Deep Reinforcement Learning-Based Energy Management for Hybrid Electric Vehicles

Synthesis Lectures on Advances in Automotive Technology

Editor
Amir Khajepour, ***University of Waterloo***

The automotive industry has entered a transformational period that will see an unprecedented evolution in the technological capabilities of vehicles. Significant advances in new manufacturing techniques, low-cost sensors, high processing power, and ubiquitous real-time access to information mean that vehicles are rapidly changing and growing in complexity. These new technologies—including the inevitable evolution toward autonomous vehicles—will ultimately deliver substantial benefits to drivers, passengers, and the environment. Synthesis Lectures on Advances in Automotive Technology Series is intended to introduce such new transformational technologies in the automotive industry to its readers.

Vehicle Suspension System Technology and Design
Avesta Goodarzi and Amir Khajepour
2017

Deep Reinforcement Learning-Based Energy Management for Hybrid Electric Vehicles

Yuecheng Li and Hongwen He

www.morganclaypool.com

ISBN: 9781636393018	paperback
ISBN: 9781636393025	PDF
ISBN: 9781636393032	hardcover

DOI 10.2200/S01156ED1V01Y202112AAT016

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON ADVANCES IN AUTOMOTIVE TECHNOLOGY

Lecture #16
Series Editor: Amir Khajepour, *University of Waterloo*
Series ISSN
Print 2576-8107   Electronic 2576-8131

Deep Reinforcement Learning-Based Energy Management for Hybrid Electric Vehicles

Yuecheng Li
Beijing Institute of Specialized Machinery

Hongwen He
Beijing Institute of Technology

SYNTHESIS LECTURES ON ADVANCES IN AUTOMOTIVE TECHNOLOGY #16

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

The urgent need for vehicle electrification and improvement in fuel efficiency has gained increasing attention worldwide. Regarding this concern, the solution of hybrid vehicle systems has proven its value from academic research and industry applications, where energy management plays a key role in taking full advantage of hybrid electric vehicles (HEVs). There are many well-established energy management approaches, ranging from rules-based strategies to optimization-based methods, that can provide diverse options to achieve higher fuel economy performance. However, the research scope for energy management is still expanding with the development of intelligent transportation systems and the improvement in onboard sensing and computing resources. Owing to the boom in machine learning, especially deep learning and deep reinforcement learning (DRL), research on learning-based energy management strategies (EMSs) is gradually gaining more momentum. They have shown great promise in not only being capable of dealing with big data, but also in generalizing previously learned rules to new scenarios without complex manually tunning.

Focusing on learning-based energy management with DRL as the core, this book begins with an introduction to the background of DRL in HEV energy management. The strengths and limitations of typical DRL-based EMSs are identified according to the types of state space and action space in energy management. Accordingly, value-based, policy gradient-based, and hybrid action space-oriented energy management methods via DRL are discussed, respectively. Finally, a general online integration scheme for DRL-based EMS is described to bridge the gap between strategy learning in the simulator and strategy deployment on the vehicle controller.

KEYWORDS

hybrid electric vehicles, learning-based energy management strategies, deep reinforcement learning, deep neural networks, optimality, generalization, information fusion, interpretability, knowledge distillation

Contents

1 Introduction . . . 1

1.1 Motivation . . . 1

1.2 HEV Powertrain . . . 2

1.3 Literature Review . . . 3

1.3.1 The Rule-Based EMS . . . 3

1.3.2 The Optimization-Based EMS . . . 4

1.3.3 The Learning-Based EMS . . . 6

2 Background: Deep Reinforcement Learning . . . 9

2.1 Reinforcement Learning in Energy Management . . . 10

2.2 The State Space and Action Space in Reinforcement Learning . . . 13

2.2.1 Discrete State Space and Discrete Action Space . . . 14

2.2.2 Continuous State Space and Continuous Action Space . . . 15

2.2.3 Hybrid Action Space . . . 16

2.2.4 State and Action Space in HEV Energy Management . . . 16

2.3 Literature Review on Deep Reinforcement Learning . . . 17

3 Learning of EMSs in Continuous State Space–Discrete Action Space . . . 23

3.1 Energy Consumption Model of a Series Hybrid Electric Vehicle . . . 23

3.2 Energy Management Based on Deep Q-Learning Method . . . 27

3.2.1 Discrete Action Space in Energy Management . . . 27

3.2.2 Learning Theory of DQN-Based EMSs . . . 28

3.2.3 Training of DQN-Based EMSs . . . 30

3.3 Improvements for Stable Learning in Discrete Action Space . . . 31

3.3.1 Improvement in DQN Structure . . . 32

3.3.2 Improvement in Estimation of Target Q-Value . . . 34

3.3.3 Improvement in Experience Replay . . . 36

3.3.4 Stable Training Method for DQN-Based EMSs . . . 39

3.4 The Learning and Evaluation of EMSs in Continuous State Space . . . 41

3.4.1 The Baseline Strategy: DP-Based EMSs . . . 41

3.4.2 Learning Evaluation . . . 41

3.4.3 Optimization Evaluation . . . 43
3.4.4 Generalization Evaluation . . . 47

4 Learning of EMSs in Continuous State–Continuous Action Space . . . 51

4.1 Energy Management Based on DDPG Method . . . 51
4.1.1 Learning Theory of DDPG-Based EMSs . . . 52
4.1.2 Training of DDPG-Based EMSs . . . 54
4.2 Continuous Energy Management Enabled by Trip Information . . . 56
4.2.1 Utilization of History Trip Information . . . 56
4.2.2 Reward Definition . . . 58
4.2.3 Training Algorithm . . . 60
4.2.4 Learning Evaluation . . . 62
4.2.5 Optimization Evaluation . . . 64
4.2.6 Generalization Evaluation . . . 69
4.3 Comparison Between Continuous DRL-Based and MPC-Based EMSs . . . 72
4.3.1 The Contrast Method: MPC-Based EMSs . . . 72
4.3.2 Comparison Under Blended Mode . . . 73
4.3.3 Comparison Under CS Mode . . . 75
4.4 Summary . . . 76

5 Learning of EMSs in Discrete-Continuous Hybrid Action Space . . . 77

5.1 Energy Consumption Model of a Power-Split HEB . . . 77
5.2 Searching of Optimal EMSs in Hybrid Action Space by DDPG . . . 81
5.2.1 HEB Energy Management and its Hybrid Action Space . . . 81
5.2.2 Learning of EMSs in Hybrid Action Space . . . 82
5.3 EMSs Considering Terrain Information Under Hybrid Action Space . . . 83
5.3.1 Generation of Velocity-Slope Profiles . . . 85
5.3.2 Pre-Training Method . . . 87
5.3.3 Training Algorithm . . . 87
5.4 Comparative Case Analysis . . . 88
5.4.1 Development of Contrast Strategies . . . 88
5.4.2 Learning Process in Hybrid Action Space . . . 90
5.4.3 Optimization Evaluation Considering Terrain Information . . . 92
5.4.4 Generalization Evaluation . . . 96
5.4.5 Influence of Terrain Information . . . 96
5.5 Summary . . . 98

6 An Online Integration Scheme for DRL-Based EMSs 101

6.1 Reconstruction of Parameterized EMSs in Matlab/Simulink 101

6.1.1 Extraction of Parameterized EMSs 101

6.1.2 Reconstruction in Matlab/Simulink 104

6.2 Evaluation by HIL Test 106

6.3 Summary 108

7 Conclusions 109

Bibliography 111

Authors' Biographies 123

CHAPTER 1

Introduction

1.1 MOTIVATION

The world is actively promoting the application and development of renewable energy, and constantly speeding up the transformation of the energy system from fossil energy-based to low-carbon green energy-based. This has considerably contributed to the development and production of hybrid electric vehicles (HEVs). As a vehicle electrification technology that is not limited by battery range and charging infrastructure construction, hybrid technology is also one of the most commercially successful energy-efficient technologies for vehicles.

With their diverse on-board energy sources and flexible working modes, HEVs are more helpful to improve the vehicle energy efficiency. For example, under frequent start-stop urban driving conditions, HEVs can work in pure electric mode to avoid idling emissions; they can adjust the engine operating point by coordinating the generator set, thus improving the engine thermal efficiency, etc. For plug-in hybrid electric vehicles (PHEVs), they can ensure a certain pure electric range, but also rely on engine drive at high-speed and long-distance driving, eliminating the driver's mileage anxiety. However, due to the coupled characteristics of multiple energy flows in HEV powertrains, energy management becomes critical for realizing their energy-saving potential and reducing application costs.

Along with the rapid technical development of new energy vehicles, intelligent transportation systems are also flourishing. Intelligent and connected vehicles have also become another breakthrough in the development of the next generation of vehicles. They are designed for safety and efficiency, and featured with information sharing, environment awareness, intelligent decision making, and automated collaboration. Simultaneously, the environmental perception information available for vehicle energy-saving control is enriched. However, traditional control methods are relatively limited in their ability to process and mine multi-source, high-dimensional sensory data. Therefore, it will be helpful to explore new approaches for energy management in an intelligent connected environment from two aspects: (1) work on new intelligent energy-efficient control methods to explore the potential correlation between data information involved during vehicle driving and efficient powertrain energy distribution; and (2) work on how to enable strategies to learn more efficient energy management schemes from data on their own.

Therefore, with machine learning as the core, especially deep reinforcement learning (DRL) which has been prominent in intelligent decision-making in recent years, we carry out research on learning-based energy-saving control strategies and their stable learning methods.

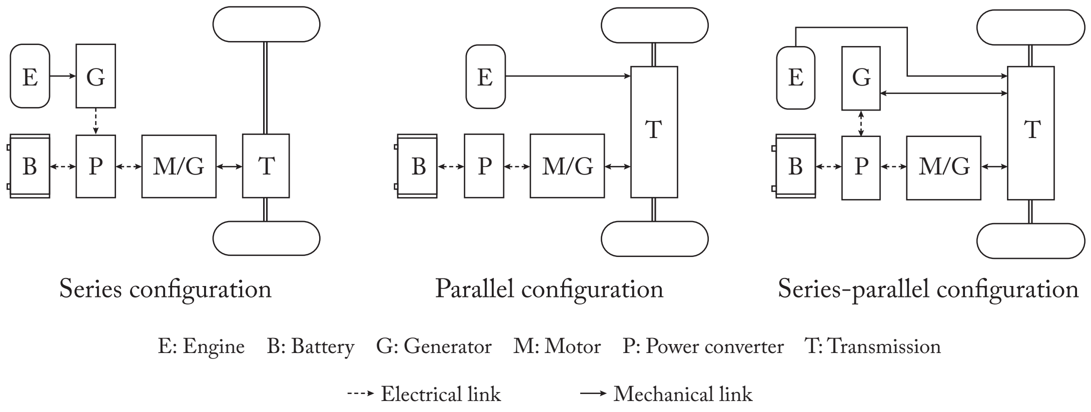

E
G
B
P
M/G
T
Series configuration
Parallel configuration
Series-parallel configuration
E: Engine B: Battery G: Generator M: Motor P: Power converter T: Transmission
Electrical link
Mechanical link

Figure 1.1: Typical configurations of HEVs.

These works explore the value of intelligent control methods in tapping the energy-saving potential of HEVs. Hopefully they could provide referenceable algorithmic experiences for relevant research on intelligent energy-saving control methods for HEVs in network-connected environments.

1.2 HEV POWERTRAIN

The on-board energy system of HEVs usually has two or more energy sources. Currently, hybrid powertrains mostly use internal combustion engine or fuel cell as the main energy supply unit, supplemented by auxiliary energy supply or storage units, such as battery [1], super capacitor [2], air compressor [3], hydraulic pump [4], super flywheel [5], etc. The "internal combustion engine + battery" is the hybrid solution adopted by most HEVs on the market today. According to different power coupling methods, usually the typical hybrid systems can be divided into three categories: series configuration, parallel configuration, and series-parallel configuration, as shown in Figure 1.1.

Series HEVs are driven directly by electric motors. The engine power is converted by the generator into electrical energy, which in turn charges the battery or drives the motor directly. The significant advantage is that the engine can be decoupled from the driving cycle with a simple powertrain, making it easier to regulate the operation of engine-generator set. However, multiple energy conversions can lead to a reduction in powertrain efficiency. Meanwhile, to develop vehicles of this configuration, the selection of each power component needs to be matched according to the power capabilities, which may lead to an increase in manufacturing costs.

Parallel HEVs can be driven by either the engine or electric motors alone, or by both together. The electric motor can recover braking energy or surplus engine power. Compared to the series configuration, this one does not require a generator, and we could utilize a smaller engine and motor as its power components, resulting in lower manufacturing costs. However, its engine operation is not fully decoupled from driving cycles.

The powertrain of series-parallel HEVs is relatively complex. The engine, generator, and motor can be coupled by gear transmissions [6], planetary gears [7], driveshafts [8], ground [9], etc. The engine is partially or completely decoupled from the driving cycle. In addition, together with clutches and brakes, this configuration allows flexible and diverse operating modes, combining the advantages of the other two configurations. In addition, from the global automotive market, many HEVs launched by Toyota, Honda, and GM are of this configuration, reflecting its advantages and value in improving the powertrain efficiency.

1.3 LITERATURE REVIEW

Due to the limitation of battery capacity, the energy management strategy (EMS) of hybrid powertrains needs to coordinate the motor to balance the engine load on the one hand, but also to balance the state of charge (SoC) of the battery on the other hand, namely charge sustaining (CS). HEVs can further evolve into plug-in hybrid electric vehicles (PHEVs) by increasing the battery capacity and adding the charge depleting (CD) mode. Due to the simplicity and usability of this CD+CS mode, it has been widely used in practical engineering applications for PHEV energy management as well [10, 11]. Nevertheless, the CD+CS mode still has some shortcomings. For example, some research cases show that there is still a fuel economy gap of about 22% compared to the deterministic dynamic planning algorithm (with known operating conditions) [12], and the long-term CD mode may lead to a reduction of motor efficiency in driving conditions with high power demand [13]. As a result, many studies have also explored the blended mode (BM) [14–16], where both the engine and motor are working collaboratively during the whole trip. However, the tuning of BM strategy needs to be coordinated with the length of a trip, otherwise its effect may still be inferior to a well-tuned CD+CS strategy [17, 18]. The above is an overview of the basic modes of HEV and PHEV energy management, and the comparison of each mode is shown in Figure 1.2.

According to recent research, energy management methods can be generally classified into three categories: rule-based methods, optimization-based methods, and learning-based methods, as described in the following literature review.

1.3.1 THE RULE-BASED EMS

Rule-based EMSs mainly improve the energy efficiency of the whole vehicle from the following aspects: (1) start the engine by using electric motor to avoid the engine working in inefficient operation areas; (2) reduce idle fuel consumption by adopting anti-idling technologies; (3) adjust the engine operation point by electric motors and generators to improve the engine efficiency; and (4) recover braking energy by adopting regenerative braking.

Rule-based EMSs can be divided into two categories: deterministic rule-based strategies [19] and fuzzy rule-based strategies [20, 21]. Deterministic rule-based EMSs calculate

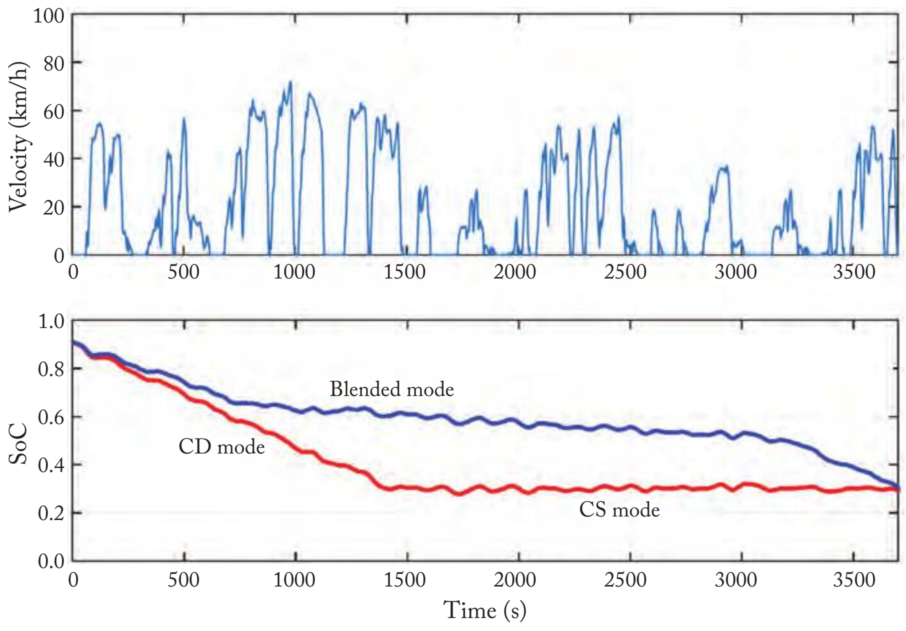


Figure 1.2: Comparison of basic energy management modes.

control signals and operating mode demands by pre-defined rules and thresholds. The fuzzy rule-based EMS incorporates expert knowledge to fuzzify the control rules, which improves the tunability and robustness of rule-based methods. There are two main approaches to obtain the control parameter settings in a rule-based strategy. Usually, they can be extracted from the optimal strategy under specific driving cycles [19]. They can also be optimized using optimization algorithms, for example, using hybrid genetic algorithms to find the optimal control parameters [22].

In general, rule-based EMSs are easy to implement and simple to understand, so they have a wide range of applications in practice. But its application effect is prone to be limited by specific driving cycles, making it difficult to achieve the optimal fuel economy.

1.3.2 THE OPTIMIZATION-BASED EMS

Optimization-based energy management usually uses optimization algorithms to calculate optimal or suboptimal strategies in the feasible domain, with basic elements including control system models, optimization objectives, and constraints. Although optimization-based methods can achieve better fuel economy, there are also some challenges: high complexity of the algorithm, large computational load, etc., making its practical application relatively more difficult [23]. This category of methods is also one of the major research topics in hybrid system control. Under the simulation environment, there are many good optimization-based methods. However, the advantages and disadvantages of each method cannot be easily summarized in a simple way, because their performance can be influenced by many factors such as sampling time, model accuracy, and parameter definition in the simulation environment, etc.

Clara Marina Martinez et al. classified the main optimization-based EMSs into eight categories according to the types of algorithms, including dynamic programming (DP), equivalent consumption minimization strategy (ECMS), model predictive control (MPC), derivative-free algorithms, neural networks, game theory, sliding-mode controller, convex programming and analytic solutions [23]. Here, we briefly summarize the current status of research on each type of method according to their degree of optimization, including instantaneous optimization, local optimization, approximate optimization, and global optimization, as follows.

ECMS is a representative method among instantaneous optimization strategies. Its original idea was to convert the electric consumption of HEVs into the equivalent of fuel consumption to solve the optimization problem [24]. Later, it was proved that ECMS is based on Pontryagin's Minimum Principle (PMP) [25]. Usually, the ECMS adopts a pre-defined equivalence factor to meet the real-time computing requirements of online applications and unknown driving cycles with partial sacrifice of fuel economy. However, the PMP-based energy management can also obtain an optimal solution that is similar to the global optimum given appropriate coefficients [26].

Among the local optimization strategies, i.e., optimization within a look-ahead window, two representative methods are adaptive equivalent consumption minimization strategies (A-ECMSs) [27–29] and MPC-based EMSs [30–32]. They both utilize some observable or predictable future driving information in the optimization process. The difference is that the former adopts this information to find an optimal equivalent factor for the calculation of control actions, while the latter uses driving profiles within the look-ahead window to pre-plan the optimal energy distribution scheme. Both of them can perform well in simulation environments, but their optimization is likely to be affected by the accuracy of future driving information, which is a common issue encountered in practical applications of energy management and remains a research focus.

For approximate optimization strategies, several typical methods can be classified into this category, including heuristic methods (such as particle swarm optimization [33], simulated annealing [34], genetic algorithm [35], etc.), neural networks [36], game theory [37], etc. The algorithmic differences between such strategies are relatively large. A detailed literature review of such strategies is available from Clara Marina Martinez [23], Andreas A. Malikopoulos [38], and others.

The global optimization strategy specifically refers to Dynamic Programming- (DP) based energy management. DP is a powerful tool for solving Markov Decision Process (MDP) problems. It can be adopted reliably and extensively for energy management of various HEVs, especially when the entire trip information is known in advance. Usually, the online application of DP methods is limited by two aspects: the premise that the dynamic properties of the environment is known, and the problem of abrupt increase in calculation amount caused by the discretization of state and action space. Considering that DP-based EMSs are difficult to be applied directly to on-board controllers, researchers usually consider them as benchmark strategies for rule extraction [19], effect comparison [8], etc. On the other hand, by leveraging intelligent transportation systems and cloud computing, DP-based EMSs can also be integrated with other methods to improve energy efficiency [12, 39, 40].

1.3.3 THE LEARNING-BASED EMS

The field of machine learning has flourished in the last decade, with many research breakthroughs demonstrating its potential for applications in data mining and complex control problems. Meanwhile, the application of machine learning in HEV energy management, which is referred to as learning-based EMSs here, has also been increasing. Although learning-based EMSs belong to the optimization-based strategies as well, such methods have great potential for future applications in the intelligent networked transportation environment due to their characteristics such as self-learning capability, adaptability of the algorithm, and excellent multi-source and high-dimensional data mining capabilities.

Learning-based EMSs mostly belong to near-optimal strategies. According to the role of machine learning in energy management, these methods can be briefly described as follows.

One of the early applications of machine learning in energy management is to utilize it for optimization of rule-based strategy parameter, such as the heuristic random search method [33] and the gradient-based optimization method [21], both of which are offline optimization methods. Similarly, machine learning algorithms can also be used to directly extract optimal strategies to replace the rule-based EMSs [15, 41]. Most of these machine learning algorithms are supervised learning, which are highly dependent on sample data and require comprehensive training sets to achieve reliable generalization capabilities.

Machine learning methods also have wide applications in local optimization strategies, e.g., feature extraction of driving cycles [42, 43], prediction of future driving information [44, 45], and global driving cycle construction [39, 40]. These applications could facilitate A-ECMSs and MPC-based EMSs for adaptability enhancement. Both supervised and unsupervised machine learning algorithms are involved in these applications, but mostly are applied to assist the operation of primary control algorithms. The stability of these methods is more reliable due to their reliance on classical control methods. However, the extension of their application scenarios in complex connected vehicle environments may still be limited, i.e., their ability to process multi-source high-dimensional vehicle and traffic information still needs further research.

In contrast to the aforementioned learning-based EMSs, the strategy training of reinforcement learning-based EMSs does not need labeled data, but is guided by control targets. These algorithms could improve the EMS by directly learning from the state transition data, and realize end-to-end control from observation to energy distribution [46, 47]. Such EMSs could achieve relatively better robustness, because they rely more on the potential association between state observations and action values to improve strategies than on labeled data [48–50]. Meanwhile, another outstanding advantage of such EMSs is that they can be combined with deep learning methods, such as deep neural network (DNN) [51], convolutional neural network (CNN) [52], Structured Control Nets (SCN) [53], etc., to enhance their perception capability on multi-source high-dimensional data, or scalability for control problems with complex continuous action outputs.

SUMMARY

In this chapter we briefly discussed some of the main energy management methods and indicated how learning-based methods begin to gain momentum in HEV energy management. This book aims to discuss one of the most promising learning-based EMSs, namely deep reinforcement learning- (DRL) based energy management, and their applications in different HEV energy management problems. The outline of the book is presented as follows.

Chapter 2 provides some necessary background knowledge of DRL. The reinforcement learning-based energy management and its basic learning theory is introduced, which also serves as the foundation of DRL-based EMSs. This chapter also presents different types of state space and action space in reinforcement learning, along with their applicability in different energy management problems.

Concerning continuous-state and discrete-action energy management problems, Chapter 3 provides an energy management method based on deep action-value learning. Improvements from the network structure, estimation of target Q-value, and the experience replay are described for the stable training. This method is evaluated on a series HEV in terms of learning, optimization, and generalization performance.

For continuous-state and continuous-action energy management problems, Chapter 4 mainly discusses a solution based on policy-gradient learning. To improve the adaptability of this continuous EMS for PHEVs, an SoC planning method using history trip information is integrated to guide the strategy learning. It is also compared with the real-time EMSs based on MPC to examine its optimality.

Regarding energy management problems with multiple discrete actions and continuous actions, Chapter 5 provides a learning-based method to search optimal EMSs in hybrid action space. A pre-training stage is also introduced to leverage the empirical knowledge about the optimal EMS. Besides, the value of terrain information in learning-based EMSs is discussed in specific scenarios, the fuel economy performance, and the hierarchical decision extraction and interpretation of the parameterized strategy.

To ensure the compatibility with current mainstream development processes of the vehicle control strategies, Chapter 6 gives an online integration scheme for DRL-based EMSs by knowledge distillation and strategy reconstruction in Matlab/Simulink.

Finally, concluding remarks are given in Chapter 7.

CHAPTER 2

Background: Deep Reinforcement Learning

In energy management of HEVs, taking an energy distribution scheme derived from the given EMS will bring in changes of the vehicle state and driving state. Meanwhile, energy consumption of the powertrain occurs simultaneously with the transition of vehicle states. This instantaneous energy (or fuel) consumption and the sum of energy (fuel) it consumes over the future will provide a criterion for judging the strategy performance. Then, a new energy distribution scheme should be calculated according to the current vehicle states to accomplish the energy management. This described process contains main elements including the interaction of the decision maker with the controlled object and the environment it belongs to, the policy (or strategy), states, actions, and costs (or rewards). Because the state transition process of the vehicle shows a distinct Markovian property [54], we will model and formulate the HEV energy management problem based on MDP theory. The general modeling part will be described in this chapter, while the similarities and differences of the modeling process for different energy management problems will be described in the relevant sections of subsequent chapters.

For control problems modeled as MDP, Monte Carlo methods, DP, and reinforcement learning are all available to find optimal strategies, and their idea of finding optimal strategies are grounded in the value function, which is used to describe the magnitude of the potential value of performing a certain amount of action in a given state [55, 56]. Although classical DP methods are usually reliable in finding the global optimal control strategy, however, the premise that all environment information, i.e., the state transfer probability, is known in advance, and its exponentially increasing computational load severely limit its online application. The Monte Carlo method, on the other hand, gets rid of the assumption that the global environmental information is known and solves the optimal strategy only from the empirical data, but its solution efficiency is relatively lower. Reinforcement learning follows a different idea, adopting the value function estimation idea of Temporal Difference (TD) learning. In this way, the idea of iterative estimation of the value function in DP is retained, and the characteristics of Monte Carlo methods that do not require global environmental information are also combined, making reinforcement learning more suitable for the actual control scenarios [56].

In recent years, DRL, which combines the strengths of both deep learning and reinforcement learning, has gained great momentum. It is often used for end-to-end decision control in complex control systems, where an agent is trained to autonomously learn the optimal control strategy from the raw data environment. DRL-related research has achieved remarkable results in many fields, such as computer games [57], the Go games [58], robot control [59], visual navigation [60], autonomous driving [61], etc. Meanwhile, its application cases in HEV energy management are also gradually increasing, showing the feasibility and potential of DRL for application in HEV energy management in the future connected vehicle environment.

In this chapter, the background of DRL algorithm and the idea of HEV energy management based on DRL is described.

2.1 REINFORCEMENT LEARNING IN ENERGY MANAGEMENT

Reinforcement learning is a class of machine learning algorithms designed to solve sequential decision problems, especially MDP problems. In this section, we will focus on reinforcement learning-based energy management problems to explain the MDP model of energy management and its basic learning theory.

The goal of reinforcement learning is to find a control strategy that could achieve maximum cumulative rewards over the future by mapping the states to actions, as Equation (2.1) describes. For HEVs, the controlled object is hybrid powertrain systems. The state $s(t)$ at moment t could be represented by powertrain state, driving environment, driving demands, etc. The energy distribution scheme is the action $a(t)$ at moment t. The reward r can be evaluated by real-time evaluation metrics of the energy distribution scheme, such as instantaneous fuel consumption, SoC deviations, etc. This control strategy is namely the EMS, pi:

$$\pi^* = \underset{a(t) \in A}{\arg\max}\, \mathbb{E}\left[\sum_{t=0}^{N-1} r\left(s(t), a(t)\right) T_s \,\middle|\, s(0) = s_0 \right], \tag{2.1}$$

where π^* denotes the optimal EMS, A denotes the action space, T_s denotes the sampling time (default is 1s in this book), s_0 denotes the initial state, and N is the time sequence length of the finite-step MDP problem.

By means of exploration and feedback mechanisms, the reinforcement learning-based EMS interacts with the vehicle and its driving environment in order to discover and learn the optimal energy distribution scheme, enabling the application of past experience to new scenarios. Figure 2.1 illustrates this interactive learning process, and the main elements of this interaction are described, respectively, as follows.

(1) The Agent and the EMS π: The agent is the core of reinforcement learning and acts as the decision maker. In energy management, it is equivalent to the vehicle controller that controls the energy distribution in real time. The control program of energy management deployed on the controller is the EMS π, which is used to map the decision from state $s(t)$ to action $a(t)$.

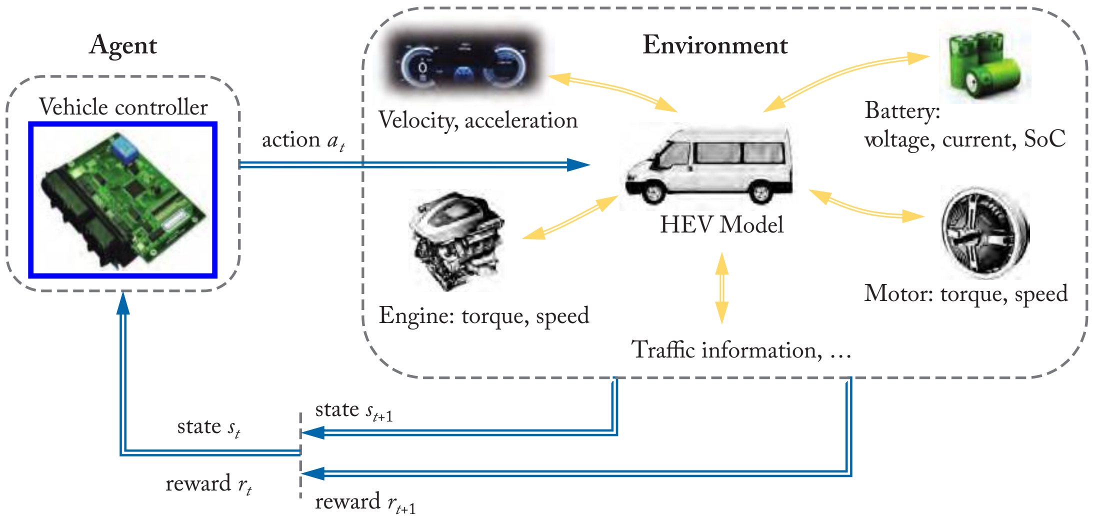


Figure 2.1: The agent-environment interaction.

(2) The environment: The environment in reinforcement learning represents the controlled system, which is the hybrid powertrain in HEV energy management. When the controller (*agent*) sends an action signal, the powertrain will respond to this control signal, and the response can be presented by the instantaneous fuel consumption, change of SoC, etc.

(3) The state space S: The state is a description of the environment the agent is in. The state contains a series of state vectors $s(t)$ ($s(t) \in S, t = 1, 2, \dots, N$) that represent the characteristics of the vehicle state, road and traffic conditions, driving demand, etc. The state vector $s(t)$ should reflect the vehicle state and driving requirement as adequately as possible and be easy to observe. Given that the driving force is mainly determined by vehicle longitudinal dynamics, usually, we could select the velocity v, SOC of the battery, clutch or brake state *clutch*, etc. to describe the state of the vehicle itself. The driving demand could be represented by the desired acceleration a obtained from pedal signal, the calculated driving torque or power demand, etc. The driving cycle information could be obtained from history velocity, such as the velocity during the past seconds: $v_{-1}, v_{-2}, ...v_{-k}$. Road slope θ_{slope} could naturally describe the terrain information. Therefore, a state space that contains the aforementioned states could be expressed as $S = \{v, SOC, a, clutch, T_{axle}, P_{axle}, v_{-1}, v_{-2}, ...v_{-k}, \theta_{slope}\}$. By merging all relevant state data within a certain time window, we can get the state vector $s(t) = [v(t), SOC(t), a(t), clutch(t), T_{axle}(t), P_{axle}(t), v_{-1}(t), v_{-2}(t), ...v_{-k}(t), \theta_{slope}], s(t) \in S$.

(4) The action space A**:** The action refers to the control action of an agent. The action space A is the ensemble of all actions, and it contains all possible action vectors $a(t)$ that determine the energy distribution of the hybrid powertrain: $a(t) \in A, t = 1, 2, \ldots, N$.

(5) The state transition $s \rightarrow s'$**:** The response of the controlled object after executing action a at the current moment. It will make the environment transfer from the current state s to the next state s'. This process is known as the state transfer and exhibits Markovian properties.

(6) The reward R**:** The state transition will generate a reward signal r that evaluates how good the current strategy is. For energy management, the higher the reward is, the better the energy distribution scheme is.

Accordingly, the interaction process shown in Figure 2.1 can be further described as follows: at moment t, the agent makes a decision based on the current state $s(t)$ and sends an action command $a(t)$; then, the environment responds to the action command, undergoes a state shift $s(t) \rightarrow s(t+1)$, and provides a reward feedback $r(t+1) = r(s(t), a(t))$; after that, the agent begins its decision at the next moment $t+1$.

On the other hand, Equation (2.1) shows that the key to find the optimal strategy is how to select an action that could yield a high expected reward return. Thus, in MDP problems, the action-value function is defined to convey the expected reward return over the future after taking action a, as Equation (2.2) shows:

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{N-T} r(s(t), a(t)) \gamma^t \,\middle|\, s(0) = s_T, a(0) = a_T \right], \tag{2.2}$$

where s_T and a_T denote the state and action at moment T, respectively. γ denotes the discount rate, determining the present value of future rewards. When $gamma = 0$, the action value Q is determined by the instantaneous reward only, making the strategy optimization vulnerable to local optimality. When $gamma = 1$, the reward at any moment in the MDP problem is equally important for the decision, but estimating the action value Q in this case will become quite difficult due to the diversity of action vectors and long control sequences in energy management. Referring to some successful applications of DRL [57, 62–65], in this manuscript, the discount rate is set to $gamma = 0.9$ to balance estimation efficiency and convergence.

The optimal strategy π^* has the highest action value, namely the optimal action value $Q^*(s, a)$ as follows:

$$Q^*(s, a) = \max_\pi Q_\pi(s, a). \tag{2.3}$$

If $Q^*(s, a)$ has been obtained, the optimal strategy as shown in Equation (2.1) could be reformulated as:

$$\pi^* = \arg\max_{a(t) \in A} \left[Q^*(s(t), a(t)) \,|\, s(0) = s_0 \right]. \tag{2.4}$$

Generally, according to Bellman Equation, we could reformulate Equations (2.3) and (2.4) into multiple single-step decisions to solve the optimal decision process, as depicted by

Equation (2.5):

$$Q^* (s(t), a(t)) = \mathbb{E}\left[r (s(t), a(t)) + \max_{a(t+1)\in A} Q^* (s(t+1), a(t+1))\right]. \tag{2.5}$$

The goal of strategy learning in reinforcement learning is consistent with the solution of Bellman Equation. Taking the value-based reinforcement learning approach as an example, the action-value function is iteratively updated along with the agent-environment interaction: guided by a random actual action value *Target*, the estimated action value *Estimate* is updated iteratively by a given step *StepSize* until it converges, i.e., the estimation error *Target* – *Estimate* converges to zero, as shown in Equation (2.6):

$$Estimate_{new} \leftarrow Estimate_{old} + StepSize\,(Target - Estimate_{old}). \tag{2.6}$$

However, to update the action value directly by Equation (2.2) requires traversing the entire control sequence again and again, which makes the solving process quite inefficient. Therefore, in reinforcement learning, the idea of TD update is commonly adopted to accelerate the learning process of $Q(s, a)$ estimation, as shown in the following Equation (2.7):

$$Q_{\text{new}} (s, a) \leftarrow Q_{old} (s, a) + \alpha \left[\left(r + \gamma \max_{a'} Q_{old} \left(s', a'\right)\right) - Q_{old} (s, a)\right], \tag{2.7}$$

where $[(r + \gamma \max_{a'} Q_{old} (s', a')) - Q_{old} (s, a)]$ is the TD error (δ), $Q_{old} (s, a)$ represents the $Estimate_{old}$, $(r + \gamma \max_{a'} Q_{old} (s', a'))$ represents the *Target*, (s, a) denotes $(s(t), a(t))$, (s', a') denotes $(s(t+1), a(t+1))$, and r denotes $r(s(t), a(t))$.

After solving the Bellman equation by iterative learning, we can finally obtain the optimal EMS $pi*$ by Equation (2.4).

The above is a classical reinforcement learning theory based on the direct estimation of action-value function, which is usually known as Q-learning. Many of the DRL methods share a similar underlying learning philosophy as Q-learning to some extent, but they differ in the modeling and calculation of action-value function, strategy modeling and update methods, etc. Thus, based on the aforementioned action value estimation idea, we will introduce some typical DRL-based energy management methods that apply to different HEV configurations in subsequent chapters.

2.2 THE STATE SPACE AND ACTION SPACE IN REINFORCEMENT LEARNING

State and action space are two essential elements in reinforcement learning, and vary with different types of control problems. In this section, we will illustrate some typical state and action spaces, as well as the curse of dimensionality in MDP problems, by a simplified game, Grid-World.

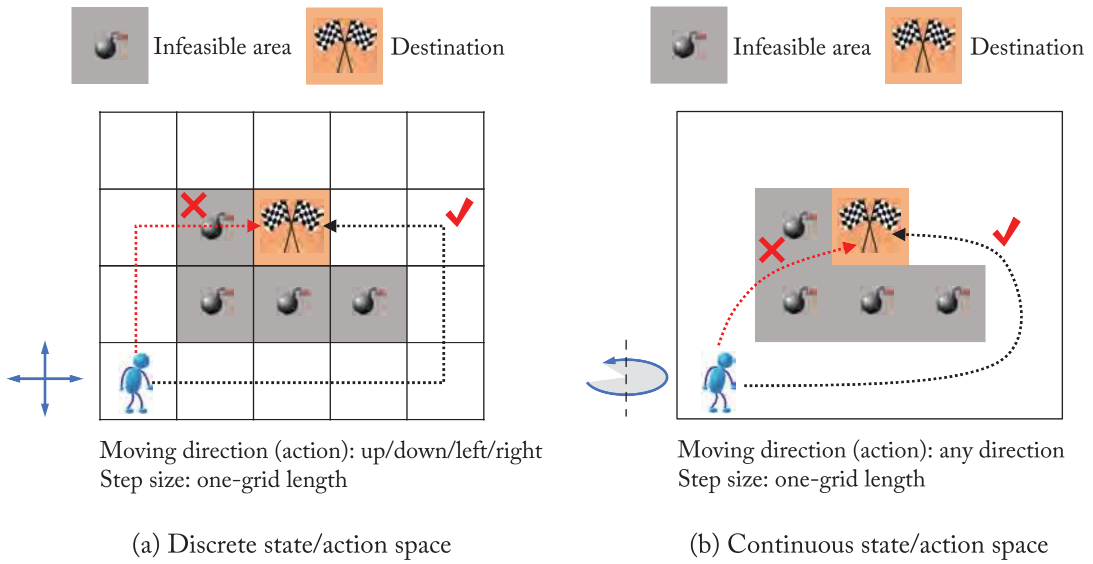

Infeasible area
Destination
Moving direction (action): up/down/left/right
Step size: one-grid length
(a) Discrete state/action space
Infeasible area
Destination
Moving direction (action): any direction
Step size: one-grid length
(b) Continuous state/action space

Figure 2.2: The state and action spaces in simplified GridWorld.

