Green Energy and Intelligent Transportation 1 (2022) 100018

Contents lists available at ScienceDirect

Green Energy and Intelligent Transportation

journal homepage: www.journals.elsevier.com/green-energy-and-intelligent-transportation

Full length article

A transferable energy management strategy for hybrid electric vehicles via dueling deep deterministic policy gradient

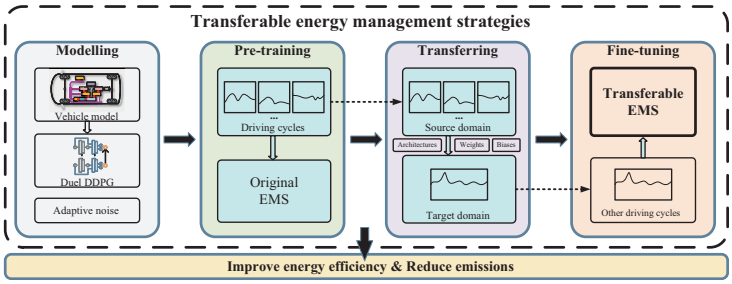
Jingyi Xu [a,1], Zirui Li [a,b,1], Guodong Du [a,c], Qi Liu [a], Li Gao [a], Yanan Zhao [a,*]

[a] School of Mechanical Engineering, Beijing Institute of Technology, 100081, Beijing, China
[b] Department of Transport and Planning, Delft University of Technology, Delft, 2628 CD, the Netherlands
[c] Institute of Dynamic System and Control, ETH Zurich, Sonneggstrasse 3, 8092, Zurich, Switzerland

HIGHLIGHTS

- The combination of deep deterministic policy gradient and transfer learning leads to the solid generalization ability.
- The dueling architecture used in the critic network improves the stability of the energy management strategies.
- The adaptive parameter space noise and action space noise are used to balance agents' exploration and exploitation.

GRAPHICAL ABSTRACT



ARTICLE INFO

Keywords:
Energy management strategies
Deep reinforcement learning
Dueling network architecture
Transfer learning

ABSTRACT

Due to the high mileage and heavy load capabilities of hybrid electric vehicles (HEVs), energy management becomes crucial in improving energy efficiency. To avoid the over-dependence on the hard-crafted models, deep reinforcement learning (DRL) is utilized to learn more precise energy management strategies (EMSs), but cannot generalize well to different driving situations in most cases. When driving cycles are changed, the neural network needs to be retrained, which is a time-consuming and laborious task. A more efficient transferable way is to combine DRL algorithms with transfer learning, which can utilize the knowledge of the driving cycles in other new driving situations, leading to better initial performance and a faster training process to convergence. In this paper, we propose a novel transferable EMS by incorporating the DRL method and dueling network architecture for HEVs. Simulation results indicate that the proposed method can generalize well to new driving cycles, with comparably initial performance and faster convergence in the training process.

1. Introduction

With the increasing awareness of energy saving and environmental protection, more and more emphasis has been put on hybrid electric vehicles (HEVs), which involve two or more energy sources [1]. Thus there is a considerable need for energy management strategies (EMSs) to distribute power supplements among several power sources. As the driving conditions continue to change, there is a potential direction for research into how EMSs can adapt to different real HEV applications.

Various EMSs for HEVs have been conducted in recent years. Zhang et al. [2] proposed a comprehensive hierarchical classification scheme for the first time, offline EMSs and online EMSs, respectively. The offline

* Corresponding author.
E-mail addresses: 3220200359@bit.edu.cn (J. Xu), 31201905255@bit.edu.cn (Z. Li), guoddu@ethz.ch (G. Du), 3120195257@bit.edu.cn (Q. Liu), ligaobit@bit.edu.cn (L. Gao), zyn@bit.edu.cn (Y. Zhao).
[1] Jingyi Xu and Zirui Li contribute equally.

https://doi.org/10.1016/j.geits.2022.100018
Received 1 June 2022; Received in revised form 21 June 2022; Accepted 1 July 2022
Available online 13 July 2022

EMSs can be further categorized according to the information level of the driving conditions, including rule-based EMSs and global optimization-based methods. The former is considered an offline method because a series of rules are built based on the intuition and experience of engineers [3,4]. However, the rule-based designs are not robust to different vehicle types, which need high calibration efforts frequently [5]. The effectiveness of the rule-based method depends to a large extent on the precision of the rules. To reduce the reliance on professional engineers, the global optimization-based method is introduced. The global optimization-based method improves the real-time performance and fuel economy of EMSs to some extent, but it has more computational cost than rule-based methods [6]. The goal of the global optimization-based EMSs is to obtain the global optimal power split under a driving cycle. Due to its computational complexity and the need for a priori knowledge of the full driving cycle, the global optimization-based method is generally used as a benchmark instead of being directly applied in real-time control. The classical global optimization methods include dynamic programming algorithm (DP) [7], stochastic dynamic programming [8], genetic algorithm [9], game theory [10], pseudospectral method [11], convex optimization [12] and pontryagin minimization principle [13]. The DP-based EMSs require the most priori knowledge of the future information and have the best fuel economy, compared with all the other types of strategies. The online EMSs do not require a priori knowledge of the whole driving cycle, which are more suitable for application to HEVs, and can be further divided into instantaneous optimization-based EMSs, predictive EMSs, and learning-based EMSs. The power split is determined with optimal or sub-optimal solutions using the current feasible domain by the instantaneous optimization-based EMSs [14]. The classical instantaneous optimization-based methods include equivalent consumption minimization strategy (ECMS) [15], adaptive ECMS [16] and robust control [17]. Instead of utilizing current driving information, the predictive EMSs mainly employ future information to solve the allocation problem. Model predictive control is commonly employed to implement predictive energy management [18].

More recently, the learning-based methods become a promising solution for various problems of HEVs, leading to the independence on precise model data. The learning-based EMSs strive to update control parameters online by interacting with the environment to adapt to the changing traffic conditions. They generally employ massive historical and real-time driving-related data to obtain the optimal solution. Supervised learning algorithms train a neural network by feeding labeled data [19]. The generalization of supervised learning-based EMSs may be limited by the training data. Apart from supervised learning, reinforcement learning (RL) is another promising branch of learning-based control methods, such as Q-learning. RL algorithms learn to improve the strategy from raw observation data and reward feedback directly. However, discretization requirements of control states and action spaces in RL methods may result in the curse of dimensionality [20]. Therefore, as an alternative, current studies mainly focus on deep reinforcement learning (DRL) based EMSs due to their strong learning ability, where the EMS is modeled as a Markov Decision Process (MDP).

The optimal solution for DRL-based EMSs can be learned through the interaction between agents and the environment. Wu et al. [21] used the deep Q-learning network (DQN) algorithm for energy management, which solved the dimensional catastrophe problem. Based on this, Qi et al. [22] compared double deep Q-learning with DQN for energy management of plug-in hybrid vehicles and demonstrated advantages of their proposed method in terms of convergence and fuel economy. Wu et al. [23] verified that the energy management policy based on a deep deterministic policy gradient (DDPG) algorithm has a strong characterization capability of deep neural networks and can improve fuel economy significantly. Wang et al. [24] proposed to combine the dueling network structure with DQN so that it could be better evaluated in the presence of many similar-valued actions. Li et al. [25] took this idea and utilized it for DDPG, proposing a novel DDPG-based EMS for a power-split hybrid electric bus. But its setting of the action-value function could be further improved. In addition, Tang et al. [26] firstly adopted two distributed DRL algorithms to propose EMSs, namely asynchronous advantage actor-critic and distributed proximal policy optimization. The distributed algorithms could learn efficiently but need high-performance computer hardware.

Although DRL-based methods have made a significant breakthrough, their limitations are the long training time for an agent to learn the optimal solution through trial-and-error interactions with the environment. Besides, the training process must be repeated even when encountering a new but similar task. Therefore, some works have combined transfer learning with DRL to improve the training efficiency and reduce the computational cost among similar tasks. Liu et al. [27] combined proximal policy optimization and transfer learning to reduce time consumption and guarantee control performance effectively. Guo et al. [28] combined DDPG and transfer learning to derive an adaptive energy management controller for hybrid tracked vehicles, and showed that this method has the potential to be applied in real-world environments. Lian et al. [29] incorporated transfer learning into DDPG-based EMSs for HEVs to achieve the cross-type knowledge transfer among three types of HEVs with distinctly different structures. Zhou et al. [30] combined the DDPG algorithm and adaptive neural fuzzy inference systems (ANFIS) to enable knowledge implementation, and transfer for real-time energy management. The ANFIS-based model had real-time potential and good control performance, while the DDPG could regulate the input signals of the ANFIS and transfer knowledge online.

