Engineering 54 (2025) 370–387

Contents lists available at ScienceDirect

Engineering

journal homepage: www.elsevier.com/locate/eng

Research
Industrial Engineering—Article

LearningEMS: A Unified Framework and Open-Source Benchmark for Learning-Based Energy Management of Electric Vehicles

Yong Wang [a], Hongwen He [a,*], Yuankai Wu [b], Pei Wang [a], Haoyu Wang [a], Renzong Lian [c], Jingda Wu [d], Qin Li [e], Xiangfei Meng [a], Yingjuan Tang [a], Fengchun Sun [a], Amir Khajepour [f]

[a] School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100081, China
[b] College of Computer Science, Sichuan University, Chengdu 610065, China
[c] Shenzhen International Graduate School, Tsinghua University, Shenzhen 518000, China
[d] The Hong Kong Polytechnic University, Hong Kong 999077, China
[e] School of Mechanical Engineering, Guangxi University, Nanning 530000, China
[f] Department of Mechanical and Mechatronics Engineering, University of Waterloo, Waterloo, ON N2L3G1, Canada

ARTICLE INFO

Article history:
Received 15 February 2024
Revised 24 September 2024
Accepted 8 October 2024
Available online 11 December 2024

Keywords:
Energy management
Electric vehicles
Reinforcement learning
Machine learning
Open-source benchmark

ABSTRACT

An effective energy management strategy (EMS) is essential to optimize the energy efficiency of electric vehicles (EVs). With the advent of advanced machine learning techniques, the focus on developing sophisticated EMS for EVs is increasing. Here, we introduce LearningEMS: a unified framework and open-source benchmark designed to facilitate rapid development and assessment of EMS. LearningEMS is distinguished by its ability to support a variety of EV configurations, including hybrid EVs, fuel cell EVs, and plug-in EVs, offering a general platform for the development of EMS. The framework enables detailed comparisons of several EMS algorithms, encompassing imitation learning, deep reinforcement learning (RL), offline RL, model predictive control, and dynamic programming. We rigorously evaluated these algorithms across multiple perspectives: energy efficiency, consistency, adaptability, and practicability. Furthermore, we discuss state, reward, and action settings for RL in EV energy management, introduce a policy extraction and reconstruction method for learning-based EMS deployment, and conduct hardware-in-the-loop experiments. In summary, we offer a unified and comprehensive framework that comes with three distinct EV platforms, over 10 000 km of EMS policy data set, ten state-of-the-art algorithms, and over 160 benchmark tasks, along with three learning libraries. Its flexible design allows easy expansion for additional tasks and applications. The open-source algorithms, models, data sets, and deployment processes foster additional research and innovation in EV and broader engineering domains.

1. Introduction

The automotive industry has recently undergone a transformative shift fueled by the growing global emphasis on sustainability and environmental conservation. At the forefront of this revolution is the advent of electric vehicles (EVs), representing a paradigm shift away from the traditional internal combustion engine vehicles (ICEVs) [1,2]. The EV landscape encompasses a diverse array of technologies, including battery EVs (BEVs), hybrid EVs (HEVs), plug-in HEVs (PHEVs), and fuel cell EVs (FCEVs). This spectrum of electric mobility solutions addresses concerns related to greenhouse gas emissions, fossil fuel dependency, and the broader ecological impact of transportation [3]. In particular, the widespread adoption of PHEVs has experienced a significant increase in recent years, fueled by their favorable fuel economy, superior performance, and extended range [4]. The Build Your Dreams (BYD; China) electric vehicle manufacturer achieved sales of 3.02 million electric vehicles in the 2023, with PHEVs accounting for 47.9% of the total sales.

Considering the integration of multiple power sources in HEVs, PHEVs, and FCEVs, the effective implementation of an energy management strategy (EMS) assumes paramount importance. The purpose of an EMS is to judiciously allocate power among the diverse energy systems integrated into these vehicles [5].

* Corresponding author.
E-mail address: hwhebit@bit.edu.cn (H. He).

https://doi.org/10.1016/j.eng.2024.10.021

Unlike traditional vehicle control problems that often have instantaneous solutions, the EMS for these advanced vehicles involves decision-making and control over the entire driving cycle horizon. This comprehensive approach requires the consideration of both short- and long-term energy availability. To tackle the intricate energy management challenge, various EMS approaches have been proposed over the last decades. These solutions can be broadly categorized into three main groups: rule-based, optimal control, and learning-based strategies. Rule-based strategies lack optimality guarantees and require time-consuming design and calibration practices [6]. Commonly used optimization-based methods include dynamic programming (DP), equivalent consumption minimization strategy (ECMS), Pontryagin's minimum principle (PMP), and model predictive control (MPC) [7,8]. However, accurately forecasting future driving conditions poses a significant challenge. A more pressing concern is that optimization algorithms, which typically rely on iterative methods to identify the best solution, are not ideally suited for real-time control in EVs.

Machine learning approaches have garnered attention as a potential solution to address these limitations [9]. Using the Web of Science academic database, we observed a noticeable upward trend in the number of articles related to machine learning and electric vehicle energy management over the past three years: 293 in 2021, 460 in 2022, and 420 in 2023. This trend highlights the growing importance of learning-based EMS in recent years. Especially, learning-based EMS means that a software agent performs a power split without using explicit rules or specific optimization process and instead relying on inferences of machine learning algorithms. A straightforward approach to implementing a learning-based EMS involves using supervised learning with neural networks to approximate the optimal control strategies. For example, Millo et al. [10] utilized extensive vehicle simulation data, informed by DP, to develop a deep neural network (DNN) agent for PHEV. This method, known as imitation learning, relies on training data generated by expert policies or high-quality labeled data derived from optimal EMS solutions like DP [11]. However, collecting high-quality energy management data in real-world scenarios is challenging, and imitation learning often exhibits limited adaptability under unfamiliar conditions.

Reinforcement learning (RL) offers an intelligent and self-learning approach for EMS. Unlike supervised learning, RL determines the optimal actions through trial and error [12]. In this approach, an agent interacts with the driving environment, making decisions based on the current state and receiving feedback. Through iterative learning, the RL agent continuously refines the EMS to maximize objectives [13]. Additionally, deep RL (DRL) is an improvement over traditional RL in that it effectively handles high-dimensional and continuous control challenges through the use of DNNs [14]. This advancement has generated significant interest in applying DRL to energy management problems. Central to DRL-based EMS is the Markov decision process (MDP) [15,16], in which states, rewards, and actions are defined according to the environment model, optimization objectives, and control targets. Actions play a critical role in this framework, with different algorithms being suited to different control challenges. The current research on DRL-based EMS predominantly falls into three categories of algorithms: ① discrete control space: exemplified by methods such as deep Q-networks (DQNs) [17,18] and double DQN (DDQN) [19]. For discrete action DRL algorithms, developing effective action discretization rules is essential [20]. ② Continuous control space: this category includes algorithms such as deep deterministic policy gradient (DDPG) [21], Soft Actor–Critic (SAC) [13,22], twin delayed DDPG (TD3) [23], and proximal policy optimization (PPO) [24]. These algorithms are capable of directly outputting continuous actions, such as engine power and motor torque. ③ Hybrid action space: for complex hybrid vehicles, which require control over both engine power and gear shifts, discrete and continuous control variables can be combined [25,26].

The field of RL remains highly active and has witnessed rapid developments over the past few years. New learning frameworks, such as transfer learning and offline RL, have emerged as part of these advancements. Transfer learning in EMS uses knowledge from pre-trained DRL models to rapidly adapt to novel driving conditions or vehicle types, enhancing efficiency and performance [27,28]. Offline RL involves training an agent using a fixed data set collected beforehand without engaging in interactions with the environment during the training process [29]. This approach is particularly valuable when real-time interaction with the vehicle is not feasible or when safety concerns arise. In our previous study [30], we demonstrated that offline RL can utilize historical data sets to optimize the suboptimal EMS while adhering to safety constraints.

With various learning-based methods in hand, a researcher must determine which one is applicable to EMS. While the number of publications has steadily increased over the last years [20,31], the domain of EMS confronts several pivotal challenges. ① The absence of a standard for unified vehicle models has led researchers to adopt disparate vehicle models. This disparity extends to subsystems such as batteries, engines, and fuel cells, in which modeling precision and optimization objectives diverge. ② The variety of metrics and the lack of standardization in the construction of the agent and environment for learning-based EMS can lead to potentially misleading results. Systematic evaluation and comparison are essential not only to understand the strengths of existing algorithms but also to identify their limitations and suggest future research directions. ③ Recent research in DRL-based EMS often focuses on refining states, actions, and rewards within specific contexts, making it challenging to replicate these improved methods. ④ Deploying a learning-based EMS from simulation environments to real-world scenarios is a complex and systematic engineering task, with few studies focusing on this deployment process. To effectively address these challenges, a unified and open platform for EMS must be established. Further, given the exploration of diverse algorithms in EMS literature, establishing a comprehensive benchmark that incorporates a range of prominent EMS is essential for a fair and unbiased assessment of their performance.

Considering the widespread use of open source platforms such as OpenAI Gym [32] in the field of artificial intelligence, as well as tools such as CARLA (Spain) [33] and Autoware (Japan) [34] in autonomous driving, a comprehensive open source platform in the EV domain is lacking. Here, we present LearningEMS, which stands as a unified EMS framework by open-sourcing algorithms, EV environments, data sets, algorithms, and deployment processes. This holistic approach equips researchers with the tools essential for exploring learning-based EMS for EVs. Fig. 1(a) shows the framework architecture that users can easily modify and redistribute or integrate into their own applications. We have made available three EV simulation environments adhering to the OpenAI Gym standard that can be used to benchmark different learning-based algorithms. The environment offers high-precision models and supports substantial customizability, allowing users to create new environments and add modules to existing environments. Subsequently, a benchmark for learning-based EMS is established, encompassing popular DRL algorithms, as well as recent approaches in imitation learning and offline RL. This facilitates a detailed analysis of energy efficiency, consistency, and adaptability of learning-based EMS. Furthermore, we evaluate and explore the impact of various configuration settings (state, action, and reward function) on performance and characteristics. We also present a policy extraction and reconstruction method tailored for EMS deployment, leading to the deployment of a

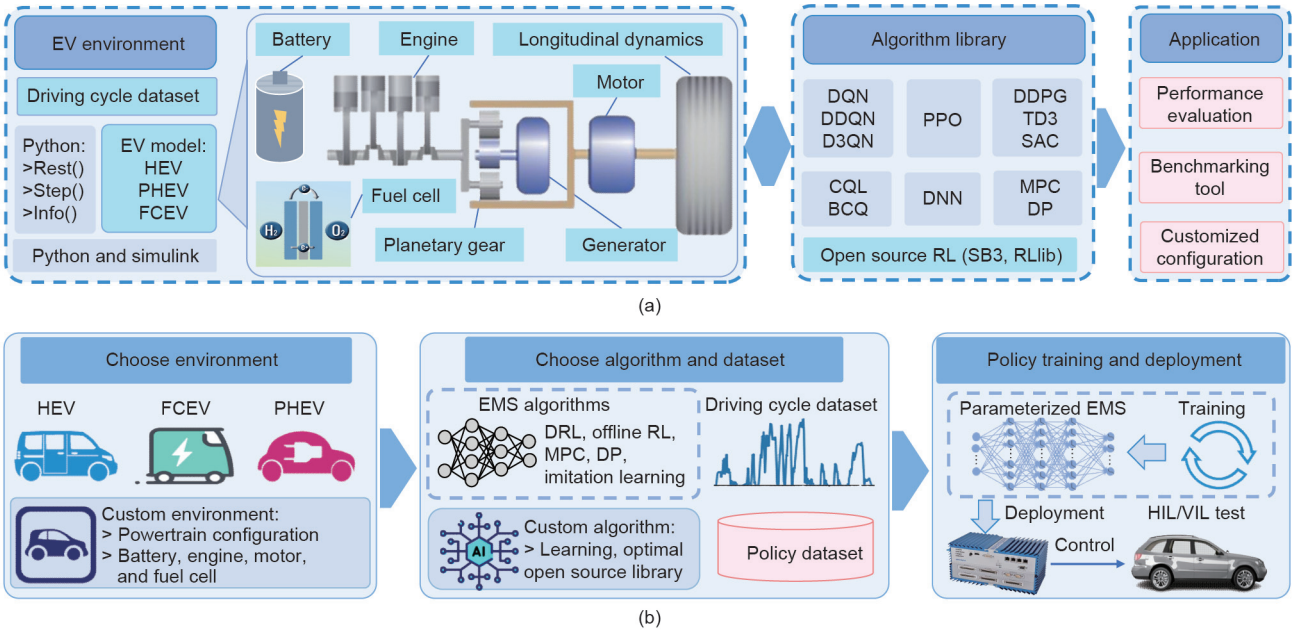

Fig. 1. The learning-based EMS framework and system design of LearningEMS. (a) It consists of three layers: the EV environment layer, the learning-based algorithm layer, and the application layer. D3QN: dueling DDQN; CQL: conservative Q-learning; BCQ: batch-constrained Q-Learning; SB3: Stable-Baselines3; RLlib: RL library. (b) Training pipeline of LearningEMS: First, choose an EV environment, users can create new environments or add modules to existing ones. Then, choose an algorithm and dataset. Finally, start training. After training the policy in simulation, it can be directly deployed into the controller, enabling hardware-in-the-loop (HIL) or vehicle-in-the-loop (VIL) experiments.

parameterized EMS onto a real controller. Notably, by leveraging LearningEMS, we have been able to streamline the process of defining new environments and training policies, enabling rapid and flexible development as illustrated in Fig. 1(b). We encourage researchers to extend the open-source LearningEMS library to advance EMS methods and innovations rapidly.

2. Vehicle models

We aim to establish a comprehensive and unified EV simulation environment, incorporating the fundamental components of EVs, including vehicle dynamics, engine model, battery model, electric motor (EM) model, and fuel cell system. These components, structured as modular entities, have been integrated into the LearningEMS environment, aligning with the OpenAI Gym standard and providing extensive customization capabilities. Users have the flexibility to create new EV environments and enhance existing ones, enabling the modeling of various vehicle types, including HEVs, PHEVs, and FCEVs. Furthermore, the environment facilitates the customization of powertrain configurations, encompassing parallel, series, power-split, and multi-mode configurations.

