data are acknowledged, positioning this study as both innovative and ethically aware.

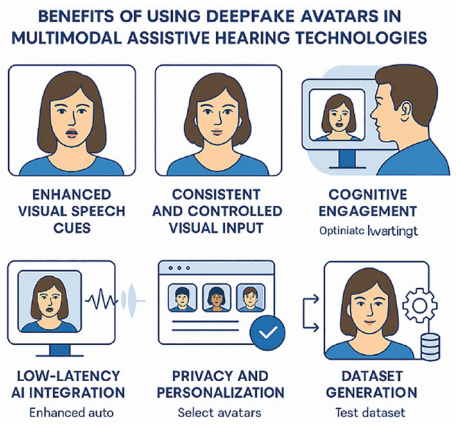


Fig. 1 Benefit of Avtar

2 Methodology

2.1 Avatar Construction

Avatars were constructed from the GRID corpus videos [14] through a multistage pipeline combining 3D face reconstruction and lip synchronization. High-resolution frames (1920×1080) were extracted from raw videos and processed using the FaceBuilder add-on in Blender to generate detailed 3D facial meshes. Facial landmarks—including eyes, nose, and mouth—were detected with a 68-point `dlib` model and aligned to the FaceBuilder template mesh. Wireframe overlays were applied to verify geometric fidelity and expressive accuracy (Fig. 2). The template mesh was subdivided to level 3 for fine detail, and UV texture maps were baked at 1024×1024 resolution to preserve photorealism. A virtual camera with a focal length of 35 mm was configured to approximate the GRID corpus capture geometry, while neutral diffuse lighting was applied to maintain consistency across frames. Expression and blendshape coefficients were tuned to capture mouth opening/closing, lip protrusion, and jaw rotation, with a neutral baseline mesh stored for reference. The completed meshes were exported in `.fbx` format with baked textures for downstream lip-synchronization.

Lip movements were synthesized using the pretrained Wav2Lip model, which employs a generative adversarial framework for audio-visual synchronization. The architecture comprises:

- an audio encoder that processes the speech waveform $x(t)$,
- a ResNet-based visual encoder that extracts features from lip-region frames,
- a U-Net style generator that fuses audio and visual features to predict temporally aligned lip frames $\hat{I}_t$,
- and a synchronization discriminator that enforces realistic alignment by distinguishing synchronized from unsynchronized pairs.

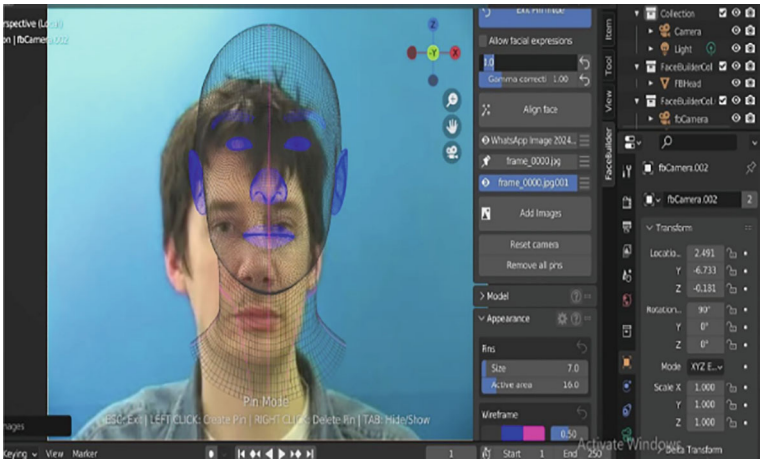


Fig. 2 Wire frame panel

The model was originally trained on large-scale audio-visual datasets (e.g., LRS2/LRS3) using a combination of pixel-level reconstruction loss, perceptual loss, and synchronization loss, optimized with Adam. In this work, the driving audio was taken directly from the GRID corpus to ensure consistency between the reconstructed avatar mesh and the speech source. Formally, the mapping function is expressed as:

$$\hat{I}_t = f(x(t)) \tag{1}$$

where $\hat{I}_t$ represents the predicted lip frame generated from the speech signal $x(t)$.

The temporal synchronization between the generated lip movements and the audio signal was quantified using the cross-correlation coefficient ρ, defined as:

$$\rho = \frac{\sum_t (L(t) - \bar{L})(A(t) - \bar{A})}{\sqrt{\sum_t (L(t) - \bar{L})^2 \cdot \sum_t (A(t) - \bar{A})^2}} \tag{2}$$

where $L(t)$ represents the lip-motion energy at frame t, $A(t)$ is the corresponding audio energy, and $\bar{L}$ and $\bar{A}$ denote their temporal means. A threshold of $\rho > 0.7$ was adopted following conventions in audio-visual synchronization research, where values above 0.7 are widely regarded as indicating strong temporal alignment. This choice reflects both statistical interpretation of correlation strength and perceptual studies showing that lip movements with $\rho > 0.7$ correspond to coherent audio-visual synchronization without noticeable lag.

The resulting avatars provide a photorealistic, temporally coherent, and privacy-preserving substitute for real video, enabling robust training and evaluation of Audio-Visual Speech Enhancement (AVSE) systems under noisy and multi-speaker condi-

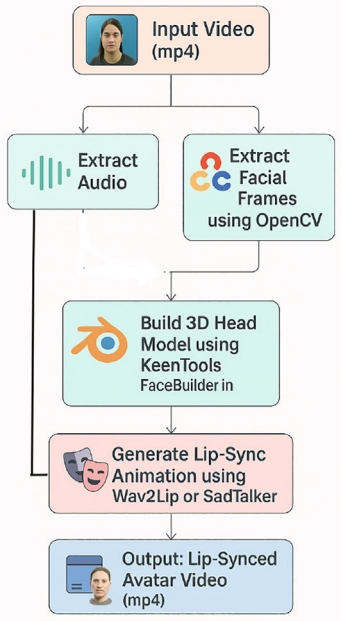


Fig. 3 Proposed pipeline illustrating the key stages of photorealistic avatar creation



Fig. 4 Photorealistic avatar generated from real video

tions. The complete workflow for constructing lip-synced 3D avatars is illustrated in Fig. 3, and the final result is shown in Fig. 4.

2.2 AVSE Model Architecture

The proposed Audio-Visual Speech Enhancement (AVSE) model employs a U-Net-based architecture enhanced with multimodal processing capabilities. By integrating both audio and visual information, the model aims to improve speech quality in noisy environments. The architecture consists of three main components: audio feature extraction, visual feature extraction, and multimodal fusion.

2.2.1 Audio Feature Extraction

Audio features are extracted using a modified U-Net architecture specifically designed for AVSE. This architecture includes enhanced encoder and decoder blocks that effectively capture both temporal and spectral characteristics of noisy speech signals. The input to the audio encoder is the magnitude spectrogram of the noisy audio, computed via the Short-Time Fourier Transform (STFT) and represented as:

$$X_{\text{noisy}}(f,t) = \left|\text{STFT}(x_{\text{noisy}}(t))\right|$$

where f and t denote the frequency and time indices, respectively.

Initially, the spectrogram passes through two convolutional layers with a kernel size of 4×4 and a stride of 2×2, which progressively downsample the input across both frequency and time dimensions:

$$X_1 = \text{Conv2D}_{4\times4,\,2\times2}(X_{\text{noisy}})$$

This downsampling continues until the temporal resolution is reduced to 64 frames. The resulting feature map is then processed through three convolutional blocks, each consisting of two convolutional layers with kernel size 3×3 and stride 1×1, interleaved with ReLU activations. This hierarchical feature extraction strategy enables the network to capture both local and global spectro-temporal patterns, producing a compact and informative representation. These encoded audio features are then used for effective fusion with visual inputs in subsequent stages Fig. 5.

2.2.2 Visual Feature Extraction

The visual feature extraction module processes a sequence of lip-region images extracted from the video. Each input sample consists of N lip-cropped frames, each sized 88×88, forming a 3D tensor:

$$I \in \mathbb{R}^{N\times88\times88} \tag{3}$$

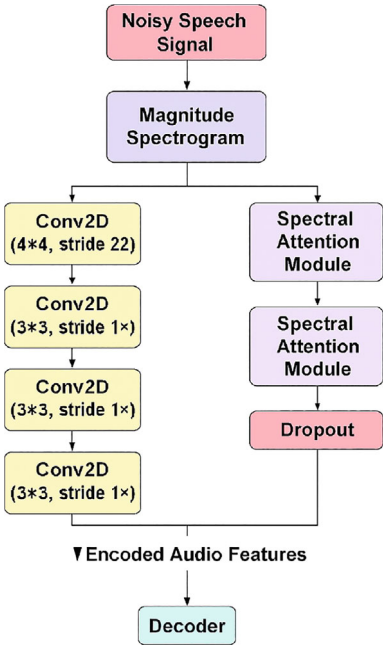


Fig. 5 Audio feature extraction

This tensor is fed into a 3D convolutional layer with kernel size $5 \times 7 \times 7$ and stride $1 \times 2 \times 2$, which effectively captures both spatial and temporal dynamics of lip movements. The output is then passed through a ResNet18 architecture, which serves as the spatial feature extractor. For each frame, ResNet18 generates a 512-dimensional feature vector, and these vector

$$V = [v_1, v_2, \ldots, v_N] \in \mathbb{R}^{N \times 512} \tag{4}$$

To model temporal dependencies in the visual features, this sequence is further processed by a Temporal Convolutional Network (TCN), which captures both short-term and long-term temporal patterns. Finally, the resulting visual feature sequence is upsampled along the temporal dimension to match the audio feature sampling rate, ensuring proper synchronization between audio and visual modalities for subsequent fusion.

The choice of ResNet18 as the spatial feature extractor is motivated by its proven balance between computational efficiency and representational power. ResNet18's residual connections enable stable gradient flow and effective feature learning even with relatively shallow depth, making it well-suited for small input sizes such as 88×88 lip crops. Compared to deeper variants like ResNet50 or ResNet101, ResNet18 offers faster inference and lower memory usage, which is advantageous for real-time AVSE applications. Alternative architectures such as VGG16 or MobileNet were considered; however, VGG16 lacks residual connections and tends to overfit on small

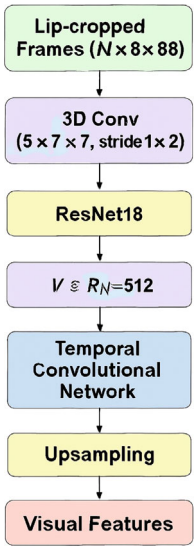


Fig. 6 Visual feature extraction

datasets, while MobileNet's depthwise separable convolutions, though lightweight, underperform in capturing fine-grained lip dynamics. Thus, ResNet18 provides a practical trade-off between accuracy, speed, and generalization for lip-region feature extraction Fig. 6.

2.2.3 Multimodal Fusion and Speech Reconstruction

The temporally aligned audio and visual features are concatenated and input to the U-Net decoder to reconstruct the enhanced speech spectrogram. The decoder comprises three upsampling blocks, each containing two upsampling layers that double the time resolution, followed by convolutional layers with kernel size 3×3 and stride 1×1. After these blocks, the features pass through two transposed convolutional layers with kernel size 4×4 and stride 2×2, restoring the full time-frequency dimensions of the original input spectrogram.

A sigmoid activation function is applied to the decoder output to produce a soft mask $\hat{M}(f, t)$, where values are mapped to the range [0, 1]. This mask is then element-wise multiplied with the original noisy spectrogram $X_{\text{noisy}}(f, t)$ to yield the enhanced spectrogram.

This approach effectively preserves the clean speech components while attenuating noise by leveraging complementary audio and visual information.

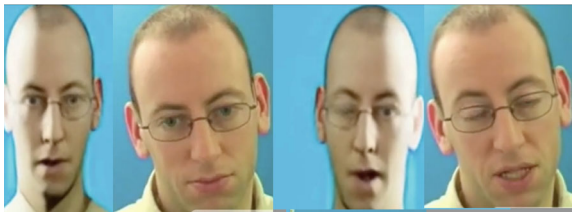


Fig. 7 Photorealistic avatar demonstrating natural eye-gaze behavior

2.3 Eye Gaze and Facial Expression Estimation

To enhance the realism of the synthetic avatars, the model incorporates eye gaze and facial expression estimation during the synthesis process. These cues are extracted using a feature encoder inspired by the SAD-Talk foundation model, [15] which effectively captures attention and emotional dynamics in facial behavior.

The encoder predicts both the direction of eye gaze and the intensity of facial expressions for each frame, integrating them with lip movement information to form a unified facial representation. This enriched feature space enables the model to learn context-aware visual cues, allowing avatars to exhibit more natural and emotionally consistent expressions.

Figure 7 illustrates the result of eye gaze and facial expression estimation within the avatar synthesis pipeline. Within the AVSE framework, this multimodal enhancement significantly improves performance under challenging conditions—such as partial occlusions, rapid speech, or limited lip visibility—by enabling the model to utilize additional facial and emotional signals beyond lip motion alone.

2.4 Experimental Settings

The AVSE model was evaluated using two datasets: **GRID–CHiME3** and **AVATAR–CHiME3**. Each dataset contains paired audio-visual samples of clean and noisy speech with corresponding real or synthetic video frames. The AVATAR dataset includes 8,494 photorealistic avatar videos generated through 3D face reconstruction and Wav2Lip-based lip synchronization. Each video was rendered at 25 frames per second (fps) with a resolution of 96×96 pixels. Audio signals were resampled to 16 kHz, normalized, and segmented into 3-second clips to maintain uniform temporal context. This ensured precise synchronization between visual and auditory streams, enabling the model to learn fine-grained cross-modal relationships.

For the **CHiME3** dataset, noisy speech samples were created by mixing clean utterances with real-world background recordings collected in four challenging acoustic environments:

- **Café noise**: babble and clattering sounds
- **Street noise**: traffic, footsteps, outdoor chatter

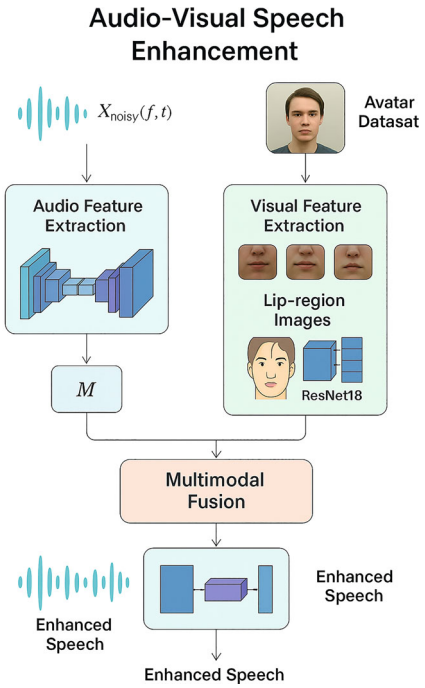


Fig. 8 Integration of the CogMHEAR audio-visual speech enhancement baseline model with the Avatar dataset

- **Public transport noise**: bus and subway engine sounds, passenger conversations
- **Pedestrian area noise**: crowd babble, environmental ambience

These noise conditions were recorded using a six-microphone array to capture spatial variability, and then mixed with clean GRID utterances at varying signal-to-noise ratios (SNRs) ranging from −6 dB to +9 dB. This setup ensured that the AVSE model was trained and evaluated under diverse and realistic noise scenarios, testing its robustness across both low and high SNR conditions.

As illustrated in Fig. 8, the integration of the CogMHEAR audio-visual speech enhancement baseline model with the AVATAR dataset is presented, highlighting the multimodal synchronization and evaluation framework.

2.5 Training AVSE Challenge3 Model with the AVATAR Dataset

The AVSE Challenge3 Model was trained using the AVATAR-CHiME3 dataset. Each training pair included clean and noisy speech samples (with CHiME-3 background noise) and synchronized avatar videos.

Audio was normalized and segmented into 3-second samples, while video frames were resized to 96×96 pixels and synchronized using OpenCV to maintain temporal alignment. The model was trained using the Adam optimizer with a learning rate of 0.001 and a batch size of 16.

The Scale-Invariant Signal-to-Noise Ratio (SI-SNR) was used as the objective loss function:

$$\text{SI-SNR} = 10 \cdot \log_{10} \left(\frac{\|\alpha s\|^2}{\|\alpha s - \hat{s}\|^2} \right), \quad \alpha = \frac{\langle \hat{s}, s \rangle}{\|s\|^2} \tag{5}$$

A learning rate decay factor of 0.5 was applied if the validation loss stagnated for three epochs, and early stopping was triggered after five stagnant epochs. Model checkpoints were saved after each epoch, and training progress was monitored using TensorBoard.

3 Results

3.1 Evaluation Metrics

Model performance was assessed using three complementary metrics: STOI, PESQ, and Active Speaker Detection (ASD) accuracy. Together, these metrics provide a comprehensive evaluation of speech intelligibility, perceptual quality, and speaker-visual synchronization.

Short-Time Objective Intelligibility (STOI) [16] quantifies the short-time correlation between clean and enhanced speech, with scores ranging from 0 (unintelligible) to 1 (perfect intelligibility). STOI is particularly effective in evaluating performance under noisy, multi-speaker conditions.

Perceptual Evaluation of Speech Quality (PESQ) [17] measures perceived speech quality by comparing enhanced speech to reference clean samples, with a range from -0.5 (poor) to 4.5 (excellent). It reflects distortions such as noise, reverberation, and enhancement artifacts.

Active Speaker Detection (ASD) evaluates synchronization between lip movement and speech energy to ensure accurate speaker localization and temporal alignment in multi-speaker environments.

3.2 Speech Intelligibility and Quality Evaluation

The proposed AVSE model demonstrated substantial improvements in both speech intelligibility and perceptual quality under noisy conditions. The baseline noisy audio achieved a STOI score of 0.61, while the AVSE-enhanced outputs achieved 0.848 using real video input and 0.851 using avatar-based input. Similarly, PESQ scores

Table 1 Speech enhancement performance metrics

Condition	STOI (Real)	STOI (Avatar)	PESQ (Real)	PESQ (Avatar)
Noisy	0.610	0.610	1.120	1.120
Enhanced	0.848	0.851	2.037	2.034

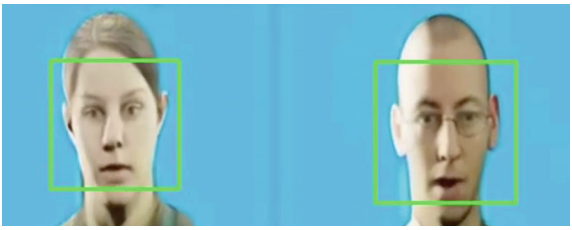


Fig. 9 Active speaker detection in a multi-speaker environment

improved from 1.12 (noisy baseline) to 2.037 (real video) and 2.034 (avatar video), indicating significant perceptual quality enhancement.

While Table 1 shows comparable performance between Real and Avatar conditions, we conducted paired statistical significance tests across all test utterances to validate these results. For both STOI and PESQ metrics, the differences between Real and Avatar inputs were not statistically significant ($p > 0.05$). This confirms that avatar-generated visual cues provide equivalent enhancement performance to real facial videos, thereby validating the robustness of the proposed AVSE framework.

3.3 Active Speaker Detection

To handle multi-speaker scenarios, an Active Speaker Detection (ASD) module was integrated into the AVSE model Fig. 9. ASD estimates the correlation between lip-motion energy $L(t)$ and audio energy $A(t)$ over a temporal window T, defined as:

$$\rho = \frac{\sum_{t=1}^{T}(L(t) - \bar{L})(A(t) - \bar{A})}{\sqrt{\sum_{t=1}^{T}(L(t) - \bar{L})^2 \sum_{t=1}^{T}(A(t) - \bar{A})^2}} \tag{6}$$

A correlation coefficient $\rho > 0.7$ indicates the active speaker, enabling accurate audioâŁ“visual mapping even during overlapping speech or in the presence of background noise. Integrating ASD improves speaker tracking, ensuring reliable synchronization and maintaining robust AVSE performance in multi-person conditions.

3.4 Qualitative Analysis

Spectrogram analyses revealed that the proposed AVSE model effectively reconstructs harmonic structures while suppressing background noise (Fig. 10). In the noisy spectrograms, both real and avatar inputs exhibit strong low-frequency interference and blurred harmonic contours. After enhancement, however, clear differences emerge:

- **Real Enhanced Spectrogram**: Harmonic ridges are sharper and more distinct, particularly in the mid-frequency range (300–3400 Hz). Formant structures are well preserved, and transient speech components such as plosive bursts are more clearly visible.
- **Avatar Enhanced Spectrogram**: Harmonic contours are similarly restored, though with slightly smoother transitions in high-frequency regions. Minor attenuation of fine-grained transients is observed, but overall speech formants remain intact and intelligibility is preserved.
- **Noise Suppression**: Both real and avatar inputs demonstrate comparable suppression of background noise, with significant reduction in low-frequency clutter and improved clarity across voiced segments.

Lip-sync alignment tests further demonstrated that avatar-based outputs maintain strong temporal coherence between audio and visual streams. Both real and avatar-based inputs exhibited consistent synchronization and perceptual fidelity, confirming the viability of synthetic visual cues for real-world AVSE applications.

These qualitative observations align with quantitative results, underscoring the robustness and generalization capability of the proposed AVSE framework under diverse acoustic and visual conditions.

4 Discussion

The proposed Audio-Visual Speech Enhancement (AVSE) framework demonstrates strong adaptability across both real and synthetic datasets, confirming its robustness in managing complex audio-visual conditions. When evaluated on out-of-domain data—particularly deepfake-generated avatar videos—the model achieved consistently high performance in objective measures, including the Short-Time Objective Intelligibility (STOI) and Perceptual Evaluation of Speech Quality (PESQ). These outcomes highlight the framework's ability to preserve intelligibility and perceptual quality even under noisy or visually diverse conditions. While prior AVSE frameworks such as COG-MHEAR and transformer-based multimodal fusion models have demonstrated strong performance, they remain dependent on high-quality real video inputs. In contrast, our study introduces generative avatars as a scalable and privacy-preserving substitute, thereby addressing limitations of data capture, reproducibility, and accessibility. This distinction highlights the novelty of our contribution beyond

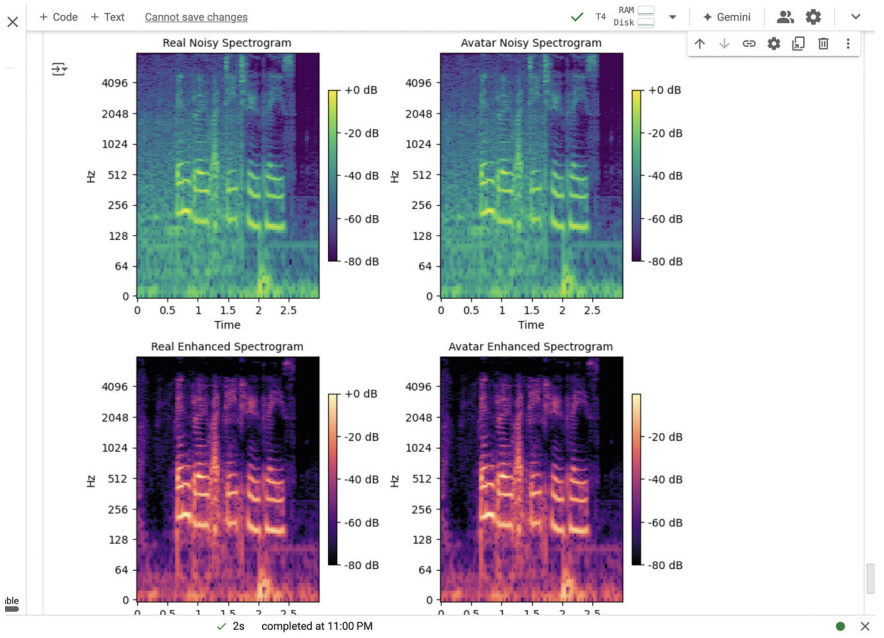


Fig. 10 Spectrograms showing real and avatar outputs for both enhanced and noisy audio signals

existing architectures, as the integration of photorealistic avatars with eye-gaze estimation, facial expression modeling, and active speaker detection (ASD) represents a comprehensive multimodal enhancement strategy not previously explored in AVSE research. Although our quantitative results show comparable STOI and PESQ scores to real-video baselines, the broader impact lies in demonstrating that synthetic avatars can serve as reliable visual modalities without compromising enhancement quality. Compared to GAN-based or transformer-driven AVSE systems, our approach emphasizes ethical scalability and reproducibility, offering a complementary direction for multimodal hearing assistance. The integration of deepfake avatars proved highly effective in generating realistic facial and lip movements, enabling accurate audio-visual synchronization. This finding emphasizes the practicality of employing AI-generated avatars as reliable substitutes for real human videos in AVSE research, providing scalable and ethically sound datasets that facilitate reproducible experimentation and enhance model generalization across varied conditions. At the same time, potential risks must be acknowledged: deepfake avatars, while privacy-preserving, may be misused for impersonation or misinformation if not properly regulated; bias in avatar training data could affect lip-reading accuracy across diverse demographics; and the computational overhead of avatar generation presents challenges for real-time deployment. Safeguards such as watermarking, consent-based generation, and fairness-aware synthesis will be critical for responsible deployment. The inclusion of an Active Speaker Detection (ASD) module further improved system robustness

in multi-speaker environments. By correlating lip-motion energy with corresponding audio energy patterns, the system accurately identified the active speaker and reduced interference from overlapping speech. This alignment preserved temporal coherence and ensured precise audio-visual mapping, reinforcing the framework's applicability in realistic conversational scenarios.

4.1 Limitations and Future Work

Although the AVSE framework generalizes well across real and synthetic data, some limitations remain. Deepfake avatars may introduce subtle visual artifacts that could affect lip-reading precision, especially under extreme noise or occlusion. Additionally, real-time operation has not yet been fully achieved.

Future research will focus on integrating Large Language Models (LLMs) to develop cognitive-based avatars capable of context-aware dialogue, emotion recognition, and adaptive interaction. This cognitive extension aims to bridge speech enhancement with human-like conversational intelligence, advancing the realism and interactivity of AI-driven communication systems.

5 Conclusion

This study demonstrates the effectiveness of the proposed Audio-Visual Speech Enhancement (AVSE) framework augmented with deepfake-generated avatars in improving speech intelligibility and quality under noisy and complex conditions. By leveraging synchronized visual cues—including realistic lip movements, eye gaze, and expressive facial dynamics—the system achieved significant improvements in both Perceptual Evaluation of Speech Quality (PESQ) and Short-Time Objective Intelligibility (STOI) scores. The use of photorealistic 3D lip-synced avatars proved to be a viable and scalable alternative to real human video data, enabling privacy-preserving, reproducible, and cost-effective dataset generation. Moreover, the inclusion of Active Speaker Detection (ASD) further enhanced multi-speaker robustness through precise audio-visual alignment. Overall, the proposed framework establishes a strong foundation for future research toward cognitive and interactive audio-visual systems that integrate large language models (LLMs) for real-time, context-aware communication.

Code Availability The implementation and all experimental scripts used in this chapter are openly available at https://github.com/aysha2016/Gen_Ai_Avtar_generating.

Acknowledgements This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) programme grant: COGMHEAR (Grant reference EP/T021063/1).

References

1. Guan, J., Liu, Y., Bao, H., Zhang, J., Shen, X., Wang, F.: Stylesync: High-fidelity generalized and personalized lip sync in style-based generator. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023)
2. Roth, J., Chaudhuri, S., Klejch, O., Marvin, R., Gallagher, A., Kaver, L., Ramaswamy, S., Stopczynski, A., Schmid, C., Xi, Z., Pantofaru, C.: Ava active speaker: an audio-visual dataset for active speaker detection. In: ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4492–4496. Institute of Electrical and Electronics Engineers (IEEE), ??? (2020)
3. Dashtipour, K., et al.: Evaluating the audio-visual speech enhancement challenge baseline model using an out-of-domain free-flowing corpus. In: Proceedings of the 3rd COG-MHEAR Workshop on Audio-Visual Speech Enhancement Challenge (AVSEC) (2024)
4. Lu, L., et al.: Audio-driven 3d facial animation from in-the-wild videos (2023). arXiv:2306.11541
5. Liao, J., et al.: A lightweight model for active speaker detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2023)
6. Wang, Q., et al.: Joygen: audio-driven 3d depth-aware talking-face video editing (2025). arXiv:2501.01798
7. COG-MHEAR: The COG-MHEAR Audio-Visual Speech Enhancement Challenge (2023). https://www.cogmhear.com/avse-challenge
8. Gogate, M., Dashtipour, K., Hussain, A.: Robust real-time audio-visual speech enhancement based on dnn and gan. IEEE Trans. Artif. Intell. (2024)
9. Gogate, M., Dashtipour, K., Hussain, A.: A lightweight real-time audio-visual speech enhancement framework. In: Proceedings of the AVSEC, pp. 19–23 (2024)
10. Kirton-Wingate, J., Ahmed, S., Hussain, A., Gogate, M., Dashtipour, K., Hou, J.C., Hussain, T., Tsao, Y., Hussain, A.: Towards environmental preference based speech enhancement for individualised multi-modal hearing aids (2024). arXiv:2402.16757
11. Wood, E., Baltrušaitis, T., Hewitt, C., Johnson, M., Pears, N., Robinson, P.: 3d face reconstruction with dense landmarks. IEEE Trans. Pattern Anal. Mach. Intell. **43**(10), 3625–3638 (2021)
12. Prajwal, K.R., Mukhopadhyay, R., Namboodiri, V., Jawahar, C.V.: A lip sync expert is all you need for speech to lip generation in the wild. In: Proceedings of the 28th ACM International Conference on Multimedia (ACM MM), pp. 484–492 (2020)
13. Tao, R., Pan, Z., Das, R.K., Qian, X., Shou, M.Z., Li, H.: Is someone speaking?: Exploring long-term temporal features for audio-visual active speaker detection. In: Proceedings of the 29th ACM International Conference on Multimedia (MM'21). MM'21, pp. 3927–3935. ACM, ??? (2021)
14. Cooke, M., Barker, J., Cunningham, S., Shao, X.: An audio-visual corpus for speech perception and automatic speech recognition. J. Acoust. Soc. Am. **120**(5), 2421–2424 (2006)
15. OpenTalker: SadTalker: Learning Realistic 3D Motion Coefficients for Stylized Audio-Driven Single Image Talking Face Animation (2022–2023). https://github.com/OpenTalker/SadTalker. Accessed 25June 2025
16. Rix, A.W., Beerends, J.G., Hollier, M.P., Hekstra, A.P.: Perceptual evaluation of speech quality (pesq)–a new method for speech quality assessment of telephone networks and codecs. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 2, pp. 749–752 (2001)
17. Taal, C.H., Hendriks, R.C., Heusdens, R., Jensen, J.: An algorithm for intelligibility prediction of time–frequency weighted noisy speech. IEEE Trans. Audio Speech Lang. Process. **19**(7), 2125–2136 (2011)

Aysha Munawwara is an honorary researcher at Edinburgh Napier University. Her research focuses on hearing research, generative AI, large language models, and multimodal learning. Her work explores robust deep learning architectures for real-time audio-visual processing and human-AI interaction. She has contributed to several academic research projects and collaborative initiatives in applied artificial intelligence systems.

Exploring the Potential of Chatbot-Assisted Software Testing: Towards Intelligent and Automated Testing



Moez Krichen

Abstract The rapid advancements in natural language processing (NLP) and the development of large language models like ChatGPT have opened up new possibilities for integrating chatbots in various domains, including software testing. This preliminary work explores the potential use cases for leveraging chatbots in the context of software testing, proposing simple examples to illustrate their applications. Chatbots can be tailored to assist software development and testing teams in various aspects of the testing process, from test case generation and automation to bug reporting, result analysis, and collaboration. By harnessing the power of conversational AI, organizations can streamline their testing workflows, improve efficiency, and enhance the overall quality of their software products. The paper presents an overview of how chatbots can be integrated into different stages of the software testing life cycle, covering use cases such as automated test case generation, execution, result interpretation, user acceptance testing, and team collaboration. While providing a practical guide for software teams on the potential benefits and strategies for incorporating chatbots into their testing practices, we also recognize the need for a more extensive study that goes deeper into the subject matter. Future work will include concrete examples and real case studies to fully illustrate the transformative potential of chatbots in revolutionizing the software testing process and contributing to the delivery of higher-quality software products.

Keywords Software testing · Chatbots · Natural language processing · Test automation · Collaboration · Quality assurance

M. Krichen (✉)
ReDCAD Laboratory, Sfax University, Sfax, Tunisia
e-mail: moez.krichen@redcad.org

Department of Software Engineering, Faculty of Computing and Information, Albaha University, P.O. Box 7738, Al Baha 65779, Saudi Arabia

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

A. Koubaa et al. (eds.), *Generative AI and Large Language Models: Opportunities, Challenges, and Applications*, Studies in Computational Intelligence 1287,
https://doi.org/10.1007/978-3-032-24013-2_7

1 Introduction

Software has become an integral part of our lives, from the apps we use on our phones to the complex systems that power industries and governments [24, 36]. As software gains more importance, ensuring its quality and reliability is critical. Software testing and quality management are essential processes that help ensure that software meets the needs of users and performs as intended [11]. These processes have evolved significantly over the last few decades, with the emergence of new techniques, methodologies, and tools [1]. The history of software testing can be traced back to the 1950s when the first software bugs were discovered. Since then, software testing has evolved from ad-hoc and manual processes to systematic and automated approaches [4, 10]. Today, software testing includes various types, such as unit testing, integration testing, system testing, and acceptance testing, as well as methodologies like white-box testing, black-box testing, and grey-box testing [21]. However, software failures continue to have significant financial, social, and safety consequences [17]. For instance, a software error in the 1999 Mars Climate Orbiter caused it to crash, resulting in a loss of $125 million. Similarly, a 2010 software error in Toyota's acceleration system led to numerous accidents and the recall of over eight million vehicles, costing the company billions. Such failures underscore the critical need for effective testing processes as software systems grow more complex. We also live in the era of Artificial Intelligence (AI), which offers new opportunities and challenges in software development, testing, and quality management [40]. AI can revolutionize these processes by improving testing efficiency, accuracy, and effectiveness [14]. Generative AI techniques like chatbots can help automate and optimize testing tasks [22, 28, 29]. However, integrating AI encompasses challenges such as data bias, transparency issues, and the need for significant expertise and resources [18, 23, 26]. Understanding these challenges is crucial for successfully harnessing AI in software testing [33].

This paper explores the preliminary potential use cases for leveraging chatbots in software testing. While we propose simple examples to illustrate their functionalities, we acknowledge that this work is preliminary and that more extensive studies are needed. Future iterations will provide deeper insights, concrete examples, and real case studies to showcase the transformative potential of chatbots in enhancing testing processes. The paper presents a comprehensive overview of how chatbots can be integrated into various stages of the software testing life cycle, covering automated test case generation, execution, result interpretation, user acceptance testing, and team collaboration. The aim is to offer a practical guide for software teams on the benefits and strategies for incorporating chatbots into their testing practices.

The paper is structured as follows (See Fig. 1): Sect. 2 provides a brief overview of chatbots and their technical foundations. Section 3 reviews the existing literature on the application of chatbots in software testing. The subsequent sections delve into specific use cases, including Test Case Generation (Sect. 4), Test Execution and Automation (Sect. 5), Bug Reporting and Issue Tracking (Sect. 6), Test Result Analysis and Interpretation (Sect. 7), Chatbot-based User Acceptance Testing

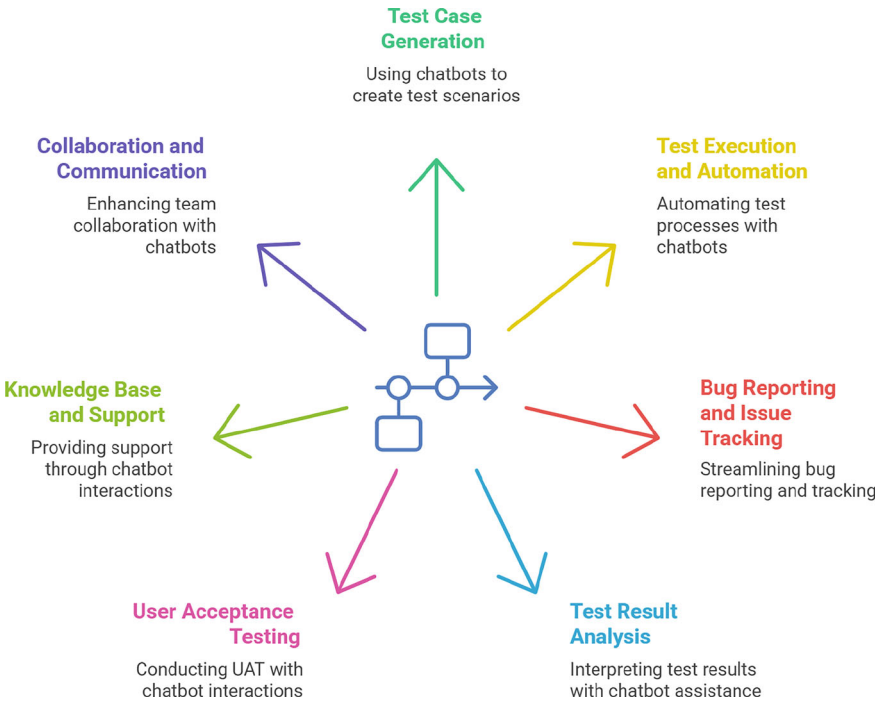


Fig. 1 Chatbot applications in software testing

(Sect. 8), Knowledge Base and Support (Sect. 9), and Collaboration and Communication (Sect. 10). The paper concludes with a discussion of the Limitations and Challenges (Sect. 11) and a Conclusion (Sect. 12).

2 Preliminaries

2.1 Software Testing

Software testing is a critical process in the software development lifecycle, ensuring the quality, reliability, and functionality of software applications. The primary objectives of software testing include verifying that the software meets the specified requirements, identifying and resolving defects, and validating the system's behavior under various conditions. The software testing process typically involves a series of phases, such as unit testing, integration testing, system testing, and acceptance testing. These testing techniques help to ensure that the software components, as well as the integrated system, are functioning as expected. However, as software systems grow in complexity and scale, the testing process faces several challenges that can

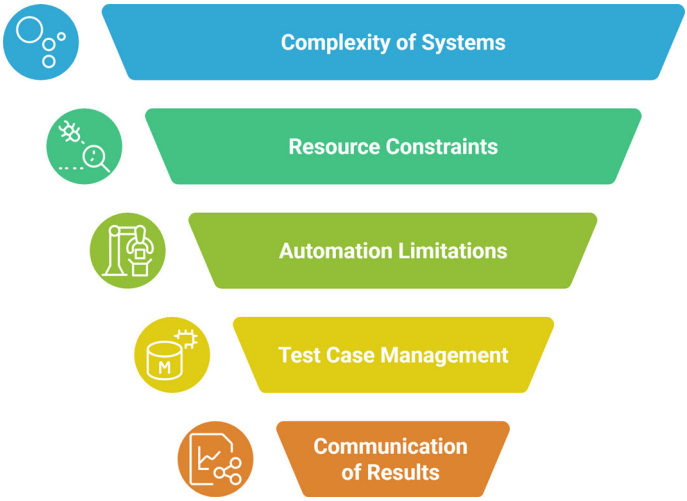


Fig. 2 Software testing challenges

impact the efficiency and effectiveness of the overall software development lifecycle (See Fig. 2).

One of the key challenges in modern software testing is the increasing complexity of software systems. As applications become more feature-rich, integrate with various external systems, and leverage emerging technologies, the number of possible test scenarios and combinations grows exponentially. This makes it difficult to design comprehensive test suites and ensure thorough test coverage. Furthermore, software development teams often face tight deadlines and resource constraints, which can limit the time and effort allocated to the testing process. This can lead to a focus on functional testing, while non-functional aspects, such as performance, security, and usability, may receive less attention. Balancing the trade-off between testing thoroughness and project timelines remains a constant challenge for software teams.