In all the above RL methods, a vital issue to be addressed is how the agents balance the relationship between exploration and exploitation [31]. The agent uses exploitation methods to select actions that maximize the cumulative reward, and utilizes exploration methods to acquire knowledge about the environment for selecting better actions. The e-greedy method is often used in the Q-learning algorithm to choose the control action. In every step, with probability 1 - ε, the agent fully exploits the information saved in the Q-values (action-value function), and with probability ε, the agent selects a random action to explore the state space [32]. One of the exploration methods in the DDPG algorithm is to add noise while selecting actions. The most common method is to add noise to the action space. Plappert et al. [33] proposed adding noise directly to the agent's parameter, leading to more consistent exploration. Xu et al. [34] compared effects of different types of noise on DRL-based transferable EMSs to find out the most suitable type of noise for transfer learning.

Referring to the previous research, in this work we propose a transferable energy management strategy via a dueling DDPG algorithm to address the robustness issues. Then, training weights are saved to initialize a new DDPG network. The training process after transferring is performed to acquire the optimal transferable EMS. Compared with other previous works concerning DRL-based energy management, the possible contributions of this proposed method may lie in three aspects: (1) a novel transferable EMS, which combines DDPG and transfer learning, is proposed for solid generalization ability. DDPG algorithm combines the advantages of DQN and the actor-critic architecture, which is appropriate for evaluating the strategies for network parameters transferring; (2) dueling architecture is used in the critic network of the DDPG algorithm to improve the stability of the network, which is inspired by the utilization of dueling network architecture in DQN; (3) the adaptive parameter space noise is used in our proposed method to balance exploration and exploitation. The utilization of other different noise, action space noise, is also evaluated in our simulation experiments, which indicates the out-performance of the adaptive parameter space noise.

The rest of this paper is organized as follows: Section 2 presents the HEV modeling and energy management problem formulation, Section 3 introduces the proposed method, Section 4 details the simulation results, and the conclusion and the future work are depicted in Section 5.

2. Problem description and system framework

In this section, the problems of DDPG-based EMS are analyzed from aspects of the network introduction and parameter setting. Then, the framework of the transferable EMS via improved DDPG is presented.

2.1. Problem description

The MDP, which can be characterized as $(\boldsymbol{S},\boldsymbol{A},\boldsymbol{P},R,\gamma)$, can be used to simulate a DRL problem that satisfies the Markov property. $\boldsymbol{S}$ represents a finite set of state spaces. $\boldsymbol{A}$ is a finite set of action spaces. $\boldsymbol{P}$ denotes a state transition probability matrix. R represents a reward function. γ denotes a discount factor.

As one of the most typical actor-critic DRL methods, DDPG is an off-policy and model-free algorithm. DDPG has an actor network $\mu(s|\theta^{\mu})$, a critic network $Q(s,\boldsymbol{a}|\theta^{Q})$, an actor target network $\mu'(s'|\theta^{\mu'})$, and a critic target network $Q'(s',\boldsymbol{a}'|\theta^{Q'})$. The actor target network has the same structure as the actor network, while the critic target network has the same structure as the critic network. s is the agent state as the input of actor network and critic network. $\boldsymbol{a}$ is the agent action as the output of the actor network and the input of the critic network. θ represents parameters of the corresponding network. s' and $\boldsymbol{a}'$ are defined similarly with s and $\boldsymbol{a}$ respectively. At every training interval, the parameters of the actor and critic networks are used to update the parameters of the actor target network and the critic target network respectively, to ensure the stability of the network during the training process.

The DDPG algorithm is improved and then used to learn the optimal policy of Prius EMSs in this work. The original neural network is used as the baseline in this work. The original and improved DDPG-based EMSs are formulated according to the following MDP.

(1) *Stcate space, S.* The state of the system,

$$\boldsymbol{s} = \{SoC,\ v,\ acc\} \tag{1}$$

which consists of *SoC*, the velocity of Prius v, and the acceleration *acc*, represents the movement of the vehicle and the power situation.

(2) *Action space, A.* At each episode, the agent can select actions in continuous engine torque T_{eng} and engine rotational speed n_{eng}, respectively:

$$\boldsymbol{a} = \{T_{\text{eng}},\ n_{\text{eng}}\} \tag{2}$$

(3) *Reward junction, R.* There are two aspects of the reward function, energy consumption and SoC sustaining. The multi-objective reward function is defined as:

$$r = -\{\alpha[fuel(t) + elec(t)] + \beta[SoC_{\text{ref}} - SoC(t)^n]\} \tag{3}$$

9where α is the weight of Prius consumption, including the fuel consumption of engine $fuel(t)$ and the electricity consumption of motor $elec(t)$, β is the weight of battery charge-sustaining, and SoC_{ref} represents the SoC reference value. The term $[SoC_{\text{ref}} - SoC(t)^n]$ is used to make the SoC in the training process close to the reference value. The goal of the reward function is to minimize the energy consumption, and retain the battery SoC at an appropriate range for better charge and discharge characteristics.

2.2. System framework

The framework of the proposed transferable EMS is shown as Fig. 1, which is divided into four modules, including the modelling module, the pre-training module, the transferring module, and the fine-tuning module. In the modelling module, the improved DDPG-based algorithm is designed based on the vehicle model (see Section 3.1 and Section 3.2). The adaptive noise to affect exploration and dueling network architecture to improve the robustness are considered in the algorithm. This improved algorithm is pre-trained in multiple driving cycles to acquire an energy management strategy in the source domain. Then, the weights and biases of the pre-trained networks are used to initialize a new one sharing the same architecture in the transferring module. In the end, in the different but similar driving cycles, the new network is fine-tuned instead of learning from scratch to reach convergence fast in the target domain (see Section 3.3). This work aims to design a transferable energy management strategy by combining the improved DDPG and transfer learning to enhance energy efficiency quicker and more stable.

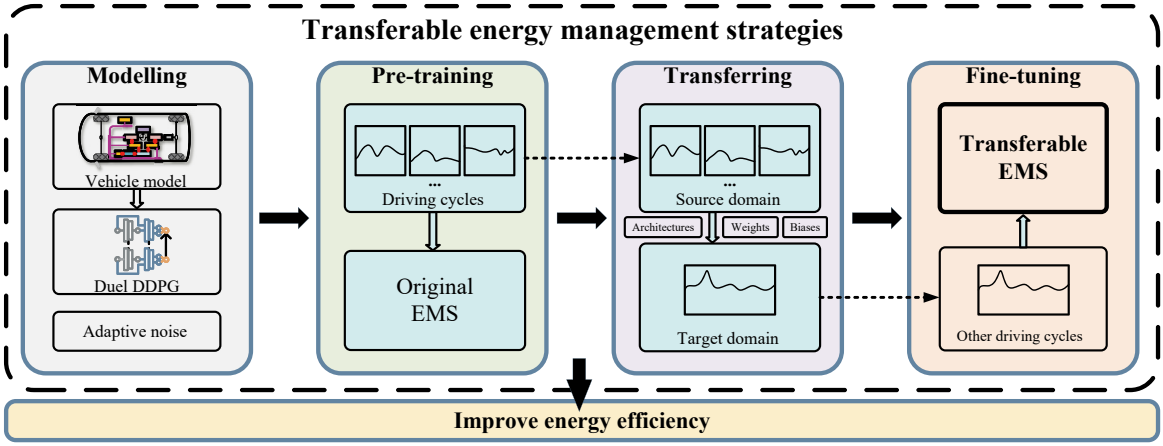


Fig. 1. Framework of the proposed EMS.

3. Transferable EMS via dueling DDPG

In this section, the proposed energy management method based on the dueling DDPG for HEVs is elaborated, as shown in Fig. 2. First, we need to model a HEV using its powertrain system and power flow configurations. Then, the proposed dueling DDPG algorithm is utilized to solve the energy management problems by decoupling the state value and the action advantage value. In the end, transfer learning is implemented for the target domain using the knowledge from the source domain.

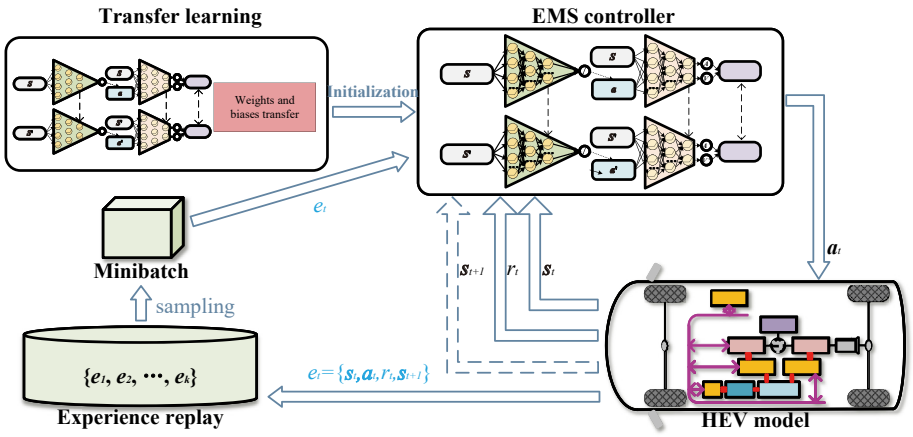


Fig. 2. Transferable EMS via dueling DDPG.