2.2.1 DISCRETE STATE SPACE AND DISCRETE ACTION SPACE

Figure 2.2a shows the simplified GridWorld. In this gridded flat world, the controlled object is manipulated by the strategy to move in one of the four possible directions (up/down/left/right), to go to the adjacent grid. If it enters the infeasible area, it will be punished, while entering the destination, it will earn some rewards. The goal of the game is to get as many rewards as possible in a limited time. As the gridded world is bounded and the object could only move from grid to grid, all possible states of the object could be enumerated as $S = \{s_1, s_2, \ldots, s_{4\times5}\}$, which corresponds to the 4×5 grids of the GridWorld. There are only four optional control actions to move the object, so its action space could be expressed as $A = \{a_1, a_2, a_3, a_4\}$. Such state/action spaces that can be represented by enumeration are classified as discrete state/action spaces.

When applying Q-learning in discrete state/action spaces, we could strictly follow the steps described in Section 2.1 to obtain the optimal strategy. First, calculate the action-value function $Q(s, a)$ so that we could get the action value of any possible action under a given state, as Equation (2.3) shows. Because of the discrete state/action space, the action-value function could be expressed in a tabular form, i.e., by retrieving the specific state and action values in a Q-table, the corresponding action value could be obtained. Then, we can select the action that shows the highest action value as the control output (Equation (2.4)).

2.2.2 CONTINUOUS STATE SPACE AND CONTINUOUS ACTION SPACE

If we change the manipulation rules slightly into Figure 2.2b, where the controlled object could move in any direction, θ from 0° to 360°, the location of the object will not be limited by grids. In this case, it is difficult to list all sets of possible states and control actions by enumeration, thus such state/action space is referred to as continuous state/action space. In this continuous flat world, the state of the control object can be represented by the abscissa value and ordinate value (x and y), so the state space here is referred to as a two-dimensional continuous state space. Taking the lower-left corner of the area as the origin, right and up as the positive x and y direction, respectively, this continuous state space can be written as $S = \{x, y | x \in [0, 5], y \in [0, 4]\}$. Similarly, the continuous action space can be written as $A = \{\theta | \theta \in [0, 360)\}$.

However, as the state and action cannot be described by enumeration, it is hard to express the action-value function in a tabular form when applying Q-learning, and the method of Equation (2.4) cannot be applied directly either. To solve this problem, the continuous state/action space can be discretized into discrete state/action space, which is similar to the discretization in the classic DP algorithm. The higher the degree of discretization (the smaller the discrete scale) is, the more accurate the description of the continuous state/action space will be.

For example, if the continuous plane is discretized by one-tenth of the unit length, we could approximately consider the motion of this object a transition between grids. Then, the state space will be enumerated as $S = \{s_1, s_2, \ldots, s_{40\times50}\}$. Similarly, we can describe the action space approximately as $A = \{a_1, a_2, a_3, \ldots, a_{36}\}$ if it is discretized by 10°. Accordingly, the action-value function can still be represented by a $(40 \times 50) \times 36$ action value table, and solved by Q-learning.

The curse of dimensionality caused by discretization, however, severely limits the application of traditional reinforcement learning methods, such as Q-learning, Dyna, etc. [46]. The curse of dimensionality is a phenomenon used by Bellman in dynamic programming research to describe the exponential increase in the amount of data caused by increasing dimensionality in a high-dimensional space, and the associated difficulties in data analysis, organization, and computation [66]. Although this phenomenon is not obvious in low-dimensional space, Figure 2.3 can still visually describe it: for a one-dimensional unit interval, when the discretization increases from 3 to 9 grids, the sampling and computational load only increase by three times; while when the dimensionality increases to three, the triple increase in discretization level leads to an exponential increase in the sampling and computational load.

Therefore, when discretization is applied to deal with continuous state/action space in Equations (2.3) and (2.4), the computational load will increase sharply and the optimization becomes more and more difficult as the dimensionality or discrete accuracy increases. On the other hand, excessive reduction of discrete accuracy may also lead to poor optimization.

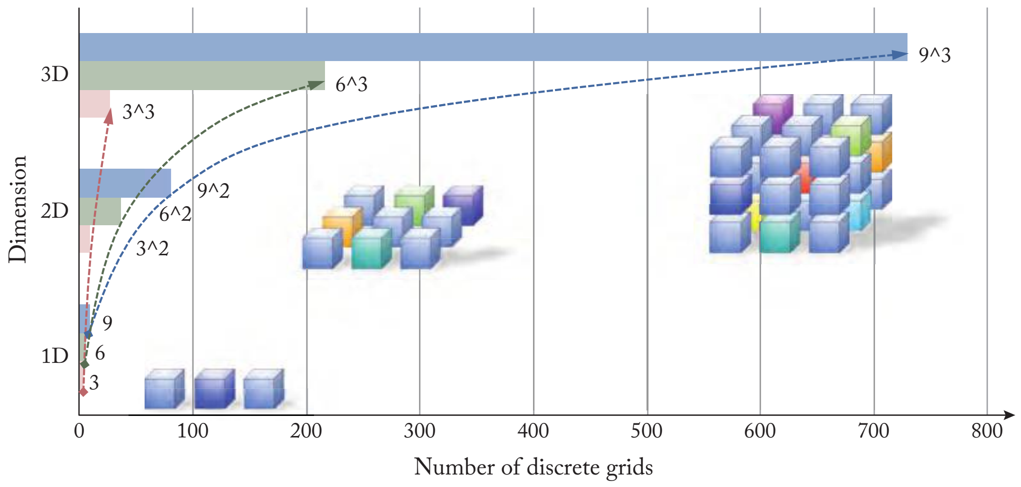


Figure 2.3: Illustration of the curse of dimensionality due to discretization.

2.2.3 HYBRID ACTION SPACE

If the manipulation rules in Figure 2.2b are further changed: while control the moving direction, the strategy should also select a proper step size l from the set $\{l_1 = 0.5, l_2 = 1, l_3 = 1.5\}$. In this case, the action space will contain both continuous actions and discrete actions, denoted as $A = \{l_1, \theta\} \cup \{l_2, \theta\} \cup \{l_3, \theta\}$, where $\theta \in [0, 360)$. Hybrid action space are also widely found in practical control problems, such as HEV energy management problems that includes clutch or brake states, or mode switching, etc.

When solving MDP problems with hybrid action space by the aforementioned reinforcement learning method, it is necessary to discretize the part of continuous actions, convert the hybrid action space into discrete action space, but still, the potential dimensional issue may be encountered.

2.2.4 STATE AND ACTION SPACE IN HEV ENERGY MANAGEMENT

HEV energy management belongs to the sequential decision, which can be modeled as MDP problems. This naturally provides the foundation of applying DRL to formulate an intelligent EMS. However, due to the variety among different types of HEV powertrain configuration, MDP models of energy management also vary in their dimension and scale of state/action space, and the type of action space, etc. which is the reason for carrying out related research work in this book. Some typical cases are as follows.

(1) For series HEVs, as the engine is totally decoupled from the driving cycles, we could constrain the optimal EMS works following a set of pre-defined optimal engine operation

points. Thereby, the action space can be categorized as discrete action space. A relatively small action space also facilitates the effective search of optimal EMS by simple algorithms.

(2) For power-split and series-parallel HEVs, the engine is partially coupled with the driving cycles, and there is more freedom in the powertrain control. This usually leads to an increase in action variables and larger action spaces, so strategy learning algorithms that could search in continuous state/action spaces would be more applicable.

(3) For some coaxial series-parallel HEVs, their working modes are determined by the clutch connected to the engine, thus, the corresponding action space consists of several continuous actions and a discrete action that controls the clutch. For HEV configurations consisting of dual planetary gears, their working modes are determined by multiple clutches and brakes, i.e., the action space contains multiple discrete and continuous variables at the same time. In the former case, the binary discrete action could be selected simply by taking a continuous output action as the probability of closing a clutch. In the latter case, however, when multiple discrete actions exist in hybrid action spaces, it would be more beneficial to explore direct strategy searching approaches.

(4) For PHEVs, due to the wide variation range of SoC, which means a wide feasible state space, random exploration or monolithic rewards may fail in strategy learning. Therefore, it is desirable to leverage some bootstrapping strategies to ensure effective exploration in the state space.

2.3 LITERATURE REVIEW ON DEEP REINFORCEMENT LEARNING

In reinforcement learning, the essential core is that the agent must derive efficient representations of the environment from high-dimensional sensory inputs and use these to generalize past experience to new situations [57]. Although classic reinforcement learning has achieved successful applications in many fields, they generally are more suitable for cases with low-dimensional observable state space where environment features can be easily extracted manually. Their application to complex problems, however, is very limited due to the curse of dimensionality [67].

Meanwhile, deep learning, as a class of machine learning methods based on neural networks that explores the inherent patterns and representations of data, has excelled in complex information processing such as image, text, and speech in recent years [68]. Alex Krizhevsky et al. also point out in their research on computer vision that deep networks trained with large amounts of raw data as input are instead more likely to learn effective representations than manually extracted features [69].

Therefore, Volodymyr Mnih et al. pioneered combining the decision-making ability of reinforcement learning with the perceptual ability of deep learning, and proposed deep reinforcement learning for the first time, providing a brand new idea to bridge the divide between multi-

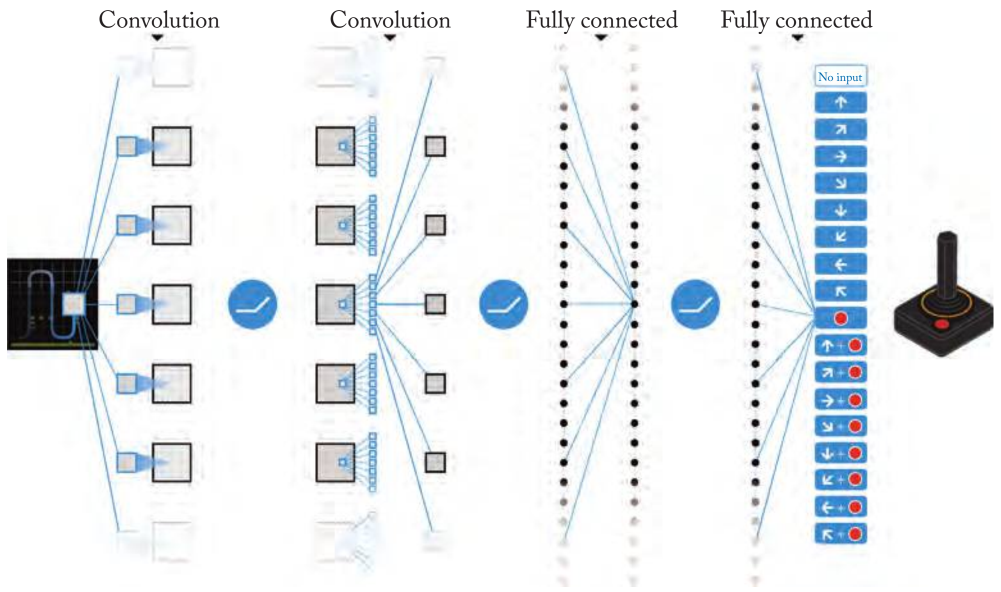


Figure 2.4: Schematic illustration of the DQN used in Atari games [57].

source high-dimensional perceptual signals and control actions. In Atari games, the agents they designed directly learn the mapping relationship from the original RGB image to the action-value function using convolutional neural networks, as shown in Figure 2.4, and achieved a performance that surpassed human players [57, 67]. It is also the first artificial agent that is capable of learning to excel at a diverse array of challenging tasks, which greatly boosts the development of artificial intelligence. Instead of the tabular Q-map in Q-learning, deep neural networks are leveraged in this method to represent the action-value function, namely the Deep Q-network (DQN), without discretizing the continuous state space and avoiding dimensional curse. Take the continuous state space $S = \{x, y | x \in [0, 5], y \in [0, 4]\}$ described in Section 2.2 as an example: if it is discretized as $S = \{s_1, s_2, ..., s_{40\times50}\}$, the tabular action-value function needs to be updated iteratively for each of the 40×50 elements. While using the neural network f to represent the action-value function, on the one hand, the network input dimension only needs to keep consistent with the dimension of continuous state space, which is more practical for the environment with multi-source high-dimensional information. Also, the parameter update of the action-value function $Q = f(x, y)$ can be performed by gradient descent using batch data, making the solution efficiency considerably improved.

Similar to Q-learning, the training of DQN is also based on action value estimation, and the success of DQN also depends to a large extent on the integration of experience replay [57]. Tom Schaul et al. further improved experience replay by assigning priorities to state transition data by their importance, and organizing the storage and replay of empirical data by priority, so as to increase the frequency of more meaningful samples being replayed and improve the effects of the algorithm [70]. To enable DQN to cope with large-scale learning tasks, Dan Horgan et al. further extended this method by proposing distributed generation, Preferential selection of empirical data for independent policy updates [71].

As an action value-based method, Q-learning and DQN are prone to overestimate the Q-value, degrading the training and performance of the strategy [72, 73]. Hado van Hasselt et al. adopt two neural networks for the estimation of Q-value, one for the action selection and the other for action value estimation, to tackle this problem [73]. In some MDP scenarios, the influence of environmental information on Q-value estimation may outweigh action selection, in this regard, Ziyu Wang et al. proposed a new DQN structure to separate the state value estimation and the state-dependent action advantage, which eventually led to a better strategy evaluation [74].

Usually, to avoid strategy learning from falling into local extremes, the exploration mechanism would be introduced to achieve diversity and novelty of state transition data and improve the learning effect. ε-greedy algorithm is a commonly used exploration strategy that contributes to better training results even in complex environments [57]. Bradly C. Stadie et al. evaluated the novelty of states by training predictive models, which encouraged strategies to visit new states more often. Considering the stochastic characteristic of MDP problems and the uncertainty of current action values, Ian Osband et al. proposed a deep exploration strategy (Bootstrapped DQN) to improve the algorithm learning performance by referring to Thompson sampling [75]. In this research, random sampling of Q-value with a certain distribution is simulated by parallel learning of multiple action-value functions, followed by the ε-greedy exploration [75].

Due to the relatively large (task-specific) DQNs and their extensive training, policy transfer and integration methods in DRL are also evolving to solve complex problems. The policy distillation method proposed by Andrei A. Rusu et al. aims to extract single-game policy or consolidate multiple policies into a single policy, and outperforms the previous agents [76]. Similarly, Emilio Parisotto et al. combine imitation learning and knowledge distillation to train agents, which perform equally well in hybrid tasks and the learning efficiency is improved in new environments as well [77, 78].

Apart from action value-based DRL, policy gradient-based DRL methods are also widely studied. Policy gradient-based DRL directly represents the strategy as a parametrized representation, such as neural networks, and searches the optimal strategy by gradient descent. Thus, similar to DQN, policy gradient-based methods can search the optimal policy in continuous action space directly without discretization.

The policy gradient approach can be generally classified into stochastic and deterministic policy. The stochastic one outputs the probability distribution of the action, while the latter directly outputs the deterministic action value [79, 80]. Compared with the stochastic policy, deterministic policies are slightly weaker in action space exploration, but the policy is improved directly in the gradient direction of increasing the action value, boosting the learning efficiency;

however, the deterministic policy gradient approach can be considered as a special case where the variance of the stochastic policy converges to zero [80].

Among stochastic policy methods, John Schulman et al. proposed the Trust Region Policy Optimization (TRPO) method, which utilizes neural networks to model the stochastic policy, and bound its parameter updates to a trust region to ensure monotonic improvement [81]. Although TRPO shows good generality, its data utilization efficiency is relatively low, and Open AI further proposed the Proximal Policy Optimization (PPO) method, which proposes two algorithm implementation schemes, clipped surrogate objective and adaptive KL penalty coefficient, respectively, with more concise process and better performance [63].

Among deterministic policy methods, Timothy P. Lillicrap et al. proposed the deep deterministic policy gradient (DDPG) by combing Actor-Critic architecture and the idea of DQN [62]. In DDPG, the off-policy update is adopted and it performs quite well and stably in many complex environments without fine-tunning. To further improve the data utilization efficiency, DeepMind proposes a DDPG method that can set up a configurable number of strategy updates, which effectively accelerates the strategy learning process by appropriately increasing the number of strategy updates during each agent-environment interaction [82]. Mel Vecerik et al. efficiently solved the MDP problem with sparse rewards by combining expert demonstration data samples in the DDPG method [83].

In addition, considering that solving MDP problems using DRL requires a large number of iterative learning, especially when high-dimensional information inputs such as images are involved, GPUs are usually required for acceleration, so asynchronous settings have been increasingly applied to diverse DRL algorithms. By learning asynchronously and in parallel in multiple environments, this allows us to discard the need for memory space for sample databases in off-policy training. With asynchronous measures, strategies trained on multi-core CPUs alone can also meet or exceed the performance of those trained on GPUs. Currently, most poplar DRL algorithms can work well in an asynchronous learning way, such as asynchronous DQN [84], asynchronous Actor-Critic [84], asynchronous PPO [85], etc. In general, parallel learning for asynchronous methods is also available in two different ways as follows. (1) Learn the policy gradient or the gradient of the action-value function in parallel, then complete the global policy update by asynchronous neural network update methods [84]. (2) Generate state transition data via agent-environment interactions in multiple environments in a distributed manner, then update the global policy with reasonable experience replay methods [71].

Based on the above two types of DRL algorithms, researchers have also proposed many improvements for various decision task types and scales, such as hierarchical reinforcement learning [86], meta reinforcement learning [87], transfer learning-enabled reinforcement learning [88], multi-agent reinforcement learning [89], etc. Meanwhile, with the development of DRL, research on their applications in different fields has been increasing as well. In robot control, Jemin Hwangbo et al. trained a control strategy for legged robots using the TRPO algorithm and achieved top-quality control effects: as robots dynamics are required in agent-environment interaction, to accelerate the training process, they trained an actuator network that models complex actuator/software dynamics in advance to avoid training with real robots, and used it for strategy training [59]. OpenAI constructed a strategy network combining long short-term memory networks and trained it by PPO and generalized advantage estimation, which for the first time defeated a human professional in Dota 2 game which features long time horizons, partially observed state, and high-dimensional action and observation spaces [90]. Deep reinforcement learning is also widely used in the energy field, for example, in heating, ventilation, air conditioning and domestic water supply systems, new energy vehicles and HEVs, and distributed generation and electrical storage, where such methods have been extensively researched and applied [46].

CHAPTER 3

Learning of EMSs in Continuous State Space–Discrete Action Space

When applying Q-learning or DP for energy management, their solving efficiency may be limited by the premise of state space discretization. For data-driven, end-to-end learning-based EMSs, we desire not only to reduce their reliance on empirical parameter tuning, but also a higher requirement for its data mining capability, i.e., the energy-saving control schemes should be learned quickly from multidimensional environmental information. The DQN method, as an early breakthrough in DRL, combines deep learning with Q-learning to construct action-value functions and achieve continuity of state space while solving MDP problems, which dramatically expands the application of reinforcement learning in complex environments. On the other hand, DQN directly traverses and searches the entire action space to find the maximum action value when making decisions, such a strategy enables a fairly concise and efficient learning algorithm in discrete action spaces. Therefore, to address energy management problems with continuous state—discrete action spaces, this chapter describes an energy management method based on deep Q-learning, and further conduct research on its learning stability, optimization, and adaptability on diverse driving cycles.

3.1 ENERGY CONSUMPTION MODEL OF A SERIES HYBRID ELECTRIC VEHICLE

In this chapter, we take a series-HEV (SHEV) with a wheelbase of 2.65 m as the research case (Figure 3.1), whose parameters are provided in Table 3.1. The vehicle is impelled by two identical electric propulsion systems; two power sources are equipped onboard: the battery pack and the auxiliary power unit that consists of the engine and generator. Since the main focus here is energy management, it is assumed that drive forces between two axles are evenly distributed.

In DRL-based energy management, the energy consumption model of this SHEV should not only quickly provide accurate real-time energy consumption assessment, but also act as an environment for the interactive learning process shown in Figure 2.1, simulating state transitions of the vehicle powertrain in real time. Therefore, we will establish the power request model of

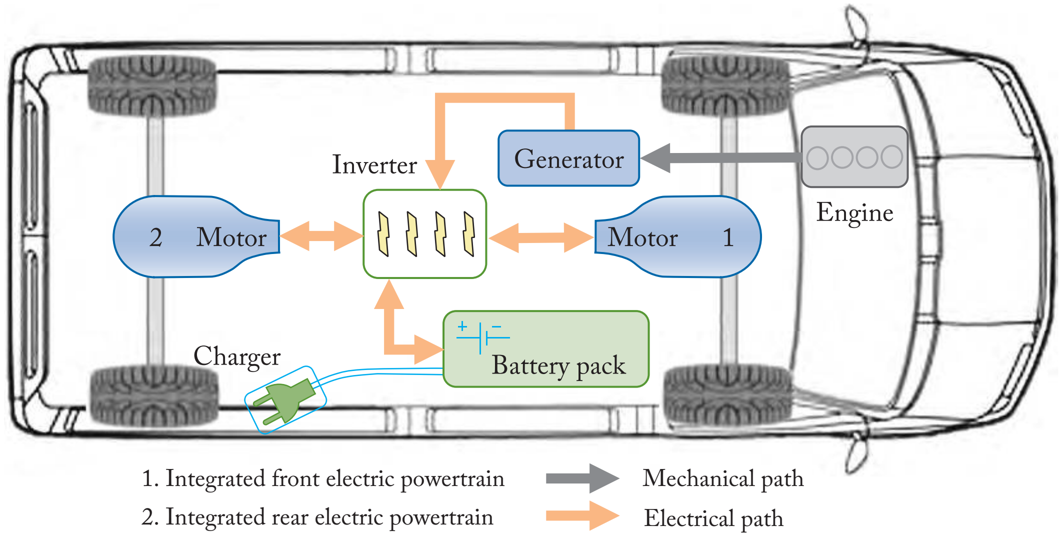


Figure 3.1: Structure of the SHEV configuration.

Table 3.1: General parameters of the SHEV

Vehicle	Curb weight Front area Front/rear final drive ratio Rolling radius	3500 kg 3.9 m^2 5.857 0.447 m
Engine	Maximum power/speed Maximum torque/speed	62 kW/3500 rpm 227 Nm/1900 rpm
Generator	Rated/maximum speed Rated/maximum torque	2400/4000 rpm 118/277 Nm
Front/rear motor	Rated/maximum speed Rated/maximum torque	2800/7200 rpm 170/320 Nm
Battery pack	Capacity Voltage	25 Ah 347.8 V

the SHEV, APU modeling, battery modeling, and motor modeling, respectively, to complete the control-oriented energy consumption model as follows.

(1) Power request model of the SHEV

Energy management mainly deals with the power allocation, so the propelled power becomes crucial. The vehicle longitudinal dynamics is adopted to calculate the driver's request power P_{req} by the force balance equation (Equation (3.1)), where driving resistance of this SHEV mainly consists of four parts: inertial force F_j, rolling resistance F_f, resistance due to road slope F_i, and aerodynamic drag F_w:

$$\begin{cases} P_{req} = F_{req} v \\ F_{req} = F_j + F_f + F_i + F_w \\ F_j = m a_{acc} \\ F_f = mgf \cos\theta_{road} \\ F_i = mg \sin\theta_{road} \\ F_w = C_D A_{front} v^2/21.15, \end{cases} \tag{3.1}$$

where F_{req} denotes the request driving force, m denotes the curb weight (kg), a_{acc} denotes the acceleration (m/s^2), θ_{road} denotes road slope which is not considered in this chapter, C_D denotes aerodynamic coefficient (0.65), A_{front} denotes the fronted area of vehicle (m^2), v denotes the velocity (m/s), f denotes the rolling coefficient, and g denotes the acceleration of gravity.

To guarantee the drivability, energy management needs to coordinate the power output of the battery pack and the APU while satisfying P_{req}:

$$P_{req} = (P_{batt} - P_{APU}) (\eta_{inv}\eta_{mot})^{\mathrm{sgn}(P_{req})}, \tag{3.2}$$

where P_{batt} denotes terminal power (discharge/charge) of battery, P_{APU} denotes the output power of generator, η_{inv} denotes the efficiency of inverter, and η_{mot} denotes the efficiency of motors; assume that regenerative braking is fully adopted.

(2) APU modeling

Assuming the APU can respond quickly when receiving control signals, a quasi-static fuel and electricity consumption model are built by efficiency maps (Figure 3.2). The torque and speed transfer between the engine (T_{eng}, W_{eng}) and generator (T_{gen}, W_{gen}) are described by the following torque balance equation:

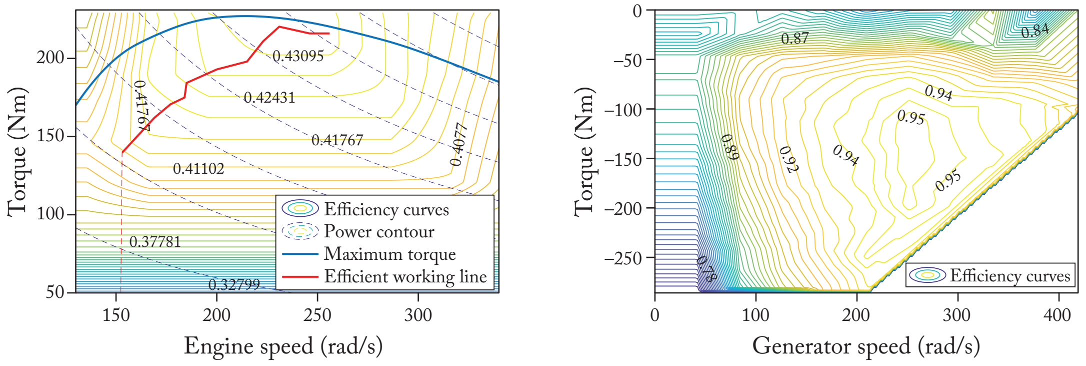
$$T_{gen} = T_{eng}\ , \quad W_{gen} = W_{eng}. \tag{3.3}$$

Terminal power of the APU (P_{APU}, W) and fuel consumption rate of engine ($\dot{m}_{fuel}$, kg/s) are:

$$\begin{cases} P_{APU} = T_{gen} W_{gen} \eta_{gen} \\ P_{eng} = T_{eng} W_{eng} \\ \dot{m}_{fuel} = P_{eng} / \left(D \eta_{eng}\right), \end{cases} \tag{3.4}$$

where η_{eng} and η_{gen} denote the efficiency of generator and engine and D denotes the gasoline lower heating value (4.25×10^7 J/kg).

On the other hand, considering the APU is decoupled with driving cycles, an efficient operation trajectory is predefined to ensure its working efficiency as shown in Figure 3.2a. In practice, given the request engine power P_{eng}, the controller will obtain corresponding engine



(a) Efficiency map of the engine for fuel consumption

(b) Efficiency map of the generator for electricity consumption

Figure 3.2: Efficiency map of the APU.

torque and speed on this trajectory by mapping: $[T_{eng}, W_{eng}] = find(P_{end})$. Meanwhile, the engine torque and speed must satisfy the boundary constraints as follows:

$$T_{eng}^{\min} \leq T_{eng} \leq T_{eng}^{\max}\ , \quad W_{eng}^{\min} \leq W_{eng} \leq W_{eng}^{\max}. \tag{3.5}$$

(3) Motor modeling

Since we mainly focus on the energy flow of the powertrain, a quasi-static electricity consumption model is developed for the drive motor using efficiency maps obtained from bench tests as well. The relationship between the total power of drive motors and the driving power request is shown in Equation (3.6):

$$P_{mot} = P_{req}/\eta_{mot}^{\operatorname{sgn}(P\_req)}, \tag{3.6}$$

where the motor efficiency η_{mot} is obtained from the efficiency map.

(4) Battery modeling

The equivalent circuit model, consisting of internal resistance and open-circuit voltage (OCV) as Figure 3.3 shows, is adopted for battery package modeling. Because the equivalent circuit model has been adopted in massive research of vehicle energy management based on both optimization algorithms [45, 91–93] and learning methods [41, 94–96], this model is considered to be sufficient for our current research. However, when reliable battery test data is accessible, the DRL methods can handle more sophisticated battery models as long as reasonable SoC values could be updated.

The battery power and SoC are derived according to the voltage balance and power balance as follows:

$$P_{batt} = U_{oc}\,(SoC)\,I_{batt} - I_{batt}^2 R\,(SoC) \tag{3.7}$$

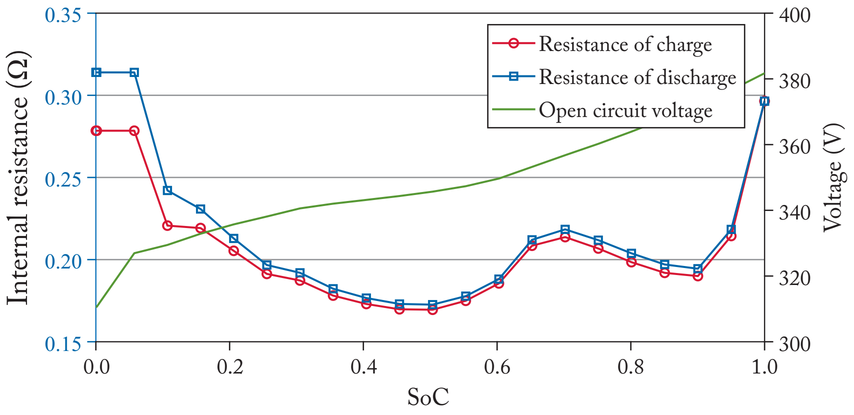


Figure 3.3: Basic parameters for the battery model.

$$\dot{SoC} = -I_{batt}/Q_{batt}, \tag{3.8}$$

where $U_{oc}(SoC)$ denotes the OCV of battery pack, I_{batt} denotes the battery current, $R(SoC)$ denotes the internal resistance, and Q_{batt} denotes the battery capacity.

3.2 ENERGY MANAGEMENT BASED ON DEEP Q-LEARNING METHOD

3.2.1 DISCRETE ACTION SPACE IN ENERGY MANAGEMENT

In this section, a charge sustaining (CS) strategy is considered in energy management as an example to demonstrate the DQN-based EMS, i.e., the driving energy comes from engine fuel consumption, while the battery is mainly used to compensate or absorb the transient energy changes.

The formulation of energy management is consistent with the general modeling approach in Section 2.1, including the agent and EMS π, optimization goal of finding the optimal strategy that achieves the maximum expected reward return, as shown in Equation (2.4). The SHEV model described in the last section, denoted by F, represents the environment that the EMS interacts with. The environment-agent interaction is as follows:

$$\begin{cases} [s', \ r] = F\,(s, \ a|a \sim \pi) \\ s = s'. \end{cases} \tag{3.9}$$

The definition of reward function r is closely related to the goal of energy management. To consider the fuel economy and the SoC charge sustaining constraints $SoC(t_f) \approx SoC(t_0)$, the reward function is defined as Equation (3.10):

$$\begin{cases} r = r(s,a) = -\tanh\left(\chi \dot{m}_{fuel} + \varphi \left|\Delta SoC_{CS}\right|\right) \\ \Delta SoC_{cs} = SoC - SoC_{sust}, \end{cases} \tag{3.10}$$

where SoC_{sust} denotes the expected CS level, φ denotes factor of SoC deviation and χ denotes factor of fuel consumption rate. To punish frequent engine start/stop, a penalty is imposed to the reward with a constant penalty factor ρ (a positive real number) whenever the engine starts: $r = r - \rho$.

According to the SHEV model, we select the following features as the state variables: SoC, engine power P_{eng}, current velocity v, the desired acceleration of the driver acc, the deviation between current SoC and the CS level ΔSoC_{CS}. Then, we can write the state space as Equation (3.11). Each dimension of state in this five-dimensional state space S is a continuous variable, and thus belongs to the continuous state space. Solving this MDP problem by discretization may lead to an excessive computational load, and relative studies also point out the weaknesses of using discrete methods like DP [97] or Q-learning [98] in energy-saving control of vehicles. Therefore, based on the well-demonstrated DQN algorithm [57], this problem could be addressed by using neural networks to construct action-value function $Q(s,a)$, i.e., replace tabular mapping with parameterized function mapping to avoid dimensional catastrophe problem:

$$\begin{cases} S = \{SoC, P_{eng}, v, acc, \Delta SoC_{CS} | SoC \in [0,1], P_{eng} \in [P_{eng}^{\min}, P_{eng}^{\max}], \\ \qquad v \in [0, v^{\max}], acc \in [acc^{\min}, acc^{\max}], \Delta SoC_{CS} \in [-1,1]\} \\ s = \left[SoC(t), P_{eng}(t), v(t), acc(t)\right], \quad s \in S. \end{cases} \tag{3.11}$$

Because the power demand is determined by the driver or driving cycles, the powertrain state can be uniquely determined by giving the APU output power, i.e., APU output power is the control action a for this energy management problem. Considering there is a predefined efficient working trajectory of the engine and the continuous operation of actual engine systems, we could take ΔP_{eng} , the increment or decrement value of P_{eng}, as the control action. The discrete action space can be expressed as Equation (3.12), where *Stop* means the engine will be shut off:

$$\begin{cases} A = \begin{Bmatrix} a_1 = 10\text{ kW}, a_2 = 5\text{ kW}, a3 = 1\text{ kW}, a_4 = 0.4\text{ kW}, \\ a_5 = 0\text{ kW}, a_6 = -1\text{ kW}, a_7 = -5\text{ kW}, a_8 = \text{Stop} \end{Bmatrix} \\ a \in A. \end{cases} \tag{3.12}$$

3.2.2 LEARNING THEORY OF DQN-BASED EMSs

Deep learning is the key to the development of DRL, but also the key to the construction of continuous action-value function in this book. The tutorial of Stanford [99] provides detailed background knowledge of feedforward multi-layer neural network as a reference. On the basis of DNNs, the DQN-based energy management method is introduced as follows.

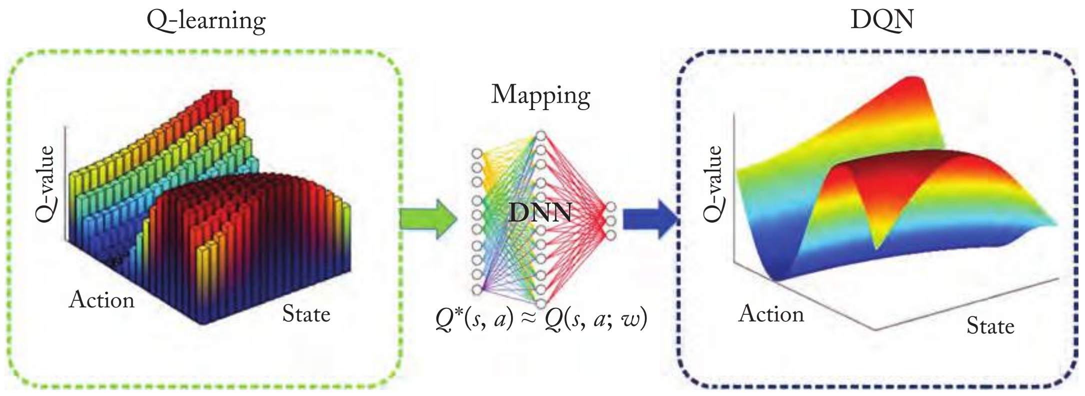


Figure 3.4: The evolution from Q-learning to DQN.

Deep Q-learning, as a classical DRL algorithm, also employs action-value function estimation as the basis for action selection, but it introduces deep learning to replace the previous tabular expression, avoiding the discretization of continuous state space [57, 67]. The evolution from Q-learning to DQN is shown in Figure 3.4.

In this chapter, DNN is adopted to construct DQN as a mapping function from state-action to action value of energy management. Furthermore, if there is a need to deal with more complex driving environment information in energy management, such as on-board vision information, CNNs can also be used for DQNs, as shown in Figure 2.4. In the constructed DQN, the number of neurons in the input layer is 5, which is consistent with the dimensionality of the state space S. There are three fully connected hidden layers with the rectified linear unit (ReLU) as activation functions. The number of neurons in the output layer is 8, which is consistent with the action space A dimensions, and linear function $f(x) = x$ is chosen as the activation function.

Denoting all parameters in DQN as θ^Q, the action-value function of energy management can be expressed as $Q(s, a|\theta^Q)$, and the corresponding EMS will be $\pi = \arg\max_a Q(s, a|\theta^Q)$.

With the well-defined state space S, action space A, and reward r, the action-value function can be calculated by iterative Equation (2.7). However, in contrast to the update by direct value replacement in Equation (2.7), the parameter set θ^Q of deep Q-value function is updated by gradient descent in order to achieve a gradual convergence of the DQN output values with the actual action value estimates. Thus, the TD error can be defined as the loss function L_Q for DQN training, as shown in the following equation:

$$L_Q(\theta^Q) = \left[\left(r + \gamma \max_{a'} Q(s', a'|\theta^Q)\right) - Q(s, a|\theta^Q)\right]^2. \tag{3.13}$$

Based on gradient back propagation, the gradient of L_Q with respect to DQN parameters can be obtained by Equation (3.14):

$$\nabla_{\theta^Q} L_Q(\theta^Q) = 2\left[\left(r + \gamma \max_{a'} Q(s', a'|\theta^Q)\right) - Q(s, a|\theta^Q)\right] \nabla_{\theta^Q} Q(s, a|\theta^Q). \tag{3.14}$$

Then, with $\nabla_{\theta^Q} L_Q$, the DQN can be updated by Equation (3.15). ADAM algorithm is adopted for adjustment of learning rate α_l with default parameter settings [100]:

$$\theta^Q = \theta^Q + \alpha_l \nabla_{\theta^Q} L_Q(\theta^Q). \tag{3.15}$$

The above is the learning theory of DQN-based EMS. Different from the regular supervised learning, the training of DRL is not only to minimize the loss function, but also to observe whether the new strategy can obtain higher cumulative reward return, so as to control the training process comprehensively. Theoretically, with all the following conditions satisfied, we can consider the training to be converged. (1) The action space is reasonably chosen. (2) The reward r can reasonably evaluate the effect of commanded actions. (3) The cumulative reward return continuously increases and gradually stabilizes during training. (4) There are no large fluctuations in loss values. At this point, the DQN can be used as a mapping from the vehicle state to the action value of energy management, then the corresponding EMS π can be obtained.

3.2.3 TRAINING OF DQN-BASED EMSs

Section 3.2.2 introduces the learning theory of EMSs based on DQN, which is basically still supervised training, but as a temporal control problem, the training samples and labeled data need to be obtained during agent-environment interaction. Therefore, this section will describe the training method of DQN-based EMSs.

According to Equation (3.14), the gradient computation requires data including the current state s, current action a, reward r after taking action a, and the next state s', which together make up the entire data involved in a state transition. Usually, they are stored as an experience in the form of a tuple, $e = (s, a, s', r)$. In practice, however, if online updates are applied, i.e., a DQN update is performed every time an experience tuple is obtained, but it is likely to lead to training failure. This is because in temporal control problems, not only does the strong correlation between successive state transfer data may lead to inefficient learning, but the approach also leads to a poor feedback pattern in training: the current network parameters directly determine the experience data used for the next parameter update [57].

To avoid updating DQN exclusively from the latest experiences, experience replay is incorporated in the training of DQN-based EMS [57, 101]. The temporal experience samples e are stored by a circular queue into an experience pool $\mathcal{D}$ (sample capacity $size = 10^4$). One update of DQN will require sampling a minibatch (size of 32) of experience samples from $\mathcal{D}$,

$(s, a, s', r) \sim U(\mathcal{D})$, then calculate update gradient of DQN by Equation (3.16):

$$\nabla_{\theta^Q} L_Q(\theta^Q) = \mathbb{E}_{(s,a,s',r)\sim U(\mathcal{D})}\left[2\left(r + \gamma \max_{a'} Q(s', a'|\theta^Q) - Q(s, a|\theta^Q)\right)\nabla_{\theta^Q} Q(s, a|\theta^Q)\right]. \tag{3.16}$$

With experience replay, not only can DQN updates be synchronized with the agent-environment interaction, but also the correlation between the experiences used for updates is weakened, making it possible to construct action value mappings with deep learning. Meanwhile, experience replay also allows experience samples to be learned repeatedly, which indirectly improves the efficiency of data utilization for strategy learning.