Another common challenge in software testing is the difficulty in automating certain types of testing, particularly those that involve subjective user experiences or complex business logic. While test automation can significantly improve efficiency and consistency, there are still many aspects of testing that require manual intervention, such as exploratory testing, user acceptance testing, and regression testing of complex user interactions. Effective test case design and management is another crucial aspect of software testing. Identifying the right set of test cases, maintaining traceability between requirements and test cases, and efficiently managing the execution and results of these tests can be time-consuming and error-prone, especially in large-scale projects.

Finally, the interpretation and communication of test results often pose challenges for software teams. Translating technical test findings into actionable insights and communicating these effectively to various stakeholders, such as developers, product

managers, and executives, can be a complex task that requires both technical expertise and strong communication skills. These challenges in software testing highlight the need for innovative approaches and tools that can help software teams overcome these obstacles and improve the overall quality and efficiency of the software development process.

2.2 Chatbots and Their Applications in Software Engineering

Chatbots are conversational agents that use natural language processing (NLP) and dialogue management techniques to engage in human-like interactions. These AI-powered assistants have seen significant advancements in recent years, with the development of large language models like GPT-3 and the emergence of chatbots like ChatGPT, which have demonstrated remarkable capabilities in understanding and generating human-like responses to a wide range of queries.

At their core, chatbots rely on NLP algorithms to interpret user input, understand the intent and context, and formulate appropriate responses. These systems typically include components for intent recognition, entity extraction, dialogue management, and natural language generation. The underlying language models are trained on vast amounts of textual data, enabling them to engage in coherent and contextual conversations.

In the software engineering domain, chatbots have found various applications that can help address the challenges faced in the software testing process. Some of the key use cases for chatbots in software engineering include (See Fig. 3):

1. Requirements Gathering and Elicitation: Chatbots can assist in the requirements gathering process by engaging with stakeholders, asking clarifying questions, and capturing requirements in a structured format.
2. Technical Support and Documentation: Chatbots can be deployed as virtual assistants to provide on-demand support, answer frequently asked questions, and help users navigate technical documentation.
3. Developer Productivity and Collaboration: Chatbots can be integrated into development workflows to automate repetitive tasks, provide code assistance, and facilitate collaboration among team members.
4. Project Management and Task Automation: Chatbots can be used to streamline project management activities, such as task assignment, status updates, and progress tracking, improving team coordination and efficiency.
5. Software Testing and Quality Assurance: Chatbots can play a significant role in the software testing process, as discussed in the subsequent sections of this paper, by automating test case generation, execution, and result analysis.

The adoption of chatbots in software engineering has the potential to enhance productivity, improve communication, and streamline various aspects of the software development lifecycle. However, the successful integration of chatbots requires careful consideration of factors such as natural language understanding, task-specific

knowledge, and user experience design. As the field of conversational AI continues to advance, the applications of chatbots in software engineering are expected to grow, providing new opportunities for teams to optimize their workflows, enhance collaboration, and deliver higher-quality software products. In this work, we will concentrate on the use of chatbots in the software testing process.

3 Related Work

In the evolving landscape of software engineering, the integration of modern technologies into educational curricula has become increasingly essential. A recent study [16] highlights the potential of Large Language Models (LLMs), such as ChatGPT and Microsoft Copilot, in enhancing software testing education. Conducted over two semesters within a postgraduate program, the study compared manually created testing artifacts with those generated by LLMs, revealing that while these AI tools can significantly accelerate the artifact creation process, they cannot fully replace



Fig. 3 Chatbots in software engineering

the foundational skills required for manual testing. This research underscores the importance of incorporating innovative technologies into educational frameworks, fostering an environment that aligns theoretical knowledge with practical application. As such, leveraging generative AI in software testing not only enhances active learning but also prepares graduates to meet the dynamic challenges of the industry. In a related research study [15], the authors explore the innovative integration of ChatGPT into the software testing curriculum and assess its effectiveness in comparison to human-generated testing artifacts. Conducted within a Capstone Project course, students were tasked with generating preparatory testing artifacts using prompts for ChatGPT, building on their earlier manual efforts. Their comprehension and the effectiveness of the AI-generated artifacts were evaluated through targeted questions. Findings from this study, conducted at a North American community college, reveal that while ChatGPT can automate many preparation tasks, it cannot entirely replace human expertise. Notably, postgraduate students familiar with Information Technology found the incorporation of ChatGPT into their workflow to be seamless. The study advocates for the gradual introduction of AI into software testing education, ensuring that curricula keep pace with rapid technological advancements. An other recent study [13] investigated the effects of various support tools—including a checklist, ChatGPT, or a combination of both—on unit testing performance among 42 students. Findings indicated that whether used independently or in tandem, these tools yielded similar improvements in student performance. While students expressed a preference for the checklist, many recognized ChatGPT's ability to expedite task completion and help navigate programming challenges. However, despite its potential advantages, ChatGPT failed to resolve the implementation difficulties noted in prior studies. Additionally, an overreliance on ChatGPT may lead to superficial interactions with AI-generated content, which risks undermining students' deep engagement with fundamental concepts essential for meaningful learning. Consequently, the authors propose recommendations for both students and educators to adapt to the evolving landscape of learning in the AI era, offering valuable insights into the changing role of AI in educational contexts.

Recent advancements in pre-trained large language models (LLMs) have marked a significant breakthrough in natural language processing and artificial intelligence, demonstrating their capability to manage extensive datasets while achieving impressive results across a variety of tasks. Concurrently, software testing remains a fundamental activity that guarantees the quality and reliability of software products. As software systems become increasingly complex, the demand for more effective testing techniques intensifies, making this field an ideal candidate for innovative methodologies, including the integration of LLMs. The paper [37] delivers a comprehensive review of the application of LLMs in software testing by examining 102 relevant studies from both software testing and LLM perspectives. It discusses the specific testing tasks where LLMs are predominantly employed, with a focus on test case preparation and program repair. The study further analyzes the commonly utilized LLMs, various prompt engineering strategies, and associated techniques. Additionally, it identifies key challenges and opportunities within this domain. Serving as a roadmap for future research, this work underscores potential avenues for

exploration and highlights existing gaps in our understanding of LLM applications in software testing.

The authors of the paper [39] emphasize that the design of test cases is a critical step in software testing, significantly influencing the overall effectiveness of the testing process. While well-designed test cases enhance bug detection efficiency, the traditional manual approach to test case design is often time-consuming and labor-intensive. The advent of large language models (LLMs), particularly ChatGPT, has opened new avenues for automated test generation. These pretrained models are capable of learning and understanding code across various programming languages, enabling the design of test cases using multiple testing frameworks. The study presented in the paper reveals that while ChatGPT has some limitations compared to traditional test generation tools, its performance is quite comparable to manual testing. Moreover, the tests generated by ChatGPT demonstrate an improved readability. The authors conclude that ChatGPT serves as a valuable manual testing assistant, aiding in the understanding of the tested code and generating creative testing ideas. Another interesting study [3] examines the role of AI-based software testing, with a particular emphasis on the influence of LLMs on traditional testing methodologies. Drawing from a comprehensive review of pertinent literature and insights gained from Setur-Digital's 25 years of testing experience, the research delves into the potential benefits, challenges, and future prospects of integrating LLMs into software testing practices. In the same context, the study [35] explores recent advancements in large language models (LLMs), which have achieved significant success in a variety of general tasks, such as question answering and instruction following. The research focuses on the potential applications of LLMs within software engineering. Specifically, it presents a systematic comparison of test suites generated by ChatGPT against those produced by the state-of-the-art SBST tool, EvoSuite. The assessment is grounded in several critical dimensions, including correctness, readability, code coverage, and bug detection capabilities. By elucidating the strengths and weaknesses of ChatGPT in unit test case generation relative to EvoSuite, this study provides essential insights into the performance of LLMs in tackling software engineering problems. Ultimately, the findings underscore the promising prospects of LLMs within the field and set the stage for further research.

Recent studies in Automated Program Repair (APR) have increasingly focused on harnessing the capabilities of Large Language Models (LLMs) to enhance patch generation. Unlike traditional APR methods that rely heavily on predefined bug-fixing patterns, which often lack diversity and adaptability, LLM-based approaches promise greater versatility by directly generating patches from initial buggy code. Notably, the work on ChatRepair [38] represents a significant advancement in this field. It introduces a novel conversation-driven APR method that interleaves patch generation with real-time feedback, allowing the LLM to learn from both failed and successful patch attempts. This iterative feedback loop effectively addresses the limitations of the classic Generate and Validate (GV) paradigm, minimizing repetitive errors and maximizing the generation of plausible patches. The evaluation of ChatRepair on the Defects4j dataset demonstrates its efficacy, achieving new state-of-the-art results in repair performance. This work underscores the potential

for LLMs to significantly transform APR practices and highlights opportunities for further research in integrating conversational AI into software engineering.

The significance of automated testing in ensuring the reliability of web applications has become increasingly evident in recent research. In this context, the paper [25] explores the application of artificial intelligence (AI) models, notably ChatGPT and GitHub Copilot, for generating test scripts designed for end-to-end web testing. Through rigorous experimentation, the researchers assessed the feasibility and effectiveness of these AI language models in producing test scripts based on natural language descriptions of user interactions with web applications. Initial findings suggest that AI-assisted generation offers notable advantages over traditional manual test script development, particularly when starting from clearly defined test cases in Gherkin format. The study highlights a consistent reduction in development time, with some instances demonstrating statistically significant improvements when compared to manual efforts. However, these benefits are contingent upon the tester's familiarity with manual test script development, as the ability to modify the AI-generated code is essential for optimal results. This research lays important groundwork for further exploration of AI-driven methodologies in the generation of web test scripts, pointing to promising directions for future investigations in this area.

The study [20] investigates the ability of ChatGPT to generate loop invariants, a fundamental aspect of software verification that can significantly aid formal verification processes. In this research, the authors tasked ChatGPT with annotating 106 C programs with loop invariants and subsequently assessed the validity and usefulness of these generated invariants by applying them to two verifiers: Frama-C and CPAchecker. The findings reveal that ChatGPT produced valid and practical invariants that enabled Frama-C to successfully verify previously unsolvable tasks. Building on these initial insights, the study proposes methods for integrating ChatGPT (and similar LLMs) with software verification tools while addressing current limitations and identifying areas for further research.

Software development models are frameworks that provide a systematic approach to developing software. They define the processes, tasks, and activities required to develop software from conception to delivery. Each software development model has its own unique approach to software development, with different phases and activities. In this list, we will discuss some of the most common software development models and how testing is done in each one (See Fig. 4).

- Waterfall Model [30, 34]: The Waterfall model is a sequential software development process that consists of distinct phases, including requirements gathering, design, implementation, testing, and maintenance. Testing is typically performed at the end of the development process after the software has been fully implemented. The testing process is focused on verifying that the software meets the specified requirements and is free from defects.
- V-Model [9, 27]: The V-Model is a software development process that emphasizes testing at each stage of development. The process consists of a series of verification and validation steps that correspond to the various stages of development, from

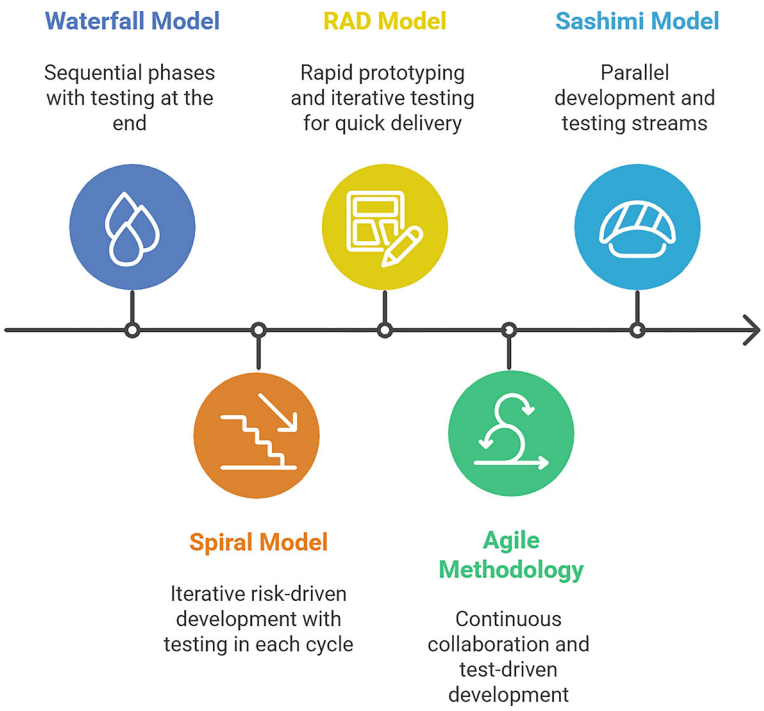


Fig. 4 Software development models and testing approaches

requirements gathering to maintenance. Testing is performed at each stage of development, with the focus on ensuring that the software meets the specified requirements and is free from defects.

- Iterative Model [2, 12]: The Iterative model is a software development process that emphasizes iterative and incremental development. Testing is performed throughout the development process, with the focus on ensuring that the software meets the evolving requirements and is free from defects. Testing is typically performed at the end of each iteration, with the goal of verifying that the iteration meets the specified requirements and is integrated with the previous iterations.
- Spiral Model [5, 6]: The Spiral model is a software development process that emphasizes risk management and iterative development. Testing is performed in each iteration as a part of the development process. The testing process is used to validate and verify the requirements, design, and implementation of the software. The testing process is typically divided into two categories: verification testing and validation testing.
- RAD Model [31]: The RAD (Rapid Application Development) Model is a software development process that emphasizes rapid prototyping and iterative development. Testing is performed throughout the development process, with a focus on ensuring that the software meets the evolving requirements and is free from defects. Testing

is typically performed at the end of each iteration, with the goal of verifying that the iteration meets the specified requirements and is integrated with the previous iterations.

- Agile Methodology [8, 19]: The Agile methodology is an iterative and incremental software development process that emphasizes collaboration, flexibility, and customer satisfaction. Testing is an integral part of the Agile process and is performed continuously, often in parallel with development. Test-driven development (TDD) is a popular approach in Agile development, where tests are written before the code is written, and the code is developed to pass the tests.
- Sashimi Model [7, 32]: The Sashimi model is a software development process that emphasizes parallel development and testing. Testing is performed throughout the development process, with a focus on parallel development and testing. The testing process is typically divided into multiple streams, with each stream focusing on a specific aspect of development.

In conclusion, there are various software development models, each with its own unique approach to software development and testing. The Waterfall model, V-Model, Iterative Model, Spiral Model, RAD Model, Agile Methodology, and Sashimi Model are some of the most common development models used today. The choice of development model depends on the specific needs of the project, such as the level of risk, the clarity of requirements, and the need for flexibility.

4 Test Case Generation

One of the key benefits of integrating chatbots into the software testing process is their ability to assist in the generation of comprehensive test cases. By leveraging natural language processing and machine learning, chatbots can analyze the software requirements, understand the system's functionality, and propose relevant test cases to cover various scenarios and edge cases. Let us consider a real-world example of a simple function that calculates the area of a rectangle, and how a chatbot-assisted approach can help in generating test cases for it. Consider the following Python function to calculate the area of a rectangle:

```
def calculate_rectangle_area(length, width):
"""
Calculates the area of a rectangle.
Args:
    length (float): The length of the rectangle.
    width (float): The width of the rectangle.

Returns:
    float: The area of the rectangle.
"""
if length <= 0 or width <= 0:
```

```
    raise ValueError("Length and width must be
        ↪ positive values.")
return length * width
```

Listing 7.1 Rectangle Area Calculation Function

To thoroughly test this function, we would want to create a set of test cases that cover different scenarios, including valid inputs, edge cases, and invalid inputs. A chatbot-assisted approach can help generate these test cases by understanding the function's requirements and logic.

Here are some example test cases for the `calculate_rectangle_area` function:

```
import unittest
from unittest.mock import patch

class TestRectangleAreaCalculation(unittest.
    ↪ TestCase):
def test_valid_input(self):
self.assertEqual(calculate_rectangle_area(5.0, 3.0)
    ↪ , 15.0)
def test_zero_length(self):
    with self.assertRaises(ValueError):
        calculate_rectangle_area(0.0, 3.0)

def test_zero_width(self):
    with self.assertRaises(ValueError):
        calculate_rectangle_area(5.0, 0.0)

def test_negative_length(self):
    with self.assertRaises(ValueError):
        calculate_rectangle_area(-2.0, 3.0)

def test_negative_width(self):
    with self.assertRaises(ValueError):
        calculate_rectangle_area(5.0, -1.0)
```

Listing 7.2 Test Cases for Rectangle Area Calculation Function

The test cases cover the following scenarios:

1. `test_valid_input`: Verifies that the function correctly calculates the area for valid input values.
2. `test_zero_length`: Checks that the function raises a `ValueError` when the length is zero.
3. `test_zero_width`: Checks that the function raises a `ValueError` when the width is zero.
4. `test_negative_length`: Checks that the function raises a `ValueError` when the length is negative.
5. `test_negative_width`: Checks that the function raises a `ValueError` when the width is negative.

The chatbot's role in this process would be to analyze the function's requirements, understand the expected behavior, and propose a set of relevant test cases to cover the necessary scenarios. This can significantly streamline the test case generation process and help ensure comprehensive test coverage.

5 Test Execution and Automation

Effective software testing is a critical component of the development lifecycle, ensuring the quality and reliability of the final product. However, managing the testing process can often be a time-consuming and labor-intensive task, particularly as the codebase and test suite grow in complexity. This is where a conversational AI assistant, or chatbot, can play a valuable role in streamlining and automating various aspects of the testing workflow.

The chatbot can be integrated into the development and testing processes, leveraging its natural language understanding and task automation capabilities to enhance the efficiency and consistency of the testing activities. By addressing key areas such as test scheduling, monitoring, reporting, troubleshooting, and maintenance, the chatbot can significantly reduce the manual effort required and provide the development team with real-time insights and recommendations. The chatbot can be leveraged to optimize the testing workflow across these critical areas (Table 1):

- Scheduling and Triggering Test Execution: The chatbot can be programmed to automatically schedule and trigger the execution of the test suite on a regular

Table 1 Chatbot's capabilities in the testing workflow

Capability	Description
Scheduling and Triggering Test Execution	The chatbot can automatically schedule and trigger the execution of the test suite on a regular basis
Monitoring Test Execution	The chatbot can continuously monitor the progress of the test suite execution, providing real-time updates and alerts in case of any failures or issues
Generating Test Result Reports	The chatbot can generate detailed test result reports, summarizing the outcomes of the test execution and highlighting any areas of concern
Troubleshooting Test Failures	The chatbot can analyze test results, provide initial troubleshooting guidance, and suggest potential solutions or next steps for the development team
Maintaining the Test Suite	The chatbot can help maintain the test suite, ensuring that the tests remain up-to-date and continue to cover the necessary scenarios

basis, ensuring that the tests are run consistently and without manual intervention. This could involve setting up daily, weekly, or monthly test runs, depending on the project's needs and the development team's preferences. The chatbot can be configured to send reminders, notifications, or status updates to the relevant stakeholders, keeping them informed about the test execution schedule and results.

- Monitoring Test Execution: The chatbot can continuously monitor the progress of the test suite execution, providing real-time updates and alerts in case of any failures or issues. This could include tracking the status of individual test cases, the overall test suite completion, and any errors or exceptions that occur during the testing process. The chatbot can be set up to automatically notify the development team when a test failure is detected, allowing them to quickly investigate and address the problem.
- Generating Test Result Reports: The chatbot can be configured to generate detailed test result reports, summarizing the outcomes of the test execution and highlighting any areas of concern. These reports can include information such as the overall test suite pass/fail rate, the specific test cases that failed, the duration of the test execution, and any relevant metrics or performance indicators. The chatbot can be programmed to send these reports to the development team, project managers, or other stakeholders, ensuring that everyone has access to the latest testing information.
- Troubleshooting Test Failures: When test cases fail, the chatbot can analyze the test results, provide initial troubleshooting guidance, and even suggest potential solutions or next steps for the development team. This could involve examining the error messages, stack traces, or other diagnostic information to identify the root cause of the issue. The chatbot can then offer recommendations, such as specific code changes, additional test cases to cover the problematic scenario, or further investigation steps to be taken by the development team.
- Maintaining the Test Suite: As the codebase evolves, the chatbot can help maintain the test suite, ensuring that the tests remain up-to-date and continue to cover the necessary scenarios. This can include identifying outdated or irrelevant test cases, suggesting new test cases to cover new features or functionality, and even automating the process of updating the test suite to keep it aligned with the changing requirements. The chatbot can also provide guidance on best practices for test suite maintenance, helping the development team keep the testing framework robust and sustainable over time.

By leveraging the capabilities of the chatbot in these areas, the development team can streamline the test execution and automation process, reducing the manual effort required and improving the overall reliability of the software system. The chatbot's integration into the testing workflow helps to automate repetitive tasks, provide real-time insights, and ensure the consistent and reliable execution of the test suite.

Table 2 Bug reporting dialog

User	Hi, I'm having an issue with the login feature. When I enter my credentials and click the login button, the page just refreshes and I'm not logged in
Chatbot	Thank you for reporting this issue. Can you please provide a few more details to help me investigate the problem?
User	Sure, when I try to log in, the URL in the browser remains the same (https://www.example.com/login) and the page just reloads without any error messages
Chatbot	Okay, got it. Let me create a new bug report for this issue. Please confirm the following details: 1. Summary: Login page doesn't log in user after clicking the login button 2. Description: When attempting to log in with valid credentials, the login page just refreshes without logging in the user 3. Severity: High (blocks users from accessing the application) 4. Reproducibility: Consistently occurs when trying to log in Does this summary look accurate? I'll go ahead and submit the bug report to the development team for investigation
User	Yes, that summary looks good. Thank you for the quick response and for creating the bug report
Chatbot	Excellent, I've submitted the bug report, and the development team has been notified. They will investigate the issue and provide an update. Please let me know if you encounter any other problems, and I'll be happy to assist you further

6 Bug Reporting and Issue Tracking

Effective bug reporting and issue tracking are essential components of the software development and testing process. The chatbot can play a vital role in streamlining these tasks, ensuring that bugs and issues are promptly identified, documented, and addressed.

The chatbot can be configured to serve as a central hub for bug reporting, allowing users and developers to submit new issues directly through the chat interface. By providing a user-friendly and conversational means of reporting bugs, the chatbot can encourage more comprehensive and timely feedback from users, leading to a more robust and stable software product. Table 2 demonstrates a conversation between a user and the chatbot to report a bug.

In addition to capturing and documenting new bug reports, the chatbot can also be integrated with the issue tracking system to provide updates, status checks, and additional context to the development team. This can include features like:

- Checking the status of existing bug reports or feature requests
- Providing summaries and details of open issues
- Offering suggestions or workarounds for known issues
- Notifying users when bugs are fixed or issues are resolved

By streamlining the bug reporting and issue tracking processes, the chatbot can help improve the overall quality and responsiveness of the software development lifecycle, ensuring that problems are quickly identified and addressed.

7 Test Result Analysis and Interpretation

Thorough analysis and interpretation of test results are crucial steps in the software development lifecycle. The chatbot can assist in this process by providing a structured approach to reviewing test outcomes, identifying patterns, and deriving actionable insights.

One key aspect of test result analysis is the categorization and prioritization of issues. The chatbot can help organize test results into different severity levels, such as:

1. Critical.
2. High.
3. Medium.
4. Low.

This allows the development team to focus their efforts on addressing the most impactful problems first. For example, consider the summary of test results for a recent software release presented in Table 3. Based on the information in the table, the development team can prioritize their efforts and allocate resources accordingly. The critical and high-severity issues should be addressed as soon as possible, while the medium and low-severity issues can be tackled in subsequent sprints or releases.

In addition to categorizing issues, the chatbot can also help identify patterns and trends in the test results, such as:

- Repeated failures in specific functional areas or user flows.
- Correlations between certain test cases or environmental factors.

Table 3 Software release test results summary

Severity	Issue count	Examples	Impact	Priority
Critical	3	Inability to log in, data loss, crashes	Blocks core functionality	High
High	8	Incorrect calculations, missing features	Significantly impacts user experience	High
Medium	15	Layout issues, minor UI bugs	Moderately impacts usability	Medium
Low	22	Typos, minor visual inconsistencies	Minimal impact on overall functionality	Low

- Variations in performance or stability across different platforms or configurations.

By surfacing these insights, the chatbot can assist the development team in understanding the root causes of problems and making more informed decisions about how to improve the software quality.

Furthermore, the chatbot can provide recommendations and suggestions for interpreting the test results, such as:

- Proposing additional test cases or scenarios to validate fixes and regressions.
- Recommending specific debugging techniques or tools to investigate complex issues.
- Advising on the best approach to communicate test results and their implications to stakeholders.

By integrating the chatbot into the test result analysis and interpretation process, organizations can streamline their software quality assurance efforts, make more informed decisions, and ultimately deliver higher-quality software to their users.

8 Chatbot-Based User Acceptance Testing

User Acceptance Testing (UAT) is a critical step in the software development lifecycle, where the final product is evaluated by end-users to ensure that it meets their requirements and expectations. Traditionally, UAT has been a manual and resource-intensive process, often involving extensive coordination between the development team, product managers, and a selected group of users. The chatbot can revolutionize the UAT process by providing a more efficient and scalable approach. By integrating the chatbot into the UAT workflow, organizations can leverage its natural language processing capabilities, knowledge base, and automated testing functionalities to streamline the entire testing process. One of the key advantages of a chatbot-based UAT approach is the ability to engage a larger and more diverse group of users. The chatbot can be made available to a broader audience, allowing for more comprehensive user feedback and a better understanding of how the software performs in real-world scenarios.

For example, consider a financial management software application that has recently undergone a major update. Instead of relying on a small group of pre-selected users to conduct the UAT, the organization decides to incorporate a chatbot-based approach. The chatbot is programmed with a comprehensive set of test scenarios, covering various aspects of the software's functionality. These test scenarios can be categorized and presented in a table, as shown in Table 4. By incorporating these test scenarios into the chatbot's knowledge base, the organization can ensure that users are guided through a comprehensive set of tasks and provide feedback on the software's functionality. The chatbot can then aggregate the user feedback, measure the success rate of each test scenario, and generate detailed reports for the development team to review and address any identified issues.

Table 4 UAT test scenarios

Test scenario	Description	Expected outcome	Actual outcome	Result
Create a new budget	Users should be able to create a new budget, including setting a budget amount and adding expense categories	Budget is created successfully, and expense categories are added	Budget created successfully, but unable to add expense categories	Fail
Track monthly expenses	Users should be able to track their monthly expenses by adding new transactions and categorizing them	Transactions are added and categorized correctly	Transactions added successfully, but some categories are missing	Pass
Generate financial report	Users should be able to generate a financial report that includes their budgets, expenses, and account balances	Financial report is generated with all relevant data	Financial report generated, but some data is missing	Fail
Manage bank accounts	Users should be able to connect their bank accounts, view account balances, and transfer funds between accounts	Bank accounts connected, balances displayed, and funds transferred successfully	Bank accounts connected, but unable to transfer funds	Fail
Set financial goals	Users should be able to set financial goals, such as savings targets or debt reduction plans, and track their progress	Financial goals set and progress tracked accurately	Financial goals set, but progress tracking is not working correctly	Fail

9 Knowledge Base and Support

As part of an effective chatbot-based solution for software testing, a comprehensive knowledge base is essential to provide users with the necessary information and support resources. This knowledge base should be tailored to the specific needs and challenges of software testing, ensuring that the chatbot can deliver accurate and relevant responses to users. The key components of the knowledge base for soft-

Table 5 Knowledge base and support resources for software testing chatbots

Resource	Description	Accessibility
Software Testing FAQs	Frequently asked questions and answers covering a wide range of software testing topics, such as test planning, test case design, test execution, and defect management	Prominently displayed on the knowledge base homepage, with a search function to quickly find relevant FAQs
Test Case Library	A comprehensive library of pre-designed test cases, organized by software module, functionality, and test type (e.g., unit tests, integration tests, end-to-end tests)	Easily searchable and filterable, allowing users to quickly find and reuse relevant test cases
Testing Methodologies and Frameworks	Detailed guides and tutorials on popular software testing methodologies (e.g., Agile, Waterfall, Behavior-Driven Development) and testing frameworks (e.g., Selenium, Cypress, Jest)	Integrated into the knowledge base with clear explanations and examples to help users understand and adopt these approaches
Defect Tracking and Resolution	Information on the organization's defect tracking system, including how to report bugs, view the status of reported issues, and collaborate with the development team on bug fixes	Prominently displayed, with clear instructions on the defect reporting and resolution process
Testing Tool Support	Guides and tutorials on the use of various testing tools and utilities, such as performance testing, load testing, and security testing tools	Organized by tool type and integrated with the knowledge base, allowing users to quickly find the information they need
Support Resources	Contact information for the software testing support team, access to community forums, and any available self-help tools or diagnostic utilities	Easily accessible, with clear instructions on how to engage with the support team or utilize the available resources

ware testing chatbots are outlined in Table 5. This includes a Software Testing FAQs section, a Test Case Library, information on Testing Methodologies and Frameworks, Defect Tracking and Resolution guidance, Testing Tool Support, and relevant Support Resources.

By providing a well-structured and comprehensive knowledge base tailored to software testing, the chatbot-based solution can empower users to find the information they need quickly, troubleshoot issues independently, and leverage best practices and resources for effective software testing. This, in turn, can improve the overall

quality of the software under test, reduce the burden on the testing team, and enhance the user experience for both the testing team and the end-users of the software. Regular reviews and updates to the knowledge base, based on user feedback, support ticket trends, and evolving testing requirements, will ensure that the chatbot maintains its relevance and continues to provide valuable assistance throughout the software testing lifecycle.

10 Collaboration and Communication

Effective collaboration and communication are crucial for the success of any software testing effort. The chatbot-based solution should facilitate seamless interaction and information sharing between various stakeholders, including the testing team, development team, project managers, and end-users. The chatbot should provide features and functionalities to enable efficient team collaboration, such as (See Fig. 5):

- Shared Test Case Library: Allow team members to access, comment on, and contribute to a centralized repository of test cases, facilitating collaboration and reuse. The chatbot could provide a user-friendly interface for browsing, searching, and filtering the test case library, making it easy for team members to find and work with relevant test cases.
- Defect Tracking Integration: Integrate the chatbot with the organization's defect-tracking system, enabling team members to report, track, and discuss issues directly within the chatbot interface. This integration could allow team members to view the status of reported issues, add comments, and receive updates on their resolution.
- Virtual Meetings and Discussions: Provide the ability to initiate or join virtual meetings and discussions within the chatbot, allowing team members to collaborate in real-time. For example, the chatbot could host a virtual meeting where team members can share their screens, annotate test cases, and discuss any issues or con-

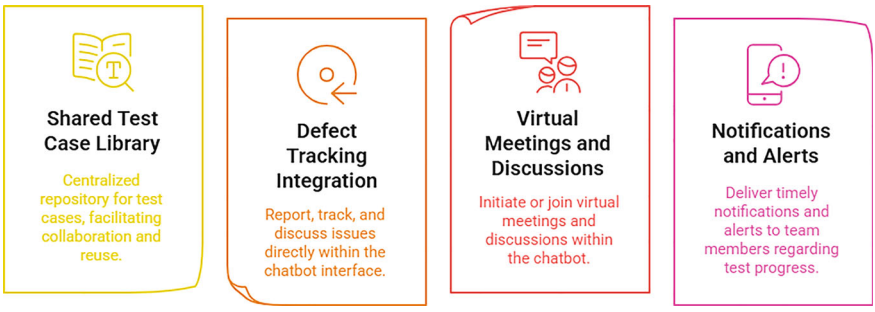


Fig. 5 Chatbot collaboration features

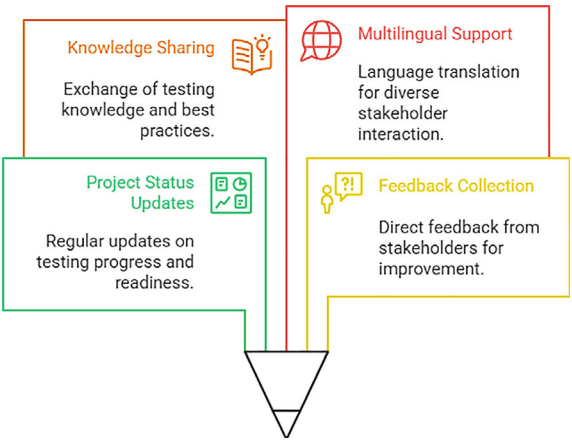


Fig. 6 Enhancing stakeholder engagement

cerns. The chatbot could also provide features like screen sharing, whiteboarding, and live captioning to enhance the virtual meeting experience.

- Notifications and Alerts: Deliver timely notifications and alerts to team members regarding test execution progress, failed test cases, and other critical events, keeping everyone informed. The chatbot could be configured to send push notifications, email alerts, or in-app messages to ensure team members stay up-to-date on the testing process.

The chatbot should also facilitate effective communication with various stakeholders, such as project managers, business analysts, and end-users, by offering features like (See Fig. 6):

- Project Status Updates: Enable the chatbot to provide regular updates on the status of the software testing process, including test coverage, defect trends, and release readiness. The chatbot could generate customized reports and dashboards to give stakeholders a comprehensive view of the testing progress and help them make informed decisions.
- Feedback Collection: Allow stakeholders to provide feedback, suggestions, and concerns directly through the chatbot, ensuring their voices are heard and addressed. The chatbot could guide stakeholders through a structured feedback process, gather their input, and route it to the appropriate teams for review and action.
- Knowledge Sharing: Encourage the sharing of testing-related knowledge, best practices, and lessons learned among stakeholders, fostering a culture of continuous improvement. The chatbot could maintain a knowledge base of testing resources, host Q&A sessions, and facilitate the exchange of ideas and experiences between stakeholders.
- Multilingual Support: Offer multilingual support to accommodate stakeholders from diverse backgrounds and locations, ensuring effective communication. The

chatbot could provide language translation capabilities, allowing stakeholders to interact in their preferred languages and breaking down communication barriers.

By facilitating collaboration and communication, the chatbot-based solution can help to align the testing team with the broader project goals, improve transparency, and foster a collaborative environment that enhances the overall software testing process. The chatbot's ability to connect team members, share information, and engage stakeholders can ultimately lead to better software quality and a more efficient testing workflow.

11 Limitations and Challenges

While the integration of chatbots in software testing holds significant promise, there are several limitations and challenges that need to be considered (See Fig. 7).

1. Accuracy and Reliability of Chatbots One of the primary concerns with using chatbots in software testing is the accuracy and reliability of their responses. Chatbots, even advanced models like ChatGPT, can sometimes provide inaccurate or inconsistent answers, especially when dealing with complex technical details or edge cases. Ensuring the trustworthiness and dependability of chatbot responses is crucial for their effective integration into the software testing process.
2. Contextual Understanding and Domain-Specific Knowledge Chatbots, despite their natural language processing capabilities, may struggle to fully comprehend the context and nuances of software testing tasks. Bridging the gap between the chatbot's general knowledge and the specific domain expertise required for effective software testing can be a significant challenge.
3. Integration with Existing Testing Frameworks and Tools Seamlessly integrating chatbots into the existing software testing infrastructure, such as test management tools, bug tracking systems, and CI/CD pipelines, can be a complex and time-consuming process. Overcoming technical and organizational hurdles to achieve a smooth integration is essential for the successful adoption of chatbots in software testing.
4. Privacy and Security Concerns The use of chatbots in software testing may raise concerns about data privacy and security, especially when dealing with sensitive information related to software systems, test cases, and bug reports. Ensuring the secure handling of data and compliance with relevant regulations is crucial for the widespread adoption of chatbots in the software testing domain.
5. Scalability and Performance Considerations As the volume and complexity of software testing tasks increase, the scalability and performance of chatbot-based solutions need to be carefully evaluated. Ensuring that chatbots can handle high-load scenarios and maintain consistent performance is essential for their long-term viability in software testing.
6. User Acceptance and Adoption The successful integration of chatbots in software testing ultimately depends on the willingness and acceptance of software

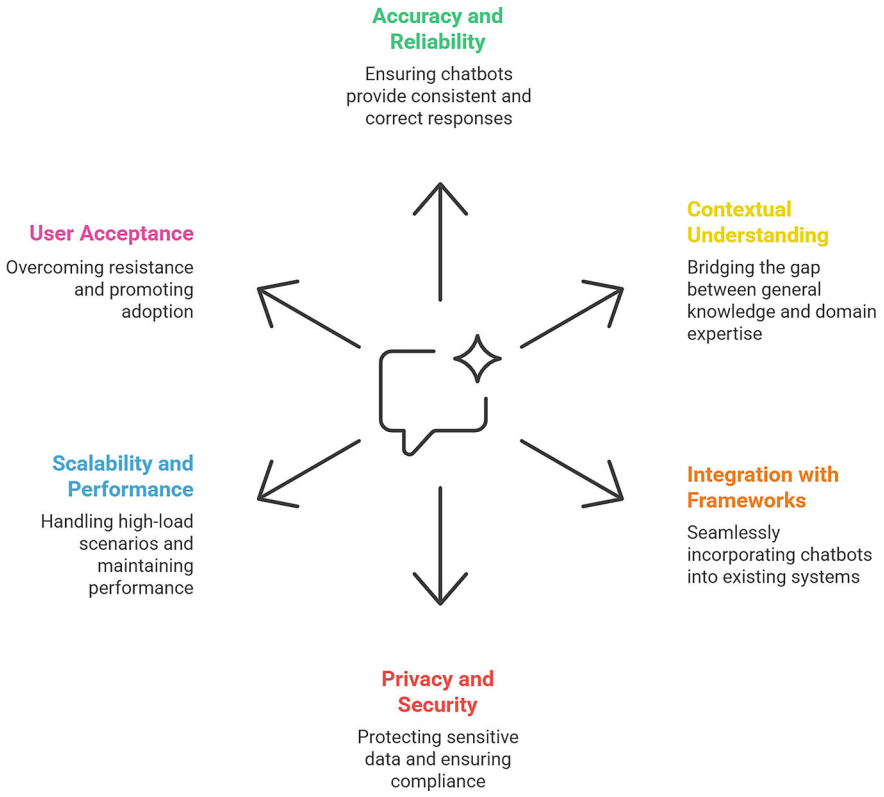


Fig. 7 Challenges in chatbot integration for software testing

testing teams to embrace this new technology. Overcoming resistance to change, providing comprehensive training, and demonstrating the tangible benefits of chatbot-based solutions are crucial for driving user adoption.

Addressing these limitations and challenges will require a collaborative effort between researchers, chatbot developers, software testing experts, and software organizations. By working together to overcome these hurdles, the full potential of chatbots in software testing can be realized, leading to more efficient, effective, and reliable software development processes.

12 Conclusion

The growing complexity of software systems and the evolving challenges faced in software testing have highlighted the pressing need for innovative solutions that enhance the efficiency and effectiveness of the testing process. In recent years, chat-

bots, propelled by advancements in natural language processing (NLP) and dialogue management, have emerged as a promising technology capable of addressing some of these key challenges within the realm of software testing.

In this preliminary work, we have explored a range of applications for chatbots within the software engineering domain, particularly emphasizing their role in the software testing process. While we have proposed simple examples to illustrate their functionalities, we recognize that our exploration is just the beginning of a more extensive investigation. Our initial findings suggest that chatbots can assist in various aspects, including requirements gathering, providing technical support and documentation, enhancing developer productivity, and facilitating collaboration, as well as streamlining project management and task automation.

More significantly, chatbots hold the potential to transform the software testing phase by automating critical tasks such as test case generation, execution, and result analysis. By leveraging the conversational capabilities of chatbots, software teams can improve their testing workflows, reduce manual intervention, and ultimately enhance the overall quality of their software products.

As the field of conversational AI continues to develop, the integration of chatbots in software engineering is expected to become increasingly widespread. Researchers and practitioners alike should actively pursue diverse methodologies for designing, deploying, and integrating chatbots into the software development lifecycle to effectively tackle the pressing challenges in software testing. This includes a deeper examination of how chatbots can be tailored to fit various testing environments and methodologies.