3.1. Hybrid electric vehicle modelling

The proposed transferable EMS is based on the Prius model, one of the most classical HEVs models, which has been extensively studied [29,34,35]. The power request model and the power system model based on the configuration of the vehicle model, are firstly built for our transferable EMS.

3.1.1. Prius configuration

Prius is equipped with the Hybrid Synergy Drive system, consisting of an internal combustion engine ICE, an electric motor MG2, and a generator MG1. Prius is also equipped with a low-capacity nickel-metal hydride (Ni-MH) battery used to drive the motor and generator. These systems in Fig. 3 are integrated with a power splitting planetary gear, which provides various power flow configurations for different operations.

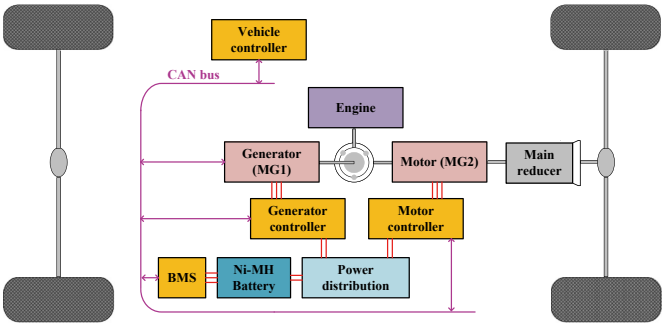


Fig. 3. Architecture of Prius powertrain.

3.1.2. Power request model

After building the Prius model, the vehicle power demand is calculated using the longitudinal force balance equation. The longitudinal force F consists of rolling resistance F_f , aerodynamic drag F_w, gradient resistance F_i and inertial force Fa [36]:

$$\begin{cases} F = F_f + F_w + F_i + F_a \\ F_f = mg \cdot f \\ F_w = \frac{1}{2}\rho \cdot A_f \cdot C_D \cdot v^2 \\ F_i = mg \cdot i \\ F_a = m \cdot a \end{cases} \tag{4}$$

where m is the curb weight, g is the gravitational constant, f is the rolling friction coefficient, ρ is the air density, A_f is the fronted area, C_D is the aerodynamic coefficient, v is the speed regarding a certain driving cycle, i is the road slope (not considered in this work), and a is the acceleration.

3.1.3. Powertrain system model

The engine, the electric motor, and the generator of the Prius are modeled by their corresponding efficiency maps from bench tests. The Ni-MH battery is modeled by an equivalent circuit model ignoring temperature changes and battery aging:

$$\begin{cases} P(t) = I(t) \cdot V_{oc}(t) - R_0 \cdot I^2(t) \\ I(t) = \dfrac{V_{oc}(t) - \sqrt{V_{oc}^2(t) - 4 \cdot R_0 \cdot P(t)}}{2R_0} \\ SoC(t) = \dfrac{Q_0 - \int_0^t I(t)\mathrm{d}t}{Q} \end{cases} \tag{5}$$

where P is the output power, I denotes the current, V_{oc} is the open-circuit voltage, R_0 is the internal resistance, SoC is the state of charge, Q_0 is the initial battery capacity, and Q is the nominal battery capacity. Details on Prius parameters are shown in Table 1.

Table 1
Parameters of Prius.

Components	Parameters	Values
Engine	Maximum power, P_e (kW)	56
	Maximum torque, T_e (Nm)	120
Motor	Maximum power, P_m (kW)	50
	Maximum torque, T_m (Nm)	400
Battery	Capacity, Q (kW)	1.54
	Voltage, V_{oc} (V)	237
Vehicle	Curb weight, m (kg)	1,449
	Roll resistance coefficient, f	0.013
	Air resistance coefficient, f_A	0.26
	Frontal area, A_f (m^2)	2.23
	Wheel radius, r (m)	0.287
Transmission	Final gear ratio, i_g	3.93
	Characteristic parameter, C	2.6

3.2. Dueling DDPG algorithm

Wang et al. [24] applied the dueling network architecture to DQN, which could lead to dramatic improvements and better policy evaluation in the presence of many similar-valued actions. Instead of directly outputting the Q estimate, this module combines two streams of fully-connected layers. This architecture decouples value and state-dependent advantage in deep Q-networks, which share a common feature learning module. As shown in Fig. 4, the estimate value of Q

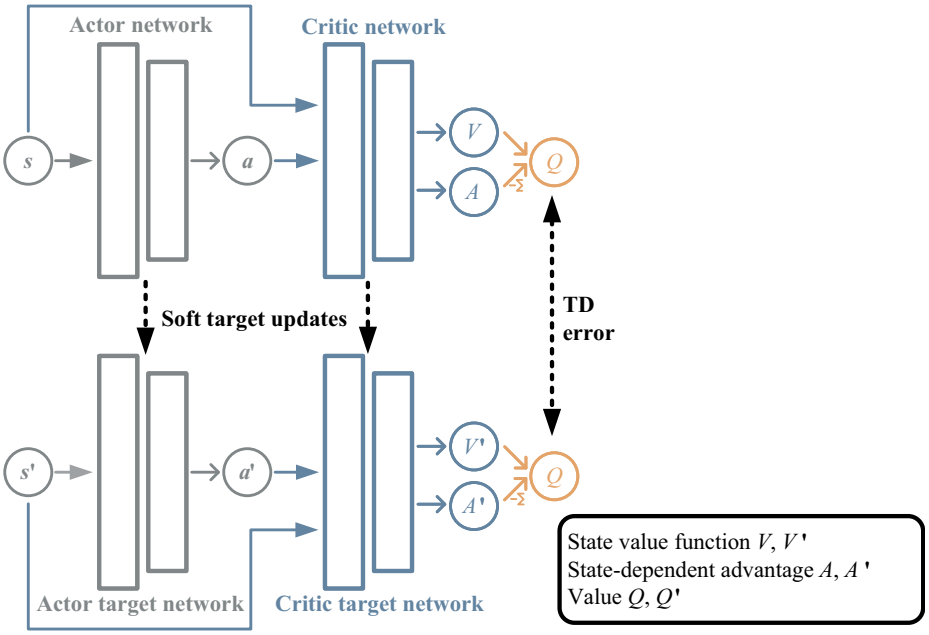


Fig. 4. Architecture of dueling DDPG.

requires the derivation of two intermediate parameter functions, the state value function $V(s,a)$ and the state-dependent action advantage function $A(s,a)$. The dueling architecture can separate learning of the state value and action advantages, promoting learning efficiency.

Since the learning of the optimal EMS with DDPG is realized by the update of network parameters and the critic network Q is built to guide the strategy learning, this dueling network is utilized in the critic network for the DDPG-based EMS [25]. Its action-value function (forward propagation of critic network) is expressed as:

$$Q(s,a) = V(s,a) + A(s,a) \tag{6}$$

However, there is unidentifiable that a given Q value cannot recover V and A uniquely. If δ denotes a constant, for the above equation, $V(s, a) = V, A(s,a) = A$ and $V(s,a) = V + \delta, A(s,a) = A - \delta$ lead to the same Q value $(V + A)$. This uncertainty will affect the performance of the network training.

To address this issue, we adopted the method proposed in [24]:

$$Q(s,a) = V(s,a) + \left[A(s,a) - \frac{1}{|\mathscr{A}|}\sum A(s,a)\right] \tag{7}$$

where $|\mathscr{A}|$ denotes the dimensional value of the agent action. On the one hand, this structure decreases the uncertainty of network parameters, and on the other hand, it increases the stability of the optimization.

3.3. Transfer learning

Traditional DRL algorithms are used to solve the problem with training and test data in the same domain. However, once the domain is changed, the network must be retrained, which is a complicated and time consuming process. Transfer learning is instrumental in solving this problem. When two domains are similar, network parameters can be stored and reused along with transfer learning approaches.

Given a source domain M_s and a target domain M_t, set the knowledge that can be gained from M_s to be D_s. M_s provides priori knowledge D_s that is accessible for M_t. The goal of transfer learning is to learn an optimal policy π^* for M_t by fully leveraging D_s. Thus, with transfer learning, the agent of the target domain learns better and faster in M_t [37].

A network that specializes in obtaining the source EMS is used in our work. Since driving cycles of M_s and M_t have the same feature space and are correlated with each other, source domain knowledge can be transferred to a novel but relevant target domain [28]. The majority of parameters in the neural network are the same, and only parameters of the output layer should be retrained. Thus, both the source network and the target network use the same dueling DDPG architecture shown in Fig. 4, and weights and biases of the source network except for the last layer are used to initialize the target network that will be trained on the new driving cycles.

In DDPG, the agent utilizes exploration to acquire knowledge about the environment and applies exploitation to select a control action based on current knowledge. During the transfer process, to coordinate between exploitation and exploration and reach the convergence value faster, the adaptive noise is added to the proposed algorithm. The methods to add noise in networks can be divided into two main categories: adding noise in the action space, and adding noise directly to the agent's parameters.