On the other hand, when constructing a DQN, the network parameters need to be initialized first. To keep the input and output variance of each layer of the network as consistent as possible and to ensure the transferability of network information, the Xavier initialization [102] is used here. However, the EMS corresponding to the initialized DQN is merely a randomly generated strategy, and the experience data generated by it is obviously inadequate for strategy learning. Therefore, it is necessary to introduce exploration policies in reinforcement learning. Here, ε-greedy exploration is adopted. A random process $\mathcal{R}$, which will generate random actions when called, is defined first. Then, explore the action space with exploration probability ε, i.e., randomly selecting actions $a = \mathcal{R}(s)$, while selecting actions with probability $1 - \varepsilon$ utilizing the greedy strategy of maximizing the action value ($a = \arg\max_a Q(s, a|\theta^Q)$). The exploration probability ε decreases gradually as strategy training proceeds.

In summary, the training process of DQN-based EMSs can be depicted in general by Figure 3.5, where the DQN is continuously updated with the agent-environment interaction, and the agent (EMS) is always aligned with the updated DQN to iteratively learn the optimal energy management strategy. When training is completed, the latest parameter set θ^Q of the DQN is saved and we will get the learned EMS $\pi = \arg\max_a Q(s, a|\theta^Q)$.

3.3 IMPROVEMENTS FOR STABLE LEARNING IN DISCRETE ACTION SPACE

Although deep Q-learning provides an effective approach for solving energy management problems with continuous sate space—discrete action space, its learning stability is still quite sensitive to the tuning of parameters, training data, and reward definition, due to the complexity of the temporal control problem, the trial-and-error search mechanism, and the highly nonconvex characteristics of DNN itself. Therefore, according to some successful research results in Deep Q-learning, this section will introduce how to construct a stable training method for such energy management methods from three aspects, including the optimization of DQN structure, the improvement in action value estimation, and the enhancement of data efficiency, respectively. For the sake of clarity, the DQN described in Section 3.2 is referred to as the original DQN in later sections.

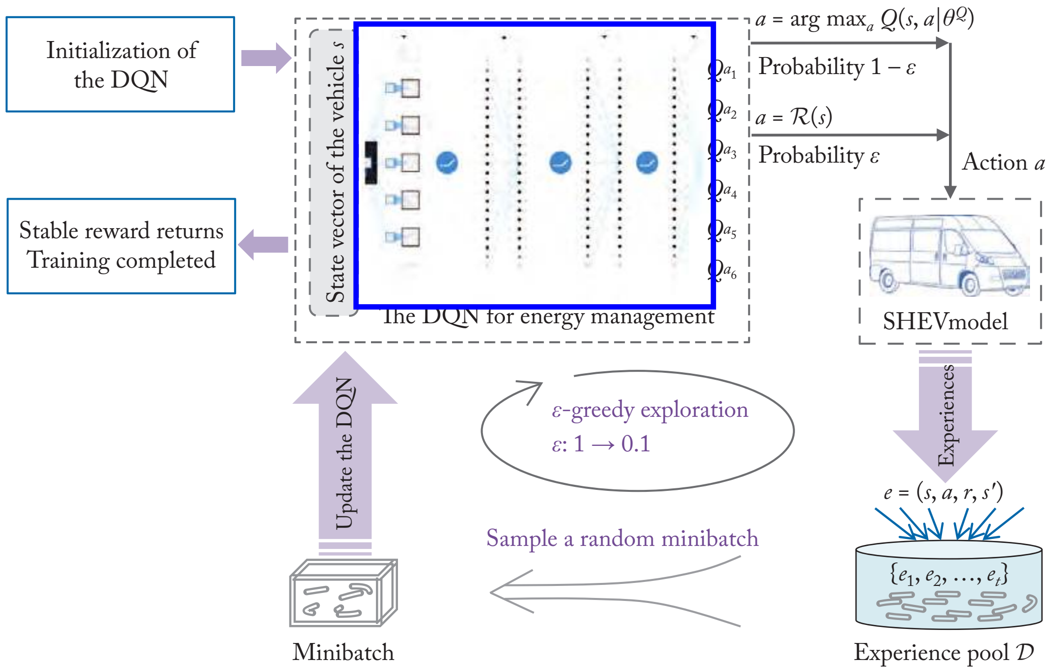

Initialization of the DQN
Stable reward returns Training completed
State vector of the vehicle s
The DQN for energy management
Q_{a_1}
Q_{a_2}
Q_{a_3}
Q_{a_4}
Q_{a_5}
Q_{a_6}
$a = \arg\max_a Q(s, a|\theta^Q)$
Probability $1 - \varepsilon$
$a = \mathcal{R}(s)$
Probability ε
Action a
SHEVmodel
Experiences
$e = (s, a, r, s')$
$\{e_1, e_2, \ldots, e_t\}$
Experience pool $\mathcal{D}$
ε-greedy exploration
ε: $1 \to 0.1$
Sample a random minibatch
Update the DQN
Minibatch

Figure 3.5: The training of DQN-based EMSs.

3.3.1 IMPROVEMENT IN DQN STRUCTURE

The reward function is the key to guiding the learning of action-value function. For energy management, an efficient energy distribution strategy can be simultaneously influenced by both the driving conditions and the energy distribution scheme. Good fuel economy (or higher cumulative rewards) can be obtained either because the vehicle is in good driving conditions with power components working efficiently, or because an efficient energy allocation strategy is implemented. In Section 3.2, only the final action value $Q(s, a)$ is considered, without distinguishing between the state value $V(s)$ of the state the vehicle is in when making a decision, and the additional value obtained by performing the chosen action, i.e., the advantage $A(s, a)$ resulting from performing a certain action.

To further clarify the relationship between $Q(s, a)$, $V(s)$, and $A(s, a)$, their definitions can be summarized as shown in Equation (3.17) [81]:

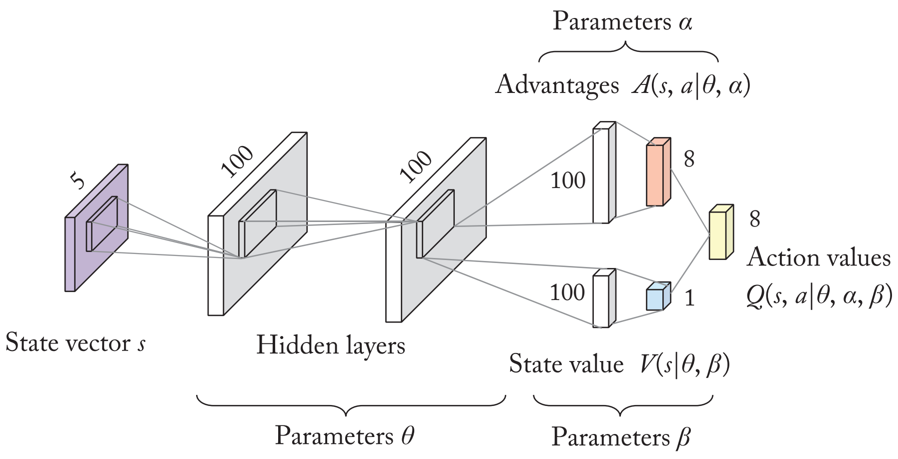

Parameters α
Advantages $A(s, a|\theta, \alpha)$
5
100
100
100
8
8
Action values $Q(s, a|\theta, \alpha, \beta)$
100
1
State vector s
Hidden layers
State value $V(s|\theta, \beta)$
Parameters θ
Parameters β

Figure 3.6: Schematic of the dueling DQN architecture for energy management.

$$\begin{cases} Q(s(t), a(t)) = \mathbb{E}_{s(t+1),a(t+1),\ldots}\left[\sum_{l=0}^{T-t} r(s(t+l), a(t+l))\gamma^l\right] \\ V(s(t)) = \mathbb{E}_{a(t),s(t+1),\ldots}\left[\sum_{l=0}^{T-t} r(s(t+l), a(t+l))\gamma^l\right] \\ A(s(t), a(t)) = Q(s(t), a(t)) - V(s(t)). \end{cases} \tag{3.17}$$

Drawing on the dueling architecture, the DQN described in Section 3.2 can be separated into two streams. As described in Figure 3.6, one stream is the state value function $V(s|\theta, \beta)$ that is independent of the action variables, and the other is the action variables related to the action advantage $A(s, a|\theta, \alpha)$. Then, we can get:

$$Q(s, a|\theta, \alpha, \beta) = V(s|\theta, \beta) + A(s, a|\theta, \alpha), \tag{3.18}$$

where θ denotes all shared parameters, α denotes parameters specific to action advantages, and β denotes the parametersspecific to value function.

By structurally separating the value generated by states from the value generated by actions in the network, it is beneficial to allow the learning system to distinguish whether the high reward (or low fuel consumption) is generated by better driving conditions or benefits from a relatively efficient control strategy. This means that the potential impact of control actions on energy management can be tapped more precisely.

However, given an output $Q(s, a)$ of the new DQN, the corresponding $V(s)$ and $A(s, a)$ cannot be uniquely reduced according to Equation (3.18), that is, the action-value function obtained by this equation is less discriminative. For example, adding and subtracting a constant to $V(s)$ and $A(s, a)$, respectively, yields the same $Q(s, a)$ as the original, but conversely, given

$Q(s,a)$ it is impossible to uniquely determine $V(s)$ and $A(s,a)$. If Equation (3.18) is directly applied to learning of DQN-based energy management, the weak discriminative trait will lead to poor performance in strategy training. To tackle this problem, and to prevent the failure of $V(s)$ during training (e.g., DQN training results in $Q(s,a|\theta,\alpha,\beta) \approx A(s,a|\theta,\alpha)$), the state-dependent action advantage can be set to have zero advantage at the chosen action, as shown in Equation (3.19):

$$Q(s,a|\theta,\alpha,\beta) = V(s|\theta,\beta) + \left[A(s,a|\theta,\alpha) - \max_{a'} A(s,a'|\theta,\alpha) \right], \tag{3.19}$$

where, for greedy strategy $a^* = \arg\max_{a'} Q(s,a'|\theta,\alpha,\beta) = \arg\max_{a'} A(s,a'|\theta,\alpha)$, we can get $Q(s,a^*|\theta,\alpha,\beta) = V(s|\theta,\beta)$.

On the other hand, this zero-setting method will bring additional instability into the network training. With the change of driving conditions, the optimal energy distribution schemes will change continuously, and each branch of the action advantage needs to be updated quickly to adapt to the change of optimal action advantage. For this reason, the advantage function $A(s,a|\theta,\alpha)$ can be set to zero at the average action advantage, instead of the chosen action. Since the action space dimension is 8, the corresponding DQN output will be calculated as follows:

$$Q(s,a|\theta,\alpha,\beta) = V(s|\theta,\beta) + \left[A(s,a|\theta,\alpha) - \frac{1}{8} \sum_{a'} A(s,a'|\theta,\alpha) \right]. \tag{3.20}$$

In this way, the update of the advantage branches only needs to be consistent with the rate at which the mean advantage changes, allowing for improved stability in DQN training. Also, Equation (3.20) is only part of the forward propagation of DQN without additional computation. Because the input and output of the improved network are exactly the same as the original DQN, it does not affect the training process of the aforementioned DQN-based EMS.

3.3.2 IMPROVEMENT IN ESTIMATION OF TARGET Q-VALUE

As described in Sections 2.1 and 3.2, the key to guide the DQN update is to compute the actual action value and thus obtain the loss function for DQN training. In the original DQN, the actual action value *Target* is calculated by both the reward and DQN, and the estimated action value *Estimate* is calculated by the DQN alone. Looking at this calculation process, the current network parameters are still highly relevant for the calculation of the loss function for the next update. In other words, while the DQN weights are continuously updated, the optimization objectives are also changed, and the historical policy acquisition may get quickly overwritten by new updates, leading to deviations in the neural network training. This is very detrimental to the training of DNNs.

Therefore, in order to take into account the strategies learned in historical training phases and to attenuate the instability of strategy learning, a new DNN, called T-DQN (Target DQN), is introduced here in addition to the original DQN. The original DQN network will be referred

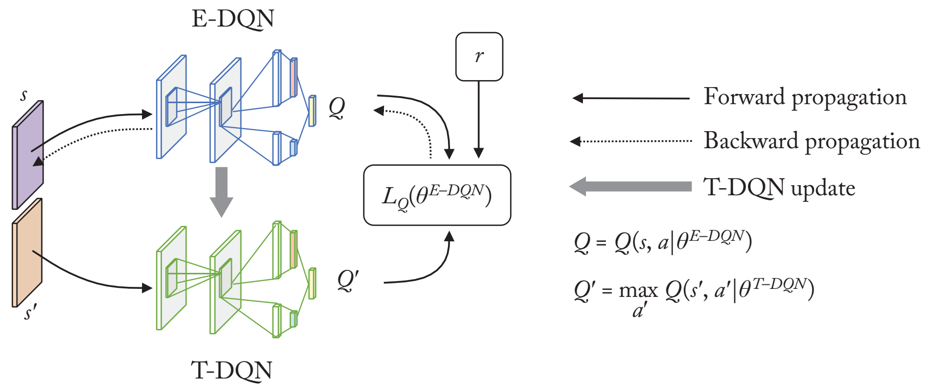


Figure 3.7: The dataflow graph of DQNs with T-DQN.

to as E-DQN (Evaluation DQN), and the T-DQN and E-DQN share the same network structure and initial parameters.

The loss function is redefined as follows with a more stable *Target*. Different from the old loss function (Equation (3.13)), the actual action value is calculated by reward and the T-DQN, instead of the E-DQN:

$$\begin{cases} Target = r + \gamma \max_{a'} Q(s', a'|\theta^{T-DQN}) \\ L_Q(\theta^{E-DQN}) = E_{(s,a,s',r)\sim U(\mathcal{D})}\left[Target - Q(s,a|\theta^{E-DQN})\right]^2, \end{cases} \tag{3.21}$$

where θ^{E-DQN} and θ^{T-DQN} denote the parameters of E-DQN and T-DQN, respectively.

Following the computation of loss function, the gradient backpropagation is still performed through the E-DQN, while the T-DQN parameters slowly follow the continuously updated E-DQN at a slower rate τ ($\tau \ll 1$), as in Equation (3.22), or replicate the network parameters of E-DQN at a lower frequency, $\theta^{T-DQN} = \theta^{E-DQN}$:

$$\theta^{T-DQN} = \tau\theta^{E-DQN} + (1-\tau)\theta^{T-DQN}. \tag{3.22}$$

The dataflow graph of DQNs with T-DQN is shown in Figure 3.7.

On the other hand, because of the greedy strategy in action selection with T-DQN, the update of E-DQN will follow the *Target* with maximum TD error, which is likely to lead to an overestimation of action value and thus affects the learning effect. Therefore, with two sets of DQNs, the action selection and action value estimation parts of the *Target* calculation are further decoupled here: the action selection is implemented based on E-DQN, while the current greedy strategy is evaluated based on T-DQN, which is known as the double DQN method [73]. Therefore, the *Target* in Equation (3.21) can be further formulated into Equation (3.23), and calculating loss function with this *Target* will improve the issue of Q-value overestimation:

$$Target = r + \gamma Q(s', \max_{a'}(s', a'|\theta^{E-DQN})|\theta^{T-DQN}). \tag{3.23}$$

3.3.3 IMPROVEMENT IN EXPERIENCE REPLAY

In Section 3.2.3, the experience replay method samples past experiences by uniform random sampling, and does not take into account the differences of these experiences. In practice, however, there is a portion of sample data in the experience pool whose state transition and reward feedback would be more valuable and meaningful for strategy learning. For example, with the same driving cycles, the state transition data $e = (s, a, s', r)$ obtained after performing a more efficient energy allocation scheme will be more beneficial to the stable and efficient EMS learning compared to data obtained from an inefficient scheme, i.e., it would be more valuable to guide the learning of DQN-based EMS toward direction with higher returns.

According to prioritized experience replay presented by DeepMind [70], this section will introduce how to implement this method in DQN-based EMSs. Prioritized experience replay focuses on how to define the criteria for evaluating the importance of sample data. The extent to which it can improve on strategy learning is the most desirable metric for a sample, but such a metric cannot be known precisely in advance. Another reasonable surrogate is the TD error of experiences, which reflects how surprising the experience is and the extent to which it can influence the magnitude of the strategy update. Meanwhile, we could easily obtain the TD error from the loss function, so it is quite appropriate to use this metric as a measure of sample importance and sampling priority. The TD error δ of $e = (s, a, s', r)$ in the improved DQN is as follows:

$$\delta = \left[r + \gamma Q(s', \max_{a'}(s', a' | \theta^{E-DQN}) | \theta^{T-DQN}) \right] - Q(s, a | \theta^{E-DQN}). \tag{3.24}$$

Contrary to uniform random sampling, greedy sampling replays experiences with larger absolute value of TD error from the experience pool for training. However, reward spikes in energy management can lead to increased TD error noise, and greedy sampling can easily introduce more errors in the training. In addition, greedy sampling can lead to experiences with small TD errors not being replayed long after they are deposited into the experience pool, but only a small portion of the experiences with high TD errors will be preferred for training. This is likely to cause a slow decline in training error, while the reduced sample diversity may lead to overfitting of the strategy training as well.

Stochastic prioritized sampling is a replay method interpolating between greedy sampling and random uniform sampling, which can effectively alleviate the issues associated with greedy sampling. This sampling method ensures that the probability of an experience e being sampled is monotonic with respect to its priority of e, while ensuring that even experiences with the lowest priority have a chance to be replayed [71]. Specifically, the sampling probability P_i of the state transition experience e_i is defined as follows:

$$P_i = \frac{p_i^{\sigma}}{\sum_k p_k^{\sigma}}, \tag{3.25}$$

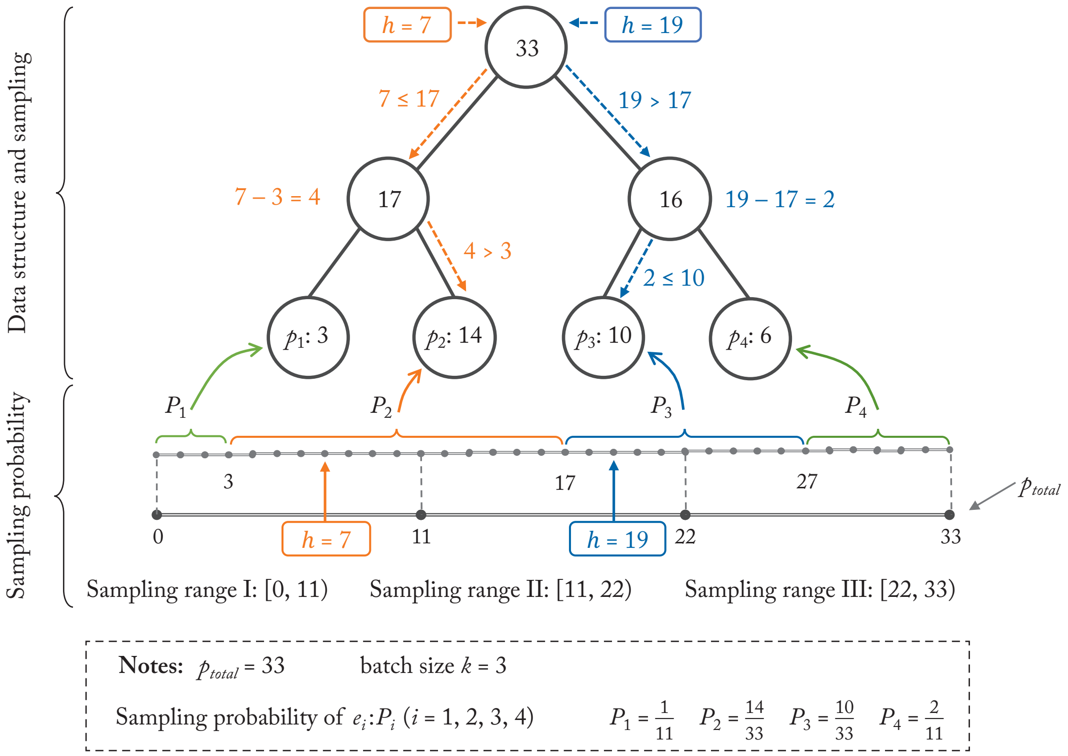

Data structure and sampling
Sampling probability
$h = 7$
$h = 19$
33
$7 \le 17$
$19 > 17$
17
16
$7 - 3 = 4$
$19 - 17 = 2$
$4 > 3$
$2 \le 10$
p_1: 3
p_2: 14
p_3: 10
p_4: 6
P_1
P_2
P_3
P_4
3
17
27
p_{total}
0
11
22
33
$h = 7$
$h = 19$
Sampling range I: [0, 11)
Sampling range II: [11, 22)
Sampling range III: [22, 33)
Notes: $p_{total} = 33$
batch size $k = 3$
Sampling probability of e_i:P_i (i = 1, 2, 3, 4)
$P_1 = \frac{1}{11}$ $P_2 = \frac{14}{33}$ $P_3 = \frac{10}{33}$ $P_4 = \frac{2}{11}$

Figure 3.8: Illustration of sampling from the sum-tree.

where the priority of experience e_i is $p_i = |\delta_i| + \varepsilon$ (ε is a small positive constant to avoid zero sampling probability), the exponent σ determines how much prioritization is adopted, with $\sigma = 0$ corresponding to the uniform case. In practice, the implementation requires TD error clipping to reduce the impact of extreme values on the robustness of the algorithm.

Considering the large capacity of the experience pool, a "sum-tree" data structure is adopted by DeepMind to store the priority sequence with the corresponding experiences, which ensures that the experience pool can be efficiently updated and sampled from. Figure 3.8 illustrates the prioritized sampling process with the "sum-tree." The computational time complexity of finding the highest priority experience sample is $O(1)$, and allowing $O(\log N)$ updates and sampling.

The priority p_i of the experience is stored in the leaf node in the sum-tree. The value of each internal node is the sum of its child nodes, and the parent node contains the sum over all priorities, p_{total}. To sample a minibatch of size k, the range $[0, p_{total}]$ is divided into k ranges equally. Next, a value h_i is uniformly sampled from each range. Finally, experiences e_i corre-

Algorithm 3.1 Retrieve experience samples from the sum-tree

The initial value of N is the order number of the parent node in the sum-tree;
h is the random number sampled from each range;
$leaf\_node$ is the leaf node;
$N.left\_child$ and $N.right\_child$ represent the left child node and right child node, respectively;
$N.left\_child.value$ denotes the value of node N.

```
1: def retrieve($N, h$):
2: if $N$ is a leaf node $leaf\_node$ then
3:     return $N$
4: end if
5: if $h \le N.left\_child.value$ then
6:     return retrieve($N.left\_child, h$)
7: else
8:     return retrieve($N.right\_child, h - N.left\_child.value$)
9: end if
```

sponding to h_i $(i = 1, 2, ..., k)$ are retrieved from the tree, realizing prioritized experience replay: $e_i \sim P_i$. The pseudo code of retrieving process is described in Algorithm 3.1.

Take the sum-tree structure shown in Figure 3.8 as an example. It stores four experiences and their priorities. As the number of experiences to be sampled is $k = 3$, the range $[0, 33]$ is divided into three ranges: $[0, 11)$, $[11, 22)$, and $[22, 33]$. Assume that the random number sampled from [0, 11) is $h = 7$, the corresponding retrieving process is as follows. According to Algorithm 3.1, starting from the parent node, compare h with its left child node value, if h is not bigger than its left child node value, then select the left child node, otherwise select the right child node. Repeat the same procedure iteratively to complete the retrieval and obtain the sample e_2 with priority p_2. Figure 3.8 also depicts the correspondence between the sampling range and the final retrieved samples, which intuitively shows that the probability of obtaining the experience sample e_2 is consistent with Equation (3.25).

But on the other hand, prioritized experience replay changes the distribution consistency of the replayed minibatch samples with the expected sample distribution, which may introduce biases in the sampling process and thus affect the direction of strategy convergence. For this reason, importance-sampling weights w_i (IS weights) need to be introduced to correct this bias, as shown in Equation (3.26), and replace δ_i with $w_i \delta_i$ in the DQN loss function [70]. For the

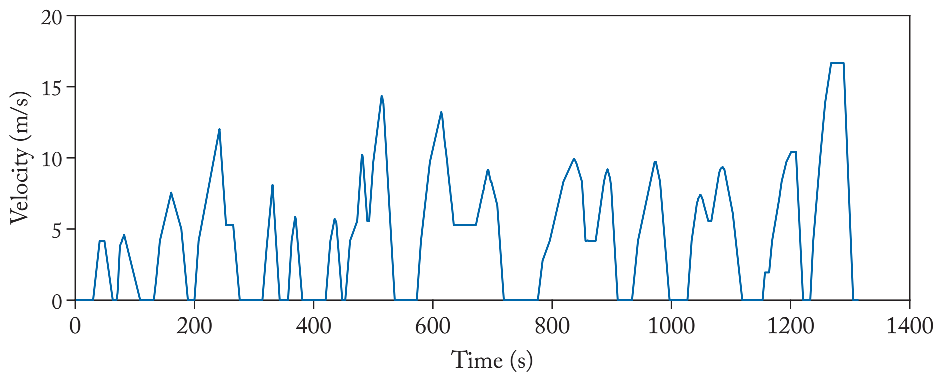


Figure 3.9: The China typical urban driving cycle.

sake of stability, the IS weights are usually normalized by dividing them by $\max_i w_i$:

$$w_i = \left(\frac{1}{size}\frac{1}{P(i)}\right)^{\beta}, \tag{3.26}$$

where *size* denotes the size of experience pool and β is the exponent factor for IS weights. When $\beta = 1$, the non-uniform probabilities will get fully compensated. In practice, it is recommended to linearly anneal β from its initial value of β_0 to 1, and β approaches 1 only at the end of strategy learning.

3.3.4 STABLE TRAINING METHOD FOR DQN-BASED EMSs

In this chapter, the China typical urban driving cycle (CTUDC) is chosen as the velocity profile for EMS training, as shown in Figure 3.9. The time length of one CTUDC is 1314 s, with a full trip distance of 5.898 km, average velocity of 4.5 m/s, and standard deviation of speed 4.2 m/s.

During strategy training, the input data of DQN needs to be normalized in advance. Data of state variables relevant to driving cycles, including v and acc, are preprocessed by the Z-score normalization method; all parameters for normalization (mean value and standard deviation of each state variable) are calculated based on training driving cycles, and they will be recorded and fixed for normalization of corresponding state variables during an EMS test on testing driving cycles using the same method. According to predefined engine operation trajectory, the maximum engine output power is about 55 kW, thus, by dividing the maximum power, the engine power P_{eng} is normalized linearly into $[0, 1]$. Originally, because $SoC \in [0, 1]$ and $\Delta SoC \in [-1, 1]$, they can be fed into the networks directly. In this way, all input state data can be approximately of similar range regardless of extremums.

With all improvements described in Section 3.3, a stable training of DQN-based EMS can be mostly ensured. Algorithm 3.2 shows the pseudocode for the training of DQN-based EMSs.

Algorithm 3.2 The training of DQN-based EMSs

Initialize dueling E-DQN by Xavier uniform initializer; initialize T-DQN; initialize experience pool $\mathcal{D}$; initialize exploration rate $\varepsilon = 1$; maximum iteration number $N_{iteration}$; time length of the training driving cycle T_{CTUDC}; size of minibatch $Minibatch = 32$; $\Delta = 0$; $p_1 = 1$.

```
1:  for $IterationRound = 1 : N_{iteration}$ do
2:    Observe initial state $s(0)$
3:    for $t = 0 : (T_{CTUDC} - 1)$ do
4:      Select action by current policy $a(t) = \arg\max_a Q(s(t), a|\theta^{E-DQN})$ (with probability
        $1 - \varepsilon$) or random generation $a(t) = \mathcal{R}(s(t))$ (with probability $\varepsilon$)
5:      Execute action $a(t)$ by calling the SHEV model
6:      Observe reward $r(s(t), a(t))$ and the next state $s(t+1)$
7:      Store $e(t) = (s(t), a(t), r(s(t), a(t)), s(t+1))$ into $\mathcal{D}$, and set its priority as $p_t =$
        $max_i\, p_i$
8:      if $\mathcal{D}$ is full then
9:        for $j = 1 : Minibatch$ do
10:         Sample a minibatch experiences from $\mathcal{D}$:
            $e_j = (s, a, s', r)_j \sim P_j = p_j^\sigma \Big/ \sum_k p_k^\sigma$
11:         Compute the IS weights of $e_j$: $w_j = \left(size \cdot P_j\right)^{-\beta} / \max_i w_i$
12:         Compute the TD-error $\delta_j$ and loss function $L_Q^j(\theta^{E-DQN})$
13:         Update the priority of $e_j$: $p_j \leftarrow \left|\delta_j\right|$
14:         Compute cumulative gradient: $\Delta \leftarrow \Delta + w_j \delta_j \nabla_{\theta^{E-DQN}} Q(s, a|\theta^{E-DQN})$
15:       end for
16:       Update E-DQN: $\theta^{E-DQN} = \theta^{E-DQN} + \alpha_l \Delta$
17:       Reset cumulative gradient $\Delta = 0$
18:       Update T-DQN: $\theta^{T-DQN} = \tau\theta^{E-DQN} + (1 - \tau)\theta^{T-DQN}$
19:       Anneal exploration possibility: $\varepsilon = \max(0.1, \varepsilon - (N_{iter} T_{CTUDC})^{-1})$
20:     end if
21:   end for
22: end for
23: Output the final E-DQN as the trained DQN-based EMS:
    $\pi = \arg\max_a Q(s, a|\theta^{E-DQN})$
```

3.4 THE LEARNING AND EVALUATION OF EMSs IN CONTINUOUS STATE SPACE

3.4.1 THE BASELINE STRATEGY: DP-BASED EMSs

For MDP problems with known global environment information, DP is an effective offline solution to obtain the global optimal control strategy and provides a baseline for fuel economy evaluation of other developed EMSs. DP also transforms the Bellman equation into multiple single-step decisions to iteratively solve the global optimal strategy. First, the state and action space of the energy management problem needs to be discretized before applying DP. Then, the DP algorithm will traverse the discrete state-action space to obtain the tabular cost function representation by inverse calculation, and then find the global optimal decision sequence by forward search [103].

In this book, a generic DP Matlab toolbox (available from [104, 105]) is utilized to derive the baseline, and the final SoC is constrained to its sustaining level of 60% by using the boundary line method of this toolbox. As both DP and DRL are developed for MDP problems, the established SHEV model and formulated energy management problem in aforementioned sections can be mostly applied for derivation of the DP-based EMS directly. For this baseline strategy, its application differences with the DQN-based EMSs exists in four aspects: (1) the reward function (namely the cost function in DP) is replaced solely by the engine fuel consumption rate $\dot{m}_{fuel}$; (2) accurate velocity and road profiles are assumed to be a prior; (3) only SoC is selected as the state variable in DP and it is discretized into 50 grids between 40% and 80%; and (4) engine control signals (the speed and torque of the engine) are chosen as control actions and both of them are discretized into 100 grids within their feasible domains.

It should be noted that the DP-based EMS in this book does not take into account realistic constraints such as frequent start/stop of the engine and control signal continuity, but only solves the numerical optimal solution in the feasible domain of energy management problems to provide a theoretically optimal fuel economy control benchmark, thus facilitating the evaluation of the DRL-based EMSs. In addition, it is difficult to reach or exceed the ideal fuel economy benchmark when the general optimization-based EMSs consider realistic constraints of the powertrain and lacks sufficient future driving cycle information, which is one of the challenges in current energy management research.

3.4.2 LEARNING EVALUATION

In this section, the learning process of the improved DQN-based EMSs is discussed. Generally, for DRL-based EMSs, we could evaluate their learning ability by the SoC trajectories during training, the average single-step reward, the convergence of action value, and the loss function. Among them, the key indicator of strategy learning is to observe whether the rewards gained during training are increasing, which is consistent with the basic idea of reinforcement learning.

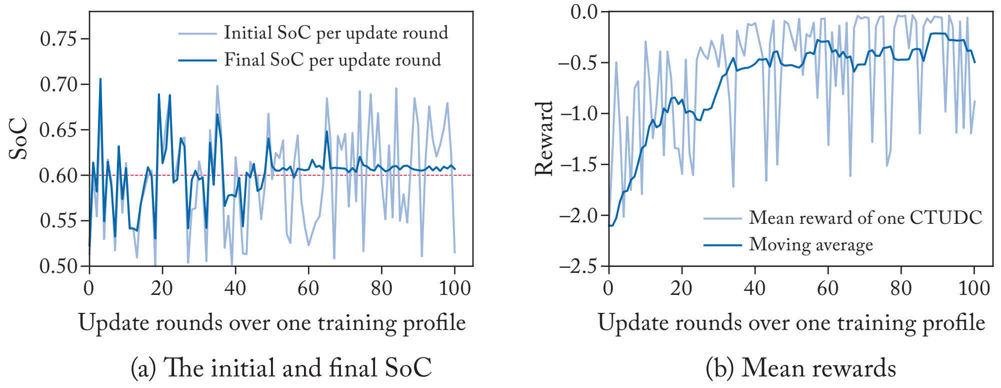


Figure 3.10: Visualization of strategy learning process.

To enhance generalization, the initial SoC value at the beginning of each update round is randomly selected from $[SoC_{sust} - 0.1, SoC_{sust} + 0.1]$. The training of DQN-based EMS on one CTUDC takes approximately 12 min on a computer with a processor of i7-2600CPU @ 3.4 GHz. As the strategy learning is realized by iterative interactions of the real-time strategy over the training profile, the learning process can be described by the number of times the training profile is traversed, i.e., the number of update rounds over the training profile.

Figure 3.10 depicts the performance of the EMS during learning. Figure 3.10a shows that with strategy updates, the final SoC after each update round gradually approaches the sustaining level even with different initial SoC values, which is consistent with the goal of charge sustaining. Figure 3.10b shows that the moving average of the average single-step reward obtained by the strategy on the training profile also rises with the learning process. To visualize the strategy learning effect more intuitively, Figure 3.11 shows the 100 km equivalent fuel consumption per 100 km on each update round, whose moving average decreases gradually as well. On the other hand, the fluctuations in the reward and equivalent fuel consumption recorded in real time are mainly caused by the initial state variability and the exploration strategy, while their overall trend is consistent with the strategy learning expectation. This also indicates, to a certain extent, that DRL can still derive policy improvement directions from the data in the presence of changes and perturbations.

As the core of the DQN-based EMS, the training of DQN is presented in Figure 3.12. In Figure 3.12a, the DQN training error (loss function) gradually decreases as the training goes on, indicating that the DQN parameters gradually converge. Meanwhile, the average action value estimation of DQN also gradually improves and stabilizes, as shown in Figure 3.12b. The training of DQN further suggests that the strategy gradually learns the requirements for charge sustaining and achieves higher action value estimates.

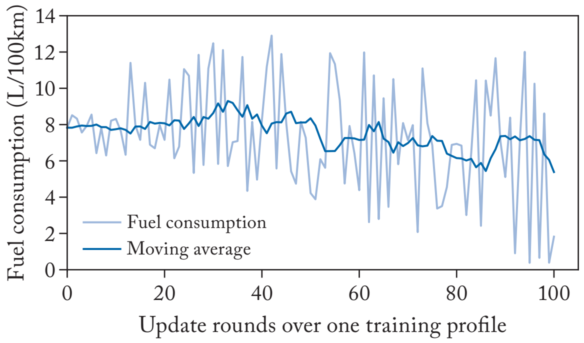


Figure 3.11: Fuel economy performance during strategy training.

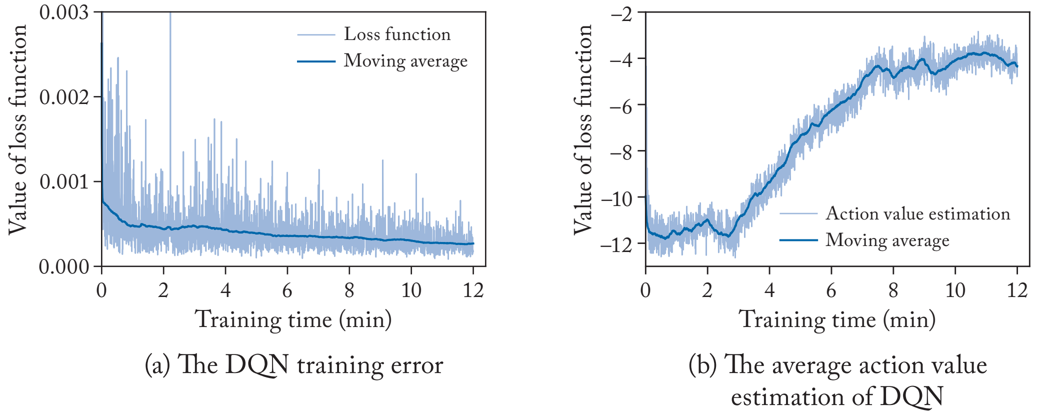


Figure 3.12: Visualization of DQN training process.

3.4.3 OPTIMIZATION EVALUATION

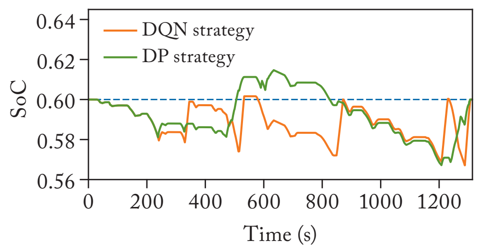
In this section, the optimization performance of the improved DQN-based EMS will be examined by comparing with the fuel economy baseline, DP-based EMSs. Fuel consumption, as a direct evaluation metric of the HEV energy management reflects the overall energy-saving performance of the strategy, while its comparison with the fuel economy benchmark (fuel consumption of the DP-based EMS) reveals the extent to which it can tap the vehicle's energy saving potential, i.e., the optimization capability.

Table 3.2 compares the performance of the two kinds of EMSs, whose initial SoC (SoC_{init}) are both set as 0.6. The differences in final SoC values (SoC_{final}) from sustaining level are compensated into final fuel consumption, denoted by $Fuel_c$, for a fair comparison. The fuel consumption of the DQN-based EMS is about 9.8% higher than the baseline, but relatively less in engine starts, which is more practical in the actual application considering the higher fuel

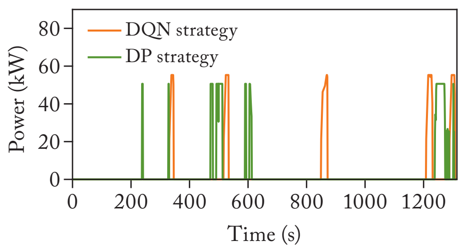
Table 3.2: Results comparison for optimization validation

Method	SoC_{init}	SoC_{final}	N_{eng}	T_{cal} (s)	$Fuel_c$ (g)	Gap_{fuel_c}
DP	0.6000	0.6000	12	24.5507	226.5	—
DQN	0.6000	0.6000	5	1.2640	248.7	9.80%

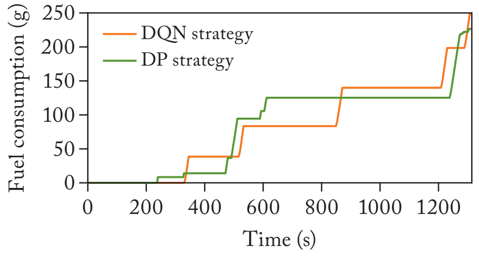
Notes: N_{eng}: engine start times, T_{cal}: calculation time, Gap_{fuel_c}: the gap of fuel economy with the DP-based EMS.



(a) SoC trajectories



(b) Engine output power



(c) Cumulative fuel consumption

Figure 3.13: The performance of two strategies on the training profile.

consumption rate during idle and starting [106]. The DP strategy, as an offline optimization method, takes about 24.6 s to compute in the simulation environment, while the trained DQN strategy is more advantageous in terms of computational speed (about 1.3 s and the single-step operation takes about 1 ms) owing to the fact that only the forward computation of DNNs is involved.