In this section, we introduce three different EVs: a power-split HEV, a FCEV, and a plug-in series hybrid electric tracked vehicle (SHTV). These models provide comprehensive coverage for passenger, commercial, and specialty vehicles. The powertrains of the three EVs are listed in Fig. 2, and the main parameters of the EVs are listed in Table 1. The power-split HEV [35] belongs to the second generation of the Toyota Prius hybrid system. As shown in Fig. 2(a), the core power-split component is a planetary gear, which is used to realize power coupling among the engine, motor, and generator. In addition, the HEVs are equipped with a small capacity lithium battery used to drive the traction motor and generator. The FCEV is a fuel cell hybrid electric logistics light truck. Fig. 2(b) is a simplified schematic diagram of a FCEV, where a fuel cell stack serves as the primary power source to meet the energy requirements of the vehicle. Specifically, it powers the traction motor, which functions as an EM. The SHTV [17,36] is independently driven by dual motors, and its powertrain configuration is shown in Fig. 2(c). In this structure, the power sources are composed of the engine-generator set and battery pack, where the engine works as a generator to power the EM or to recharge the battery. Note that the SHTV is only used under special conditions. It exhibits significantly different attributes, that is, a large battery pack and stronger lateral force.

2.1. Power request model

In the power request model, the longitudinal dynamics of the vehicle are represented by the longitudinal force (F_r), which primarily comprises four components: the rolling resistance F_f, aerodynamic drag F_w, gradient resistance F_i, and inertial force F_a.

$$\begin{cases} F_r = F_f + F_w + F_i + F_a \\ F_f = G \cdot f \\ F_w = \frac{1}{2}\rho \cdot A_f \cdot C_D \cdot v^2 \\ F_i = G \cdot i \\ F_a = m \cdot \text{acc} \end{cases} \tag{1}$$

where G is the gravity force of the vehicle, f is the rolling resistance coefficient, ρ is the air density, A_f is the frontal area, C_D is the coefficient of air resistance, v is the longitudinal velocity without respect to wind speed, i is the road slope (road slope is not considered in this paper), m is the vehicle mass, and acc is the acceleration.

The power request P_{req} of wheeled vehicles is calculated by the production of longitudinal resistance F_r and the longitudinal velocity v.

$$\begin{cases} P_{req} = F_r \cdot \bar{v} + M_r \cdot \omega \\ \bar{v} = \frac{v_1 + v_2}{2} \\ \omega = \frac{v_1 - v_2}{B} \end{cases} \tag{2}$$

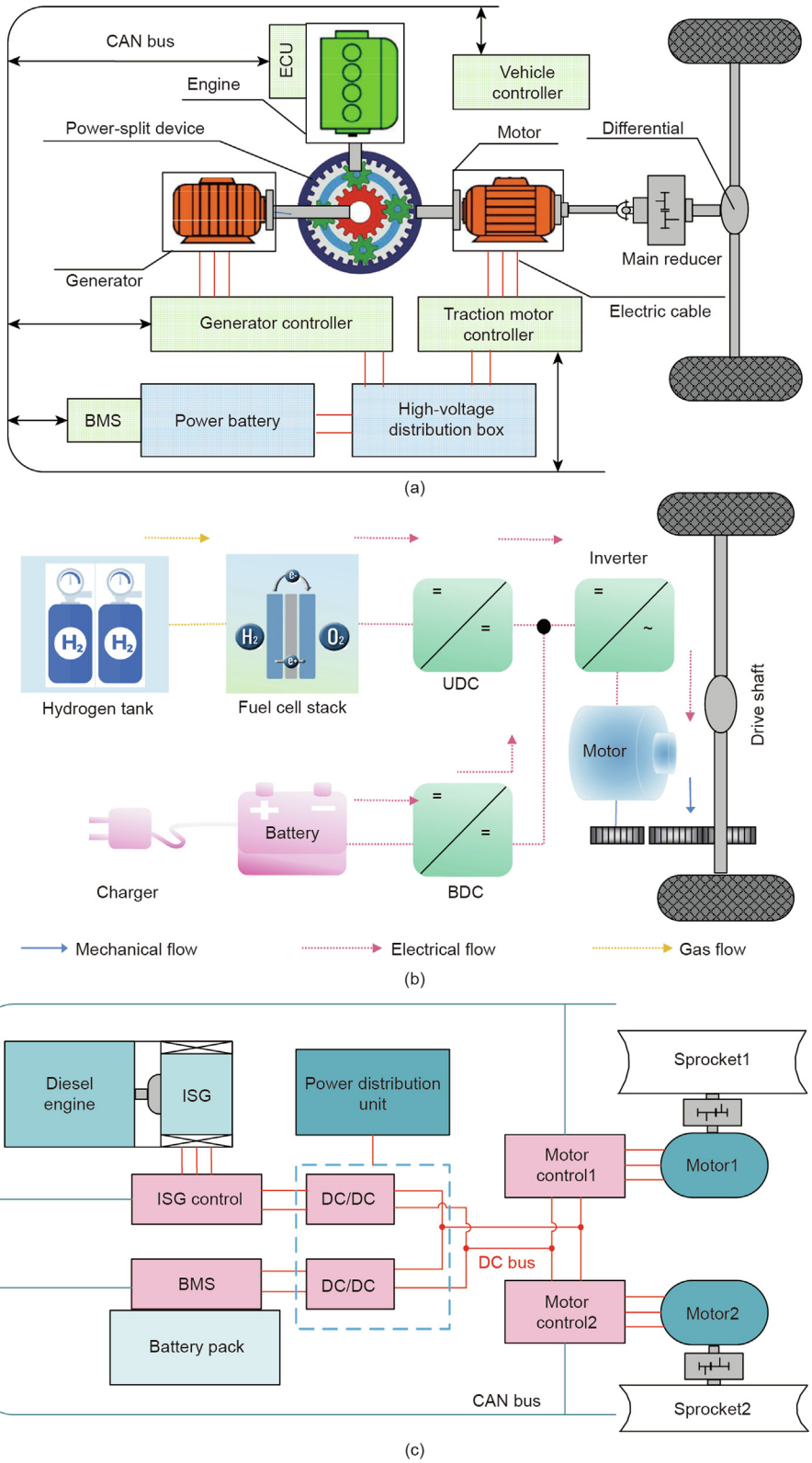

Fig. 2. Powertrain architecture. (a) The power-split hybrid electric vehicle. CAN: controller area network; ECU: electronic control unit; BMS: battery management system. (b) Fuel cell hybrid electric truck. DC: direct current; UDC: unidirectional DC/DC converter; BDC: bidirectional DC/DC converter. (c) SHTV. ISG: integrated starter generator.

where $\bar{v}$ is the center velocity of gravity of the tracked vehicle, M_r is the moment of the turning resistance, ω is the angular speed, v_1 and v_2 are the velocity of two tracks, and B is the vehicle tread. The moment of turning resistance M_r is calculated as follows:

$$\begin{cases} M_r = \frac{\mu \cdot G \cdot L}{4} \\ R_s = R_{s,\max}\left(0.925 + 0.15\frac{R_a}{B}\right)^{-1} \\ R_a = \frac{\bar{v}}{|\omega|} \end{cases} \tag{3}$$

Table 1
Main parameters of EVs.

Components	Parameters	HEV	SHTV	FCEV
Vehicle	Curb weight (kg)	1449	6200	4500
	Air resistance coefficient	0.26	0.40	0.45
	Frontal area (m^2)	2.23	4.30	6.10
	Drive wheel radius (m)	0.287	0.211	0.391
	Ground contact length (m)	N/A	2.48	N/A
	Vehicle tread (m)	N/A	1.95	N/A
Traction motor	Maximum power (kW)	50	110	120
	Maximum torque (N•m)	400	800	360
Generator	Maximum power (kW)	37.8	130.0	N/A
	Maximum torque (N•m)	75	870	N/A
	Maximum power (kW)	56	150	100
Engine/fuel cell	Maximum torque (N•m)	120	720	N/A
	Cell number	N/A	N/A	360
Battery	Capacity (kW•h)	1.54	60	30
	Voltage (V)	237	540	628
Transmission	Final drive ratio	3.930	6.745	N/A
	Characteristic parameter	2.6	N/A	N/A

N/A: not available.

where R_s and $R_{s,max}$ are the empirical-based lateral resistance coefficient and its maximum value, respectively, L is the length of the track on ground, and R_a is the turning radius.

2.2. Engine and EM model

Internal combustion engine (ICE) supplies the main power to the HEV and PHEV. The engine model is established to calculate fuel consumption v_{fuel}, which is determined by quasi-static nonlinear maps. The engine fuel rate $\dot{m}_{fuel}$ is determined by the engine torque T_{eng} and speed ω_{eng}.

$$\begin{cases} \dot{m}_{fuel} = f(\omega_{eng}, T_{eng}) \\ v_{fuel} = \int_0^T \dot{m}_{fuel}\, dt \end{cases} \tag{4}$$

where $t \in [0, T]$ is the time horizon of specific driving cycle. T is the total length of the trip. $f(\cdot)$ is the function representing the fuel consumption as a function of engine torque and engine speed. The EM is the electrical power supplier of EVs. The motor can operate in the drive mode or regeneration mode for EMS. The motor efficiency can be modeled as a function of the instantaneous EM torque T_{EM} and angular speed ω_{EM}. Subsequently, the power of the EM (P_{EM}) can be written as follows:

$$\begin{cases} \eta_{EM} = f_{EM}(T_{EM}, \omega_{EM}) \\ P_{EM} = T_{EM} \cdot \omega_{EM} \cdot \eta_{EM}^{\alpha_0} \end{cases} \tag{5}$$

where α_0 equals 1 when the EM works as a generator, and it equals −1 when the EM works as an EM. Here, η_{EM} is the motor efficiency and $f_{EM}(\cdot)$ is the efficiency mapping function.

2.3. Engine and electric EM

The battery pack is modeled using an equivalent circuit model in Eq. (6).

$$\begin{cases} P_{bat}(t) = V_{oc}(t) \cdot I(t) - R_0 \cdot I^2(t) \\ I(t) = \frac{V_{oc}(t) - \sqrt{V_{oc}^2(t) - 4 \cdot R_0 \cdot P_{bat}(t)}}{2R_0} \\ SOC(t) = \frac{Q_0 - \int_0^t I(t) dt}{Q} \end{cases} \tag{6}$$

where SOC is the state of charge, V_{oc} is the open-circuit voltage, $I(t)$ is the current at time t, R_0 is the internal resistance, P_{bat} is the output power in the charge-discharge cycles, Q_0 is the initial battery capacity, and Q is the nominal battery capacity.

Battery aging models can be used to reflect degradation characteristics. While different complexities of aging models can affect the energy management strategies, this study primarily uses a simplified battery aging model. Additionally, our open-source code includes more complex aging models, such as the energy-throughput-based model from our previous research [22,23]. The degradation rate of battery operation γ_{bat} is affected by the charge/discharge rate (C-rate). The relationship between the battery life correction factor and C-rate can be fitted from experiment data:

$$\gamma_{bat} = \mu_1 |C_{rate}|^2 + \mu_2 |C_{rate}| + \mu_3 \tag{7}$$

where $\mu_1, \mu_2,$ and μ_3 are the curve-fitting coefficients, and C_{rate} is the C-rate. Lithium-ion batteries can operate for about 5000 full cycles in a lifetime. The battery degradation cost $C_{bat,\,degr}$ can be calculated as follows:

$$C_{bat,degr} = p_{bat} \cdot \int_0^t \gamma_{bat}^{-1} P_{batdis} dt \tag{8}$$

where p_{bat} is the battery price per kW•h that is 1500 kW•h. P_{batdis} is the battery discharge power.

2.4. Engine and EM model

The efficiency of the fuel cell system under different power conditions can be obtained from experimental data. Thus, the mass flow rate of the hydrogen consumption ($\dot{m}_{H_2}$) can be calculated as follows:

$$\dot{m}_{H_2} = \frac{P_{fcs}}{\eta_{fcs} \cdot LHV_{H_2}} \tag{9}$$

where η_{fcs} is the fuel cell system efficiency, P_{fcs} is the fuel cell system output power, and LHV_{H_2} is the hydrogen low calorific value. The fuel cell hydrogen cost can be calculated as follows:

$$C_{fcs,H_2} = p_{H_2} \cdot \int_0^t \dot{m}_{H_2} dt \tag{10}$$

where p_{H_2} is the hydrogen price per kilogram (60 RMB·kg^{-1}).

The fuel cell degrades rapidly under four typical conditions including load changing, start/stop, low power, and high power condition [37]. In this study, the fuel cell system is set to continue operating until the vehicle power system is shut down. Thus, the start/stop condition is not considered in the EMS design. The fuel cell voltage degradation rate γ_{fcs} can be calculated as follows:

$$\gamma_{fcs} = \kappa_{low} \cdot T_{low} + \kappa_{high} \cdot T_{high} + \kappa_{cha} \cdot \Delta P_{fcs} \tag{11}$$

where κ_{low} is the degradation rate under low power condition, T_{low} is the low power condition duration, κ_{high} is the degradation rate under high power condition, T_{high} is the high power condition duration, κ_{cha} is the degradation rate under load changing condition, and ΔP_{fcs} is the fuel cell power slope.

The fuel cell is considered to reach the end of its life when 10% of the voltage at rated power is lost. The fuel cell operation degradation cost ($C_{fcs,degr}$) can be calculated as follows:

$$C_{fcs,degr} = k_{fcs} \cdot \gamma_{fcs} \cdot P_{fcs,rate} \cdot \frac{p_{fcs}}{V_{fcs,end} \cdot 1000} \tag{12}$$

where k_{fcs} is the fuel cell life correction factor, $V_{fcs,end}$ is fuel cell voltage drop at the end-of-life, $P_{fcs,rate}$ is the rated power of the fuel cell, and p_{fcs} is the fuel cell price per kilowatt (4000 RMB· kW^{-1}).