Moreover, to successfully adopt chatbots within software engineering, it is essential to possess a comprehensive understanding of the underlying technologies as well as the development of robust frameworks and methodologies that guide the effective integration of these AI-powered assistants into the software testing process. This will not only help in overcoming existing limitations but also ensure that the integration is both seamless and beneficial.

In conclusion, while our findings illustrate the considerable promise of chatbot integration in software testing for improving quality, efficiency, and scalability, they also underscore the need for additional in-depth studies. Future research should focus on providing concrete examples and empirical evidence, including real case studies, to fully explore the transformative potential of chatbots in enhancing software testing practices. By doing so, we aim to contribute to the delivery of higher-quality software products that align with the diverse and evolving needs of contemporary users, ultimately advancing the field of software engineering.

References

1. Adeyemo, H.B.: Comparative analysis of modeling and software testing tools: a survey. Saudi. J. Eng. Technol. **6**(10), 371–377 (2021)
2. Baliarda, C.P., Borau, C.B., Rodero, M.N., Robert, J.R.: An iterative model for fractal antennas: application to the sierpinski gasket antenna. IEEE Trans. Antennas Propag. **48**(5), 713–719 (2000)
3. Bayrıı, V., Demirel, E.: Ai-powered software testing: The impact of large language models on testing methodologies. In: 2023 4th International Informatics and Software Engineering Conference (IISEC), pp. 1–4 (2023). https://doi.org/10.1109/IISEC59749.2023.10391027
4. Bluemke, I., Malanowska, A.: Software testing effort estimation and related problems: A systematic literature review. ACM Comput. Surv. (CSUR) **54**(3), 1–38 (2021)
5. Boehm, B., Egyed, A., Kwan, J., Port, D., Shah, A., Madachy, R.: Using the winwin spiral model: a case study. Computer **31**(7), 33–44 (1998)
6. Boehm, B.W.: A spiral model of software development and enhancement. Computer **21**(5), 61–72 (1988)
7. Bolotaolo, E.P., Talirongan, F.J.B., Talirongan, H.: Visualizing the academic performance of tertiary student using sashimi model. Irish Interdis. J. Sci. & Res. (IIJSR) (2021)
8. Chan, F.K., Thong, J.Y.: Acceptance of agile methodologies: a critical review and conceptual framework. Decis. Support Syst. **46**(4), 803–814 (2009)
9. Clark, J.O.: System of systems engineering and family of systems engineering from a standards, v-model, and dual-v model perspective. In: 2009 3rd Annual IEEE Systems Conference, pp. 381–387. IEEE (2009)
10. Dauda, I.A., Nuhu, B.K., Abubakar, J., Abdullahi, I.M., Maliki, D.: Software failures: a review of causes and solutions (2021)
11. De Macedo, G.T., Fontão, A.d.L., Gadelha, B.F.: Prototyping in software quality assurance: a survey with software practitioners. In: Proceedings of the XXI Brazilian Symposium on Software Quality, pp. 1–10 (2022)
12. Duflo, M.: Random Iterative Models, vol. 34. Springer Science & Business Media (2013)
13. Fang, Z., Li, J., Liang, A., Bai, G.R., Huang, Y.: A comparative study on chatgpt and checklist as support tools for unit testing education. In: Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering, pp. 871–882 (2025)
14. Haenlein, M., Kaplan, A.: A brief history of artificial intelligence: on the past, present, and future of artificial intelligence. Calif. Manage. Rev. **61**(4), 5–14 (2019)
15. Haldar, S., Pierce, M., Capretz, L.F.: Wip: Assessing the effectiveness of chatgpt in preparatory testing activities. In: 2024 IEEE Frontiers in Education Conference (FIE), pp. 1–5 (2024). https://doi.org/10.1109/FIE61694.2024.10893214
16. Haldar, S., Pierce, M., Fernando Capretz, L.: Exploring the integration of generative ai tools in software testing education: a case study on chatgpt and copilot for preparatory testing artifacts in postgraduate learning. IEEE Access **13**, 46070–46090 (2025). https://doi.org/10.1109/ACCESS.2025.3545882
17. Helvik, B.E., Vizarreta, P., Heegaard, P.E., Trivedi, K., Mas-Machuca, C.: Modelling of software failures. Guide to Disaster-Resilient Communication Networks, pp. 141–172 (2020)
18. Hourani, H., Hammad, A., Lafi, M.: The impact of artificial intelligence on software testing. In: 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), pp. 565–570. IEEE (2019)
19. Ilieva, S., Ivanov, P., Stefanova, E.: Analyses of an agile methodology implementation. In: Proceedings. 30th Euromicro Conference, 2004, pp. 326–333. IEEE (2004)
20. Janßen, C., Richter, C., Wehrheim, H.: Can chatgpt support software verification? In: International Conference on Fundamental Approaches to Software Engineering, pp. 266–279. Springer (2024)
21. Khan, K., Yadav, S.: A literature review on software testing techniques. Optimization of Automated Software Testing Using Meta-Heuristic Techniques, pp. 59–75 (2022)

22. Korzynski, P., Mazurek, G., Altmann, A., Ejdys, J., Kazlauskaite, R., Paliszkiewicz, J., Wach, K., Ziemba, E.: Generative artificial intelligence as a new context for management theories: analysis of chatgpt. Central Europ. Manag. J. (2023)
23. Krichen, M.: How artificial intelligence can revolutionize software testing techniques. In: Innovations in Bio-Inspired Computing and Applications: Proceedings of the 13th International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2022) Held During December 15–17, 2022, pp. 189–198. Springer (2023)
24. Laplante, P.: A brief history of software professionalism and the way forward. Computer **53**(09), 97–100 (2020)
25. Leotta, M., Yousaf, H.Z., Ricca, F., Garcia, B.: Ai-generated test scripts for web e2e testing with chatgpt and copilot: a preliminary study. In: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering, pp. 339–344 (2024)
26. Lima, R., da Cruz, A.M.R., Ribeiro, J.: Artificial intelligence applied to software testing: a literature review. In: 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–6. IEEE (2020)
27. Mathur, S., Malik, S.: Advancements in the v-model. Int. J. Comput. Appl. **1**(12), 29–34 (2010)
28. Okanović, D., Beck, S., Merz, L., Zorn, C., Merino, L., van Hoorn, A., Beck, F.: Can a chatbot support software engineers with load testing? approach and experiences. In: Proceedings of the ACM/SPEC International Conference on Performance Engineering, pp. 120–129 (2020)
29. Peres, R., Schreier, M., Schweidel, D., Sorescu, A.: On chatgpt and beyond: how generative artificial intelligence may affect research, teaching, and practice. Int. J. Res. Mark. (2023)
30. Petersen, K., Wohlin, C., Baca, D.: The waterfall model in large-scale development. In: Product-Focused Software Process Improvement: 10th International Conference, PROFES 2009, Oulu, Finland, June 15–17, 2009. Proceedings 10, pp. 386–400. Springer (2009)
31. Rudianto, R.: The implementation of the rad model in the development of tender selection programs using the ahp method. J. Inf. Telecomm. Eng. **3**(2), 232–239 (2020)
32. Schwartz, S.T., Alfaro, M.E.: Sashimi: a toolkit for facilitating high-throughput organismal image segmentation using deep learning. Methods Ecol. Evol. **12**(12), 2341–2354 (2021)
33. Singhal, P., Kundu, S., Gupta, H., Jain, H.: Application of artificial intelligence in software testing. In: 2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART), pp. 489–492. IEEE (2021)
34. Sun, Z.: A waterfall model for knowledge management and experience management. In: Fourth International Conference on Hybrid Intelligent Systems (HIS'04), pp. 472–475. IEEE (2004)
35. Tang, Y., Liu, Z., Zhou, Z., Luo, X.: Chatgpt vs sbst: a comparative assessment of unit test suite generation. IEEE Trans. Softw. Eng. **50**(6), 1340–1359 (2024). https://doi.org/10.1109/TSE.2024.3382365
36. Tracy, K.W.: Software: A Technical History. Morgan & Claypool (2021)
37. Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., Wang, Q.: Software testing with large language models: survey, landscape, and vision. IEEE Trans. Softw. Eng. **50**(4), 911–936 (2024). https://doi.org/10.1109/TSE.2024.3368208
38. Xia, C.S., Zhang, L.: Automated program repair via conversation: Fixing 162 out of 337 bugs for $0.42 each using chatgpt. In: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, pp. 819–831 (2024)
39. Yi, G., Chen, Z., Chen, Z., Wong, W.E., Chau, N.: Exploring the capability of chatgpt in test generation. In: 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security Companion (QRS-C), pp. 72–80 (2023). https://doi.org/10.1109/QRS-C60940.2023.00013
40. Zhang, C., Lu, Y.: Study on artificial intelligence: the state of the art and future prospects. J. Ind. Inf. Integr. **23**, 100224 (2021)

Moez Krichen is a professor of Computer Science, AlBaha University, Saudi Arabia and a member of the Research Laboratory on Development and Control of Distributed Applications (REDCAD), Sfax. He received a Ph.D. degree in computer science from Joseph Fourier University, France, in 2007, and a HDR (Ability to Conduct Researches) degree in computer science from the University of Sfax, Tunisia, in 2018. His main research interests include model-based conformance, load, and security testing methodologies for real-time, distributed, and dynamically adaptable systems. Moreover, he works on applying formal methods to several modern technologies like smart cities, the Internet of Things (IoT), smart vehicles, drones, and healthcare systems. Currently, he is also working on formal aspects related to deep learning, data mining, blockchain, smart contracts, and optimization.

Geometric Intelligence: A Taxonomic Framework for the Evolution of Generative AI in Meshes and CAD

Abdulrahman S. Al-Batati, Serry Sibaee, Khaled Gabr, and Yasser Alhabashi

Language as a Bridge Between Reasoning and Geometry.

Abstract Generative artificial intelligence (AI) has advanced rapidly from 2D image synthesis to the creation of complex 3D forms, uniting geometric computation with large language model (LLM) reasoning. This chapter presents a taxonomic framework and longitudinal analysis of the scientific foundations and recent developments at the intersection of language and geometry, where LLMs are employed to generate, interpret, and refine three-dimensional structures. It outlines how explicit, implicit, and parametric representations enable the translation of linguistic intent into physical geometry. The discussion traces the technological evolution from early implicit-field models to contemporary multimodal and agentic systems, highlighting their architectures, datasets, and integration strategies. Key challenges, including geometric validity, manufacturability, and ethical responsibility, are examined alongside emerging directions such as real-time co-creative agents and physics-aware generation. Overall, the chapter presents a structured view of how the fusion of language and geometry is reshaping design automation, paving the way toward intelligent, interpretable, and constraint-aware generative design systems.

Keywords Generative AI · Large language models · Text-to-3D generation · Geometric reasoning · Implicit neural representations · CAD · Boundary Representation (B-Rep) · Multimodal learning · Design automation

A. S. Al-Batati (✉) · S. Sibaee · K. Gabr · Y. Alhabashi
Robotics and Internet-of-Things Lab, Prince Sultan University, Riyadh, Saudi Arabia
e-mail: aalbatati@psu.edu.sa

S. Sibaee
e-mail: ssibaee@psu.edu.sa

K. Gabr
e-mail: khammad@psu.edu.sa

Y. Alhabashi
e-mail: yalhabashi@psu.edu.sa

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

A. Koubaa et al. (eds.), *Generative AI and Large Language Models: Opportunities, Challenges, and Applications*, Studies in Computational Intelligence 1287,
https://doi.org/10.1007/978-3-032-24013-2_8

1 Introduction

Generative Artificial Intelligence (AI) has progressed from producing two-dimensional imagery and text to synthesizing complex three-dimensional (3D) content. The convergence of geometry processing, differentiable rendering, and large-scale language modeling is establishing a new paradigm in computational creativity. Modern systems now *design*, *reason*, and *construct* 3D artifacts through text-based interaction. This progression is illustrated in Fig. 1, which summarizes major milestones from early mesh-based learning to language-guided design frameworks. This chapter explores how large language models (LLMs) serve as reasoning engines for 3D generation, bridging data-driven synthesis and rule-based design. It surveys the evolution of text-to-3D and text-to-CAD systems, emphasizing integration challenges and the emerging role of language-guided geometric reasoning in AI-assisted design.

1.1 Motivation and Research Problem

3D generative modeling has evolved from explicit shape synthesis using meshes and voxels to implicit neural representations and language-conditioned generation. As shown in Fig. 1, this evolution traces a clear shift from discrete geometry toward multimodal and structured reasoning systems. Early methods such as autoencoders and GANs learned structured shape distributions but lacked semantic grounding for controllable design [1, 7, 14]. Notably, AtlasNet [16] introduced patch-based surface generation, while implicit approaches like Signed Distance Fields (SDFs) and Neural Radiance Fields (NeRFs) enabled differentiable modeling of geometry and appearance [35, 38], yet remained disconnected from the procedural logic of computer-

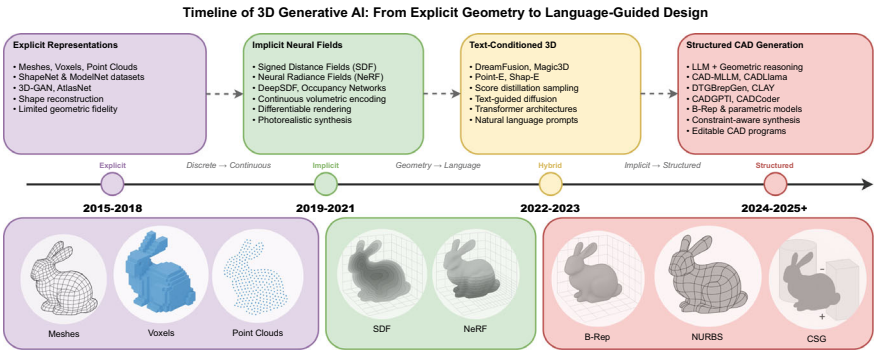


Fig. 1 Timeline of breakthroughs in generative AI across explicit (meshes), implicit (SDFs, NeRFs), and structured (CAD/B-Rep) 3D representations. The trajectory illustrates the convergence of geometric learning and language modeling toward unified 3D design intelligence

aided design (CAD), which relies on parametric and constraint-based modeling [15, 26]. Bridging these paradigms (continuous neural geometry and discrete parametric design) remains an open challenge [3, 58].

LLMs offer a promising path toward this integration. Their capabilities in symbolic reasoning, multimodal understanding, and sequential prediction make them suitable for translating natural-language design intent into geometric or programmatic form [31, 59]. Recent frameworks demonstrate text-to-CAD generation via LLM-guided geometric synthesis [27, 51, 55], though challenges persist in geometric validity and constraint satisfaction [17, 23]. This chapter addresses the need to unify geometric reasoning and natural language understanding to achieve consistent, editable, and physically valid 3D representations guided by linguistic intent.

1.2 Historical Background and Context

The integration of AI and 3D design has evolved through key phases. Between 2015 and 2018, datasets like ShapeNet enabled neural shape synthesis via models such as 3D-GAN and AtlasNet [7, 16]. From 2019 to 2021, implicit neural representations like DeepSDF and NeRF introduced continuous volumetric encoding for differentiable rendering [35, 38]. Since 2022, diffusion-based and transformer-driven systems such as DreamFusion and Magic3D have enabled text-conditioned 3D generation [29, 40]. More recently, hybrid models like CLAY, DTGBrepGen, and CAD-MLLM integrate LLM reasoning with geometric solvers to produce CAD-compatible outputs [28, 55, 60].

1.3 Aim and Scope

This chapter examines how LLMs extend beyond text understanding to spatial and structural reasoning. It articulates a unifying perspective across three traditionally separate domains, explicit geometry, implicit neural fields, and parametric CAD, framing them as complementary rather than competing representations. The discussion emphasizes the distinct yet interdependent roles of discrete surface meshes, which underpin high-fidelity visualization and physics-based simulation, and parametric CAD models, which encode design intent, constraints, and editability. The chapter positions large language models as the connective layer that enables translation, coordination, and hybridization across these representations, driving their convergence into coherent, unified generative design systems. It analyzes three primary geometric representations: (1) **explicit forms** (meshes, voxels, point clouds), (2) **implicit fields** (SDFs, NeRFs), and (3) **structured formats** (B-Rep, CSG, NURBS) [3, 26]. Finally, it explores hybrid paradigms unifying LLM-driven reasoning with geometric solvers, establishing a foundation for intelligent, interpretable, and constraint-aware generative design [23, 51].

2 Geometric Foundations for Generative 3D AI

2.1 Fundamental 3D Data Structures

The foundation of generative modeling for three-dimensional (3D) data lies in the choice of geometric representation. Each representation captures a different trade-off between fidelity, computational efficiency, and differentiability [5, 7, 35, 38]. These representations form the structural basis upon which neural architectures for 3D generation are designed. A concise comparison is provided in Table 1. This organization mirrors the historical and conceptual progression illustrated in Fig. 1, from explicit discrete geometry to implicit neural fields and, more recently, to structured and parametric CAD representations.

To provide a coherent organizational framework, the representations surveyed in this section are organized into three overarching families: explicit geometric representations, implicit neural fields, and parametric or symbolic CAD representations. While these categories are conceptually distinct, several modern approaches occupy hybrid positions at the interfaces between these families, reflecting the increasing convergence of representation paradigms in contemporary generative systems.

2.1.1 Explicit Geometric Representations

Voxel Grids

Voxel grids discretize 3D space into uniform volumetric cells, where each voxel encodes occupancy, color, or signed distance. Although conceptually simple, the cubic memory complexity $\mathcal{O}(N^3)$ limits practical resolution; sparse and octree variants reduce this overhead [50]. Voxel-based representations are therefore primarily employed in early generative models and as intermediate discretizations for implicit or hybrid pipelines.

Point Clouds

A point cloud $\mathcal{P} = \{p_i \in \mathbb{R}^3\}_{i=1}^N$ represents a 3D surface as an unordered set of sample points. Learning methods must account for permutation invariance and often rely on distance-based reconstruction losses such as the Chamfer distance:

$$d_{CH}(\mathcal{P}_1, \mathcal{P}_2) = \sum_{p \in \mathcal{P}_1} \min_{q \in \mathcal{P}_2} \|p - q\|_2^2 + \sum_{q \in \mathcal{P}_2} \min_{p \in \mathcal{P}_1} \|p - q\|_2^2.$$

While efficient, point clouds lack explicit surface topology [1]. This absence of connectivity motivates subsequent reconstruction into meshes or implicit surfaces for tasks requiring watertight geometry or physical simulation.

Surface Meshes

Meshes connect vertices through edges and faces, enabling explicit surface definition with high geometric detail. A triangular mesh $\mathcal{M} = (V, E, F)$ consists of vertex positions V, edges E, and faces F. Mesh-based methods are standard in computer graphics and finite-element analysis, but their irregular topology complicates differentiable learning [9]. Recent approaches such as PolyGen and MeshGPT directly model mesh connectivity sequence-by-sequence using autoregressive Transformers, effectively treating geometry as a discrete language [36, 47]. Hybrid methods such as Deep Marching Tetrahedra (DMTet) bridge implicit and explicit representations by optimizing signed distance fields on a deformable tetrahedral grid, enabling end-to-end differentiable mesh extraction [45]. These developments position explicit mesh generation as a scalable alternative to purely implicit neural fields, particularly in applications requiring surface fidelity and physical validity.

Primitive-Based Explicit Representations

A growing class of explicit representations models geometry using localized primitives that directly encode spatial extent and appearance, enabling efficient rendering and manipulation.

3D Gaussian Splatting (3DGS). 3D Gaussian Splatting (3DGS) is an explicit, primitive-based scene representation that has emerged as a strong alternative to Neural Radiance Fields since 2023 [22]. It represents geometry and appearance as a collection of anisotropic 3D Gaussian primitives that can be efficiently rasterized in screen space.

Each Gaussian primitive is parameterized by a mean position $\boldsymbol{\mu} \in \mathbb{R}^3$, a covariance matrix $\Sigma \in \mathbb{R}^{3\times 3}$ (often factorized into a rotation q and scale s), an opacity α, and view-dependent color modeled using spherical harmonics coefficients $\mathbf{c}_{\ell m}$:

$$G(\mathbf{x}) = \alpha \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

By avoiding volumetric ray marching, 3DGS achieves real-time rendering performance while maintaining high-fidelity appearance and geometry, and its localized parameterization facilitates editing and deformation in interactive workflows.

Surfels and Oriented Surface Primitives. Surfels (surface elements) extend point cloud representations by incorporating local surface orientation and spatial extent [39]. Each surfel is defined by a position $\mathbf{x} \in \mathbb{R}^3$, a normal vector $\mathbf{n}$, and a finite radius r, representing a locally planar surface patch.

Recent work has revisited surfels in neural rendering and splatting, particularly through 2D Gaussian Splatting and nexel-based representations [8, 43]. In contrast to 3D Gaussian primitives, surfel-based methods model geometry as oriented disks in 3D space:

$$S(\mathbf{x}) = \exp\left(-\frac{\|\Pi_{\mathbf{n}}(\mathbf{x} - \boldsymbol{\mu})\|^2}{2\sigma^2}\right),$$

where $\Pi_{\mathbf{n}}$ denotes projection onto the local tangent plane.

Collectively, primitive-based representations bridge unstructured point samples and fully connected surface meshes, offering efficient rasterization and flexible geometric manipulation within explicit pipelines.

2.1.2 Implicit Neural Field Representations

Implicit neural field representations model geometry as continuous functions over $\mathbb{R}^3$, enabling differentiable optimization and resolution-independent surface reconstruction. Geometry is recovered through function evaluation or iso-surface extraction rather than stored explicitly.

Implicit Surfaces

Implicit representations define geometry as the level set of a continuous function $f_\theta : \mathbb{R}^3 \to \mathbb{R}$ parameterized by neural weights θ. The shape is defined as:

$$\mathcal{S} = \{x \in \mathbb{R}^3 \mid f_\theta(x) = 0\}.$$

Signed Distance Functions (SDFs) and Occupancy Networks [33, 38] allow continuous surface definition and gradient-based optimization. These methods capture fine geometric detail but typically require post-processing, such as marching cubes, to obtain explicit meshes suitable for rendering or simulation.

Volumetric Radiance Fields

Volumetric neural fields extend implicit modeling to jointly represent geometry and appearance throughout 3D space.

Neural Radiance Fields (NeRFs). NeRFs represent volumetric scenes as continuous neural fields that map spatial coordinates and viewing direction to color and density [35]. The model learns a function:

$$F_\theta : (\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma),$$

where $\mathbf{x}$ is a 3D position, $\mathbf{d}$ is a viewing direction, $\mathbf{c}$ is emitted color, and σ is volume density. While NeRFs achieve photorealistic novel-view synthesis, they are computationally expensive and poorly suited to direct geometric editing or manufacturing-oriented workflows.

Hybrid and Multi-Resolution Grid Representations. Hybrid grid-based representations introduce structured discretization into implicit neural fields to improve computational efficiency without fully materializing explicit geometry. Tri-plane representations project 3D features onto three orthogonal planes (XY, YZ, XZ) [6, 43], with features aggregated as

$$f(\mathbf{x}) = f_{xy}(x, y) + f_{yz}(y, z) + f_{xz}(x, z).$$

Multi-resolution hash grids further improve efficiency by storing learned features in a hierarchy of spatial hash tables [46], enabling compact representation of high-frequency detail and fast inference.

Together, hybrid grid representations balance the expressive power of implicit neural fields with structured encodings, making them particularly well suited for large-scale and real-time generative and reconstruction systems.

2.1.3 Parametric and Symbolic CAD Representations

Parametric and symbolic CAD representations encode geometry through structured primitives, constraints, and construction logic, enabling exact, editable, and manufacturing-aware models that fundamentally differ from purely geometric encodings.

B-Rep and NURBS

B-Reps and Non-Uniform Rational B-Splines (NURBS) form the foundation of modern CAD modeling [3, 26]. B-Reps define solids by their topological entities (faces, edges, and vertices) along with geometric embeddings. NURBS surfaces, defined by control points P_{ij} and basis functions $N_{i,p}$, enable precise and smooth parametric modeling:

$$S(u, v) = \frac{\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{ij} P_{ij}}{\sum_{i=0}^{n} \sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{ij}}.$$

These representations support exact boundary descriptions, dimensional parameterization, and constraint enforcement, making them central to engineering design and downstream manufacturing workflows.

Constructive Solid Geometry (CSG)

Constructive Solid Geometry (CSG) represents solids through Boolean combinations of primitive shapes. Complex objects are defined by tree-structured programs composed of union, intersection, and difference operations [42]:

$$S = (P_1 \cup P_2) \setminus P_3.$$

CSG provides a symbolic and procedural abstraction of geometry with direct correspondence to manufacturing constraints. Recent learning-based approaches aim to infer CSG programs from meshes or point clouds, enabling recovery of structured and editable CAD representations from unstructured geometric data [44].

Unlike purely geometric representations, symbolic CAD models such as CSG explicitly encode design intent, including construction order and logical operations, making them particularly suitable for parametric design and the generation of manufacturable objects.

2.1.4 Discussion: Representation Trade-Offs and Hybridization

No single representation across explicit geometric, implicit neural, and parametric CAD families simultaneously optimizes fidelity, differentiability, editability, and manufacturability. As a result, modern generative systems increasingly adopt hybrid strategies that combine complementary representations to balance these trade-offs. Together, these representations define the geometric substrate upon which modern text-to-3D and text-to-CAD systems operate. As shown in Fig. 1, recent systems increasingly traverse multiple representation families within a single pipeline, motivating hybrid and language-guided approaches.

2.2 Neural Encoding and Datasets for 3D Geometry and CAD Generation

Building on the geometric representations summarized in Table 1, modern generative systems learn compact latent encodings that capture both topological and semantic characteristics of 3D structures. These neural encoders map a geometric input x to a latent code $z = E_\phi(x)$, from which decoders $D_\psi(z)$ reconstruct or synthesize new 3D instances [1, 9]. Latent spaces serve as continuous shape manifolds that support interpolation, text conditioning, and multimodal alignment across geometry, vision, and language.

Autoencoders and variational autoencoders (VAEs) first demonstrated how latent embeddings could define generative shape spaces, while triplane encodings [41, 57] improved efficiency by projecting 3D features onto three orthogonal 2D planes. While GANs traditionally offered sharper high-fidelity generation, VAEs provided a continuous latent space more suitable for smooth interpolation, a trade-off that modern diffusion models effectively bridge. These compact yet expressive encodings underpin large-scale text-conditioned models such as CLAY [60]. Meanwhile, implicit neural fields such as DeepSDF, Occupancy Networks, and NeRF unify geometry, appearance, and semantics in continuous coordinate mappings, acting as differen-

Table 1 Comparison of 3D representations: advantages, computational trade-offs, key applications, and common file formats

Representation	Type	Advantages	Limitations	Typical uses	File formats
Voxel Grid	Discrete volumetric	Simple, regular structure; supports 3D convolutions	Memory-intensive ($\mathcal{O}(N^3)$); limited spatial resolution	Early 3D CNNs, occupancy learning	BINVOX, HDF5, NPZ
Point Cloud	Discrete sampled surface	Lightweight; easy acquisition from sensors	Lacks explicit topology; unordered and irregular data	LiDAR processing, 3D recognition, point-based generative models	PLY, PCD, LAS, XYZ
Mesh	Explicit surface graph	High geometric fidelity; editable connectivity	Irregular structure; differentiation is difficult	Computer graphics, rendering, physics simulation	OBJ, STL, FBX, glTF
3D Gaussian Splatting	Explicit geometric primitives	Real-time rendering; localized and editable primitives	Large memory footprint; surface ambiguity	Real-time neural rendering, scene editing	PLY (Gaussian params), Custom binary
Surfels/Oriented Primitives	Discrete surface elements	Accurate surface representation; sharper geometry than point clouds	Limited global topology; complex fusion	Surface reconstruction, splatting-based rendering	PLY, Custom surfel formats
Implicit Field (SDF/Occupancy)	Continuous volumetric	Compact, differentiable representation; supports smooth interpolation	Training instability; lacks explicit surfaces	Shape synthesis, 3D reconstruction, neural geometry	NPZ, HDF5, Implicit network weights
NeRF	Volumetric radiance field	Photorealistic, view-consistent rendering; differentiable formulation	High computation cost; limited editability	Novel-view synthesis, neural rendering	Model checkpoints, JSON + weights
Hybrid Multi-Resolution Grids	Hybrid implicit neural encoding	High-frequency detail with low memory; fast training and inference	Implicit structure; limited semantic interpretability	3D-aware generative models, fast neural fields	Hash tables, binary grids, checkpoints
Constructive Solid Geometry (CSG)	Symbolic solid modeling	Encodes design intent; exact and manufacturable geometry	Hard to infer; limited expressiveness for organic shapes	CAD systems, parametric and generative design	CSG scripts, OpenSCAD (.scad)
B-Rep - NURBS	Parametric surface	Exact geometry; supports constraints and parametric design	Complex and non-differentiable; hard to integrate with neural models	CAD modeling, mechanical and engineering design	STEP, IGES, Parasolid (X_T)

tiable priors for pipelines like DreamFusion and Magic3D [29, 35, 38, 40]. Recent frameworks extend this concept through hybrid reasoning, where symbolic and geometric features are jointly optimized. CLAY integrates diffusion-based VAEs for text-geometry co-generation, and DTGBrepGen fuses B-Rep encoders with large language models (LLMs) in a topology-geometry pipeline, learning shared latent manifolds across modalities [28, 55].

Progress in neural encoding has been inseparable from the growth of large-scale 3D and CAD datasets, summarized in Table 2. Foundational repositories such as ShapeNet and ModelNet established the benchmarks for supervised shape reconstruction and classification [5, 7], later extended by multimodal datasets like Objaverse, MVImgNet, and CO3D that align 3D geometry with 2D imagery and textual descriptions [19, 31]. Despite recent progress, the availability of native language supervision remains a fundamental limitation in 3D learning. Classical 3D datasets such as ShapeNet, ModelNet, ScanNet, and ABC were constructed primarily for geometry analysis, reconstruction, and classification, and typically provide only coarse categorical labels rather than free-form natural language descriptions. As a consequence, most text-to-3D methods cannot be trained directly on paired 3D–text data. Instead, they rely on weakly captioned metadata (e.g., object names, tags, or filenames) or leverage pretrained text–image models such as CLIP to bridge language and geometry by supervising 3D generation through rendered 2D views. This paradigm effectively transfers semantic knowledge from large-scale vision–language corpora, bypassing the need for dataset-level textual annotations in 3D. Recent efforts attempt to mitigate this gap by automatically generating captions for 3D assets using vision–language models, although such supervision remains synthetic and substantially noisier than native human-authored descriptions.

However, current datasets often suffer from class imbalance and a lack of engineering metadata, limiting the ability of models to generalize to functional CAD tasks beyond simple furniture categories. Across representations, voxelized and point-cloud datasets (e.g., ModelNet, ShapeNet, ScanNet, SemanticKITTI) primarily support recognition and coarse reconstruction, while mesh and implicit-field datasets (e.g., ShapeNetCore, ABC, DeepSDF) enable higher-fidelity geometry learning but remain largely shape-centric and semantically sparse. NeRF- and Gaussian-based datasets (e.g., DTU, LLFF, Tanks and Temples) further shift emphasis toward view synthesis and photorealism, providing weak or indirect textual supervision rather than explicit design intent. These resources enable diffusion-based 3D reconstruction and text-guided training through rich visual-geometric context. For structured design, datasets like Fusion360, BRepNet, and OmniCAD introduce boundary representations, sketches, and procedural CAD operations [3, 52, 56]. In particular, OmniCAD's 450,000 parametric models with textual metadata have become a benchmark for evaluating text-to-CAD reasoning [61]. In contrast to object-centric shape datasets, these CAD-oriented corpora explicitly encode construction history, constraints, and topology, making them essential for learning editable and functionally valid geometry. Complementary evaluation datasets such as CADPrompt and CADCodeVerify assess the accuracy, completeness, and geometric validity of LLM-

Table 2 Survey of representative 3D and CAD datasets organized by geometric representation, modalities, scale, language supervision, and primary tasks

Dataset	Domain	Inputs	Outputs	Scale	Language supervision	Primary tasks
Voxel representations						
ModelNet10/40 [54]	Synthetic CAD	Meshes	Occupancy voxels	4.9k/12.3k objs	No	Classification, reconstruction
ShapeNet (vox) [7]	Synthetic CAD	Meshes	Occ./TSDF voxels	∼51k objs	No	Reconstruction, generation
ScanNet [11]	Indoor scenes	RGB-D, poses	TSDF volumes	1.5k scenes	No	Scene understanding, mapping
Point clouds						
ShapeNet (pc)	Synthetic CAD	Meshes	Point clouds	∼51k objs	No	Recognition, generation
ModelNet (pc)	Synthetic CAD	Meshes	Point clouds	12.3k objs	No	Classification
SemanticKITTI [2]	Autonomous driving	LiDAR, RGB	Labeled points	∼43k scans	No	Semantic segmentation
Waymo Open [49]	Autonomous driving	LiDAR, RGB	Labeled points	∼230k scenes	No	Detection, tracking
Meshes						
ShapeNetCore	Synthetic CAD	CAD models	Meshes	∼51k objs	No	Shape modeling, generation
ABC [25]	Mechanical CAD	B-Rep CAD	Meshes + surfaces	∼1M models	No	Surface reconstruction
FAUST/MPI-FAUST [4]	Human body	3D scans	Registered meshes	300/400	No	Deformation, correspondence
Thingi10K [62]	3D printing	User meshes	Cleaned meshes	10k models	No	Mesh repair, analysis
Implicit fields						
DeepSDF [38]	Synthetic CAD	Meshes	SDF samples	∼20k shapes	No	Implicit shape learning
OccupancyNet [33]	Synthetic CAD	Meshes	Occ. samples	∼13k shapes	No	Surface reconstruction
NeRF-based						
DTU [18]	Object-centric	Multi-view RGB	Meshes/NeRF	124 scenes	Partial	View synthesis
LLFF [34]	Real scenes	RGB, poses	Radiance fields	8 scenes	Partial	Novel-view synthesis
Gaussian/Multi-plane						
Tanks and Temples [24]	Large scenes	Multi-view RGB	Gaussian splats	14 scenes	Partial	Photorealistic rendering
Objaverse/Objaverse-XL [12, 13]	Internet-scale	Meshes, textures	Renders, latents	∼800k objs	Partial	Text-to-3D diffusion
CAD/Programmatic						
Fusion 360 Gallery [52]	Mechanical design	Parametric CAD	B-Rep solids	∼8k designs	No	Parametric CAD generation
DeepCAD [53]	Programmatic CAD	CAD programs	Sketch/extrusion seq.	∼33k	No	Program synthesis
CSGNet [44]	Synthetic shapes	CSG programs	Rendered geometry	∼200k	No	Symbolic reasoning
ShapeAssembly [20]	Part-based CAD	Assembly programs	Hierarchical shapes	∼20k	No	Part-based generation

generated design scripts [23, 32], establishing quantitative metrics for language-geometry translation.

Together, these advances in latent encoding and large-scale multimodal datasets form the foundation for current text-to-3D and text-to-CAD research. They bridge low-level geometric learning with high-level semantic reasoning, enabling the emergence of a Unified Multimodal Foundation Model (Fig. 3) that integrates symbolic language, diverse geometric representations, and physics priors to solve the long-standing challenge of editability and functional validity in generative design.

3 Methodological Taxonomy of Language-Geometry Integration

3.1 Frameworks and Integration Paradigms

Recent advances in 3D generative AI demonstrate a clear taxonomy of methods through which language models interact with geometric reasoning. These systems differ in how they couple linguistic understanding, spatial representation, and procedural synthesis. Four primary integration paradigms can be distinguished [27, 31, 55, 59]. The conceptual differences among these pipelines are summarized in Fig. 2, contrasting 2D diffusion, 3D shape synthesis, and parametric CAD program generation.

3.1.1 Programmatic Generation

Programmatic frameworks translate text directly into executable design programs. Systems such as CAD-Llama, CAD-Coder, and Text2CAD use LLMs to parse natural-language prompts into CAD command sequences or parametric scripts [17, 23, 27]. Figure 2 contextualizes this paradigm within a broader generative pipeline, highlighting how language understanding is extended from pixel-based synthesis to fully parametric command execution. It is essential to note that these paradigms are not mutually exclusive; recent systems frequently hybridize these approaches, for instance, by initializing with implicit fields before optimizing for explicit mesh topology. The resulting code can be rendered or re-executed within CAD environments, enabling reversible, editable design workflows.

3.1.2 Spatial Reasoning and Layout Understanding

Spatially grounded systems emphasize the comprehension of geometric context and relational layout. Frameworks such as LI3D, AnyHome, and LL3M employ multimodal reasoning to interpret spatial configurations and scene semantics [30, 31].

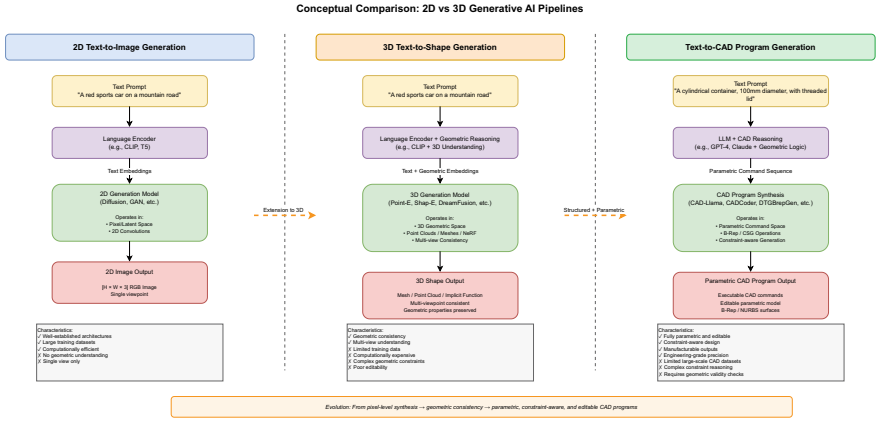


Fig. 2 Conceptual comparison between text-to-image diffusion, text-to-mesh synthesis, and text-to-CAD program generation. Text-to-image models map linguistic prompts to 2D pixels through diffusion priors; text-to-mesh methods extend this to 3D surface generation; text-to-CAD approaches further translate text into structured, parametric command sequences executable in CAD systems

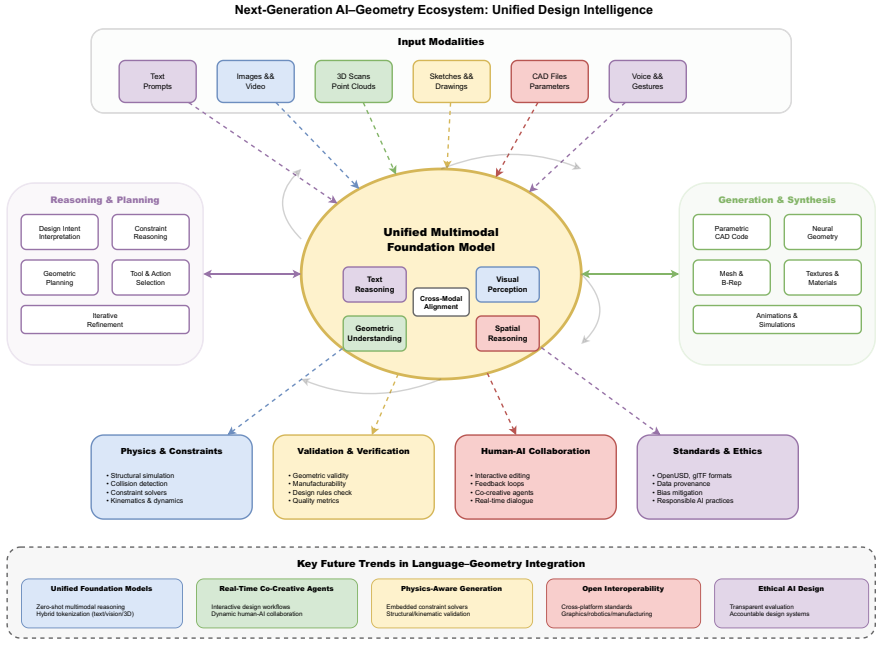


Fig. 3 Next-generation AI-geometry ecosystem integrating multimodal reasoning and physics-based simulation for real-time co-design

They combine geometric encoders (e.g., point or voxel features) with language-conditioned decoders to perform scene reconstruction, affordance prediction, or 3D question answering. These models are often pre-trained using spatially aligned text-geometry pairs (e.g., ScanNet, CO3D) and fine-tuned with spatial reasoning objectives.