Figure 3.13 shows the SoC trajectory, engine power, and cumulative fuel consumption of the DQN strategy compared with the DP strategy on a training driving cycle. Figure 3.13a shows that the DQN strategy is able to keep the SoC at the desired charge sustaining level at the end of the trip with no battery overcharge or overdischarge, performing as expected. From Figure 3.13b, it can be seen that the DQN strategy has fewer engine starts on CTUDC, and accordingly its

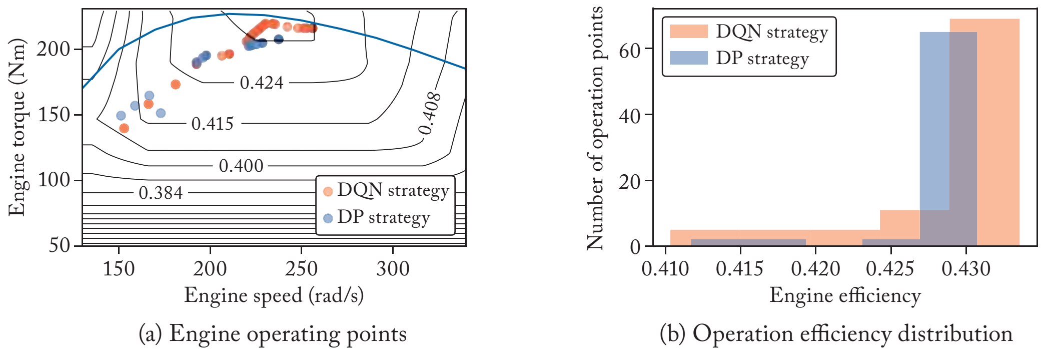
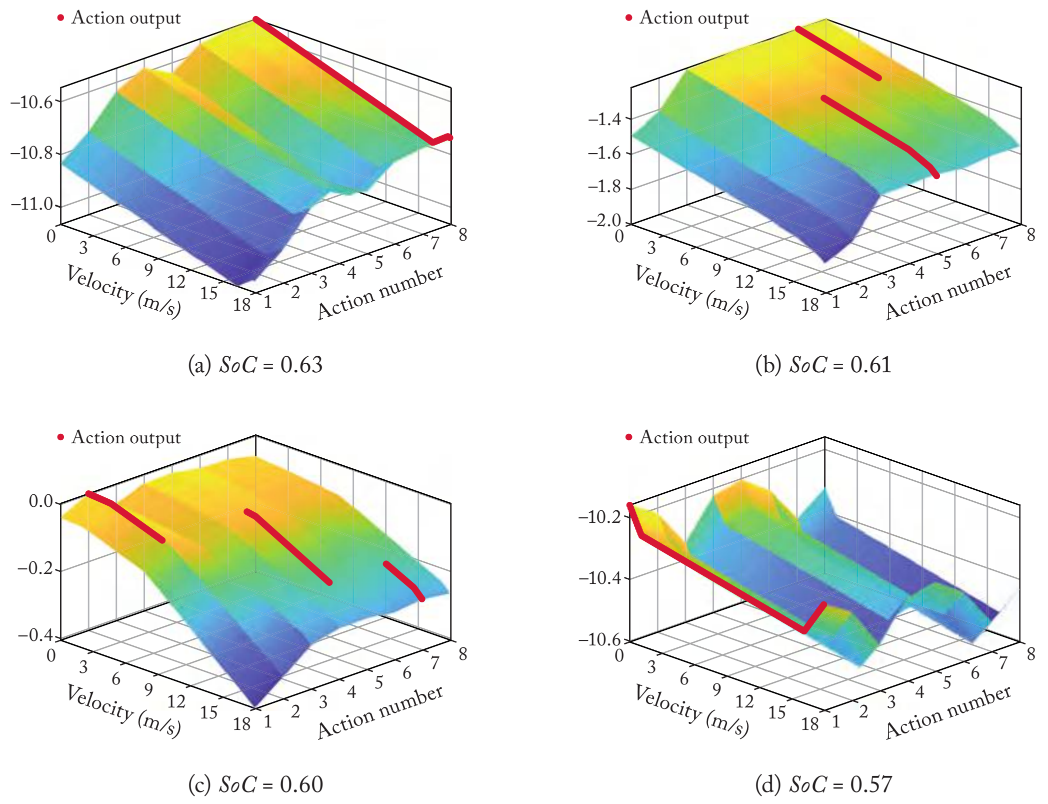


Figure 3.14: The engine performance of two strategies on the training profile.

average output power is slightly higher than that of the DP strategy. This result indicates that the engine start penalty term in the reward function can effectively guide the strategy to reduce the engine start-stop frequency. Figure 3.13c indicates that the DP strategy chooses to sacrifice certain fuel economy in the mid-trip to obtain overall low fuel consumption. This superiority of the DP strategy mainly owes to the termination state constraints and the boundary line method adopted by the DP solver [105]. In contrast, strategy learning in the DQN strategy is guided exclusively by immediate rewards and it is not advisable to set a higher discount rate considering the training stability, which may be the main reason for the gap between DQN strategies and the baseline. But overall, the DQN strategy performs well on the training profile.

Figure 3.14 compares the engine operation of the two strategies: the DQN strategy ensures that the engine works often in the efficient area, and the engine efficiency distribution and frequency are similar to those of the DP strategy.

Since the core of DQN strategies lies in the action-value mapping, but limited by the state diversity and the difficulty of high-dimensional spatial visualization, only the visual analysis of action value mapping in individual scenarios is selected here to understand the learning and optimization of the DQN strategy. Figure 3.15 shows the variation of the action-value function output with SoC and vehicle velocity for a certain driving state ($acc(t) = 0.5$ m/s^2), where the action number corresponds to each of the eight optional actions in the action space A, and the action output represents the action value of the selected action in corresponding states. Overall, the action value of energy management is higher when the SoC is close to the CS level, and decreases when the power is above or below the CS level. When SoC is above the CS level ($SoC = 0.63$), the Q-value of actions related to reducing or turning off the engine power is higher, so the strategy chooses to turn off the engine regardless of the velocity (action a_8), and the driving power is entirely provided by the battery. When the SoC is below the steady holding level ($SoC = 0.57$), the value of the action related to increasing the engine power is higher, so the strategy chooses to increase the engine power quickly (action a_1) to charge the power cell. With



(a) $SoC = 0.63$ (b) $SoC = 0.61$ (c) $SoC = 0.60$ (d) $SoC = 0.57$

Figure 3.15: The performance of two strategies on the training profile.

lower SoC ($SoC = 0.57$), the value of the action related to increasing the engine power is higher, so the strategy chooses to increase the engine power quickly (action a_1) to charge the battery. Around the CS level ($SoC = 0.61$), the strategy tends to consume some electricity: with slow velocity, the strategy reduces the engine power (action a_7), while as the velocity rises, the strategy will choose to maintain the current engine power (action a_5). When $SoC = 0.60$, the strategy tends to increase the engine power appropriately (action a_2) at lower velocity to maintain the SoC stable and obtain a higher reward, while at medium and high velocity, it tends to consume electricity to reduce fuel consumption: maintain (action a_5) or reduce (action a_7) the engine power.

In summary, the optimization capability of the DQN-based EMS can be effectively verified by analyzing the fuel economy, engine start-stop frequency, and charge sustaining performance.

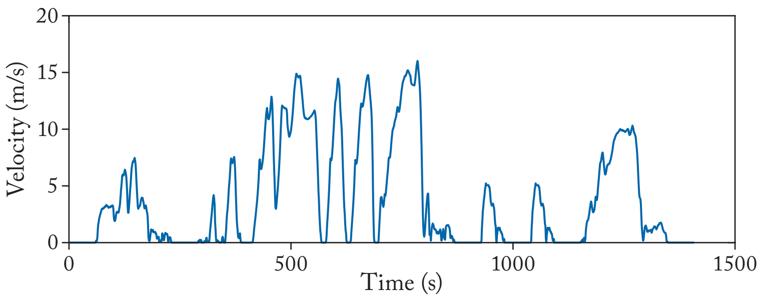


Figure 3.16: The driving cycle for tests: WVUCITY.

Table 3.3: Test results of DQN-based and DP-based strategies on WVUCITY

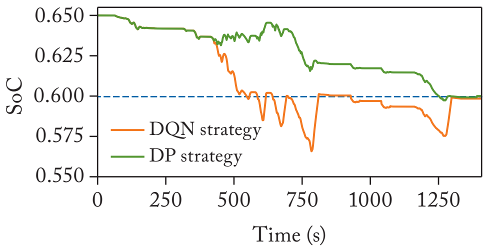
Group	Method	SoC_{init}	SoC_{final}	N_{eng}	T_{cal} (s)	$Fuel_c$ (g)	Gap_{fuel_c}
1	DP	0.6500	0.6000	16	25.4628	147.5	—
	DQN	0.6500	0.5984	6	1.4537	163.2	10.64%
2	DP	0.6000	0.6000	16	25.2785	245.0	—
	DQN	0.6000	0.5988	12	1.3740	269.8	10.12%
3	DP	0.5500	0.6000	22	25.8424	343.3	—
	DQN	0.5500	0.5985	11	1.4342	371.3	8.16%

Notes: N_{eng}: engine start times, T_{cal}: calculation time, Gap_{fuel_c}: the gap of fuel economy with the DP-based EMS.

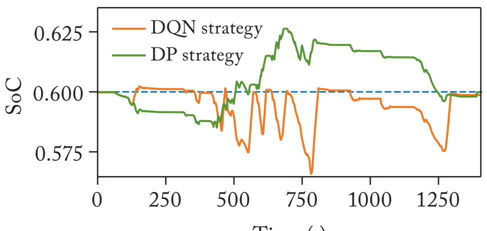
3.4.4 GENERALIZATION EVALUATION

Since the training of DQN-based strategies is based on CTUDC, we will examine the performance of the strategy under different driving cycles and its generalization ability in this section. Here, the WVUCITY (duration 1408 s, total mileage 5.319 km, average velocity 3.8 m/s, standard deviation of velocity 4.6 m/s) is chosen as the test velocity profile for generalization verification, as shown in Figure 3.16.

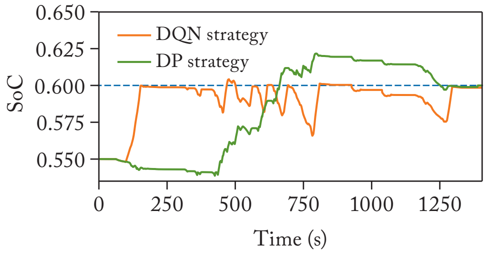
We carry out strategy tests with three different initial SoC values, and Table 3.3 summarizes the test results of the DQN strategy and the baseline. In the three tests, the DQN strategy manages to regulate the SoC to near the CS level at all trip terminations (Figure 3.17). Even without any priori knowledge of the new driving cycle, the DQN strategy exhibits a consistent energy-saving performance, with the average fuel consumption approximately 9.6% higher than the benchmark. Meanwhile, the DQN strategy is able to reduce the engine start frequency to a lower level despite more dramatic velocity changes; the number of engine starts is reduced



(a) Initial SoC = 0.65

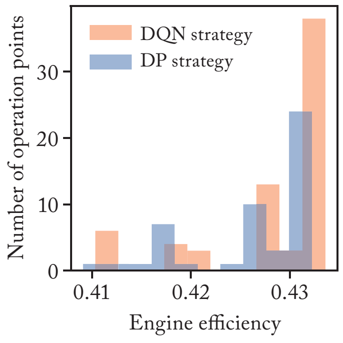


(b) Initial SoC = 0.60

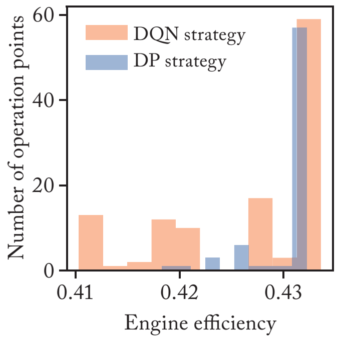


(c) Initial SoC = 0.55

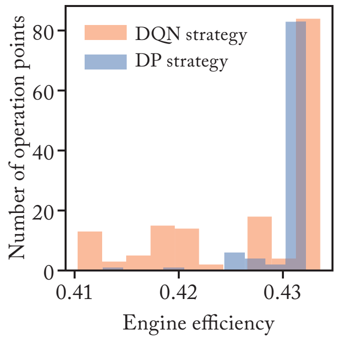
Figure 3.17: SoC trajectories of the DQN-based EMS on the test profile.



(a) Initial SoC = 0.65



(b) Initial SoC = 0.60



(c) Initial SoC = 0.55

Figure 3.18: SoC trajectories of the DQN-based EMS on the test profile.

by about 46% on average compared to the baseline. The DQN strategy still shows a significant advantage in computational speed (about 1 ms per simulation step).

Figure 3.18 shows the engine efficiency statistics for the three sets of tests, indicating that the DQN strategy also ensures that the engine works in the efficient area as much as possible. Besides, as the initial SoC level increases, engine operation time also decreased. Based on the above, we can conclude that the DQN strategy shows good generalization ability on the test profile.

SUMMARY

(1) Taking a SHEV as an example, this chapter establishes its energy consumption model as a policy interaction environment for DRL, and introduces continuous state space—discrete action space in energy management based on this vehicle model. Based on the discrete action space, a mapping method for action-value function of energy management is introduced, and the discrete kind of EMS learning theory based on deep Q-learning and its training method are stated.

(2) Based on the original DQN-based EMS, three aspects of improvements are introduced to enhance its training stability, including optimization of the DQN structure, improvement in estimation of target Q-value, and prioritized experience replay.

(3) The learning of the improved DQN-based EMS is evaluated in terms of the training error, convergence of the action-value function, SoC trajectory, reward returns, fuel consumption, etc. A global optimal EMS based on DP is constructed as the benchmark and the fuel economy baseline, which provides assessment of the optimization and generalization capability of DQN-based EMSs. The results indicate that the improved DQN-based EMS designed for continuous state space—discrete action space has an average gap of about 9.7% in energy savings from the baseline and excels in reducing the engine start-stop frequency and calculation speed.

CHAPTER 4

Learning of EMSs in Continuous State–Continuous Action Space

Depending on the modeling approach or configuration types, some energy management problems modeled as the MDP have single or multiple continuous control actions. For such problems, traditional optimization methods usually adopt discretization solutions, but their application scenarios and computational amount are vulnerable to dimensional issues. The study of continuous energy management methods that can directly search for the optimal policy in the continuous state—continuous action space will be of great benefit to the expansion of application range to a certain extent. In addition, based on the continuous energy management method, this chapter also introduces a PHEV energy management solution integrating history cumulative trip information (HCTI) to improve the EMS learning effect across a wider feasible domain of SoC.

4.1 ENERGY MANAGEMENT BASED ON DDPG METHOD

The method presented in this chapter is independent of powertrain topology such that it is applicable to any type of HEV with continuous state and continuous action space. Therefore, we simply adopt the SHEV in Chapter 3 as the study case for continuous DRL-based EMSs. The energy consumption model is described in Section 3.1. The formulation of energy management, the agent-environment interaction, and the definition of continuous state space in Section 3.2 still apply here. The difference lies in the following two aspects.

(1) The goal of PHEV energy management. Since the study case here is a PHEV, the pattern of electricity consumption is no longer limited by charge sustaining. Meanwhile, it has been shown that for PHEV energy management, reasonable scaling of battery capacity in simulation can effectively reduce the length of test profiles and accelerate the simulation speed, and the accelerated simulation results can still effectively evaluate the strategy [8]. Therefore, the SHEV in the last chapter will be treated as a PHEV (expected driving range of 100 km and feasible SoC domain of [0.2, 1.0]) for explanation of continuous EMSs.

Typically, when PHEVs operate in electric mode, the powertrain energy flow is unique and deterministic according to the even torque distribution assumption between the front and rear axle, and no energy management is required in this circumstance. Thus, this chapter mainly deals with the case where there is no limitation in energy consumption pattern, i.e., the energy flow is totally determined by EMSs to balance fuel and electricity consumption for better fuel economy within the expected driving range. This mode of energy management is referred to as Blended Mode (BM).

(2) The definition of continuous action space. We still control the engine following the pre-defined optimal working trajectory, and take the variation of engine power ΔP_{eng} as the control action. However, ΔP_{eng} will not be discretized, but treated as a continuous action. Then, the one-dimension action space can be expressed as $A = \{\Delta P_{eng} | \Delta P_{eng} \in (-10 \text{ kW}, 10 \text{ kW})\}$. Specially, when $\Delta P_{eng} < -4$ kW, the engine will be shut off; the engine will only start when constraints in Equation (3.5) are satisfied.

4.1.1 LEARNING THEORY OF DDPG-BASED EMSs

To deal with continuous action outputs, we will adopt a parameterized approximation method, the DNN, to express the EMS, namely the strategy network, or action network (Actor). The actor network μ, parameterized by θ^{μ}, takes state s as input and outputs action a. The corresponding EMS π is as shown as follows:

$$\pi = \mu(s|\theta^{\mu}). \tag{4.1}$$

Figure 4.1 shows the structure of the strategy network (Actor). The neurons in the input layer coincide in number with the dimension of the state space S. There are three fully connected hidden layers consisting of 100-100-100 units and each layer is followed by ReLU activation functions. An output layer processed by a sigmoid function is connected to the inner product layer to output the bounded action $\Delta P_{eng} = 10\mu(s|\theta^{\mu})$ kW, $\Delta P_{eng} \in (-10 \text{ kW}, 10 \text{ kW})$.

Different with the Actor network, the Critic network (Q, as shown in Equation (4.2) and Figure 4.1) takes state s and action a as input and outputs estimated action-value $Q(s,a)$ which evaluates the current action. Considering the dueling architecture in DQN, the Critic is designed to have two streams, one for the action and another one for the state, and the neuron number of the input layer is the sum of state space dimension and action space dimension. Both streams are processed by three fully connected layers consisting of 100-100-100 units, respectively. Each fully connected layer is followed by ReLU activation functions. A linear output layer is connected to the final inner product layer to output of single scalar $Q(s,a)$:

$$Q(s,a) = Q(s,a|\theta^{Q}), \tag{4.2}$$

where θ^{Q} denotes parameters of the Critic.

As a stochastic policy method, in the original Actor-Critic algorithm, the Actor outputs the probability distribution of each action, and the control action is randomly generated based on

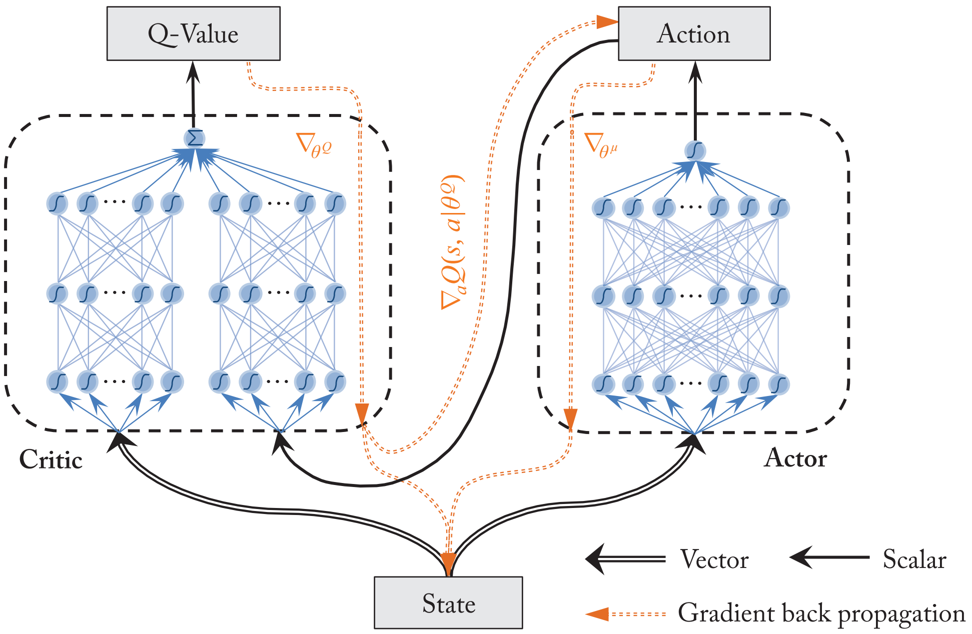


Figure 4.1: The architecture of Actor-Critic networks and gradients back propagation.

their corresponding probability [56]. However, the algorithm itself is not stable enough because it introduces multiple nonlinear approximation functions, and the samples are not efficiently utilized as a stochastic policy method. Therefore, DeepMind proposed a novel deterministic policy gradient method based on the Actor-Critic structure, where the Actor outputs deterministic actions and the strategy is improved directly based on the policy gradient [80]. Then, the deterministic Policy Gradient (DDPG) algorithm was proposed by further combining deterministic policy gradient with DQN, which excelled in Atari games [62]. In this chapter, we choose to combine the DDPG to implement the update and improvement of the parameterized EMS, so as to solve the energy management optimization in the continuous state-action space.

First, the training goal of the strategy network during iterative learning can be re-expressed as follows. Update the policy network parameters so that its action output at a given state, evaluated by the evaluation network, can achieve a higher action value, i.e., update the policy network parameters to obtain a higher action value. It can be seen from Figure 4.1 that the update direction can be mathematically expressed as the gradients of action value $Q(s, a)$ with respect to parameters θ^μ of the strategy network μ, which could be calculated by the chain rule, as shown in Equation (4.3):

$$\nabla_{\theta^\mu} \mu = \nabla_a Q(s, a|\theta^Q) \nabla_{\theta^\mu} \mu(s|\theta^\mu). \tag{4.3}$$

Equation (4.3) gives the policy gradient and then gradient descent can be applied for parameter update of the strategy network.

On the other hand, we should notice that to make the Actor update count, it is also necessary to ensure that the Critic network Q is constantly improving in terms of evaluative rationality during training. Hence, similar to DQN, the TD error is likewise defined as the loss function L_Q which is used to train the Critic network Q, as shown in Equation (4.4):

$$\begin{cases} Target = r + \gamma Q(s', \mu(s'|\theta^\mu)|\theta^Q) \\ L_Q(s, a|\theta^Q) = \left[Target - Q(s, a|\theta^Q)\right]^2, \end{cases} \tag{4.4}$$

where *Target* refers to the actual action value estimated by reward feedback.

Accordingly, the Critic network update gradient $\nabla_{\theta^Q} L_Q$ is calculated as follows:

$$\nabla_{\theta^Q} L_Q(s, a|\theta^Q) = 2\left[y - Q(s, a|\theta^Q)\right] \nabla_{\theta^Q} Q(s, a|\theta^Q). \tag{4.5}$$

The forward propagation (Equations (4.1) and (4.2)) and the backward propagation (Equations (4.3) and (4.4)) of AC networks are depicted in Figure 4.1 as well. Finally, the update of the two sets of AC networks is summarized as:

$$\begin{cases} \theta^Q = \theta^Q + \alpha_l \nabla_{\theta^Q} L_Q \\ \theta^\mu = \theta^\mu + \alpha_l \nabla_{\theta^\mu} \mu, \end{cases} \tag{4.6}$$

where α_l is the learning rate computed by ADAM method [100].

4.1.2 TRAINING OF DDPG-BASED EMSs

According to the strategy learning theory in Section 4.1, we can find that it is necessary to update the Actor network μ and the Critic network Q synchronously and alternately, and their updates are mutually dependent. This makes it critical to ensure stable learning of continuous DRL-based EMSs. This section will describe how to optimize the data flow of parameter updates to improve training stability.

First, similar to the DNQ method, experience replay is also used here to break the high correlation between sequential data and make full use of hardware computational resources, and network updates are performed using batch samples. The experience samples $e = (s, a, s', r)$ will be deposited into the experience pool $\mathcal{D}$ in a circular queue with the experience pool capacity $size = 10^4$ and minibatch size of 32.

Second, to reduce the variation of actual action value during strategy training, a replica of this AC networks, namely the target AC networks (Q_T and μ_T), are introduced, which are parameterized by θ_T^Q and θ_T^μ, respectively. During training, the parameters of the target networks will slowly track that of the original AC networks, as shown in Equation (4.7):

$$\begin{cases} \theta_T^Q = \theta_T^Q + \tau(\theta^Q - \theta_T^Q) \\ \theta_T^\mu = \theta_T^\mu + \tau(\theta^\mu - \theta_T^\mu), \end{cases} \tag{4.7}$$

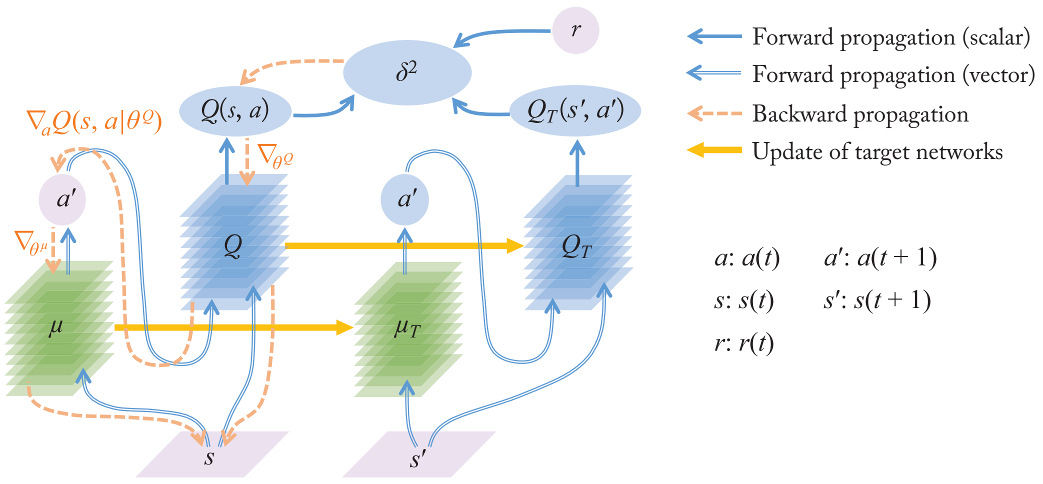


Figure 4.2: The dataflow graph of network forward propagation and gradient backward propagation.

where τ ($\tau \ll 1$) is the tracking rate of target AC networks' update.

As the target network parameters are updated relatively slowly, the calculated actual action value is also relatively stable, thus contributing to stable training. Then, the loss function can be redefined as:

$$L_Q(s,a|\theta^Q) = \mathbb{E}_{(s,a,s',r)\sim U(\mathcal{D})}\left[\left(r + \gamma Q_T(s', \mu_T(s'|\theta_T^\mu)|\theta_T^Q)\right) - Q(s,a|\theta^Q)\right]^2, \tag{4.8}$$

where the target AC networks are used to generate the actual action value target for the Critic update.

Therefore, the new update gradient of the AC network will be:

$$\nabla_{\theta^Q} L_Q = \mathbb{E}_{(s,a,s',r)\sim U(\mathcal{D})}\left[2\left(r + \gamma Q_T(s', \mu_T(s'|\theta_T^\mu)|\theta_T^Q) - Q(s,a|\theta^Q)\right)\nabla_{\theta^Q} Q(s,a|\theta^Q)\right]. \tag{4.9}$$

$$\nabla_{\theta^\mu}\mu = \mathbb{E}_{s\sim U(\mathcal{D})}\left[\nabla_a Q(s,a|\theta^Q)\nabla_{\theta^\mu}\mu(s|\theta^\mu)\right]. \tag{4.10}$$

For the two sets of AC networks, their dataflow graph of forward propagation and gradient back-propagation processes are depicted in Figure 4.2. The forward propagation will map the state vector to the action vector, while the gradient backward propagation will help update all network parameters for strategy improvement.

The dataflow graph of network forward propagation and gradient backward propagation With these two measures, the sable training of continuous DRL-based EMSs can be basically guaranteed. The next section will specify the details of the training method in the context of PHEV energy management.

4.2 CONTINUOUS ENERGY MANAGEMENT ENABLED BY TRIP INFORMATION

It is still challenging to derive a nearly optimal online EMS that is applicable in diverse driving cycles for a specific PHEV, especially when there is no priori knowledge of future trip information. To deal with the uncertain of future trip, short-term velocity predictors are introduced in many MPC-based EMSs: given a moving prediction horizon at each time step, optimal control algorithms can be used to solve the short-term energy management problem under the framework of MPC [32, 44]. Meanwhile, recent progresses in predictions based on deep learning [107, 108] and driving cycle reconstruction utilizing traffic data [12, 39] can also facilitate the relevant research. However, the application effects of these methods are still quite correlated with predictor accuracy and availability of traffic data. Some MPC-based EMSs are designed based on an average power concept when future trip information is unavailable [31, 109], yet this approach is more suitable for vehicles with regular and stable driving conditions. Therefore, it is beneficial to evaluate and integrate new methods into implementation of EMSs suitable for the entire trip and SoC range without priori knowledge.

On the other hand, for PHEVs, exploring and improving the strategy in continuous action space and full battery capacity range requires massive sample data, which is not conducive to accelerating the learning speed and may even cause policy learning failure. In this case, accurate analysis of vehicle and driving states, and efficient exploration guidance in a high-dimensional space will be helpful for strategy learning.

Therefore, in this section, history cumulative trip information is integrated for effective SoC guidance in continuous DRL-based EMSs. The overall schematic of this energy management method is shown in Figure 4.3. Generally, its implementation is divided into the following three periods.

(1) Period I. Deep reinforcement learning in simulation: this period is the training of EMSs. The DDPG algorithm is adopted to update the parameterized EMS in the simulation environment. The update is based on state transition data generated by iterative interactions among the SHEV model, its history cumulative trip information (HCTI), and the EMS.

(2) Period II. Strategy download: when the training gets convergent, parameters and structure of the neural network are saved as the trained EMS.

(3) Period III. Strategy online application: the downloaded EMS can be directly used for online application by simply mapping state to actions for energy management.

4.2.1 UTILIZATION OF HISTORY TRIP INFORMATION

As explained in Section 1.3, for PHEVs, BM mode provides better coordination of the onboard power sources compared to CD+CS mode and ensures gradual battery depletion along the whole trip. The EMS obtained by DP can be considered as the optimal BM strategy when

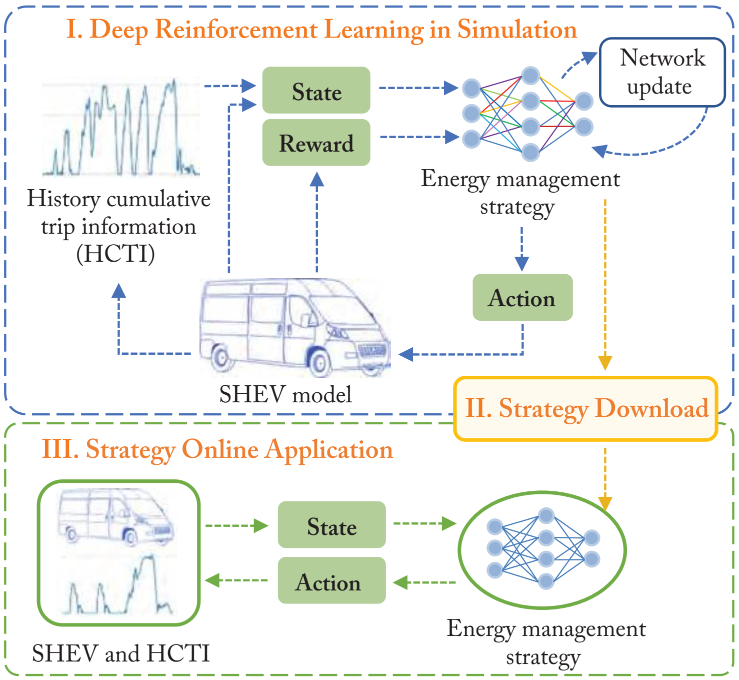

I. Deep Reinforcement Learning in Simulation
State
Reward
Network update
History cumulative trip information (HCTI)
Energy management strategy
Action
SHEV model
II. Strategy Download
III. Strategy Online Application
State
Action
SHEV and HCTI
Energy management strategy

Figure 4.3: The overall schematic of DRL-based EMS enabled by history trip information.

the entire trip information is known, and usually, its SoC trajectory also shows an evenly decreasing trend [19]. Therefore, we can assume that it will be helpful to search for a near-optimal EMS when a reasonable and appropriate electricity consumption guide is given.

When there is no external information source, the past driving information is the most convenient source to collect the driving cycle information. To understand the role of historical trip information in energy management, four trips with a distance of 100 km are constructed based on standard driving cycles (CTUDC, JN1015, MANHATTAN, WVUCITY), and DP-based EMSs are derived on these trips separately. The initial state and final state constraints are set as $SoC_{init} = 1.0$ and $SoC_{sust} = 0.20$, respectively. The results are shown in Figure 4.4 where all the SoC trajectories tend to decrease gradually to sustaining level over the entire trip despite different trips. This phenomenon and its better fuel economy of BM are also demonstrated in many research [23, 110], and Sun C. et al. also pointed out that space-domain-indexed SoC reference, SoC reference values indexed by space, can consistently bring in good fuel economy performance [12].

Therefore, to ensure that the SOC trajectory generated by DRL-based EMS is close to the global optimal SOC trajectory without priori knowledge of the future trip, a linear planning method of the SoC reference trajectory, utilizing HCTI to obtain the space-domain-indexed

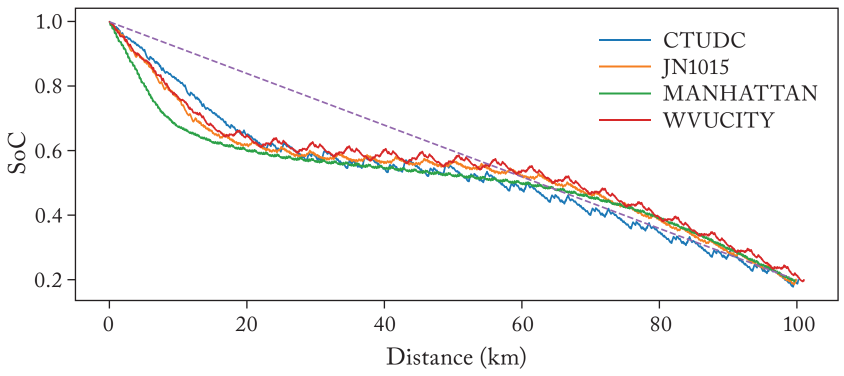


Figure 4.4: Global SoC trajectories optimized by DP.

SoC trajectory, is introduced here as the dashed line in Figure 4.4 shows. The details are described below.

Once the vehicle gets fully charged, the travel distance d will be set as the initial state, $d(0) = 0$ km. The space-domain-indexed SoC is calculated in discrete-time by:

$$\begin{cases} SoC_{ref}(t) = SoC_{init} - \lambda \min(d(t), L) \\ \lambda = (SoC_{init} - SoC_{sust})/L, \end{cases} \tag{4.11}$$

where $SoC_{ref}(t)$ is the space-domain-indexed SoC at moment t. λ is the descent rate of SoC in the space domain and L denotes the expected driving range with a fully charged battery, 100 km.

Besides, the travel distance d after last charge, the SoC deviation ΔSoC (Equation (4.12)) are added as new state variables into state space S:

$$\Delta SoC_{ref}(t) = SoC(t) - SoC_{ref}(t). \tag{4.12}$$

4.2.2 REWARD DEFINITION

Because the driving mileage is difficult to predict in advance and the battery power is limited, it is hard to obtain the optimal EMS over the entire trip without *a priori* information about the future global driving cycle. Thus, the PHEV working modes are set according to the expected maximum driving range L: the PHEV works at BM within the driving range ($d(t) < L$), and switches to CS mode to meet the driving demand as much as possible when $d(t) \geq L$, although the vehicle still should be charged in time at this circumstance.

Reward for BM: Energy Management Enabled by History Trip Information

Within the expected driving range (100 km), the SoC of this SHEV is expected to decrease evenly. Meanwhile, the fuel consumption needs to be minimized during the whole trip. Thus, the reward signal for the training of BM is defined as:

$$r = -\tanh\left(\chi \dot{m}_{fuel} + \varphi \left|\Delta SoC_{ref}\right|\right), \tag{4.13}$$

where χ and φ denote the factor of fuel consumption rate and SoC deviation, respectively. Intuitively, a larger φ means that tracking the SoC guidance more closely can get higher reward, while a larger χ means that reducing fuel consumption can bring higher reward. The basic principle for parameter tuning of φ and χ is to improve the fuel economy as much as possible while ensuring that the trained strategy can always meet SoC requirements; in this chapter, φ and χ are set to be 50 and 150, respectively, after repeated tuning.

The main influence factors are summarized as follows.

(1) Different strategies have different requirements for SoC. The SoC requirements are as follows: for the blended mode of EMS, the SoC needs to track the changing reference value derived from history cumulative trip information, while for CS mode of EMS, the SoC needs to maintain the constant sustaining level. So, due to the different SoC requirements, the two parameters need to be adjusted accordingly for better learning as well.

(2) A large χ may slow down the learning process. Because of the nonlinear engine efficiency map and changing strategies during training, the signal of fuel consumption rate is much nosier and irregular compared with the signal of SoC deviation, making it difficult to learn, especially at the beginning period when the possibility of taking a random strategy is 1. Thus, the fact is that increasing χ in reward signal does not necessarily make the learning better, and it is likely to lead to instability or even divergency in DRL-based EMSs' training.

On the other hand, a larger χ can make the strategy learn to track the SoC guidance more quickly at the beginning period. As long as the item of SoC deviation decreases gradually, the item of fuel consumption rate will be considered more in the learning. Therefore, a proper parameter tuning should allow the strategy learning to consider more about satisfaction of SoC deviation at first and, in a later period, to consider more about fuel economy.

(3) The two parameters cannot be too small. As all the networks are initialized with random weights by Xavier uniform initializer, the initial output of Critic should be a number close to zero. Meanwhile, as shown by the loss function of Critic, the update of Critic is highly depended on the reward r. Therefore, in the beginning of training, if reward r is too small than the initial output of Critic, the update may get misguided by the initial output of Critic, which is harmful to stable training.

As a result, the two parameters could be set to be the same at first. Then, we can adjust them accordingly with evaluation of the training process, including the stability, convergence speed, final performance of the strategy.

The history trip information is considered both in reward signal by $\left|\Delta SoC_{ref}\right|$ defined in Equation (4.13), and in input state vector s that belonging to state space $s \in$

$S = \{SoC, P_{eng}, v, acc, d, \Delta SoC_{ref}\}$, to derive the DRL-based EMS (blended mode), $\pi_{BM} = \mu_{BM}(s|\theta^{\mu})$.

π_{BM} is trained on one training driving cycle with fixed initial SoC of 0.8, which is assumed to be the same as SoC_{ref}. The initial observation state is $s(0) = [0.8, 0, 0, 0, 0, 0]$. The training driving cycle will be given in the next section.

Reward for CS Mode: Energy Management for Charge Sustaining

Once the SHEV has traveled beyond the expected driving range and still has not been charged in time, the EMS should switch to CS mode: keep SoC stable near the low level while balancing fuel economy. New Actor-Critic networks are established for EMS of CS mode: $\pi_{CS} = \mu_{CS}(s|\theta^{\mu})$. As there is no need updating the space-domain-indexed SoC, we can define the state space as $S = \{SoC, P_{eng}, v, acc, \Delta SoC_{CS}\}$, where $\Delta SoC_{CS} = SoC - SoC_{sust}$. The corresponding reward function is basically consistent with Equation (3.10), but the SoC_{sust} is set as the lower limit of the SoC feasible domain (0.2). Here, χ and φ are set to be 40 and 36, respectively, after repeated tuning.

CS mode of DRL-based EMS is trained on the same training driving cycle but with random initial SoC from $[0.1, 0.3]$, denoted SoC_{rand}, making sure the EMS can switch to CS mode successfully regardless of different final SoC values at the end of blended mode. The initial observation state $s(0) = [SoC_{rand}, 0, 0, 0, (SoC_{rand} - SoC_{sust})]$.