(1) Action space noise

When the agent selects actions using the actor network, the noise $\mathscr{N}$ is added to the action space. The final selected action a_t at each step satisfies:

$$a_t = \mu(s|\theta^\mu) + \mathscr{N} \tag{8}$$

Action space noise could be a simple Gaussian noise or a more advanced Ornstein-Uhlenbeck (OU) correlated noise process [31]. Gaussian noise satisfies $\mathscr{N} \sim \mathscr{N}(0, \sigma^2 I)$, where σ^2 denotes variance and the expected value is set to 0. An OU process can be used as a temporally correlated noise. Just like the Gaussian noise mentioned above, the expected value of OU noise $\mathscr{N} \sim \mathscr{N}(0, \sigma^2)$ is set to 0, and the variance can be set to multiple values.

(2) Parameter space noise

While adding noise in the action space to explore, there is no guarantee that the same action will be chosen in the same state each time, leading to inconsistent exploration. The parameter space noise solves this problem and directly perturbs the actor network parameters to get a rich set of behaviors. The final selected action a_t at each step satisfies:

$$\begin{cases} a_t = \mu\left(s\middle|\tilde{\theta}^{\mu}\right) \\ \tilde{\theta} = \theta + \mathcal{N}(0, \sigma^2 I) \end{cases} \tag{9}$$

Following the results of the comparison in [34], in this work, the parameter space noise is added in the process of action choosing. Besides, this work adjusts the scale of the parameter space noise over time, inspired by [33]. This is achieved by updating the variance of noise σ accordingly, which satisfies:

$$\sigma_{t+1} = \begin{cases} \gamma\sigma_t, & \text{if } d(\pi, \tilde{\pi}) \le \delta \\ \frac{1}{\gamma}\sigma_t, & \text{if } d(\pi, \tilde{\pi}) \rangle \delta \end{cases} \tag{10}$$

where γ is the threshold value, π and $\tilde{\pi}$ denote the non-perturbed and perturbed policies, and $\mathrm{d}(\pi, \tilde{\pi})$ represents the distance between π and $\tilde{\pi}$, which is calculated in the action space.

4. Simulation results

The experimental simulation is designed to showcase the performance of the proposed approach and to evaluate the claims that: (1) our proposed transferable EMS has better generalization capability and a more efficient training process for different driving cycles; (2) the dueling network architecture added in our proposed approach outperforms the original DDPG in both the source and target domains; (3) the adaptive parameter space noise adopted in our proposed method can promote exploration by agents and remain comparably stable after convergence.

4.1. Implementation and experimental setup

In this work, driving cycles are all selected from standard data, following the research on [34]. Source tasks are performed over multiple driving cycles, such as Urban Dynamometer Driving Schedule (UDDS) [38], WVUSUB [36], JN1015 [29]. Target tasks are conducted on European Driving Cycle (NEDC) - LA92 [32], which is different from driving cycles used in the source domain. To further validate the effectiveness of our method on the data in the real environments, we collect the velocity changes using our own mobile platform in the real world, which is named as the real-world driving cycle (RWDC). The velocity changes and distributions of these driving cycles are illustrated in Figs. 5 and 6 and Figs. 7 and 8. As can be seen, velocities of source driving cycles mostly distribute below 20 m/s, and JN1015 has an obvious regularity. In contrast, the target driving cycle, NEDC-LA95, has more velocities over 25 m/s than the source driving cycles, and combines regular velocity segments and irregular ones, which shows a more complicated driving situation. Besides, the velocities of the other target driving cycle RWDC mainly distribute between 20 m/s and 25 m/s. Using a wide range of driving cycles for training in the source domain improves the generalization ability of the trained model, which leads to better transfer results. A driving cycle with a degree of similarity to the source driving cycles is chosen for the target domain, since similarity is necessary for transfer learning.

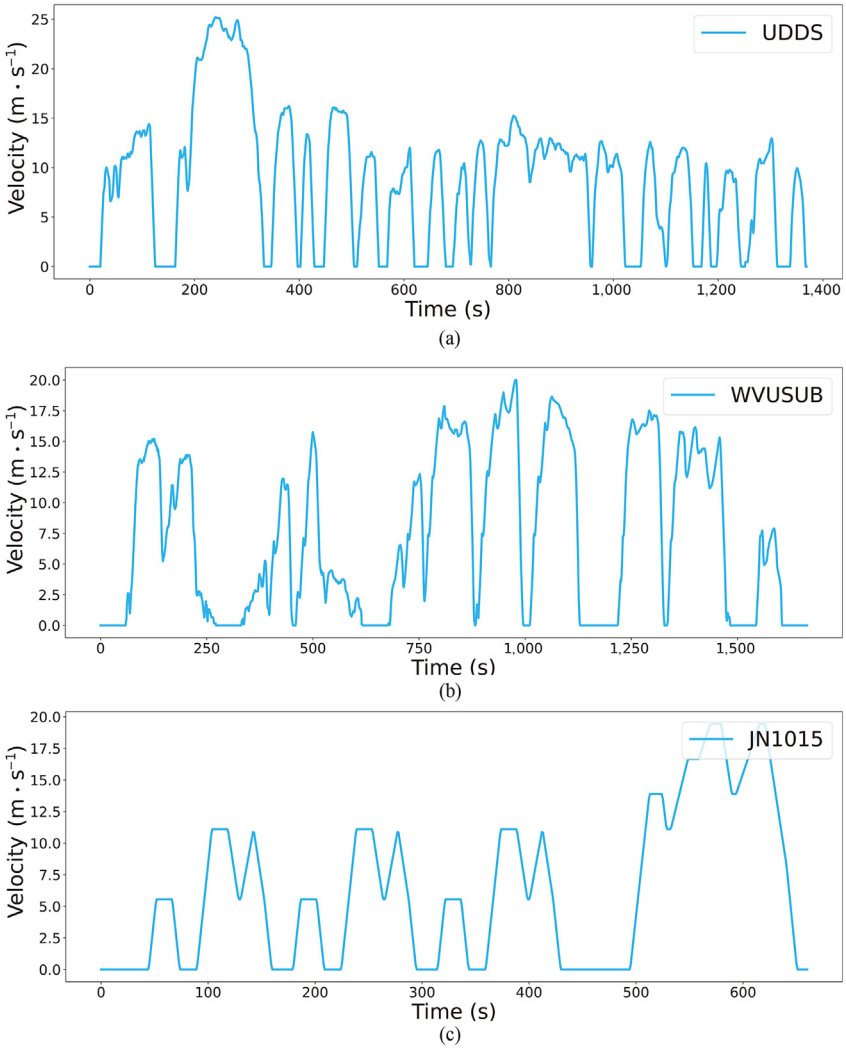


Fig. 5. Velocity changes of driving cycles used in the source domain. (a) Velocity change of UDDS. (b) Velocity change of FTP75. (c) Velocity change of JN1015.

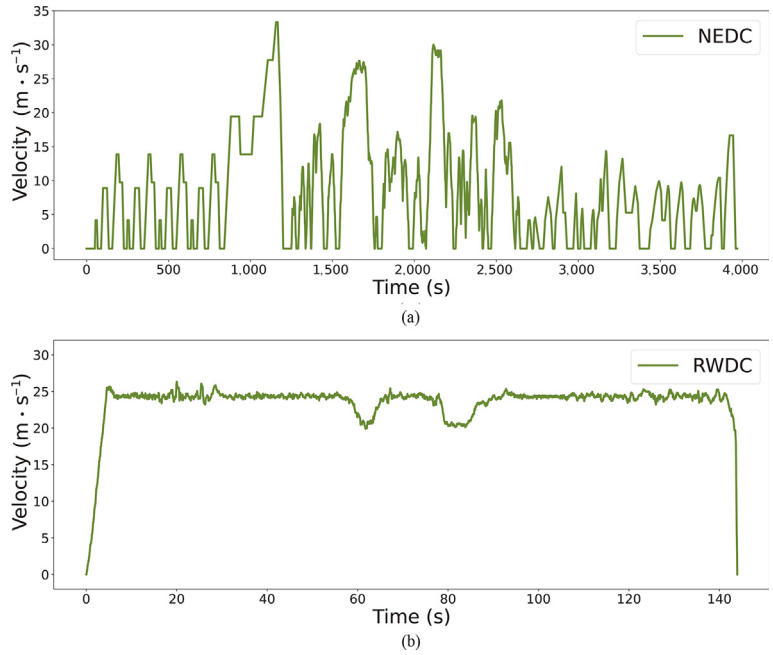


Fig. 6. Velocity changes of driving cycles used in the target domain.(a) Velocity change of NEDC LA92. (b) Velocity change of RWDC.