3.1.3 Multimodal Alignment

A central theme in recent work is the alignment of linguistic, visual, and geometric embeddings within a shared latent space [41, 57, 60]. Systems such as CAD-MLLM, CAD-GPT, and ShapeGPT adopt dual-encoder or cross-attention architectures to align language and geometry representations. This alignment allows zero-shot shape editing and natural-language-driven model refinement. These methods often employ contrastive or diffusion-based objectives to enforce consistency between textual semantics and geometric features.

3.1.4 Agentic and Feedback-Driven Systems

A new direction in language-geometry integration focuses on autonomous agents capable of the "Design-Reason-Construct" loop, making a paradigm shift where systems actively construct artifacts via tool use rather than passive generation. Frameworks such as Query2CAD, 3DFroMLLM, and CAD-Assistant employ LLM planning with iterative environment feedback to ensure geometric correctness [32, 48]. Unlike static text-to-mesh pipelines, these agents utilize tool-use mechanics (e.g., calling constraint solvers or physics simulators) to validate intermediate outputs. This feedback loop allows the system to detect and correct non-manifold geometry or unmanufacturable overhangs, effectively closing the gap between probabilistic generation and deterministic engineering constraints.

3.2 Technological Milestones (2018–2025)

The convergence of generative modeling and language-driven geometry reasoning has progressed through several key technological milestones, each expanding the representational scope and multimodal capability of 3D AI systems [19, 50, 59]. A chronological summary of these developments, including representative frameworks, representations, and their core contributions, is presented in Table 3.

Between 2018 and 2019, early breakthroughs such as AtlasNet, DeepSDF, and Occupancy Networks demonstrated the feasibility of neural 3D shape generation through latent surface learning and implicit representations [16, 33, 38]. These methods established differentiable priors that laid the foundation for later text-conditioned synthesis. The period from 2020 to 2021 marked the introduction of differentiable

Table 3 Major breakthroughs in generative AI for 3D creation (2018–2025), highlighting frameworks, representations, and contributions

Framework	Year	Representation	Contribution	Reference
AtlasNet	2018	Mesh (explicit)	Learned surface parameterization using latent templates	[16]
DeepSDF	2019	Implicit SDF	Continuous signed distance field representation for high-fidelity shape learning	[38]
Occupancy Networks	2019	Implicit occupancy	Probabilistic occupancy prediction and differentiable surface extraction	[33]
NeRF/EG3D/DeepCAD	2020–2021	Neural fields/CAD	Differentiable rendering and CAD topology learning	[15, 26, 35]
DreamFusion/Magic3D	2022	Diffusion/SDF hybrid	Text-to-3D via score distillation and diffusion priors	[29, 40]
Point-E/Shap-E/ProlificDreamer	2023	Point/diffusion	Lightweight diffusion models for 3D generation from text	[21, 37, 50]
CLAY/CAD-MLLM/DTGBrepGen/Turbo3D	2024–2025	Multimodal CAD	LLM-integrated geometric reasoning and real-time 3D co-generation	[28, 48, 55, 60]

rendering and structured modeling, with NeRF enabling continuous volumetric rendering and frameworks like EG3D and DeepCAD bridging the gap between graphics-based neural fields and CAD-oriented topology learning [15, 26, 35].

From 2022 onward, text-conditioned diffusion models revolutionized 3D generation. DreamFusion and Magic3D applied score distillation and diffusion priors to generate textured 3D assets directly from natural-language prompts [29, 40], inaugurating the first multimodal text-to-3D pipelines. By 2023, systems such as Point-E, Shap-E, and ProlificDreamer improved geometric quality and efficiency through lightweight diffusion networks, volumetric distillation, and view-consistency constraints [21, 37, 50].

The most recent phase, spanning 2024–2025, has seen the emergence of multimodal and reasoning-driven systems that integrate large language models with geometric solvers and CAD logic. Frameworks such as CLAY, CAD-MLLM, DTGBrepGen, and Turbo3D unify symbolic reasoning, geometric computation, and real-time interaction [28, 48, 55, 60]. Collectively, these milestones (summarized in Table 3) trace the transformation of generative AI from surface-based modeling to intelligent, language-guided, and physically aware 3D co-design systems.

To achieve true physical grounding, future systems must integrate dynamic validation. Beyond static shape optimization, generated mechanisms should be simulated under load constraints to verify kinematic viability, ensuring that "co-creative agents" provide real-time functional feedback, not just visual suggestions.

4 Challenges, Trends, and Future Directions in Language-Geometry Integration

Integrating large language models (LLMs) with 3D geometric reasoning presents vast opportunities for intelligent design automation but also major challenges in validity, scalability, and responsibility. Current text-to-3D and text-to-CAD systems often struggle with geometric and physical consistency, producing shapes that lack manufacturable topology or constraint satisfaction [28, 55]. Diffusion-based methods achieve high visual realism but rarely yield editable or watertight geometry suitable for CAD applications [60]. Moreover, the training of multimodal 3D-language models demands extensive data and computation, often relying on synthetic or unverified sources. This raises issues of data provenance, bias, and intellectual property [10]. Finally, linguistic ambiguity remains a persistent obstacle, underspecified prompts frequently lead to unstable or invalid geometric outputs, underscoring the need for better grounding and feedback mechanisms [31, 59].

Furthermore, specific technical hurdles remain. Mode collapse in diffusion models can lead to repetitive or generic shapes, limiting the ability to capture the long-tail diversity characteristic of engineering components. Texture–geometry consistency presents another bottleneck, where generated appearance fails to align with underlying mesh topology, resulting in visual artifacts or physically implausible surfaces. Crucially, constraint satisfaction failures are common in language-driven design pipelines: large language models may produce syntactically valid CAD programs that violate geometric, topological, or manufacturability constraints.

This exposes a central open challenge in generative design: reconciling continuous neural representations, which are differentiable but often lack guarantees of validity, with discrete parametric CAD representations, which are exact and interpretable but difficult to integrate into gradient-based learning. Addressing this gap requires rigorous, task-specific evaluation. Recent benchmarks such as CADPrompt and CADCodeVerify [23, 32] begin to formalize geometric and program-level correctness, yet their adoption remains limited compared to perceptual metrics such as FID that dominate visual generative modeling.

Practical Considerations

From a practitioner's perspective, computational cost is often a decisive factor in selecting a generative approach. Implicit neural field methods, including NeRF-based pipelines and diffusion-driven mesh generation, typically require substantial GPU resources and long inference times, ranging from minutes to hours per instance, due to iterative optimization and high-resolution sampling. In contrast, programmatic CAD generation via large language models can be near-instantaneous at inference time, but depends on robust downstream geometric kernels, constraint solvers, and post-processing steps to ensure validity and manufacturability. Memory overhead

further constrains scalability: voxel grids and high-resolution meshes scale cubically or superlinearly with resolution, limiting end-to-end deployment on consumer-grade hardware.

Emerging trends suggest a transition toward unified, physics-aware, and ethically governed generative systems. Future models are expected to integrate text, vision, and geometry through shared multimodal tokenization, enabling coherent cross-domain reasoning [59]. Interactive co-creation frameworks such as CAD-MLLM and Turbo3D demonstrate how human-AI collaboration can support iterative, real-time design refinement [48]. Embedding constraint and physics solvers within generative pipelines promises physically valid and functionally grounded designs, while open standards like OpenUSD and glTF foster interoperability and reproducibility [3]. As AI becomes a partner in creative reasoning, transparency, authorship, and accountability must remain central to the evolution of responsible, language-driven 3D design intelligence [10]. Ethical transparency requires more than acknowledgment; it necessitates concrete mechanisms such as "model cards" for generative CAD systems, detailing training data sources (e.g., distinguishing between open-source and proprietary CAD repositories) and licensing limitations. We propose that future benchmarks explicitly evaluate "attribution scores", measuring how much a generated design copies from training set examples, to mitigate copyright risks in industrial applications. Figure 3 visualizes this envisioned ecosystem, illustrating how multimodal reasoning, physics-aware synthesis, and ethical governance converge in next-generation co-design frameworks.

5 Conclusion

The integration of language models with geometric reasoning represents a major leap in generative AI, evolving from data-driven 3D reconstruction to intelligent, multimodal design systems. Modern frameworks such as CLAY, CAD-MLLM, and DTGBrepGen enable joint reasoning across text, vision, and geometry, transforming shape generation into interactive human-AI co-creation [31, 60]. Future research will center on three directions: unified multimodal foundation models, physics- and constraint-aware generation for manufacturable outputs, and ethical transparency in AI-assisted creativity [3, 10]. As summarized conceptually in Fig. 3, the unification of text, geometry, and reasoning defines the foundation of future generative design intelligence. Together, these advances point toward a new era of computational creativity, where reasoning and geometry converge to create expressive, reliable, and collaborative design intelligence.

Acknowledgements This research was supported by Prince Sultan University (PSU). The authors would like to express their sincere gratitude to the Research and Initiatives Center at PSU for their technical and institutional support. The resources and collaborative environment provided by the university were instrumental in enabling the conceptualization and development of this work.
The authors also acknowledge the use of large language modelâŁ"based artificial intelligence tools during the preparation of this chapter to assist with language refinement, clarity, and editorial orga-

nization of the text. All scientific content, interpretations, technical descriptions, and conclusions were conceived, written, reviewed, and verified by the authors. The AI tools were not used to generate original research content, data, or figures, and no AI system qualifies for authorship. The authors have exercised due diligence to ensure the accuracy, originality, and proper citation of all material, and confirm that the manuscript is free from plagiarism and copyright infringement.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning Representations and Generative Models for 3D Point Clouds (2018). https://doi.org/10.48550/arXiv.1707.02392
2. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: a Dataset for Semantic Scene Understanding of LiDAR Sequences. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9296–9306 (2019).https://doi.org/10.1109/ICCV.2019.00939
3. Berger, E., Dammann, M.P., Mehlstäubl, J., Saske, B., Braun, F., Paetzold-Byhain, K.: Challenges and opportunities in the integration of generative AI with computer-aided design. Proc. Design Soc. **5**, 881–890 (2025). https://doi.org/10.1017/pds.2025.10102
4. Bogo, F., Romero, J., Loper, M., Black, M.J.: FAUST: Dataset and Evaluation for 3D Mesh Registration. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3794–3801 (2014). https://doi.org/10.1109/CVPR.2014.491
5. Caytuiro, N., Sipiran, I.: 3D Shape Generation: A Survey (2025). https://doi.org/10.48550/arXiv.2506.22678
6. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., De Mello, S., Gallo, O., Guibas, L.J., Tremblay, J., Khamis, S., et al.: Efficient geometry-aware 3d generative adversarial networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16123–16133 (2022)
7. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository (2015).https://doi.org/10.48550/arXiv.1512.03012
8. Chen, D., Li, H., Ye, W., Wang, Y., Xie, W., Zhai, S., Wang, N., Liu, H., Bao, H., Zhang, G.: Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. IEEE Trans. Visual. Comput. Graph. (2024)
9. Chen, S., Chen, X., Pang, A., Zeng, X., Cheng, W., Fu, Y., Yin, F., Wang, Y., Wang, Z., Zhang, C., Yu, J., Yu, G., Fu, B., Chen, T.: MeshXL: Neural Coordinate Field for Generative 3D Foundation Models (2024). https://doi.org/10.48550/arXiv.2405.20853
10. Daareyni, A., Martikkala, A., Mokhtarian, H., Ituarte, I.F.: Generative AI meets CAD: enhancing engineering design to manufacturing processes with large language models. Int. J. Adv. Manuf. Technol. (2025). https://doi.org/10.1007/s00170-025-15830-2. Jun
11. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: ScanNet: richly-annotated 3D reconstructions of indoor scenes. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2432–2443 (2017). https://doi.org/10.1109/CVPR.2017.261
12. Deitke, M., Liu, R., Wallingford, M., Ngo, H., Michel, O., Kusupati, A., Fan, A., Laforte, C., Voleti, V., Gadre, S.Y., VanderBilt, E., Kembhavi, A., Vondrick, C., Gkioxari, G., Ehsani, K., Schmidt, L., Farhadi, A.: Objaverse-XL: A Universe of 10M+ 3D Objects (2023). https://doi.org/10.48550/arXiv.2307.05663
13. Deitke, M., Schwenk, D., Salvador, J., Weihs, L., Michel, O., VanderBilt, E., Schmidt, L., Ehsani, K., Kembhavi, A., Farhadi, A.: Objaverse: A Universe of Annotated 3D Objects (2022). https://doi.org/10.48550/arXiv.2212.08051

14. Feng, Y., Feng, Y., You, H., Zhao, X., Gao, Y.: MeshNet: Mesh Neural Network for 3D Shape Representation (2018). https://doi.org/10.48550/arXiv.1811.11424
15. Ganin, Y., Bartunov, S., Li, Y., Keller, E., Saliceti, S.: Computer-Aided Design as Language (2021). https://doi.org/10.48550/arXiv.2105.02769
16. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A Papier-Mache Approach to Learning 3D Surface Generation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 216–224 (2018). https://doi.org/10.1109/CVPR.2018.00030
17. Guan, Y., Wang, X., Xing, X., Zhang, J., Xu, D., Yu, Q.: CAD-Coder: Text-to-CAD generation with chain-of-thought and geometric reward (2025). https://doi.org/10.48550/arXiv.2505.19713
18. Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., Aanæs, H.: Large scale multi-view stereopsis evaluation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 406–413 (2014). https://doi.org/10.1109/CVPR.2014.59
19. Jiang, C.: A Survey on Text-to-3D Contents Generation In The Wild (2024). https://doi.org/10.48550/arXiv.2405.09431
20. Jones, R.K., Barton, T., Xu, X., Wang, K., Jiang, E., Guerrero, P., Mitra, N.J., Ritchie, D.: ShapeAssembly: Learning to generate programs for 3D shape structure synthesis. ACM Trans. Graph. **39**(6), 234:1–234:20 (2020). https://doi.org/10.1145/3414685.3417812
21. Jun, H., Nichol, A.: Shap-E: Generating Conditional 3D Implicit Functions (2023). https://doi.org/10.48550/arXiv.2305.02463
22. Kerbl, B., Kopanas, G., Leimkuehler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4) (2023). https://doi.org/10.1145/3592433
23. Khan, M.S., Sinha, S., Sheikh, T.U., Stricker, D., Ali, S.A., Afzal, M.Z.: Text2CAD: Generating Sequential CAD Models from Beginner-to-Expert Level Text Prompts (2024). https://doi.org/10.48550/arXiv.2409.17106
24. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Trans. Graph. **36**(4), 78:1–78:13 (2017). https://doi.org/10.1145/3072959.3073599
25. Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., Panozzo, D.: ABC: A Big CAD Model Dataset For Geometric Deep Learning (2019). https://doi.org/10.48550/arXiv.1812.06216
26. Lambourne, J.G., Willis, K.D.D., Jayaraman, P.K., Sanghi, A., Meltzer, P., Shayani, H.: BRepNet: A topological message passing system for solid models (2021). https://doi.org/10.48550/arXiv.2104.00706
27. Li, J., Ma, W., Li, X., Lou, Y., Zhou, G., Zhou, X.: CAD-Llama: Leveraging Large Language Models for Computer-Aided Design Parametric 3D Model Generation. In: 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 18563–18573 (2025). https://doi.org/10.1109/CVPR52734.2025.01730
28. Li, J., Fu, Y., Chen, F.: DTGBrepGen: A Novel B-rep Generative Model through Decoupling Topology and Geometry (2025). https://doi.org/10.48550/arXiv.2503.13110
29. Lin, C.H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.Y., Lin, T.Y.: Magic3D: High-Resolution Text-to-3D Content Creation (2023). https://doi.org/10.48550/arXiv.2211.10440
30. Lu, S., Chen, G., Dinh, N.A., Lang, I., Holtzman, A., Hanocka, R.: LL3M: Large Language 3D Modelers (2025). https://doi.org/10.48550/arXiv.2508.08228
31. Ma, X., Smart, B., Bhalgat, Y., Chen, S., Li, X., Ding, J., Gu, J., Chen, D.Z., Peng, S., Bian, J.W., Torr, P.H., Pollefeys, M., Nießner, M., Reid, I.D., Chang, A.X., Laina, I., Prisacariu, V.A.: When LLMs step into the 3D World: A Survey and Meta-Analysis of 3D Tasks via Multi-modal Large Language Models (2025). https://doi.org/10.48550/arXiv.2405.10255
32. Mallis, D., Karadeniz, A.S., Cavada, S., Rukhovich, D., Foteinopoulou, N., Cherenkova, K., Kacem, A., Aouada, D.: CAD-Assistant: Tool-Augmented VLLMs as Generic CAD Task Solvers (2025). https://doi.org/10.48550/arXiv.2412.13810

33. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: learning 3D reconstruction in function Space. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4455–4465 (2019). https://doi.org/10.1109/CVPR.2019.00459
34. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: practical view synthesis with prescriptive sampling guidelines. ACM Trans. Graph. **38**(4), 29:1–29:14 (2019). https://doi.org/10.1145/3306346.3322980
35. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis (2020). https://doi.org/10.48550/arXiv.2003.08934
36. Nash, C., Ganin, Y., Eslami, S.M.A., Battaglia, P.: PolyGen: An Autoregressive Generative Model of 3D Meshes. In: Proceedings of the 37th International Conference on Machine Learning, pp. 7220–7229 (2020)
37. Nichol, A., Jun, H., Dhariwal, P., Mishkin, P., Chen, M.: Point-E: A System for Generating 3D Point Clouds from Complex Prompts (2022). https://doi.org/10.48550/arXiv.2212.08751
38. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: DeepSDF: learning continuous signed distance functions for shape representation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 165–174 (2019). https://doi.org/10.1109/CVPR.2019.00025
39. Pfister, H., Zwicker, M., Van Baar, J., Gross, M.: Surfels: surface elements as rendering primitives. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 335–342 (2000)
40. Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: DreamFusion: Text-to-3D using 2D Diffusion (2022). https://doi.org/10.48550/arXiv.2209.14988
41. Qi, Z., Fang, Y., Sun, Z., Wu, X., Wu, T., Wang, J., Lin, D., Zhao, H.: GPT4Point: a unified framework for point-language understanding and generation. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 26407–26417 (2024). https://doi.org/10.1109/CVPR52733.2024.02495
42. Requicha, A.G.: Representations for rigid solids: theory, methods, and systems. ACM Comput. Surv. (CSUR) **12**(4), 437–464 (1980)
43. Rong, V., Held, J., Chu, V., Rebain, D., Droogenbroeck, M.V., Kutulakos, K.N., Tagliasacchi, A., Lindell, D.B.: Nexels: Neurally-Textured Surfels for Real-Time Novel View Synthesis with Sparse Geometries (2025). https://doi.org/10.48550/arXiv.2512.13796
44. Sharma, G., Goyal, R., Liu, D., Kalogerakis, E., Maji, S.: Csgnet: Neural shape parser for constructive solid geometry. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5515–5523 (2018)
45. Shen, T., Gao, J., Yin, K., Liu, M.Y., Fidler, S.: Deep marching tetrahedra: A hybrid representation for high-resolution 3D shape synthesis. In: Proceedings of the 35th International Conference on Neural Information Processing Systems, pp. 6087–6101. NIPS '21, Curran Associates Inc., Red Hook, NY, USA (2021)
46. Shue, J.R., Chan, E.R., Po, R., Ankner, Z., Wu, J., Wetzstein, G.: 3d neural field generation using triplane diffusion (2022). arxiv:2211.16677
47. Siddiqui, Y., Alliegro, A., Artemov, A., Tommasi, T., Sirigatti, D., Rosov, V., Dai, A., Nießner, M.: MeshGPT: generating triangle meshes with decoder-only transformers. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 19615–19625 (2024). https://doi.org/10.1109/CVPR52733.2024.01855
48. Sun, C., Han, J., Deng, W., Wang, X., Qin, Z., Gould, S.: 3D-GPT: procedural 3D modeling with large language models. In: 2025 International Conference on 3D Vision (3DV), pp. 1253–1263 (2025). https://doi.org/10.1109/3dv66043.2025.00119
49. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D.: Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2443–2451 (2020). https://doi.org/10.1109/CVPR42600.2020.00252

50. Wang, C., Peng, H.Y., Liu, Y.T., Gu, J., Hu, S.M.: Diffusion models for 3D generation: a survey. Comput. Vis. Media **11**(1), 1–28 (2025). https://doi.org/10.26599/CVM.2025.9450452
51. Wang, S., Chen, C., Le, X., Xu, Q., Xu, L., Zhang, Y., Yang, J.: CAD-GPT: Synthesising CAD Construction Sequence with Spatial Reasoning-Enhanced Multimodal LLMs. Proc. AAAI Confer. Artif. Intell. **39**(8), 7880–7888 (2025). https://doi.org/10.1609/aaai.v39i8.32849
52. Willis, K.D.D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J.G., Solar-Lezama, A., Matusik, W.: Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences (2021). https://doi.org/10.48550/arXiv.2010.02392
53. Wu, R., Xiao, C., Zheng, C.: DeepCAD: A Deep Generative Network for Computer-Aided Design Models (2021). https://doi.org/10.48550/arXiv.2105.09492
54. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1912–1920 (2015). https://doi.org/10.1109/CVPR.2015.7298801
55. Xu, J., Wang, C., Zhao, Z., Liu, W., Ma, Y., Gao, S.: CAD-MLLM: Unifying Multimodality-Conditioned CAD Generation With MLLM (2025). https://doi.org/10.48550/arXiv.2411.04954
56. Xu, X., Lambourne, J., Jayaraman, P., Wang, Z., Willis, K., Furukawa, Y.: BrepGen: A B-rep Generative Diffusion Model with Structured Latent Geometry. ACM Transactions on Graphics **43**(4), 1–14 (2024). https://doi.org/10.1145/3658129. Jul
57. Yin, F., Chen, X., Zhang, C., Jiang, B., Zhao, Z., Liu, W., Yu, G., Chen, T.: ShapeGPT: 3D Shape Generation With a Unified Multi-Modal Language Model. IEEE Trans. Multimedia **27**, 4107–4120 (2025). https://doi.org/10.1109/TMM.2025.3535389
58. Yuan, Z., Shi, J., Huang, Y.: OpenECAD: An efficient visual language model for editable 3D-CAD design. Computers & Graphics **124**, 104048 (Nov 2024https://doi.org/10.1016/j.cag.2024.104048
59. Zhang, L., Le, B., Akhtar, N., Lam, S.K., Ngo, T.: Large Language Models for Computer-Aided Design: A Survey (May 2025).https://doi.org/10.48550/arXiv.2505.08137
60. Zhang, L., Wang, Z., Zhang, Q., Qiu, Q., Pang, A., Jiang, H., Yang, W., Xu, L., Yu, J.: CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets (May 2024).https://doi.org/10.48550/arXiv.2406.13897
61. Zhang, S., Guan, Z., Jiang, H., Ning, T., Wang, X., Tan, P.: Brep2Seq: A dataset and hierarchical deep learning network for reconstruction and generation of computer-aided design models. Journal of Computational Design and Engineering **11**(1), 110–134 (2024). https://doi.org/10.1093/jcde/qwae005. Feb
62. Zhou, Q., Jacobson, A.: Thingi10K: A Dataset of 10,000 3D-Printing Models (Jul 2016https://doi.org/10.48550/arXiv.1605.04797

Abdulrahman S. Al-Batati is a Senior Research Assistant at the Robotics and Internet-of-Things Lab (RIOTU), Prince Sultan University, Saudi Arabia. He holds a Master's degree in Robotics and a Bachelor's degree in Mechanical Engineering. His research interests include robotic system design, unmanned aerial vehicles (UAVs), computer-aided design (CAD), additive manufacturing, and reinforcement learning. He has authored more than ten peer-reviewed publications and holds one granted patent.

Serry Sibaee is a Research Engineer at the Robotics and Internet-of-Things Lab (RIOTU), Prince Sultan University, Saudi Arabia. He holds a Bachelor's degree in Software Engineering from Prince Sultan University with second-class honors. His research interests focus on natural language processing, large language models, and multimodal AI systems, with particular emphasis on Arabic NLP. He has contributed to multiple peer-reviewed publications in ACL, Springer, MDPI,

and related venues, and has worked on projects involving Arabic language modeling, retrieval-augmented generation, and dataset construction.

Khaled Gabr is a Software Robotics Engineer at Prince Sultan University, Saudi Arabia. He holds a degree in Mechatronics Engineering from the Higher Technology Institute, Egypt, and has over ten years of professional experience in robotics and autonomous systems. His career includes contributions to drone and autonomous robot development at Systemtrio, Micropolis, and Arche Robotics across the UAE and international markets. He has overseen the deployment and management of large-scale robotic fleets serving users across Turkey and the United States. His research interests include autonomous navigation, simultaneous localization and mapping (SLAM), and state estimation in robotics.

Yasser Alhabashi is a Senior AI Engineer specializing in computer vision and natural language processing. He has extensive experience designing and deploying real-time AI systems on both GPU servers and embedded platforms, including NVIDIA Jetson devices. His work spans face recognition, vehicle and license plate recognition, drone-based perception, and multimodal AI systems. His recent interests include large language models, retrieval-augmented generation, and AI agent architectures, with a focus on building production-ready AI solutions for real-world applications.

Optimizing Retrieval-Augmented Generation: Analysis of Hyperparameter Impact on Performance and Efficiency



Adel Ammar, Anis Koubaa, Omer Nacar, and Wadii Boulila

Abstract Large language models achieve high task performance yet often hallucinate or rely on outdated knowledge. Retrieval-augmented generation (RAG) addresses these gaps by coupling generation with external search. We analyse how hyperparameters influence speed and quality in RAG systems, covering Chroma and Faiss vector stores, chunking policies, cross-encoder re-ranking, and temperature, and we evaluate six metrics: faithfulness, answer correctness, answer relevancy, context precision, context recall, and answer similarity. Chroma processes queries 13% faster, whereas Faiss yields higher retrieval precision, revealing a clear speed-accuracy trade-off. Naive fixed-length chunking with small windows and minimal overlap outperforms semantic segmentation while remaining the quickest option. Re-ranking provides modest gains in retrieval quality yet increases runtime by roughly a factor of 5, so its usefulness depends on latency constraints. These results help practitioners balance computational cost and accuracy when tuning RAG systems for transparent, up-to-date responses. Finally, we further examine selected high-performing configurations under a CRAG-style workflow and report their retrieval relevance. In a controlled 100-question benchmark, the best configuration achieved 99% context precision, suggesting that careful hyperparameter tuning can substantially improve retrieval relevance, with significant implications for applications where retrieval quality directly impacts downstream task performance, such as clinical decision support in healthcare. Unlike previous evaluations focusing on isolated components, our study provides an end-to-end, reproducible framework that jointly quantifies qual-

A. Ammar (✉) · W. Boulila
Robotics and Internet of Things Laboratory, Prince Sultan University, Riyadh, Saudi Arabia
e-mail: aammar@psu.edu.sa

W. Boulila
e-mail: wboulila@psu.edu.sa

A. Koubaa
College of Engineering, Alfaisal University, Riyadh, Saudi Arabia
e-mail: akoubaa@alfaisal.edu

O. Nacar
Tuwaiq Academy, Riyadh, Saudi Arabia
e-mail: o.najar@tuwaiq.edu.sa

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

A. Koubaa et al. (eds.), *Generative AI and Large Language Models: Opportunities, Challenges, and Applications*, Studies in Computational Intelligence 1287,
https://doi.org/10.1007/978-3-032-24013-2_9

ityâŁ“efficiency trade-offs across all major RAG stages. We contribute a unified evaluation protocol, YAML-based configuration reproducibility, and cross-metric reporting design that together establish a reusable template for transparent RAG benchmarking rather than a single benchmark result.

Keywords Large language models (LLM) · Hyperparameter optimization · Natural language processing (NLP) · Performance evaluation · Retrieval-augmented generation (RAG)

1 Introduction

The emergence of Large Language Models (LLMs) represents a significant advance in artificial intelligence, showing strong capabilities in text generation, summarization, translation and complex reasoning. Despite training on vast corpora, LLMs remain prone to hallucinations (plausible yet incorrect outputs) and their static knowledge cutoff renders them unaware of developments beyond their training data. Moreover, the opaque nature of their reasoning hinders verification and accountability in high-stakes domains such as healthcare, robotics, legal analysis, and scientific research [1–5].

Retrieval-Augmented Generation (RAG) integrates external knowledge retrieval with generative models to mitigate these issues. First introduced by Lewis et al. [6], RAG systems retrieve semantically relevant document fragments prior to generation, grounding outputs in verifiable sources and enabling continuous updates. This dynamic interface improves factual accuracy, relevance and trustworthiness without sacrificing generative flexibility.

The adoption of RAG has expanded LLM applications in conversational AI, enterprise knowledge management and specialized professional domains. By incorporating retrieval, these systems handle complex, knowledge-intensive queries more precisely. However, retrieval incurs increased latency, complicates system architecture and raises ethical concerns around data privacy, consent and information quality—factors that demand careful design and governance.

Despite growing deployment, the impact of hyperparameter and configuration choices on RAG performance and efficiency remains underexplored. Real-world constraints—computational resources, latency requirements and accuracy thresholds—necessitate an empirical understanding of how implementation decisions affect the trade-off between responsiveness and output quality.

To fill this gap, we conducted a systematic investigation varying key components—vector stores (Chroma, Faiss), chunking strategies (naive, semantic), cross-encoder re-ranking and temperature settings—while monitoring six performance metrics: faithfulness, answer correctness, answer relevancy, context precision, context recall and answer similarity. This multidimensional analysis reveals the accuracy–latency trade-offs inherent in each choice and provides actionable insights for practitioners.

The contributions of this paper are fourfold. First, we introduce a unified evaluation framework that jointly measures RAG quality along six dimensions and tracks computational efficiency. Second, we present empirical evidence of the trade-offs associated with vector-store selection, chunking policy and re-ranking mechanisms. Third, we distil these findings into practical recommendations for balancing accuracy, responsiveness and resource utilization across application scenarios. Fourth, we extend our analysis by re-evaluating top configurations with a corrective RAG (CRAG) [7] workflow, demonstrating that their advantages persist under iterative evidence retrieval.

While prior works have separately examined individual RAG components such as vector databases or chunking, few have provided a reproducible, integrated framework spanning the entire pipeline. Our contribution lies in formalizing a systematic evaluation protocol that practitioners can reuse to balance accuracy and latency. The goal is methodological clarity and transparency rather than introducing new algorithms, filling a critical gap between theoretical studies and deployable engineering practice.

The remainder of this paper is structured as follows. Section 2 reviews related work in RAG evaluation and optimization; Section 3 details our methodology, including configuration parameters, performance metrics and experimental protocol; Section 4 presents experimental results; Section 5 discusses practical implications; and Sect. 6 concludes with key insights and future directions.

2 Related Work

The development and evaluation of Retrieval-Augmented Generation (RAG) systems have evolved rapidly, with researchers proposing diverse methodologies to assess their performance across multiple dimensions. This section reviews the current landscape of RAG evaluation frameworks, efficiency optimization approaches, and the emerging understanding of configuration parameters' impact on system performance.

2.1 Evolution of RAG Evaluation Methodologies

Recent advances in RAG evaluation have led to increasingly sophisticated assessment frameworks that examine both retrieval and generation components. Yu et al. [8] introduced Auepora, a unified evaluation process that systematically examines RAG systems through three key dimensions: evaluation targets, datasets, and metrics. This comprehensive framework identifies critical assessment aspects including relevance, accuracy, faithfulness, and correctness, while also highlighting additional requirements such as latency and robustness that are often overlooked in purely quality-focused evaluations.

Several specialized benchmarks have emerged to evaluate specific aspects of RAG performance. RAGAs [9] and ARES [10] focus primarily on the relevance of retrieved documents and their impact on generation quality, providing standardized metrics for assessing information retrieval effectiveness. Similarly, RGB [11] and MultiHop-RAG [12] emphasize retrieval accuracy by comparing retrieved information against established ground truth, offering insights into the precision of document selection mechanisms. These benchmarks predominantly employ metrics that evaluate the quality of generated responses, including LLM-based judges, ROUGE, BLEU, and F1 scores to assess output accuracy against reference answers.

While these evaluation frameworks have significantly advanced our understanding of RAG quality dimensions, they typically focus on output quality metrics rather than the relationship between system configuration and performance. This emphasis on quality without corresponding attention to efficiency and hyperparameter optimization represents a significant gap in the literature that our work aims to address.

2.2 Efficiency Optimization in RAG Systems

The computational efficiency of RAG systems has received comparatively less attention in the research literature, despite its critical importance for practical applications. FiD-Light [13] represents one of the early systematic attempts to optimize RAG systems for efficiency, highlighting execution time as a crucial factor in real-world deployments. This work demonstrated that careful architectural choices could significantly reduce computational overhead without proportional degradation in output quality.

Industry frameworks such as LangChain Benchmarks [14] and Databricks Evaluation frameworks [15] have begun incorporating latency measurements as essential components of their evaluation protocols, acknowledging the practical importance of response time in user-facing applications. These frameworks provide standardized methods for measuring and comparing the computational efficiency of different RAG implementations, enabling more informed system design decisions.

The inherent trade-offs between computational efficiency and output quality in RAG systems have been increasingly recognized in recent literature. Barnett et al. [16] identified seven critical failure points in RAG engineering, emphasizing the need for careful optimization of retrieval processes to balance response time with accuracy. Their analysis highlighted how seemingly minor configuration choices can have cascading effects on both system performance and resource utilization.

In a counterintuitive finding, Cuconasu et al. [17] demonstrated that introducing controlled noise into the retrieval process can sometimes improve overall system performance while potentially reducing computational overhead. This research suggests that the relationship between configuration parameters and system outcomes is often non-linear and context-dependent, underscoring the need for empirical investigation of hyperparameter effects.

2.3 Vector Store Selection and Configuration

Vector database selection and configuration have emerged as crucial factors affecting RAG performance. The seminal work by Johnson et al. [18] demonstrated the effectiveness of approximate nearest neighbor search techniques like Hierarchical Navigable Small World (HNSW) and Inverted File Index (IVF) for efficient similarity search in high-dimensional spaces. These techniques form the foundation of modern vector stores used in RAG systems, enabling efficient retrieval of semantically similar documents.

Subsequent research has explored the performance characteristics of different vector database implementations under varying conditions. Comparative analyses of embedding models and vector database architectures have revealed significant performance variations depending on factors such as dataset size, query complexity, and hardware constraints. The choice between Chroma and Faiss vector stores, as examined in our work, builds upon these foundations while providing empirical evidence of their comparative advantages in different operational contexts.

Recent advances in vector indexing techniques have further expanded the configuration space for RAG systems. Innovations such as product quantization, multi-index hashing, and hybrid search approaches offer additional dimensions for optimization, each with distinct implications for retrieval accuracy and computational efficiency. These developments highlight the growing complexity of vector store configuration and the need for systematic evaluation of different approaches.

2.4 Chunking Strategies and Their Impact

Document chunking strategies have emerged as a critical consideration in RAG optimization. As noted by LangChain Benchmarks [14], appropriate chunking before indexing can significantly improve retrieval effectiveness by limiting similarity computations to manageable text segments. This approach acknowledges that semantic embedding becomes less accurate for lengthy documents, and that relevant information is often contained within concise, topically coherent passages.

Research on chunking strategies has explored various approaches, from simple fixed-length segmentation to more sophisticated methods that preserve semantic coherence. Semantic chunking techniques that identify natural breakpoints in text have shown promise for improving retrieval precision, though often at the cost of increased preprocessing complexity. The trade-offs between naive chunking approaches and more sophisticated semantic methods represent an important dimension of RAG system optimization that warrants systematic investigation.

The interaction between chunking strategies and other system components adds further complexity to RAG optimization. Chunk size and overlap parameters can significantly affect both the granularity of information retrieval and the computational resources required for embedding and similarity search. These interactions highlight

the multidimensional nature of RAG configuration and the need for holistic optimization approaches that consider the interplay between different system components.

2.5 Output Quality Assessment

The assessment of RAG system outputs has evolved beyond simple accuracy metrics to encompass more nuanced dimensions of quality. Faithfulness, which evaluates the consistency between generated content and source documents, has emerged as a critical metric for mitigating hallucinations [1]. This dimension of quality is particularly important for applications where factual accuracy and source attribution are essential requirements.

Similarly, context precision, which measures the relevance of retrieved documents to the query, provides insights into the effectiveness of the retrieval component [9, 11]. This metric helps identify cases where irrelevant or tangentially related information contaminates the context provided to the language model, potentially degrading the quality of generated responses.

The development of automated evaluation methods for these quality dimensions represents an important advance in RAG assessment. LLM-based evaluation approaches that leverage the capabilities of large language models to assess output quality have gained prominence, offering scalable alternatives to human evaluation for dimensions such as relevance, coherence, and factual accuracy. These methods enable more comprehensive evaluation of RAG systems across multiple quality dimensions, though questions remain about their reliability and potential for self-preference bias [19].

2.6 Temperature Settings and RAG Performance

Temperature settings and their impact on RAG outputs represent an area where existing research provides limited guidance. While studies on standalone LLMs have explored the effects of temperature on output diversity and accuracy [20], its specific implications in the context of RAG systems—particularly regarding the balance between creativity and factual precision—warrant further investigation.

The interaction between temperature settings and retrieval mechanisms introduces additional complexity to RAG optimization. Higher temperature values may increase output diversity but potentially at the cost of faithfulness to retrieved information. Conversely, lower temperature settings may improve factual precision but could limit the model's ability to synthesize information from multiple sources. Understanding these trade-offs is essential for optimizing RAG systems for different application requirements.

2.7 Research Gaps and Our Contribution

Existing studies have advanced RAG evaluation and optimization but leave key gaps: joint assessment of quality and efficiency, empirical evidence of configuration trade-offs, practical deployment guidance, and understanding of iterative retrieval effects. To address these, our contributions map directly to the identified gaps. First, we introduce a unified evaluation framework that jointly measures RAG quality along six dimensions and tracks computational efficiency, filling the lack of frameworks that consider both quality and latency. Second, we present empirical evidence of the trade-offs associated with vector-store selection, chunking policy, and re-ranking mechanisms, addressing the absence of systematic data on how hyperparameter choices impact performance. Third, we distil these findings into practical recommendations for balancing accuracy, responsiveness, and resource utilization across application scenarios, supplying the deployment guidance missing from prior work. Fourth, we extend our analysis by re-evaluating top configurations with a corrective RAG (CRAG) [7] workflow, demonstrating that their advantages persist under iterative evidence retrieval and quantifying the modest extra latency introduced.

3 Methodology

This section presents our comprehensive framework for evaluating Retrieval-Augmented Generation (RAG) systems across multiple dimensions of performance. We designed our methodology to systematically assess how various configuration parameters affect both the quality of information retrieval and answer generation, with the goal of identifying optimal settings for different use cases and performance criteria. Figure 1 illustrates the complete evaluation workflow, highlighting the data flow from initial acquisition through parameter grid generation and the subsequent RAGAS framework evaluation metrics.