3. Benchmark EMS methods

In this section, we provide a brief introduction to the algorithms implemented in our benchmark and categorize them as follows: ① optimization control, ② DRL, ③ imitation learning, and ④ offline RL. The key attributes of these algorithms are presented in Table 2. DP uses the ground truth of the future driving cycle and serves as the global optimal EMS. MPC predicts the future 10 s driving cycle and employs an optimization algorithm such as DP to optimize the energy consumption. DNN learns the optimal action from DP. Deep Q-learning (DQL), deep double Q-learning (D3QL), DDPG, TD3, and SAC are all off-policy DRL algorithms, differing mainly in their action implementations. For on-policy DRL, the PPO is considered. Additionally, the offline RL algorithm, conservative Q-learning (CQL), is also selected as a baseline for our study.

3.1. Energy management objective

In this study, energy management of electric vehicles is modeled as a long-term sequential decision process objective to minimize the total energy consumption while maintaining battery SOC within reasonable limits. The optimization objective can be formulated (J_{EMS}) as follows:

$$J_{EMS} = \min \sum_{t=0}^{T} \text{cost}(t) + \alpha f_s(\text{SOC}(t)) \tag{13}$$

where cost(t) is the energy consumption including fuel consumption, electricity consumption, and hydrogen consumption, $f_s(\text{SOC}(t))$ is the SOC maintaining function, and α is the trade-off between energy consumption and SOC maintaining.

3.2. Optimization control methods

In optimization control methods, the EMS of the EV is framed as a nonlinearly constrained optimization formulation minimizing the objective function given in Eq. (13). The output of the control methods during optimization should be constrained to ensure allowable operations of key components.

DP seeks the shortest path backward in time. That is, it obtains the minimum cost function of every grid at every stage backward in time. It utilizes future driving cycles, which are not available in practice. In this study, DP acts as the global optimum and provides upper limits. The global optimization algorithm requires future information as input. Although we cannot obtain the true future, we can predict it. MPC utilizes prediction algorithms to forecast the future driving cycle and minimizes a series of cost functions over the prediction horizon. With the appropriate prediction method, the MPC-based strategy can achieve performance similar to DP under specific working conditions.

Table 2
Benchmark energy management methods.

Algorithm	Method	Future information	Action format
DP	Optimization	✓	Discrete
MPC	Optimization	✓	Discrete
DNN	Imitation learning	X	Continuous
DQN	Off policy DRL	X	Discrete
D3QN	Off policy DRL	X	Discrete
DDPG	Off policy DRL	X	Continuous
TD3	Off policy DRL	X	Continuous
SAC	Off policy DRL	X	Continuous
PPO	On policy RL	X	Continuous
CQL	Offline DRL	X	Continuous

D3QN: dueling DDQN; ✓: requires predicted information; X: no need for predicted driving condition.

3.3. Formulating energy management problem as MDP

RL is the closest machine learning paradigms to human learning. For RL-based EMS, the energy management problem is formulated as a MDP, which is a framework for learning the optimal EMS policy from interaction to minimize the total energy consumption, as shown in Fig. 3(a). An MDP consists of:

(1) A set S of states: State is used to describe the condition of the agents at a specific time-step in the environment. A typical state at time point t for the HEV is defined as $s_{HEV} = \{v_t, \text{acc}_t, \text{SOC}_t\}$. where v_t, acc_t, and SOC_t are the speed, acceleration, and battery SOC, respectively. For the SHTV, the state is defined as the variables related to SHTV itself, $s_{SHTV} = \{v_t, \text{acc}_t, w_t, wa_t, \text{SOC}_t\}$. The variables w_t and wa_t are vehicle yaw rate and lateral acceleration, respectively. For the FCEV, the state space includes vehicle speed, acceleration, fuel cell power P_{fcs}, and battery SOC, $s_{FCEV} = \{v_t, \text{acc}_t, \text{SOC}_t, P^t_{fcs}\}$.

(2) A set of action A: The action represents the control variable and allocates power to the energy sources of the vehicle. In the case of HEV, we consider the engine's output power as the action. For the SHTV, the actions are chosen as the engine torque and speed because power is allocated through the torque and rational speed of its energy converters. In the context of the FCEV. The action space (a) can be described as follows:

$$a = \left\{\Delta P_{fcs} = P^t_{fcs} - P^{t-1}_{fcs}, \Delta P_{fcs} \in [-10\text{ kW}, 10\text{ kW}]\right\} \tag{14}$$

(3) A reward function R: The reward function $R(s_{t+1}; s_t; a_t)$ is associated with entering state s_{t+1} from state s_t using action a_t. The reward of the HEV is defined as the sum cost of the fuel consumption while maintaining charge-sustaining constraints. Because the battery capacity of the HEV is relatively small, we expect the SOC to remain in a range and maintain the battery SOC. Reward (r) can be defined as the instantaneous feedback (r_{HEV}) in Eq. (15):

$$r_{HEV} = -\left\{v_{fuel} + \omega_1[\text{SOC}_{ref} - \text{SOC}(t)]^2\right\} \tag{15}$$

where v_{fuel} is the fuel consumption and ω_1 is a positive weight coefficient (presumed as 300 in this manuscript). SOC_{ref} is the desired reference SOC. The reward of SHTV (r_{SHTV}) is defined as the sum cost of the fuel and electricity consumption while satisfying the power demand requirement as follows:

$$r_{SHTV} = -\left\{v_{fuel} \cdot \text{Price}_{fuel} + Q_b \cdot \Delta\text{SOC} \cdot \text{Price}_{electric} + \omega_2[\text{SOC}_{ref} - \text{SOC}(t)]^2\right\} \tag{16}$$

where Price_{fuel} is the fuel price, which is 6.5 RMB·L^{-1}, and $\text{Price}_{electric}$ is the electric price, which is 0.97 RMB·(kW·h)$^{-1}$. Q_b is the battery capacity (Ah). ω_2 is the weight coefficient of the comprehensive cost and battery SOC. For the FCEV, the fuel cell hydrogen cost and operation degradation cost are minimized, and the battery

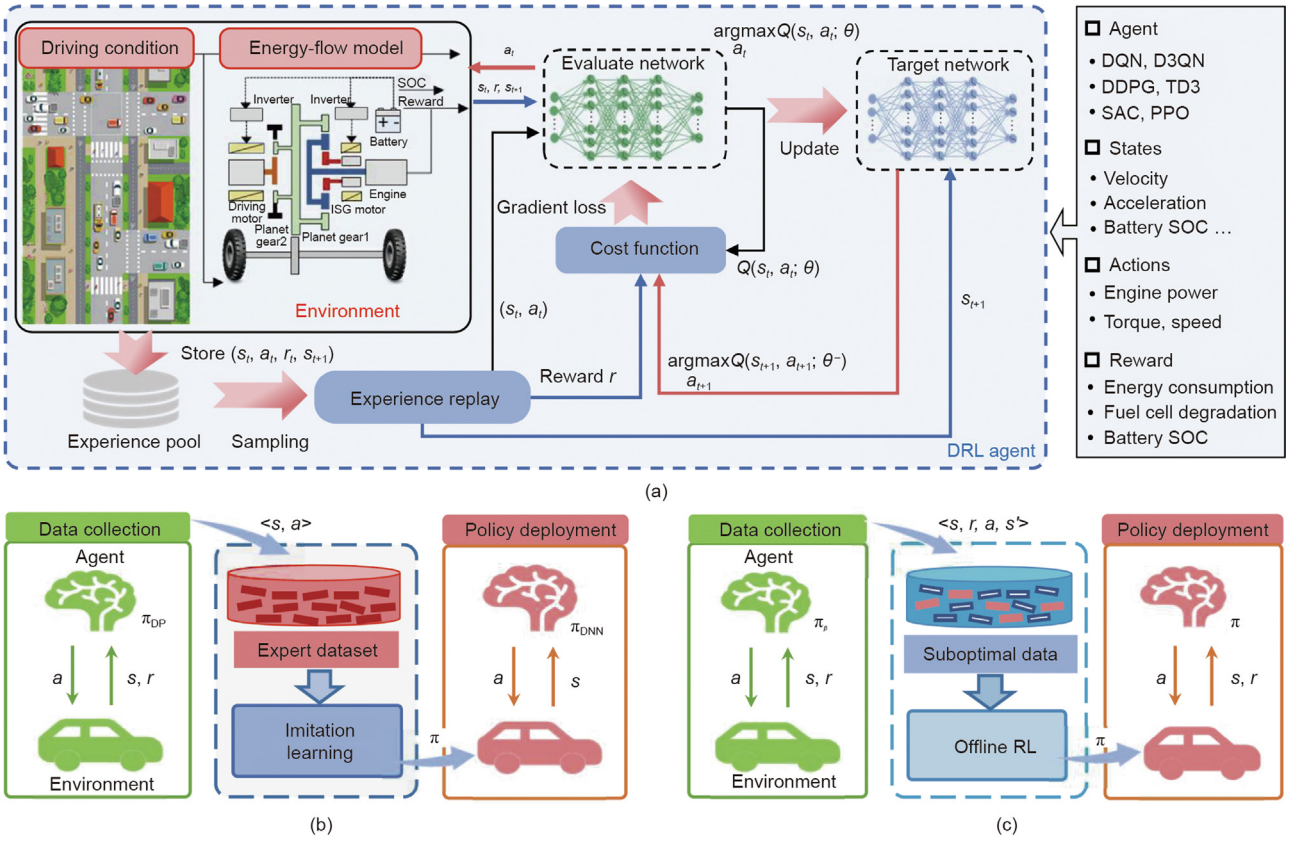

Fig. 3. Overall schematic of learning-based EMS. (a) The structure of the DRL energy management framework. θ is the neural network parameter, and θ^- is the updated neural network parameter. (b) Imitation learning for EMS. π_{DP} is the expert policy generated by DP algorithm, π_{DNN} is the policy learned by the DNN, and π_β is the behavioral policy from offline data. (c) Offline RL for EMS.

SOC should be maintained. Thus, the reward function (r_{FCEV}) is written as follows:

$$r_{FCEV} = -\left\{C_{fcs,H_2} + C_{fcs,degr} + \omega_3[SOC_{ref} - SOC(t)]^2\right\} \quad (17)$$

where ω_3 is the weight coefficient.

After executing a power allocation a_t with observation s_t, the RL agent receives feedback r_t at each time step t regarding the performance of its action. Using this feedback, it iteratively updates its action policy to reach an optimum control policy π that maximizes the expected cumulative reward R_t:

$$R_t = \sum_{i=0}^{T-1} \gamma^i r_{t+i} \quad (18)$$

with the discount parameter γ that reflects the importance of future rewards. i denotes the time step of a driving condition.

(4) A transition probability function P: this assigns a probability distribution that represents the probability of entering a state from state S using action A.

3.4. Formulating energy management problem as MDP

3.4.1. Discrete action space

DQN [18]: The DQN algorithm represents the first successful integration of the classical Q-learning framework with DNNs. According to the original paper, two tricks are essential for the algorithm to achieve human-level performance. First, DQN uses a DNN called the target network to represent the action value function, the target network can be learned by updating the parameters θ to minimize the error in the Bellman equation. By approximating the Q-function $Q(s_i, a_i; \theta) \approx Q^*(s_i, a_i)$ at iteration i, we can compute the target values $Y = r + \gamma \max_{a'} Q(s', a'; \theta_i)$, and update θ for the network using the following loss $L(\theta)^{DQN}$function:

$$L(\theta)^{DQN} = \frac{1}{n}\sum_{i=1}^{n}[Y - Q_\theta(s, a; \theta_i)]^2 \quad (19)$$

where s' and a' are the state and action at the next time-step, respectively, $n \in \mathbf{N}^+$. Q_θ is Q-function with parameters θ. $\max_{a'}$ stands for maximum. Another trick is to use the experience replay mechanism. At each time-step t, the agent's experiences (s_t, a_t, r_t, s_{t+1}) are stored in the replay memory. The experience replay mechanism aims to obtain uncorrelated data samples, thereby reducing the error of gradient estimation for the stochastic optimization problem.

Dueling DDQN (D3QN) [19]: The D3QN algorithm is an advanced version of the DDQN algorithm [38]. DDQN can be regarded as an improved DQN that tends to overestimate the action values. In DDQN, two networks can be parameterized by two sets of weights θ and θ'. The first network is known as the online network, which is used to choose the action from the maximum Q-value. In addition, the target network is used to estimate the Q-value that remains the same as the DQN. The target function in DDQN is given as follows:

$$Y^{DDQN} = r + \gamma Q\left(s', \text{argmax}_{a'} Q(s', a'; \theta_i'); \theta_i\right) \quad (20)$$

D3QN seamlessly integrates the benefits of both the dueling architecture and the double Q-learning approach. By adopting the dueling architecture, it can separately model state values and action advantages, thereby providing a more effective

representation of the value function. The incorporation of double Q-learning enhances the accuracy of Q-value estimation and alleviates any concerns regarding overestimation. Note that in both DQN and D3QN, the most representative DRL algorithms, the state space is continuous but the action space is discrete. In EMS, the control variables are typically continuous. Discretizing these variables can introduce errors, leading to performance degradation due to the loss of precision.

3.4.2. Continuous action space

DDPG [21]: The DDPG algorithm is proposed to overcome the challenges presented by continuous action spaces in RL. The DDPG algorithm uses two neural networks: one for learning the Q-function (the critic network) and the other for learning the policy (the actor network). For the actor part, it learns a deterministic target policy by mapping states to a specific action. The critic part is used to estimate the Q-value of a state-action pair. Here, θ^{μ} denotes the weights of the actor network, and θ^{Q} the weights of the critic network. The critic is updated by minimizing the loss function as follows:

$$L(\theta)^{\mathrm{DDPG}} = \frac{1}{n}\sum_{i=1}^{n}\left[Y^{\mathrm{DDPG}} - Q\left(s,a;\theta^{Q}\right)\right]^{2} \tag{21}$$

where $Y^{\mathrm{DDPG}} = r + \gamma Q\left(s', \mu(s';\theta^{\mu});\theta^{Q}\right)$. The actor is updated by maximizing the expected return $J^{\theta^{\mu}}$ using a sampled policy gradient:

$$\begin{aligned}\nabla_{\theta^{\mu}} J^{\theta^{\mu}} &\approx \mathbf{E}\left[\nabla_{\theta^{\mu}} Q(s,\mu \mid (s;\theta^{\mu});\theta^{\mu})\right] \\ &= \mathbf{E}\left[\nabla_{\mu(s)} Q\left(s,\mu(s);\theta^{Q}\right)\nabla_{\theta^{\mu}}\mu(s;\theta^{\mu})\right]\end{aligned} \tag{22}$$

where $\nabla_{\theta^{\mu}}$ is the gradient, $\mathbf{E}[\cdot]$ denotes the expectation, and $\mu(s)$ is the action taken by the actor for state s.