Instead of applying the penalty term in the reward function, a more flexible solution for the issue of frequent engine starts is illustrated in Section 4.2.5 by adjusting the strategy output frequency.

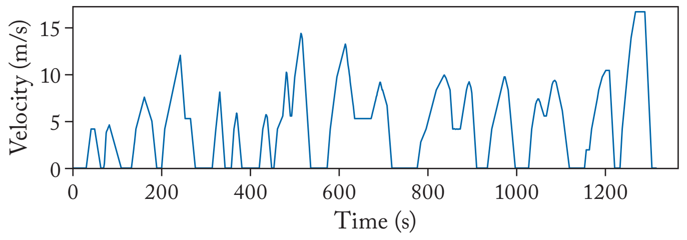
4.2.3 TRAINING ALGORITHM

As shown in Figure 4.5, the China typical urban driving cycle (CTUDC) is selected as the trip for training of DRL-based EMS; to test its adaptability, a mixed driving cycle is constituted as the trip for test. The length of one trip for training and testing is 5.898 km (1314 s) and 66.971 km (8166 s), respectively.

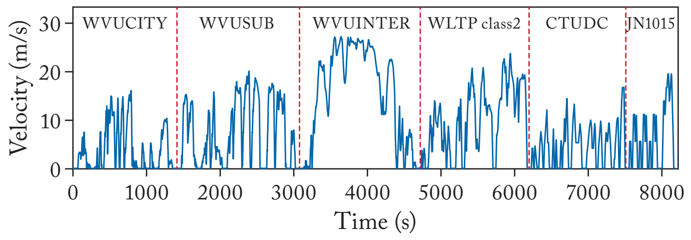
The characteristics of each driving cycle are summarized in Table 4.1, which indicates that the test trip is statistically different with the training trip, making relevant verifications more objective.

Based on the learning theory of DDPG-based EMSs, the training of EMS enabled by cumulative history trip information for PHEVs is described as follows. First, the training of blended mode strategy, π_{BM}.

Before feeding the data into AC networks for training, data normalization is required, which is similar to that in Section 3.3.4. The travel distance d is normalized linearly into $[0, 1]$ by the expected range L. Originally, because SoC and ΔSoC_{ref} is relatively small, they can be fed into the networks directly.



(a) Trip for training



(b) Trip for testing

Figure 4.5: The driving cycles for training and testing.

Table 4.1: Characteristics of selected driving cycles

Driving Cycles	Max. Vel. (m/s)	Avg. Vel. (m/s)	Max. Accel. (m/s^2)	Max. Decel. (m/s^2)	Time Ratio ($Time_{acc}/Time_{dec}$)
CTUDC	16.67	4.49	0.91	−1.04	1.44
WVUCITY	16.01	3.78	1.14	−3.24	1.30
WVUSUB	20.02	7.19	1.29	−2.16	1.28
WVUINTER	27.15	15.22	1.42	−1.86	1.06
WLTP Class2	23.67	9.92	1.00	−1.17	1.15
JN1015	19.44	6.30	0.79	−0.83	1.11
Mixed	27.15	8.20	1.42	−3.24	1.21

Next, with experience relay and the update of space-domain-indexed SoC reference, the training of π_{BM} can be realized, which is illustrated in Figure 4.6. The initialization method for AC networks adopted is Xavier uniform initializer [102]. In ε-greedy exploration, the random process $\mathcal{R}$ will generate random actions from interval $[-10, 10]$; the probability of strategy exploration and exploit are ε and $1-\varepsilon$, respectively, with ε annealing gradually with training.

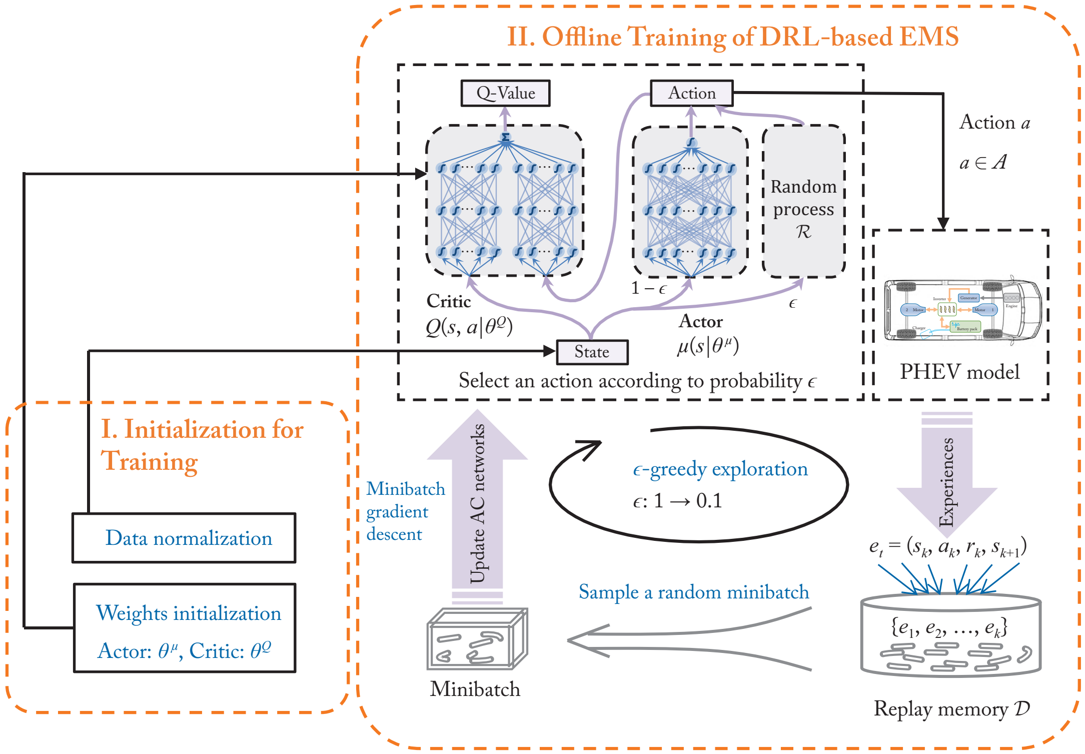

II. Offline Training of DRL-based EMS
Q-Value
Action
Action a
$a \in A$
Random process $\mathcal{R}$
Critic
$Q(s, a|\theta^Q)$
$1-\epsilon$
ϵ
Actor
$\mu(s|\theta^\mu)$
State
Select an action according to probability ϵ
PHEV model
I. Initialization for Training
Data normalization
Weights initialization
Actor: θ^μ, Critic: θ^Q
Minibatch gradient descent
Update AC networks
ϵ-greedy exploration
ϵ: $1 \rightarrow 0.1$
Experiences
$e_t = (s_k, a_k, r_k, s_{k+1})$
Sample a random minibatch
$\{e_1, e_2, \ldots, e_k\}$
Minibatch
Replay memory $\mathcal{D}$

Figure 4.6: The training of DDPG-based EMSs.

The pseudocode for training of π_{BM} is shown in Algorithm 4.3, where the *IterationRound* is counted from the first update. The training of π_{CS} is generally the same with π_{BM}, but requires three changes: (1) line 9 needs to be deleted; (2) line 10 is replaced by "Obtain next state of SoC deviation by $\Delta SoC_{CS}(t+1) = SoC(t+1) - SoC_{sust}$;" and (3) line 11 is replaced by "Collect new state $s(t) = [SoC(t+1), P_{eng}(t+1), v(t+1), acc(t+1), \Delta SoC_{CS}(t+1)]$."

4.2.4 LEARNING EVALUATION

With a processor of i7-2600CPU@3.4GHz, we trained the DDPG-based EMS for 96 rounds of updates, and the strategy is updated iteratively every second during the training trip. The total training time is about 9.7 min on average. The learning of both modes (BM and CS mode) is discussed as follows.

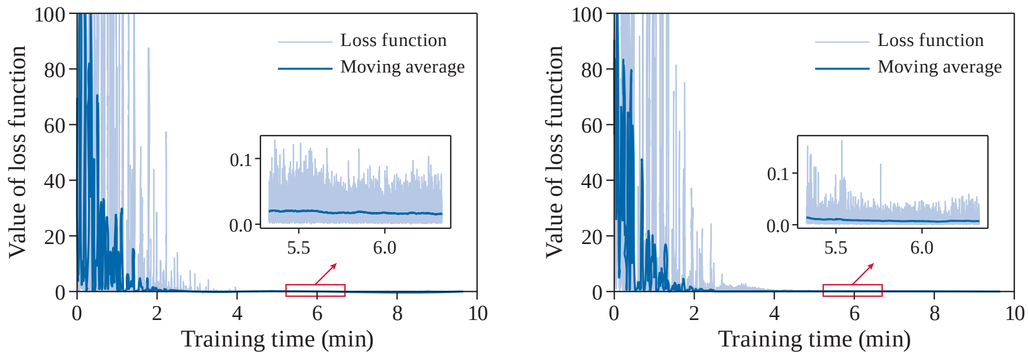
(1) The learning of EMS for Blended Mode: the loss function of DDPG-based EMS for BM shows apparent convergence after iteratively training as shown in Figure 4.7a, which means the estimation of action-value function has been basically stable and can gradually provide more stable guidance for the update of EMS (BM).

Algorithm 4.3 The training of DDPG-based EMSs.

Initialize AC networks by Xavier uniform initializer; initialize target AC networks; initialize experience pool $\mathcal{D}$; initialize exploration rate $\varepsilon = 1$; maximum iteration number $N_{iteration}$; time length of the training driving cycle T_{CTUDC}; size of minibatch $Minibatch = 32$.

```
1: for $IterationRound = 1 : N_{iteration}$ do
2:     Observe initial state $s(0)$
3:     for $t = 0 : (T_{CTUDC} - 1)$ do
4:         Select action by current policy $a(t) = \mu(s(t)|\theta^{\mu})$ (with probability $1 - \varepsilon$) or random
           generation $a(t) = \mathcal{R}(s(t))$ (with probability $\varepsilon$)
5:         Execute action $a(t)$ by calling the SHEV model
6:         Observe reward $r(s(t), a(t))$ and the next state $s(t+1)$
7:         Observe next state, including: $[SoC(t+1), P_{eng}(t+1), v(t+1), acc(t+1)]$
8:         Update history trip information $d(t+1) = d(t) + v(t)$
9:         Get SoC deviation $\Delta SoC_{ref}(t+1) = SoC(t+1) - [SoC_{init} - \lambda d(t+1)]$
10:        Collect new state vector $s(t+1) = [SoC(t+1), P_{eng}(t+1), v(t+1), acc(t+
           1), d(t+1), \Delta SoC_{ref}(t+1)]$
11:        Store $e(t) = (s(t), a(t), r(s(t), a(t)), s(t+1))$ into $\mathcal{D}$
12:        if $\mathcal{D}$ is full then
13:            Sample a random minibatch of experiences $e_j$ $(j = 1, 2, ..., Minibatch)$
14:            Calculate actual action value of $e_j$: $Target_j = r + \gamma Q_T(s', \mu_T(s'|\theta_T^{\mu})|\theta_T^{Q})$
15:            Calculate loss function of $e_j$: $L_Q^j = [Target_j - Q(s, a|\theta^Q)]^2$
16:            Update the Critic network to minimize the loss function $L_Q = \frac{\sum_j L_Q^j}{Minibatch}$
17:            Calculate the policy gradient of $e_j$: $\nabla_{\theta^{\mu}}^{j} \mu = \nabla_a Q(s, a|\theta^Q) \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})$
18:            Calculate the average policy gradient $\nabla_{\theta^{\mu}} \mu = \frac{\sum_j \nabla_{\theta^{\mu}}^{j} \mu}{Minibatch}$ and update the Actor net-
               work
19:            Update the target Critic network $Q_T$: $\theta_T^Q = \theta_T^Q + \tau(\theta^Q - \theta_T^Q)$
20:            Update the target Actor network $\mu_T$: $\theta_T^{\mu} = \theta_T^{\mu} + \tau(\theta^{\mu} - \theta_T^{\mu})$
21:            Anneal exploration possibility: $\varepsilon = \max(0.1, \varepsilon - (N_{iter} T_{CTUDC})^{-1})$
22:        end if
23:    end for
24: end for
25: Output the final parameterized strategy network $\mu$ as the trained $\pi_{BM}$
```

The convergence of Actor network, π_{BM}, is reflected by the change of final SOC after each update round of one training trip as shown in Figure 4.8a. The final SoC varies obviously



(a) Loss for blended mode

(b) Loss for CS mode

Figure 4.7: The loss function of Critic during training.

in the beginning period of training, and gradually converges to terminal reference SoC (0.7528, calculated by Equation (4.11)) with updating. Because regenerative braking at the end of the training trip will lead to SoC rise, so the strategy finally learns to sacrifice the consistency of final SoC for higher mean rewards during the entire training trip, as indicated in Figure 4.8a.

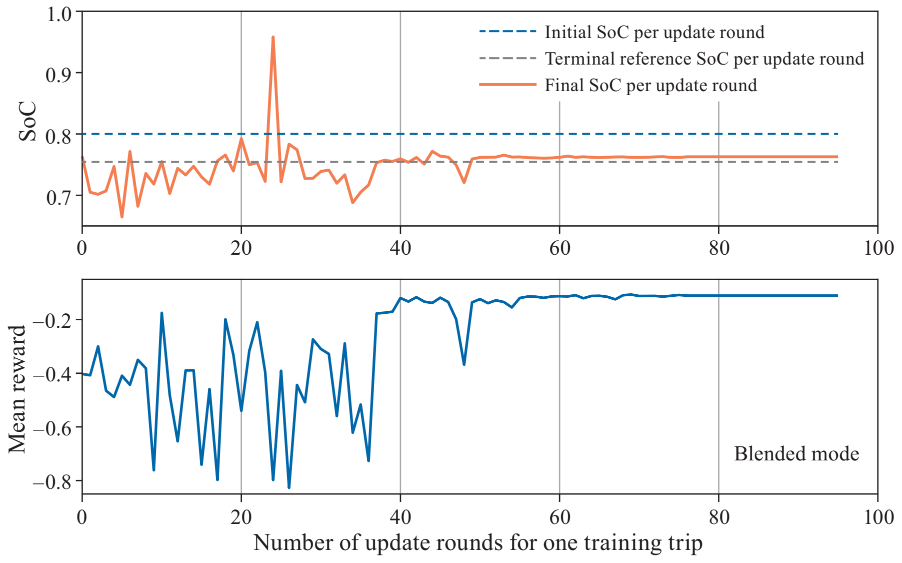
(2) The learning of EMS for Charge Sustaining Mode: after about 4 min of training, the loss function of EMS for the CS model has converged significantly, as shown in Figure 4.7b. Compared with BM, the CS model shows an overall smaller training loss because the SoC sustaining goal remains constant all the time, making it easier to learn.

This convergence is also reflected by indicators of EMS (CS Mode) as shown in Figure 4.8b. We vary the initial SoC randomly at the beginning of each update round as the gray line shows. Accordingly, the final SoC varies during the initial training period. But gradually, it converges to the sustaining level regardless of diverse initial SoC values. The mean reward also rises gradually, but not as smooth as that in the training of BM due to the diversity in initial state.

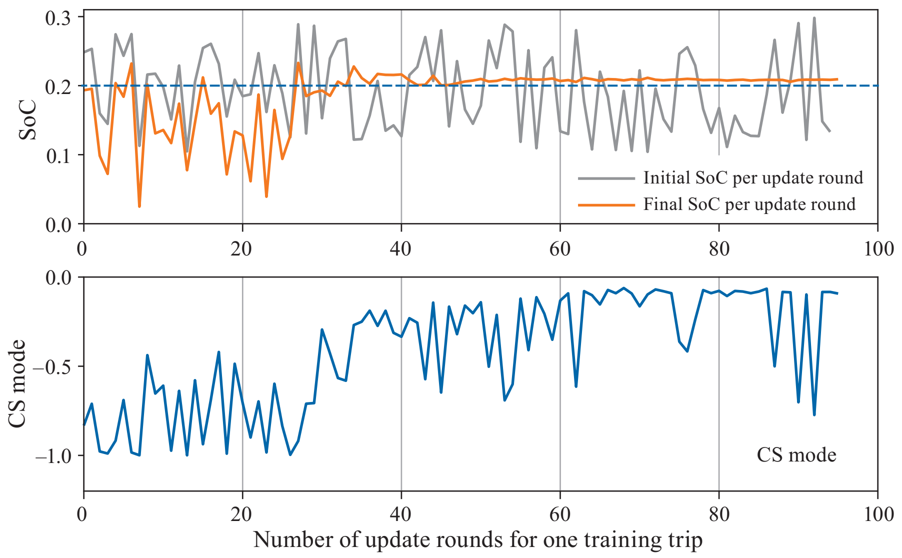
4.2.5 OPTIMIZATION EVALUATION

The online application of DDPG-based EMSs is illustrated in Figure 4.9. As the operation of the engine is decoupled with driving requirements, thus to avoid frequent engine starts and shutoffs, we adjust the original output control signal to a lower frequency, which is referred to as output frequency adjustment in the following sections: clip control signals within these 10 s by interval $[\min, \max] = [a(t_0) - 1, a(t_0) + 1]$ (kW), and $a(t_0)$ is updated every 10 s, thus limiting the strategy output in a rolling time domain.

The derived DDPG-based EMSs, both the BM strategy and the CS mode strategy, are tested on one original training trip to examine its optimality. The DP-based EMS is still consid-



(a) The blended mode



(b) The CS mode

Figure 4.8: The initial SoC, final SoC, and mean reward during EMS training.

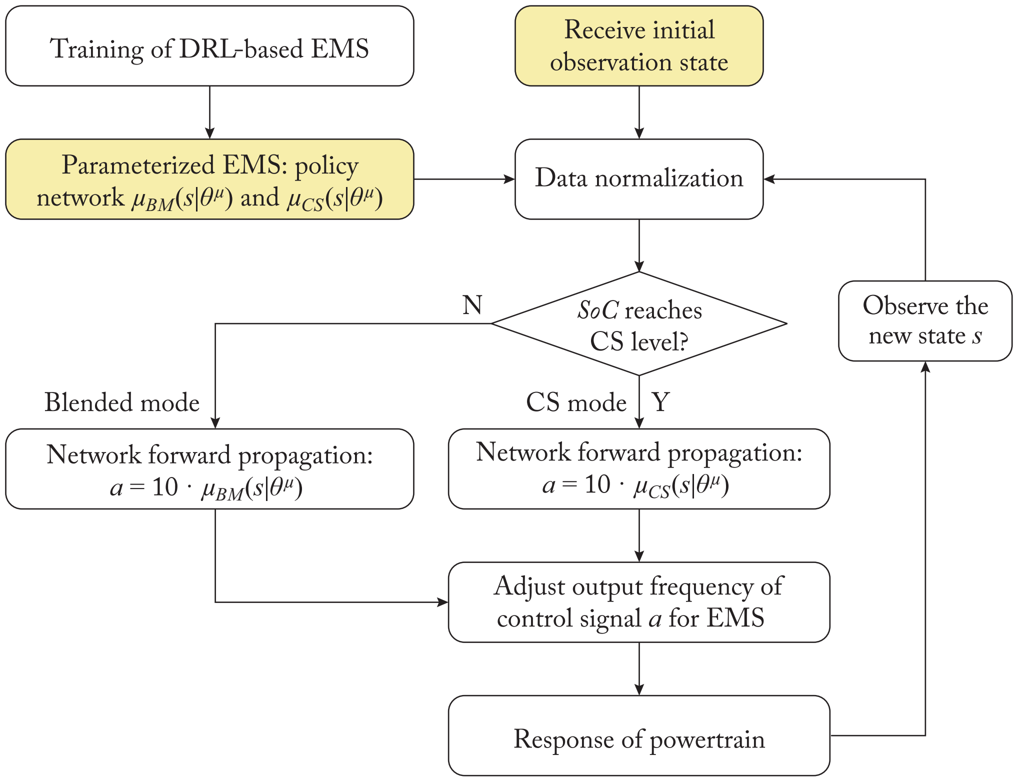

Training of DRL-based EMS
Receive initial observation state
Parameterized EMS: policy network $\mu_{BM}(s|\theta^{\mu})$ and $\mu_{CS}(s|\theta^{\mu})$
Data normalization
SoC reaches CS level?
N
Y
Observe the new state s
Blended mode
CS mode
Network forward propagation: $a = 10 \cdot \mu_{BM}(s|\theta^{\mu})$
Network forward propagation: $a = 10 \cdot \mu_{CS}(s|\theta^{\mu})$
Adjust output frequency of control signal a for EMS
Response of powertrain

Figure 4.9: The online application of DDPG-based EMS for a PHEV.

ered as the benchmark with the best fuel economy. Meanwhile, for validity verification of output frequency adjustment in online applications of DRL-based EMSs, additional simulations are carried out. The abbreviations and meanings of the specific EMSs involved here are as follows.

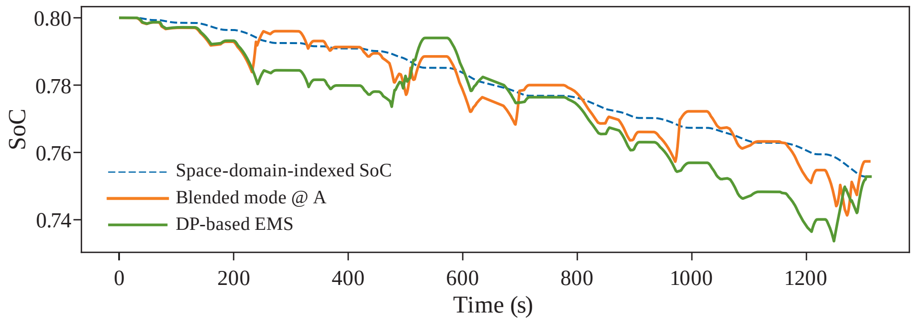
(a) DP-based EMS: DP-based EMS as described in Section 3.4.1.

(b) Blended Mode@O: Original BM of DRL-based EMS without output frequency adjustment.

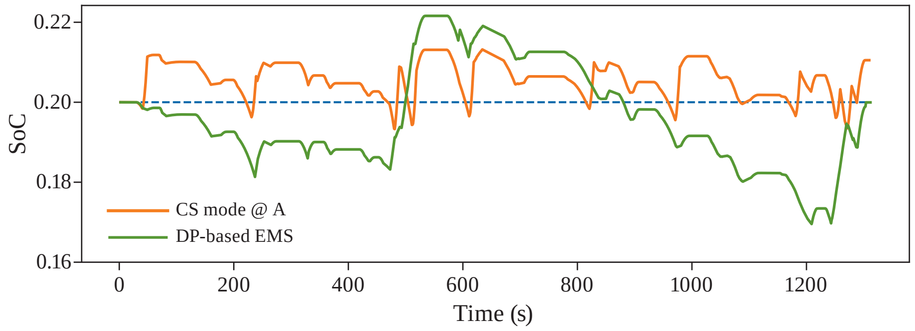
(c) Blended Mode@A: BM of DRL-based EMS with output frequency adjustment.

(d) CS Mode@O: Original CS mode of DRL-based EMS without output frequency adjustment.

(e) CS Mode@A: CS mode of DRL-based EMS with output frequency adjustment.



(a) Blended mode @ A and DP-based EMS



(b) CS mode @ A and DP-based EMS

Figure 4.10: SoC trajectories of different EMSs on one training trip.

First, the BM strategy is examined. All initial SoC values are set as 0.8. For DP-based EMSs, the final SoC is constrained by the expected final SoC after one training trip in DRL-based EMS, which is calculated to be 0.7528.

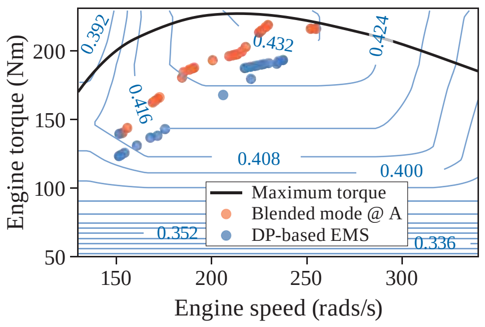
The SoC trajectories of Blended Mode@A and DP-based EMS are shown in Figure 4.10a. Overall, the trajectory of Blended Mode@A follows the space-domain-indexed SoC more closely, realizing an even descent of SoC over the training trip. The deviation of final SoC for Blended Mode@A is 0.0046, indicating the history trip information has been effectively considered by the BM strategy.

The detailed simulation results for one training trip are shown in Group No. 1, Table 4.2. For fairly comparison, the fuel consumption should be compensated because of varies SoC final values [111], so the deviation between final SoC and expected final SoC has been compensated in fuel consumption calculation by vehicle model [12]. From Group No. 1, the compensated fuel consumption of Blended Mode@O only has a gap of 5.31% compared with the DP-based EMS, which is less than Blended Mode@A (8.55%), but generally, they both have achieved good fuel economy. The engine operating points and its distribution of efficiency are depicted

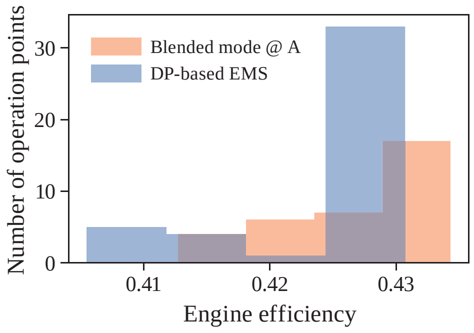
Table 4.2: Simulation results of DRL-based EMS for one training trip

Group	Method	SoC_{init}	SoC_{final}	N_{eng}	T_{cal} (s)	$Fuel_c$ (g)	Gap_{fuel_c}
1	DP	0.8000	0.7528	8	24.9478	129.9	—
	BM@O	0.8000	0.7548	26	1.3065	136.8	5.31%
	BM@A	0.8000	0.7574	6	1.4684	141.0	8.55%
2	DP	0.2000	0.2000	11	23.9792	228.1	—
	CS@O	0.2000	0.2079	51	1.2470	235.0	3.02%
	CS@A	0.2000	0.2105	10	1.2810	248.5	8.94%

Notes: N_{eng}: engine start times, T_{cal}: calculation time, Gap_{fuel_c}: the gap of fuel economy with the DP-based EMS.



(a) Engine operating points



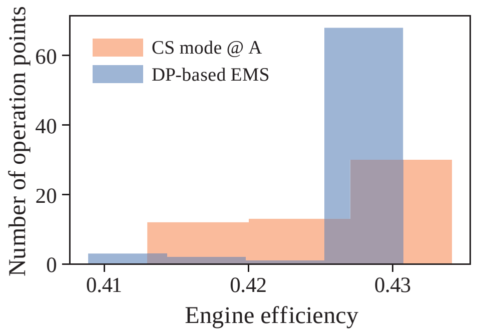
(b) Operation efficiency distribution of the engine

Figure 4.11: Simulation results of Blended Mode@A.

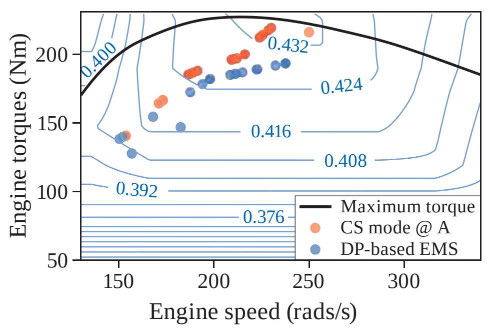
in Figure 4.11 and the average operation efficiency of Blended Mode@A and DP-based EMS is 0.4254 and 0.4260, respectively. This demonstrates that Blended Mode@A can guarantee an efficient engine operation, verifying its optimality on the training trip.

On the other hand, with output frequency adjustment, the number of engine starts in Blended Mode@A decreases dramatically from 26 to 6, which is also less than that of the DP-based EMS, verifying the validity of output frequency adjustment in the application of Blended Mode@A.

Second, the CS mode is examined. All initial SoC values are set as the sustaining level of 0.2. For the DP-based EMSs, its final SoC is constrained by sustaining level of SoC. Figure 4.10b shows the SoC trajectories of CS Mode@A and DP-based EMS. Similarly, the SoC trajectory of CS Mode@A is closer to the target sustaining level than the DP-based EMS, indicating the CS Mode@A puts more emphasis on SoC sustaining under current parameter



(a) Engine operating points



(b) Operation efficiency distribution of the engine

Figure 4.12: Simulation results of CS Mode@A.

settings. Fundamentally, this phenomenon is caused by differences in the way the two policies are derived. However, it is worth noting that increasing the weight of fuel consumption in reward signal does not necessarily make the SoC trajectory closer to that of DP-based EMS, and it is likely to lead to instability or even divergency in DRL-based policy's training, which is probably because the fuel consumption signal is much noisier, making it harder to learn. Despite this point, the CS Mode@A has successfully realized charge sustaining of SoC near the sustaining level.

The corresponding simulation results are also shown in Group No. 2, Table 4.2. CS Mode@O achieves quite good fuel economy with a gap of only 3.02% compared with the DP-based EMS, but CS Mode@A remarkably reduces the number of engine start times from 51 to 10, even less than the baseline strategy, and achieves reasonable fuel economy with the gap of 8.94% from the DP-based EMS. Corresponding engine operating points and distribution of efficiency are depicted in Figure 4.12, and the average operation efficiency of CS Mode@A and DP-based EMS is 0.4256 and 0.4282, respectively, which are similar to simulation results of Blended Mode@A.

4.2.6 GENERALIZATION EVALUATION

CS Mode@A with Different Initial SoC

Two initial SoC values are chosen, 0.25 and 0.15, respectively, to examine whether the DDPG-based EMS can switch to CS mode smoothly with different final SoC values.

Figure 4.13 shows the simulation results, indicating that whatever the initial SoC is at the beginning of CS mode, SoC sustaining can be appropriately realized by CS Mode@A. Detailed results are summarized in Table 4.3.

On the other hand, comparing Group No. 1 in Table 4.2 and Group No. 1 in Table 4.3, it is interesting to notice that the BM strategy achieves better fuel economy than the CS Mode with

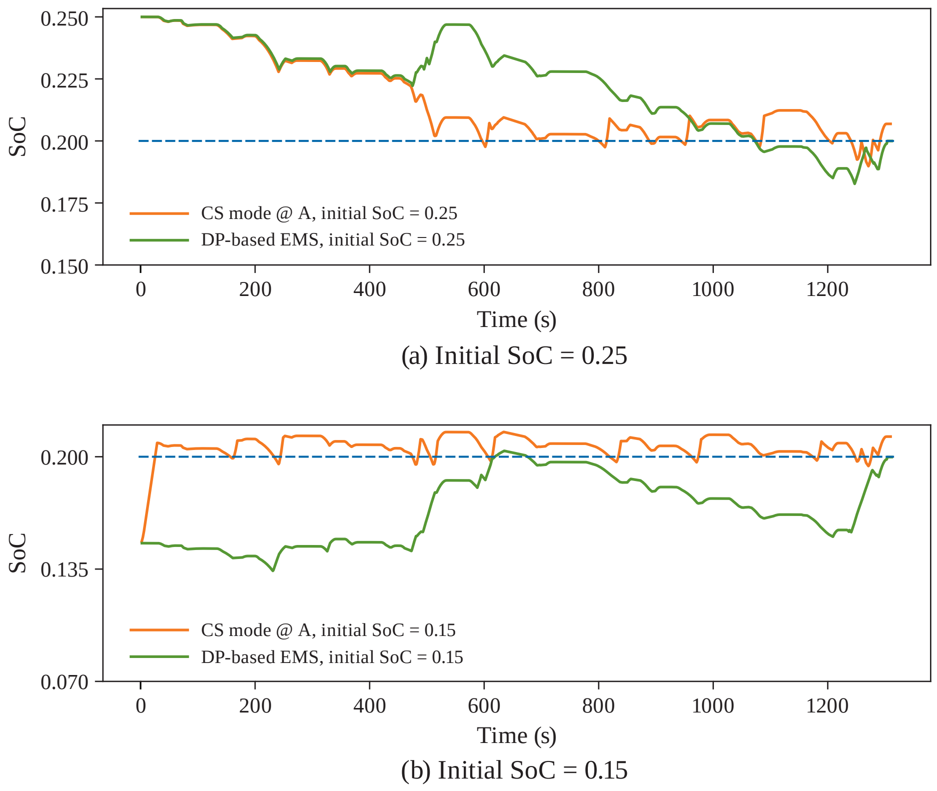


Figure 4.13: SoC trajectories of CS Mode@A with different initial SoC values.

or without output frequency adjustment. This proves, from another perspective, that utilizing history trip information in strategy learning is indeed beneficial for the improvement of fuel economy before SoC reaches the sustaining level.

Performance on the Long Training Trip

To examine the performance of DDPG-based EMSs, including BM@A and CS@A, it will be tested on a long trip (total mileage of about 106.157 km) consisting of 18 consecutive training trips.

In addition, we take the obtained SoC trajectory of the DP-based EMS as *a priori* and derive a DDPG-based EMS; this priori is utilized as the time-domain indexed SoC reference. To derive this method, the original SoC deviation in state space is replaced by a new SoC deviation from the time-domain indexed SoC reference, and the network parameters remain unchanged. Finally, three methods are compared as follows.

(1) DP: the DP-based EMS with initial SoC of 1.000 and final SoC constrain of 0.2000.

Table 4.3: Generalization test results of the DDPG-based EMS in CS mode

Group	Method	SoC_{init}	SoC_{final}	N_{eng}	T_{cal} (s)	$Fuel_c$ (g)	Gap_{fuel_c}
1	DP	0.2500	0.2000	7	26.9772	132.2	—
	CS@O	0.2500	0.2090	31	1.2314	140.7	6.43%
	CS@A	0.2500	0.2069	6	1.2945	147.2	11.35%
2	DP	0.1500	0.2000	10	24.2241	324.5	—
	CS@O	0.1500	0.2079	52	1.2976	338.4	4.28%
	CS@A	0.1500	0.2116	11	1.3160	354.5	9.24%

Notes: N_{eng}: engine start times, T_{cal}: calculation time, Gap_{fuel_c}: the gap of fuel economy with the DP-based EMS.

Table 4.4: Simulation results of DRL-based EMS for a long training trip

Group	Method	SoC_{init}	SoC_{final}	N_{eng}	T_{cal} (s)	$Fuel_c$ (g)	Gap_{fuel_c}
1	DP	1.00	0.2000	80	400.2769	2473.3	—
2	DDPG@T	1.00	0.2276	76	22.1353	2711.6	9.63%
3	DDPG@S	1.00	0.2090	108	22.8873	2689.9	8.76%

Notes: N_{eng}: engine start times, T_{cal}: calculation time, Gap_{fuel_c}: the gap of fuel economy with the DP-based EMS.

(2) DDPG@T: the DDPG-based EMS utilizing the SoC trajectory of the DP-based EMS as SoC reference, which is indexed by time.

(3) DDPG@S: the DDPG-based EMS utilizing history trip information in space-domain.

On the long trip, the SoC of DDPG@S descents evenly within the expected driving range and then maintains at sustaining level as shown in Figure 4.14, which is consistent with the expected performance. The DP-based EMS chooses to consume electricity quickly at first and then keeps SoC drops slowly between 0.6 and 0.4. This is because the energy loss in battery is relatively less within this SoC range due to smaller internal resistance, but it is hard to derive such policy without the entire driving profile. The SoC of DDPG@T is quite close to that of DP-based EMS with this priori.

Detailed results are summarized in Table 4.4. The fuel economy performance gap of DDPG@S is 8.76% which is consistent with performances on a short training trip. However, in this scenario, the engine starts more by even SoC descent than the baseline strategy. The fuel economy performance gap of DDOG@T is slightly larger but its number of engine start

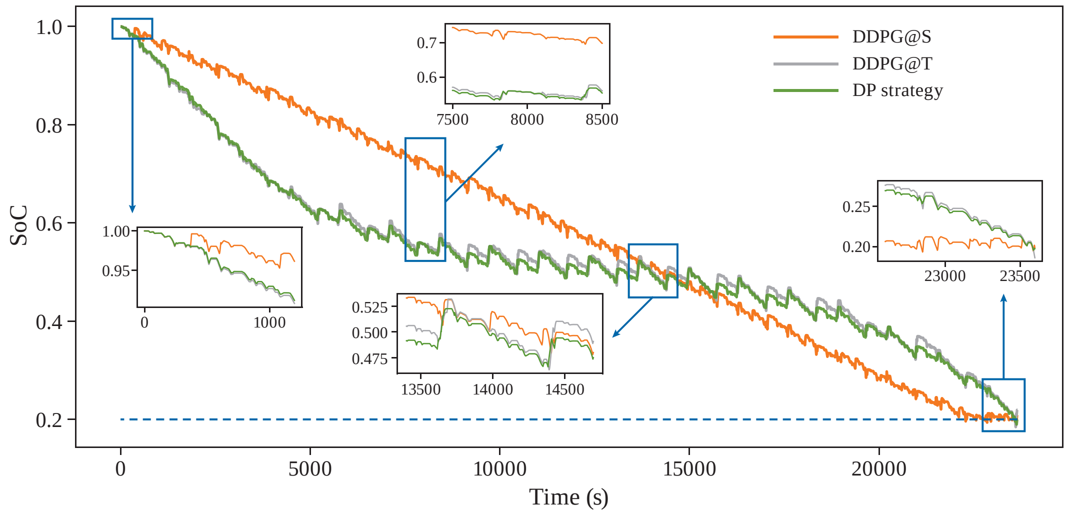


Figure 4.14: SoC trajectories of the DDPG-based EMS and DP-based EMS.

decreases dramatically, even less than the DP-based EMS, which could be much more beneficial for energy-saving in practice. Therefore, studying how to combine global SoC trajectory planning methods with DRL-based EMS should be a potentially valuable research direction. Overall, the performance of the DDPG-based EMS on a long training trip is in line with expectations.

4.3 COMPARISON BETWEEN CONTINUOUS DRL-BASED AND MPC-BASED EMSs

4.3.1 THE CONTRAST METHOD: MPC-BASED EMSs

Similar to the EMS described in this chapter that utilizes history trip information to obtain a space-domain-indexed global SoC trajectory as guidance, classic MPC-based EMSs also require a global SoC trajectory as a constraint. So, MPC-based EMSs are introduced here as the contrast strategies, as shown in Figure 4.15. In this brief, the MPC-based EMS takes the same space-domain-indexed global SoC trajectory as a constraint for terminal SoC regulation in each control horizon; DP is adopted to solve the control problem when predicted velocities are provided in each receding horizon (10 s after the current moment).

By comparing three kinds of driving cycle predictors, including velocity predictors using the exponentially varying method, the Markov-chain method, and the neural network method, Sun et al. showed that the prediction error over the next 10 s ranged from 2 m/s to 4 m/s [44]. So, to evaluate the MPC-based EMS more precisely, the horizon velocity predictor here works by adding certain Gaussian noises with specific standard deviation to the real future velocities in

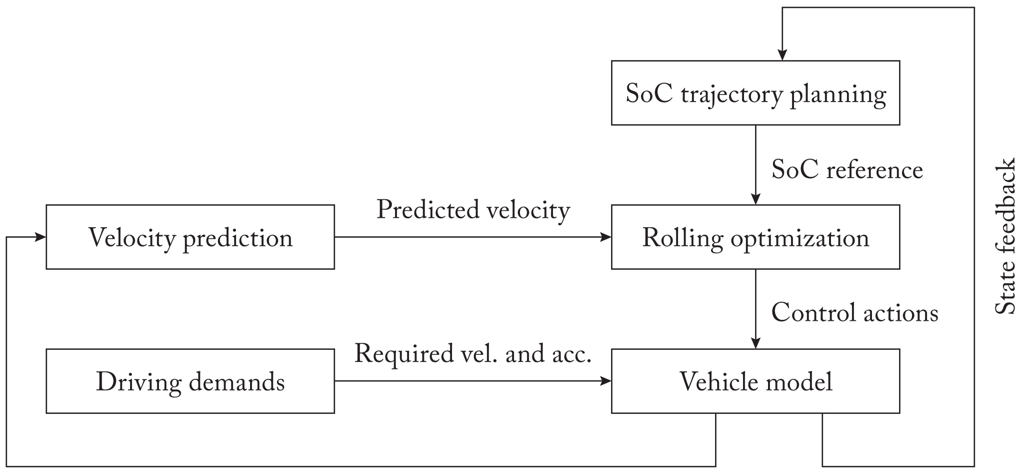


Figure 4.15: The schematic diagram of MPC-based EMSs.

each receding horizon, resulting in three kinds of MPC-based EMSs with different prediction errors as follows.