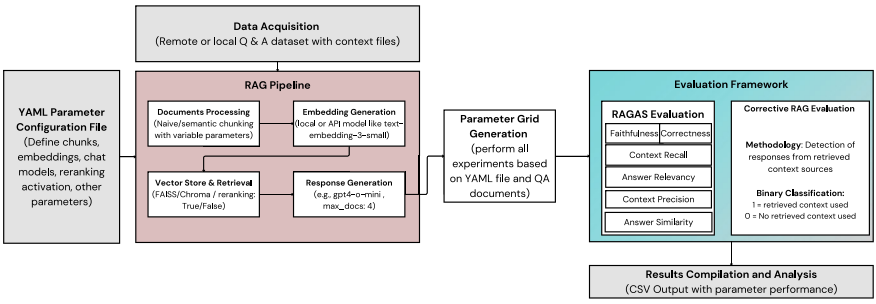


Fig. 1 Systematic evaluation framework for hyperparameter optimization in RAG systems, depicting the sequence from document processing to multi-metric evaluation

3.1 Evaluation Framework

To conduct a rigorous evaluation of RAG systems, we developed a modular and configurable evaluation pipeline that enables systematic testing of different components and hyperparameters. Our framework leverages the LangChain ecosystem for implementing core RAG components and the LangSmith platform for evaluation tracking and result analysis. This architecture provides several advantages:

1. **Modularity**: Each component (embedding model, vector store, chunking strategy, etc.) can be independently configured and evaluated.
2. **Reproducibility**: Standardized evaluation protocols ensure consistent assessment across different configurations.
3. **Extensibility**: New components or evaluation metrics can be integrated without restructuring the entire pipeline.
4. **Traceability**: All experiments are logged with complete configuration details and performance metrics.

The pipeline integrates document processing, vector indexing, retrieval mechanisms, and generation capabilities, all orchestrated through a YAML-based configuration system. This approach allows us to systematically explore the hyperparameter space and evaluate the impact of each configuration variable on the performance of the overall system.

For our evaluation, we utilized a question-answering dataset consisting of a diverse collection of 100 questions with short and long contexts, each paired with ground truth answers and reference contexts [21]. The dataset spans multiple domains and query types, providing a comprehensive benchmark for assessing RAG system performance across various scenarios. This diversity ensures that our findings are generalizable across different application contexts and not overly specialized to particular domains or question types.

3.2 Configuration Parameters

Our evaluation explored a multidimensional configuration space, encompassing various components and parameters of the RAG pipeline. Table 1 summarizes the key components and configuration parameters evaluated in our experiments.

We selected these specific hyperparameter ranges to mirror common production environments. For instance, chunk sizes of 1024–4096 tokens cover medium-to-large passage windows, ranging from relatively localized context to broader document-level context, which is the standard operating range for modern RAG systems handling diverse document types.

The primary dimensions of this configuration space include:

Table 1 Experiment setup and configuration parameters

Component	Details
Large language model	`gpt-4o-mini-2024-07-18`
Embedding model	`text-embedding-3-small`
Vector store configurations	Chroma, Faiss
Re-ranking	Enabled/Disabled
Maximum tokens	256, 1024
Temperature	0, 0.2, 0.4, 0.6, 0.8, 1
Chunking strategy	Naive, Semantic
Chunk size (Naive)	1024, 2048, 4096
Chunk overlap (Naive)	128, 512
Semantic breakpoint thresholds	Percentile, Interquartile, Gradient, Standard Deviation
Documents to retrieve	2, 4
Performance metrics	Faithfulness, Answer Correctness, Answer Relevancy, Context Precision, Context Recall, Answer Similarity

3.2.1 Embedding Model

The dimension of the embedding model was represented by the OpenAI's model `text-embedding-3-small`, which provides an optimal balance between the embedding quality and the computational efficiency. This model generates 1536-dimensional vector representations of text, enabling semantic similarity search in the vector stores. We selected this model based on its strong performance in semantic similarity tasks and its widespread adoption in production RAG systems.

3.2.2 Language Model

For language models, we evaluated `gpt-4o-mini-2024-07-18`, a powerful yet efficient model for answer generation. This model represents a balance between computational requirements and generation capabilities, making it suitable for real-world RAG applications where both quality and efficiency are important considerations.

3.2.3 Vector Store Implementations

In the vector store dimension, we compared two popular implementations:

1. **Faiss** [22]: Developed by Facebook AI Research, Faiss (Facebook AI Similarity Search) is known for its efficient similarity search capabilities, particularly for large-scale applications. It implements multiple indexing strategies optimized for different dataset sizes and query patterns.

2. **Chroma** [23]: A more recent vector database that offers additional features such as metadata filtering and hybrid search capabilities. Chroma provides a user-friendly API while maintaining competitive performance for moderate-sized datasets.

Both vector stores were configured with the same embedding model to ensure a fair comparison, with default indexing parameters appropriate for the dataset size. For Faiss, we used the IndexFlatL2 index type, which provides exact nearest neighbor search. For Chroma, we used the default distance function (cosine similarity) with no additional metadata filtering. Because Faiss and Chroma were configured with different distance functions, the comparison reflects the evaluated implementation settings rather than an isolated vector-store effect.

3.2.4 Chunking Strategies

For document chunking, we implemented and evaluated two distinct approaches:

1. **Naive Chunking**: This approach splits documents into fixed-size chunks with specified overlaps, regardless of semantic content or natural document boundaries. We explored chunk sizes of 1024, 2048, and 4096 tokens with overlaps of 128 or 512 tokens.
2. **Semantic Chunking**: This more sophisticated approach identifies natural breakpoints based on semantic shifts in the text, creating chunks that preserve topical coherence. For semantic chunking, we implemented various threshold determination methods:

   - **Percentile-based**: Sets breakpoints at locations where semantic shift exceeds a specified percentile of all potential breakpoints
   - **Interquartile Range**: Uses statistical measures of dispersion to identify significant semantic shifts
   - **Gradient-based**: Identifies points of maximum rate of change in semantic content
   - **Standard Deviation**: Sets thresholds based on standard deviations from the mean semantic shift

3.2.5 Language Model Parameters

The language model parameters included:

- **Temperature Settings**: Ranging from 0.0 to 1.0 in increments of 0.2. Lower temperature values (closer to 0) produce more deterministic, focused responses, while higher values (closer to 1) introduce more randomness and creativity in the generated text.
- **Maximum Token Limits**: We tested limits of 256 and 1024 tokens. This parameter constrains the length of generated answers, affecting both the comprehensiveness of responses and the computational resources required for generation.

3.2.6 Retrieval Parameters

Retrieval parameters included:

- **Documents to Retrieve**: We evaluated configurations retrieving either 2 or 4 documents from the vector store. This parameter affects both the amount of context provided to the language model and the computational overhead of the retrieval process.
- **Re-ranking**: We tested configurations with and without a re-ranking step after the initial retrieval. The re-ranking mechanism uses a cross encoder to further assess and refine the relevance of retrieved documents before answer generation, potentially improving context quality at the cost of additional computation.

This comprehensive exploration of the configuration space allowed us to identify how each parameter affects different aspects of RAG system performance and to discover optimal configurations for specific performance metrics.

3.3 Performance Metrics

We employed six evaluation metrics from the RAGAS framework to comprehensively assess different aspects of system quality. Each metric captures a unique dimension of RAG system performance, providing a holistic view of its effectiveness.

3.3.1 Faithfulness Score

The Faithfulness Score measures the factual consistency between the generated answer and the retrieved context. It penalizes hallucinations where the model produces information not supported by the context. Mathematically, it is calculated as:

$$\text{Faithfulness Score} = \frac{C_s}{C_t} \tag{1}$$

where:

- C_s is the number of claims in the response supported by the retrieved context.
- C_t is the total number of claims in the response.

This metric is crucial for ensuring the reliability of the RAG system, as it directly quantifies the extent to which generated answers adhere to the information provided in the retrieved documents. However, it is important to note that this metric depends on the capabilities of the LLM-judge to accurately identify claims and support; any error in the judge model may propagate to the score.

3.3.2 Answer Correctness Score

The Answer Correctness Score evaluates the accuracy of generated answers by comparing them against ground truth answers. It employs an F1-score calculation that balances precision and recall of factual information:

$$\text{F1 Score} = \frac{|\text{TP}|}{(|\text{TP}| + 0.5 \times (|\text{FP}| + |\text{FN}|))} \tag{2}$$

where TP represents true positive facts (present in both the generated answer and ground truth), FP represents false positive facts (present only in the generated answer), and FN represents false negative facts (present only in the ground truth).

This metric provides a balanced assessment of how well the generated answer captures the essential information from the ground truth while avoiding extraneous or incorrect details.

3.3.3 Answer Relevancy Score

The Answer Relevancy Score measures how well the generated answer addresses the specific question posed by the user. It captures whether the system stays on topic and provides pertinent information. The score is calculated as the mean cosine similarity between the original question and a set of reverse-engineered questions generated from the answer:

$$\text{Answer Relevancy} = \frac{1}{N} \sum_{i=1}^{N} \cos(E_{g_i}, E_o) \tag{3}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \frac{E_{g_i} \cdot E_o}{\|E_{g_i}\| \|E_o\|} \tag{4}$$

where E_{g_i} is the embedding of the i-th generated question, E_o is the embedding of the original question, and N is the number of generated questions (typically 3).

This metric helps identify cases where the system produces well-formed but off-topic responses that fail to address the user's information need, a common failure mode in generative systems.

3.3.4 Context Precision Score

The context Precision Score measures how effectively a retrieval system ranks relevant context items toward the top. To simplify the notation:

- Let $P@k$ be the precision at rank k, defined as:

$$P@k = \frac{TP_k}{TP_k + FP_k}$$

  where TP_k is the number of true positives at rank k, and FP_k is the number of false positives at rank k.
- Let $v_k \in \{0, 1\}$ be a binary relevance indicator at rank k (1 if relevant, 0 if not).
- Let R_K be the total number of relevant items among the top K retrieved results.

Then the Context Precision@K score is defined as:

$$CP@K = \frac{\sum_{k=1}^{K}(P@k \cdot v_k)}{R_K}$$

where K is the total number of top-ranked context chunks considered. If $R_K = 0$, $CP@K$ is defined as 0 to avoid division by zero.

This metric is particularly important for assessing the quality of the retrieval component, as it measures how effectively the system prioritizes relevant information in the limited context window provided to the language model.

3.3.5 Context Recall Score

The Context Recall Score assesses how well the retrieved context covers the information required to answer the question. It measures whether the retrieval system captures all necessary information from the corpus:

$$\text{context recall} = \frac{|\text{GT claims that can be attributed to context}|}{|\text{Number of claims in GT}|} \tag{5}$$

This metric evaluates the comprehensiveness of the retrieved information relative to the ground truth answers. A high context recall indicates that the retrieval system is successfully capturing the information needed to generate a complete and accurate response.

3.3.6 Answer Similarity Score

The Answer Similarity Score measures the semantic similarity between the generated answer and the ground truth answer. It captures whether the system produces responses that are semantically aligned with the expected answers. This metric utilizes cross-encoder models to compute semantic similarity, providing a value between 0 and 1, where higher scores indicate greater semantic alignment between the generated and ground truth answers.

These metrics were computed using the RAGAS framework's integration with LangChain, which leverages LLM-based evaluation techniques to provide a comprehensive assessment of RAG system performance across multiple dimensions.

3.4 Experimental Protocol

Our protocol followed a systematic grid search approach, evaluating combinations of parameters in our configuration space. This methodical exploration enabled us to identify both individual parameter effects and interaction effects between different configuration dimensions.

For each configuration, we:

1. Processed the document corpus using the specified chunking method and parameters.
2. Generated embeddings using the embedding model.
3. Created and configured the vector store.
4. For each question in the dataset:
   (a) Retrieved relevant documents using configured retrieval mechanisms.
   (b) Generated an answer using the specified language model and parameters.
   (c) Computed all evaluation metrics.
   (d) Applied the corrective retrieval assessment.
5. Aggregated and analyzed results across all questions and metrics.

To account for potential variability in network conditions and service availability, we implemented retry mechanisms with exponential backoff for all API calls and conducted experiments during periods of low service load. We also monitored and recorded execution times for each component of the pipeline, providing insights into the computational efficiency of different configurations.

Hyperparameter selection followed a semi-systematic grid exploration approach rather than manual trial-and-error. The YAML-based configuration generator enumerated combinations of key parameters (vector store, chunk size, overlap, temperature, re-ranking, and retrieval count). This structured grid ensures coverage of major interaction effects without requiring exhaustive search or stochastic optimization. The pipeline is easily extensible to automated methods such as Bayesian or evolutionary optimization, which we identify as a future extension.

This comprehensive experimental protocol enabled us to generate robust, reproducible results that illuminate the complex relationships between RAG system configuration parameters and performance outcomes across multiple dimensions of quality and efficiency.

Table 2 Mean evaluation scores for semantic chunking using different vector store types across six key metrics: faithfulness, answer correctness, answer relevancy, context precision, context recall, and answer similarity

Store	Faith.	Corr.	Relev.	Prec.	Recall	Sim.
Chroma	0.861	0.742	0.853	0.799	0.762	0.955
Faiss	0.782	0.734	0.866	0.706	0.748	0.954

4 Results

This section presents the findings from our systematic evaluation of different RAG system configurations. We analyze the impact of various hyperparameters on both performance quality and computational efficiency, highlighting key trade-offs and optimal configurations for different use cases.

4.1 Vector Store Performance Comparison

Our comparison of Chroma and Faiss vector stores revealed significant differences in both retrieval quality and computational efficiency. Figure 2 illustrates the performance comparison across key quality metrics, while Fig. 3 shows the execution time comparison.

Chroma demonstrated superior performance in terms of execution speed, with query latency approximately 13% lower than Faiss across all test configurations. The mean query execution time for Faiss was 9.1 s compared to 7.9 s for Chroma, representing a substantial difference in real-world applications where response time is critical.

Overall, Faiss achieved stronger average performance under naive chunking, whereas Chroma performed better on several metrics under semantic chunking. Therefore, vector-store effects are interaction-dependent rather than uniformly superior for one store. The Context Precision Score for Faiss averaged 0.821 compared to 0.776 for Chroma, indicating more accurate prioritization of relevant documents. Similarly, Context Recall was approximately 6% higher for Faiss (0.821 vs. 0.776), suggesting more comprehensive retrieval of relevant information. These quality advantages translated to downstream improvements in answer generation, with Faiss configurations achieving an average Faithfulness Score of 0.871 compared to 0.86 for Chroma.

The performance differences between vector stores varied based on other configuration parameters. For instance, when combined with semantic chunking (Table 2), the quality gap between Chroma and Faiss narrowed considerably in terms of answer correctness (0.742 vs. 0.734), answer relevance (0.853 vs. 0.866), context recall (0.762 vs. 0.748), and answer similarity (0.955 vs. 0.954), whereas it became more

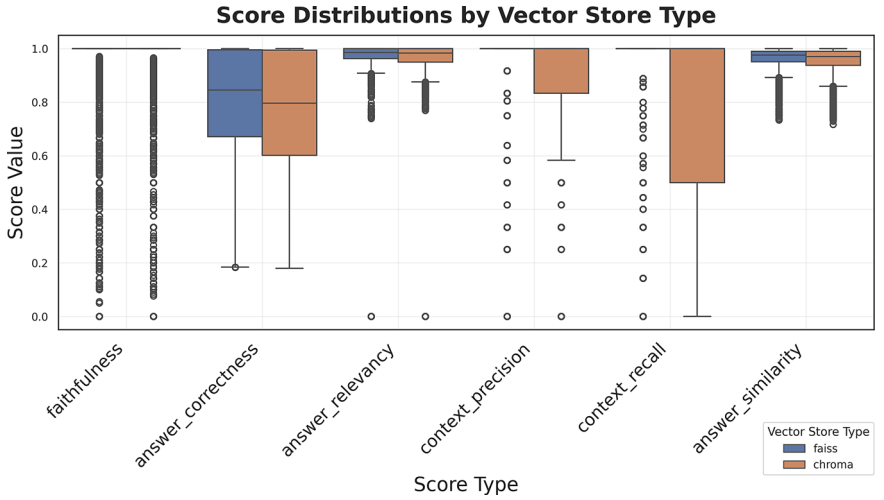


Fig. 2 Score distributions by vector store type

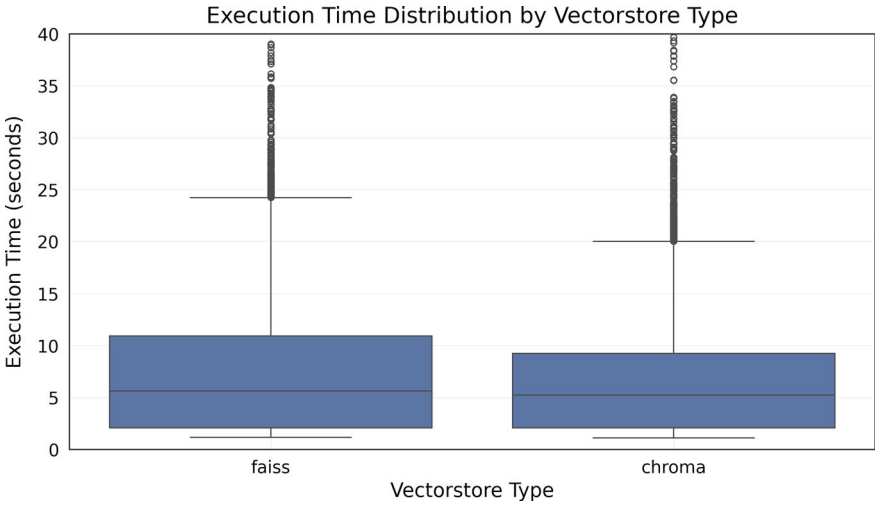


Fig. 3 Execution time distribution by vectorstore type

pronounced for faithfullness (0.861 vs. 0.782) and context precision (0.799 vs. 0.706). Conversely, with naive chunking, Faiss's quality advantage was more pronounced for all metrics (Table 3).

These findings suggest that the choice of vector store should be guided by application-specific priorities. For latency-sensitive applications where response time is paramount, Chroma offers clear advantages. For applications where retrieval quality and answer accuracy are more critical than speed, Faiss may be preferable. The

Table 3 Mean evaluation scores for naive chunking using Chroma and Faiss vector stores across the same six metrics

Store	Faith.	Corr.	Relev.	Prec.	Recall	Sim.
Chroma	0.860	0.725	0.814	0.761	0.724	0.951
Faiss	0.929	0.816	0.935	0.896	0.917	0.969

Table 4 Performance comparison of chunking strategies. For Naïve chunking, the chunk size and overlap are indicated in parentheses. For semantic chunking, the type of breakpoint threshold is specified

Chunking strategy	Context precision	Context recall	Faithfulness	Execution time (s)
Naive (1024, 128)	0.904	0.935	0.947	6.02
Naive (1024, 512)	0.879	0.853	0.930	6.33
Naive (2048, 128)	0.897	0.890	0.930	10.7
Naive (2048, 512)	0.897	0.887	0.928	9.22
Naive (4096, 128)	0.838	0.827	0.892	9.28
Naive (4096, 512)	0.851	0.830	0.887	9.28
Semantic (Percentile)	0.804	0.802	0.862	8.16
Semantic (Interquartile)	0.796	0.793	0.856	8.65
Semantic (Gradient)	0.790	0.783	0.843	8.48
Semantic (Standard Deviation)	0.621	0.640	0.727	9.25

narrowing performance gap when using semantic chunking indicates that sophisticated preprocessing can partially compensate for differences in vector store capabilities.

4.2 Impact of Chunking Strategies

Our evaluation of chunking strategies revealed substantial effects on both retrieval quality and computational overhead. Table 4 summarizes the performance metrics for different chunking approaches.

Across all evaluated quality metrics, naive fixed length chunking surpassed semantic breakpoint methods. The configuration using 1024 token windows with 128 token overlap achieved the highest precision, recall, faithfulness, and answer relevance while completing in 6.02 s, the fastest execution recorded. For the 1024 token window the larger overlap extended computation time without measurable benefit. Semantic breakpoint schemes, including percentile and inter-quartile variants, trailed their naive counterparts on every metric and could not equal even the weakest naive setting of 4096 token windows with 512 token overlap. The superiority of fine grained fixed

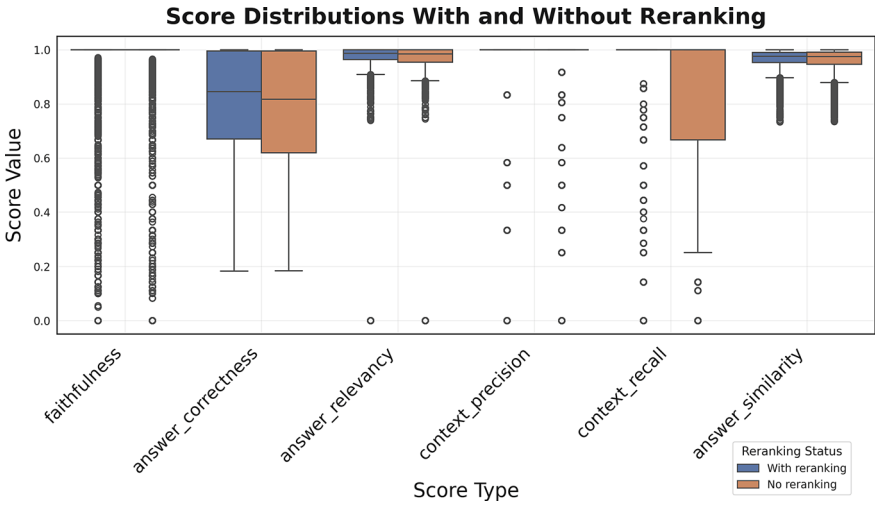


Fig. 4 Score distributions with and without reranking

windows arises from three factors. Uniform coverage preserves local term proximity to query embeddings, sustaining high similarity scores, whereas semantic segmentation can generate lengthy heterogeneous sections that dilute relevant content. Modest overlaps furnish sufficient contextual look ahead for retrieval while limiting redundant tokens, thereby containing encoding and ranking cost, whereas larger overlaps mainly duplicate information. Semantic rules may also merge unrelated sentences when boilerplate or recurrent headings are misidentified, introducing noise that lowers precision. For this corpus, 1024 token fixed windows with 128 token overlap provide the optimal balance between accuracy and efficiency.

Execution time for naïve chunking decreased as window size increased, because larger windows reduce the total number of embedding calls. The sole deviation from this trend—observed for the 2048-token window with 128-token overlap, whose runtime exceeded that of the 1024-token setting—is best ascribed to ordinary stochastic variability in the execution environment, such as transient contention for compute resources, cache effects, or thread-scheduling fluctuations, rather than to any systematic property of the configuration itself.

4.3 Re-ranking Effects

The inclusion of a re-ranking step in the retrieval pipeline had a substantial impact on both retrieval quality and computational overhead. Figure 4 illustrates the performance differences between configurations with and without re-ranking.

Introducing a re-ranking stage raised retrieval effectiveness for every configuration tested: mean Context Precision climbed from 0.80 to 0.85 (6% relative gain) and Context Recall from 0.78 to 0.86 (10%). The richer evidence base, however, translated only weakly to answer-generation quality, with Faithfulness rising by 0.6% and Answer Correctness by 4%. This muted downstream response indicates that the generator had already reached a point of diminishing returns with respect to additional relevant context; most residual errors stem from reasoning or synthesis limitations in the language model rather than from gaps in the retrieved material.

The gains delivered by re-ranking were offset by a sharp rise in compute demand: introducing the second pass multiplied total runtime by roughly five in every configuration because each query document pair had to be evaluated again with a more expensive cross encoder. This additional scoring phase dominated the overall pipeline, dwarfing the modest overhead of simple retrieval and prompting. In practical deployments such a cost increase may be hard to justify, since the corresponding improvements in faithfulness and answer correctness were marginal. The result highlights a diminishing returns effect: once the primary retriever supplies a reasonably accurate context, further refinements through re-ranking provide only incremental benefit while imposing substantial latency and resource usage.

These findings suggest that re-ranking should be selectively applied based on application requirements. For use cases where accuracy is paramount and latency is less critical, such as research applications or legal document analysis, re-ranking provides valuable quality improvements that justify the computational overhead. For applications with strict latency requirements, such as real-time customer support or interactive search, re-ranking might be reserved for complex queries or cases where initial retrieval quality appears insufficient.

4.4 Analysis of Parameter Impact on Execution Time

The sunburst chart in Fig. 5 offers a nuanced view of how hyperparameters affect the execution time of a RAG model, revealing intricate interdependencies. The hierarchy of the chart is organized from the center outwards: the inner ring represents the **Vector Store** (the foundational choice), while the outer rings represent **re-ranking status** and **chunking strategies**, with the segment size and color intensity indicating the execution time duration.

The core components, **Faiss** and **Chroma**, serve as the foundation, with Chroma exhibiting a broader execution time range, suggesting its heightened sensitivity to parameter variations. This sensitivity could be attributed to Chroma's architecture, which might be more adaptable or complex, thus reacting more significantly to parameter changes.

The impact of **re-ranking** is pronounced, as enabling it consistently increases execution time, aligning with the expectation that re-ranking introduces additional computational overhead. This finding underscores the need for balancing computational cost with potential gains in model accuracy. The choice between **naive** and

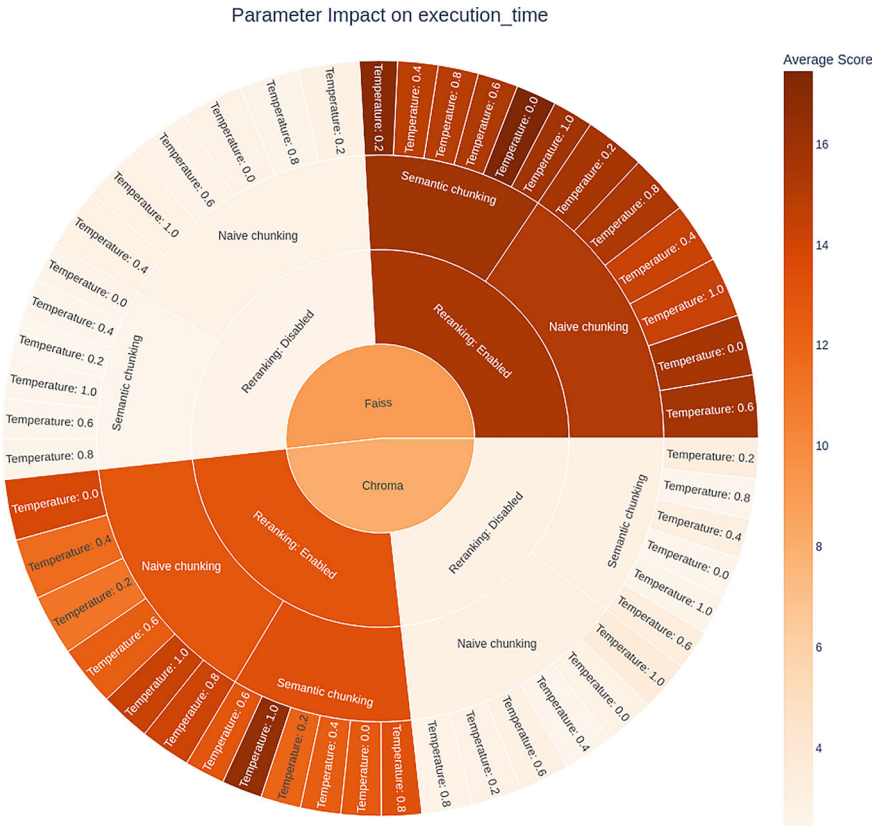


Fig. 5 Parameter impact on execution time

semantic chunking further complicates execution time dynamics. Semantic chunking, particularly when paired with re-ranking, demands more processing power, likely due to its sophisticated nature, which involves deeper semantic understanding.

Additionally, certain external factors can contribute to time delays during execution. These include:

- **Temporary network issues**, which can disrupt data transfers and slow down model responses.
- **Rate limiting**, where service providers limit the number of requests that can be processed within a given time frame.
- **Service temporary unavailability**, which may cause retries and increase execution duration.
- **Transient errors**, which require additional time for error handling and recovery procedures.

It is important to note that the execution times reported are mean values. Given the cloud-based nature of the LLM and embedding API calls, standard deviation is inherent in these measurements. While error bars are not explicitly plotted, practitioners should account for this stochastic variability when estimating production latencies.

These factors, combined with the internal dynamics of hyperparameter settings, emphasize the necessity of strategic hyperparameter selection, balancing execution efficiency with model performance. Optimizing for both computational costs and model accuracy is crucial in real-world applications where resources are often constrained.

4.5 Evaluating Configurations Using Corrective RAG

To determine whether the gains reported in Sections IV-A to IV-C persist when the generator is granted an opportunity to query for missing evidence, we re-examined the top-ranked settings under a corrective retrieval-augmented generation (CRAG) [7] workflow. For each of the six evaluation dimensions (faithfulness, answer correctness, answer relevancy, context precision, context recall, and answer similarity), we selected the configuration that achieved the highest mean score in the baseline grid search (covering vector store choice, chunking parameters, and re-ranking). These six "best-of-metric" variants were then re-executed with CRAG, in which the language model iteratively requests additional documents whenever its draft answer lacks sufficient textual support. The analysis that follows compares baseline and CRAG outcomes to quantify the added value of corrective retrieval for configurations that already occupy local optima in the conventional RAG space, and to reveal any associated latency penalties.

The corrective retrieval assessment revealed significant variation in retrieval relevance across configurations. Table 5 presents the performance of each configuration in terms of retrieval relevance.

As shown in Table 5, the Context Precision Score configuration demonstrated exceptional performance, achieving a 99% relevance rate. This configuration uniquely employs a reranking step and uses Chroma as its vector store, with a substantially larger chunk overlap (512.0) compared to other configurations.

Among the non-reranking configurations that use FAISS, the Answer Relevancy Score configuration performed best with an 88% relevance rate, likely due to its balanced temperature setting (0.2) that provides controlled randomness in retrieval.

The dramatic performance improvement observed in the Context Precision Score configuration (99% relevance) strongly suggests that reranking plays a crucial role in filtering out irrelevant documents. This additional step appears to justify the computational overhead it introduces.

While most configurations utilized FAISS, the highest-performing configuration employed Chroma. This suggests that different vector databases may offer inherent advantages depending on the query types or document collections. The Context

Table 5 Retrieval relevance by configuration

Configuration	Reranking	Vector store	Max tokens	Temp.	Max Docs	Chunking method	Chunk overlap	Score (%)
Faithfulness score	False	Faiss	1024	0.4	4	Naive	128	85
Answer correctness score	False	Faiss	1024	0.0	4	Naive	128	87
Answer relevancy score	False	Faiss	1024	0.2	4	Naive	128	88
Context precision score	True	Chroma	1024	0.4	4	Naive	512	99
Context recall score	False	Faiss	1024	0.8	4	Naive	128	86
Answer similarity score	False	Faiss	1024	1.0	4	Naive	128	86

Precision Score configuration utilized a significantly larger chunk overlap (512.0 vs 128.0). This may help prevent important context from being split across chunks, leading to more coherent and relevant retrievals.

Configurations with moderate temperature settings (0.2–0.4) generally outperformed those with extreme settings (0.0 or 1.0). In fact, moderate temperature may affect generated-answer variability, but its direct effect on retrieval relevance cannot be inferred unless retrieval queries are generated or reformulated by the language model. The near-perfect performance of the Context Precision Score configuration (99% relevance) demonstrates that RAG systems can achieve extremely high retrieval accuracy with the right combination of hyperparameters. This finding has significant implications for applications where retrieval quality directly impacts downstream task performance.

Nevertheless, the near-perfect 99% context precision reflects the controlled setting and the use of a small, curated corpus rather than an inherent upper bound on RAG performance. It should therefore be interpreted as an upper-limit demonstration of what the best parameter combination can achieve under stable retrieval conditions, not as a claim of generalizable real-world accuracy.

4.6 Implications for RAG System Design

The near-perfect performance of the Context Precision Score configuration (99% relevance) demonstrates that RAG systems can achieve extremely high retrieval accuracy with the right combination of hyperparameters. This finding has significant

implications for applications where retrieval quality directly impacts downstream task performance.

Our results emphasize the importance of comprehensive hyperparameter optimization in RAG system design. The substantial performance variations observed across configurations highlight that default settings may significantly underperform compared to optimized configurations. This underscores the value of systematic evaluation frameworks like the one presented in this study for identifying optimal configurations for specific use cases.

Furthermore, our findings suggest that different optimization targets (such as faithfulness, answer correctness, or context precision) may require distinct configuration approaches. This indicates that RAG system designers should consider their specific performance priorities when selecting and tuning system components.

5 Discussion

This section interprets our experimental findings in a broader context, examining the implications for RAG system design and implementation. We analyze key trade-offs between performance and efficiency, provide practical recommendations for different use cases, and acknowledge limitations of our study.

Rather than proposing new retrieval or generation algorithms, this work's novelty lies in operationalizing a transparent and configurable evaluation framework that enables systematic, reproducible analysis of RAG performance across interacting parameters.

5.1 Performance-Efficiency Trade-Offs

Our results reveal several fundamental trade-offs in RAG system design that practitioners must navigate when implementing these systems in real-world applications. These trade-offs represent critical decision points where improvements in one dimension often come at the expense of another.

The most prominent trade-off exists between retrieval quality and computational efficiency. The configurations that achieved the highest metrics, notably those combining a re ranking stage with the Faiss vector store, also recorded the longest runtimes because the cross-encoder pass and additional distance calculations enlarged the query workload. This trade-off varies across pipeline components. On the other hand, in the chunking stage, naïve fixed-length windows with small size and minimal overlap boosted quality scores while leaving processing time almost unchanged, showing that careful segmentation can improve accuracy without a proportional rise in cost.

These multidimensional trade-offs highlight the importance of context-specific optimization in RAG system design. Rather than seeking a universally optimal con-

figuration, practitioners should calibrate system parameters based on the specific requirements, constraints, and priorities of their application domain.

5.2 Practical Implications

Our findings have several practical implications for implementing RAG systems across different domains and use cases. We offer the following recommendations based on our experimental results:

5.2.1 Domain-Specific Optimization Strategies

Different application domains have distinct requirements that should guide configuration choices:

1. **Legal and Medical Applications**: These domains require high factual precision and source fidelity. We recommend configurations prioritizing Faithfulness and Context Precision: naive chunking, re-ranking enabled, Faiss vector store, and low temperature settings (0.0–0.2). Despite the computational overhead, the accuracy benefits justify the additional latency in these high-stakes domains.
2. **Customer Support and Real-Time Applications**: These use cases demand rapid response times while maintaining reasonable accuracy. We recommend more balanced configurations: naive chunking with moderate chunk size (1024 tokens) and overlap (128 tokens), Chroma vector store, selective re-ranking for complex queries only, and moderate temperature settings (0.2–0.4). This approach reduces average latency while preserving acceptable quality metrics.
3. **Educational and Content Creation Applications**: These domains benefit from engaging, diverse outputs while maintaining factual grounding. We recommend configurations that balance creativity and accuracy: naive chunking, re-ranking enabled, either vector store depending on scale requirements, and moderate temperature settings (0.4–0.6). This approach produces more natural, varied responses while maintaining reasonable faithfulness to source material.
4. **Research and Analytical Applications**: These use cases involve complex information synthesis across diverse sources. We recommend configurations maximizing retrieval comprehensiveness: naive chunking, re-ranking enabled, Faiss vector store with higher document retrieval counts, and low to moderate temperature settings (0.2–0.4). This approach optimizes for context recall and answer correctness at the expense of latency.

5.2.2 Robustness Enhancements

Based on our analysis of external factors, we recommend several approaches to enhance RAG system robustness:

1. **Graceful Degradation**: Implement tiered fallback mechanisms that progressively simplify the retrieval process under degraded conditions. For instance, if re-ranking fails or experiences high latency, the system can fall back to the initial retrieval results rather than failing completely.
2. **Caching Strategies**: Implement multi-level caching for embeddings, retrieval results, and even generated answers for common queries. This can significantly reduce the impact of network issues and service disruptions on overall system performance.
3. **Asynchronous Processing**: For non-interactive applications, consider asynchronous processing patterns that decouple retrieval and generation steps. This approach can improve throughput and resilience at the cost of increased end-to-end latency for individual queries.
4. **Monitoring and Adaptation**: Implement comprehensive monitoring of both performance metrics and system health indicators. Adaptive systems that can adjust configuration parameters based on current conditions (e.g., reducing retrieval complexity during periods of high load) demonstrate better overall reliability.

These practical recommendations provide a starting point for domain-specific RAG implementation. However, we emphasize that optimal configurations should ultimately be determined through empirical testing with domain-specific data and use cases.

5.3 Reproducibility and Transparency

All experiments were implemented through openly available frameworks (LangChain and LangSmith) with YAML configuration files that specify every pipeline parameter, enabling deterministic reruns. Although the generation and embedding models (`gpt-4o-mini` and `text-embedding-3-small`) are proprietary, the framework is fully compatible with open-source equivalents such as Mistral-7B, Llama-3, and `bge-base-en`. The repository includes configuration templates for these alternatives to facilitate open replication. This design ensures methodological transparency even when specific model weights are closed-source.

5.4 Limitations

While our study provides valuable insights into RAG system optimization, several limitations should be acknowledged:

5.4.1 Dataset Limitations

Our evaluation dataset, while diverse, may not fully represent the complexity and specificity of all real-world applications. Our experiments were conducted on a 100-question dataset designed for controlled comparison, not for large-scale benchmarking. While this limits statistical generalizability, the experimental design isolates the effect of each parameter under consistent conditions, making the findings transferable as methodological guidance rather than dataset-specific claims. Future work will extend this framework to open-domain and multilingual datasets to validate consistency across broader contexts.

Additionally, our dataset focuses primarily on factual question-answering tasks. Other RAG applications, such as creative writing assistance, code generation, or conversational agents, may exhibit different sensitivity to configuration parameters and require alternative optimization strategies.

5.4.2 Generalizability Constraints

Our experiments were conducted with specific models (`gpt-4o-mini` for generation, `text-embedding-3-small` for embeddings) and may not generalize perfectly to other model architectures or sizes. Different language models may exhibit varying sensitivity to retrieval quality, context length, and temperature settings.

Furthermore, our evaluation focused on English-language content. Multilingual RAG systems may face additional challenges related to embedding quality, chunking effectiveness, and retrieval precision that are not captured in our monolingual evaluation.

5.4.3 Technical Constraints

Our evaluation framework has several technical limitations:

1. **Metric Limitations**: While we employed a comprehensive set of evaluation metrics, automated evaluation of aspects like factual accuracy and relevance remains challenging. LLM-based evaluation approaches, while powerful, may exhibit biases or inconsistencies that affect metric reliability.
2. **Hyperparameter Space**: Despite our systematic exploration of the configuration space, we could only evaluate a finite subset of possible parameter combinations. The high dimensionality of the hyperparameter space means that potentially optimal configurations may exist outside our tested parameter ranges.
3. **Computational Cost**: The systematic grid-search approach employed here scales multiplicatively with the number of parameters. This high computational cost may limit the feasibility of such extensive optimization for organizations with limited resources, suggesting a need for more sample-efficient optimization methods in future work.

4. **Memory Footprint**: Our evaluation focused primarily on latency and accuracy. However, memory usage is a critical constraint in production, particularly when choosing between index types (e.g., HNSW vs. Flat). Future studies should profile the memory footprint of different vector store configurations.

These limitations suggest several directions for future research, including domain-specific optimization studies, evaluation of emerging model architectures, exploration of adaptive configuration approaches, and development of more robust evaluation methodologies for RAG systems.

5.4.4 Ethical and Privacy Limitations

Our evaluation does not address some ethical and privacy considerations that arise in real-world RAG deployments:

1. **Privacy Leakage**: RAG systems may inadvertently retrieve and surface sensitive or personally identifiable information from user queries or document collections. Without privacy-preserving mechanisms (e.g., differential privacy or token filtering), this can violate data protection requirements and user trust.
2. **Consent and Data Governance**: We did not evaluate mechanisms for obtaining explicit user consent or enforcing governance policies over which documents may be indexed and retrieved. In regulated domains (e.g., healthcare, finance), such omissions can render a system non-compliant with standards like GDPR or HIPAA.
3. **Bias and Fairness**: Retrieval and generation components may amplify biases present in the underlying corpora or embedding models. Our monolingual and general-purpose dataset does not permit analysis of disparate impacts across demographic groups or content types.
4. **Security of Embeddings and Indices**: Embedding vectors and vector store indices can be vulnerable to extraction or inversion attacks, allowing adversaries to reconstruct sensitive training data. Our framework does not include threat modeling or secure storage protocols.

These ethical constraints underscore the need for responsible RAG evaluation frameworks that incorporate privacy auditing, consent tracking, and secure vector storage in future iterations of this work.