TD3 [39]: The TD3 algorithm is an advanced algorithm built on the foundation of the DDPG. It was specifically designed to address the limitations associated with approximation errors in DDPG with the aim to enhance the overall stability and performance of the algorithm. TD3 includes the techniques of continuous double Q-learning, policy gradient, and actor-critic architecture. The difference in structure between DDPG and TD3 is that TD3 has two critic networks that contain two online networks (Q_1, Q_2) and two target networks (Q_1', Q_2').

SAC [40]: The SAC algorithm is a state-of-the-art approach in RL, which was designed for effectively handling complex, high-dimensional environments. Distinguished by its incorporation of entropy regularization, SAC optimizes a stochastic policy such that it balances exploration and exploitation. This is achieved by maximizing not only the expected return but also the entropy of the policy, leading to more robust and efficient learning. Compared with conventional DRL, SAC has an entropy term besides the reward. Thus, the policy aims to maximize the discounted entropy terms while maximizing the expected sum of rewards.

$$J(\pi) = \sum_{t=0}^{T-1}\mathbb{E}_{(s_t,a_t)\sim\rho_{\pi}}\left[r(s_t,a_t) + \alpha\mathcal{H}^{\pi}(\cdot|s_t)\right] \tag{23}$$

where, $\mathbb{E}_{(s_t,a_t)\sim\rho_{\pi}}$ is expected value over state-action pairs, sampled according to the policy distribution ρ_{π}, α denotes the trade-off coefficient that controls the switches between the expected reward and entropy term, while $\alpha \geq 0$ is the temperature parameter. The entropy $\mathcal{H}^{\pi}(\cdot|s_t)$ of policy π at state s_t is computed from its distribution as follows.

$$\mathcal{H}^{\pi}(\cdot|s_t) = -\sum_{a_t}\pi(a_t|s_t)\log\pi(a_t|s_t) \tag{24}$$

SAC concurrently learns three functions including the soft value function $V_{\psi}(s_t)$, soft Q-function $Q_{\theta}(s_t,a_t)$, and policy function $\pi_{\phi}(a_t|s_t)$. The parameters of these networks are ψ, θ, and ϕ. The soft value function $(J_V(\psi))$ is trained to minimize the squared residual error as follows:

$$J_V(\psi) = \mathbb{E}_{s_t\sim D}\left[\frac{1}{2}\left(V_{\psi}(s_t) - \mathbb{E}\left[Q_{\theta}(s_t,a_t) - \log\pi_{\phi}(a_t|s_t)\right]\right)^{2}\right] \tag{25}$$

where D is the distribution of states. The loss function $(J_Q(\theta))$ for the soft Q-function is given as follows:

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim D}\left[\frac{1}{2}\left(Q_{\theta}(s_t,a_t) - \widehat{Q}(s_t,a_t)\right)^{2}\right] \tag{26}$$

where $\widehat{Q}(s_t,a_t) = r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim p}\left[V_{\bar{\psi}}(s_{t+1})\right]$, $\bar{\psi}$ is the average of the network weights. The policy is optimized by minimizing the expected Kullback–Leibler (KL) divergence (D_{KL}):

$$J_{\pi}(\phi) = \mathbb{E}_{s_t\sim D}\left[D_{\mathrm{KL}}\left(\pi_{\phi}(\cdot|s_t)\|\frac{\exp(Q_{\theta}(s_t,\cdot))}{Z_{\theta}(s_t)}\right)\right] \tag{27}$$

where $Z_{\theta}(\cdot)$ is normalization term.

PPO [41]: The primary objective of policy gradients is to iteratively approach the optimal policy that yields the highest return. Unlike off-policy algorithms, in on-policy algorithms, all policy updates are based on data collected from the trajectory distribution induced by the current policy of the agent. These algorithms lack a memory storing past experiences and only process each data point once. Consequently, consistently enhancing the policy using the available data becomes essential without suffering from performance degradation. PPO introduces the concept that the agent performs not just one gradient update per data sample, but multiple epochs of minibatch gradient updates. Moreover, PPO incorporates a clipping objective in the policy update to prevent the updated policy from excessively deviating from the prior one. For training stability, the divergence between the old and updated policies in the main objective L^{CLIP} is limited by clipping the probability ratio $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{\mathrm{old}}}(a_t|s_t)}$ ($r_t(\theta)$*is* probability ratio, $\pi_{\theta_{\mathrm{old}}}$ is the old policy) at the interval $[1-\epsilon, 1+\epsilon]$, where ϵ is the hyperparameter defining the clipping range. Under the actor-critic framework, the loss function comprises three parts: the clipped surrogate objective L^{CLIP}, a value function error term L^{VF}, and an entropy bonus, where the entropy bonus is used to ensure sufficient exploration. Combining these terms, the following objective function is maximized per iteration:

$$\begin{cases} L^{\mathrm{CLIP}}(\theta) = \widehat{E}_t\left[\min\left(r_t(\theta)\widehat{A}_t,\ \mathrm{clip}(r_t(\theta),1-\epsilon,1+\epsilon)\widehat{A}_t\right)\right] \\ L_t^{\mathrm{CLIP+VF+\hat{S}}}(\theta) = \widehat{E}_t\left[L_t^{\mathrm{CLIP}}(\theta) - c_1 L_t^{\mathrm{VF}}(\theta) + c_2\widehat{S}[\pi_{\theta}](s_t)\right] \end{cases} \tag{28}$$

where c_1 and c_2 are coefficients, $\widehat{S}$ denotes an entropy bonus, and L_t^{VF} denotes a squared-error loss. $\widehat{E}_t$ is the empirical expectation over a batch of sampled data points, and $\widehat{A}_t$ is the estimated advantage function at timestep t.

3.5. Imitation learning

Imitation learning assumes that the data are generated by an expert or near-expert policy. Its goal is to mimic the behavior of the expert as closely as possible. Imitation learning methods often employ supervised learning techniques, such as behavioral cloning, to directly map states to actions [42]. Imitation learning can achieve good performance when the data are high-quality and cover most of the relevant situations, but it can suffer from compounding errors and distributional shift when the data are

noisy or incomplete. This can be accomplished through supervised learning techniques, which is the most common form of machine learning. In virtual or real-world problems, supervised training is conducted using ground truth data provided by an instructor or teacher who shows the machine learning system what to do. Building upon this foundation, the agent leverages DNNs to analyze input data and acquire the ability to generate a sequence of outputs that gradually converge toward the intended target or mentioned label. In EMS, global-optimization approaches such as DP act as the instructor, which is used to generate the optimal energy management trajectories. In Fig. 3(b), the overarching framework of imitation learning for EMS is illustrated. The learning objective of imitation learning is to mimic the behavior of DP. The learning objective can be formulated as follows:

$$\min_{\theta} \sum_{t=0}^{T} \text{error}(\pi_{\text{DP}}(t) - \pi_{\theta}(s_t)) \tag{29}$$

where $\pi_{\text{DP}}(t)$ is the optimal action at time point t produced by the DP algorithm, and s_t denotes some state factors such as acceleration, speed, battery SOC, trip length, and trip duration.

3.6. Offline RL algorithm

Offline RL is an innovative subset of RL methods rooted in data-driven approaches. It maximizes the utility of expert insights and pre-existing data sets to refine policy training. Fig. 3(c) illustrates the overarching framework of offline RL for EMS. In offline RL, we have an available data set (D) collected under a starting behavioral policy (π_{β}). The key concept is to utilize this established data set to create an improved decision policy (π_{off}), without interacting with the environment. Offline RL extends beyond basic imitation learning, with the aim to extract and leverage the strengths of the initial behavioral policy. It stands as a valuable approach, especially in fields where obtaining fresh data via online interactions is unfeasible or risky. Under offline RL settings, given a static data set of transitions $\mathcal{D} = \left\{(s_t, a_t, s_{t+1}, r_t)_j\right\}$, where j indexes a transition in the data set, the actions come from the behavior policy $a_t \sim \pi_{\beta}(\cdot|s_t)$, the states come from a distribution induced by the behavior policy $s_t \sim d^{\pi_{\beta}}(\cdot)$, the next state is determined by the transition dynamics $s_{t+1} \sim T(\cdot|s_t, a_t)$, and the reward is a function of state and action $r_t = r(s_t, a_t)$. In offline RL, the objective remains the same as in the online case: to find a policy that maximizes the expected return. However, it is important to realize that evaluating this objective using any trajectory distribution $(p_{\pi}(\tau))$ is challenging. This limitation arises from the potential for policy π to encounter states not covered in our static data set due to the distributional shifts [43].

CQL: Kumar et al. [44] proposed CQL, which aims to address these limitations by learning a conservative Q-function such that the expected value of the policy under this Q-function lower-bounds is its true value. The core idea of CQL is to introduce a regularization term to the Bellman error objective, which encourages Q-values to remain stable within a certain radius. CQL introduces a component that maximizes the entropy within the Bellman error objective. This leads to higher entropy values around Q-values close to the current state and action. This, in turn, encourages a more balanced distribution of Q-values across the entire state-action space, preventing overly confident estimates. In addition to the Bellman error, CQL ensures stability through the minimization of a regularization term. This term is characterized by the KL divergence between Q-values and the maximized entropy term. By minimizing this term, CQL ensures that the learned Q-values do not deviate beyond a predetermined radius, thereby establishing a lower bound on the actual Q-function. The general form of the loss function in the CQL algorithm is as follows:

$$\min_{Q} \max_{\mu} \alpha\left(\mathbb{E}_{s\sim\mathcal{D},a\sim\mu(a|s)}[Q(s,a)] - \mathbb{E}_{s\sim\mathcal{D},a\sim\widehat{\pi}_{\beta}(a|s)}[Q(s,a)]\right) + \frac{1}{2}\mathbb{E}_{s,a,s'\sim\mathcal{D}}\left[\left(Q(s,a) - \widehat{B}^{\pi_k}\widehat{Q}^{k}(s,a)\right)^2\right] + \mathcal{R}(\mu) \tag{30}$$

where μ is a policy that visits the unseen actions in data set $\mathcal{D}$, $\widehat{\pi}_{\beta}(\cdot)$ is an estimate of the behavior policy π_{β}, $\widehat{B}^{\pi_k}$ is the empirical Bellman operator, $\widehat{Q}^{k}$ is the resulting Q-function, and $\mathcal{R}(\mu)$ is the regularization term for the policy $\mu(a|s)$.

4. Benchmark test results

This section first presents details regarding the experimental setting. Subsequently, several evaluation metrics are defined. Finally, the performances of different algorithms are compared with each other in terms of energy performance, performance consistency and adaptability, and energy efficiency analysis.

4.1. Experimental setting

In this benchmark study, the simulated environment closely resembles an OpenAI Gym-like environment and shares functional structure. For each EV environment, two versions of the environment are provided, differing only in the action space: one featuring a discrete action space and the other with a continuous action space. Furthermore, tuned hyperparameters play an important role in eliciting the best results from all algorithms. To achieve fair comparisons, we used the same hyperparameters of algorithms for each environment, as detailed in Table 3. Importantly, the hyperparameters for all DRL algorithms are consistent, and they are openly accessible in the LearningEMS library.

Our code follows stringent standardization practices, facilitating enhanced readability and simplified hyperparameter tuning processes. To train and validate the learning-based EMS, we leveraged an extensive range of driving conditions, encompassing only standard driving cycles but also locally derived conditions based on real vehicle operational data. The data set comprises accumulated operational data spanning more than 10 000 km, capturing various driving scenarios such as urban, suburban, and highway environments. Notably, a subset of these conditions was selectively employed to validate the final algorithm. The driving cycles and parameters for validation are listed in Table 4, with D denoting distance, v_{mean} the average speed, $v_{\max}$ the maximum speed, and $a_{\max}$ the maximum acceleration. For the HEV, validation encompasses standard driving cycles NEDC and WLTC, along with the inclusion of the Guiyang city driving cycle (GCDC) [35] derived from real EVs. The FCEV validation data set includes standard driving cycles WTVC, CHTC, and FTP75. In the case of SHTV, the validation is executed using locally derived scenarios designated as driving cycle 1 (DC-1) and driving cycle 2 (DC-2). The driving data for SHTV encompasses not only speed and acceleration but also yaw rate

Table 3
Hyperparameter configurations of different algorithms.

Hyperparameter	DRL	Offline RL	DNN
Batch size	128	128	256
Discounted factor	0.99	0.99	—
Learning rate of actor	0.0001	0.0003	0.001
Learning rate of critic	0.001	0.003	—
Hidden dimension of actor	256/128	128	128
Hidden dimension of critic	256/128	128	—
Optimizer	Adam	Adam	Adam

Table 4
Parameters of driving cycles.

Driving cycle	D (km)	T (s)	v_{mean} ($m \cdot s^{-1}$)	v_{max} ($m \cdot s^{-1}$)	a_{max} ($m \cdot s^{-1}$)
NEDC	11.00	1180	9.5	33.3	1.0
WLTC	23.25	1800	12.9	36.5	1.7
GCDC	10.80	1230	8.7	21.0	2.1
DC-1	14.00	1531	9.1	15.8	2.1
DC-2	16.00	1721	9.3	15.8	0.8
WTVC	23.27	1800	11.4	24.4	1.0
CHTC	15.88	1653	9.6	27.0	1.3
FTP75	17.77	1854	9.6	25.4	1.5

v_{mean}: average speed of driving cycles; NEDC: new European driving cycle; WLTC: worldwide harmonized light vehicles test cycle; WTVC: world transient vehicle cycle; CHTC: China heavy-duty commercial vehicle test cycle; FTP75: Federal test procedure.

and lateral acceleration, reflecting the unique characteristics of tracked vehicles.

4.2. Evaluation metrics

To evaluate the performances of these algorithms in energy management tasks, several evaluation metrics were particularly defined:

(1) Energy efficiency performance: The overall fuel and electricity consumption of EVs was calculated in a specific driving cycle. For comparison purposes, the energy consumption of DP served as the global optimum. The energy efficiency performance of each algorithm was assessed by comparing its total cost with that of the DP, providing a direct metric to evaluate the performance of the EMS methods.