(1) MPC-based EMS@0: real future velocity is adopted as the predicted velocity in each receding horizon, i.e., this is an exactly accurate prediction.

(2) MPC-based EMS@1: add Gaussian noise with a standard deviation of 1 (m/s) to real future velocity as the predicted velocity in each receding horizon, i.e., the prediction error is 1 m/s.

(3) MPC-based EMS@2: add Gaussian noise with a standard deviation of 2 (m/s) to real future velocity as the predicted velocity in each receding horizon, i.e., the prediction error is 2 m/s.

4.3.2 COMPARISON UNDER BLENDED MODE

To examine how the trained DRL-based EMS performs on different driving cycles, the testing trip is utilized for generalization verification. Three MPC-based EMSs are also applied for contrast comparisons.

The SoC trajectories of different EMSs on one testing trip are shown in Figure 4.16. Overall, the DDPG-based EMS and MPC-based EMSs share a similar descent rate of electricity consumption, both avoiding the monotonous mode of EV + CS.

Detailed results are summarized in Table 4.5. The fuel economy gap of DDPG-based EMS@A with the benchmark is about 7.86%, which is consistent with the performance on the

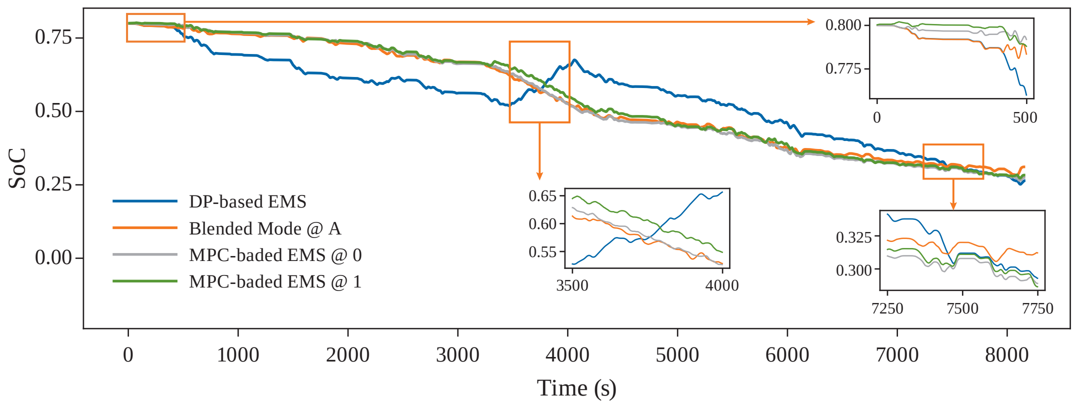


Figure 4.16: SoC trajectories of Blended Mode strategies on the testing trip.

Table 4.5: Simulation results of Blended Mode strategies for the testing trip

Group	Method	SoC_{init}	SoC_{final}	N_{eng}	T_{cal} (s)	$Fuel_c$ (g)	Gap_{fuel_c}
1	DP	0.80	0.2642	122	115.8667	2761.4	—
2	BM@O	0.80	0.2972	243	7.8423	2847.5	3.12%
3	BM@A	0.80	0.3111	87	7.8624	2978.5	7.86%
4	MPC@0	0.80	0.2750	519	1405.0662	2954.8	7.00%
5	MPC@1	0.80	0.2831	524	1405.8996	3005.2	8.83%
6	MPC@2	0.80	0.2947	509	1405.3478	3022.4	9.45%

Notes: N_{eng}: engine start times, T_{cal}: calculation time, Gap_{fuel_c}: the gap of fuel economy with the DP-based EMS.

training trip. Meanwhile, the number of engine starts dramatically decreases by about 28.7% than the benchmark, and this is even better than its performance on a long training trip.

Compared with MPC-based EMSs, the fuel economy of DDPG-based EMS@A is between MPC-based EMS@0 and MPC-based EMS@1, and is better than MPC-based EMS@2, but it is much more superior in avoiding frequent engine start and computational speed, because the network forward propagation in the DDPG-based EMS only involved four times of matrix multiplication mainly.

The cumulative frequency of engine operation efficiency is shown in Figure 4.17. Among the three online methods, MPC-based EMS@0 has the widest range of engine efficiency distribution, and the overall trend of DRL-based EMS is closest to MPC-based EMS@0. This can

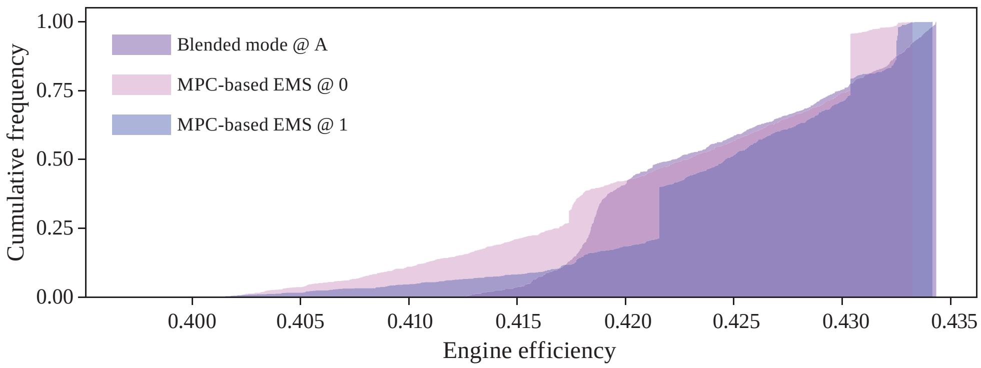


Figure 4.17: Cumulative frequency of engine operation efficiency in Blended Mode.

Table 4.6: Simulation results of CS mode for WVUCITY

Group	Method	SoC_{init}	SoC_{final}	N_{eng}	T_{cal} (s)	$Fuel_c$ (g)	Gap_{fuel_c}
1	DP	0.20	0.2000	18	25.5243	246.2	—
2	CS@O	0.20	0.2002	41	1.2809	252.6	2.60%
3	CS@A	0.20	0.2028	11	1.3191	268.8	9.18%
4	MPC@0	0.20	0.2007	74	248.9467	268.0	8.85%
5	MPC@1	0.20	0.2022	68	248.6890	268.4	9.02%
6	MPC@2	0.20	0.2026	67	249.7623	268.7	9.14%

Notes: N_{eng}: engine start times, T_{cal}: calculation time, Gap_{fuel_c}: the gap of fuel economy with the DP-based EMS.

also explain the fuel economy performances of online EMSs listed in Table 4.5 from the other aspect.

4.3.3 COMPARISON UNDER CS MODE

The standard driving cycle of WVUCITY is used for the generalization test of CS mode, and test results are shown in Table 4.6. The SoC trajectories of the CS@A strategy and the three MPC strategies are relatively close to each other and vary all around the charge sustaining level. The CS@A strategy is still relatively more advantageous in terms of calculation speed and engine start-stop times.

Overall, the DDPG-based EMSs' performance on the testing trip (about 8.5% fuel economy gap from the benchmark) is basically consistent with performance on training trip (about 8.8% gap), which is nearly equal to the MPC-based EMS with prediction error of 1 m/s, and interestingly, its number of engine start decreases by about 34% than the benchmark on testing trip. Further, without output frequency adjustment, there is only about 2.9% fuel economy gap from the benchmark on average, even providing about 4.7% improvement than the MPC-based EMS with accurate prediction. Both aspects demonstrate the good robustness of the DDPG-based EMS on unseen trips. Meanwhile, the time that one simulation step takes in DDPG-based EMSs in the simulation environment, less than 0.001 s on average, is considerably shorter than MPC-based EMSs.

4.4 SUMMARY

(1) Taking a PHEV as an example, this chapter explains the continuous state—continuous action space in vehicle energy management. To address the problem of strategy learning in continuous action space, a parameterized EMS modeling method is introduced, and the corresponding DDPG-based energy management method is illustrated.

(2) Aiming at PHEVs with a larger battery, history trip information is integrated into SoC trajectory planning for effective strategy learning guidance in the DDPG-based EMS. A continuous DRL-based EMS enabled by history trip information is accordingly described, together with the principle of parameter tuning in reward function, off-line training procedures, and the output frequency adjustment trick.

(3) The DDPG-based EMS enabled by history trip information is systematically examined from offline training to online applications. Its learning ability, optimality, and generalization are validated by comparisons with fuel economy benchmark and MPC-based real-time EMSs. Simulation results indicate that without priori knowledge of future trips, original DDPG-based EMSs achieve an average 3.5% gap from the benchmark, superior to the MPC-based EMS with accurate prediction. By applying output frequency adjustment, the engine start frequency is effectively reduced and the average fuel economy gap is about 8.7%, comparable to the MPC-based EMS with the prediction error of 1 m/s. The computational speed of this continuous energy management method is still considerably fast (about 1 ms per a single step).

CHAPTER 5

Learning of EMSs in Discrete-Continuous Hybrid Action Space

For some HEVs, depending on the system configuration, the integration degree of vehicle control strategies, and modeling methods, both discrete and continuous actions can exist in the same action space, making it difficult to describe them monolithically by either discrete action space or continuous action space. Taking a power-split hybrid electric bus (HEB) as an example, this chapter will introduce how to address EMS learning problems in such hybrid action spaces by combing the idea of action value learning and policy gradient update. Furthermore, an energy management method considering terrain information is described, and accordingly, the influence of the multi-source information on learning-based EMSs is discussed in terms of fuel economy, strategy performance under specific driving scenarios, and the strategy decisions.

5.1 ENERGY CONSUMPTION MODEL OF A POWER-SPLIT HEB

The HEB modeling and its working modes are presented in this section, which will facilitate the illustration of the optimal EMS search in hybrid action space. The powertrain of this power split HEB consists of the transmission part and power units. The transmission part mainly consists of the planetary gears (PGs) and the final drive. Power units mainly include the battery pack, the engine and two sets of motor/generator (M/G). The configuration is illustrated in Figure 5.1 with its parameters summarized in Table 5.1. The detailed powertrain modeling is described as follows.

The Transmission Part

The transmission part consists of two sets of planetary gears (PGs), with the ring gear of PG1 as both the carrier of PG2 and the output shaft. The ring gear of PG2 is fixed with the shell of transmission. The connection and disconnection of the engine are determined by clutch 1 (CL1), and those of M/G2 with the output shaft are determined by clutch 2 (CL2). Therefore, depending on the state of clutches, four modes of operation are available, but their responses all obey the kinetic and torque balance characteristics of planetary gears. To reduce the control-

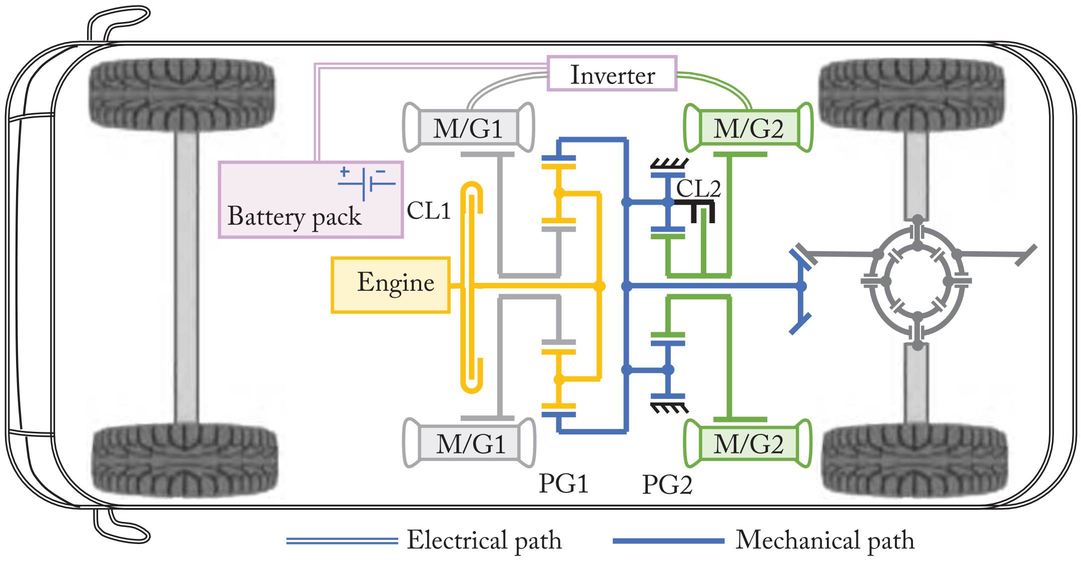


Figure 5.1: Configuration of the power-split HEB.

Table 5.1: Parameters of components in the HEB

Vehicle	Curb weight Final drive ratio	11200 kg 4.88
PG1	Gear ratio of sun and ring gear	2.63
PG2	Gear ratio of sun and ring gear	2.11
Engine	Maximum power/speed Maximum torque/speed	146 kW/2570 rpm 804 Nm/1500 rpm
M/G1	Maximum torque/speed	340 Nm/6000 rpm
M/G2	Maximum torque/speed	830 Nm/6000 rpm
Battery	Capacity Voltage	37 Ah 657 V

oriented model's complexity, the inertial dynamics of transmission part is not considered here. Taking the mode with CL1 engaged and CL2 disengaged as an example, the transmission part can be modeled as follows:

$$\begin{cases} T_{axle} = G_f(T_{r1} + T_{c2}) \\ \omega_{axle} = \omega_{out}/G_f \\ T_{M1} : T_e : T_{r1} = 1 : -(1+k_1) : k_1 \\ T_{M2} : T_{out} : T_{c2} = 1 : -(1+k_2) : k_2 \\ \omega_{M1} + k_1\omega_{out} = (1+k_1)\omega_e \\ \omega_{M2} = (1+k_2)\omega_{out} \end{cases}, \tag{5.1}$$

where T_{axle} denotes the required driving torque; G_f denotes the final drive ratio; T_{r1} and T_{c2} denote the output torque of ring gear of PG1 and carrier of PG2, respectively; T_{M1} and T_{M2} denote the torque of M/G1 and M/G2, respectively; T_e denotes the engine torque; k_1 and k_2 denote the gear ratios of PG1 and PG2; ω_{axle}, ω_{out}, ω_{M1}, ω_{M2}, and ω_e denote the speed of driving axle, output shaft, M/G1, M/G2 and the engine, respectively.

In Equation (5.2), because T_{axle} and ω_{axle} represent the driving demand, which are required by the driver as Equation (5.2) presents, there are two degrees of freedom left to determine the transmission's state. Therefore, we choose T_e and ω_e as the control actions in this mode. The control actions of the engine are bounded by $T_e^{\min} \le T_e \le T_e^{\max}$ and $\omega_e^{\min} \le \omega_e \le \omega_e^{\max}$:

$$T_{axle} = \left(ma_{acc} + mgf\cos\theta_{road} + mg\sin\theta_{road} + \frac{C_D A_{front} v^2}{21.15} \right) R_{roll}, \tag{5.2}$$

where R_{roll} is the rolling radius (0.5 m).

Then, according to Equations (5.1) and (5.2), and the command control signal T_e and ω_e, the state of the transmission part can be determined.

Similarly, according to the given operation mode and control actions (T_e and ω_e), the desired kinetic and torque responses of M/G1 and M/G2 at different modes can be derived from Equations (5.1) and (5.2), which are summarized in Table 5.2 and described as follows.

Mode (A) Hybrid mode with low gear: the engine is engaged with the carrier of PG1 and M/G2 is connected to the sun gear of PG2.

Mode (B) Hybrid mode with high gear: the engine is engaged with the carrier of PG1 and M/G2 is connected to the output shaft.

Mode (C) Electric mode with low gear: the engine is shut down with the carrier of PG1 being braked; M/G2 provides the traction power or harvests braking power.

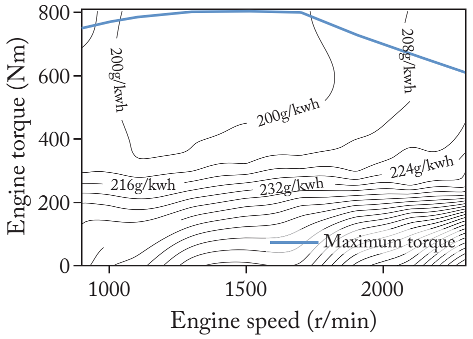
Mode (D) Electric mode with high gear: the engine is shut down with the carrier of PG1 being braked; M/G2 provides the traction power or harvests braking power.

The Power Units

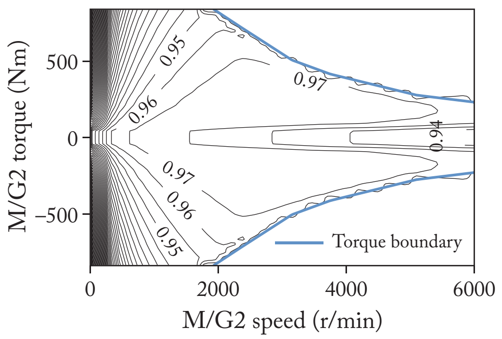
Quasi-static energy consumption models are built for power units. The engine and motors are represented by fuel consumption or efficiency maps derived from bench experiments. The engine fuel consumption rate $\dot{m}_{fuel}$ (kg/s) is represented by a fuel consumption map with respect to

Table 5.2: The responses of M/G1 and M/G2 according to the characteristics of planetary gears

Mode	Mode A	Mode B	Mode C	Mode D
CL1	Engaged	Engaged	Disengaged	Disengaged
CL2	Disengaged	Engaged	Disengaged	Engaged
T_{M1}	$-\frac{1}{1+k_1}T_e$	$-\frac{1}{1+k_1}T_e$	0	0
T_{M2}	$\frac{1}{1+k_2}T_{axle}-\frac{k_1}{(1+k_1)(1+k_2)}T_e$	$T_{axle}-\frac{k_1}{1+k_1}T_e$	$\frac{1}{1+k_2}T_{axle}$	T_{axle}
ω_{M1}	$(1+k_1)\omega_e-k_1\omega_{axle}$	$(1+k_1)\omega_e-k_1\omega_{axle}$	$-k_1\omega_{axle}$	$-k_1\omega_{axle}$
ω_{M2}	$(1+k_2)\omega_{axle}$	ω_{axle}	$(1+k_2)\omega_{axle}$	ω_{axle}



(a) Fuel consumption map



(b) Efficiency map of M/G2

Figure 5.2: Efficiency map of power units.

engine torque and speed, as shown in Figure 5.2a. A similar method is utilized for M/G1 and M/G2. Figure 5.2b shows the efficiency map of M/G2.

The battery will provide or gain the power of M/G1 and M/G2 by discharging or charging, which could be obtained from Equation (5.1). To provide reasonable SoC update, an internal resistance model is used for battery modeling. The dynamics of the battery model is shown as follows:

$$\dot{SoC} = -\frac{U_{oc}(SoC)-\sqrt{U_{oc}(SoC)^2-4(T_{M1}\omega_{M1}\eta^i_{M1}+T_{M2}\omega_{M2}\eta^i_{M2})R(SoC)}}{2R(SoC)Q_{batt}}, \tag{5.3}$$

where $U_{oc}(SoC)$ denotes the open-circuit voltage of battery, $R(SoC)$ denotes the internal resistance, Q_{batt} denotes the battery capacity, η_{M1} and η_{M2} denotes the efficiency of M/G1 and

M/G2 which are obtained from their efficiency maps, $i = 1$ when corresponding M/G works as a generator, and $i = -1$ when M/G works as a motor.

5.2 SEARCHING OF OPTIMAL EMSs IN HYBRID ACTION SPACE BY DDPG

5.2.1 HEB ENERGY MANAGEMENT AND ITS HYBRID ACTION SPACE

With the energy consumption model described in Section 5.1, to address the energy management problem of this HEB, the controller needs to choose a mode, and corresponding control signals of the engine torque and speed (at hybrid modes), simultaneously. The selection of mode is a discrete action and the control actions of the engine are continuous actions. Accordingly, there are four mutually exclusive discrete actions {ModeA, ModeB, ModeC, ModeD} (mode selection) and two continuous actions $\{T_e,\ \omega_e\}$ in the action space, namely a hybrid action space as shown in Equation (5.4):

$$\begin{cases} A = \{\text{ModeA},\ T_e,\ \omega_e\} \cup \{\text{ModeB},\ T_e,\ \omega_e\} \cup \{\text{ModeC}, \\ \qquad\qquad T_e = 0,\ \omega_e = 0\} \cup \{\text{ModeD},\ T_e = 0,\ \omega_e = 0\} \\ a = [Mode(t),\ T_e(t),\ \omega_e(t)],\quad a \in A. \end{cases} \tag{5.4}$$

HEB energy management state variables should provide as much environmental information as possible while being easily observable. According to vehicle longitudinal dynamics and HEB powertrain model, we choose current velocity v and SoC as state variables for vehicle state; the state of CL1 (*clutch*) is considered to reflect the engine state; terrain information, i.e., road slope (θ_{road}), is used as a state variable for road conditions; driving requirements is represented by state variable of vehicle acceleration (*acc*) that is obtained from the analytical pedal signal. To make full utilization of our knowledge about environment dynamics, we additionally add required driving torque T_{axle}, required driving power P_{axle} and history velocity during the past three seconds ($v_{-1},\ v_{-2},\ v_{-3}$) into state space, among which T_{axle} and P_{axle} are calculated based on vehicle acceleration and terrain information. As a result, the state space is summarized as the following equation:

$$\begin{cases} S = \{v, clutch, SoC, \theta_{road}, acc, T_{axle}, P_{axle}, v_{-1}, v_{-2}, v_{-3}\} \\ s(t) = [v(t), clutch(t), SoC(t), \theta_{road}(t), acc(t), T_{axle}(t), \\ \qquad\qquad P_{axle}(t), v_{-1}(t), v_{-2}(t), v_{-3}(t)],\quad s \in S. \end{cases} \tag{5.5}$$

The goal of HEB energy management is to minimize fuel consumption by continuously commanding a power allocation scheme based on the constantly changing vehicle status, road conditions, and driving requirements. Besides, the driving demand should always be satisfied, and the SoC should maintain near the CS level without extreme drops/increases. Hence, the reward is defined as follows:

$$r = r(s,a) = -\left[\chi \dot{m}_{fuel} + \varphi(SoC(t) - SoC_{sust})^2\right], \tag{5.6}$$

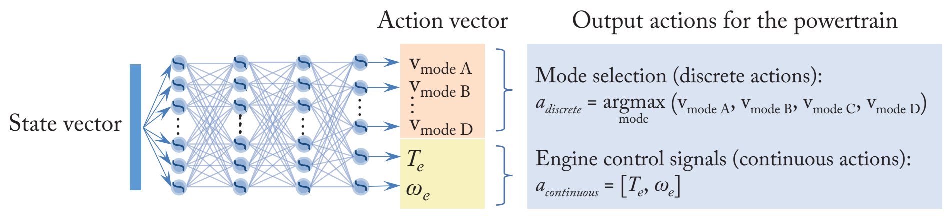

Action vector
Output actions for the powertrain
State vector
$v_{\text{mode A}}$
$v_{\text{mode B}}$
$v_{\text{mode D}}$
T_e
ω_e
Mode selection (discrete actions):
$a_{discrete} = \underset{\text{mode}}{\text{argmax}}\,(v_{\text{mode A}}, v_{\text{mode B}}, v_{\text{mode C}}, v_{\text{mode D}})$
Engine control signals (continuous actions):
$a_{continuous} = [T_e, \omega_e]$

Figure 5.3: The architecture of Actor network for hybrid action spaces.

where $SoC_{sust} = 0.6$. To avoid extreme rewards, the reward signal needs to be clipped [57], and both the clipping method and the method of using a bounded reward function are adequate for strategy learning. The tuning principle of the weighting factors in Section 4.2.2 also applies here. A penalty ρ will be implemented to reward every time starting the engine: $r = r - \rho$.

5.2.2 LEARNING OF EMSs IN HYBRID ACTION SPACE

In deep Q-learning, the output of the DQN is the action-value of each discrete action, so that the greedy strategy can be used to select the action with the largest action value, realizing optimal strategy search in discrete action spaces. While in the DDPG method, the strategy network directly outputs the value of each continuous action and improves the strategy under the guidance of the Critic network, realizing the strategy seeking in continuous action spaces. The comparison shows that the key to strategy learning in the discrete action space is action-value learning, while the key to that in the continuous action space is finding the policy gradient. Therefore, how to construct a parameterized policy model that can simultaneously implement action-value learning for discrete actions and policy gradient updates for continuous actions becomes the key to solve the policy optimization in hybrid action spaces. In this chapter, based on Matthew Hausknecht et al.'s research concerning DRL in parameterized action space [112], an optimal EMS learning method developed for hybrid action space is introduced as follows by combing the idea of DQN and DDPG.

The Modeling of Parameterized EMSs with a Hybrid Action Space

A DNN is used for the estimation of the EMS here, taking continuous state variables as inputs and outputting deterministic actions, as shown in Figure 5.3. In this parameterized strategy network μ, the state vector is processed by three fully connected layers, within each of which there are 100 ReLU activation functions; the output layer is processed by six hyperbolic tangent activation functions to generate the values of each mode and bounded actions. The bounded actions will be subsequently re-scaled into their intended ranges.

In contrast to the Actor network for continuous EMS, the output of the Actor network for hybrid action spaces is divided into two parts. (1) For working mode selections, the strategy network outputs continuous values to represent the action values of each mode, as the mode selection part in Figure 5.3 shows. (2) Continuous control actions are directly represented by two continuous outputs, as the engine control part in Figure 5.3 shows. Thus, the EMS π can be expressed by forward propagation of the strategy network μ:

$$\pi = a = [\mathrm{v}_{\mathrm{modeA}},\ \mathrm{v}_{\mathrm{modeB}},\ \mathrm{v}_{\mathrm{modeC}},\ \mathrm{v}_{\mathrm{modeD}},\ T_e,\ \omega_e] = \mu(s|\theta^{\mu}), \tag{5.7}$$

where θ^{μ} denotes all parameters of μ, and $[\mathrm{v}_{\mathrm{modeA}},\ \mathrm{v}_{\mathrm{modeB}}, \mathrm{v}_{\mathrm{modeC}}, \mathrm{v}_{\mathrm{modeD}}]$ denotes the value of each operation mode.

With the actor vector, control signals of the engine can be directly obtained, while the corresponding mode, $Mode(t)$, is obtained by choosing the maximally valued mode as below. Thus, it enables the simultaneous output of discrete and continuous actions:

$$Mode(t) = \arg\max_{\mathrm{mode}}(\mathrm{v}_{\mathrm{modeA}},\ \mathrm{v}_{\mathrm{modeB}},\ \mathrm{v}_{\mathrm{modeC}},\ \mathrm{v}_{\mathrm{modeD}}). \tag{5.8}$$

The Learning Process

The learning theory of EMSs in hybrid action space is basically the same as that stated in Section 4.1. With the parameterized strategy network μ, the strategy learning also involves the following key procedures.

(1) The construction of the Critic network Q parameterized by θ^{Q}. $Q(s, a|\theta^{Q}) = V(s|\theta^{V}) + A(\mu(s|\theta^{\mu})|\theta^{A})$, where θ^{V} and θ^{A} denotes the parameters of state stream and action stream, respectively, as shown in Figure 4.1. Replicas of AC networks are built: μ_T and Q_T.

(2) Establishment of an experience pool $\mathcal{D}$ for storage of experiences and experience replay.

(3) Calculate the loss of Critic network (Equation (4.8)) and update it to minimize the loss function.

(4) Calculate the policy gradient (Equation (4.10)) to update the Actor network μ.

(5) Update the target AC networks (Equation (4.7)).

During the strategy learning, the Critic network takes all outputs of the Actor as inputs, and there is no need to mark the selected working mode. For policy gradient computation, the Critic network will provide gradient with respect to both the four discrete working modes and the two continuous actions, rather than providing gradient only for continuous actions. Besides, the ε-greedy exploration is also combined. The exploration mechanism needs to randomly choose the working mode first, and then, if the engine is controllable, it subsequently needs to generate random engine speed and torque in the feasible domain.

5.3 EMSs CONSIDERING TERRAIN INFORMATION UNDER HYBRID ACTION SPACE

Usually, it is difficult to obtain an optimal EMS without any priori information about the future driving cycle, such as the velocity profiles and the terrain information which are highly correlated

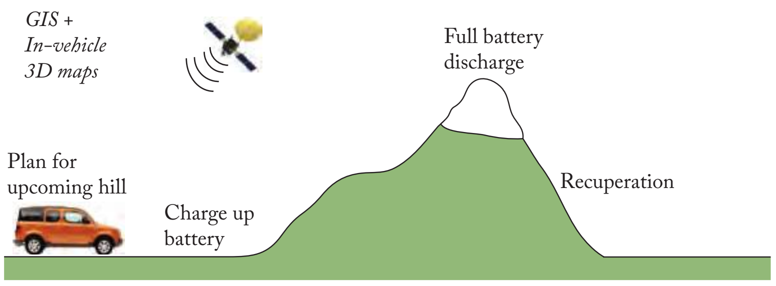


Figure 5.4: The influence of terrain on HEV energy management [113].

to the driving style, traffic conditions, weather, etc. Since the effect of velocity is quite critical and intuitive under good driving conditions, research concerning long-term driving cycle reconstruction and predictive energy management has gained much momentum in the last decade. However, for commercial vehicles with high mileage and heavy load capability, the variable road terrain will have a more noticeable impact on their energy consumption [9]. Also, timely access to current or future terrain information will also facilitate wise energy management, leading to more informed coordination of power distribution. For example, when a 3D road map or GPS system anticipates that the vehicle is about to reach a long uphill segment, energy management can recharge the battery in time to improve energy efficiency by reasonably coordinating charging, power consumption, and regenerative braking, as illustrated in Figure 5.4.

Since a major advantage of DRL-based EMSs is their enhanced ability to handle environmental information without considering the curse of dimensionality, an energy management method incorporating terrain information is introduced in this section for this power-split HEB, combined with the learning-based EMS for hybrid action spaces. Meanwhile, in order to make full use of experts' empirical knowledge, this section also introduces a strategy pre-training stage into the learning-based EMS to improve its training effect. The learning-based EMS considering terrain information will finally be used to validate the effectiveness of strategy learning in hybrid action spaces on the one hand, and to provide a demonstration for analyzing the impact of multi-source information, especially terrain information, in the learning-based EMS on the other hand.

The application topology of the learning-based EMS considering terrain information is shown in Figure 5.5, where the EMS for this HEB is represented by a DNN. The strategy will be saved and downloaded into the vehicle controller after being trained by DDPG. In Figure 5.5, the observed terrain information, together with current vehicle state, are fed into the DNN as input state vector s(t), then DNN will output corresponding power split scheme, i.e., the action vector a(t), to HEB's powertrain. As the response of powertrain, this HEB will give a reward signal r(t) and transfer to a new state at moment t+1. Here, the reward will only be collected during strategy training period. We will explain this method from three aspects, including the

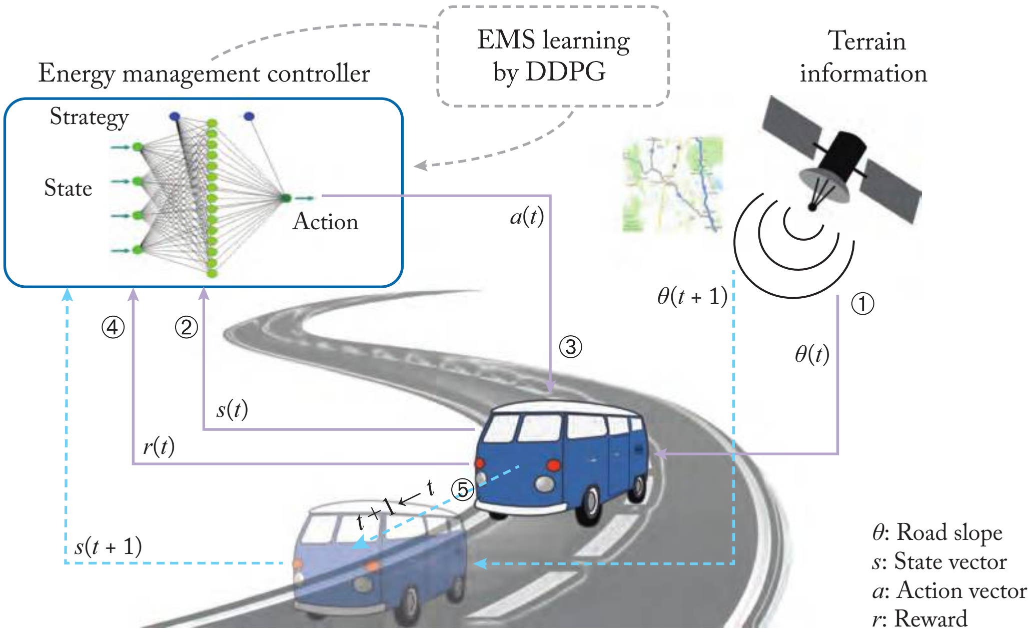


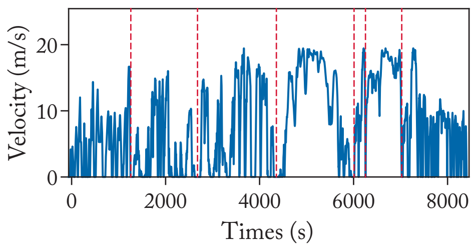
Figure 5.5: A schematic overview of learning-based energy management considering terrain information.

training/testing data preparation, the pre-training method, and the training of EMS considering terrain information.

5.3.1 GENERATION OF VELOCITY-SLOPE PROFILES

According to the design requirements of this HEB, it should climb hills with road slope of at least 12% at the velocity of 12 km/h; the maximum velocity is 70 km/h. Two velocity profiles are constructed by ten standard driving cycles for training and test of the proposed strategy, as shown in Figure 5.6a. The training velocity file is sequentially constituted by CTUDC, WVUCITY, WVUSUB, WVUINTER, IM240, HWFET, and UDDS, with a total length of 64.315 km. Due to the limitation of maximum velocity, those with maximum velocity over 70 km/h are regulated into a range between 0 and 70 km/h. Similarly, the test velocity profile, Figure 5.6b, is sequentially constituted by three repetitions of WLTP Class 1 (low velocity), WLTP Class 1 (medium velocity), and JN1015, with a total length of 63.523 km.

Ideally, the data should be independent and identically distributed for machine learning applications, but in practice this is often not the case. Therefore, researchers often choose to extend the training set coverage or construct the dataset manually to assist relevant studies. For EMSs considering terrain information, in order to construct the training profiles, we should


(a) The velocity profile for training

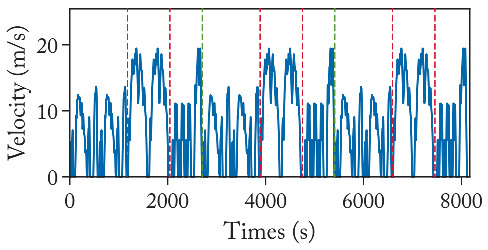

(b) The velocity profile for test

Figure 5.6: The velocity profiles.

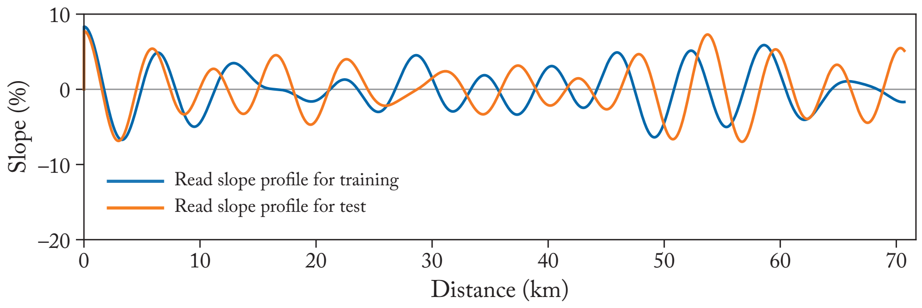


Figure 5.7: Road slope profiles for training and test.

Table 5.3: Statistical comparison between training and test road slope-velocity profiles

Velocity Profiles	**Maximum**	**Mean Value**	**Std. Dev.**	$Time_{acc}/Time_{dec}$
Training	19.44 m/s	7.65 m/s	6.49 m/s	1.22
Test	19.44 m/s	7.82 m/s	6.11 m/s	1.04
Slope Profiles	**Maximum**	**Minimum**	**Mean Value**	**Std. Dev.**
Training	9.6%	–7.7%	0.5%	3.5%
Test	8.9%	–8.0%	0.1%	3.7%

collect the terrain data related to specific types of vehicles and their typical driving scenarios. Due to limited available data resources, however, we adopt a simulation method to customize road profiles for this HEB [114]. First, two sets of simulated arc terrains are applied for training and test velocity profiles, then road slope profiles could be derived accordingly, as shown in Figure 5.7. The training and test road slope-velocity profiles are compared in Table 5.3, showing that they are statistically different and sufficient for strategy evaluation.

5.3.2 PRE-TRAINING METHOD

Xavier initialization method is also adopted for AC networks in this chapter. As there are multiple working modes, and the engine speed and torque are two independent continuous control actions, the action space becomes larger. This makes it necessary to accumulate more samples during stochastic exploration to promote strategy learning. Considering that the vehicle model and the training profile are known before training, we are able to obtain the global optimal EMS on the training profile. Therefore, it is potentially worthwhile to combine this prior knowledge of the global optimal strategy with the learning ability of DRL-based EMSs, so that the EMS can start learning on the basis of the empirical experience rather than from scratch. To implement such idea, this section introduces a pre-training stage in DRL-based energy management methods.

In practice, we found that pre-training the Critic network with optimal experience samples, i.e., the state transition samples optimized by DP, is helpful for convergent strategy learning. Detailed pre-training methods are described as below.

(1) The CTUDC related fraction of the training profile is picked for data generation. DP is applied on this fraction to derive the optimal state transition experiences. The mode selection part of the action vector is filled by one-hot encoding. The entire training profile is not utilized for data generation, which is because the training dataset may be quite large or being constantly updated. Selecting a short profile for pre-training would be more realistic.

(2) Optimal samples are replicated to fill in the replay memory $\mathcal{D}$.

(3) Sample random minibatches of experiences from $\mathcal{D}$ and calculate the loss of Critic network.

(4) Update the Critic network and the target Critic network by batch gradient descent, with the parameters of the two Actor networks frozen.

(5) Repeat from step 3 until the Critic gets converged, for example, the moving average of the loss function gets smaller than 0.01.

5.3.3 TRAINING ALGORITHM

Algorithm 5.4 shows the pseudocode of the DRL-based EMS with terrain information. Combined with Section 5.2, the overall training process is outlined here in five function parts, which are not isolated but interactive with each other, as shown in Figure 5.8.

(1) Initialization. This includes the initialization of AC networks and the normalization of state data. Velocity, acceleration, road slope, required power and torque are normalized by Z-score normalization method. The respective mean value and standard deviation should be saved for normalization during testing. State variable of SoC and the clutch will be directly fed into the networks.

(2) Pre-training. The Critic networks are pre-trained after the initialization stage.