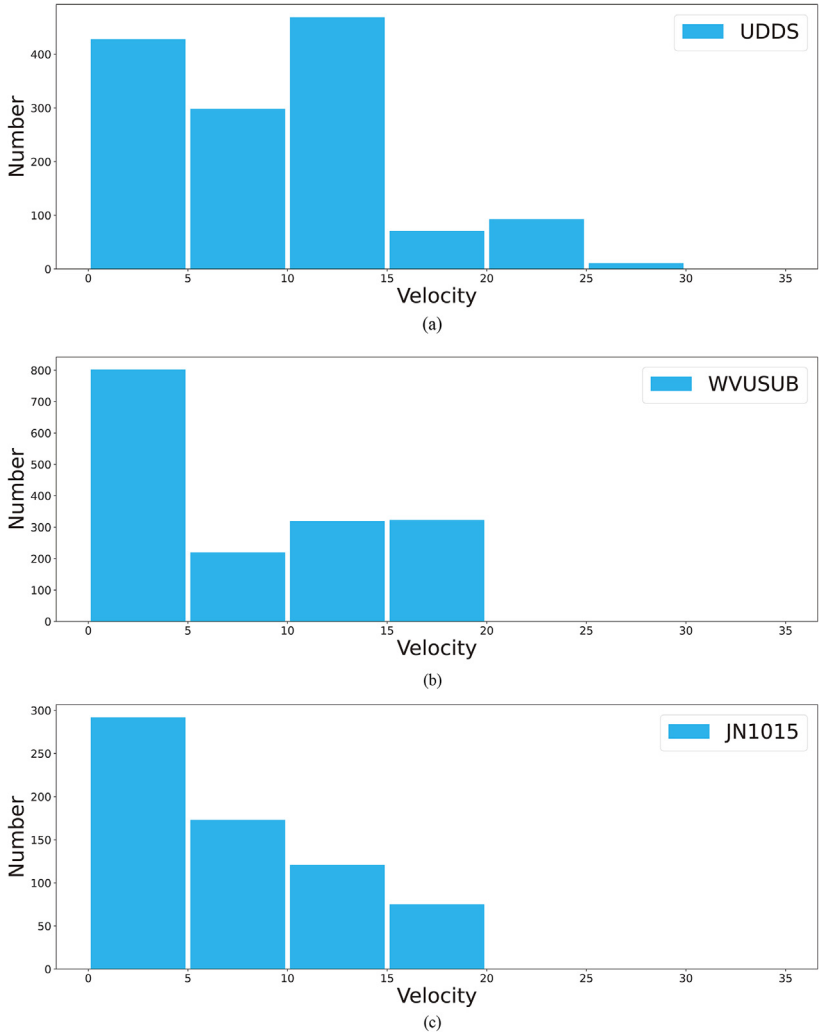


Fig. 7. Velocity distributions of driving cycles used in the source domain. (a) Velocity distribution of UDDS. (b) Velocity distribution of FTP75. (c) Velocity distribution of JN1015.

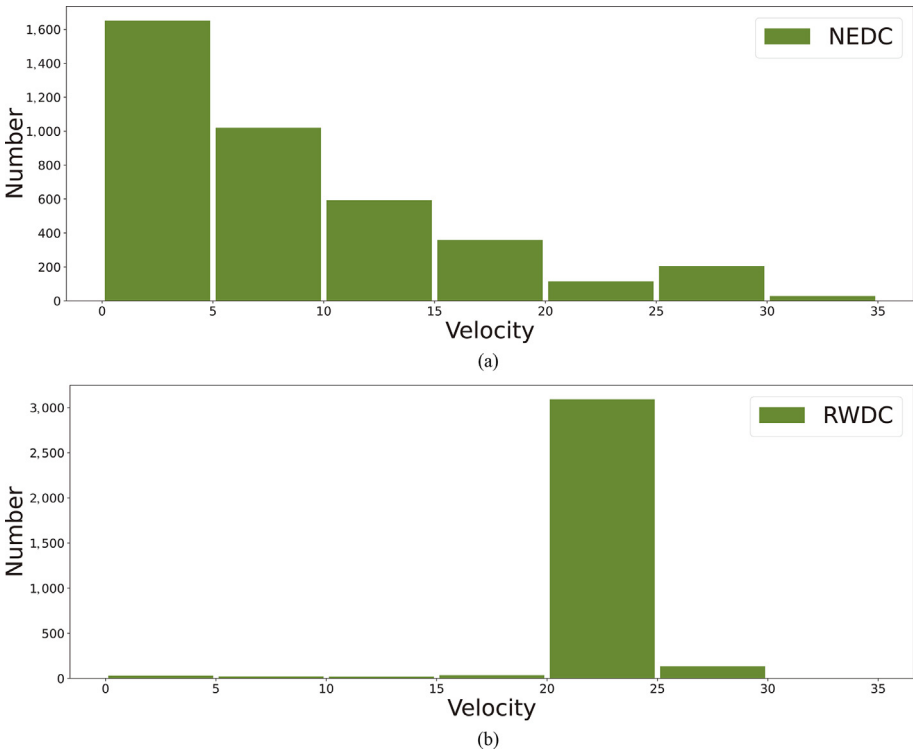


Fig. 8. Velocity distributions of driving cycles used in the target domain. (a) Velocity distribution of NEDC LA92. (b) Velocity distribution of RWDC.

In the following experiments, some of the parameters involved in the MDP are set to constant values to control the variables. SoC_{ref} is selected as 0.6 according to the minimum charge–discharge internal resistance. α is selected as 1, β is set to 350, and n is set to 2, according to the previous work [36].

Note that to highlight the effect of the proposed algorithm, the number of hidden layers of the network for DDPG is set to 2, rather than many layers. Further details about hyperparameters of DDPG in M_s and domain M_t are given in Table 2.

Table 2
Network hyperparameters.

Parameters	Source domain	Target domain
Number of episodes (K)	1000	300
Replay memory size (M)	50,000	50,000
Learning rate of actor network (lr_a)	0.000,1	0.000,01
Learning rate of critic network (lr_c)	0.000,1	0.000,01
Discount factor (γ)	0.9	0.9
Target network update frequency (τ)	0.01	0.01
Mini-batch size ($batch$)	64	64

Besides, following the study of [34], we keep the parameter settings for both parameter space noise and action space noise in the source and target domains. The expected values of both noises are set to 0. The variance of the action space noise is set to 0.06, while the variance of the parameter space noise is set to 0.03. Adaptive noise is used so that the noise can be better adapted to the learning of the network. In this case, γ is selected as 1.01 so that the network could realize better exploitation and exploration. Except for the evaluation of different noises, the adaptive parameter space noise is used in our method and the other baseline methods.

For each experiment, we use the average reward value (Ave), the maximum reward value (Max), the minimum reward value (Min), the reward value of the initial episode (Init), and the iteration episode number for convergence (Iter) as metrics to evaluate the performance of our proposed method and all the baseline methods. Energy consumption ratio with DP as a benchmark (ECR) is also calculated following the previous work [29] to show the better capability of different methods in energy consumption control.

4.2. Evaluation for transferable training

In the first simulation, we validate our first claim that our proposed transferable EMS has the better generalization capability and a more efficient training process for different driving cycles. We compare four methods in the target domain NEDC, including the original DDPG learning from scratch (O-DDPG-LFS), the dueling DDPG learning from scratch (D-DDPG-LFS), the transferred original DDPG initialized from the source domain (O-DDPG-Trans), and the transferred dueling DDPG initialized in the source domain (D-DDPG-Trans). All these methods are trained or fine-tuned in the target domain. The simulation results are shown in Fig. 9 and Table 3. As can be seen in Fig. 9(a) and (b), O-DDPG-Trans costs less time for convergence than O-DDPG-LFS, but the mean reward of O-DDPG-Trans fluctuates in the last 10 episodes. In addition, Fig. 9(c) and (d) show that D-DDPG-Trans outperforms D-DDPG-LFS on the efficiency of converging for training, which only uses around 30 training episodes to converge. The simulation results in Table 3 also show that the transferred methods fine-tuned in the target domain have better performance on all the metrics than the methods learning from scratch. Note that the initial performance of agents using transfer learning is much better than networks learning from scratch, which indicates that transferable training improves the generalization of the network and allows for fast convergence in new driving cycles, which is more advantageous than normal training. At the same time, this supports the real-time nature of the proposed EMS. In addition, the comparison of energy consumption indicates that D-DDPG-LFS has the best energy consumption control ability since it is trained directly in the target domain. The ECR of D-DDPG-Trans is similar to D-DDPG-LFS, and is lower than the