(2) Performance consistency: To evaluate the consistency of the EMS algorithms to different conditions, consistency was introduced as follows:

$$\begin{cases} \mu_u &= \sum_{i=k}^{k+N} \left((f_{eco})_i \right) / N \\ \sigma &= \sqrt{\frac{1}{N} \sum_{i=k}^{k+N} \left((f_{eco})_i - \mu_u \right)^2} \end{cases} \tag{31}$$

where f_{eco} represents the energy economy, k is the convergence step, N is the number of consecutive time steps, i represents each individual time step in the range from k to $k+N$, and μ_u and σ represent the mean and standard deviation of the energy economy, respectively.

(3) Performance adaptability: This refers to the ability of learning-based EMS to adapt and perform effectively in novel, unseen conditions, a critical aspect that is often overlooked in existing studies. In this context, algorithm adaptability is achieved through the implementation of two distinct training methodologies: single-condition training (SCT) and multi-condition training (MCT). SCT is performed and tested under the same specific driving cycles. However, MCT involves training an EMS strategy in a diverse set of driving cycles and testing its performance under previously unseen conditions.

4.3. Performance and consistency by comparative evaluation

In this section, a SCT approach was employed. Each algorithm undergoes separate training for each validation scenario, enabling a comprehensive evaluation of both energy efficiency performance and algorithmic consistency. Fig. 4(a) illustrates the results of all algorithms under different conditions and EV environments, whereas more detailed results can be found in Tables S1–S3 in Appendix A. DNN and MPC exhibit optimal performance, with continuous action space DRL algorithms PPO, CQL, TD3, DDPG, and SAC closely following in terms of energy consumption. However, PPO exhibits poor robustness. DQN and DDQN with discrete action space demonstrate comparatively lower energy efficiency. A detailed analysis of each EMS method is given below:

DNN: In our experiments, the DNN is trained with label data from the global optimization algorithm, with the aim of approximating the strategy of DP. DNN achieves outstanding results on all three EV environments, with DP performances of 96.7%, 98.1%, and 95.1% (Figs. 4(b)–(d)). Notably, it secures the highest rank in the HEV and FCEV environments. This indicates that the DNN, utilizing the same data set, effectively replicates the strategy of DP. However, notably, due to the SCT, the DNN model displays indications of overfitting, denoted as overfitting DNN (O-DNN). Further validation is required to verify the performance under new conditions.

MPC: MPC is a common and effective EMS method. Under experimental settings, we assume that the MPC algorithm exhibits perfect predictive accuracy, foreseeing the future driving conditions with 100% precision for the upcoming 10 s. This variant of the MPC is labeled perfect MPC (P-MPC). Thus, it can achieve the best energy economy for SHTV models, with comprehensive performance reaching 95.7% of DP (Fig. 4(d)). It also secured the second rank for HEV with a performance level of 95.2% compared to DP (Fig. 4(b)). The accurate prediction contributes to the robustness of MPC because it has demonstrated excellent performance across various conditions and vehicles. However, accurately predicting future driving conditions is a significant challenge for MPC in EV energy management problems. In the real world, achieving 100% accurate predictions is unrealistic, potentially posing performance risks during vehicle traveling. Moreover, the MPC algorithm lacks generalization, necessitating real-time recalculations when faced with new conditions.

PPO: The on-policy PPO algorithm exhibits significant variations in terms of performance across different EV environments. According to Fig. 4(a), the performance of PPO significantly varies across different driving cycles: NEDC, WLTC, and GCDC, achieving 81.6%, 75.5%, and 95.8% of the DP, respectively. Despite being evaluated under the same HEV environment, the results exhibited considerable variation, showing subpar performance in the initial two conditions and noteworthy improvement in the final one. Surprisingly, PPO performed exceptionally well in the FCEV environments, securing an impressive second place, with an energy efficiency of 97.5%. The PPO algorithm exhibits poor consistency that can be attributed to several factors. First, the convergence and sample efficiency of PPO are typically influenced by hyperparameter tuning and learning rate settings. Furthermore, PPO is an on-policy algorithm, meaning that it exclusively uses data generated by the current policy for training. If the current policy is not sufficiently robust, it may get stuck in local optimal across different environments, resulting in performance disparities.

DDPG, TD3, and SAC: We analyzed DDPG, TD3, and SAC together because they share several common characteristics. These common characteristics reflect their shared foundations in actor-critic architecture, off-policy learning, and suitability for continuous action spaces. As shown in Fig. 4(a), these three algorithms produce comparable results; however, SAC has a slight

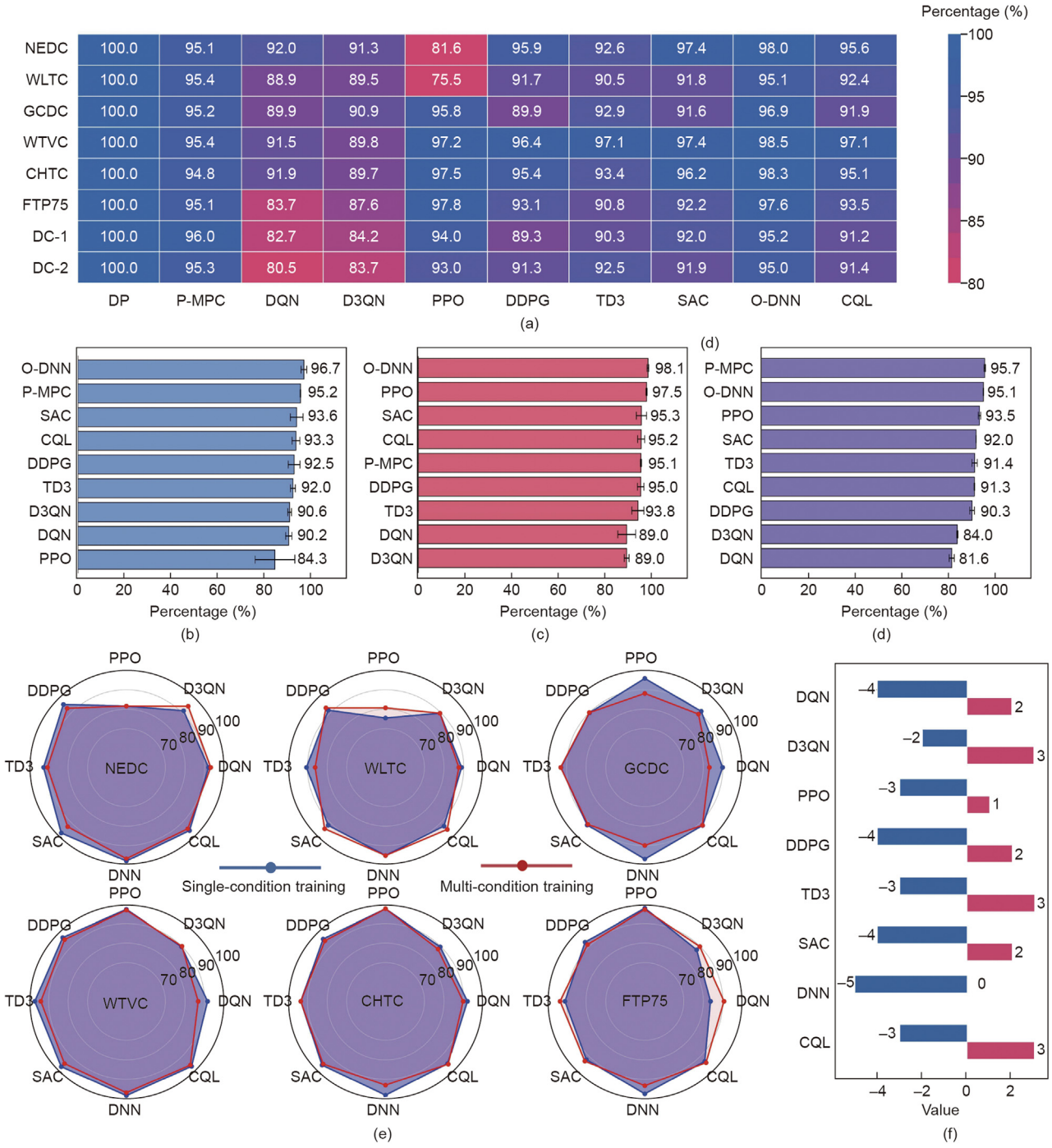

Fig. 4. Performance of different EMS algorithms. (a) Result of energy efficiency, where the matrix number represents the rate of the cost value of different algorithms relative to the DP under the same condition. (b) Comprehensive performance of different algorithms for HEV; the error bar refers to the standard deviation. (c) Comprehensive performance of different algorithms for FCEV. (d) Comprehensive performance of different algorithms for SHTV. (e) Comprehensive performance between SCT and MCT. (f) Metrics for improvement and deterioration in results. The red bars represent the instances where MCT outperformed SCT, whereas the blue bars indicate the opposite.

advantage. This can be attributed to the fact that SAC is a relatively newer algorithm with certain enhancements compared to DDPG and TD3. Indeed, the variations in hyperparameters can also influence the performance of algorithms. Compared to the PPO algorithms, the performances of DDPG, TD3, and SAC are more robust, indicating that off-policy algorithms designed for continuous action spaces are better suited for EMS problems. Note that these observations underscore the suitability of off-policy algorithms for EMS tasks and the importance of careful hyperparameter tuning to optimize algorithm performance.

DQL and DDQL: When comparing the results shown in Fig. 4(a), the performance of DQN and D3QN is significantly poorer under the three EV environments. Especially for FCEV and SHTV, where their results are only 89.0% and 89.0%, 81.6% and 84.0% of the DP in Figs. 4(c) and (d), making them the worst performing among all algorithms. The observed variations in the results can be ascribed to the discretization of the action space. For example, the SHTV has two continuous actions, and after discretization, 2^n possible control modes exist. To mitigate the high dimensionality of the action space, our experiments simplified this using 20

control modes. This coarse-grained control granularity had an adverse impact on the final results. In summary, the discretization of action spaces has a significant impact on the final EMS results, especially for vehicles with complex control variables.

CQL: In contrast DNN learning from an optimal EMS strategy, CQL employs a unique approach. It learns not from the optimal EMS strategy directly but rather from the data generated during training of the SAC algorithm. This distinctive learning mechanism leads to CQL exhibiting performance levels similar to SAC, with values of 93.3%, 95.2%, and 91.3% (Figs. 4(b)–(d)). Through extensive learning, CQL eventually converges to a robust EMS strategy, showcasing results comparable to those achieved by SAC. Under some conditions, CQL even surpasses SAC in terms of energy efficiency. CQL demonstrates that ORL can learn a reasonable EMS policy from non-expert data. The performance is contingent on the quality of the data, which will be further analyzed in subsequent sections.

4.4. Adaptability analysis

A notable characteristic of learning-based EMS, in contrast to rule-based and optimization-based methods, is their inherent ability for generalization. Thus, this section focuses solely on eight learning-based algorithms. Using an MCT training approach, a comparison was performed with the SCT approach in Section 4.3 to assess the adaptability of different algorithms. For DRL methods, the training data set comprises speed trajectories covering more than 1000 km. The models are subsequently validated on HEV and FCEV environments. However, because of the complexities associated with tracked vehicles, which involve considerations of steering and lateral movement, and the limited availability of real-world data, the SHTV is not discussed further in this section. In the case of imitation learning, the optimal strategies computed based on DP serve as data to train the DNN model. The model is then validated under test conditions. For offline RL, the CQL model is trained using data generated after the convergence of the SAC agent, ensuring high-quality data. The final results are presented in Table 5, revealing similar conclusions to those obtained earlier. The continuous action space algorithms outperform others, albeit with a lower consistency observed in PPO and DNN.

The radar chart in Fig. 4(e) illustrates the comprehensive energy efficiency of different algorithms using the SCT and MCT training modes. Overall, all algorithms show favorable results under different training approaches; the performance with MCT consistently exceeds 80% that of DP. Further analysis of the six radar charts in Fig. 4(e) provides information on the adaptability performance of the algorithms. Fig. 4(f) summarizes the variations in the performance across all algorithms under different training modes. Notably, across six validation scenarios and for the eight EMS algorithms (DQN, D3QN, PPO, DDPG, TD3, SAC, DNN, and CQL), MCT shows 28 instances of performance degradation and 16 instances of improvement compared to SCT. This suggests that, following extensive data training, DRL can develop a generalized EMS policy, allowing the deployment of learned policies in new driving scenarios. Importantly, our experimental results show no significant deterioration in the final energy efficiency performance. In imitation learning, the results of MCT worsen for five out of six conditions compared to those of SCT, indicating poorer adaptability of the DNN algorithm. Interestingly, the results for the ORL algorithm CQL are enhanced in three conditions, accompanied by only a slight reduction in the energy efficiency performance in the remaining three conditions. The overarching trend indicates that CQL and SAC exhibit comparable energy consumption performances. In fact, under certain conditions, CQL even excels in learning EMS strategies, surpassing the performance achieved by SAC. This observation underscores the adaptability and effectiveness of CQL across diverse data sets.

Imitation learning and ORL are closely related approaches, both involving the learning of EMS from data. In our experiments, DNN was trained with label data to approximate the strategy of DP. CQL learnt from the data generated by the SAC algorithm. This distinct learning mechanism results in significant differences in their performance. The DNN effectively approximates the DP strategy through supervised learning with commendable performance. However, it is crucial to highlight that ORL, represented by CQL in our experiments, can learn superior EMS strategies from non-expert historical data. In summary, the learning paradigm of offline RL, when compared with imitation learning, demonstrates greater practical utility. Imitation learning exhibits poorer adaptability and is unable to guarantee favorable outcomes under new conditions. In contrast, DRL and ORL demonstrate better adaptability and generalization capabilities. It also highlights the potential of CQL to learn and improve EMS policies in scenarios in which the data may not be generated by domain experts.