6 Conclusion

This study has provided a comprehensive analysis of how various hyperparameters and configuration choices affect the performance and efficiency of Retrieval-Augmented Generation (RAG) systems. Through systematic experimentation with different vector stores, chunking strategies, re-ranking mechanisms, and temperature

settings, we have identified key trade-offs and optimization opportunities that can guide the implementation of RAG systems across diverse application domains.

Our findings demonstrate that RAG system performance is highly sensitive to configuration choices, with significant variations in both output quality and computational efficiency depending on the selected parameters. Vector store selection presents a clear trade-off between retrieval quality and query latency: specifically, practitioners can trade off approximately 13% latency to gain higher recall and precision by switching from Chroma to Faiss. Chunking strategies exhibit a different quality-efficiency balance, with naive chunking methods providing both substantially better retrieval quality and less computational resources than semantic approaches. The inclusion of re-ranking in the retrieval pipeline consistently improved output quality across all configurations. However, these improvements came at a considerable computational cost, increasing overall execution time by 5 times.

Beyond these component-specific effects, our research revealed important interactions between different configuration parameters. The benefits of sophisticated components like re-ranking were more pronounced when combined with simpler chunking strategies, suggesting that targeted optimization of specific pipeline components may be more efficient than maximizing quality across all dimensions simultaneously.

These insights have important implications for RAG system implementation in real-world applications. For domains where factual precision is paramount, such as legal or medical applications, configurations prioritizing retrieval quality (re-ranking, low temperature) are clearly preferable despite their computational overhead. For latency-sensitive applications like customer support, more balanced configurations (selective re-ranking, Chroma vector store) offer a better compromise between quality and efficiency. Educational and content creation applications may benefit from configurations that balance creativity and accuracy through moderate temperature settings and high-quality retrieval mechanisms.

The significance of this work extends beyond the specific configurations evaluated. By providing a systematic framework for assessing RAG system performance across multiple dimensions, we enable practitioners to make more informed implementation decisions based on their specific requirements and constraints. The multidimensional evaluation approach, incorporating both quality metrics and efficiency considerations, offers a more holistic view of system performance than traditional evaluations focused solely on output quality.

Several promising directions for future research emerge from this work. First, exploring adaptive configuration approaches that dynamically adjust parameters based on query characteristics and system conditions could enhance both performance and efficiency. Second, investigating the impact of emerging embedding models and retrieval techniques on the identified trade-offs could reveal new optimization opportunities. Third, extending this evaluation framework to domain-specific applications would provide more targeted guidance for practitioners in specialized fields.

In conclusion, this research contributes to a more nuanced understanding of RAG system optimization that balances quality, efficiency, and reliability considerations.

By illuminating the complex interplay between different configuration parameters and their combined effects on system outcomes, we provide practical insights for implementing effective RAG systems across diverse application domains. As these systems continue to evolve and find new applications, the optimization principles and evaluation methodologies presented in this work will remain valuable guides for both researchers and practitioners in the field. The presented framework provides a foundation for standardized RAG benchmarking under transparent, configurable conditions. Beyond quantitative optimization, future work should expand toward responsible deployment, integrating privacy-aware retrieval, open-source evaluation pipelines, and automated hyperparameter optimization to ensure both reproducibility and ethical compliance.

Acknowledgements The authors would like to thank Prince Sultan University and Alfaisal University for their support.
GPT-4, Claude 3.5, and Gemini 3 were used solely to enhance the clarity and phrasing of the manuscript, and all content was thoroughly reviewed and validated by the authors to ensure accuracy and integrity.

References

1. Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B. et al.: A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions. arXiv preprint arXiv:2311.05232 (2023)
2. Zhang, Y., Li, Y., Cui, L., Cai, D., Liu, L., Fu, T., Huang, X. et al.: Siren's song in the AI ocean: a survey on hallucination in large language models. arXiv preprint arXiv:2309.01219 (2023)
3. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. ACM Comput. Surv. **55**(12) (2023)
4. Ammar, A., Koubaa, A., Benjdira, B., Nacar, O., Sibaee, S.: Prediction of Arabic legal rulings using large language models. Electronics **13**(4), 764 (2024)
5. Koubaa, A., Ammar, A., Boulila, W.: Next-generation human-robot interaction with chatgpt and robot operating system. Softw.: Pract. Exp. **55**(2), 355–382 (2025)
6. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-T., Rocktäschel, T., Riedel, S.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474 (2020)
7. Yan, S.-Q., Gu, J.-C., Zhu, Y., Ling, Z.-H.: Corrective retrieval augmented generation (2024). [Online]. https://openreview.net/forum?id=JnWJbrnaUE
8. Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., Liu, Z.: Evaluation of retrieval-augmented generation: a survey. In: Zhu, W., et al. (eds.) Big Data. BigData 2024. Communications in Computer and Information Science, vol. 2301, pp. 102–120. Springer, Singapore (2025)
9. Es, S., James, J., Anke, L.E., Schockaert, S.: Ragas: automated evaluation of retrieval augmented generation. In: Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, pp. 150–158 (2024)
10. Saad-Falcon, J., Khattab, O., Potts, C., Zaharia, M.: Ares: an automated evaluation framework for retrieval-augmented generation systems. arXiv preprint arXiv:2311.09476 (2023)
11. Chen, J., Lin, H., Han, X., Sun, L.: Benchmarking large language models in retrieval-augmented generation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, no. 16, pp. 17 754–17 762 (2024)

12. Tang, Y., Yang, Y.: Multihop-rag: benchmarking retrieval-augmented generation for multi-hop queries. arXiv preprint arXiv:2401.15391 (2024)
13. Hofstätter, S., Chen, J., Raman, K., Zamani, H.: Fid-light: efficient and effective retrieval-augmented text generation. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1437–1447 (2023)
14. LangChain, Evaluating rag architectures on benchmark tasks. https://langchain-ai.github.io/langchain-benchmarks/notebooks/retrieval/comparing_techniques.html (2023). Accessed: 2025-04-03
15. Leng, Q., Portes, J., Havens, S., Zaharia, M., Carbin, M.: Long context rag performance of large language models. arXiv preprint arXiv:2411.03538 (2024)
16. Barnett, S., Kurniawan, S., Thudumu, S., Brannelly, Z., Abdelrazek, M.: Seven failure points when engineering a retrieval augmented generation system. In: Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI, pp. 194–199 (2024)
17. Cuconasu, F., Trappolini, G., Siciliano, F., Filice, S., Campagnano, C., Maarek, Y., Tonellotto, N., Silvestri, F.: The power of noise: redefining retrieval for rag systems. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 719–729 (2024)
18. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data **7**(3), 535–547 (2019)
19. Panickssery, A., Bowman, S.R., Feng, S.: Llm evaluators recognize and favor their own generations. arXiv preprint arXiv:2404.13076 (2024)
20. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S. et al.: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
21. Rag model benchmark dataset. https://smith.langchain.com/public/62557875-8c4d-4bba-a905-37851309b87e/d, accessed: 2025-02-06
22. Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., Jégou, H.: The faiss library. arXiv preprint arXiv:2401.08281 (2024)
23. Xie, X., Liu, H., Hou, W., Huang, H.: A brief survey of vector databases. In: 2023 9th International Conference on Big Data and Information Analytics (BigDIA), pp. 364–371. IEEE (2023)

Culturally Aware LLM-Based Framework for Cardiometabolic Healthcare: The MENA Region as a Case Study

Samia Kouki, Nacim Yanes, Heba Khalil, and Nor Shahriza Abdul Karim

Abstract This chapter explores the role of culturally adaptive artificial intelligence (AI) and large language models (LLMs) in addressing cardiometabolic health issues within the MENA region. Cardiometabolic diseases, including diabetes and cardiovascular disorders, are prevalent in many MENA countries, posing significant public health challenges. The chapter begins by outlining the unique cultural and linguistic characteristics of the region that influence health behaviors and perceptions. It emphasizes the importance of developing AI and LLM tools that are not only technologically advanced but also culturally relevant and sensitive to local norms and values. This chapter reviews current advancements in AI and their applications in healthcare, with a focus on successful case studies from the MENA region. Moving beyond descriptive analysis, it proposes concrete recommendations and best practices for developing Clinical Decision Support Systems (CDSS) based on LLMs (LLM-CDSS). These systems are envisioned to be grounded in multimodal data that integrates electronic health records, clinical notes, and culturally relevant information such as language, dietary patterns, and religious practices. By outlining strategies for personalized health interventions, the chapter emphasizes community engagement and co-design, ensuring that AI solutions reflect the lived realities of local populations. Ethical considerations are addressed through alignment with cultural and religious values, reinforcing trust and adoption. Ultimately, the chapter argues that culturally adaptive AI and LLMs-based systems can substantially improve healthcare delivery and

S. Kouki (✉)
Higher Colleges of Technology, CIS Division, Ras Al Khaimah, UAE
e-mail: skouki@hct.ac.ae

N. Yanes
University of Gabes, ISGGB, Gabes, Tunisia

University of Manouba, ENSI, RIADI (LR99ES26), Manouba, Tunisia

H. Khalil
University of Sharjah, Faculty of Health Sciences, Sharjah, UAE

N. S. A. Karim
Prince Sultan University, KSA, Riyadh, Saudi Arabia

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027

A. Koubaa et al. (eds.), *Generative AI and Large Language Models: Opportunities, Challenges, and Applications*, Studies in Computational Intelligence 1287,
https://doi.org/10.1007/978-3-032-24013-2_10

outcomes for cardiometabolic conditions in the MENA region, offering a roadmap for building equitable, effective, and context-sensitive health technologies.

Keywords Culturally adaptive AI · Large language models (LLMs) · Cardiometabolic diseases · MENA region · Linguistic diversity · Health behaviors · Public health

1 Introduction

Cardiometabolic diseases, including diabetes, cardiovascular disorders, hypertension, and related conditions, are among the leading causes of morbidity and mortality across the MENA region. Rising rates of obesity, insulin resistance, and heart disease, combined with diabetes prevalence well above global averages, generate substantial clinical and economic burdens. These challenges are shaped not only by biological and lifestyle risk factors but also by cultural, linguistic, and religious dynamics that influence health behaviors, care-seeking practices, and adherence to treatment.

This chapter examines how culturally adaptive AI and LLMs can help address these complex, interconnected challenges. It reviews the epidemiological landscape of cardiometabolic health in the MENA region and highlights structural barriers, including health literacy gaps, limited access to advanced digital tools, and fragmented data ecosystems, that constrain effective prevention and treatment. Building on this foundation, the chapter outlines strategies for developing LLMs trained on multimodal datasets integrating electronic health records, diagnostic images, clinical notes, and culturally grounded information such as dietary norms, gender-specific care pathways, and religious practices.

Recent advances in AI, including global models such as GPT-5 [1] and Med-PaLM [2], along with regional initiatives like BiMediX2 [3] and MedArabiQ [4] demonstrate the potential of culturally and linguistically tailored systems to enhance patient education, clinical reasoning, and care delivery. However, realizing this potential requires deliberate cultural adaptation to ensure that AI tools meaningfully reflect the lived realities of diverse MENA populations.

By combining epidemiological insights, technological innovation, and cultural context, this chapter presents a roadmap for building equitable and clinically effective AI tools. It concludes with actionable recommendations on multimodal dataset development, governance frameworks, and clinical evaluation protocols, offering guidance for policymakers, researchers, and healthcare professionals committed to reducing the burden of cardiometabolic disease in the region.

2 Overview of Cardiometabolic Health Issues in the MENA Region

2.1 Cardiometabolic Health: Definition and Global Significance

Cardiometabolic health refers to the proper functioning of the cardiovascular and metabolic systems, characterized by the absence of major risk factors like central obesity, insulin resistance, hyperglycemia, dyslipidemia, and hypertension [5, 6]. It directly impacts quality of life by supporting physical function, mobility, and psychosocial well-being while reducing fatigue, depression, and other health limitations [7–9]. Maintaining this health is essential for preventing serious chronic diseases and reducing the global burden of noncommunicable diseases, which remain the leading cause of death worldwide [10, 11].

Maintaining optimal cardiometabolic health is essential not only for individual longevity and well-being but also for reducing chronic disease-associated societal and economic burdens. Poor cardiometabolic health substantially increases the risk of type 2 diabetes mellitus (T2DM), atherosclerotic cardiovascular disease, non-alcoholic fatty liver disease, chronic kidney disease, and cerebrovascular complications [12, 13]. At the population level, inadequate cardiometabolic health is associated with a rise in hospitalizations, increased healthcare costs, and growing disparities in disease outcomes, underscoring its influence on both health equity and system sustainability [13]. Cardiometabolic disorders have significant economic consequences, with cardiovascular illnesses accounting for more than USD 1 trillion in medical expenses, lost productivity, and disability each year, along with diabetes contributing an estimated USD 966 billion in healthcare costs [14]. Therefore, promoting cardiometabolic health alleviates pressure on healthcare systems, reduces hospitalizations, and supports equitable outcomes across populations [15].

2.2 Prevalence and Determinants of Cardiometabolic Disease in the MENA Region

Cardiometabolic diseases are increasingly prevalent across the Middle East and North Africa, representing a growing public health challenge that exceeds global averages. In 2019, ischemic heart disease affected approximately 2.55 million people, with an age-standardized incidence rate of 613.87 per 100,000, demonstrating a high cardiovascular risk [16]. The overall burden of cardiovascular disorders has increased significantly, with a 140.9% increase in cases between 1990 and 2019, while the age-standardized prevalence decreased slightly by 1.3% due to population growth and aging [17]. Patients with stroke frequently present with multiple comorbidities, including hypertension and diabetes, in 66% and 38% of cases, respectively [18],

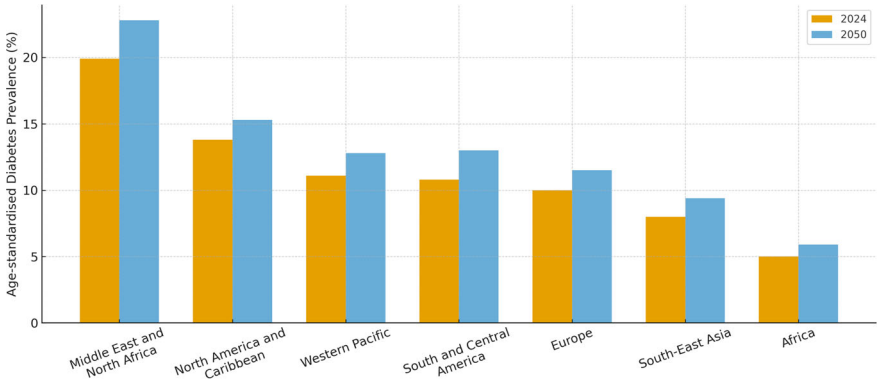


Fig. 1 Age-Standardized diabetes prevalence by global region in 2024 and 2050 [23]

emphasizing the clustering of cardiometabolic risk factors. In the MENA region, type 2 diabetes mellitus (T2DM) affects nearly 37 million individuals, representing a prevalence rate of 9.7% [19, 20]. The prevalence of gestational diabetes mellitus is also rising, with a pooled prevalence of 13.0% and peaks as high as 20.7% in Qatar and 14.7% across GCC countries [21]. These risks are exacerbated by obesity, which affects more than 30% of adults in countries such as Egypt, Qatar, and Bahrain, with childhood obesity having reached an alarming epidemic level [22].

To highlight the scale of this regional challenge, Fig. 1 compares age-standardized diabetes prevalence across major global regions for 2024 and projected levels for 2050. Figure 1 reveals a consistent rise worldwide, with the MENA region showing the highest prevalence in both periods. North America, the Western Pacific, and South & Central America also exhibit high and increasing rates, while Europe, South-East Asia, and Africa present lower but steadily rising trends. Overall, the figure underscores the global intensification of diabetes, with MENA remaining the most affected region [23].

To further contextualize the heightened burden observed in the MENA region, Fig. 2 presents a comparative analysis of three key population-level health indicators in 2017: the Sociodemographic Index (SDI), High Body Mass Index (BMI), and High Fasting Plasma Glucose (FPG), all expressed as Observed-to-Expected (O/E) ratios. The figure examines eight selected Middle Eastern and North African countries, Kuwait, the United Arab Emirates (UAE), Saudi Arabia, Bahrain, Jordan, Tunisia, Algeria, and Egypt, showing consistently high values across both metabolic risk indicators (BMI and FPG). These elevated O/E ratios underscore the substantial cardiometabolic risk facing the region [24].

The determinants of cardiometabolic disease in MENA are multifactorial, driven by a complex interplay of lifestyle, dietary, environmental, genetic, and socio-cultural factors. Rapid urbanization and lifestyle transitions in the MENA region have driven a shift from traditional Mediterranean-style diets, rich in fruits, vegetables, whole grains, and olive oil, to Westernized dietary patterns dominated by calorie-dense,

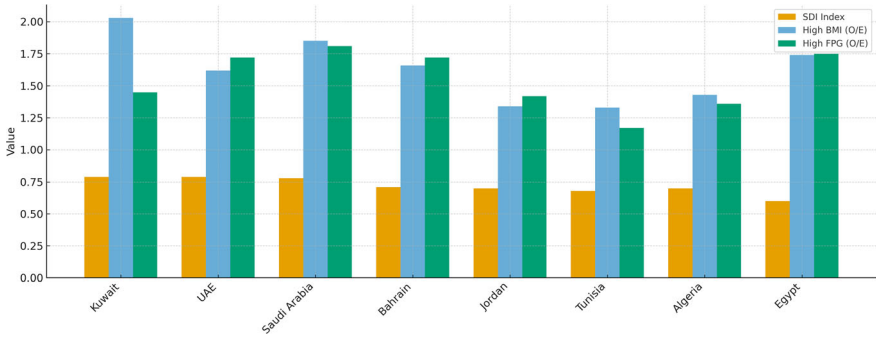


Fig. 2 Metabolic risk exposures and sociodemographic index in selected MENA countries [24]

processed foods. This nutritional transition, coupled with economic development that facilitates easy access to high-calorie foods and limits opportunities for physical activity, has contributed to rising rates of obesity, insulin resistance, and hypertension [25–28]. Sedentary behavior is increasingly prevalent, and prolonged screen time coupled with low levels of overall physical activity is strongly associated with a range of cardiometabolic disorders, even among individuals engaging in moderate-intensity exercise [29].

Smoking, including waterpipe use (also known as shisha or hookah), remains a major public health concern in the MENA region, with the highest prevalence observed in Eastern Mediterranean countries [30, 31]. In Lebanon, cigarette smoking affects 48.6% of males and 21.5% of females, whereas in Jordan, prevalence reaches 50.4% among males and 9.1% among females. Palestine reports similarly high rates of cigarette smoking, with 53.4% of males and 3.1% of females [32]. Epidemiological evidence suggests a strong association between cardiovascular disease and cigarette and waterpipe use, exposing individuals to elevated levels of carbon monoxide, nicotine, and other toxicants [33]. These exposures contribute to increased heart rate, elevated blood pressure, and endothelial dysfunction, all of which are risk factors for various cardiometabolic disorders [34].

Cultural and gender-specific norms in the Middle East and North Africa region significantly shape health behaviors and influence susceptibility to chronic disease. Women in the MENA region often encounter restrictions on outdoor activities due to cultural expectations, family criticism, and limited access to appropriate sports facilities [35]. Added responsibilities related to childcare and household duties further reinforce sedentary lifestyles [36, 37]. These constraints are associated with poorer sleep quality, greater insomnia severity, and higher stress levels, alongside increased psychosocial and nutritional support needs compared to men [38]. By contrast, men are generally more physically active and socially engaged; however, they often demonstrate lower adherence to recommended dietary practices [38, 39]. Despite these differences, men exhibit a higher prevalence of ischemic heart disease and myocardial infarction than women, reflecting the complex interplay of behavioral, biological, and sociocultural risk factors.

Cultural and religious beliefs in the MENA region exert complex, dual effects on the management of cardiometabolic conditions. Religion and spirituality are commonly employed as coping mechanisms for diabetes and other high-risk cardiometabolic disorders and have been shown to enhance disease acceptance and promote adherence to self-care behaviors [40, 41]. Spirituality and religiosity may support diabetes self-management by fostering motivation, providing coping strategies, facilitating social support, and guiding the adoption of healthy behaviors [42]. Conversely, certain cultural beliefs and social norms can delay engagement with formal healthcare services. Additionally, stigma surrounding conditions such as obesity or diabetes, coupled with the conspicuous use of monitoring devices like glucometers, may discourage individuals from seeking care or fully adhering to recommended interventions [43].

Dietary practices in the MENA region present both challenges and opportunities for cardiometabolic health. Religious and social traditions, particularly Ramadan, create unique considerations for managing conditions such as diabetes, hypertension, and other cardiometabolic disorders. While fasting during Ramadan has been associated with potential metabolic benefits, such as improved insulin sensitivity and favorable lipid profiles in some individuals, these benefits depend upon careful dietary and lifestyle management [44]. The daily fast from dawn to sunset is typically followed by evening meals that are often rich in calories, refined carbohydrates, and sugars. These shifts in eating patterns, combined with changes in sleep schedules and physical activity, can lead to fluctuations in blood glucose, dehydration, and increased cardiovascular risk if not carefully monitored and managed [14, 45, 46]. Simultaneously, culturally tailored dietary strategies can improve cardiometabolic outcomes. Structured education on meal planning during Ramadan enhances adherence, while promoting traditional diets rich in legumes, olive oil, and fresh produce, together with community-based nutrition initiatives adapted to local contexts, supports effective glycemic control [46–48].

Genetic factors further contribute to the burden of cardiometabolic and metabolic disorders in the MENA region. Consanguinity, particularly first-cousin unions, is deeply embedded within many countries' cultural and social structures. Rates of consanguineous marriages vary across the region, with 28.46%, 54%, and > 50% reported in Morocco, Qatar, and Saudi Arabia, respectively [49, 50]. Such unions increase homozygosity in offspring, elevating the risk of autosomal recessive conditions, including inherited metabolic disorders (IMDs), that can predispose individuals to obesity, insulin resistance, dyslipidemia, and other cardiometabolic abnormalities [51, 52]. Compounding these biological and social influences, low health literacy, misconceptions about disease causation, and inconsistent adherence to preventive behaviors delay early detection and effective self-management [53]. These factors show that cardiometabolic risk arises from the interaction of environmental, cultural, and individual determinants. Integrating these dimensions supports earlier risk detection, more personalized and culturally tailored interventions, and improved adherence and health outcomes.

2.3 Current Healthcare Challenges in the Middle East and North Africa Region

The MENA region continues to face substantial challenges in healthcare infrastructure and economic capacity, which impede the effective prevention, diagnosis, and management of cardiometabolic disorders [54]. Access to advanced medical technologies and specialized treatments remains particularly limited in low- and middle-income countries [55]. Socioeconomic barriers, including poverty and low household income, further constrain individuals' ability to seek care and adhere to treatment regimens. In areas with high costs of healthcare services and medications, this often leads to delayed diagnoses and suboptimal management [56]. Moreover, the insufficient availability of adequately equipped public health facilities exacerbates these challenges, compromising the delivery of quality care [57].

Limited health literacy is a key determinant of suboptimal cardiometabolic outcomes in the MENA region. Many individuals lack awareness and understanding of cardiometabolic diseases and their risk factors, including the influence of lifestyle behaviors such as diet and physical activity on conditions such as diabetes and hypertension [57]. This knowledge gap contributes to delayed diagnosis, inadequate disease management, and poor treatment protocol adherence [54, 58]. Targeted educational initiatives for both patients and healthcare providers, including structured training programs and user-friendly educational tools, are essential for improving health-seeking behaviors and clinical outcomes [59]. Fragmented health information systems further impede effective data collection, disease monitoring, and population-level management of cardiometabolic disorders [60, 61]. Despite high mobile phone and internet coverage, the use and adoption of digital health and telehealth solutions in the region remain limited [62]. These technologies have considerable potential to bridge care gaps, particularly in remote and underserved areas, but require further development, integration, and culturally adapted implementation to optimize their impact on cardiometabolic health [63].

The elevated prevalence of cardiometabolic disorders in the MENA region cannot be attributed solely to shortcomings in healthcare infrastructure or economic constraints; it is also deeply rooted in cultural determinants, including established dietary practices, religious beliefs, lifestyle behaviors, and social norms that shape health outcomes. Cultural and social norms strongly influence health behaviors and decision-making, with family-centered structures, gender dynamics, and religious traditions, such as Ramadan fasting, playing a central role in shaping treatment adherence and perceptions of care [64–66].

These culturally embedded factors perpetuate the burden of disease and limit the impact of conventional interventions, underscoring the need for strategies that are both context-specific and culturally adaptive. Emerging technologies, particularly AI and LLMs, offer considerable potential to address these challenges by integrating clinical, behavioral, and cultural data to support early detection, personalized risk assessment, patient education, and sustained self-management, even in resource-limited settings [67, 68]. When tailored to reflect the values, practices, and

needs of local populations, these technologies can enhance adherence, strengthen health literacy, and reduce the growing burden of cardiometabolic disease [69, 70]. Furthermore, culturally and linguistically adapted AI- and LLM-based solutions have the capacity to improve accessibility, ensure contextual relevance, and foster active patient engagement, thereby bridging care gaps and empowering individuals to play a proactive role in managing their health [71, 72]. Building on this potential, AI and LLMs can complement culturally tailored nutrition education by delivering personalized dietary guidance, real-time monitoring, and adaptive feedback during fasting periods such as Ramadan, thereby enhancing adherence, optimizing metabolic outcomes, and supporting long-term cardiometabolic health in the MENA population [73].

Developing culturally adaptive AI for the MENA region requires addressing linguistic diversity, cultural norms, and ethical considerations. The region's linguistic landscape, characterized by Modern Standard Arabic, diverse dialects, and frequent code-switching with French, English, and Berber, poses major challenges for natural language processing. These complexities demand LLMs capable of handling dialectal variation and context-sensitive meaning to ensure inclusivity and accessibility [74]. Recent progress in dialect identification and integrating dialectal corpora into NLP pipelines has improved recognition and contextual relevance, but significant gaps remain [75]. Equally important are ethical safeguards. Protecting sensitive health data, ensuring fairness in algorithmic outputs, and providing transparent, explainable decision-making are essential for building trust [76, 77]. Over-reliance on Western-centric datasets can introduce bias and compromise the accuracy of AI recommendations for MENA populations, potentially widening health disparities. Integrating ethical, cultural, and religious considerations into AI design is therefore crucial not only for equitable outcomes but also for fostering trust and encouraging widespread adoption [78, 79]. To be effective in the MENA region, AI and LLM-based health interventions must be both technologically sophisticated and culturally responsive, requiring careful consideration of linguistic diversity, social norms, and ethical standards. Integrating local practices and values into AI tools is essential, as culturally misaligned recommendations can undermine effectiveness, weaken trust, reduce adherence, and ultimately limit their impact on health outcomes [79].

3 Adoption of AI and LLMs for Healthcare in the MENA Region

3.1 State of the Art

AI in healthcare has advanced significantly over the past decade, with LLMs shifting attention away from earlier rule-based systems. Unlike rigid expert systems, LLMs trained on large corpora offer flexible tools for medical knowledge retrieval and patient interaction. Studies show that models like GPT-5 [1], Med-PaLM [2],

and BioGPT [81] can perform at or above board-exam levels, supporting clinical reasoning and education [82].

In the MENA region, research has emphasized linguistic and cultural adaptation. Arabic medical corpora have historically been underrepresented, limiting the usefulness of global models. Recent contributions such as BiMediX2 [3] and MedArabiQ [4] address this gap by training bilingual models and creating benchmarks for Arabic healthcare tasks. These initiatives mark important steps toward culturally and linguistically relevant AI systems.

Beyond text, multimodal models are gaining traction. Reviews highlight their potential in radiology, oncology, and pathology by combining imaging with text-based reasoning [83]. Frameworks like PeFoMed [84] reduce the resource demands of fine-tuning, offering practical solutions for regions where large-scale training may be costly.

Applications across MENA healthcare include clinical decision support, radiology, mental health, and administrative tasks. For example, LLMs in Saudi Arabia showed potential in oral lesion diagnosis but struggled with ambiguous cases [85]. Patient studies reveal cautious acceptance of AI, with a preference for physician oversight [85]. Arabic datasets such as MAQA [87] and MentalQA [88] further expand educational and patient-facing applications, particularly in mental health.

Administrative applications, like automated notetaking and discharge summaries, could reduce burdens in systems facing workforce shortages. However, barriers such as uneven adoption of Electronic Health Records (EHRs) and data privacy concerns remain significant [89, 90].

Cultural adaptation is a recurring theme. Healthcare in MENA is influenced by family-centered decision-making and religious norms [91]. Health literacy challenges also demand patient-friendly, culturally sensitive explanations [92]. Developing local datasets and governance structures, while addressing privacy and bias, is key to ensuring equitable adoption [87, 88]

3.2 Cultural Considerations in AI Development for the MENA Region

Culturally adaptive AI development requires an understanding that healthcare decisions are shaped not only by clinical evidence but also by linguistic accessibility, cultural practices, and trust in medical systems. Previous studies highlight that language heterogeneity, social norms, and ethical values must be embedded in AI design to ensure inclusivity [74, 80]. In MENA health contexts particularly for cardiometabolic conditions such as diabetes and cardiovascular disease patients' engagement with AI-driven tools is deeply intertwined with family involvement, religious obligations, and cultural expectations. These dynamics necessitate AI solutions that go beyond universal algorithms, creating systems that are responsive to local realities.

Language and Dialect Diversity. One of the most pressing challenges in developing AI for MENA healthcare is the linguistic landscape. Arabic is a diglossic language, with Modern Standard Arabic (MSA) coexisting alongside numerous dialects, such as Maghrebi, Gulf, and Levantine Arabic. Everyday communication often involves code-switching between dialects and colonial or global languages, such as French, English, and Berber. This multilingualism complicates data preparation, model training, and evaluation, especially given the lack of large, annotated datasets for Arabic dialects [74].

The integration of dialectal corpora into NLP pipelines has narrowed the gap between MSA and spoken dialects, enabling AI systems to generate contextually appropriate and culturally sensitive outputs [93]. These linguistic challenges are not theoretical; they directly affect real clinical interactions. For example, a patient in Morocco may describe hypoglycemia symptoms using the Maghrebi dialect, while a patient in Kuwait may describe the same experience using Gulf Arabic. Without dialect-aware NLP models, AI systems may misinterpret these expressions or fail to recognize that both refer to low blood sugar.

Cultural Norms and Values. Healthcare behaviors in the MENA region are strongly influenced by sociocultural norms, including family-centered decision-making, religious practices, and gender dynamics. Clinical decisions are often shaped by relatives rather than individual patients, with evidence showing that family influence can override patient preferences, particularly for women facing serious illnesses. Religious practices such as Ramadan fasting significantly affect chronic disease management, especially diabetes, producing mixed clinical outcomes that require context-aware clinical support. Gender-related barriers further restrict women's timely access to care, contributing to delayed treatment and poorer health outcomes. Collectively, these factors highlight the necessity for adaptive and culturally informed healthcare systems capable of accommodating social hierarchies, religious observances, and gender-sensitive constraints. For instance, female patients may postpone cardiometabolic check-ups if no female clinician is available, requiring AI appointment tools to support gender-specific filtering. Likewise, lifestyle recommendations must reflect cultural constraints, such as preferring indoor or gender-segregated exercise spaces. A recent review further identified modesty, provider gender preference, and family involvement as systemic barriers [94]. These patterns highlight the need for AI interventions that are not only linguistically accurate but also socially sensitive respecting family involvement while ensuring inclusivity and equity.

Ethical Considerations. Finally, ethical concerns remain central to the deployment of AI in healthcare across the MENA region. Core issues such as privacy, bias, and transparency are globally relevant, but in MENA societies they take on additional significance due to cultural and religious sensitivities. Concerns about stigma and confidentiality are particularly acute in relation to conditions such as diabetes, cancer, and reproductive health, where disclosure of personal data may carry social consequences [95].

Algorithmic bias also represents a major challenge. AI models trained predominantly on Western datasets risk producing recommendations that are inaccurate

for MENA populations, reinforcing inequities in healthcare delivery [96]. Transparency and explainability are equally critical, as patients and healthcare providers may distrust systems that function as "black boxes" without clear accountability [97]. For AI to be trusted, it must align with local ethical frameworks, religious values, and community expectations. Ethical safeguards should therefore include not only universal principles of fairness and accountability but also culturally grounded practices, such as sensitivity to fasting, dietary restrictions, and gender roles. For instance, AI systems should avoid recommending pork-derived insulin or medications containing alcohol unless permissible alternatives are clearly communicated, ensuring compliance with religious expectations.

Ultimately, the integration of these multidimensional considerations, linguistic, cultural, and ethical, is not merely a technical requirement but a foundational commitment to equity. A culturally adaptive AI system, by design, works to mitigate the health disparities that disproportionately affect vulnerable groups within the MENA region. For example, by processing dialectal Arabic, it ensures health information is accessible to those with limited literacy in Modern Standard Arabic. By respecting gender norms and family structures, it can deliver tailored advice that empowers women within their social context, rather than excluding them. And by auditing algorithmic bias and incorporating local data, it prevents the reinforcement of existing inequities often found in models trained on Western populations. Therefore, an AI system that is truly responsive to the MENA context is inherently a tool for promoting more equitable healthcare outcomes, ensuring that the benefits of technological advancement are accessible to all segments of its diverse societies.

4 Proposed LLM-Based CDSS for Cardiometabolic Health

4.1 Contextual Foundations of the Proposed LLM-CDSS

Proposed framework. To develop culturally adaptive AI tools for cardiometabolic care in the MENA region, systems must be localized to the region's linguistic, social, and religious context. Within this context, the related work reviewed in the following section positions our study among existing efforts while identifying gaps in current systems. Although prior research has advanced understanding of Arabic NLP, sociocultural factors, and ethical AI, little attention has been paid to designing interventions that explicitly address the cultural and socioeconomic conditions driving the high prevalence of cardiometabolic diseases in the MENA region. Our work seeks to address this gap by exploring how AI tools can be adapted to these unique regional realities to enable more effective and culturally responsive healthcare solutions. The adaptation of culture and socioeconomic characteristics would enhance comprehension, trust, and adherence while reducing the risk of culturally mismatched recommendations.

The proposed approach introduces an LLM-CDSS that integrates patient-specific data with culturally, linguistically, and socioeconomically relevant features. Conventional CDSSs, which typically rely on single-modal inputs such as laboratory results, medical images, or textual records [98], are limited in their ability to capture the complexities of modern healthcare. In the MENA region, where cultural norms, socioeconomic conditions, and health behaviors significantly shape health outcomes, advancing toward multimodal integration is essential. By combining biomedical data with cultural and contextual information, next-generation CDSSs can generate clinically sound yet socially responsive recommendations [99].

1. The main innovation of our proposed framework is that it moves from this general principle to a concrete, implementable architecture. While [99] identifies the "what," our work provides the "how" through three key contributions:
2. A Detailed Multimodal Architecture: We propose a specific, layered technical architecture (Fig. 1) that operationalizes this integration, detailing the data flow from raw, multimodal inputs to culturally-grounded outputs.
3. A Mapped Taxonomy of Contextual Features: We provide a tangible taxonomy of culturally relevant features (Tables 1, 2, 3 and 4), specifying exactly which sociocultural signals to capture (e.g., "mentions of family," "Ramadan fasting status") and how an LLM can personalize outputs based on them.
4. An LLM-Centric Reasoning Engine: Our framework is specifically engineered for LLMs, introducing a "Contextual Feature Selection" layer that dynamically prioritizes the most relevant cultural or clinical features for a given situation.

Table 1 Culture-related features for multimodal LLM CDSS

Feature	Example signals	LLM personalization
Family/collectivist decision-making	Mentions of family	Suggest caregiver-inclusive plans
Stigma around obesity/ diabetes	Expressions of shame, hiding device use	Provide discreet monitoring tips, normalize condition
Gender norms limiting women's activity	Barriers to gyms, family criticism	Offer home-based/indoor plans, female-only options
Waterpipe & cigarette use	Tobacco type, frequency, social context	Deliver risk education, cessation support
Urbanization & sedentary lifestyle	Car-centric living, long commutes, screen time	Recommend micro-activities, sitting-break prompts
Consanguinity/genetic risk	Reports of cousin marriage, family history	Flag for clinician counseling (educational only)

Table 2 Social-related features for multimodal LLM CDSS

Feature	Example signals	LLM personalization
Diet shift to Westernized/ processed foods	Food diaries; grocery receipts; mentions of fast food/SSBs	Suggest culturally familiar, affordable swaps; portion guidance
Sedentary behavior & screen time	Self-report, device logs, "hours sitting" phrases	Set step/standing goals; screen-time breaks; posture prompts
Smoking prevalence & patterns by sex/age	Pack-years; waterpipe session duration; social cues	Tailor nicotine harm education; offer gender-sensitive scripts
Care access & affordability	Insurance/copay tier; distance/time to clinic; pharmacy access	Cost-aware med options; refill nudges; telehealth alternatives
Caregiving/household workload (esp. women)	Mentions of childcare/eldercare duties; time scarcity	Short, flexible routines; batch cooking; appointment timing tips
Sleep quality & stress	Insomnia terms; shift work; stress/ self-report scales	Sleep hygiene coaching; timing of meds/meals; mindfulness prompts

Table 3 Educational-Related Features for multimodal LLM CDSS

Feature	Example Signals	LLM Personalization
Health literacy level	Readability probes; Teach-Back indicators; education level	Set AR/EN reading level; add visuals/icons; minimize jargon
Misconceptions about disease/ meds	Statements reflecting myths (e.g., "insulin causes failure")	Provide myth-busting snippets with citations
Digital literacy & channel preference	WhatsApp/SMS/app/IVR usage; font/voice preferences	Choose channel/format (text, TTS audio, AR/EN) accordingly
Health literacy level	Readability probes; Teach-Back indicators; education level	Set AR/EN reading level; add visuals/icons; minimize jargon
Misconceptions about disease/ meds	Statements reflecting myths (e.g., "insulin causes failure")	Provide myth-busting snippets with citations

4.2 Contextual Feature Integration for Culturally Adaptive Decision Support

Multimodal LLMs can integrate diverse data sources, clinical records, labs, imaging, behavioral patterns, and socio-cultural information, into a unified framework,

Table 4 Religious-Related Features for multimodal LLM CDSS

Feature	Example signals	LLM personalization
Ramadan fasting intent & exemptions	Yes/no/conditional fasting; past hypoglycemia events	Generate safe fasting plans (meal timing, hydration)
Ramadan eating/sleep pattern shifts	Iftar/suhoor timing; evening rich meals; sleep disruption	Adjust medication info (education only), activity to cooler hours
Halal/gelatin and dietary rules	Ingredient avoidance; supplement/gelatin notes	Suggest halal-compliant brands/forms; alternative foods
Spiritual coping & community support	Mentions of prayer/spiritual motivation; mosque/ community groups	Leverage supportive framing; community-based programs
Ramadan fasting intent & exemptions	Yes/no/conditional fasting; past hypoglycemia events	Generate safe fasting plans (meal timing, hydration)

producing richer and more interpretable patient representations while overcoming the limitations of single-modality approaches [100]. Evidence across medical domains shows that multimodal approaches significantly improve the accuracy, robustness, and clinical relevance of AI/ML systems compared with unimodal models. For example, integrating imaging with electronic health records has been shown to improve diagnostic precision in oncology, while combining genomic data with clinical features has advanced risk prediction in cardiology [101]. Similarly, multimodal approaches have strengthened disease classification, prognosis modeling, and treatment recommendation across diverse clinical tasks [102].

The key contribution of this work is extending multimodal LLM-based healthcare systems to integrate not only biomedical data but also the broader contextual determinants that shape health outcomes in the MENA region. Existing systems typically focus on clinical records or diagnostics while overlooking socio-cultural and environmental factors. Our framework fills this gap by embedding cultural practices, religious beliefs, dietary habits, linguistic diversity, lifestyle patterns, and family-centered decision-making into the CDSS design. These elements are especially critical in MENA, where health decisions are strongly influenced by social norms, Ramadan fasting, gender dynamics, multilingual communication, and extended family networks. By incorporating these features, the system improves contextual relevance, trust, and patient adoption, supporting more equitable and culturally aligned AI-driven healthcare.