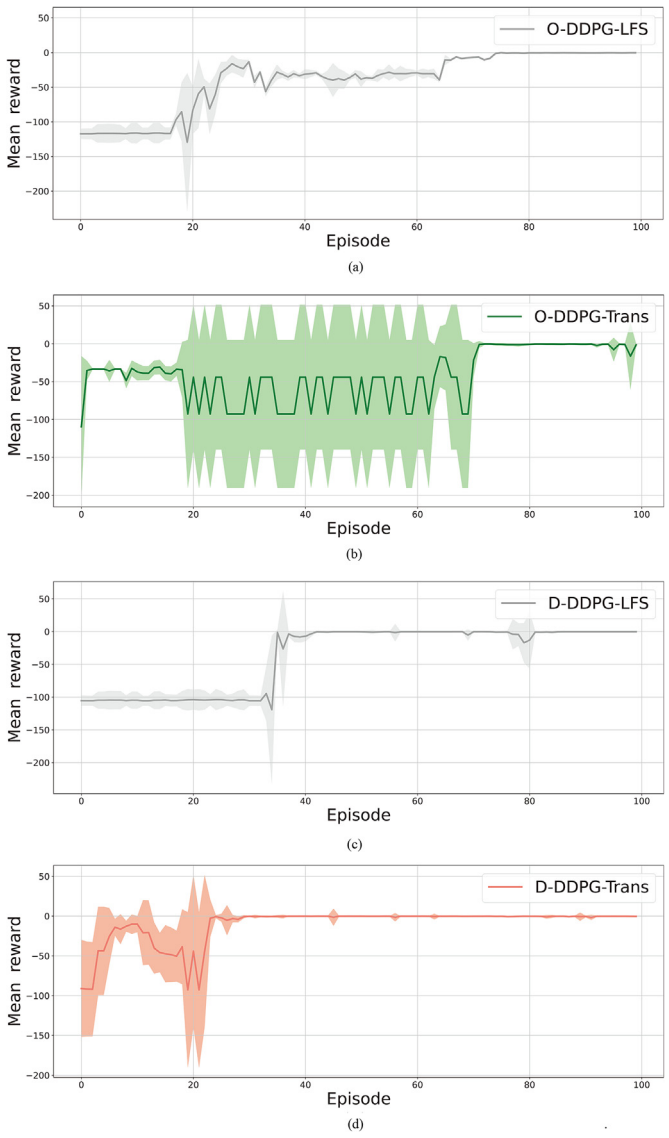


Fig. 9. Comparison of DDPG with and without dueling architecture in the target domain. (a) Original DDPG learning from scratch(O-DDPG-LFS). (b) Transferred original DDPG initialized from the source domain (O-DDPG-Trans). (c) Dueling DDPG learning from scratch(D-DDPG-LFS). (d) The transferred dueling DDPG initialized from the source domain (D-DDPG-Trans).

ones of O-DDPG-Trans and O-DDPG-LFS, indicating a better transferable performance of DDPG with dueling architecture than original DDPG.

In Fig. 10, we also illustrate the SoC change of different methods. Recall that we set the SoC reference value to 0.6, so the final values of SoC in the training processes are all around 0.6. As can be seen in Fig. 10(a) and (c), the SoC of our dueling DDPG learning from scratch is faster than O-DDPG-LFS to maintain around 0.6. Besides in Fig. 10(b) and (d), in contrast to the methods learning from scratch, the SoC of the two transferable DDPG-based methods change from the beginning. The SoC change of D-DDPG-Trans is more stable than O-DDPG-Trans, and is maintained around 0.6 faster. The reason could be that our dueling network architecture utilizes an improved action-value function to decrease the uncertainty.

To further evaluate the performance of the DDPG with dueling architecture in the real-world application, we conduct the simulation on the real-world driving cycle RWDC, and the results are shown in Figs. 11 and 12. The mean reward of D-DDPG-Trans convergences faster than D-DDPG-LFS, which indicates that the model pre-trained on the standard driving cycles leads to an effective transfer for real-world data. In

Table 3
Comparison of methods with and without the dueling architecture.

Approach	w/o dueling O-DDPG-Pre					w/dueling D-DDPG-Pre				
Before transferring	Ave −38.01	Max **−0.12**	Min init −99.36–**56.58**	Iter 73	ECR N/A	Ave **−25.88**	Max −0.13	Min −99.36	Init −56.61	Iter **56**	ECR N/A
	O-DDPG-Trans					**D-DDPG-Trans**					
After transferring	Ave −40.86	Max −0.19	Min init −109.55–109.55	Iter 71	ECR 86.10	Ave **−10.79**	Max −0.11	Min −92.82	Init **−90.92**	Iter **24**	ECR 84.62
	O-DDPG-LFS					**D-DDPG-LFS**					
Without transferring	Ave −42.33	Max −0.23	Min init −129.52–117.33	Iter 75	ECR 85.03	Ave −32.89	Max −0.09	Min −119.40	Init −105.30	Iter 42	ECR **84.44**

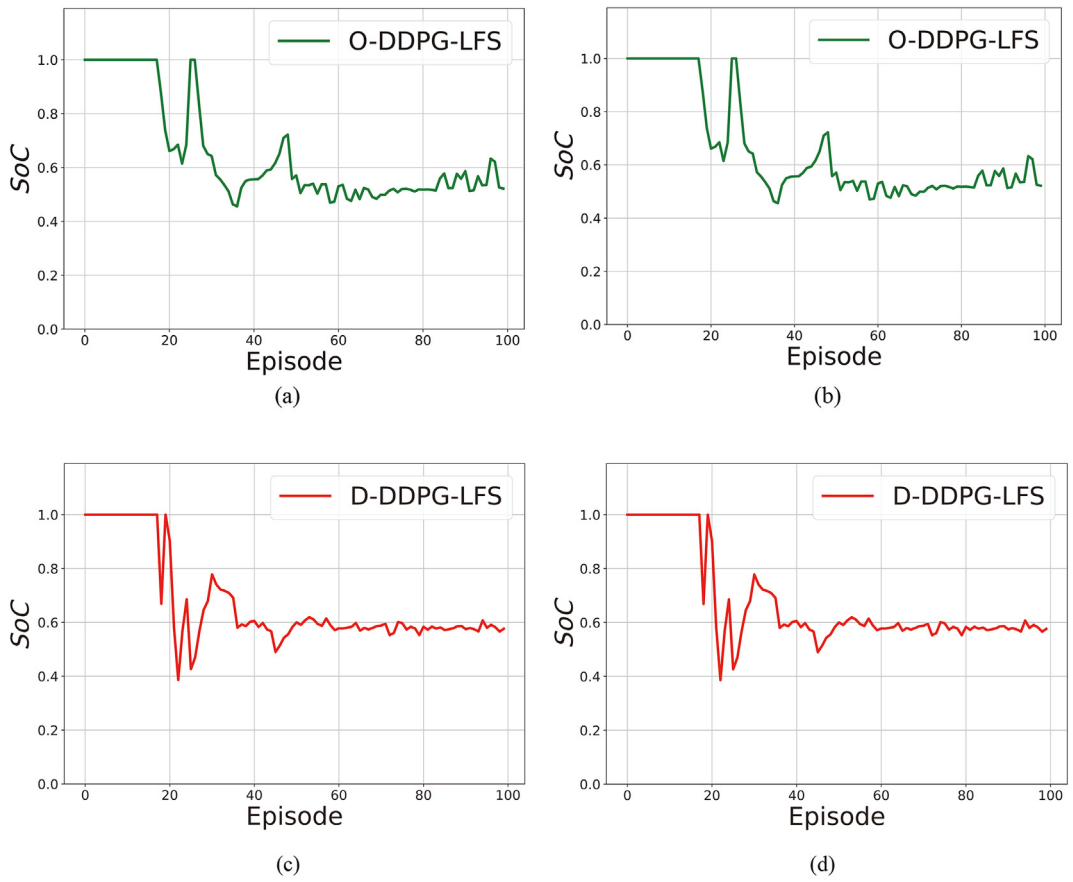


Fig. 10. Comparison of *SoC* of DDPG with and without dueling architecture in the target domain. (a) *SoC* of O-DDPG-LFS. (b) *SoC* of O-DDPG-Trans. (c) *SoC* of D-DDPG-LFS. (d) *SoC* of D-DDPG-Trans.

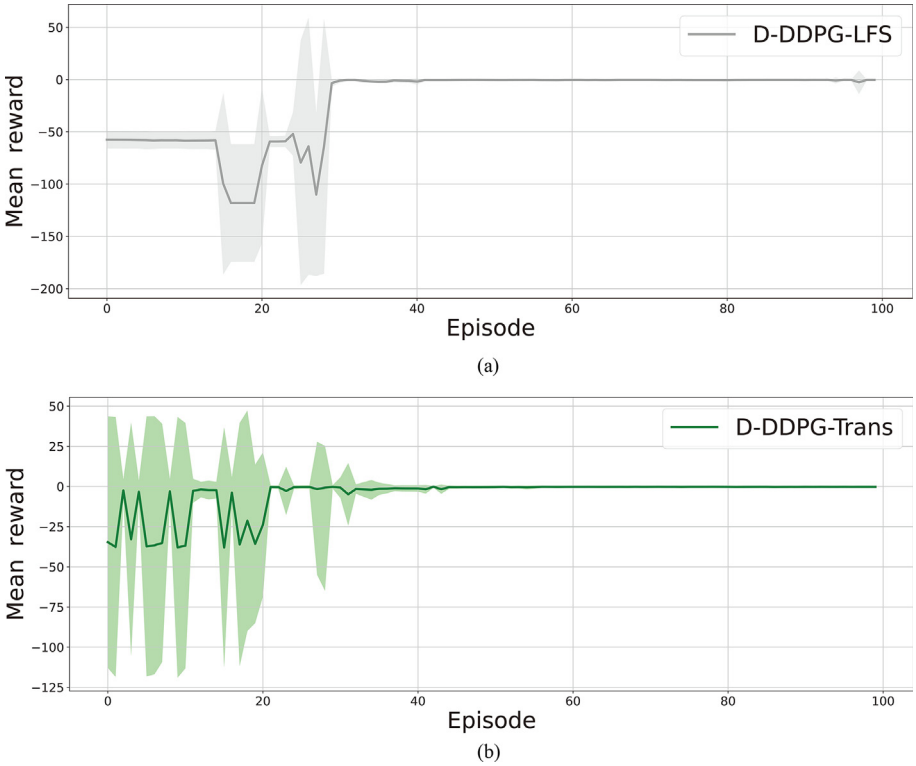


Fig. 11. Comparison of DDPG with and without dueling architecture on the real-world data. (a) D-DDPG-LFS on RWDC. (b) D-DDPG-Trans on RWDC.