4.5. Energy efficiency analysis

The focus of this study, that is, the energy economy performance, is investigated to unravel the reason for the lower energy of the EMS. We started by evaluating two energy-related variables: engine power and battery SOC, which are directly linked to the energy consumption processes according to the energy-flow model of EVs. To this end, we focus on optimal DRL methods (SAC showing the best overall performance under the HEV environment according to Fig. 4(a), and PPO demonstrating the best performance on the FCEV). In addition, we also examined the performances of imitation learning and offline RL methods. Fig. 5(a) illustrates the distribution of engine operating points for the HEV under the WLTC condition. The DNN algorithm has fewer operating points in the low-efficiency region compared to SAC and CQL. A comparison of the point distribution of different methods is illustrated in Fig. 5(b). Notably, a low fuel consumption rate indicates high efficiency of engine operation at that point. This suggests that the engine tends to operate more in the high-efficiency operating range for DNN. Moreover, compared to SAC, CQL has a higher distribution of operating points in the range of the 200–250 fuel consumption rate. The results in Table 5 further confirm the validity of this observation, with SAC, DNN, and CQL showing energy consumption performances of 94.3%, 95.1%, and 94.8% of DP in the WLTC condition, respectively.

An analysis of the experimental results for FCEV is presented in Figs. 5(c) and (d). Fig. 5(c) illustrates the distribution of fuel cell power and its corresponding efficiency. The fuel cell system efficiency peaks at 54% in the 30–40 kW range. To comprehensively consider the balance between power demand and the total cost, the output power of PPO, DNN, and CQL algorithms is primarily distributed in the high-efficiency range. However, there are distinctions among them. PPO's power is mainly distributed in the 25–50 and 60–70 kW, DNN is primarily distributed in the 35–55 kW, and CQL is mainly distributed in the 30–70 kW, with a small amount in the inefficient 70–90 kW. This observation confirms the validity of the results in Table 5, where PPO, DNN, and CQL exhibit overall performances of 96.8%, 97.0%, and 96.2%, respectively. The SOC trajectories of the different EMS strategies are shown in Fig. 5(d).

5. Discussions

Although learning-based algorithms can enhance the energy economy of EVs, several critical issues merit discussion. In the domain of DRL, the thoughtful design of an action space, state space, and reward function holds significant importance. These

Table 5
Energy efficiency performance of different EMS algorithms in the MCT experiment.

Methods	HEV				FCEV			
	Conditions	Results/SOC	DP/SOC	Ratio	Conditions	Results/SOC	DP/SOC	Ratio
DQN	NEDC	0.439/0.596	0.409/0.599	93.1%	WTVC	102.207/0.477	88.592/0.480	86.7%
	WLTC	1.089/0.586	0.953/0.585	87.5%	CHTC	75.917/0.471	68.119/0.470	89.7%
	GCDC	0.356/0.502	0.296/0.495	83.1%	FTP75	85.587/0.483	77.503/0.480	90.6%
D3QN	NEDC	0.406/0.535	0.384/0.535	94.6%	WTVC	99.079/0.485	89.295/0.491	90.1%
	WLTC	1.030/0.520	0.922/0.524	89.5%	CHTC	79.117/0.500	69.697/0.501	88.1%
	GCDC	0.333/0.498	0.296/0.495	88.9%	FTP75	87.312/0.509	78.473/0.510	89.9%
PPO	NEDC	0.478/0.536	0.390/0.540	81.6%	WTVC	89.176/0.454	86.316/0.450	**96.8%**
	WLTC	1.139/0.505	0.919/0.508	80.7%	CHTC	71.454/0.502	69.697/0.501	**97.5%**
	GCDC	0.336/0.498	0.296/0.495	88.1%	FTP75	81.360/0.525	79.016/0.520	**97.1%**
DDPG	NEDC	0.432/0.583	0.402/0.590	93.1%	WTVC	95.987/0.521	91.194/0.520	**95.0%**
	WLTC	1.011/0.564	0.944/0.563	93.4%	CHTC	74.639/0.508	70.247/0.510	94.1%
	GCDC	0.331/0.501	0.298/0.500	90.0%	FTP75	86.498/0.523	79.016/0.520	91.4%
TD3	NEDC	0.417/0.531	0.378/0.520	90.6%	WTVC	96.494/0.505	90.723/0.511	94.0%
	WLTC	1.085/0.550	0.935/0.548	86.2%	CHTC	74.594/0.503	69.697/0.501	93.4%
	GCDC	0.314/0.420	0.292/0.480	93.0%	FTP75	84.075/0.511	78.473/0.510	93.3%
SAC	NEDC	0.433/0.590	0.402/0.590	92.8%	WTVC	93.176/0.481	88.592/0.480	**95.1%**
	WLTC	1.077/0.606	1.016/0.605	94.3%	CHTC	73.065/0.503	69.697/0.501	**95.4%**
	GCDC	0.325/0.496	0.296/0.495	91.1%	FTP75	84.148/0.506	78.473/0.510	93.3%
DNN	NEDC	0.403/0.547	0.390/0.540	**96.7%**	WTVC	93.524/0.516	90.723/0.511	**97.0%**
	WLTC	1.068/0.610	1.016/0.605	**95.1%**	CHTC	75.462/0.512	70.247/0.510	93.1%
	GCDC	0.346/0.568	0.311/0.570	89.9%	FTP75	85.035/0.535	79.435/0.530	93.4%
CQL	NEDC	0.405/0.527	0.382/0.530	94.3%	WTVC	96.503/0.550	92.875/0.551	**96.2%**
	WLTC	1.071/0.602	1.016/0.605	94.8%	CHTC	74.102/0.526	70.787/0.520	**95.5%**
	GCDC	0.318/0.481	0.292/0.480	91.8%	FTP75	84.172/0.531	79.435/0.530	94.4%

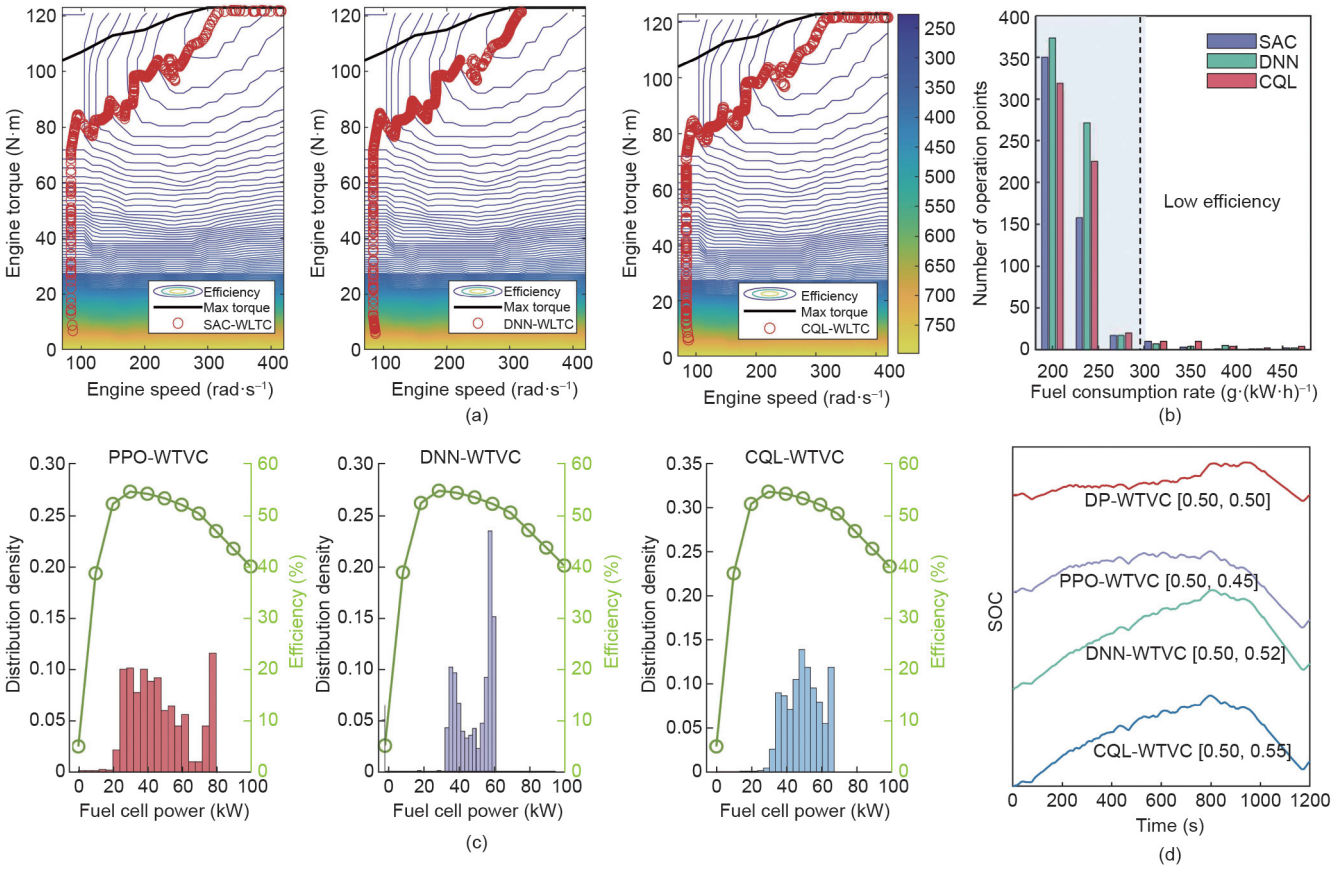

Fig. 5. Energy efficiency analysis through comparisons of engine operation points. (a) Distribution of working points for the HEV engine. (b) Fuel consumption distributions by the three different strategies. (c) Efficiency distribution of the fuel cell power. (d) Battery SOC comparison.

three key components significantly influence the algorithm's effectiveness and performance in addressing real-world challenges. Therefore, in this section, we provide a thorough discussion of configuring state, reward, and action settings specifically for the application of RL in EV energy management. Furthermore, we introduce a policy extraction and reconstruction method for learning-based EMS deployment and conduct hardware-in-the-loop experiments. Given the superior performance of the PPO algorithm observed in

previous experiments involving the FCEV environment, this section exclusively leverages the PPO algorithm for illustration and discussion.

5.1. Partially observable MDP (POMDP)

The POMDP is characterized by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, \gamma)$, where $\mathcal{S}$ represents the set of states, $\mathcal{A}$ the set of actions, $\mathcal{T}$ the state transition probabilities, $\mathcal{R}$ the reward function, Ω the set of observations, and $\mathcal{O}$ the observation probabilities. Unlike in an MDP, in a POMDP, the agent lacks direct access to the true state of the environment and instead relies on observations $o \in \Omega$, with each observation o depending on the new state s' and action a, governed by $P(o \mid s', a)$.

The conventional treatment of the EMS within RL methodologies typically models the problem as an MDP. However, a critical examination of this approach reveals its limitations. MDPs are predicated on the Markov property, implying that future states depend solely on the current state and action, independent of past states. This assumption does not fully hold in the context of vehicle energy management. For example, state variables such as vehicle velocity, acceleration, and SOC, along with actions pertaining to energy allocation, do not uniquely determine the state transition function $P_a(s, s')$. This is evident in the differing characteristics of $P_a(s, s')$ when we compared vehicles on highways against those on urban roads, indicating that these state variables often represent only partial observations of the vehicle's environment. Moreover, the presence of missing data, a common challenge in onboard sensors, further complicates the ability to reliably capture vehicle state at all times [45]. Therefore, clearly, modeling EMS as a POMDP, which accounts for the partial observability and inherent uncertainties of the vehicle's environment, offers a more accurate and robust framework for energy management in vehicles. Expanding LearningEMS to include POMDP environments, integrating these POMDP algorithms, and testing various POMDP methodologies would be highly valuable.

5.2. Reward function

The design of the reward function plays a critical role in the RL agent's learning process. Therefore, in this section, we will dive into the issue of reward function design in the context of energy management. For the FCEV, multiple objectives are considered, including hydrogen consumption, battery equivalent hydrogen consumption (battery electricity consumption and degradation), and fuel cell degradation cost. Four reward functions, denoted R1, R2, R3, and R4, are established to reflect varying considerations. Note that all costs are converted into monetary units.

R1: It solely focuses on minimizing the cost of hydrogen consumption in Eq. (10).

$$\mathrm{R1} = C_{\mathrm{fcs,H_2}} \tag{32}$$

R2: The reward function accounts for both the cost of hydrogen consumption and the battery equivalent hydrogen consumption, which encompasses electricity consumption and degradation.

$$\mathrm{R2} = C_{\mathrm{fcs,H_2}} + C_{\mathrm{bat,eH_2}} + C_{\mathrm{bat,degr}} \tag{33}$$

The battery electricity consumption $C_{\mathrm{bat,eH_2}}$ is calculated according to the battery charge/discharge efficiency and is converted into price cost:

$$C_{\mathrm{bat,eH_2}} = \int_0^t \left[\frac{P_{\mathrm{bat}}}{\eta_{\mathrm{d/c}} \cdot \eta_{\mathrm{dc}} \cdot \mathrm{LHV_{H_2}}} \right] \mathrm{d}t \cdot p_{\mathrm{H_2}} \tag{34}$$

where $\eta_{\mathrm{d/c}}$ is the battery discharge/charge efficiency, η_{dc} is the DC/DC efficiency, and $p_{\mathrm{H_2}}$ is the hydrogen price per kilogram.

R3: The third reward function introduces the factor of hydrogen fuel cell degradation cost.

$$\mathrm{R3} = C_{\mathrm{fcs,H_2}} + C_{\mathrm{fcs,degr}} \tag{35}$$

R4: This integrates all three optimization objectives: hydrogen consumption cost, battery-related costs (consumption and degradation), and fuel cell degradation cost $C_{\mathrm{bat,degr}}$:

$$\mathrm{R4} = C_{\mathrm{fcs,H_2}} + C_{\mathrm{fcs,degr}} + C_{\mathrm{bat,eH_2}} + C_{\mathrm{bat,degr}} \tag{36}$$

Fig. 6(a) illustrates the total cost under four different reward functions during the WTVC operating condition, which includes hydrogen consumption, battery cost, and fuel cell degradation cost. Clearly, a more comprehensive consideration of factors in the design of the reward function results in lower costs. In the R1 and R2 strategies, the absence of the consideration of fuel cell degradation leads to higher total cost. With the incorporation of degradation considerations, the costs associated with fuel cell degradation undergo a substantial reduction, dropping from their previous values of 40.83 and 38.62 to a mere 1.92 and 2.14, respectively. Fig. 6(b) showcases the fuel cell output power profiles for the four strategies. The R1 and R2 strategies exhibit a wider range of output power variation, with instances of power levels falling below 20 kW and exceeding 80 kW. Both the low- and high-power operations contribute to the accelerated degradation of the fuel cell. In contrast, the R3 and R4 strategies only exhibit power levels below 20 kW during the initial startup phase, attributable to the incorporation of fuel cell degradation cost into the reward function.