The tables below (Tables 1, 2, 3 and 4) present the key features recommended for inclusion in the proposed multimodal LLM, complementing conventional clinical datasets and organized into categories namely cultural, social, religious, and educational factors.

To clarify the integration of culture-related features within the LLM, Tables 1, 2, 3 and 4 explicitly map patient signals to their internal representations and corresponding personalized recommendations. Each cultural feature is incorporated

through a structured three-step process: (1) identification of a relevant cultural cue from patient narratives or structured inputs; (2) transformation of this cue into an internal representation, such as a control token, feature flag, or contextual embedding; and (3) conditioning of the LLM's reasoning process to generate culturally aligned outputs. This design enables the model to systematically account for factors including family-centered decision-making, stigma, gender norms, and tobacco use, rather than treating them as generic textual inputs.

Example—Family/Collectivist Decision-Making

- **Patient signal:** "My family helps decide my meals and medications."
- **LLM integration:** Activation of a *family_involvement* control token appended to the input embedding sequence.
- **Personalized output:** Caregiver-inclusive recommendations that support shared decision-making and provide guidance suitable for discussion with family members.

The proposed framework, which relies on structured feature integration, may invite comparison to a simpler alternative: using prompt engineering to inject cultural, social, and educational context into a fine-tuned, general-purpose generative model. While prompt-based methods are viable for prototyping, our integrated architecture is chosen for its superiority in building a robust CDSS. Primarily, it enhances clinical reliability and safety by explicitly providing structured, pre-validated cultural features as direct model inputs, thereby grounding the LLM's reasoning in concrete data and drastically reducing the risk of hallucination or incorrect assumptions about complex cultural nuances. Furthermore, our approach enables dynamic, data-driven personalization; instead of static prompts, our contextual feature selection layer can automatically detect and weight relevant factors, such as prioritizing fasting status during Ramadan, creating a responsive system that adapts to a patient's evolving context. Finally, this method is more scalable and efficient, systematically integrating dozens of complex, interrelated features without overburdening the model's context window. This ensures that all cultural and clinical data is coherently considered for every decision, moving beyond merely querying a model to engineering a culturally grounded clinical reasoning system.

4.3 Framework Architecture

The architecture of the proposed LLM-CDSS is organized into multiple layers, each responsible for data ingestion, preprocessing, semantic representation, reasoning, and user interaction. Together, they form an end-to-end pipeline that delivers actionable insights to clinicians, patients, and caregivers, with adaptability supported through continuous feedback.

Given its foundation in LLM-based reasoning, the proposed framework offers strong capabilities in semantic interpretation [103], contextual adaptation [104], and

personalized support [105]. While adversarial and diffusion methods can help address challenges such as data scarcity or domain shift, an LLM-first approach is more suitable for culturally adaptive, context-rich CDSS tasks, including patient education, report generation, diagnostic assistance, and personalized recommendations. This makes the LLM-centric design particularly well aligned with the needs and characteristics of MENA healthcare environments.

Figure 1 shows the main components and their interactions, and the following subsections describe each layer in detail (Fig. 3).

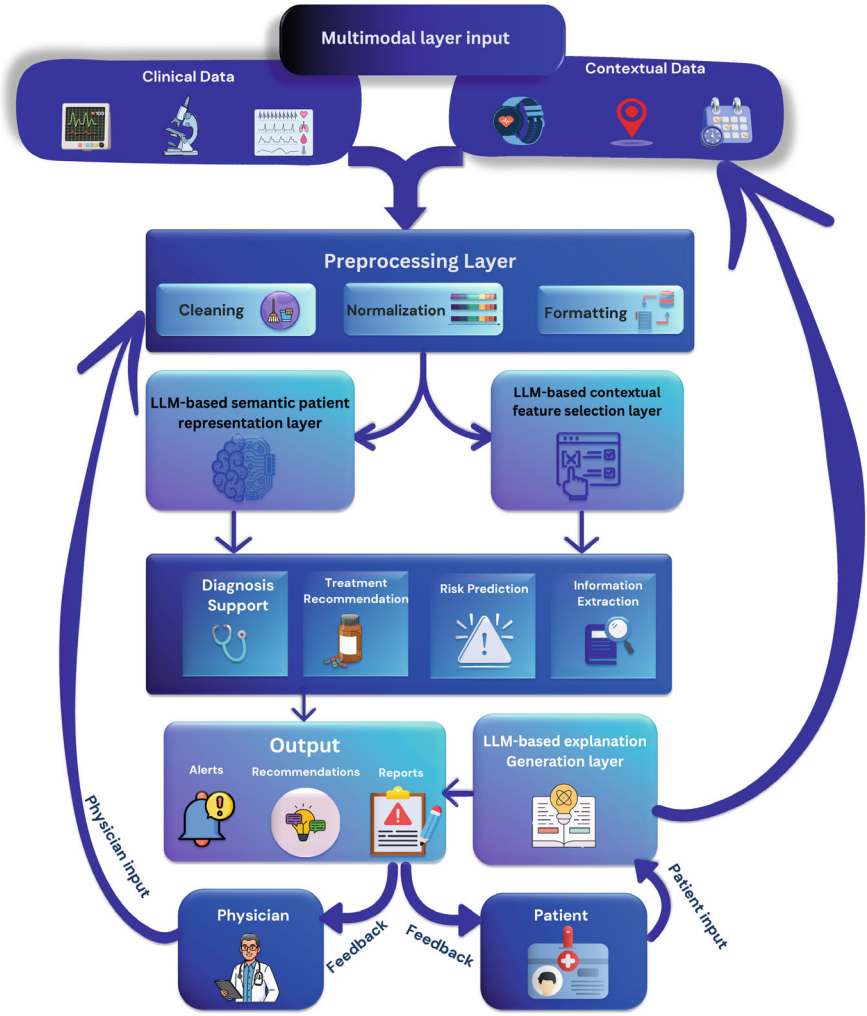


Fig. 3 Multimodal LLM-based clinical decision support system

Input Layer (Multimodal Data Sources). The Multimodal Input Layer in the LLM-CDSS is a sophisticated pipeline that standardizes and integrates diverse data sources into machine-readable vectors. By combining patient-related biomedical data with contextual socio-cultural data, the system enhances its ability to support clinical decision-making, ultimately leading to improved healthcare outcomes.

Clinical Data Stream. The Clinical Data Stream consolidates heterogeneous biomedical information required for reliable and reproducible diagnostic reasoning by integrating both **structured/high-dimensional** and **unstructured/sparse** data modalities. Structured inputs include clinical concepts, demographic attributes, diagnosis and procedure codes, medication lists, laboratory values, vital signs, physiological time-series signals (e.g., ECG, EEG, continuous monitoring), semi-structured fields, and high-dimensional genomic or molecular profiles. Each is encoded using the method best suited to its structure: ontology-based embeddings for clinical concepts; normalized numerical vectors for laboratory and physiological measurements; one-hot or low-dimensional learned embeddings for categorical attributes; and temporal encoders for sequential physiological signals. The framework also supports **medical imaging modalities**, including radiology, pathology, ultrasound, and retinal scans, which are processed using convolutional or vision transformer–based encoders to generate compact visual representations. On the unstructured side, clinical notes, patient messages, and other free-text records are encoded using transformer-based language models, while audio inputs from telemedicine interactions are transformed into acoustic feature embeddings and video data from digital phenotyping or teleconsultation are processed via visual feature extractors. Together, these modality-specific representations form a comprehensive multimodal input space that is passed to the Pre-processing Layer for alignment, normalization, and downstream fusion.

Figure 4 illustrates the Clinical Data Stream, which consolidates structured and unstructured inputs. Each modality is encoded through its dedicated module and then passed to the Pre-processing Layer to form a unified Multimodal State Vector for downstream decision support.

Contextual-related Data stream. This stream integrates research evidence, clinical guidelines, and socio-cultural factors, including cultural values, religious practices, language, dialect, and social context, using tailored input-layer techniques such as Retrieval-Augmented Generation, control tokens, and contextual embeddings (Fig. 5).

To incorporate research findings and evidence-based clinical guidelines, the system employs a Retrieval-Augmented Generation (RAG) architecture [106]. Clinical and regional guidelines are embedded using SciBERT [107] and indexed in a FAISS-HNSW vector store [108], enabling fast and accurate retrieval of biomedical evidence to ground the LLM's outputs. Patient messages, intake forms, and clinician notes are tokenized and encoded using MARBERT [109], a multilingual Arabic-capable transformer, to support semantic understanding and contextual cue detection. A fine-tuned MARBERT dialect classifier identifies the patient's dominant Arabic variety (e.g., Gulf, Maghrebi, Levantine, MSA, code-mixed), which is

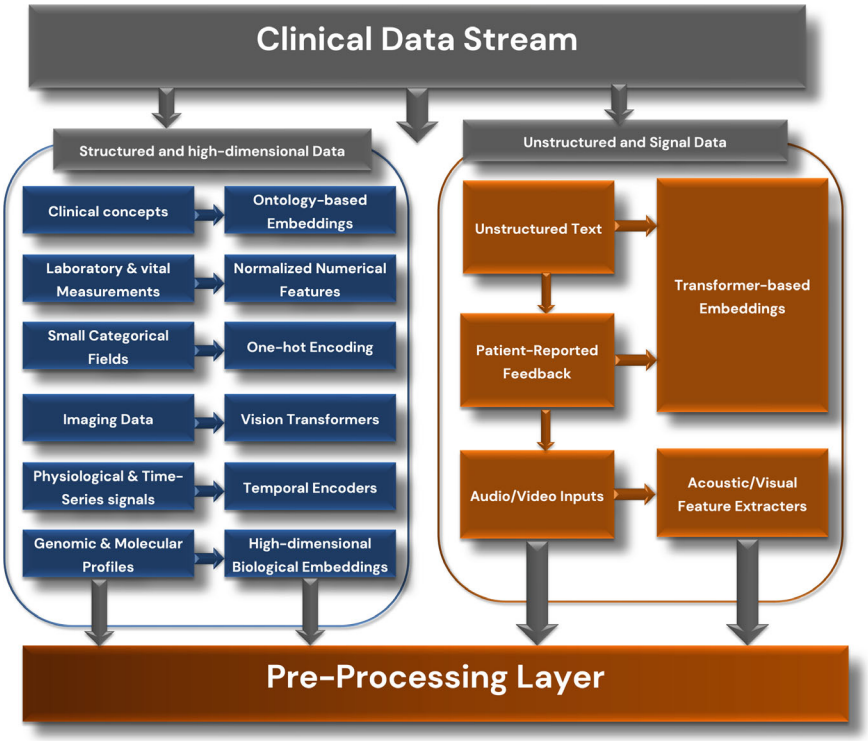


Fig. 4 Clinical *data stream*

encoded as a categorical feature and conveyed to the LLM through a dialect-specific control token (e.g., < DIALECT_GULF >).

Cultural values, language, and social context extracted from patient communication are converted into structured indicators that guide the CDSS in generating culturally aligned recommendations. Religious cues, such as "Ramadan," "iftar," "suhoor," "halal," and "haram", are detected through a combined lexicon- and pattern-based layer, while a fine-tuned DistilBERT classifier [110] determines whether fasting or halal-related constraints are active. These signals are encoded as structured flags (e.g., fasting_status = active, halal_sensitive = true) and injected into the LLM via control tokens such as < RAMADAN_FASTING > and < HALAL_RESTRICTION > .

Family involvement and collective decision-making are identified using dependency patterns and targeted rules (e.g., "my husband decides," "we decided as a family"), complemented by a lightweight classifier that assigns a family involvement score. This score (e.g., family_involvement = high) is passed to the LLM through a control token such as < FAMILY_HIGH > . Similarly, social constraints, including



Fig. 5 Contextual-related *data* stream

gender limitations, stigma, and mobility barriers, are detected using specialized lexicons (e.g., "mixed gym," "ashamed to inject," "cannot go alone"), and a small classifier (logistic regression or DistilBERT) estimates attributes such as stigma_level, gender_constraint, and mobility_restriction. These are then discretized and encoded using control tokens such as < STIGMA_MODERATE > and < GENDER_LIMIT_HIGH > .

Both clinical and cultural data streams undergo shared preprocessing, cleaning, normalization, and feature engineering, before being projected into a unified latent space. The resulting multimodal state vector fuses clinical embeddings, cultural embeddings, numerical features, and control tokens, giving the LLM a structured and culturally contextualized patient representation.

Preprocessing Layer. Architecturally, this block acts as a **multimodal embedding and fusion layer,** mapping each input modality; numerical labs, imaging-derived features, text, wearable signals, and contextual factors, into aligned embedding spaces before performing cross-modal attention and joint feature integration. This ensures that downstream LLM reasoning operates on a fully integrated, multimodal patient state. It comprises but not limited to the following techniques:

Data Cleaning and Validation Methods. The Data Cleaning and Validation stage removes duplicates, incorrect values, and structural inconsistencies from EHRs, clinical notes, and contextual datasets to prevent biased learning. Duplicate records are addressed using composite primary keys or probabilistic record-linkage techniques [111]. Range-based and temporal consistency validators are applied to detect implausible measurements and longitudinal discrepancies [112–114] Unit harmonization ensures standardized clinical measurements, such as glucose unit alignment, to support interoperability [115]. Anomaly detection methods, including clustering and deep density estimation, are used to identify sparse or erroneous data points that may distort model learning [116]. System-level validation frameworks further enforce internal and external consistency to maintain high data quality [117, 118]. Finally, standardized data models, such as HL7 FHIR and the OMOP Common Data Model, promote interoperability across sources and ensure unified data formatting [119, 120].

Data Filtering. Data Filtering is technically implemented as a Feature Engineering preprocessing module within the CDSS pipeline. Its primary function is to optimize the input fed to the LLM by creating a compact, high-quality feature vector relevant specifically to cardiometabolic risk and the immediate cultural context. This process involves filtering out unrelated clinical variables to prioritize relevant features such as blood glucose, BMI, dietary markers, and fasting-related data. The filtering techniques used in the CDSS pipeline are strategically differentiated based on the data type, Clinical Data requires statistical relevance, while Cultural Data requires contextual relevance and constraint enforcement.

- **Clinical Data:** For structured clinical data, such as lab results and BMI, the core technique used is Mutual Information (MI) Ranking. This technique is used to select the most predictive structured clinical features, such as BMI, fasting

blood glucose, and HbA1c, by measuring both linear and non-linear relationships with cardiometabolic outcomes. MI prioritizes variables with strong predictive power while removing redundant ones, ensuring that only the most relevant clinical features are used. This improves both the accuracy and efficiency of cardiometabolic prediction models.

- **Contextual Data:** The Semantic Annotation technique [121] is used to process the NLP output by applying an ontology or knowledge graph to standardize cultural concepts. It serves as a critical filtering layer that ensures all culturally specific expressions refer to the same underlying concept. For example, different dialectal phrases describing low blood sugar; such as Gulf Arabic "*naqs al-sukkar*," Levantine "*wateet sukkar*," or Tunisian "*tahbet sukkar*" are all mapped to the single standardized concept Hypoglycemia. This unified representation is essential for accurately conditioning the LLM on culturally relevant information.

Data Normalization. The normalization process is implemented as a dedicated preprocessing module that ensures two objectives: (1) Numerical stability for structured data, and (2) Linguistic consistency for unstructured text, which is vital for effective LLM tokenization and embedding.

It is achieved through two distinct technical processes:

- **Linguistic Normalization**: This process focuses on the cultural data stream by eliminating ambiguity in unstructured text. Techniques such as lexical lookup tables map colloquial terms (e.g., "tahbet li sukkar") to clinical equivalents, while non-Arabic insertions are converted to canonical forms.
- **Numerical Normalization**: Numerical normalization prepares structured clinical data by standardizing units (e.g., converting all glucose values to mg/dL) and applying scaling methods such as Min-Max or Z-score normalization. This ensures consistent ranges across variables and prevents high-magnitude features from disproportionately influencing the LLM's embedding space.

Table 5 presents some examples of linguistic and numerical normalizations:

Table 5 Examples of *data normalization*

Data type	Raw input	Normalization technique used	Normalized output (LLM Input)
Linguistic	"I feel like I have tahbet li sukkar." (Dialectal)	Dialectal/Lexical Standardization (Lookup Table)	"I feel like I have Hypoglycemia."
Numerical	BMI = 28.5 Systolic BP = 145	Min–Max Scaling (assuming max BMI = 50, max BP = 200)	BMI = 0.57 Systolic BP = 0.725
Numerical	Weight = 90 kgs Weight = 198 lbs	Unit Harmonization (Conversion Rule)	Weight = 90 kgs Weight = 90 kgs

Data Labeling. Annotates datasets with structured labels. Adds culturally relevant metadata, including Ramadan fasting status, waterpipe smoking patterns, or gender-specific lifestyle constraints. This step involves three structured steps to enhance the dataset's clinical and cultural applicability:

1. Define the Labels: A focused, fixed list of culturally critical metadata categories is established (e.g., Fasting Status, Waterpipe Smoking Pattern, and Gender-Specific Lifestyle Constraints), along with clear, discrete labels for each (e.g., Fasting during Ramadan, Women-only gyms).
2. Develop a Labeling Framework: Clear guidelines and training are established to ensure consistency among annotators. This step also involves implementing automated tools, such as Natural Language Processing (NLP) algorithms, to assist in automatically identifying keywords and phrases corresponding to the defined labels across large datasets.
3. Apply and Validate Labels: The defined labels are applied to the dataset using a combination of manual annotation (for nuanced cultural contexts) and automated methods. A final Quality Control process is implemented to review labeled subsets, ensuring accuracy and consistency before the metadata is used in the preprocessing layer.

Data Balancing. This component is applied when needed to correct class imbalance. Studies [122, 123] report medication nonadherence and disease progression, along with the consistent underrepresentation of migrants, women, and older adults in cardiovascular research, factors that contribute to limited and biased clinical datasets. To address this, the proposed framework integrates **Synthetic Minority Oversampling Technique (SMOTE)** within the Pre-Processing Layer's Data Balancing module. By generating synthetic minority samples, SMOTE improves the representation of clinical and culturally relevant features before they enter the LLM-based semantic representation layer.

Feature Engineering. This component creates new variables that blend clinical and cultural dimensions, for example, computing fasting duration based on religious calendars or constructing composite lifestyle risk scores that reflect diet, smoking habits, and sedentary behavior. In our framework, we adopt a semi-automated feature engineering approach that integrates algorithmic techniques with domain expertise to iteratively generate, evaluate, and refine features [124]. Automated feature generation employs genetic algorithms and meta-learning–based ranking to efficiently produce and prioritize large sets of candidate features, while human experts perform guided selection to identify the most relevant and meaningful features for the specific application. This iterative process ensures that the final feature set is both clinically grounded and culturally aligned.

Data Privacy. This component de-identifies patient data by removing or masking personal identifiers while preserving analytical value, ensuring compliance with the Health Insurance Portability and Accountability Act (HIPAA), the General Data Protection Regulation (GDPR), and MENA-specific data governance requirements.

It employs a combination of techniques, including identifier removal, pseudonymization, k-anonymity–based generalization and suppression, differential privacy for aggregated outputs, and strict encryption with role-based access control, to ensure that all data entering the modeling pipeline remain both privacy-preserving and analytically useful.

Both the Preprocessing Layer and the Representation & Reasoning Layer address feature relevance, but the former does so through statistical methods, while the latter refines importance semantically through LLM attention mechanisms.

Representation and Reasoning Layer. This stage functions as the bridge between raw, preprocessed multimodal data and the reasoning capabilities of the LLM. It ensures that patient information is semantically meaningful and that only the most culturally and clinically relevant features are prioritized for downstream tasks.

LLM-based semantic patient representation layer. This layer is designed to convert multimodal patient data into structured embeddings that retain semantic meaning. Textual data, such as clinical notes or patient-reported symptoms, is tokenized and embedded into high-dimensional vectors, while numerical and categorical data, including cholesterol levels, smoking type, or gender, are normalized and mapped into the same representational space. Importantly, cultural factors such as Ramadan fasting practices, dialectical language use, and family-centered decision-making are embedded alongside biomedical features. By unifying medical and cultural dimensions within a single semantic representation, this layer generates a holistic patient profile that enables the LLM to reason across both domains simultaneously, ensuring that downstream outputs are contextually grounded as well as clinically accurate.

Figure 6, summarizes the sequential steps within the representation Layer, showing how heterogeneous clinical, contextual, and linguistic inputs are encoded, fused, and prepared for downstream reasoning in the CDSS.

- ***Input Collection:*** *The system receives multiple data streams, including text, numerical clinical values, contextual metadata, and linguistic features.*
- ***Independent Encoding:*** *Each data type is processed through its own encoder to convert it into a structured, machine-readable representation.*
- ***Feature Fusion:*** *The encoded outputs are merged using a fusion mechanism—either an MLP or an attention-based model.*
- ***Unified Patient Profile Generation:*** *The fusion process produces a single integrated semantic profile that captures all modalities.*
- ***Downstream Processing:*** *This unified profile is then used by the CDSS for diagnosis, risk prediction, and personalized clinical recommendations.*

LLM-Based Contextual Feature Selection. The LLM-Based Contextual Feature Selection layer is responsible for identifying and emphasizing the clinical and cultural features that are most relevant for each patient and each task. It uses a combination of attention mechanisms, context-aware weighting, and simple rule-based filtering to ensure that the model focuses on the signals that truly matter while suppressing noise. In practice, this means that the system dynamically adjusts feature importance

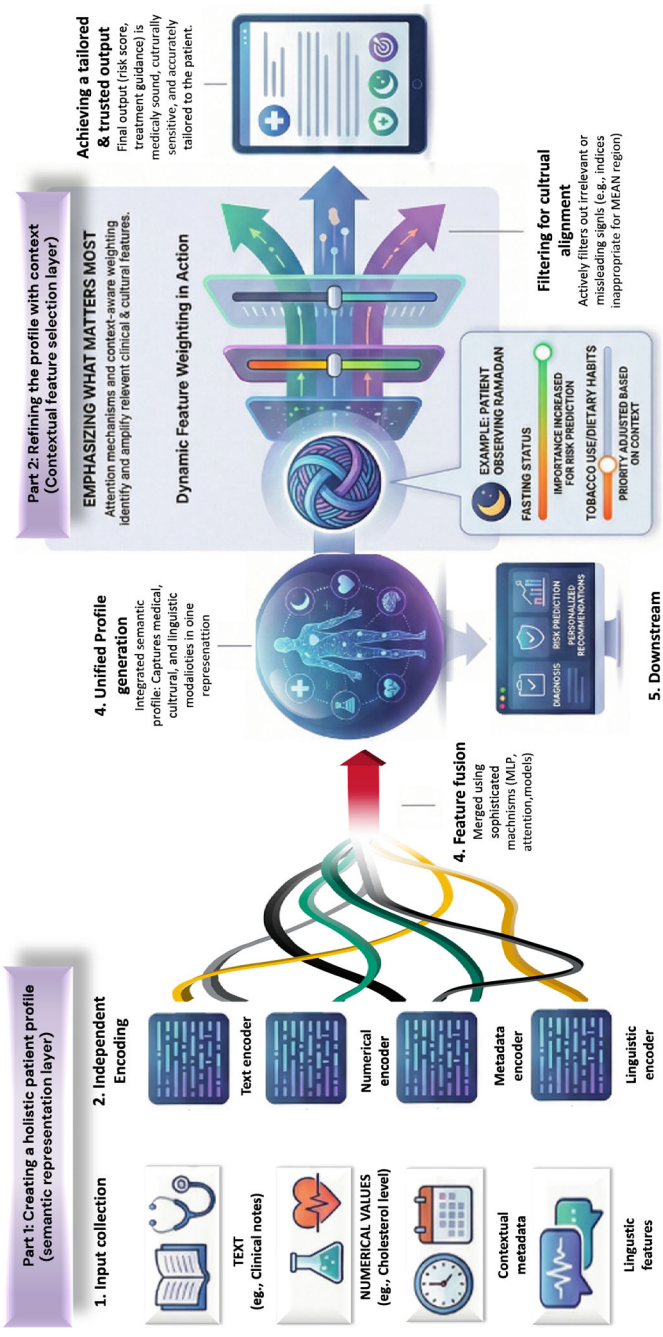


Fig. 6 LLM-based *semantic patient representation layer*

based on cultural, temporal, and behavioral context. For example, during Ramadan, fasting status and meal timing receive greater emphasis in risk prediction, while in non-fasting periods, factors such as tobacco use, sedentary behavior, or dietary habits may carry more weight.

To further improve cultural alignment, the layer filters out irrelevant patterns, such as Western-centric dietary indices that do not reflect eating habits in the MENA region and reduces reliance on weak or misleading signals. By combining learned attention with context-conditioned feature weighting, the layer strengthens the influence of the most clinically and culturally meaningful factors. As a result, all downstream outputs, whether recommendations, risk scores, or treatment guidance, remain medically sound, culturally sensitive, and accurately tailored to the lived realities of patients in the MENA region.

Output Layer. The Output Layer integrates predictive modeling, clinical rules, and structured language generation to produce safe, accurate, and culturally contextualized outputs tailored to the MENA region's healthcare needs.

The Output Layer forms the actionable interface of the framework, transforming multimodal and context-aware insights into meaningful guidance for clinicians, healthcare providers, and patients. Its role is to deliver recommendations that are both clinically valid and culturally adapted to the MENA context.

This layer is implemented using a hybrid architecture that combines neural prediction models, rule-based clinical logic, and controlled natural language generation (NLG):

1. **Neural Prediction Heads**
2. Task-specific neural heads operate on the LLM-derived patient embedding to produce calibrated outputs such as:

   - **Risk scores** (e.g., CMD risk, hypertension, complications) using small classification or regression heads.
   - **Alert signals** (e.g., "critical," "urgent," "routine") using binary or multi-label classifiers.
   - These models are trained end-to-end using SMOTE-balanced data to ensure accuracy and robustness.

3. **Rule-Based and Guideline Logic**
4. To ensure medical safety, model predictions are combined with a rule engine encoding clinical guidelines and culturally relevant constraints.
5. Examples include:

- Fasting-aware hypoglycemia alerts during Ramadan.
- Drug–disease contraindication rules.
- This hybrid approach ensures that alerts follow **evidence-based and culturally aligned** logic.

6. **Recommendation Generation**
7. Recommendations (medication, lifestyle, diet) are generated by mapping risk profiles and cultural attributes to predefined guideline templates.

8. Cultural factors, such as fasting status, gender-based access, waterpipe use, and local dietary patterns, automatically determine the appropriate variant of each recommendation.
9. **Natural Language Generation (NLG)**: Final messages are produced through **controlled, template-based NLG**, optionally refined by the LLM.
10. This prevents hallucination and ensures clarity while allowing dialect-aware, culturally sensitive phrasing.
11. **Calibration and Thresholding:** The system applies probability calibration methods (e.g., Platt scaling, isotonic regression) and clinically validated thresholds to ensure consistent and interpretable outputs for both alerts and recommendations.

Illustrative examples of system outputs are provided below; however, these examples are not exhaustive and do not limit the scope or extensibility of the proposed framework.

Alerts and Warnings. The system generates real-time alerts on critical risks such as sudden spikes in glucose levels, elevated cardiovascular risk scores, or contraindications in treatment during fasting periods like Ramadan. These warnings are not only clinical but also context-aware; for instance, a diabetic patient who is fasting might receive a tailored warning about adjusting insulin dosage to prevent hypoglycemia.

Recommendations. The framework provides personalized treatment guidelines covering medications, diets, physical activity, and lifestyle adjustments. Importantly, recommendations are culturally sensitive: dietary advice considers local food preferences (e.g., rice, bread, dates) and religious practices, while physical activity suggestions acknowledge gender-specific or community-based barriers. Clinical guidelines are adapted from both global evidence-based practices and region-specific medical protocols, ensuring medical rigor without neglecting cultural feasibility.

Risk Scores. The system outputs quantitative risk assessments, such as probabilities of developing type 2 diabetes, hypertension, or cardiovascular complications. These scores integrate biomedical variables with contextual features, meaning they capture not just clinical parameters but also cultural and behavioral risk factors (e.g., sedentary lifestyle norms, communal eating patterns, tobacco and waterpipe use).

Additional Outputs. Beyond clinical recommendations, the system can support patient education by delivering accessible explanations in the patient's dialect or preferred language. It may also offer preventive health prompts aligned with cultural calendars, such as reminders before Ramadan on safe dietary and medication practices.

LLM-Based Explanation Generation Layer. This layer is positioned after the output stage, which provides alerts, warnings, recommendations, and risk scores. Its primary function is to generate clear, interpretable, and culturally adapted explanations of the system's outputs.

For Physicians and Healthcare Providers. It delivers detailed, evidence-based justifications for diagnostic suggestions, treatment plans, or risk scores, enhancing clinical trust and accountability.

For Patients. It provides simplified, linguistically appropriate, and culturally sensitive explanations to improve health literacy and treatment adherence. For instance, advice can be presented in local dialects or framed in ways that respect religious practices, such as guidance on managing insulin use during Ramadan.

For Policymakers and Stakeholders. It ensures transparency, making the system's reasoning processes auditable and compliant with governance requirements.

By embedding this layer, the framework strengthens explainability, trust, and user engagement critical factors for adoption in healthcare systems where transparency and cultural resonance are essential.

To support systematic and transparent assessment of the proposed framework, we introduce an evaluation matrix that maps system tasks to their corresponding performance metrics and acceptance criteria. Such matrices are widely used in clinical decision support research and provide illustrative examples of how similar multimodal systems may be evaluated in practice. Representative metrics include AUROC, AUPRC, and NRI for predictive accuracy, decision latency for system efficiency, and CLIP-Score or retrieval R@K for assessing text–image or text–signal alignment.

For clarity, Table 6 presents a concise example demonstrating how these metrics correspond to typical CDSS-related tasks.

Positioning the Proposed Framework Within the Arabic Medical LLM Landscape. To the best of our knowledge, no existing Arabic-focused medical LLM system integrates cultural adaptation, multimodal data fusion, and clinical workflow alignment in the manner proposed by our culturally adaptive Multimodal LLM-CDSS. State-of-the-art systems such as BiMediX [3] and MedArabiQ [4] provide strong empirical evidence of performance through large-scale experimental evaluations and implemented features, particularly in linguistic benchmarking, translation quality, and Arabic medical task performance. However, their scope remains primarily confined to model evaluation and exam-style question answering, rather than supporting end-to-end clinical decision-making. By contrast, our proposed framework introduces a high-level yet robustly justified multimodal and culturally

Table 6 Example Evaluation Matrix for CDSS-Related Tasks

Task	Example Metrics	Typical Use
Predictive accuracy	AUROC, AUPRC	Assess risk prediction reliability
Clinical utility	NRI	Compare improvement over baseline workflow
System efficiency	Decision latency	Ensure near–real-time CDSS response
Multimodal alignment	CLIP-Score, Retrieval R@K	Validate consistency between text and image/signal inputs

Table 7 Comparative Analysis of the Proposed Culturally Adaptive LLM-CDSS and existing Arabic Medical LLM Systems

Feature	Proposed framework (This Work)	BiMediX	MedArabiQ
Language support	Multilingual with emphasis on Arabic dialects + MSA	Bilingual (English + Arabic)	Arabic only (MSA + limited dialect coverage)
Interaction types	Designed for clinically grounded decision support, explanation generation, contextual reasoning	Multi-turn chat, MCQ, open-ended QA	MCQ, fill-in-the-blank, patient–doctor Q&A
Translation pipeline	Uses cultural + dialect-aware semantic representation	Semi-automated English → Arabic translation with human refinement	No translation pipeline; uses curated Arabic exam datasets
Dataset strategy	Regionally grounded multimodal datasets	1.3M bilingual interactions	Seven Arabic medical benchmark tasks
Architecture novelty	Layered architecture integrating Semantic Representation, Contextual Feature Selection, Multimodal Fusion, and Cultural Adaptation	Strength: scale + bilingual capability. No cultural adaptation	Strength: benchmark creation; no multimodal contextual reasoning
Cultural adaptation	Incorporates religious, linguistic, dietary, and family-centered patterns	Not included	Not included
Intended use Case	Cardiometabolic CDSS with culturally grounded guidance for MENA clinicians and patients	General biomedical QA + reasoning	Benchmarking LLMs for Arabic medical tasks

adaptive architecture, with each layer grounded in relevant literature on biomedical modeling, semantic representation, contextual feature selection, and culturally aligned communication. Although not yet experimentally implemented, the framework is designed to operate as a comprehensive CDSS capable of integrating biomedical data, contextual signals, and culturally grounded patient features to generate clinically actionable recommendations. The comparison above highlights these distinctions in objectives, scope, and architectural innovation (Table 7).

4.4 Challenges and Limitations

Within our proposed framework for culturally adaptive LLM-based CDSS in healthcare, several challenges and limitations emerge that directly affect its functionality and adoption in the MENA region. At the data input and preprocessing stages, the framework relies heavily on accurate and representative data collection; however,

heterogeneity across institutions, incomplete health records, and limited digital infrastructures in some parts of the region may constrain the robustness of this layer. In the semantic patient representation and contextual feature selection layers, the framework must contend with linguistic complexity, including dialectal variation and code-switching, which can hinder accurate cultural adaptation. The processing and output layers face challenges linked to clinician resistance, where AI-driven recommendations may be perceived as a threat to autonomy, and patient resistance, where cultural or religious norms may reduce adherence to AI-generated guidance. Finally, the explanation generation layer, while designed to foster trust through transparency and cultural resonance, must overcome ethical and legal barriers related to data privacy and accountability. These limitations highlight that the success of the framework depends not only on technical advancement but also on building trust, strengthening infrastructures, and embedding ethical safeguards. Some of the challenges are discussed below.

Data Availability and Accuracy. Data heterogeneity poses a major challenge for healthcare LLM-based systems, as clinical datasets vary in quality, format, and completeness, with most benchmarks being radiology-focused and Western-sourced. This limits generalizability and relevance for the MENA region, where fragmented infrastructures and incomplete records hinder robust data collection, increasing the risk of biased or suboptimal outputs. Coordinated efforts in data governance, technical infrastructure, stakeholder collaboration, and training are vital to mitigate the limitations of EHR systems. Standardization, data privacy, integration, and augmentation are key areas that require attention to enhance the quality and interoperability of healthcare data, ultimately leading to better patient care and clinical research outcomes.

Dialect Variation and Mixing. Language-related challenges also affect system performance. LLMs may conflate dialects with high lexical overlap or fail to distinguish subtle variations from Modern Standard Arabic (MSA). This limits the precision of patient education materials and clinical guidance. More extensive dialect-specific data collection and fine-tuning are required to ensure accurate, culturally sensitive communication. Our proposed framework mitigates these issues through its Semantic Representation and Contextual Feature Selection layers, which normalize diverse patient inputs and capture culturally grounded linguistic cues. Additional strategies include expanding dialect-specific data collection, integrating curated corpora for Gulf, Levantine, Egyptian, and Maghrebi Arabic, and applying dialect-aware pretraining and fine-tuning techniques to enhance the model's ability to differentiate dialects from MSA and generate culturally sensitive communication.

Resource Constraints in MENA. A further limitation lies in the unequal distribution of technological resources. Advanced AI systems require significant computational power, infrastructure, and skilled personnel resources that remain scarce in many parts of the MENA region. This digital divide risks deepening existing healthcare inequalities if not properly addressed through targeted investments and capacity building.

By combining technical optimization strategies such as model distillation, quantization, and hardware-efficient architectures [125] with strategic capacity building initiatives, clinics with limited infrastructure can successfully deploy LLM-based Clinical Decision Support Systems.

Resistance from Clinicians and Physicians. Another barrier is the hesitancy of healthcare professionals to adopt AI-driven CDSS. Concerns include threats to clinician autonomy, disruption of established workflows, and skepticism about the reliability of AI-generated recommendations. Without adequate training, trust-building, and user-centered design, adoption will remain limited.

Resistance from Patients. Similarly, patients may resist AI-based recommendations when these conflict with traditional practices, family-centered decision-making, or religious norms. Our proposed framework helps mitigate this challenge through the Explanation Generation Layer and Communicative Layer, which deliver transparent, culturally adapted messages aligned with local beliefs and values. Additional strategies include offering clear, patient-friendly explanations, incorporating culturally resonant examples, and using reinforcement learning from cultural feedback to refine tone and delivery. Together, these measures strengthen trust, respect patient autonomy, and support more collaborative and culturally acceptable decision-making.

Ethical and Legal Issues. The adoption of the proposed Multimodal LLM-CDSS raises important ethical and legal considerations, particularly related to patient privacy, informed consent, transparency, and accountability for AI-generated recommendations. Addressing these challenges requires proactive regulatory measures, interdisciplinary collaboration, and strong ethical governance. Our framework integrates several of these safeguards directly through its Explanation Generation Layer and governance-oriented design, which enhance transparency, support culturally appropriate communication, and ensure clear traceability of system outputs. Additional mitigation strategies include strengthening regulatory oversight with AI-specific policies, promoting collaboration among clinicians, technologists, ethicists, and policymakers, and developing ethical guidelines that prioritize fairness, data protection, and patient rights.

Governance and Safety. While the proposed LLM-CDSS is designed to function as a decision-support tool, providing recommendations, insights, and contextual analyses from multimodal patient data, the ultimate clinical decision-making authority appropriately remains with healthcare professionals, in alignment with prevailing regulatory expectations for AI-enabled healthcare technologies. To ensure responsible deployment, comprehensive governance mechanisms are embedded throughout the framework. These include PHI-safe data flows at the Input Layer, mandatory de-identification within the Pre-Processing Layer, and provenance tracking managed through the Data Governance Layer. Human-in-the-loop verification supports clinician oversight at the Reasoning and Output Layers, and system-wide red-team testing reinforces robustness and safety. Collectively, these measures integrate ethical, secure, and reliable operation across all stages of the LLM-CDSS.

5 Conclusion and Future Work

The development of culturally adaptive AI represents a significant opportunity to advance equitable, context-aware cardiometabolic care across the MENA region. This chapter has illustrated how the region's cardiometabolic burden arises from the interplay of biomedical risks, lifestyle transitions, and socio-cultural determinants. Against this backdrop, the proposed multimodal LLM-CDSS demonstrates how integrating clinical, linguistic, and contextual inputs can generate recommendations that are both clinically sound and culturally responsive.

A central contribution of this chapter is the introduction of a layered architecture, which constitutes the core novelty of the proposed framework. By structuring the system into distinct components ranging from multimodal preprocessing and semantic representation to contextual feature selection, LLM-based processing, and culturally aware explanation generation, the architecture establishes a coherent methodological foundation for future implementation and empirical validation. These considerations clarify the scope of the chapter: it presents a forward-looking conceptual framework designed to guide subsequent research rather than a fully operationalized system.

Nonetheless, key research challenges remain. Data heterogeneity, limited interoper-ability, and uneven digital infrastructure continue to restrict access to high-quality, representative datasets. Linguistic complexities, and the scarcity of dialect-specific corpora further constrain Arabic language modeling. In addition, ethical and legal concerns related to privacy, informed consent, and algorithmic account-ability highlight the need for stronger governance mechanisms to sustain clinician confidence and patient trust.

Future research should therefore progress along complementary trajectories. Technically, priorities include developing balanced and privacy-preserving multi-modal datasets, advancing dialect-aware pretraining for Arabic LLMs, and designing adaptive learning mechanisms capable of modeling evolving cultural and behav-ioral patterns. Clinically, rigorous prospective evaluations will be required to assess the safety, accuracy, and real-world impact of the proposed CDSS on diagnosis, risk stratification, and treatment adherence. Societally, participatory co-design with community stakeholders, continuous cultural auditing, and policy initiatives aimed at reducing digital inequities will be essential to ensure fairness, usability, and long-term uptake.

Importantly, the layered architecture presented here will serve as the blueprint for our forthcoming collaborative research projects, where we will begin implementing and empirically testing each component of the proposed framework. By advancing this agenda, future iterations of multimodal, culturally adaptive LLM-based systems can evolve from conceptual models into scalable, validated tools that enhance cardiometabolic care while contributing a broader paradigm for culturally informed AI in global health.