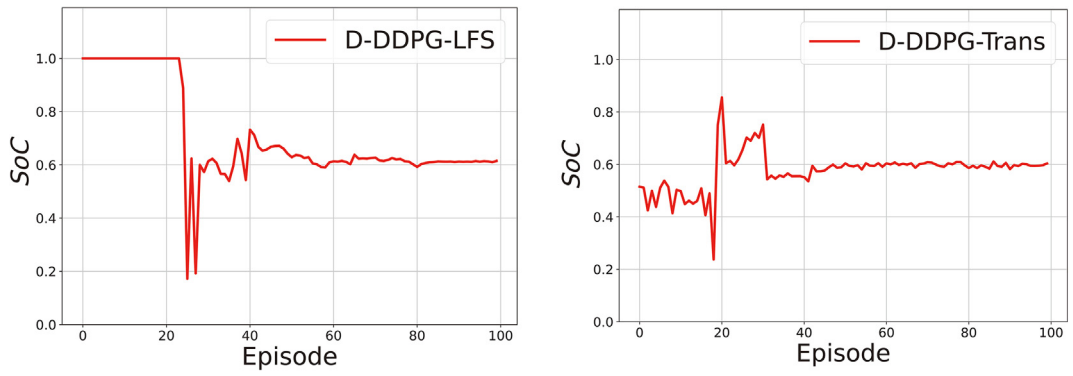


Fig. 12. Comparison of *SoC* of DDPG with and without transferring on the real-world data. (a) *SoC* of D-DDPG-LFS. (b) *SoC* of D-DDPG-Trans.

addition, we illustrate the SoC changes of these two methods. As can be seen, the SoC of D-DDPG-Trans has a lower initial value and converges faster than D-DDPG-LFS, which also shows our proposed algorithm is transferable to real applications.

4.3. Evaluation for dueling architecture

The second simulation supports our claim that the dueling network architecture added in our proposed approach outperforms the original DDPG in both the source and target domains. Following the experimental setup in the source domain, we train and evaluate our approach and original DDPG algorithm. The simulation results are shown in Fig. 13.

As can be seen in Fig. 13(a), the original DDPG pre-trained in the source domain (O-DDPG-Pre) falls into a local optimum solution initially and fluctuates more during the training process. In contrast, as Fig. 13(b) shows, our proposed method, the dueling DDPG pre-trained in the source domain (D-DDPG-Pre), can achieve the fast convergence even though the network is comparably small. By decoupling the state value function and the state-dependent action advantage function, the learning process of D-DDPG-Pre can be more stable than O-DDPG-Pre. In addition, we compare the SoC change of these two methods in Fig. 14. The SoC of D-DDPG-Pre converges faster and fluctuates less than O-DDPG-Pre before convergence.

We also compare O-DDPG-Trans with D-DDPG-Trans to further validate the effect of the dueling architecture in the target domain. The target network has the same architecture as the source domain network and uses the weights trained in the source domain to initialize the target network. As we can see in Fig. 9(b) and (d), in the target domain, O-DDPG-Trans still falls into a local optimum and converges much slower than D-DDPG-Trans. In addition, O-DDPG-Trans is very volatile and unstable. Note that fluctuations of the training process of D-DDPG-Trans

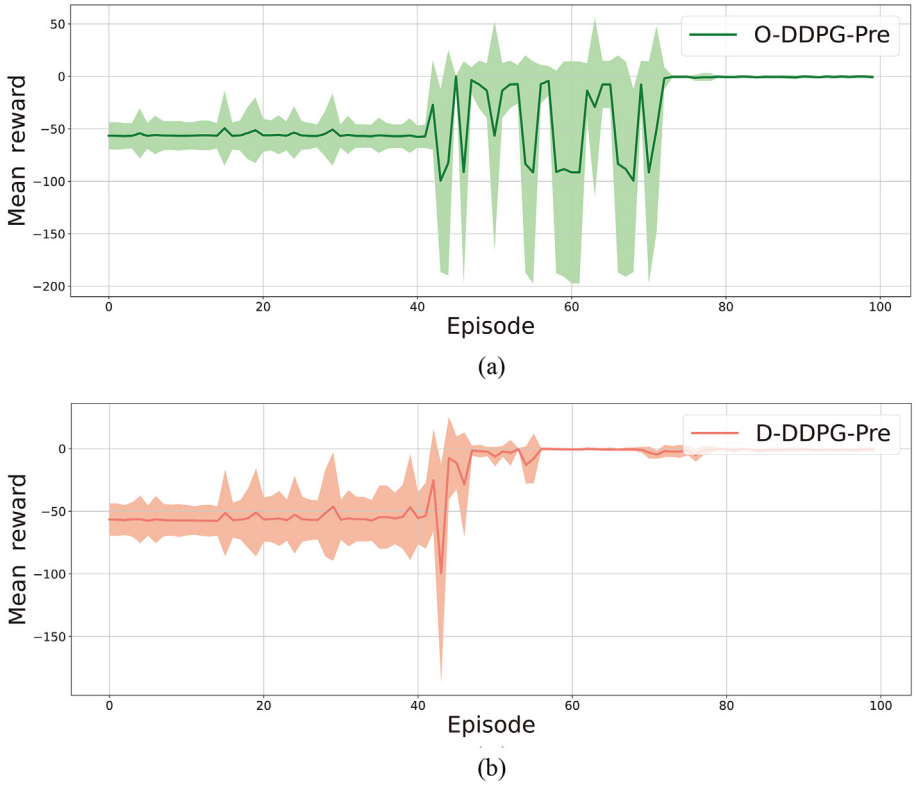


Fig. 13. Comparison of DDPG with and without dueling architecture in the source domain. (a) Original DDPG pre-trained in the source domain (O-DDPG-Pre). (b) Dueling DDPG pre-trained in the source domain (D-DDPG-Pre).

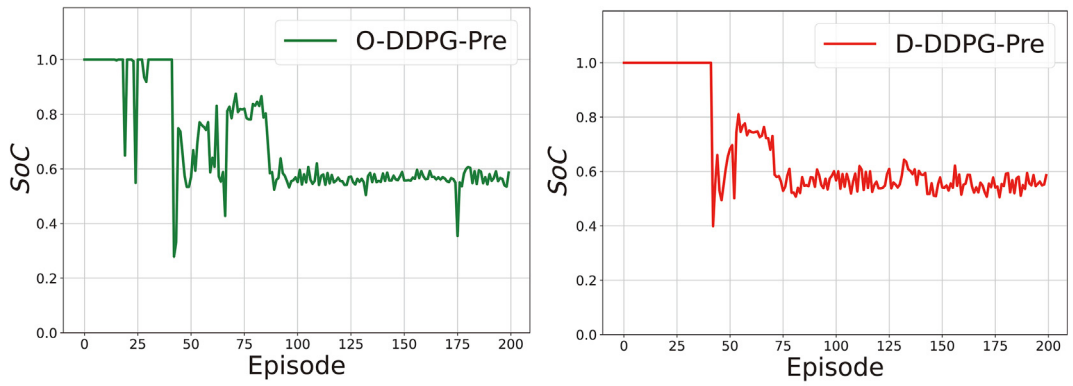


Fig. 14. Comparison of the *SoC* of DDPG with and without dueling architecture in the source domain. (a) *SoC* of O-DDPG-Pre. (b) *SoC* of D-DDPG-Pre.

before convergence are not small enough, but it shows that the exploration tends to learn a better strategy as soon as possible, rather than training around non-optimal reward values for a long time as in the case of O-DDPG-Trans.

Besides, we compare the energy consumption of O-DDPG-Trans and D-DDPG-Trans. The energy consumption is composed of fuel consumption and electricity consumption. Following the previous work [29], we transform the two kinds of consumption into CNY cost as the uniform criterion. As Fig. 15 shows, D-DDPG-Trans keeps less energy consumption during the whole training process, and is faster to converge than O-DDPG-Trans (lr = 1e-5) by around 42 episodes. Note that we keep the same learning rate 1e-5 used for all the training processes in the target domain as Table 2 shows. Therefore, the cost of O-DDPG-Trans (lr = 1e-5) has a more frequent fluctuation than D-DDPG-Trans while exploiting the consistent chosen learning rate. We further adjust the learning rate of O-DDPG-Trans to 2e-5 and 5e-6, and the results show that D-DDPG-Trans still outperforms O-DDPG-Trans.