Thus, the aforementioned experimental results underscore that the energy management for EVs is a multi-objective problem. In particular, fuel cell lifetime is a crucial factor, as evidenced by our experimental results. To emphasize its significance, we designed distinct reward functions to highlight the role of fuel cell degradation in our results. Furthermore, our experimental findings highlight that in RL-based EMS, the thoughtful design of the reward function can significantly impact comprehensive performance.

5.3. Action setting

The significance of the thoughtfully designed action spaces in enhancing the efficiency and performance of RL algorithms is evident in practical problem domains. The action space effectively establishes an upper bound for the performance on specific tasks. As illustrated in Section 3.3, distinct vehicle models require specific action settings, a differentiation attributed to the specific characteristics of EV. For the FCEV, maintaining a steady power output of the fuel cell system is crucial. Consequently, one viable approach involves adopting an incremental action space, in which the action is defined as the rate of change of power output. By focusing on controlling the rate of power change (power slope), this incremental approach allows achieving a smoother and more controlled power delivery. Therefore, this section is dedicated to the comprehensive examination of how different action control settings influence the performance of EMS. Specifically, we focus on the investigation of four distinct action configurations, denoted as PS-5, PS-10, PS-15, and PS-100. The PS-100 setting signifies a fuel cell power slope of zero, wherein the action directly corresponds to the output power of the fuel cell. In contrast, PS-5 denotes a power slope of 5 kW, which indicates dynamic power adjustments within this range. Similarly, PS-10 and PS-15 represent power slopes of 10 and 15 kW, respectively.

Fig. 6(c) presents the distribution cloud chart of fuel cell power variations. By comparing four different action settings (PS-5, PS-10, PS-15, and PS-100), notably, we can see that larger power slopes

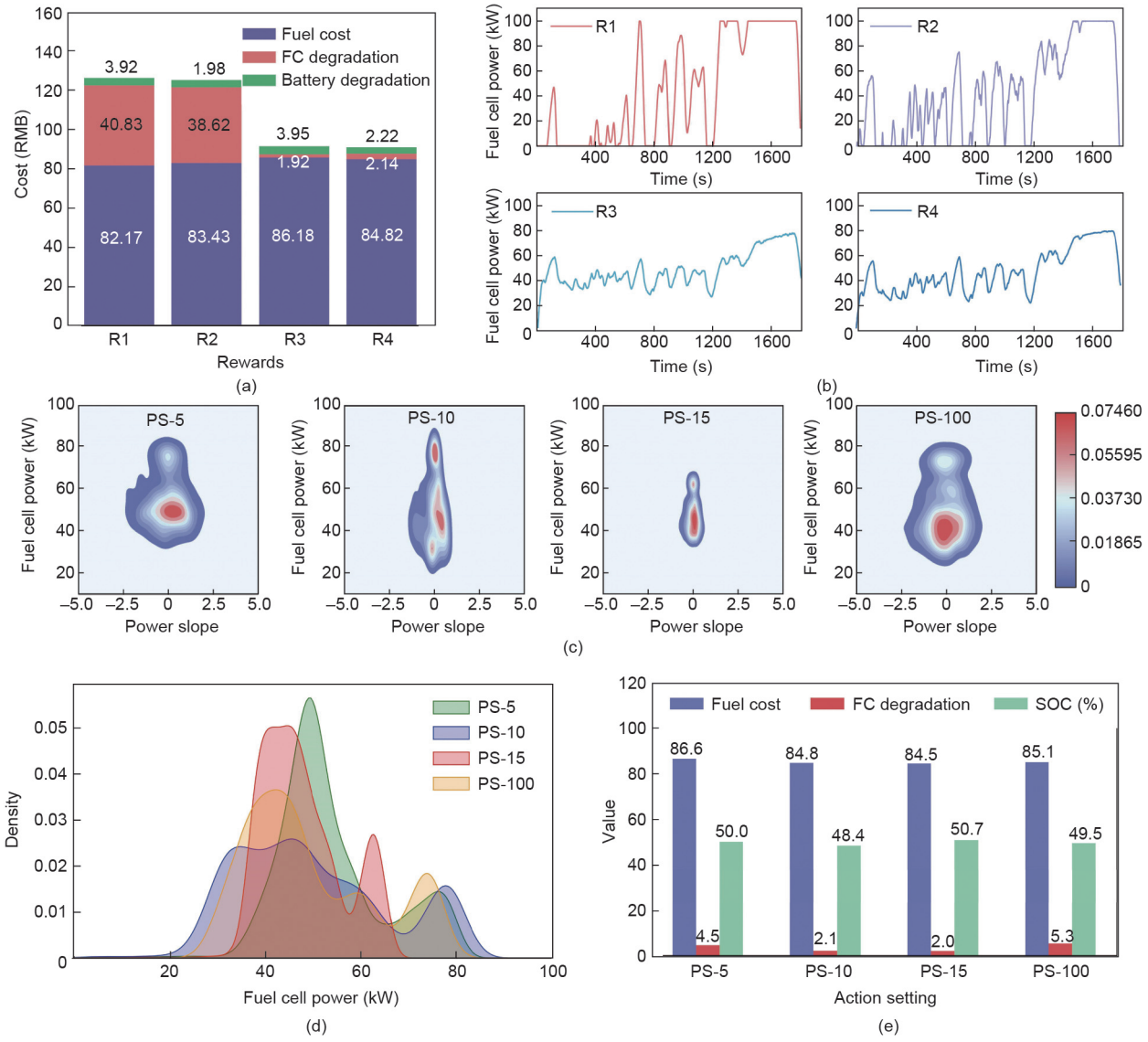

Fig. 6. Comparison of the different reward and action settings. (a) Comparing the total cost across different reward functions; the total cost comprises hydrogen consumption, battery cost, and fuel cell (FC) degradation cost. (b) Fuel cell power for the four strategies. (c) Fuel cell power distribution cloud chart. (d) Distribution of the power of the fuel cell. (e) Comparison of cost results of different strategies.

lead to smaller output actions generated by the DRL agent. For the PS-5 strategy, the action falls within the range of [−2.5, 2.5]. The PS-10 strategy encompasses action values in the range of approximately [−1.5, 1.5], while the PS-15 actions are confined to a narrower interval of [−0.6, 0.6]. The power variations for PS-100 range around approximately 2.5 kW. A comparison of the distribution in Fig. 6(d) further reveals that the PS-15 strategy results in a more concentrated fuel cell power output, indicative of smoother power delivery dynamics. Fig. 6(e) provides a comparison of the final cost values for the four strategies. The results indicate that the PS-100 strategy and PS-5 incur a higher cost of fuel cell degradation, whereas the other two strategies exhibit lower cost. This highlights that in EMS scenarios involving system lifetime, it is more beneficial to use an incremental action approach. Additionally, different values for the fuel cell power slope noticeably affect the final results. Our findings indicate that, within a reasonable range, higher action slopes lead to better results. Considering the fuel cell's power response time, we set the maximum power slope to 15% of the total power.

In summary, while directly assigning actions to output power levels might present challenges in maintaining stable control, the incremental action strategy emerges as a promising approach to mitigate power fluctuations, improve power stability, and potentially prolong the lifetime of fuel cell systems. These findings highlight the potential benefits of employing incremental action settings, especially in scenarios where system lifetime is a key consideration. Moreover, the experimentation provides insights into the trade-off between action precision and control performance, offering valuable guidance regarding the RL-based EMS.

5.4. Learning-based EMS deployment

Deploying a learning-based EMS from simulation environments to real-world scenarios is a complex and systematic engineering task. In the automotive industry, the V-model development approach is commonly used for software development, with environments often relying on the C code such as MATLAB/Simulink [46]. However, in the context of machine learning, algorithms are

predominantly developed in Python. This section outlines the process of deploying a learning-based EMS that has been trained on Python into a real-world vehicle controller. The deployment process is illustrated in Fig. 7(a). The initial step involves saving the neural network parameters of the learning-based algorithm trained in Python, which is referred to as the pretrained network. In our LearningEMS library, these parameters are saved as ".path" files. The next step involves policy extraction, in which the actor network of the agent is converted into parameterized matrices. These matrices comprise the weights and biases of each neural network. The network parameters saved in the ".path" file are then stored into multiple ".mat" files. Policy reconstruction is the third step, which involves recreating the learning-based EMS based on the network parameters obtained from the policy extraction process. By loading ".mat" files in the MATLAB/Simulink environment, a new parameterized EMS is reconstructed. The model created in Simulink can be readily compiled and uploaded onto the actual onboard controllers for online control.

Hardware-in-the-loop (HIL) plays a crucial role in the development and testing of embedded systems, particularly in the context of vehicle control units (VCUs) for vehicles. In the deployment of the EMS learned by the DRL agent from the simulation to a real controller, as depicted in Fig. 7(b), the network parameters are transferred to the Dspace hardware controller following policy extraction and reconstruction steps. Recognoizing that different algorithms necessitate custom policy reconstruction functions is essential. For example, the actor network of DDPG and TD3 algorithms directly generates action values, whereas the actor network of the PPO algorithm produces the probability of each action. Thus, designing appropriate policy reconstruction functions tailored to different algorithms is crucial.

Fig. 7(c) presents a comparison between the experimental results from the simulation in Python and those deployed on the real controller. The results of the parameterized EMS results closely align with the original EMS model, showcasing parallel fuel cell power output and battery SOC trajectories. The slight discrepancy can be attributed to the simplified vehicle model utilized in Python. These experiments underscore the success of cross-platform EMS development, with the detailed workflow and code to be open-sourced. Researchers can leverage our proposed LearningEMS library to develop advanced EMS algorithms in Python and transform them into parameterized EMS models that are compatible with mainstream vehicle development tools.

6. Conclusion

Here, we present LearningEMS, a unified framework aimed at simplifying the creation of EV simulation models, promoting method innovation, and serving as a versatile evaluation and deployment tool (Fig. 1). The study involves the implementation and comparison of various EMS algorithms, encompassing optimization, DRL, imitation learning, and offline RL algorithms. The results indicate that among the implemented algorithms, the discrete action space algorithms DQN and D3QN excel in simple EMS tasks. However, their energy performance diminishes when confronted with multiple control parameters and the need for complex action space design. Off-policy algorithms with continuous action spaces, such as DDPG, TD3, and SAC, stand out because of their prowess in optimizing energy efficiency and demonstrating consistency across diverse driving conditions. The on-policy algorithm PPO shows significant performance variations in different vehicles or operational conditions, and its consistency is relatively

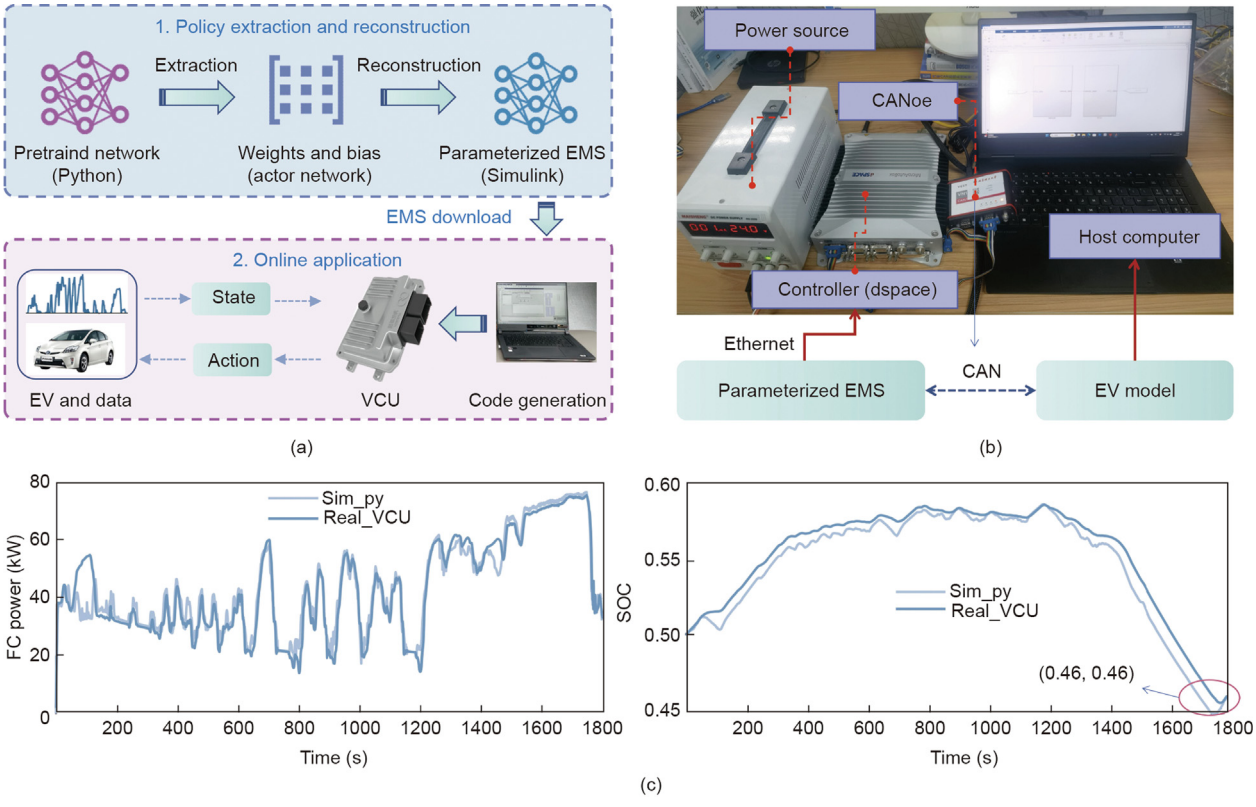

Fig. 7. EMS deployment process. (a) The deployment of a learning-based EMS from simulation to real-world involves saving network parameters, extracting policies, and reconstructing them before deploying the EMS into VCU. (b) During the HIL process, the parameterized EMS is compiled into the rapid prototyping controller, with the host computer used to interact with the Dspace and communicate with the vehicle model via CANoe (Vector; Germany). (c) Comparing the results of the FC power and SOC between simulation in Python (Sim_py) and real-world implementation in VCU (Real_VCU).

weaker. Notably, all DRL algorithms and offline RL exhibit robust adaptability, maintaining consistent performance even when deployed in novel conditions after being trained on specific strategies. Imitation learning demonstrates outstanding performance in certain conditions; the adaptability and applicability require further in-depth research. ORL, as exemplified by the CQL algorithm in our experiments, holds more significant applicability and effectiveness in real-world scenarios, particularly in learning and improving EMS strategies from non-expert historical data. Furthermore, our experimental findings highlight that in RL-based EMS, the thoughtful design of the reward function and action can significantly impact overall performance. The HIL experiments further demonstrate that deploying learning-based EMS trained on Python to real-world scenarios is feasible.