Acknowledgements The authors acknowledge the limited use of AI-based tools to assist with language editing and content refinement. All scientific content, analysis, and conclusions were developed and verified by the authors, who take full responsibility for the manuscript.

References

1. OpenAI.: GPT-5: The new multimodal model. OpenAI. (2025). https://openai.com/index/introducing-gpt-5/.
2. Google Research. (n.d.). Med-PaLM. https://sites.research.google/med-palm/.
3. Shaji Mullappilly, S., Kurpath, M. I., Pieri, S., Yahya Alseiari, S., Cholakkal, S., Aldahmani, K., Cholakkal, H.: BiMediX2: Bio-Medical EXpert LMM for Diverse Medical Modalities. (2024). arXiv e-prints, arXiv-2412. https://doi.org/10.48550/arXiv.2412.07769.
4. Daoud, M.A., Abouzahir, C., Kharouf, L., Al-Eisawi, W., Habash, N., Shamout, F.E.: Medarabiq: Benchmarking large language models on arabic medical tasks. (2025). arXiv preprint arXiv:2505.03427. https://doi.org/10.48550/arXiv.2505.03427.
5. Tan, M.M.C., Barbosa, M.G., Pinho, P., Assefa, E., Keinert, A.A.M., Hanlon, C., Consortium, M.: Determinants of multimorbidity in low- and middle-income countries: A systematic review of longitudinal studies and discovery of evidence gaps. Obes Rev, **25**(2), (2024). e13661. https://doi.org/10.1111/obr.13661
6. Wada, R., Peng, F.J., Lin, C.A., Vermeulen, R., Iglesias-Gonzalez, A., Palazzi, P., Chadeau-Hyam, M.: Hair-derived exposome exploration of cardiometabolic health: piloting a bayesian multitrait variable selection approach. Environ. Sci. Technol. **58**(12), 5383–5393 (2024). https://doi.org/10.1021/acs.est.3c08739
7. Barakou, I., Seves, B.L., Abonie, U.S., Finch, T., Hackett, K.L., Hettinga, F.J.: Health-related quality of life associated with fatigue, physical activity and activity pacing in adults with chronic conditions. BMC Sports Sci. Med. Rehabil. **17**(1), 13 (2025). https://doi.org/10.1186/s13102-025-01057-x
8. Grundy, S.M., Stone, N.J., Bailey, A.L., Beam, C., Birtcher, K.K., Blumenthal, R.S., Yeboah, J.: 2018 AHA/ACC/AACVPR/AAPA/ABC/ACPM/ADA/AGS/APhA/ASPC/NLA/PCNA guideline on the management of blood cholesterol: executive summary: a report of the american college of cardiology/American heart association task force on clinical practice guidelines. J. Am. Coll. Cardiol. **73**(24), 3168–3209 (2019). https://doi.org/10.1016/j.jacc.2018.11.002
9. Terada, T., Chirico, D., Tulloch, H.E., Scott, K., Doucet, É., Pipe, A.L., Reed, J.L.: Psychosocial and cardiometabolic health of patients with differing body mass index completing cardiac rehabilitation. Can. J. Cardiol. **35**(6), 712–720 (2019). https://doi.org/10.1016/j.cjca.2019.02.024
10. Liu, Y., Gao, Y., Yan, G., Liu, Y., Tian, W., Zhang, Y., Yu, B.: Global disease burden analysis of Cardiometabolic disease attributable to second-hand smoke exposure from 1990 to 2040. Am. J. Prev. Cardiol. **21**, 100902 (2025). https://doi.org/10.1016/j.ajpc.2024.100902
11. World Health, O.: Cardiovascular diseases (CVDs). (2025). Retrieved from https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).
12. Ndumele, C.E., Rangaswami, J., Chow, S.L., Neeland, I.J., Tuttle, K.R., Khan, S.S., Elkind, M.S.V.: Cardiovascular-kidney-metabolic health: a presidential advisory from the American heart association. Circulation **148**(20), 1606–1635 (2023). https://doi.org/10.1161/CIR.0000000000001184
13. Neeland, I.J., Lim, S., Tchernof, A., Gastaldelli, A., Rangaswami, J., Ndumele, C.E., Després, J.-P.: Metabolic syndrome. Nat. Rev. Dis. Primers. **10**(1), 77 (2024). https://doi.org/10.1038/s41572-024-00563-5

14. Sun, H., Saeedi, P., Karuranga, S., Pinkepank, M., Ogurtsova, K., Duncan, B.B., Magliano, D.J.: IDF Diabetes Atlas: Global, regional and country-level diabetes prevalence estimates for 2021 and projections for 2045. Diabetes Res. Clin. Pract. **183**, 109119 (2022). https://doi.org/10.1016/j.diabres.2021.109119
15. Brant, L.C.C., Miranda, J.J., Carrillo-Larco, R.M., Flood, D., Irazola, V., Ribeiro, A.L.P.: Epidemiology of cardiometabolic health in Latin America and strategies to address disparities. Nat. Rev. Cardiol. **21**(12), 849–864 (2024). https://doi.org/10.1038/s41569-024-01058-2
16. Ahmadi, M., Ahadi, S., Khadembashiri, M.A., Khadembashiri, M.M., Mahalleh, M., AziziKia, H., Akbari Sari, A.: Burden of ischemic heart disease in the Middle East and North Africa (MENA) and attributable risk factors: An epidemiological analysis from 1990 to 2019. Int J Cardiol Heart Vasc **50**, 101316 (2024). https://doi.org/10.1016/j.ijcha.2023.101316
17. Alhuneafat, L., Ta'ani, O.A., Jabri, A., Tarawneh, T., ElHamdan, A., Naser, A., Virani, S.S.: Cardiovascular disease burden in the Middle East and North Africa region. Curr. Probl. Cardiol. **49**(3), 102341 (2024). https://doi.org/10.1016/j.cpcardiol.2023.102341
18. Rukn, S.A., Mazya, M.V., Hentati, F., Sassi, S.B., Nabli, F., Said, Z., Moreira, T.: Stroke in the Middle-East and North Africa: A 2-year prospective observational study of stroke characteristics in the region-results from the safe implementation of treatments in stroke (SITS)-Middle-East and North African (MENA). Int. J. Stroke **14**(7), 715–722 (2019). https://doi.org/10.1177/1747493019830331
19. Khan, Y., Hamdy, O.: Type 2 Diabetes in the Middle East and North Africa (MENA). In S. Dagogo-Jack (Ed), Diabetes mellitus in developing countries and underserved communities (pp. 49–61). Cham: Springer International Publishing. (2017). https://doi.org/10.1007/978-3-319-41559-8_4.
20. Majeed, A., El-Sayed, A.A., Khoja, T., Alshamsan, R., Millett, C., Rawaf, S.: Diabetes in the Middle-East and North Africa: an update. Diabetes Res. Clin. Pract. **103**(2), 218–222 (2014). https://doi.org/10.1016/j.diabres.2013.11.008
21. Al-Rifai, R.H., Abdo, N.M., Paulo, M.S., Saha, S., Ahmed, L.A.: Prevalence of gestational diabetes mellitus in the Middle East and North Africa, 2000–2019: a systematic review, meta-analysis, and meta-regression. Front Endocrinol (Lausanne) **12**, 668447 (2021). https://doi.org/10.3389/fendo.2021.668447
22. Ng, M., Fleming, T., Robinson, M., Thomson, B., Graetz, N., Margono, C., Gakidou, E.: Global, regional, and national prevalence of overweight and obesity in children and adults during 1980–2013: a systematic analysis for the Global Burden of Disease Study 2013. Lancet **384**(9945), 766–781 (2014). https://doi.org/10.1016/S0140-6736(14)60460-8
23. International Diabetes Federation. (2025). IDF Diabetes Atlas (11th ed.). International Diabetes Federation. https://diabetesatlas.org/media/uploads/sites/3/2025/04/IDF_Atlas_11th_Edition_2025.pdf
24. Azizi, F., Hadaegh, F., Hosseinpanah, F., Mirmiran, P., Amouzegar, A., Abdi, H., Asghari, G., Parizadeh, D., Montazeri, S.A., Lotfaliany, M., Takyar, F., Khalili, D.: Metabolic health in the Middle East and North Africa. Lancet Diabetes Endocrinol. **7**(11), 866–879 (2019). https://doi.org/10.1016/S2213-8587(19)30179-2
25. Al-Daghri, N.M., Khan, N., Alkharfy, K.M., Al-Attas, O.S., Alokail, M.S., Alfawaz, H.A., Vanhoutte, P.M.: Selected dietary nutrients and the prevalence of metabolic syndrome in adult males and females in Saudi Arabia: a pilot study. Nutrients **5**(11), 4587–4604 (2013). https://doi.org/10.3390/nu5114587
26. Bekkouche, L., Bouchenak, M., Malaisse, W.J., Yahia, D.A.: The Mediterranean diet adoption improves metabolic, oxidative, and inflammatory abnormalities in Algerian metabolic syndrome patients. Horm. Metab. Res. **46**(4), 274–282 (2014). https://doi.org/10.1055/s-0033-1363657
27. Musaiger, A.O.: Overweight and obesity in eastern mediterranean region: prevalence and possible causes. J Obes **2011**, 407237 (2011). https://doi.org/10.1155/2011/407237
28. Yasein, N., Ahmad, M., Matrook, F., Nasir, L., Froelicher, E.S.: Metabolic syndrome in patients with hypertension attending a family practice clinic in Jordan. East Mediterr Health J, **16**(4), 375–380. (2010). https://pubmed.ncbi.nlm.nih.gov/20795419/.

29. Farrokhpour, M., Forogh, B., Gholampour, G., Nejadghaderi, S.A., Habibzadeh, A., Nakhostin-Ansari, A., Seyedmirzaei, H.: Burden of diseases attributable to low physical activity in the middle east and north africa: an analysis based on global burden of disease study. Balkan Med. J. **42**(2), 121–129 (2025). https://doi.org/10.4274/balkanmedj.galenos.2025.2025-1-20
30. Kargar, S., Ansari-Moghaddam, A.: Prevalence of cigarette and waterpipe smoking and associated cancer incidence among adults in the Middle East. East. Mediterr. Health J. **29**(9), 749–756 (2023). https://doi.org/10.26719/emhj.23.077
31. Khattab, A., Javaid, A., Iraqi, G., Alzaabi, A., Ben Kheder, A., Koniski, M.-L., Hasnaoui, A.: Smoking habits in the Middle East and North Africa: Results of the BREATHE study. Respir. Med. **106**, S16–S24 (2012). https://doi.org/10.1016/S0954-6111(12)70011-2
32. Nakkash, R., Khader, Y., Chalak, A., Abla, R., Abu-Rmeileh, N.M.E., Mostafa, A., Salloum, R.G.: Prevalence of cigarette and waterpipe tobacco smoking among adults in three Eastern Mediterranean countries: a cross-sectional household survey. BMJ Open **12**(3), e055201 (2022). https://doi.org/10.1136/bmjopen-2021-055201
33. Bhatnagar, A., Maziak, W., Eissenberg, T., Ward, K.D., Thurston, G., King, B.A., Rezk-Hanna, M.: Water pipe (Hookah) smoking and cardiovascular disease risk: a scientific statement from the american heart association. Circulation **139**(19), e917–e936 (2019). https://doi.org/10.1161/CIR.0000000000000671
34. Eissenberg, T., Shihadeh, A.: Waterpipe tobacco and cigarette smoking: direct comparison of toxicant exposure. Am. J. Prev. Med. **37**(6), 518–523 (2009). https://doi.org/10.1016/j.amepre.2009.07.014
35. Batalha, A., Marcal, I.R., Main, E., Ghisi, G.L.M.: Barriers to physical activity in women from ethnic minority groups: a systematic review. BMC Womens Health **25**(1), 330 (2025). https://doi.org/10.1186/s12905-025-03877-y
36. Aljehani, N., Razee, H., Ritchie, J., Valenzuela, T., Bunde-Birouste, A., Alkhaldi, G.: Exploring female university students' participation in physical activity in saudi arabia: a mixed-methods study. Front. Public Health **10**, 829296 (2022). https://doi.org/10.3389/fpubh.2022.829296
37. Chaabane, S., Chaabna, K., Doraiswamy, S., Mamtani, R., Cheema, S.: Barriers and Facilitators Associated with Physical Activity in the Middle East and North Africa Region: A Systematic Overview. Int. J. Environ Res. Public Health, **18**(4). (2021). https://doi.org/10.3390/ijerph18041647.
38. Boujelbane, M.A., Ammar, A., Salem, A., Kerkeni, M., Trabelsi, K., Bouaziz, B., Schöllhorn, W.I.: Gender-specific insights into adherence to Mediterranean diet and lifestyle: analysis of 4,000 responses from the MEDIET4ALL project. Front. Nutr. **12**, 1570904 (2025). https://doi.org/10.3389/fnut.2025.1570904
39. Sharara, E., Akik, C., Ghattas, H., Makhlouf Obermeyer, C.: Physical inactivity, gender and culture in Arab countries: a systematic assessment of the literature. BMC Public Health **18**(1), 639 (2018). https://doi.org/10.1186/s12889-018-5472-z
40. Onyishi, C.N., Ilechukwu, L.C., Victor-Aigbodion, V., Eseadi, C.: Impact of spiritual beliefs and faith-based interventions on diabetes management. World J. Diabetes **12**(5), 630–641 (2021). https://doi.org/10.4239/wjd.v12.i5.630
41. Williams, L., Gorman, R., Hankerson, S.: Implementing a mental health ministry committee in faith-based organizations: the promoting emotional wellness and spirituality program. Soc. Work Health Care **53**(4), 414–434 (2014). https://doi.org/10.1080/00981389.2014.880391
42. Molla, I.B., Hagger, V., Rothmann, M.J., Rasmussen, B.: The role of community organisation, religion, spirituality and cultural beliefs on diabetes social support and self-management in sub-saharan africa: integrative literature review. J. Relig. Health **64**(4), 2407–2464 (2025). https://doi.org/10.1007/s10943-024-02233-y
43. Almutairi, K.M., Alonazi, W.B., Vinluan, J.M., Almigbal, T.H., Batais, M.A., Alodhayani, A.A., Alhoqail, R.I.: Health promoting lifestyle of university students in Saudi Arabia: a cross-sectional assessment. BMC Public Health **18**(1), 1093 (2018). https://doi.org/10.1186/s12889-018-5999-z

44. Rouhani, M.H., Azadbakht, L.: Is Ramadan fasting related to health outcomes? A review on the related evidence. J. Res. Med. Sci, **19**(10), 987–992 (2014). PMCID: PMC4274578.
45. Lessan, N., Ali, T.: Energy metabolism and intermittent fasting: the ramadan perspective. Nutrients, **11**(5). (2019). https://doi.org/10.3390/nu11051192
46. Shaikh, S., Latheef, A., Razi, S.M., et al.: Diabetes Management During Ramadan. In Feingold, K.R., Ahmed, S.F., Anawalt, B. (Eds.): Endotext [Internet]. South Dartmouth (MA): MDText.com, Inc. Bookshelf ID: NBK581875 (2022)
47. Gul, R., Khan, I., Alam, I., Almajwal, A., Hussain, I., Sohail, N., Aftab, A.: Ramadan-specific nutrition education improves cardio-metabolic health and inflammation-a prospective nutrition intervention study from Pakistan. Front. Nutr. **10**, 1204883 (2023). https://doi.org/10.3389/fnut.2023.1204883
48. Woodside, J., Young, I.S., McKinley, M.C.: Culturally adapting the Mediterranean Diet pattern - a way of promoting more 'sustainable' dietary change? Br. J. Nutr. **128**(4), 693–703 (2022). https://doi.org/10.1017/S0007114522001945
49. Dahbi, N., El khair, A., Cheffi, K., Habibeddine, L., Talbi, J., Hilali, A., El ossmani, H.: Consanguinity, complex diseases and congenital disabilities in the Souss population (Southern Morocco): a cross-sectional survey. Egypt. J. Med. Hum. Genet., **25**(1), 27. (2024). https://link.springer.com/article/https://doi.org/10.1186/s43042-024-00490-w.
50. Yousef, N.A., ElHarouni, A.A., Shaik, N.A., Banaganapalli, B., Al Ghamdi, A.F., Galal, A.H., Elango, R.: Nationwide survey on awareness of consanguinity and genetic diseases in Saudi Arabia: challenges and potential solutions to reduce the national healthcare burden. Hum. Genomics **18**(1), 138 (2024). https://doi.org/10.1186/s40246-024-00700-x
51. Al-Mousa, H., Al-Saud, B.: Primary immunodeficiency diseases in highly consanguineous populations from middle East and North Africa: epidemiology, diagnosis, and care. Front. Immunol. **8**, 678 (2017). https://doi.org/10.3389/fimmu.2017.00678
52. Ben-Omran, T., Al Ghanim, K., Yavarna, T., El Akoum, M., Samara, M., Chandra, P., Al-Dewik, N.: Effects of consanguinity in a cohort of subjects with certain genetic disorders in Qatar. Mol. Genet. Genomic Med. **8**(1), e1051 (2020). https://doi.org/10.1002/mgg3.1051
53. Coughlin, S.S., Vernon, M., Hatzigeorgiou, C., George, V.: Health Literacy, Social Determinants of Health, and Disease Prevention and Control. J. Environ Health Sci. **6**(1) (2020). PMCID: PMC7889072
54. Sonmez, A., Sabbour, H., Echtay, A., Rahmah, A.M., Alhozali, A.M., Al Sabaan, F.S., Hassanein, M.: Current gaps in management and timely referral of cardiorenal complications among people with type 2 diabetes mellitus in the Middle East and African countries: Expert recommendations. J. Diabetes **14**(5), 315–333 (2022). https://doi.org/10.1111/1753-0407.13266
55. Ghafoor, E., Masood, S.N., Belkhadir, J., Sultan, M., Sandid, M., Baqai, S., Shegem, N.: Availability and Accessibility of Diabetes-Related Technologies in IDF-MENA Region. J. Diabetol., 13(Suppl 1). (2022). Retrieved from https://journals.lww.com/jodb/fulltext/2022/13001/availability_and_accessibility_of_diabetes_related.8.aspx.
56. Oleribe, O.O., Momoh, J., Uzochukwu, B.S., Mbofana, F., Adebiyi, A., Barbera, T., Taylor-Robinson, S.D.: Identifying key challenges facing healthcare systems in africa and potential solutions. Int. J. Gen. Med **12**, 395–403 (2019). https://doi.org/10.2147/IJGM.S223882
57. Wekesah, F.M., Kyobutungi, C., Grobbee, D.E., Klipstein-Grobusch, K.: Understanding of and perceptions towards cardiovascular diseases and their risk factors: a qualitative study among residents of urban informal settings in Nairobi. BMJ Open **9**(6), e026852 (2019). https://doi.org/10.1136/bmjopen-2018-026852
58. Bennis, A., Ogola, E.N., Klug, E., Skouri, H.N., Al Saffar, H.B., Ragy, H., Çavuşoğlu, Y.: Strategic recommendations to bridge the gaps in awareness, diagnosis and prevention of heart failure in the middle east region and Africa. J. Saudi Heart Assoc **34**(1), 53–65 (2022). https://doi.org/10.37616/2212-5043.1294
59. Liang, F., Yang, X., Peng, W., Zhen, S., Cao, W., Li, Q., Gu, D.: Applications of digital health approaches for cardiometabolic diseases prevention and management in the Western Pacific region. Lancet Reg. Health West Pac **43**, 100817 (2024). https://doi.org/10.1016/j.lanwpc.2023.100817

60. Katoue, M.G., Cerda, A.A., García, L.Y., Jakovljevic, M.: Healthcare system development in the Middle East and North Africa region: Challenges, endeavors and prospective opportunities. Front. Public Health **10**, 1045739 (2022). https://doi.org/10.3389/fpubh.2022.1045739
61. Phillimore, P., Zaman, S., Ahmad, B., Shoaibi, A., Khatib, R., Khatib, R., Critchley, J.: Health system challenges of cardiovascular disease and diabetes in four Eastern Mediterranean countries. Glob. Public Health **8**(8), 875–889 (2013). https://doi.org/10.1080/17441692.2013.830756
62. El-Jardali, F., Bou-Karroum, L., Jabbour, M., Bou-Karroum, K., Aoun, A., Salameh, S., Sinha, C.: Digital health in fragile states in the Middle East and North Africa (MENA) region: A scoping review of the literature. PLoS ONE **18**(4), e0285226 (2023). https://doi.org/10.1371/journal.pone.0285226
63. Al-Sofiani, M.E., Alyusuf, E.Y., Alharthi, S., Alguwaihes, A.M., Al-Khalifah, R., Alfadda, A.: Rapid implementation of a diabetes telemedicine clinic during the coronavirus disease 2019 outbreak: our protocol, experience, and satisfaction reports in Saudi Arabia. J. Diabetes Sci. Technol. **15**(2), 329–338 (2021). https://doi.org/10.1177/1932296820947094
64. Alkhaibari, R.A., et al.: Patient-centered care in the Middle East and North African region: a systematic literature review. BMC Health Serv. Res. **23**(1), 135 (2023). https://doi.org/10.1186/s12913-023-09132-0
65. Hassanein, M., et al.: The characteristics and pattern of care for the type 2 diabetes mellitus population in the MENA region during Ramadan: An international prospective study (DAR-MENA T2DM). Diabetes Res. Clin. Pract. **151**, 275–284 (2019). https://doi.org/10.1016/j.diabres.2019.02.020
66. Ouahid, H., et al.: Gender norms and access to sexual and reproductive health services among women in the Marrakech-Safi region of Morocco: A qualitative study. BMC Pregnancy Childbirth **23**(1), 407 (2023). https://doi.org/10.1186/s12884-023-05724-0
67. Chen, J.H., Asch, S.M.: Machine learning and prediction in medicine—beyond the peak of inflated expectations. N. Engl. J. Med. **376**(26), 2507–2509 (2017). https://doi.org/10.1056/NEJMp1702071
68. Rajkomar, A., Dean, J., Kohane, I.: Machine learning in medicine. N. Engl. J. Med. **380**(14), 1347–1358 (2019). https://doi.org/10.1056/NEJMra1814259
69. Johnson, K.W., Torres Soto, J., Glicksberg, B.S., Shameer, K., Miotto, R., Ali, M., Dudley, J.T.: Artificial intelligence in cardiology. J. Am. Coll. Cardiol. **71**(23), 2668–2679 (2018). https://doi.org/10.1016/j.jacc.2018.03.521
70. Monlezun, D.J., Omutoko, L., Oduor, P., Kokonya, D., Rayel, J., Sotomayor, C., Gallagher, C.: Digitalization of health care in low- and middle-income countries. Bull. World Health Organ. **103**(2), 148–154 (2025). https://doi.org/10.2471/BLT.24.291643
71. Alowais, S.A., Alghamdi, S.S., Alsuhebany, N., Alqahtani, T., Alshaya, A.I., Almohareb, S.N., Albekairy, A.M.: Revolutionizing healthcare: the role of artificial intelligence in clinical practice. BMC Med. Educ. **23**(1), 689 (2023). https://doi.org/10.1186/s12909-023-04698-z
72. Waqas, A., Mehmood, S., Jawwad, A.M., Pittam, B., Kundu, S., Correia, J.C., AlMughamis, N.: Telemedicine in Arab countries: innovation, research trends, and way forward. Front Digit Health **2**, 610837 (2020). https://doi.org/10.3389/fdgth.2020.610837
73. Agrawal, K., Goktas, P., Kumar, N., Leung, M.F.: Artificial intelligence in personalized nutrition and food manufacturing: a comprehensive review of methods, applications, and future directions. Front. Nutr. **12**, 1636980 (2025). https://doi.org/10.3389/fnut.2025.1636980
74. Shaalan, K., et al.: Challenges in Arabic natural language processing. In: Computational Linguistics, Speech and Image Processing for Arabic Language, pp. 59–83 (2019). https://doi.org/10.3390/computers14110497
75. Alshabanah, A., Annavaram, M.: On using Arabic language dialects in recommendation systems. In: Findings of the Association for Computational Linguistics: NAACL **2025** (2025). https://doi.org/10.18653/v1/2025.findings-naacl.115
76. Makhseed, A., Arian, H., Shuaib, A.: Beyond the algorithm: A perspective on tackling bias and cultural sensitivity in AI-guided aesthetic standards for cosmetic surgery in the Middle East and North Africa (MENA) region. Clinical, Cosmetic and Investigational Dermatology, 2173–2182 (2025). https://doi.org/10.2147/CCID.S543045

77. Baker, S., Xiang, W.: Explainable AI is responsible AI: How explainability creates trustworthy and socially responsible artificial intelligence. https://doi.org/10.48550/arXiv.2312.01555
78. Chinta, S.V., et al.: AI-driven healthcare: A survey on ensuring fairness and mitigating bias. arXiv:2407.19655 (2024). https://doi.org/10.48550/arXiv.2407.19655
79. Liao, Q.V., Wortman Vaughan, J.: AI transparency in the age of LLMs: A human-centered research roadmap. arXiv:2306.01941 (2023). https://doi.org/10.48550/arXiv.2306.01941
80. Trigui, H., et al.: Exploring AI governance in the Middle East and North Africa (MENA) region: Gaps, efforts, and initiatives. Data & Policy **6**, e83 (2024). https://doi.org/10.1017/dap.2024.85
81. Luo, R., Sun, L., Xia, Y., Qin, T., Zhang, S., Poon, H., Liu, T.-Y.: BioGPT: Generative pre-trained transformer for biomedical text generation and mining. Brief. Bioinform., **23**(6), bbac409. (2022). https://doi.org/10.1093/bib/bbac409
82. H Nazi, Z.A., Peng, W.: Large language models in healthcare and medical domain: A review. In Informatics **11**(3), p. 57) (2024). MDPI. https://doi.org/10.3390/informatics11030057
83. Al Zaabi, A., Alshibli, R., AlAmri, A., AlRuheili, I., Lutfi, S.L.: Trends and trajectories in the rise of large language models in radiology: scoping review. JMIR Med. Inform. **13**(1), e78041 (2025). https://doi.org/10.2196/78041
84. He, J., Li, P., Liu, G., He, G., Chen, Z., Zhong, S. (2024). PeFoMed: Parameter-efficient fine-tuning of multimodal large language models for medical imaging (arXiv:2401.02797). arXiv. https://doi.org/10.48550/arXiv.2401.02797
85. Al Ali, S.A., AlDehlawi, H., Jazzar, A., Ashi, H., Abuzinadah, N.E., AlOtaibi, M., Almazrooa, S.: The Diagnostic performance of large language models and oral medicine consultants for identifying oral lesions in text-based clinical scenarios: prospective comparative study. JMIR AI **4**(1), e70566 (2025). https://doi.org/10.2196/70566
86. Syed, W., Babelghaith, S.D., Al-Arifi, M.N.: Assessment of Saudi public perceptions and opinions towards artificial intelligence in health care. Medicina **60**(6), 938 (2024). https://doi.org/10.3390/medicina60060938
87. Abdelhay, M.: MAQA: Medical Arabic Q&A Dataset (Version 1.0). Harvard Dataverse. (2022). https://doi.org/10.7910/DVN/Y2JBEZ
88. Alhuzali, H., Alasmari, A., Alsaleh, H.: Mentalqa: An annotated arabic corpus for questions and answers of mental healthcare. IEEE Access (2024). https://doi.org/10.1109/ACCESS.2024.3430068
89. Jiang, X., Yan, L., Vavekanand, R., Hu, M.: Large language models in healthcare current development and future directions. In Generative AI Research. (2023). https://doi.org/10.20944/preprints202407.0923.v1
90. Alkhaledi, R., Alkhaledi, M., Hawamdeh, S.: Barriers to the adoption of EHR in GCC countries: Exploratory study. Information Needs. (2020). https://www.ideals.illinois.edu/items/116445.
91. Shaheen, A.M., Hamdan, K.M., Albqoor, M.A., Arabiat, D.H.: Perceived barriers to healthcare utilization among Jordanian families: A family centered approach. Appl. Nurs. Res. **54**, 151313 (2020). https://doi.org/10.1016/j.apnr.2020.151313
92. Holtz, B., Nelson, V., Poropatich, R.K.: Artificial intelligence in health: enhancing a return to patient-centered communication. Telemedicine and e-Health **29**(6), 795–797 (2023). https://doi.org/10.1089/tmj.2022.041
93. Alayba, A.M.: Arabic natural language processing (NLP): a comprehensive review of challenges, techniques, and emerging trends. Computers **14**(11), 497 (2025). https://doi.org/10.3390/computers14110497
94. Alshahrani, H., McConkey, R., Wilson, J., Youssef, M., Fitzsimons, D.: Female gender doubles pre-hospital delay times for patients experiencing ST segment elevation myocardial infarction in Saudi Arabia. Eur. J. Cardiovasc. Nurs. **13**(5), 399–407 (2014). https://doi.org/10.1177/1474515113507159
95. Alomair, N., Alageel, S., Davies, N., Bailey, J.V.: Factors influencing sexual and reproductive health of Muslim women: a systematic review. Reprod. Health (2020). https://doi.org/10.1186/s12978-020-0888-1

96. Näher, A.F., Krumpal, I., Antão, E.M., Ong, E., Rojo, M., Kaggwa, F., Agha-Mir-Salim, L.: Measuring fairness preferences is important for artificial intelligence in health care. The Lancet Digital Health **6**(5), e302–e304 (2024). https://doi.org/10.1016/S2589-7500(24)00059-1
97. Bagave, P., Westberg, M., Janssen, M., Ding, A.Y.: Accountability framework for healthcare ai systems: towards joint accountability in decision making. In Proceedings of the AAAI/ACM conference on AI, ethics, and society **8**(1), pp. 279–291. (2025) https://doi.org/10.1609/aies.v8i1.36548.
98. Anderson, J.L., Hall, A.G., Feldman, S.S., Mehta, T.: Exploring a need for a cardiometabolic disease staging system as a computerized clinical decision support tool: qualitative study. JMIR Form. Res., **6**(7), e3745 (2022). https://doi.org/10.2196/37456.
99. Ardic, N., Dinc, R.: Emerging trends in multi-modal artificial intelligence for clinical decision support: A narrative review. Health Inform. J., **31**(3) (2025). https://doi.org/10.1177/14604582251366141.
100. Laaouar, S., Yanes, N.: CLARA: A context-aware drug recommendation framework using multi-level integration of large language models and graph neural networks. Procedia Comput. Sci. **270**, 4085–4094 (2025). https://doi.org/10.1016/j.procs.2025.09.533
101. Rajpurkar, P., Chen, E., Banerjee, O., Topol, E.J.: AI in health and medicine. Nat. Med. **28**(1), 31–38 (2022). https://doi.org/10.1038/s41591-021-01614-0
102. Kline, A., Wang, H., Li, Y., Dennis, S., Hutch, M., Xu, Z., Luo, Y.: Multimodal machine learning in precision health: A scoping review. NPJ digital medicine **5**(1), 171 (2022). https://doi.org/10.1038/s41746-022-00712-8
103. Mansoor, I., Abdullah, M., Rizwan, M.D., Fraz, M.M.: Reasoning with large language models in medicine: A systematic review of techniques, challenges and clinical integration. J. Clin. Monit. Comput. (2025). https://doi.org/10.1007/s13755-025-00403-0
104. Chow, J.C., Li, K.: Large language models in medical chatbots: opportunities, challenges, and the need to address AI risks. Information **16**(7), 549 (2025). https://doi.org/10.3390/info16070549
105. Vrdoljak, J., Boban, Z., Vilović, M., Kumrić, M., Božić, J.: A review of large language models in medical education, clinical decision support, and healthcare administration. In Healthcare MDPI **13**(6), p. 603 (2025) https://doi.org/10.3390/healthcare13060603
106. Gargari, O.K., Habibi, G.: Enhancing medical AI with retrieval-augmented generation: A mini narrative review. Digital health **11**, 20552076251337176 (2025). https://doi.org/10.1177/20552076251337177
107. Aid, A., Azzoune, R., Abdellaoui, I., Haddouche, A.: Re-SciBERT: an entity-enriched language model to enhance biomedical relation extraction. In: 2024 1st International Conference on Innovative and Intelligent Information Technologies (IC3IT) (pp. 1–6). IEEE (2024). https://doi.org/10.1109/IC3IT63743.2024.10869408.
108. Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.E., Jégou, H.: The faiss library. IEEE Trans. Big Data. (2025). https://doi.org/10.48550/arXiv.2401.08281
109. Abdul-Mageed, M., Elmadany, A.: ARBERT & MARBERT: Deep bidirectional transformers for Arabic. In: Proceedings of the 59th Annual Meeting Of The Association For Computational Linguistics and the 11th International Joint Conference On Natural Language Processing (volume 1: long papers) (pp. 7088–7105) (2021). https://doi.org/10.18653/v1/2021.acl-long.551.
110. Abid, M.A., Rahman, A.U., Ali, S., Haider, U.: Clinical reports classification using fine-tuned DistilBERT model. In 2024 International Conference on IT and Industrial Technologies (ICIT) (pp. 1–6). IEEE. (2024). https://doi.org/10.1109/ICIT63607.2024.10859364.
111. Hejblum, B.P., Weber, G.M., Liao, K.P., Palmer, N.P., Churchill, S., Shadick, N.A., Szolovits, P., Murphy, S. N., Kohane, I.S., Cai, T.: Probabilistic record linkage of de-identified research datasets with discrepancies using diagnosis codes. Scientific Data, 6, Article 180298. (2019). https://doi.org/10.1038/sdata.2018.298.
112. Rane, J., Chaudhari, R.A., Rane, N.L.: Data analysis and information processing frameworks for ethical artificial intelligence implementation: machine-learning algorithm validation in

clinical research settings. In: Ethical Considerations and Bias Detection in Artificial Intelligence/Machine Learning Applications, **192** (2025). https://doi.org/10.1016/j.modpat.2024.100686
113. Zhao, J., Feng, Q., Wu, P., et al.: Learning from longitudinal data in electronic health record and genetic data to improve cardiovascular event prediction. Sci. Rep. **9**, 717 (2019). https://doi.org/10.1038/s41598-018-36745-x
114. Shang, X., Zhao, F., Du, J., Li, R., Xia, L., Hou, L., Xu, Y.: Development and clinical evaluation of an online automated quality control system for improving laboratory quality management. Clin. Chim. Acta **541**, 117240 (2023). https://doi.org/10.1016/j.cca.2023.117240
115. Mandel, J.C., Kreda, D.A., Mandl, K.D., Kohane, I.S., Ramoni, R.B.: SMART on FHIR: a standards-based, interoperable apps platform for electronic health records. J. Am. Med. Inform. Assoc. **23**(5), 899–908 (2016). https://doi.org/10.1093/jamia/ocv189
116. Kwon, Y., Kim, J., Lee, G., Bae, S., Kyung, D., Cha, W., Pollard, T., Johnson, A., Choi, E.: EHRCon: Dataset for checking consistency between unstructured notes and structured tables in electronic health records (arXiv:2406.16341). arXiv. (2024). https://doi.org/10.48550/arXiv.2406.16341
117. Röchner, P., Rothlauf, F.: Unsupervised anomaly detection of implausible electronic health records: a real-world evaluation in cancer registries. BMC Med. Res. Methodol. **23**(1), 125 (2023). https://doi.org/10.1186/s12874-023-01946-0
118. Hripcsak, G., Duke, J.D., Shah, N.H., et al.: Observational health data sciences and informatics (OHDSI): standardizing observational health data worldwide. J. Am. Med. Inform. Assoc. **27**(10), 1612–1620 (2020). https://doi.org/10.3233/978-1-61499-564-7-574
119. Kurtz, M., Winter, A., Löbe, M.: Suitability of the OMOP common data model for mapping datasets of medical research studies using the example of a multicenter registry. In: Collab. Discipl. Health People, Anim. Ecosyst. (pp. 170–174). IOS Press. (2024). https://doi.org/10.3233/SHTI241086.
120. Adapa, C.S.R.: Eliminating duplicate medical records: How modern solutions are revolutionizing healthcare data management. J. Comput. Sci. Technol. Stud., **7**(2), 496–505 (2025). https://doi.org/10.32996/jcsts.2025.7.2.52.
121. Pech, F., Martinez, A., Estrada, H., Hernandez, Y.: Semantic annotation of unstructured documents using concepts similarity. Sci. Program. **2017**(1), 7831897 (2017). https://doi.org/10.1155/2017/7831897
122. Gonzalez-Jaramillo, N., Marcin, T., Matter, S., Eser, P., Berlin, C., Bano, A., Wilhelm, M.: Clinical outcomes and cardiac rehabilitation in underrepresented groups after percutaneous coronary intervention: an observational study. Eur. J. Prev. Cardiol. **29**(7), 1093–1103 (2022). https://doi.org/10.1093/eurjpc/zwab204
123. Valdenor, C., Ganesan, D., Paculdo, D., Schrecker, J., Heltsley, R., Westerfield, C., Peabody, J.W.: Clinical variation in the treatment practices for medication nonadherence, drug-drug interactions, and recognition of disease progression in patients with chronic cardiometabolic diseases: a cross-sectional patient simulation study among primary care physicians. Int. J. Clin. Pract. **2022**(1), 6450641 (2022). https://doi.org/10.1155/2022/6450641
124. Dardour, S., Yanes, N.: Leveraging large language models for feature selection in drug recommendation systems. Concurr. Comput.: Pract. Exp. **37**(27–28), e70370 (2025). https://doi.org/10.1002/cpe.70370
125. Weng, C., Yang, X., Shen, Y., Xu, X., Lin, B.: Leveraging large language models for early diagnosis of inherited metabolic diseases evaluation and optimization. In: International Conference on Intelligent Computing (pp. 89–99). Singapore: Springer Nature Singapore. (2025). https://doi.org/10.1007/978-981-95-0036-9_8.

Dr. Samia Kouki is an Assistant Professor of Computer Information Science at the Higher Colleges of Technology (HCT), UAE. She has more than 25+ academic and industrial experi-ence across Tunisia, France and UAE. Her research focuses on artificial intelligence, cloud computing,

and digital transformation, with particular interest in AI-driven systems for healthcare, education, and sustainability. She is recognized for her interdisciplinary work inte-grating advanced technologies with real-world applications and industry collaboration. She has published several scientific papers in various international journals and conferences.

Dr. Nacim YANES received the master's degree in computer science applied to management from the Higher Institute of Management (ISG) of Tunis, Tunisia and his PhD in computer science from the National School of Computer Science (ENSI), La Manouba University, Tunisia. He is an Assistant Professor in the Higher Institute of Management (ISGGB), Universi-ty of Gabes, Tunisia. His current research interests include Recommenders Systems, AI-based Healthcare Recommender Systems, Recommenders Systems in Software Engineering, Serious Games and Gamification, and Outcome-based Education. He has published several scientific papers in various international journals and conferences.

Dr. Heba Khalil is an Assistant Professor of Nursing with 20 years of academic and clinical experience across Jordan, the USA, and the UAE. She earned her PhD in Nursing from the University of Pittsburgh, USA, and has a strong background in critical care, nursing educa-tion, and research mentorship, having taught at both undergraduate and graduate levels. Dr. Heba has a well-established research portfolio, with peer-reviewed publications in respect-ed journals covering nursing practice, patient care, and healthcare innovation. Beyond academia, Dr. Heba has held key leadership positions in nursing education and ac-creditation, actively contributing to curriculum development, faculty mentorship, and program accreditation processes.

Prof. Nor Shahriza Abdul Karim is a Professor in the College of Computer and Infor-mation Sciences at Prince Sultan University, Riyadh. She earned her Ph.D. in Information Science and Technology from Syracuse University, USA. Her research focuses on infor-mation systems, user behavior, and technology adoption, with notable publications on ICT ef-fectiveness, social media disclosure, persuasive technology design, health informatics, and AI. Professor Nor Shahriza is widely recognized for her interdisciplinary work that integrates be-havioral and technological perspectives to enhance collaboration, communication, and perfor-mance in academic and organizational settings. Her recent interest also revolves around digital transformation, AI in healthcare, and AI in education.