(3) Experience replay, which is similar to that adopted in Section 4.2. Considering the long training profile, the size of experience pool is set as 10^6. After the pre-training stage, the

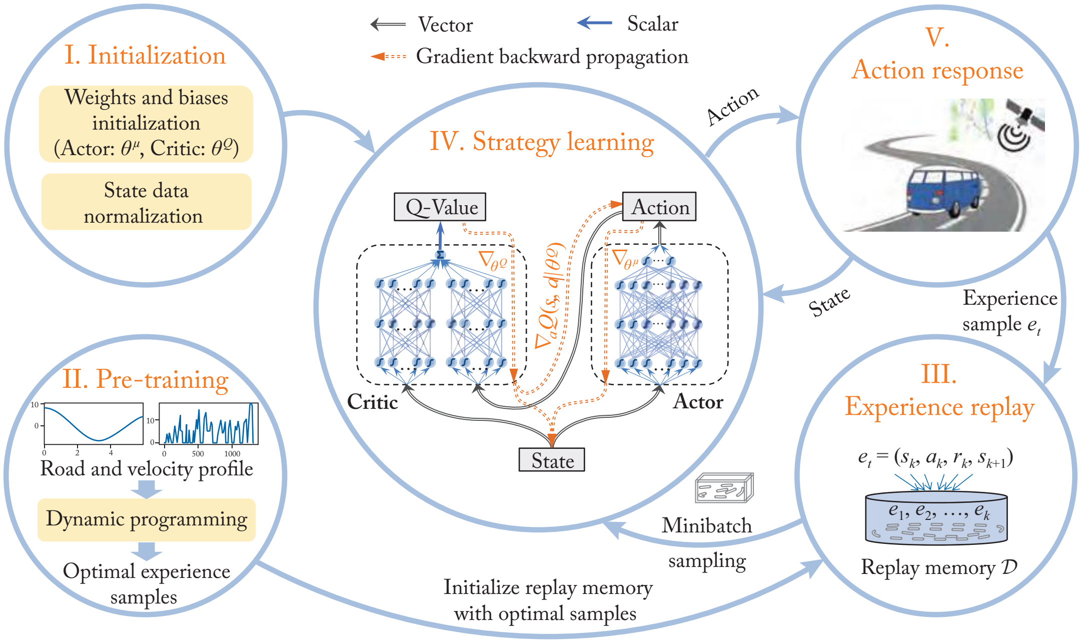


Figure 5.8: The training of a DRL-based EMS with terrain information.

experience pool has been full already. Thus, when a new sample comes during the next strategy learning stage, the oldest sample in $\mathcal{D}$ will be discarded and replaced by the new one.

(4) Strategy learning. It mainly involves loss function calculation of the Critic network, policy gradient calculation, network update, exploration, etc. In this chapter, we anneal ε from 1 to 0.1 over the first 45 update rounds.

(5) Action response. This function part is about the simulation of HEB environment dynamics and always interacts with strategy learning and experience replay.

5.4 COMPARATIVE CASE ANALYSIS

5.4.1 DEVELOPMENT OF CONTRAST STRATEGIES

The AC networks are trained on the training profile and the initial SoC value at the beginning of each update round is randomly selected from $[0.5, 0.7]$ to enhance generalization. Then, to examine the learning and optimization ability of strategy learning in the hybrid action space, four contrast EMSs are described briefly as follows.

Algorithm 5.4 The training of a DRL-based EMS with terrain information.

Initialize AC networks by Xavier uniform initializer; initialize target AC networks; initialize experience pool $\mathcal{D}$; initialize exploration rate $\varepsilon = 1$; maximum iteration number $N_{iteration}$; time length of the training driving cycle T_{train}; size of minibatch $Minibatch = 32$.

```
1:  Generate optimal experience samples by DP, initialize replay memory $\mathcal{D}$, and pre-train the
    Critic network
2:  **for** $IterationRound = 1 : N_{iteration}$ **do**
3:      Observe initial state $s(0)$
4:      **for** $t = 0 : (T_{train} - 1)$ **do**
5:          Action selection in hybrid action space: select action by random generation $a(t) = \mathcal{R}(s(t))$ (with probability $\varepsilon$), otherwise use the current policy $a(t) = \mu(s|\theta^{\mu})$ (with probability $1 - \varepsilon$) and obtain the engine control signals and the working mode $Mode(t) = \arg\max_{\text{mode}}(\text{v}_{\text{modeA}}, \text{v}_{\text{modeB}}, \text{v}_{\text{modeC}}, \text{v}_{\text{modeD}})$
6:          Execute action $a(t)$ by calling the HEB model
7:          Observe reward $r(s(t), a(t))$ and the next state $s(t+1)$
8:          Discard the oldest sample in $\mathcal{D}$, store the new experience sample $e(t) = (s(t), a(t), r(s(t), a(t)), s(t+1))$;
9:          Sample a random minibatch of experience samples from $\mathcal{D}$
10:         Update Critic network to minimize TD error: $\theta^{Q} = \theta^{Q} + \alpha_l \nabla_{\theta^{Q}} L_Q$
11:         Update Actor network to increase the estimated Q-value: $\theta^{\mu} = \theta^{\mu} + \alpha_l \nabla_{\theta^{\mu}} \mu$
12:         Update the target AC networks by $\theta_T^{Q} = \theta_T^{Q} + \tau(\theta^{Q} - \theta_T^{Q}),\ \theta_T^{\mu} = \theta_T^{\mu} + \tau(\theta^{\mu} - \theta_T^{\mu})$

13:         Decrease the exploration possibility
14:     **end for**
15: **end for**
16: Output the final parameterized Actor network $\mu$ as the trained $\pi = \mu(s|\theta^{\mu})$
```

The Baseline: DP-Based EMSs

As described in Section 3.4.1, DP-based EMSs will serve as the fuel economy benchmark. The action space contains four discrete actions for mode selection and two engine control actions which are discretized into 100 grids, respectively.

The Rule-Based EMS

With limited information, the rule-based EMS can respond quickly and is commonly used in HEBs. So, the required driving torque, brake requirement, and SoC are considered to develop a rule-based EMS as an online contrast strategy (the electric mode and hybrid mode with low gears are considered here).

Mode (A) When the SoC of battery is enough for driving requirement (10% more than sustaining level): if M/G2 could satisfy the required driving torque, pure electric mode is the priority, i.e., the clutch is disengaged and M/G2 works as a traction motor; if M/G2 could not satisfy the required driving torque, the engine will work in its optimal area as much as possible with clutch engaged, and the torque gap with required driving torque is compensated by M/G2. This mode will be maintained until the condition of mode B is met.

Mode (B) When the SoC of battery is 10% below the sustaining level: with the required driving torque satisfied, the engine operating point is shifted approaching to upper boundaries of optimal areas by delivering part of its energy to the battery via M/G1. This mode will maintain until the condition of mode A is met.

Mode (C) During brake, the battery pack will harvest energy by regenerative braking when $SoC < SoC_{\max}$. Here, the maximum value ($SoC_{\max}$) is set as 90%.

The Continuous DRL-Based EMS

The continuous DRL-based EMS (CDRL-based EMS) is derived utilizing the same DDPG method and trained on the same training profile as elaborated in the last section. However, to examine the influence of mode selection on learning-based EMSs, its action space only consists of two continuous engine control actions. The mode selection is conducted by rules: only when braking, the electric mode with low gear is chosen, otherwise, the hybrid mode with low gear is chosen; the CL1 is only engaged when the engine is on.

The Elementary DRL-Based EMS Without Terrain Information

Similar to the CDRL-based method, an elementary DRL-based EMS (EDRL-based EMS) is derived but without the inclusion of terrain information in its state space. Still, only two continuous actions for the engine are considered here, with the mode selection conducted by rules. In addition, due to the absence of terrain information, road slope is also not considered when calculating T_{axle} and P_{axle} for the state vector.

5.4.2 LEARNING PROCESS IN HYBRID ACTION SPACE

The learning process of the DRL-based EMS with terrain information and CDRL-based EMS are compared in Figure 5.9. Generally, their mean reward keeps rising before 35 update rounds, then the variation trend gets steady until termination; correspondingly, the final SoC after each update round over one training profile gets stable gradually, even with different initial SoC values. The final Actor network is saved as the trained EMS.

From Figure 5.9, the mean reward of the DRL-based method is much lower than the CDRL-based one in the beginning period. This is because of the introduction of multi-mode selection into the action space. At the beginning period, as all modes are selected randomly, there

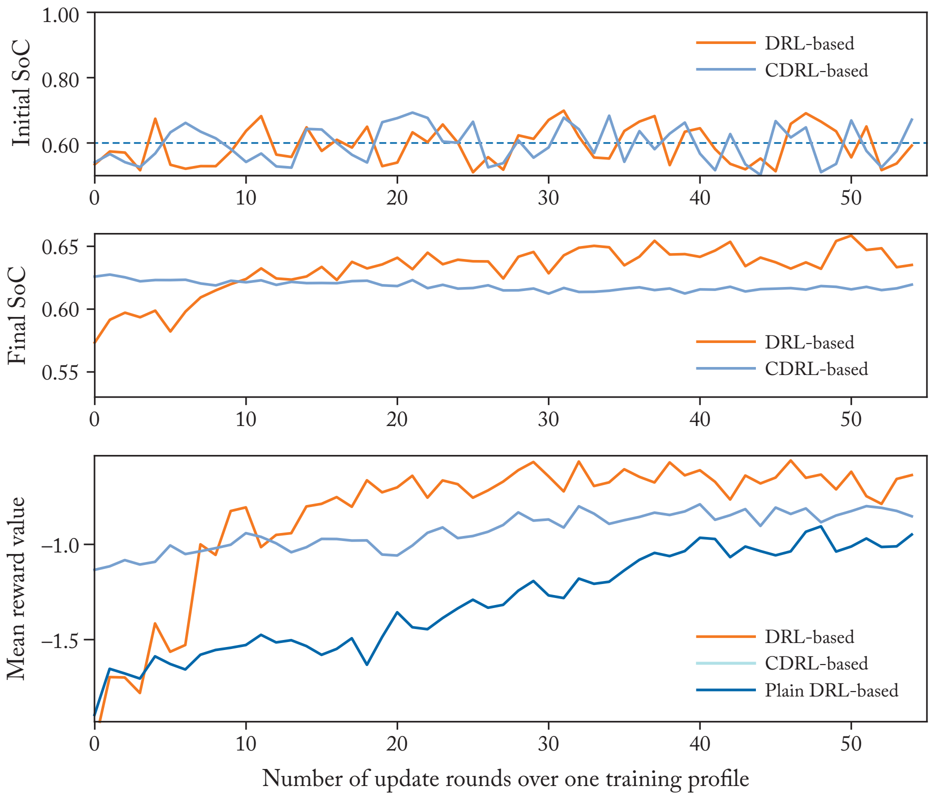


Figure 5.9: Visualization of the strategy learning process.

are more chances of operating at electric modes compared to the CDRL-based EMS, leading to more electricity consumption and lower final SoC level. Accordingly, less reward is received. But the agent rapidly learns to how to select mode efficiently, bringing even higher reward return than CDRL-based EMS later.

On the other hand, due to the random factors during training, such as the exploration and the random sampling, the convergence of final SoC values may vary and drift slightly from the desired sustaining level. The agent learns to achieve higher reward returns over the entire training profile by sacrificing the consistency in the final SoC state. It can be explained by the definition of reward function: without any future driving information, the instant reward is only determined by current fuel consumption and SoC variation value, and there is no constraint on final SoC state.

In Figure 5.9, the plain DRL-based EMS refers to a DRL-based strategy trained without pre-training and two-stream architecture in the Critic network. To make the figure concise, only its mean reward is depicted. The plain DRL-based EMS learns slower and gets relatively

Table 5.4: Results comparison for optimization validation

Group	Method	SoC_{init}	SoC_{final}	N_{eng}	T_{cal} (s)	$Fuel_c$ (g)	Gap_{fuel_c}
1	DP	0.60	0.6000	174	118.9154	5399.7	—
	DRL	0.60	0.6184	39	7.5882	5722.3	5.97%
	CDRL	0.60	0.6237	49	6.9827	5839.5	8.14%
	EDRL	0.60	0.6408	55	6.9064	5927.0	9.77%
	Rule	0.60	0.7286	23	4.1108	6359.6	17.78%
2	DP	0.65	0.6000	165	119.0049	5174.2	—
	DRL	0.65	0.6184	39	7.5771	5478.8	5.89%
	CDRL	0.65	0.6237	47	7.0151	5592.9	8.09%
	EDRL	0.65	0.6409	54	7.0268	5681.7	9.81%
	Rule	0.65	0.7288	22	4.2416	6115.4	18.19%
3	DP	0.55	0.6000	179	118.5071	5627.6	—
	DRL	0.55	0.6186	40	7.5601	5985.7	6.36%
	CDRL	0.55	0.6237	50	7.0053	6085.5	8.14%
	EDRL	0.55	0.6408	58	7.9951	6176.9	9.76%
	Rule	0.55	0.7286	23	4.1603	6596.2	17.21%

less reward, indicating the combination of pre-training and the two-stream Critic network are beneficial for efficient strategy learning.

For this training profile of 64.315 km, it takes approximately one hour for the DRL-based EMS to complete 55 update rounds on a computer with a processor of i7-2600CPU @ 3.4GHz. Compared with previous research about discrete reinforcement learning-based EMSs, even with larger state and hybrid action spaces, this method still owns a distinct advantage in training and convergence rate.

5.4.3 OPTIMIZATION EVALUATION CONSIDERING TERRAIN INFORMATION

In this section, we will examine the optimization performance of the derived DRL-based EMS on the training profile. The DP-based EMS and rule-based EMS are implemented to serve as benchmarking strategy and online strategy, respectively. The results are summarized in Table 5.4, where there are three groups of simulation with different initial SoC.

From Table 5.4, the DRL-based EMS could narrow the gap with the DP-based EMS apparently with less engine start times, demonstrating that the issue of avoiding frequent engine starts has been properly addressed during the learning of strategy by implementing corresponding penalty. Besides, compared with the rule-based EMS, the DRL-based EMS also displays

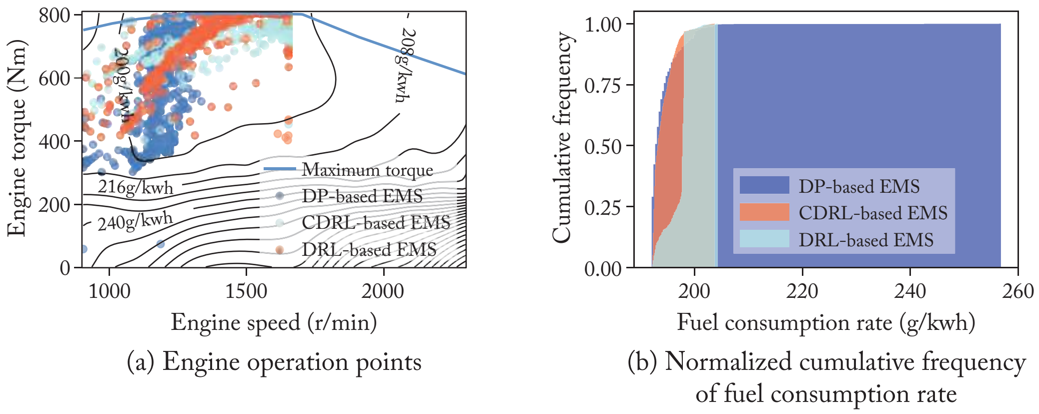


Figure 5.10: Comparison of engine operation points.

quite impressive computation speed (about 0.9 ms per simulation step) in the simulation environment.

Focusing on Group No.1, the optimization ability is further discussed from the following three aspects.

The Performance in Engine Operating Efficiency

The engine operation points of the DP-based, CDRL-based, and DRL-based EMSs with terrain information are depicted in Figure 5.10, together with their efficiency distribution. Their average fuel consumption rates are 194.10 g/kWh, 197.07 g/kWh, and 194.04 g/kWh, respectively. However, the engine operation points of both DRL-based and CDRL-based EMSs are more likely to appear around the upper torque boundary. Regarding this phenomenon, the possible reason is that the hyperbolic tangent activation function is chosen for the output layer of Actor network to output bounded actions, but the update will be slowed down severely when the hyperbolic tangent function gets saturated. But overall, we can still see that the learning is efficient with the presence of terrain information to make the engine operate at more efficient areas.

On the other hand, compared with the CDRL-based EMS, the proposed DRL-based EMS additionally takes the powertrain mode selection into consideration during strategy learning, its final strategy exhibiting a more similar distribution trend to the baseline in terms of fuel efficiency from Figure 5.10b. This also verifies the effectiveness of this energy management method in hybrid action space with multiple discrete actions.

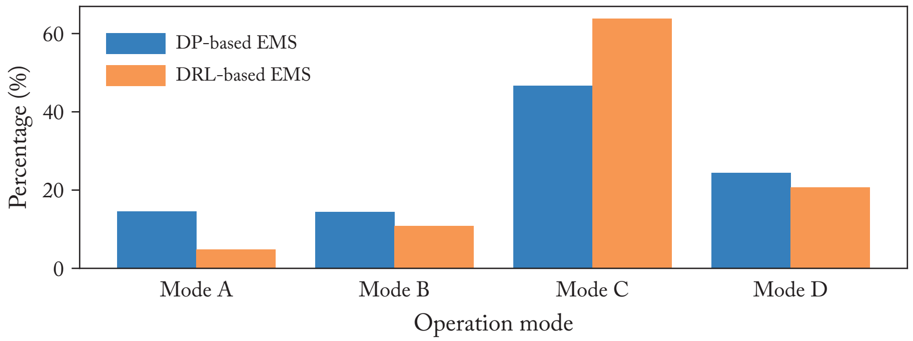


Figure 5.11: The percentage of operation time under different modes.

Table 5.5: Statistical characteristics of powertrain mode selection

Name	Method	Mode A	Mode B	Mode C	Mode D
Average v (m/s)	DP DRL	9.0 9.4	14.1 16.0	4.1 4.1	9.8 13.9
Average a_{acc} (m/s^2)	DP DRL	0.26 0.51	0.08 0.08	−0.06 −0.05	−0.08 −0.02
Average θ_{road} (%)	DP DRL	2.0 3.0	1.6 2.4	0.4 0.1	−2.1 −1.3
Average T_{axle} (Nm)	DP DRL	619.0 1018.6	408.9 504.4	33.9 33.2	−191.6 −29.9

The Performance in the Mode Selection

Although the engine operation is decoupled with driving conditions under hybrid modes, the physical output torque and speed boundaries of power units may still affect feasible control solutions for the engine under different hybrid modes. Therefore, the DRL-based and baseline strategies' performance in mode selection is further compared and discussed. The statistical results of powertrain operation modes are depicted in Figure 5.11. Generally, to harvest more electricity power over the undulating terrain, both strategies are more inclined to choose Mode C (the electric mode with low gear) and the total time operating at hybrid modes is less. Besides, due to the larger output power of the engine under DRL-based strategy from Figure 5.11, its hybrid modes (Mode A and B) take a less portion compared with the baseline.

On the other hand, the average velocity, average acceleration, average slope, and average required driving torque T_{axle} of the two strategies under different modes are summarized in Table 5.5. It shows that both strategies share similar statistical characteristics: Mode A (hybrid mode with low gear) is often adopted to meet driving conditions with larger acceleration and

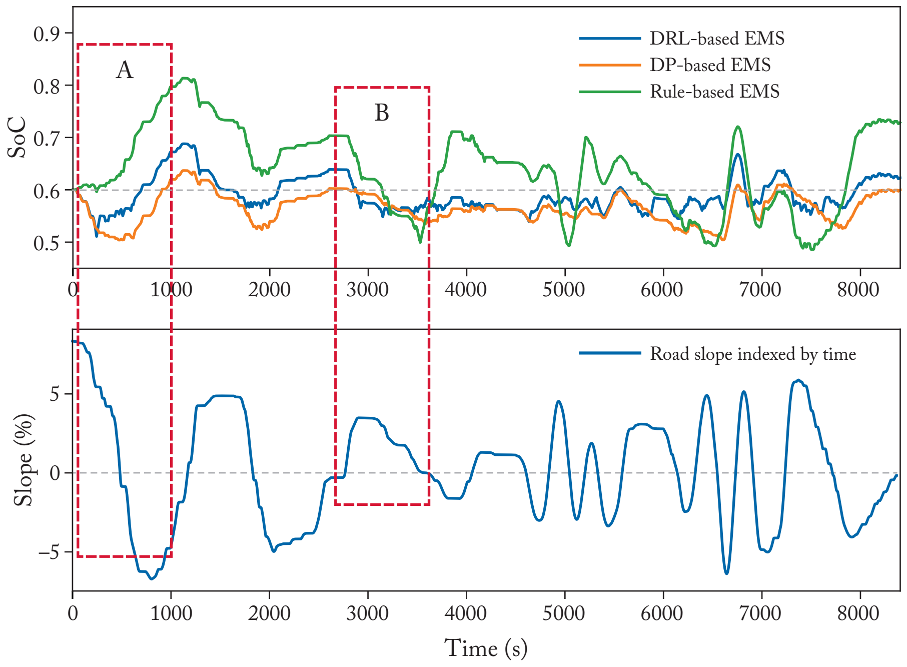


Figure 5.12: SoC trajectories of three strategies on one training profile.

required driving torque, while the circumstance for Mode B (hybrid mode with high gear) is just the opposite; the average statistical values of Mode C and D (electric modes) are less compared with hybrid modes, which is due to the mutual compensation of electric drive and brake recovery. From the above, in comparison with the baseline strategy, the DRL-based EMS indeed learns some rules of efficient mode switch in discrete parts of the hybrid action space.

The Performance at Different Terrains

Here, specific scenarios are chosen to examine why including terrain information could promote fuel economy. The SoC trajectories of the DRL-based EMS and contrast strategies are shown in Figure 5.12. As the dashed area A shows, when SoC is lower than the sustaining level, the rule-based EMS decides to charge the battery immediately and as a result, it does not take the advantage of following downhill terrains; while for the DRL-based EMS, it performs more similar with the DP-based strategy, driving using more electricity at first for later energy harvesting at downhill terrains. At uphill terrains, the dashed area B shows that the DRL-based EMS could reasonably coordinate power split schemes between the engine and the battery to avoid the dramatical drop of SoC. Overall, even with undulating terrains, the fluctuation of the DRL-based EMS's SoC trajectory is much smaller, which is helpful for the maintenance of battery health.

Table 5.6: Results comparison for generalization validation

Group	Method	SoC_{init}	SoC_{final}	N_{eng}	T_{cal} (s)	$Fuel_c$ (g)	Gap_{fuel_c}
1	DP	0.60	0.6000	174	118.2133	5525.3	—
	DRL	0.60	0.5898	40	7.5867	5894.8	6.69%
	CDRL	0.60	0.5770	43	7.1660	5954.6	7.77%
	EDRL	0.60	0.6010	58	7.1731	6005.6	8.69%
	Rule	0.60	0.6891	21	4.0442	6400.2	15.83%
2	DP	0.60	0.6000	80	115.9783	4835.2	—
	DRL	0.60	0.6260	23	7.5462	5162.5	6.77%
	CDRL	0.60	0.6278	19	7.1858	5169.8	6.92%
	EDRL	0.60	0.6416	39	7.0264	5379.4	11.25%
	Rule	0.60	0.7417	25	3.8904	5538.4	14.54%
3	DP	0.60	0.6000	135	116.0276	4881.0	—
	DRL	0.60	0.6104	29	7.5452	5202.2	6.58%
	CDRL	0.60	0.6091	30	7.1972	5326.5	9.13%
	EDRL	0.60	0.6152	48	7.0284	5403.1	10.70%
	Rule	0.60	0.7167	23	3.9373	5714.1	17.07%

In summary, the DRL-based EMS with terrain information shows its ability to promote fuel economy by optimizing multiple mode selections and energy allocations simultaneously in a hybrid action space.

5.4.4 GENERALIZATION EVALUATION

For learning-based controls, it is essential to make sure they are robust in diverse environments. Thus, the trained strategy is further examined on test profiles, which were partly or entirely unseen during strategy training. The experiment results are summarized in Table 5.6 (CDRL-based and EDRL-based EMSs are discussed in the next section). In a generalization test, the average fuel economy gap of the DRL-based EMS with benchmarking strategy is about 6.68%, which is basically consistent with its performance on optimization examinations (6.07%). Still, the advantage of the proposed method in both computation rate and avoiding frequent engine starts excels while maintaining an acceptable level of fuel economy.

5.4.5 INFLUENCE OF TERRAIN INFORMATION

As many factors matter to power split scheme decisions at different states, it could be incomplete or even unbalanced to evaluate the influence of terrain information exclusively by certain scenarios. Thus, its influence on this learning-based method is further investigated from macroscopic perspectives.

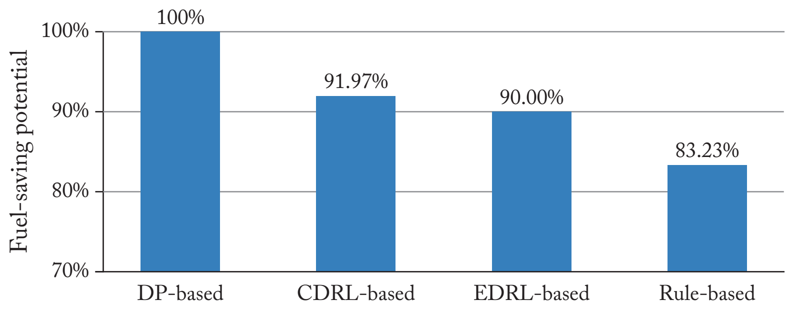


Figure 5.13: Comparison of fuel-saving potential compared with the baseline.

The Influence on Fuel Economy

In this section, to reduce the influence from mode selections, only CDRL-based, EDRL-based, and rule-based EMSs are involved, and the DP-based EMS still serves as the baseline strategy. Their simulation results on six different profiles and their gaps in fuel economy with the baseline are also summarized in Tables 5.4 and 5.6. Accordingly, over the six profiles, the average gaps of CDRL-based, EDRL-based, and rule-based EMSs with the baseline, are calculated to be 8.03%, 10.00%, and 16.77%, respectively. Assuming the baseline owns 100% fuel-saving potential, the fuel-saving potential of other EMSs are evaluated by subtracting their average gaps from the fuel-saving level of the baseline (100%), as shown in Figure 5.13. The comparison indicates that considering terrain information in the derivation of this learning-based EMS could promote its fuel-saving potential by approximately 2% on average. Further, given the fact that the average annual operating mileage of commercial vehicles in North American is around 250,000 km [9], including terrain information in energy management will be much more beneficial for energy saving of hybrid buses and other commercial vehicles operating for a long time with high mileage and heavy loads.

The Influence on Decision Making

To understand the role of terrain information in deciding on a power split scheme, we will try to interpret the trained Actor network (the parameterized EMS) in this section. As explaining neural networks is difficult, decision tree models provide a direction to understand neural networks by analyzing the feature importance of different state variables. Because of their reliance on hierarchical decisions, the feature importance of single state variable can be estimated according to how often it is used in split points of a decision tree [115]. Therefore, the method in [115] is adopted for EMS interpretation; specifically, the package for gradient boosted regression tree (GBRT) in scikit-learn is used. All state transition data generated by the DRL-based EMS during simulations on a training profile are gathered then fed to train a decision tree.

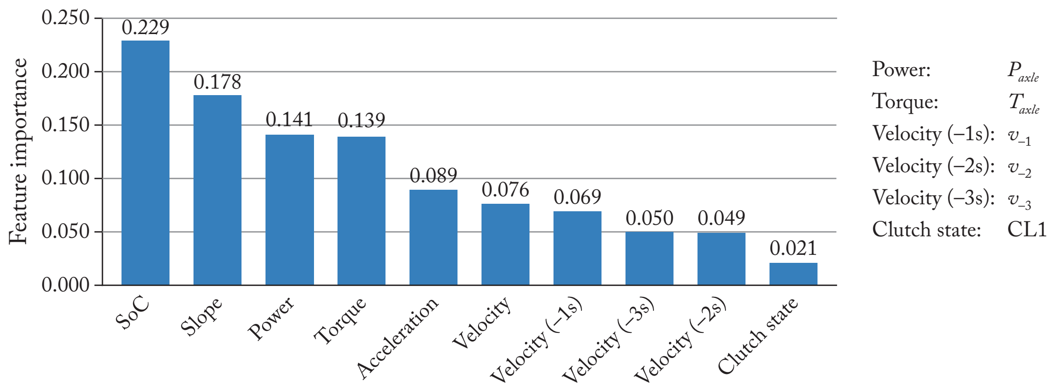


Figure 5.14: The feature importance of state variables for DRL-based EMS.

The relative importance of state variables in state space S to the power split scheme is shown in Figure 5.14. The strategy learned to put more emphasis on the SoC when making a decision, since its variation affected the reward directly during training. Terrain information also plays a crucial part in decisions, along with other state variables related to driving requirements, proving the benefit of considering terrain information from an aspect of data analysis.

On the other hand, due to the representation of required driving power and torque are more directly relevant to energy management, their importance appears higher than that of velocity and acceleration. As for the state of CL1, when the engine is on, it will keep unchanged regardless of varying engine power, while the change of clutch state matters to reward particularly when there is a need to start the engine. This may account for the lowest importance of clutch state derived from entire simulation data by GBRT. Overall, this result demonstrates that this learning method could provide an effective way to learn some knowledge of near-optimal EMS in larger state and action spaces.

5.5 SUMMARY

(1) The hybrid action space in energy management is illustrated by a case of power-split HEB. Based on a new parameterized strategy modeling approach, the corresponding energy management method is developed for this power-split HEB. This method is capable of searching the optimal EMS in a hybrid action space, realizing simultaneous optimization of the discrete mode selection and continuous energy allocations.

(2) A pre-training stage, which is developed to leverage the empirical knowledge and accelerate the training process, is also integrated into a DRL-based energy management method considering terrain information. The velocity-slope profile generation and strategy training algorithms are depicted for such EMSs. Comparative simulations are also conducted to examine the performance of the learning-based EMSs.

(3) The value of terrain information in learning-based EMSs is discussed in terms of three aspects, including the strategy performance in specific scenarios, fuel economy performance, and hierarchical decision extraction and interpretation of the strategy network. This potentially provides a reference for analyzing the impact of multi-source information fusion on strategy learning. The results show that incorporating the terrain information in DRL-based EMSs promoted the HEB's fuel economy by approximately 2%.

CHAPTER 6

An Online Integration Scheme for DRL-Based EMSs

As a new type of intelligent control methods, DRL requires massive exploration data samples, but due to the high cost of trial and error in real physical systems, most of the research on DRL is carried out based on simulators. The previous chapters also focus on the implementation and optimization of learning-based EMSs in a simulation environment. Currently, the developed vehicle control algorithms still need further assessments with the deployment in real vehicle controllers. Three aspects deserve more attention: (1) whether the control strategy development method is compatible with the mainstream development process of the vehicle control strategy; (2) whether common onboard controllers can meet the computing resource requirements of the strategy; and (3) whether the control strategy could perform consistently in real vehicle systems. Therefore, a hardware-in-the-loop (HIL) demonstration for the DRL-based EMS is described in this chapter. Because all DRL-based EMSs described in this book are represented by DNNs, they share the same hardware deployment procedure. The DRL-based EMS in Chapter 3 is utilized here for the illustration.

6.1 RECONSTRUCTION OF PARAMETERIZED EMSs IN MATLAB/SIMULINK

6.1.1 EXTRACTION OF PARAMETERIZED EMSs

In machine learning, to learn valuable data knowledge from large and cumbersome data, we usually use networks of greater complexity or size during training without paying much attention to their real-time performance. However, in the application stage, limited by the computing resources on deployment platforms, whether they have good real-time performance will be one of the key factors for algorithm evaluation. Taking a controller of Freescale MPC5644A as an example, we find that the strategy network we obtained in Chapter 3 (three hidden layers with 100 neurons per layer) could slow the single-step computation time of this VCU to about 500 ms or worse. Therefore, a strategy extraction method is introduced to facilitate the successful deployment of DRL-based EMSs on common controllers.

In practice, when the number of neurons in each hidden layer is reduced to 20, the controller could meet the requirement of 50 ms single-step computation, however, if a small network was directly used for policy training, it was prone to underfitting and failed to achieve the de-

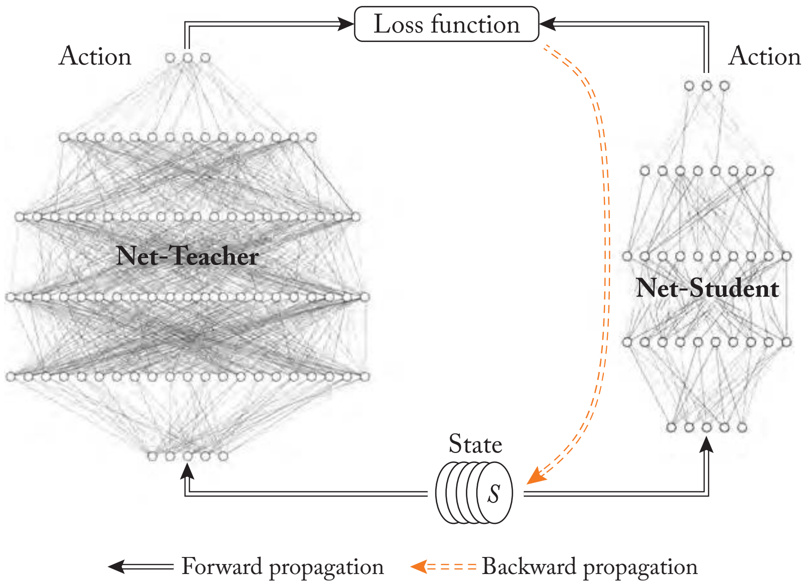


Figure 6.1: Illustration of the strategy extraction method.

sired effect. Therefore, referring to the knowledge distillation method [78], a smaller network is adopted for strategy extraction. Different from the knowledge distillation method in [78], as the EMSs are deterministic strategies, there are no soft labels like the action probability; hence, the hard labels of state transition data are directly used to train the surrogate network for strategy knowledge extraction. The overall extraction idea is illustrated in Figure 6.1. Taking the π_{BM} as an example, the extraction method is described as follows.

First, a smaller strategy network is built, namely the Net-Student $\mu_{Net-S}(s|\theta^{Net-S})$. The original strategy network of $\mu_{BM}(s|\theta^{\mu})$ will server as the Net-Teacher, $\mu_{Net-T}(s\ |\theta^{Net-T})$. The input layer, output layer, number of hidden layers, and activation functions of the Net-Student are aligned with the Net-Teacher. However, the number of neurons per hidden layer is reduced to 20, i.e., the Net-Student has fewer parameters and takes up less memory in VCUs.

Next, the collected state transition data from Net-Teacher will be used to train the Net-Student. The training goal is to minimize the output difference between the two strategy networks, resulting in the loss function as follows:

$$L_{comp}(\theta^{Net-S}) = \frac{1}{2}[\mu_{Net-T}(s|\theta^{Net-T}) - \mu_{Net-S}(s|\theta^{Net-S})]^2. \tag{6.1}$$

Compared to the training of Net-Teacher, the training of Net-Student is more efficient, thus it would be beneficial to train it by larger and more comprehensive dataset in practice.

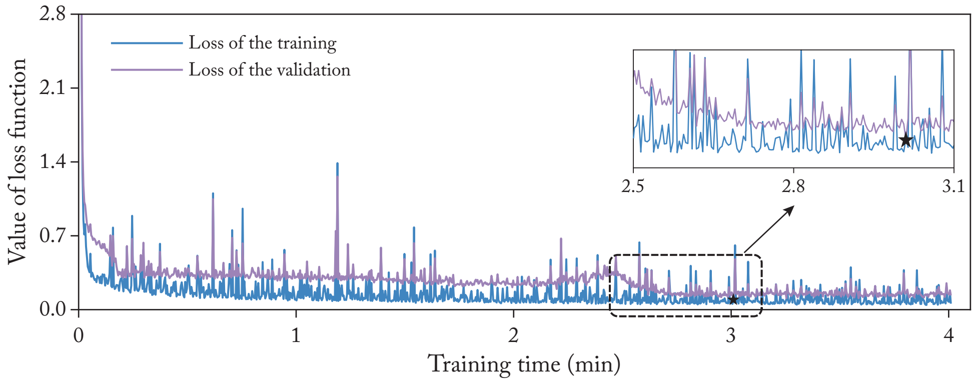


Figure 6.2: The loss function during strategy extraction.

Table 6.1: The output difference between the extracted and original EMSs

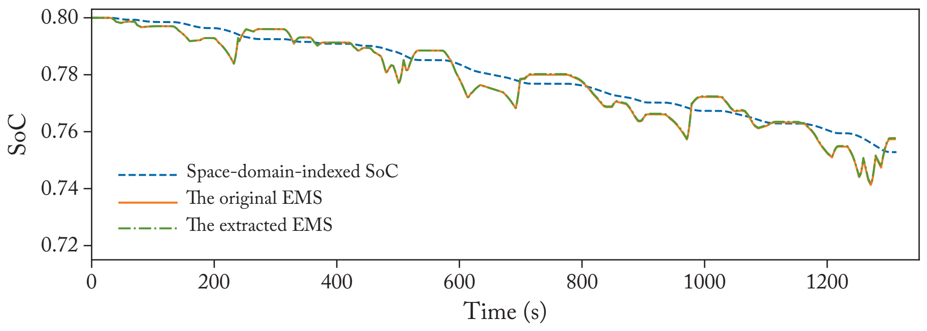
Driving Cycle	MSE (kW)	Driving Cycle	MSE (kW)
CTUDC×18	0.0843	WVUINTER	0.5655
CTUDC	0.0973	WLTP Class2	0.2632
WVUCITY	0.0756	JN1015	0.1100
WVUSUB	0.1689	Mixed	0.2137

Here, the long training profile (106.157 km) in Chapter 3 is used for state transition generation by Net-Teacher and the training of Net-Student.

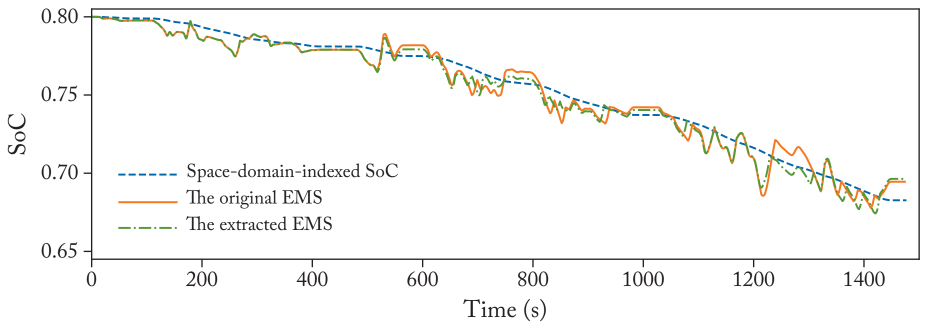
To visualize the strategy extraction, Figure 6.2 shows the mean square error between the output of Net-Teacher and Net-Student during training. Since the data generated by Net-Teacher is more regular, a smaller Net-Student can still quickly distill useful knowledge of energy management by learning the state-action mapping relationship. To avoid overfitting, the test profile in Chapter 3 is simply used here as the validation dataset. The best Net-Student on validation dataset is saved as the extracted EMS, $\pi_{comp} = \mu_{Net-S}(s|\theta^{Net-S})$.

The difference of Net-Teacher and Net-Student on the training profile and the testing profile is summarized in Table 6.1. We can see that their difference is about 0.2 kW, which is around 1% different considering the interval of control action δP_{eng}. Intuitively, the extracted strategy is almost the same as the original EMS. On the other hand, the Net-Student is trained to intimate the original EMS, while in actual applications, the relative error of the two strategies will be further narrowed under the constraint of output frequency adjustment.

Given an initial SoC of 0.8, the performance of the extracted EMS with output frequency adjustment on CTUDC and WLTP Class2 is shown in Figure 6.3. Correspondingly, their fuel



(a) Test profile: CTUDC



(b) Test profile: WLTP

Figure 6.3: SoC trajectories of the extracted and original EMSs.

economy performance is about 0.6% and 1.04% different from that of the original EMS. Therefore, we could say that this strategy extraction method works well for the DRL-based EMS.

6.1.2 RECONSTRUCTION IN MATLAB/SIMULINK

As the DRL-based EMS is usually trained and extracted in the Python environment, we will rebuild it in Matlab/Simulink environment to facilitate HIL-related experiments. Also, considering the fact that most VCU development works are based on Matlab/Simulink currently, it would be more convenient to transfer this DRL-based EMS in Matlab/Simulink environment.