To the best of our knowledge, there does not exist a unified EMS framework and comprehensive benchmark.Although several comparative studies analyze the features of state-of-the-art EMS, most of these predominantly focus on traditional EMS approaches [47,48], conducted on a specific type of EV with a limited number of standard driving cycles [20,31]. Our study presents a unified EMS framework by open-sourcing algorithms, environments, data sets, and deployment processes. This allows engineers and researchers to develop more advanced EMS using our LearningEMS framework, which can be applied to mass-produced EVs or for further innovative research. The LearningEMS framework has the potential to significantly improve energy efficiency, reduce vehicle operating costs, extend power system lifespan, and advance sustainable transportation. According to China Energy-Saving and New Energy Vehicle Technology Roadmap [49], China aims to achieve a complete electrification transformation of the automotive industry by 2035, with passenger vehicle energy consumption reaching the target of 4 L/100 km. LearningEMS represents a significant step forward in the development of advanced EMS strategies for EVs. We hope our open-source efforts can contribute to the development of more advanced and intelligent energy-saving control technologies for electric vehicles.

Our work marks an initial milestone in establishing a foundational EMS platform through comprehensive open-sourcing. Future research should focus on several key areas: ① the "sim-to-real" challenge must be addressed by enhancing model generalization and ensuring strategy safety to make simulation-trained agents reliable in real-world scenarios. ② As automotive driving technology advances, integrating driving systems with powertrain systems will be essential for achieving more efficient and cohesive control. ③ Given that learning-based methods are often black-box models, we must ensure that the algorithmic decision-making process remains transparent, interpretable, and ethical. Given the scope of this project, we anticipate initial challenges as users start using the LearningEMS platform. Therefore, our objective is to continuously maintain and enhance the open-source benchmark. We encourage researchers to participate in the open challenge to make progress and evaluate their EMS algorithms. Additionally, we hope, with community support, to continually expand EV environments, methods, data, and tasks for both training and evaluation purposes. Our aspiration is for LearningEMS to evolve into a crucial resource for a wide spectrum of EMS-related research. We eagerly anticipate witnessing achievements of the community across this diverse range of tasks.

CRediT authorship contribution statement

Yong Wang: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Formal analysis, Conceptualization. **Hongwen He:** Writing – review & editing, Project administration, Investigation, Funding acquisition. **Yuankai Wu:** Writing – review & editing, Software, Methodology. **Pei Wang:** Visualization, Software. **Haoyu Wang:** Visualization, Software. **Renzong Lian:** Writing – review & editing, Software, Methodology. **Jingda Wu:** Visualization, Methodology. **Qin Li:** Writing – original draft, Project administration. **Xiangfei Meng:** Visualization, Software. **Yingjuan Tang:** Writing – review & editing, Visualization. **Fengchun Sun:** Writing – review & editing, Supervision, Funding acquisition. **Amir Khajepour:** Writing – review & editing, Supervision, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (52172377).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.eng.2024.10.021; https://github.com/wangjail/LearningEMS.

References

[1] Powell S, Cezar GV, Min L, Azevedo IM, Rajagopal R. Charging infrastructure access and operation to reduce the grid impacts of deep electric vehicle adoption. Nat Energy 2022;7(10):932–45.
[2] Li Y, He H, Khajepour A, Chen Y, Huo W, Wang H. Deep reinforcement learning for intelligent energy management systems of hybrid-electric powertrains: recent advances, open issues, and prospects. IEEE Trans Transp Electrif 2024;10(4):9877–903.
[3] Wen CK, Ren W, Zhu QZ, Zhao CJ, Luo ZH, Zhang SL, et al. Reducing operation emissions and improving work efficiency using a pure electric wheel drive tractor. Engineering 2024;37:230–45.
[4] He H, Meng X, Wang Y, Khajepour A, An X, Wang R, et al. Deep reinforcement learning based energy management strategies for electrified vehicles: recent advances and perspectives. Renew Sustain Energy Rev 2024;192:114248.
[5] Zhang F, Hu X, Langari R, Cao D. Energy management strategies of connected HEVs and PHEVs: recent progress and outlook. Pror Energy Combust Sci 2019;73:235–56.
[6] Wu J, Huang C, He H, Huang H. Confidence-aware reinforcement learning for energy management of electrified vehicles. Renew Sustain Energy Rev 2024;191:114154.
[7] Dong P, Zhao J, Liu X, Wu J, Xu X, Liu Y, et al. Practical application of energy management strategy for hybrid electric vehicles based on intelligent and connected technologies: development stages, challenges, and future trends. Renew Sustain Energy Rev 2022;170:112947.
[8] Li Q, Liu P, Meng X, Zhang G, Ai Y, Chen W. Model prediction control-based energy management combining self-trending prediction and subset-searching algorithm for hydrogen electric multiple unit train. IEEE Trans Transp Electrif 2022;8(2):2249–60.
[9] Ahmed M, Zheng Y, Amine A, Fathiannasab H, Chen Z. The role of artificial intelligence in the mass adoption of electric vehicles. Joule 2021;5(9):2296–322.
[10] Millo F, Rolando L, Tresca L, Pulvirenti L. Development of a neural network-based energy management system for a plug-in hybrid electric vehicle. Transp Eng 2023;11:100156.
[11] Liu B, Wei X, Sun C, Wang B, Huo W. A controllable neural network-based method for optimal energy management of fuel cell hybrid electric vehicles. Int J Hydrogen Energy 2024;55:1371–82.
[12] Hou J, Chen G, Huang J, Qiao Y, Xiong L, Wen F, et al. Large-scale vehicle platooning: advances and challenges in scheduling and planning techniques. Engineering 2023;28:26–48.
[13] Wang Y, Wu Y, Tang Y, Li Q, He H. Cooperative energy management and eco-driving of plug-in hybrid electric vehicle via multi-agent reinforcement learning. Appl Energy 2023;332:120563.
[14] Tan H, Zhang H, Peng J, Jiang Z, Wu Y. Energy management of hybrid electric bus based on deep reinforcement learning in continuous state and action space. Energy Convers Manage 2019;195:548–60.
[15] Wu J, Huang Z, Hu Z, Lv C. Toward human-in-the-loop AI: enhancing deep reinforcement learning via real-time human guidance for autonomous driving. Engineering 2023;21:75–91.

[16] Yuan K, Huang Y, Yang S, Zhou Z, Wang Y, Cao D, et al. Evolutionary decision-making and planning for autonomous driving based on safe and rational exploration and exploitation. Engineering 2024;33:108–20.
[17] Han X, He H, Wu J, Peng J, Li Y. Energy management based on reinforcement learning with double deep Q-learning for a hybrid electric tracked vehicle. Appl Energy 2019;254:113708.
[18] Qi X, Luo Y, Wu G, Boriboonsomsin K, Barth M. Deep reinforcement learning enabled self-learning control for energy efficient driving. Transp Res Part C Emerg Technol 2019;99:67–81.
[19] Yang C, Lu Z, Wang W, Wang M, Zhao J. An efficient intelligent energy management strategy based on deep reinforcement learning for hybrid electric flying car. Energy 2023;280:128118.
[20] Wang H, Ye Y, Zhang J, Xu B. A comparative study of 13 deep reinforcement learning based energy management methods for a hybrid electric vehicle. Energy 2023;266:126497.
[21] Li Y, He H, Peng J, Wang H. Deep reinforcement learning-based energy management for a series hybrid electric vehicle enabled by history cumulative trip information. IEEE Trans Vehicular Technol 2019;68(8):7416–30.
[22] Wu J, Wei Z, Li W, Wang Y, Li Y, Sauer DU. Battery thermal- and health-constrained energy management for hybrid electric bus based on soft actor-critic DRL algorithm. IEEE Trans Industr Inform 2021;17(6):3751–61.
[23] Jia C, Zhou J, He H, Li J, Wei Z, Li K. Health-conscious deep reinforcement learning energy management for fuel cell buses integrating environmental and look-ahead road information. Energy 2024;290:130146.
[24] Huang R, He H, Zhao X, Gao M. Longevity-aware energy management for fuel cell hybrid electric bus based on a novel proximal policy optimization deep reinforcement learning framework. J Power Sources 2023;561:232717.
[25] Liu ZE, Li Y, Zhou Q, Li Y, Shuai B, Xu H, et al. Deep reinforcement learning based energy management for heavy duty HEV considering discrete-continuous hybrid action space. IEEE Trans Transp Electrif 2024;10(4):9864–76.
[26] Li Y, He H, Khajepour A, Wang H, Peng J. Energy management for a power-split hybrid electric bus via deep reinforcement learning with terrain information. Appl Energy 2019;255:113762.
[27] Lian R, Tan H, Peng J, Li Q, Wu Y. Cross-type transfer for deep reinforcement learning based hybrid electric vehicle energy management. IEEE Trans Vehicular Technol 2020;69(8):8367–80.
[28] Huang R, He H, Su Q. Towards a fossil-free urban transport system: an intelligent cross-type transferable energy management framework based on deep transfer reinforcement learning. Appl Energy 2024;363:123080.
[29] Agarwal R, Schuurmans D, Norouzi M. An optimistic perspective on offline reinforcement learning. In: Proceedings of the 37th International Conference on Machine Learning (PMLR 2020); 2020 Jul 13–18; online. Birmingham: Proceedings of Machine Learning Research; 2020. p. 104–14.
[30] He H, Niu Z, Wang Y, Huang R, Shou Y. Energy management optimization for connected hybrid electric vehicle using offline reinforcement learning. J Energy Storage 2023;72:108517.
[31] Wang Z, He H, Peng J, Chen W, Wu C, Fan Y, et al. A comparative study of deep reinforcement learning based energy management strategy for hybrid electric vehicle. Energy Convers Manage 2023;293:117442.
[32] Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, et al. OpenAI Gym. 2016. arXiv:1606.01540.
[33] Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V. Carla: an open urban driving simulator. In: Proceedings of the Conference on robot learning (PMLR 2017); 2017 Nov 13–15; Mountain View, CA, USA. Birmingham: Proceedings of Machine Learning Research; 2020. 2017. p. 1–16.
[34] Kato S, Tokunaga S, Maruyama Y, Maeda S, Hirabayashi M, Kitsukawa Y, et al. Autoware on board: enabling autonomous vehicles with embedded systems. In: Proceedings of the 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS); 2018 Apr 11–13; Porto, Portugal. New York City: IEEE; 2018. p. 287–96.
[35] Wang Y, Tan H, Wu Y, Peng J. Hybrid electric vehicle energy management with computer vision and deep reinforcement learning. IEEE Trans Industr Inform 2021;17(6):3857–68.
[36] Yuankai W, Renzong L, Yong W, Yi L. Benchmarking deep reinforcement learning based energy management systems for hybrid electric vehicles. In: Proceedings of the CAAI International Conference on Artificial Intelligence; 2022 Aug 27–28; Beijing, China. Berlin: Springer; 2022. p. 613–25.
[37] Li Q, Yin L, Yang H, Wang T, Qiu Y, Chen W. Multiobjective optimization and data-driven constraint adaptive predictive control for efficient and stable operation of pemfc system. IEEE Trans Ind Electron 2021;68(12):12418–29.
[38] Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N. Dueling network architectures for deep reinforcement learning. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning; 2016 Jun 19–24; New York, NY, USA. Birmingham: Proceedings of Machine Learning Research; 2016. p. 1995–2003.
[39] Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods. In: Proceedings of the International Conference on Machine Learning; 2018 Jul 10–15; Stockholm, Sweden. Birmingham: Proceedings of Machine Learning Research; 2018. p. 1587–96.
[40] Haarnoja T, Zhou A, Abbeel P, Levine S. Soft actor–critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Proceedings of the International Conference on Machine Learning; 2018 Jul 10–15; Stockholm, Sweden. Birmingham: Proceedings of Machine Learning Research; 2018. p. 1861–70.
[41] Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O. Proximal policy optimization algorithms. 2017. arXiv:1707.06347.
[42] Peng J, Ren T, Chen Z, Chen W, Wu C, Ma C. Efficient training for energy management in fuel cell hybrid electric vehicles: an imitation learning-embedded deep reinforcement learning framework. J Clean Prod 2024;447:141360.
[43] Prudencio RF, Maximo MR, Colombini EL. A survey on offline reinforcement learning: taxonomy, review, and open problems. IEEE Trans Neural Netw Learn Syst 2023;35(8):10237–57.
[44] Kumar A, Zhou A, Tucker G, Levine S. Conservative Q-learning for offline reinforcement learning. Adv Neural Inf Process Syst 2020;33:1179–91.
[45] Chen X, Mu YM, Luo P, Li S, Chen J. Flow-based recurrent belief state learning for POMDPs. In: Proceedings of the International Conference on Machine Learning; 2022 Jul 17–23; Baltimore, MD, USA. Birmingham: Proceedings of Machine Learning Research; 2022. p. 3444–68.
[46] Wang Y, Lian R, He H, Betz J, Wei H. Auto-tuning dynamics parameters of intelligent electric vehicles via Bayesian optimization. IEEE Trans Transp Electrif 2023;10(3):6915–27.
[47] Zhang F, Xiao L, Coskun S, Pang H, Xie S, Liu K, et al. Comparative study of energy management in parallel hybrid electric vehicles considering battery ageing. Energy 2023;264:123219.
[48] Soumeur MA, Gasbaoui B, Abdelkhalek O, Ghouili J, Toumi T, Chakar A. Comparative study of energy management strategies for hybrid proton exchange membrane fuel cell four wheel drive electric vehicle. J Power Sources 2020;462:228167.
[49] China Society of Automotive Engineers (SAE-China). Energy-saving and new energy vehicle technology roadmap 2.0. Beijing: SAE-China; 2020.