Taking the extracted π_{BM} as an example, the function module in Simulink makes it easy to rebuild the parameterized DRL-based EMS by matrix multiplication, as shown in Figure 6.4. The input data includes the normalized state vector and the network weights obtained from the current workspace. The output is the desired increment/decrement value of the engine power.

Then, to accomplish the HIL test, the driver model, the strategy model, and the vehicle model are necessary, as shown in Figure 6.5. Specially, the strategy model can be further divided

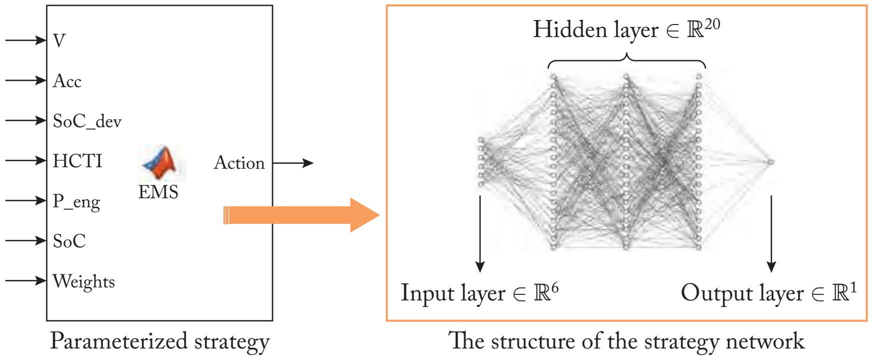

V
Acc
SoC_dev
HCTI
P_eng
SoC
Weights
EMS
Action
Parameterized strategy
Hidden layer $\in \mathbb{R}^{20}$
Input layer $\in \mathbb{R}^{6}$
Output layer $\in \mathbb{R}^{1}$
The structure of the strategy network

Figure 6.4: Schematic diagram of the parameterized EMS model.

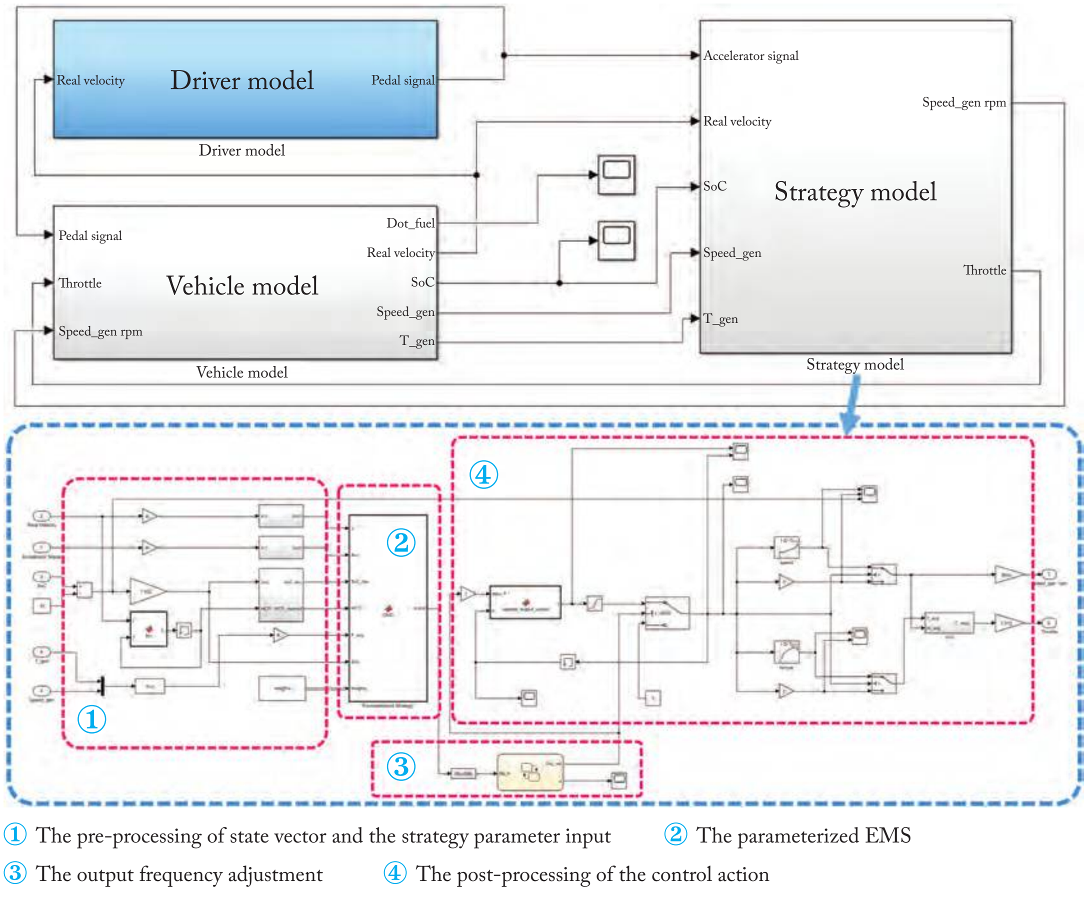


Figure 6.5: The simulation model in Simulink.

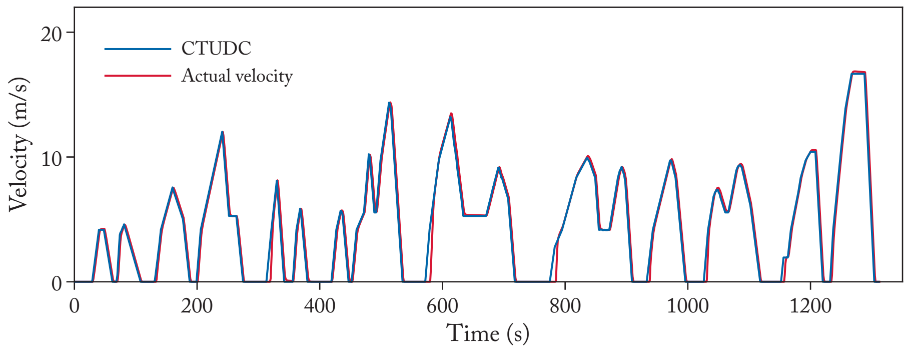


Figure 6.6: The comparison of actual velocity and target velocity.

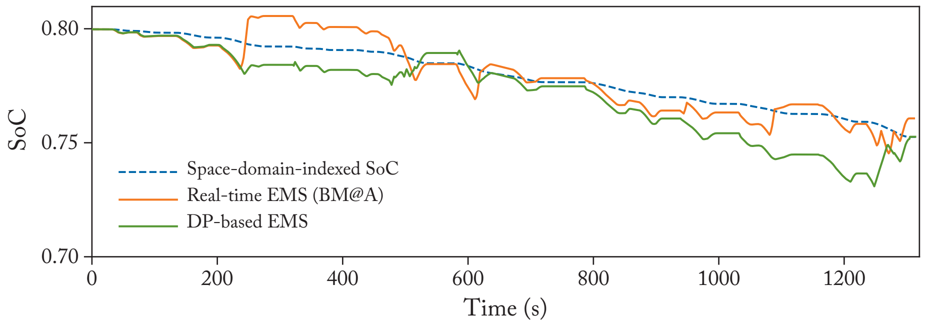
into four parts, including the pre-processing of state vector and the strategy parameter input, parameterized EMS, output frequency adjustment, and post-processing of the control action, which calculates the desired engine power and constraints the control signals.

6.2 EVALUATION BY HIL TEST

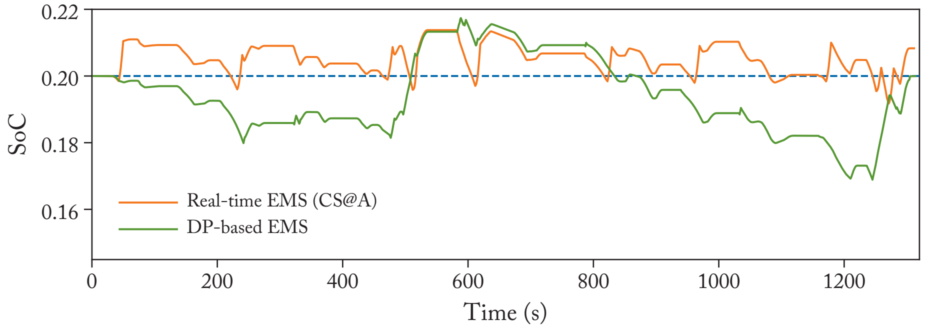
The above simulation model in Figure 6.5 will be separated into three independent nodes. The driver model node and the vehicle model node are imported into the CANoe software, while the strategy model is imported into the real controller, together with its parameters. The real controller is connected and communicated with the CANoe environment by a HIL test platform based on the MotoTron system. The strategy deployment on the real controller is referred to as the real-time DRL-based EMS.

Taking the CTUDC as the goal, the driver model tracks the target velocity in real time, and the actual velocity is shown in Figure 6.6. In order to follow the target velocity more closely, the driver model appears to be a bit more aggressive, and its maximum acceleration is about 1.12 m/s^2, which is slightly larger than that of the CTUDC (0.91 m/s^2). This is also consistent with the actual driving scenario, but leads to an increase in the overall energy consumption.

In the HIL test, the SoC trajectories of the DRL-based EMS enabled by history trip information are depicted in Figure 6.7, and Table 6.2 compares the performance of the real-time DRL-based EMS and the baseline. Comparing the strategy test results in the simulation environment (Figure 4.10 and Table 4.2), the real-time strategy can still meet our expectation: the electricity and fuel consumption are well-balanced at the blended mode, and the electricity remains stable at the CS mode. With errors from the actual driving cycle and the actual CAN bus communication, the fuel economy of the real-time strategy under BM and CS mode differs from the simulated results by only about 0.47% and 0.27%, respectively. This indicates that there is little influence on DRL-based EMS when it is migrated from the simulated environment to the hardware system (the VCU).



(a) Real-time EMS (BM@A)



(b) Real-time EMS (CS@A)

Figure 6.7: SoC trajectories of the real-time EMS from HIL test.

Table 6.2: Fuel economy of the real-time EMS from HIL test

Group	Method	SoC_{init}	SoC_{final}	N_{eng}	$Fuel_c$ (g)	Gap_{fuel_c}
1	DP-based EMS	0.8000	0.7527	9	136.1	—
	Real-time EMS (BM@A)	0.8000	0.7608	6	147.1	8.08%
2	DP-based EMS	0.2000	0.2000	16	240.0	—
	Real-time EMS (CS@A)	0.2000	0.2083	10	260.8	8.67%

On the other hand, because the real-time strategy is extracted from the original EMS with minimizing their output difference as the goal and the reconstructed real-time strategy in Matlab/Simulink is exactly the same as the extracted strategy, theoretically, the output of the real-time strategy should be close to that of the original strategy when given the same input state vector. Therefore, the obtained HIL test results are also as anticipated.

6.3 SUMMARY

Aiming at parameterized EMS based on neural networks, an EMS transfer method is introduced for the deployment on common VCUs, including the strategy extraction based on the knowledge distillation, and the strategy reconstruction in Matlab/Simulink. This method provides a way to ensure the compatibility between the development of learning-based EMSs with the current mainstream development process of the vehicle control strategies. A HIL test platform is utilized to examine the strategy transfer method and the real-time performance of the DRL-based EMS. The results demonstrates that the DRL-based EMS can work well on common controllers, as long as the strategy network are compressed to an appropriate size for the hardware.

CHAPTER 7
Conclusions

In this book, the DRL-based energy management methods are discussed for HEVs. As a kind of learning-based EMSs, the successful application of DRL in HEV energy management requires stable policy improvement during training and robust online performance. To enhance the application effect on different powertrain topologies, energy management problems, and application scenarios, several DRL-based EMSs, ranging from value-based strategy learning to policy gradient-based learning, and a general strategy transfer method for parameterized EMSs are described from Sections 3–6. Hopefully, this research work could provide some useful clues and basic algorithmic frameworks for future study on more complex and intelligent vehicle control methods with the incorporation of multi-source sensory information.

The research on deep learning and DRL has gained great momentum in the past decade and is still evolving rapidly, which would constantly bring fresh ideas to learning-based energy management in HEVs. Meanwhile, based on the current work, attention regarding the following aspects may also be needed in the future.

1. Comprehensive control-oriented HEV modeling method that considers the transient characteristics of power components and runs fast. It also plays a key role in bridging the gap between strategy simulation and strategy deployment on real hybrid powertrain systems. The influence of realistic factors on learning-based EMSs is also worthy of attention, such as the synchronization of communication signals, sensor noise, vehicle state estimation errors, etc. This is also a common challenge in the field of machine learning.

2. The transfer learning of parameterized EMSs across different types of HEVs and driving styles. As learning-based EMSs require enormous training data, this effort will further increase the portability and development cost of strategy learning.

3. Cooperative and online learning of EMSs. The parameterized EMSs described in this book were trained first and then applied without online update of the parameters. However, if the research cases are extended to a connected environment where massive real-time driving data could be collected, the issue of data diversity will not be the constraint of online strategy learning in the real world. Furthermore, combined with asynchronous or multi-agent DRL, the possibility of collaborative learning for energy-efficient control of a group of HEVs could be explored under this scenario as well.

Bibliography

[1] Imran Rahman, Pandian M. Vasant, Balbir Singh Mahinder Singh, M. Abdullah-Al-Wadud, and Nadia Adnan. Review of recent trends in optimization techniques for plug-in hybrid, and electric vehicle charging infrastructures. *Renewable and Sustainable Energy Reviews*, 58:1039–1047, 2016. DOI: 10.1016/j.rser.2015.12.353 2

[2] Clément Dépature, Walter Lhomme, Pierre Sicard, Alain Bouscayrol, and Loïc Boulon. Real-time backstepping control for fuel cell vehicle using supercapacitors. *IEEE Transactions on Vehicular Technology*, 67(1):306–314, 2017. DOI: 10.1109/tvt.2017.2728823 2

[3] F. Wasbari, R. A. Bakar, L. M. Gan, M. M. Tahir, and A. A. Yusof. A review of compressed-air hybrid technology in vehicle system. *Renewable and Sustainable Energy Reviews*, 67:935–953, 2017. DOI: 10.1016/j.rser.2016.09.039 2

[4] Sun Hui, Yang Lifu, Jing Junqing, and Luo Yanling. Control strategy of hydraulic/electric synergy system in heavy hybrid vehicles. *Energy Conversion and Management*, 52(1):668–674, 2011. DOI: 10.1016/j.enconman.2010.07.045 2

[5] Faramarz Faraji, Abbas Majazi, Kamal Al-Haddad, et al. A comprehensive review of flywheel energy storage system technology. *Renewable and Sustainable Energy Reviews*, 67:477–490, 2017. DOI: 10.1016/j.rser.2016.09.060 2

[6] Hirohito Ide, Yoshihiro Sunaga, and Naritomo Higuchi. Development of sport hybrid i-MMD control system for 2014 model year accord. *Introduction of New Technologies*, 2014. 3

[7] Jinming Liu and Huei Peng. A systematic design approach for two planetary gear split hybrid vehicles. *Vehicle System Dynamics*, 48(11):1395–1412, 2010. DOI: 10.1080/00423114.2010.512634 3

[8] Shanshan Xie, Hongwen He, and Jiankun Peng. An energy management strategy based on stochastic model predictive control for plug-in hybrid electric buses. *Applied Energy*, 196:279–288, 2017. DOI: 10.1016/j.apenergy.2016.12.112 3, 6, 51

[9] Hong Wang, Yanjun Huang, Amir Soltani, Amir Khajepour, and Dongpu Cao. Cyber-physical predictive energy management for through-the-road hybrid vehicles. *IEEE Transactions on Vehicular Technology*, 68(4):3246–3256, 2019. DOI: 10.1109/tvt.2019.2902485 3, 84, 97

[10] Yimin Gao and Mehrdad Ehsani. Design and control methodology of plug-in hybrid electric vehicles. *IEEE Transactions on Industrial Electronics*, 57(2):633–640, 2009. DOI: 10.1109/tie.2009.2027918 3

[11] Jeffrey Gonder and Tony Markel. Energy management strategies for plug-in hybrid electric vehicles. *Technical Report*, SAE Technical Paper, 2007. DOI: 10.4271/2007-01-0290 3

[12] Chao Sun, Scott Jason Moura, Xiaosong Hu, J. Karl Hedrick, and Fengchun Sun. Dynamic traffic feedback data enabled energy management in plug-in hybrid electric vehicles. *IEEE Transactions on Control Systems Technology*, 23(3):1075–1086, 2014. DOI: 10.1109/tcst.2014.2361294 3, 6, 56, 57, 67

[13] Menyang Zhang, Yan Yang, and Chunting Chris Mi. Analytical approach for the power management of blended-mode plug-in hybrid electric vehicles. *IEEE Transactions on Vehicular Technology*, 61(4):1554–1566, 2012. DOI: 10.1109/tvt.2012.2187318 3

[14] Antonio Sciarretta, Lorenzo Serrao, P. C. Dewangan, Paolino Tona, E. N. D. Bergshoeff, C. Bordons, L. Charmpa, Ph Elbert, Lars Eriksson, T. Hofman, et al. A control benchmark on the energy management of a plug-in hybrid electric vehicle. *Control Engineering Practice*, 29:287–298, 2014. DOI: 10.1016/j.conengprac.2013.11.020 3

[15] Zheng Chen, Chunting Chris Mi, Jun Xu, Xianzhi Gong, and Chenwen You. Energy management for a power-split plug-in hybrid electric vehicle based on dynamic programming and neural networks. *IEEE Transactions on Vehicular Technology*, 63(4):1567–1580, 2013. DOI: 10.1109/tvt.2013.2287102 3, 6

[16] Cong Hou, Minggao Ouyang, Liangfei Xu, and Hewu Wang. Approximate pontryagin's minimum principle applied to the energy management of plug-in hybrid electric vehicles. *Applied Energy*, 115:174–189, 2014. DOI: 10.1016/j.apenergy.2013.11.002 3

[17] Qiuming Gong, Yaoyu Li, and Zhong-Ren Peng. Trip-based optimal power management of plug-in hybrid electric vehicles. *IEEE Transactions on Vehicular Technology*, 57(6):3393–3401, 2008. DOI: 10.1109/tvt.2008.921622 3

[18] Chen Zhang and Ardalan Vahidi. Route preview in energy management of plug-in hybrid vehicles. *IEEE Transactions on Control Systems Technology*, 20(2):546–553, 2011. DOI: 10.1109/tcst.2011.2115242 3

[19] Jiankun Peng, Hongwen He, and Rui Xiong. Rule based energy management strategy for a series—parallel plug-in hybrid electric bus optimized by dynamic programming. *Applied Energy*, 185:1633–1643, 2017. DOI: 10.1016/j.apenergy.2015.12.031 3, 4, 6, 57

[20] Siguang G. Li, S. M. Sharkh, Frank C. Walsh, and Cheng-Ning Zhang. Energy and battery management of a plug-in series hybrid electric vehicle using fuzzy logic. *IEEE Transactions on Vehicular Technology*, 60(8):3571–3585, 2011. DOI: 10.1109/tvt.2011.2165571 3

[21] Hamid Khayyam and Alireza Bab-Hadiashar. Adaptive intelligent energy management system of plug-in hybrid electric vehicle. *Energy*, 69:319–335, 2014. DOI: 10.1016/j.energy.2014.03.020 3, 6

[22] Laura Tribioli, Michele Barbieri, Roberto Capata, Enrico Sciubba, Elio Jannelli, and Gino Bella. A real time energy management strategy for plug-in hybrid electric vehicles based on optimal control theory. *Energy Procedia*, 45:949–958, 2014. DOI: 10.1016/j.egypro.2014.01.100 4

[23] Clara Marina Martinez, Xiaosong Hu, Dongpu Cao, Efstathios Velenis, Bo Gao, and Matthias Wellers. Energy management in plug-in hybrid electric vehicles: Recent progress and a connected vehicles perspective. *IEEE Transactions on Vehicular Technology*, 66(6):4534–4549, 2016. DOI: 10.1109/tvt.2016.2582721 4, 5, 57

[24] Gino Paganelli, Gabriele Ercole, Avra Brahma, Yann Guezennec, and Giorgio Rizzoni. General supervisory control policy for the energy optimization of charge-sustaining hybrid electric vehicles. *JSAE Review*, 22(4):511–518, 2001. DOI: 10.1016/s0389-4304(01)00138-2 5

[25] Lorenzo Serrao, Simona Onori, and Giorgio Rizzoni. ECMS as a realization of pontryagin's minimum principle for hev control. *American Control Conference*, pages 3964–3969, IEEE, 2009. DOI: 10.1109/acc.2009.5160628 5

[26] Namwook Kim, Aymeric Rousseau, and Daeheung Lee. A jump condition of PMP-based control for phevs. *Journal of Power Sources*, 196(23):10380–10386, 2011. DOI: 10.1016/j.jpowsour.2011.07.003 5

[27] Bo Gu and Giorgio Rizzoni. An adaptive algorithm for hybrid electric vehicle energy management based on driving pattern recognition. *ASME International Mechanical Engineering Congress and Exposition*, pages 249–258, American Society of Mechanical Engineers Digital Collection, 2006. DOI: 10.1115/imece2006-13951 5

[28] Bo Geng, James K. Mills, and Dong Sun. Energy management control of microturbine-powered plug-in hybrid electric vehicles using the telemetry equivalent consumption minimization strategy. *IEEE Transactions on Vehicular Technology*, 60(9):4238–4248, 2011. DOI: 10.1109/tvt.2011.2172646 5

[29] Lin Li, Serdar Coskun, Fengqi Zhang, Reza Langari, and Junqiang Xi. Energy management of hybrid electric vehicle using vehicle lateral dynamic in velocity prediction. *IEEE Transactions on Vehicular Technology*, 68(4):3279–3293, 2019. DOI: 10.1109/tvt.2019.2896260 5

[30] Hoseinali Borhan, Ardalan Vahidi, Anthony M. Phillips, Ming L. Kuang, Ilya V. Kolmanovsky, and Stefano Di Cairano. MPC-based energy management of a power-split hybrid electric vehicle. *IEEE Transactions on Control Systems Technology*, 20(3):593–603, 2011. DOI: 10.1109/tcst.2011.2134852 5

[31] Hong Wang, Yanjun Huang, Amir Khajepour, Hongwen He, and Dongpu Cao. A novel energy management for hybrid off-road vehicles without future driving cycles as a priori. *Energy*, 133:929–940, 2017. DOI: 10.1016/j.energy.2017.05.172 5, 56

[32] Yanjun Huang, Hong Wang, Amir Khajepour, Hongwen He, and Jie Ji. Model predictive control power management strategies for hevs: A review. *Journal of Power Sources*, 341:91–106, 2017. DOI: 10.1016/j.jpowsour.2016.11.106 5, 56

[33] Xiao Lin, Harpreetsingh Banvait, Sohel Anwar, and Yaobin Chen. Optimal energy management for a plug-in hybrid electric vehicle: Real-time controller. In *Proc. of the American Control Conference*, pages 5037–5042, IEEE, 2010. DOI: 10.1109/acc.2010.5530731 5, 6

[34] Zeyu Chen, Rui Xiong, and Jiayi Cao. Particle swarm optimization-based optimal power management of plug-in hybrid electric vehicles considering uncertain driving conditions. *Energy*, 96:197–208, 2016. DOI: 10.1016/j.energy.2015.12.071 5

[35] Zheng Chen, Chris Chunting Mi, Rui Xiong, Jun Xu, and Chenwen You. Energy management of a power-split plug-in hybrid electric vehicle based on genetic algorithm and quadratic programming. *Journal of Power Sources*, 248:416–426, 2014. DOI: 10.1016/j.jpowsour.2013.09.085 5

[36] Yi Lu Murphey, Jungme Park, Leonidas Kiliaris, Ming L. Kuang, M. Abul Masrur, Anthony M. Phillips, and Qing Wang. Intelligent hybrid vehicle power control—part II: Online intelligent energy management. *IEEE Transactions on Vehicular Technology*, 62(1):69–79, 2012. DOI: 10.1109/tvt.2012.2217362 5

[37] Clément Dextreit and Ilya V. Kolmanovsky. Game theory controller for hybrid electric vehicles. *IEEE Transactions on Control Systems Technology*, 22(2):652–663, 2013. DOI: 10.1109/tcst.2013.2254597 5

[38] Andreas A. Malikopoulos. Supervisory power management control algorithms for hybrid electric vehicles: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 15(5):1869–1885, 2014. DOI: 10.1109/tits.2014.2309674 5

[39] He Hongwen, Guo Jinquan, Peng Jiankun, Tan Huachun, and Sun Chao. Real-time global driving cycle construction and the application to economy driving pro system in plug-in hybrid electric vehicles. *Energy*, 152:95–107, 2018. DOI: 10.1016/j.energy.2018.03.061 6, 56

[40] Guo Jinquan, He Hongwen, Peng Jiankun, and Zhou Nana. A novel MPC-based adaptive energy management strategy in plug-in hybrid electric vehicles. *Energy*, 175:378–392, 2019. DOI: 10.1016/j.energy.2019.03.083 6

[41] Yi Lu Murphey, Jungme Park, Zhihang Chen, Ming L. Kuang, M. Abul Masrur, and Anthony M. Phillips. Intelligent hybrid vehicle power control—part I: Machine learning of optimal vehicle power. *IEEE Transactions on Vehicular Technology*, 61(8):3519–3530, 2012. DOI: 10.1109/tvt.2012.2206064 6, 26

[42] Yuecheng Li, Hongwen He, and Jiankun Peng. An adaptive online prediction method with variable prediction horizon for future driving cycle of the vehicle. *IEEE Access*, 6:33062–33075, 2018. DOI: 10.1109/access.2018.2840536 6

[43] Liang Li, Sixiong You, Chao Yang, Bingjie Yan, Jian Song, and Zheng Chen. Driving-behavior-aware stochastic model predictive control for plug-in hybrid electric buses. *Applied Energy*, 162:868–879, 2016. DOI: 10.1016/j.apenergy.2015.10.152 6

[44] Chao Sun, Xiaosong Hu, Scott J. Moura, and Fengchun Sun. Velocity predictors for predictive energy management in hybrid electric vehicles. *IEEE Transactions on Control Systems Technology*, 23(3):1197–1204, 2014. DOI: 10.1109/tcst.2014.2359176 6, 56, 72

[45] Fengqi Zhang, Junqiang Xi, and Reza Langari. Real-time energy management strategy based on velocity forecasts using v2v and v2i communications. *IEEE Transactions on Intelligent Transportation Systems*, 18(2):416–430, 2016. DOI: 10.1109/tits.2016.2580318 6, 26

[46] José R. Vázquez-Canteli and Zoltán Nagy. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Applied Energy*, 235:1072–1089, 2019. DOI: 10.1016/j.apenergy.2018.11.002 7, 15, 21

[47] Xue Lin, Yanzhi Wang, Paul Bogdan, Naehyuck Chang, and Massoud Pedram. Reinforcement learning based power management for hybrid electric vehicles. *Proc. of the IEEE/ACM International Conference on Computer-Aided Design*, pages 32–38, IEEE Press, 2014. DOI: 10.1109/iccad.2014.7001326 7

[48] Teng Liu, Yuan Zou, Dexing Liu, and Fengchun Sun. Reinforcement learning of adaptive energy management with transition probability for a hybrid electric tracked vehicle. *IEEE Transactions on Industrial Electronics*, 62(12):7837–7846, 2015. DOI: 10.1109/tie.2015.2475419 7

[49] Jingda Wu, Hongwen He, Jiankun Peng, Yuecheng Li, and Zhanjiang Li. Continuous reinforcement learning of energy management with deep Q network for a power split hybrid electric bus. *Applied Energy*, 222:799–811, 2018. DOI: 10.1016/j.apenergy.2018.03.104 7

[50] Yuecheng Li, Hongwen He, Jiankun Peng, and Hong Wang. Deep reinforcement learning-based energy management for a series hybrid electric vehicle enabled by history cumulative trip information. *IEEE Transactions on Vehicular Technology*, 68(8):7416–7430, 2019. DOI: 10.1109/tvt.2019.2926472 7

[51] Yuecheng Li, Hongwen He, Amir Khajepour, Hong Wang, and Jiankun Peng. Energy management for a power-split hybrid electric bus via deep reinforcement learning with terrain information. *Applied Energy*, 255:113762, 2019. DOI: 10.1016/j.apenergy.2019.113762 7

[52] Hicham Chaoui, Hamid Gualous, Loic Boulon, and Sousso Kelouwani. Deep reinforcement learning energy management system for multiple battery based electric vehicles. *IEEE Vehicle Power and Propulsion Conference (VPPC)*, pages 1–6, 2018. DOI: 10.1109/vppc.2018.8605023 7

[53] Yuankai Wu, Huachun Tan, Jiankun Peng, Hailong Zhang, and Hongwen He. Deep reinforcement learning of energy management with continuous control strategy and traffic information for a series-parallel plug-in hybrid electric bus. *Applied Energy*, 247:454–466, 2019. DOI: 10.1016/j.apenergy.2019.04.021 7

[54] Paul A. Gagniuc. *Markov Chains: From Theory to Implementation and Experimentation*. John Wiley & Sons, 2017. DOI: 10.1002/9781119387596 9

[55] Ronald A. Howard. Dynamic programming and Markov processes. John Wiley, 1960. 9

[56] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2018. DOI: 10.1109/tnn.1998.712192 9, 53

[57] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015. DOI: 10.1038/nature14236 10, 12, 17, 18, 19, 28, 29, 30, 82

[58] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017. DOI: 10.1038/nature24270 10

[59] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019. DOI: 10.1126/scirobotics.aau5872 10, 21

[60] Piotr Mirowski, Matt Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Andrew Zisserman, Raia Hadsell, et al. Learning to navigate in cities without a map. *Advances in Neural Information Processing Systems*, pages 2419–2430, 2018. 10

[61] Maximilian Jaritz, Raoul de Charette, Marin Toromanoff, Etienne Perot, and Fawzi Nashashibi. End-to-end race driving with deep reinforcement learning. *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2070–2075, 2018. DOI: 10.1109/icra.2018.8460934 10

[62] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ArXiv Preprint ArXiv:1509.02971*, 2015. 12, 20, 53

[63] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv Preprint ArXiv:1707.06347*, 2017. 12, 20

[64] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *32nd AAAI Conference on Artificial Intelligence*, 2018. 12

[65] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *ArXiv Preprint ArXiv:1706.01905*, 2017. 12

[66] Richard E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 2015. DOI: 10.1515/9781400874668 15

[67] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with deep reinforcement learning. *ArXiv Preprint ArXiv:1312.5602*, 2013. 17, 18, 29

[68] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. DOI: 10.1038/nature14539 17

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. DOI: 10.1145/3065386 17

[70] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *ArXiv Preprint ArXiv:1511.05952*, 2015. 19, 36, 38

[71] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. *ArXiv Preprint ArXiv:1803.00933*, 2018. 19, 20, 36

[72] Hado V. Hasselt. Double Q-learning. *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010. 19

[73] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. *30th AAAI Conference on Artificial Intelligence*, 2016. 19, 35

[74] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *ArXiv Preprint ArXiv:1511.06581*, 2015. 19

[75] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. *Advances in Neural Information Processing Systems*, pages 4026–4034, 2016. 19

[76] Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *ArXiv Preprint ArXiv:1511.06295*, 2015. 19

[77] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *ArXiv Preprint ArXiv:1511.06342*, 2015. 19

[78] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *ArXiv Preprint ArXiv:1503.02531*, 2015. 19, 102

[79] Jan Peters. Policy gradient methods. *Scholarpedia*, 5(11):3698, 2010. DOI: 10.4249/scholarpedia.3698 19

[80] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. *International Conference on Machine Learning*, pages 387–395, PMLR, 2014. 19, 20, 53

[81] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. *International Conference on Machine Learning*, pages 1889–1897, 2015. 20, 32

[82] Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *ArXiv Preprint ArXiv:1704.03073*, 2017. 20

[83] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *ArXiv Preprint ArXiv:1707.08817*, 2017. 20

[84] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *International Conference on Machine Learning*, pages 1928–1937, 2016. 20

[85] Nicolas Heess, Dhruva T. B., Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Eslami, et al. Emergence of locomotion behaviours in rich environments. *ArXiv Preprint ArXiv:1707.02286*, 2017. 20

[86] Tejas D. Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in Neural Information Processing Systems*, pages 3675–3683, 2016. 20

[87] Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *ArXiv Preprint ArXiv:1611.05763*, 2016. 20

[88] Yee Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017. 20

[89] Haotian Fu, Hongyao Tang, Jianye Hao, Zihan Lei, Yingfeng Chen, and Changjie Fan. Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces. *ArXiv Preprint ArXiv:1903.04959*, 2019. DOI: 10.24963/ijcai.2019/323 20

[90] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *ArXiv Preprint ArXiv:1912.06680*, 2019. 21

[91] Shaobo Xie, Xiaosong Hu, Zongke Xin, and Liang Li. Time-efficient stochastic model predictive energy management for a plug-in hybrid electric bus with an adaptive reference state-of-charge advisory. *IEEE Transactions on Vehicular Technology*, 67(7):5671–5682, 2018. DOI: 10.1109/tvt.2018.2798662 26

[92] Liang Li, Chao Yang, Yahui Zhang, Lipeng Zhang, and Jian Song. Correctional DP-based energy management strategy of plug-in hybrid electric bus for city-bus

route. *IEEE Transactions on Vehicular Technology*, 64(7):2792–2803, 2014. DOI: 10.1109/tvt.2014.2352357 26

[93] Hadi Kazemi, Yaser P. Fallah, Andrew Nix, and Scott Wayne. Predictive AECMS by utilization of intelligent transportation systems for hybrid electric vehicle powertrain control. *IEEE Transactions on Intelligent Vehicles*, 2(2):75–84, 2017. DOI: 10.1109/tiv.2017.2716839 26

[94] Teng Liu, Xiaosong Hu, Weihao Hu, and Yuan Zou. A heuristic planning reinforcement learning-based energy management for power-split plug-in hybrid electric vehicles. *IEEE Transactions on Industrial Informatics*, 15(12):6436–6445, 2019. DOI: 10.1109/tii.2019.2903098 26

[95] Teng Liu, Xiaosong Hu, Shengbo Eben Li, and Dongpu Cao. Reinforcement learning optimized look-ahead energy management of a parallel hybrid electric vehicle. *IEEE/ASME Transactions on Mechatronics*, 22(4):1497–1507, 2017. DOI: 10.1109/tmech.2017.2707338 26

[96] Xuefeng Han, Hongwen He, Jingda Wu, Jiankun Peng, and Yuecheng Li. Energy management based on reinforcement learning with double deep Q-learning for a hybrid electric tracked vehicle. *Applied Energy*, 254:113708, 2019. DOI: 10.1016/j.apenergy.2019.113708 26

[97] Engin Ozatay, Simona Onori, James Wollaeger, Umit Ozguner, Giorgio Rizzoni, Dimitar Filev, John Michelini, and Stefano Di Cairano. Cloud-based velocity profile optimization for everyday driving: A dynamic-programming-based solution. *IEEE Transactions on Intelligent Transportation Systems*, 15(6):2491–2505, 2014. DOI: 10.1109/tits.2014.2319812 28

[98] Chang Liu and Yi Lu Murphey. Power management for plug-in hybrid electric vehicles using reinforcement learning with trip information. *IEEE Transportation Electrification Conference and Expo (ITEC)*, pages 1–6, 2014. DOI: 10.1109/itec.2014.6861862 28

[99] UFLDL tutorial—multi-layer neural network. http://ufldl.stanford.edu/tutorial/supervised/multilayerneuralnetworks/ 28

[100] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*, 2014. 30, 54

[101] Joseph O'Neill, Barty Pleydell-Bouverie, David Dupret, and Jozsef Csicsvari. Play it again: Reactivation of waking experience and memory. *Trends in Neurosciences*, 33(5):220–229, 2010. DOI: 10.1016/j.tins.2010.01.006 30

[102] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proc. of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 31, 61

[103] Hongwen He, Henglu Tang, and Ximing Wang. Global optimal energy management strategy research for a plug-in series-parallel hybrid electric bus by using dynamic programming. *Mathematical Problems in Engineering*, 2013. DOI: 10.1155/2013/708261 41

[104] Olle Sundstrom and Lino Guzzella. A generic dynamic programming Matlab function. *IEEE Control Applications (CCA) and Intelligent Control (ISIC)*, pages 1625–1630, 2009. DOI: 10.1109/cca.2009.5281131 41

[105] Olle Sundström, Daniel Ambühl, and Lino Guzzella. On implementation of dynamic programming for optimal control problems with final state constraints. *Oil and Gas Science and Technology—Revue de L'Institut Français du Pétrole*, 65(1):91–102, 2010. DOI: 10.2516/ogst/2009020 41, 45

[106] Yanjun Huang, Amir Khajepour, Tianjun Zhu, and Haitao Ding. A supervisory energy-saving controller for a novel anti-idling system of service vehicles. *IEEE-ASME Transactions on Mechatronics*, 22(2):1037–1046, 2017. DOI: 10.1109/tmech.2016.2631897 44

[107] Yuankai Wu, Huachun Tan, Lingqiao Qin, Bin Ran, and Zhuxi Jiang. A hybrid deep learning based traffic flow prediction method and its understanding. *Transportation Research Part C Emerging Technologies*, 90(MAY):166–180, 2018. DOI: 10.1016/j.trc.2018.03.001 56

[108] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C*, 54:187–197, 2015. DOI: 10.1016/j.trc.2015.03.014 56

[109] Yanjun Huang, Amir Khajepour, and Hong Wang. A predictive power management controller for service vehicle anti-idling systems without *a priori* information. *Applied Energy*, 2016. DOI: 10.1016/j.apenergy.2016.08.143 56

[110] S. G. Wirasingha and A. Emadi. Classification and review of control strategies for plug-in hybrid electric vehicles. *IEEE Transactions on Vehicular Technology*, 60(1):111–122, 2011. DOI: 10.1109/tvt.2010.2090178 57

[111] Yuan Zou, Teng Liu, Dexing Liu, and Fengchun Sun. Reinforcement learning-based real-time energy management for a hybrid tracked vehicle. *Applied Energy*, 171:372–382, 2016. DOI: 10.1016/j.apenergy.2016.03.082 67

[112] Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. *ArXiv Preprint ArXiv:1511.04143*, 2015. 82

[113] Chen Zhang, Ardalan Vahidi, Pierluigi Pisu, Xiaopeng Li, and Keith Tennant. Utilizing road grade preview for increasing fuel economy of hybrid vehicles. 42(15):168–173, 2009. DOI: 10.3182/20090902-3-us-2007.0058 84

[114] Chen Zhang, Ardalan Vahidi, Pierluigi Pisu, Xiaopeng Li, and Keith Tennant. Role of terrain preview in energy management of hybrid electric vehicles. *IEEE Transactions on Vehicular Technology*, 59(3):1139–1147, 2010. DOI: 10.1109/tvt.2009.2038707 86

[115] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *ArXiv Preprint ArXiv:1711.09784*, 2017. 97

Authors' Biographies

YUECHENG LI

Yuecheng Li is currently working at the Beijing Institute of Specialized Machinery. He obtained his Ph.D. from Beijing Institute of Technology in 2021 and studied in the Mechatronic Vehicle Systems Lab, University of Waterloo, as a visiting student from 2018–2019. His research interests include hybrid powertrains and energy management, intelligent control theories, and machine learning applied to vehicles.

HONGWEN HE

Hongwen He is currently a Professor with the National Engineering Laboratory for Electric Vehicles, School of Mechanical Engineering, Beijing Institute of Technology. He has authored or coauthored 126 EI-indexed papers, 82 SCI-indexed papers, and 17 ESI highly cited papers. He is the recipient of the second prize of the Chinese National Science and Technology Award, the first prize of natural science by the Ministry of Education, and the first prize of technological invention of China's automobile industry.