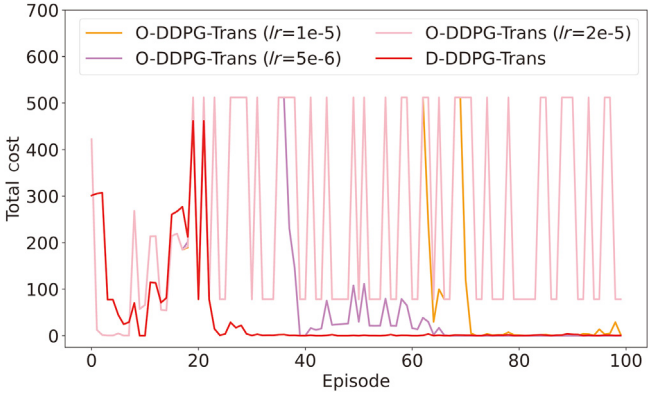


Fig. 15. Comparison of the *SoC* of DDPG with and without dueling architecture in the source domain.

4.4. Evaluation for different noises

In the third simulation, we validate that our proposed method can converge faster by adding the adaptive parameter space noise. After initializing the weights of the network in the target domain, different types of noise are added. Fig.16 shows the simulation results of adding action space noise to the dueling DDPG. The training process of the network still fluctuates a lot and fails to converge after a long time of training, indicating action space noise is inappropriate for these driving cycles and leads to a worse exploration of the agent significantly. The simulation results of adding the adaptive parameter space noise to the dueling DDPG, D-DDPG-Trans, are shown in Fig. 9(d). As can be seen, the adaptive parameter space noise leads to a rapid exploration of the agent when the network has not yet converged, and remains relatively stable after convergence. The simulation results indicate that compared with the action space noise, the adaptive parameter space noise based on the difference calculated in the action space can effectively balance the exploitation and exploration during training.

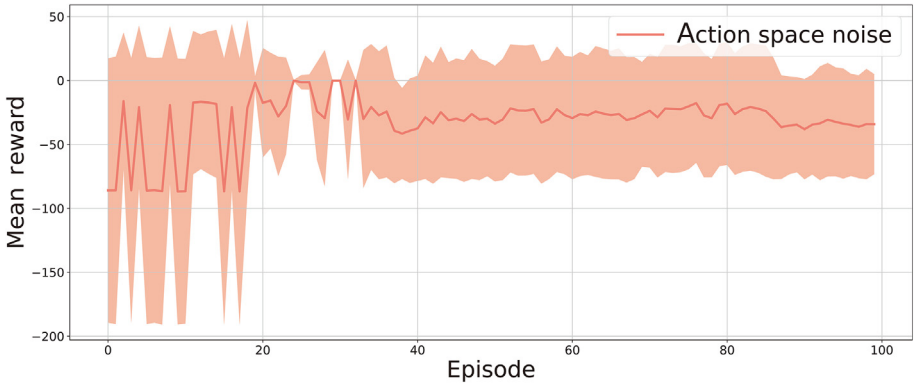


Fig. 16. Dueling DDPG with action space noise.

5. Conclusion

In this paper, we presented a novel real-time four-phase approach for the transferable EMS via dueling DDPG. Our approach utilizes the dueling DDPG for faster convergence, and uses transfer learning to achieve the stability and generalization capability of the algorithms. The adaptive parameter space noise can facilitate agents to explore new actions and fully use valuable ones after convergence. We evaluated the performance of our method on multiple source driving cycles and the similar but different target driving cycles. The experimental results suggest that our method outperforms the other methods in terms of fast converging performance and generalizes well.

Despite these encouraging results, there are several avenues for future research. First, we want to experiment with other algorithms such as twin delayed DDPG and soft actor-critic, which are not sensitive to network parameters. We furthermore plan to make more use of knowledge from the source domain in the target domain.

Declaration of competing interest

All authors disclosed no relevant relationship.

References

[1] Liu T, Tan W, Tang X, Zhang J, Xing Y, Cao D. Driving conditions-driven energy management strategies for hybrid electric vehicles: a review. Renew Sustain Energy Rev 2021;151:111521.

[2] Zhang F, Wang L, Coskun S, Pang H, Cui Y, Xi J. Energy management strategies for hybrid electric vehicles: review, classification, comparison, and outlook. Energies 2020;13(13):3352.

[3] Guo H, Wang X, Li L. State-of-charge-constraint-based energy management strategy of plug-in hybrid electric vehicle with bus route. Energy Convers Manag 2019;199:111972.

[4] Tian H, Wang X, Lu Z, Huang Y, Tian G. Adaptive fuzzy logic energy management strategy based on reasonable soc reference curve for online control of plug-in hybrid electric city bus. IEEE Trans Intell Transport Syst 2017;19(5):1607–17.

[5] Sun C, Moura SJ, Hu X, Hedrick JK, Sun F. Dynamic traffic feedback data enabled energy management in plug-in hybrid electric vehicles. IEEE Trans Control Syst Technol 2014;23(3):1075–86.

[6] Johri R, Filipi Z. Optimal energy management of a series hybrid vehicle with combined fuel economy and low-emission objectives. Proc Inst Mech Eng - Part D J Automob Eng 2014;228(12):1424–39.

[7] Wang X, He H, Sun F, Zhang J. Application study on the dynamic programming algorithm for energy management of plug-in hybrid electric vehicles. Energies 2015;8(4):3225–44.

[8] Zou Y, Kong Z, Liu T, Liu D. A real-time Markov chain driver model for tracked vehicles and its validation: its adaptability via stochastic dynamic programming. IEEE Trans Veh Technol 2016;66(5):3571–82.

[9] Lu X, Wu Y, Lian J, Zhang Y, Chen C, Wang P, et al. Energy management of hybrid electric vehicles: a review of energy optimization of fuel cell hybrid power system based on genetic algorithm. Energy Convers Manag 2020;205:112474.

[10] Dextreit C, Kolmanovsky IV. Game theory controller for hybrid electric vehicles. IEEE Trans Control Syst Technol 2013;22(2):652–63.

[11] Zhou W, Zhang C, Li J, Fathy HK. A pseudospectral strategy for optimal power management in series hybrid electric powertrains. IEEE Trans Veh Technol 2015;65(6):48134825.

[12] Nuesch T, Elbert P, Flankl M, Onder C, Guzzella L. Convex optimization for the energy management of hybrid electric vehicles considering engine start and gearshift costs. Energies 2014;7(2):834–56.

[13] Nguyen B-H, German R, Trovao JPF, Bouscayrol A. Real-time energy management of bat- tery/supercapacitor electric vehicles based on an adaptation of pontryagin's minimum principle. IEEE Trans Veh Technol 2018;68(1):203–12.

[14] Hofman T, Steinbuch M, Van Druten R, Serrarens A. Rule-based energy management strategies for hybrid vehicles. Int J Electr Hybrid Veh (IJEHV) 2007;1(1):71–94.

[15] Rezaei A, Burl JB, Zhou B. Estimation of the ecms equivalent factor bounds for hybrid electric vehicles. IEEE Trans Control Syst Technol 2017;26(6):2198–205.

[16] Zhang F, Liu H, Hu Y, Xi J. A supervisory control algorithm of hybrid electric vehicle based on adaptive equivalent consumption minimization strategy with fuzzy pi. Energies 2016;9(11):919.

[17] Morales-Morales J, Cervantes I, Cano-Castillo U. On the design of robust energy management strategies for fchev. IEEE Trans Veh Technol 2014;64(5):1716–28.

[18] Huang Y, Wang H, Khajepour A, He H, Ji J. Model predictive control power management strategies for hevs: a review. J Power Sources 2017;341:91–106.

[19] Chen Z, Mi CC, Xu J, Gong X, You C. Energy management for a power-split plug-in hybrid electric vehicle based on dynamic programming and neural networks. IEEE Trans Veh Technol 2013;63(4):1567–80.

[20] Vazquez-Canteli JR, Nagy Z. Reinforcement learning for demand response: a review of algorithms and modeling techniques. Appl Energy 2019;235:1072–89.

[21] Wu J, He H, Peng J, Li Y, Li Z. Continuous reinforcement learning of energy management with deep q network for a power split hybrid electric bus. Appl Energy 2018;222:799–811.

[22] Qi X, Luo Y, Wu G, Boriboonsomsin K, Barth M. Deep reinforcement learning enabled self-learning control for energy efficient driving. Transport Res C Emerg Technol 2019;99:67–81.

[23] Wu Y, Tan H, Peng J, Zhang H, He H. Deep reinforcement learning of energy management with continuous control strategy and traffic information for a series-parallel plug-in hybrid electric bus. Appl Energy 2019;247:454–66.

[24] Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N. Dueling network architectures for deep reinforcement learning. In: International conference on machine learning. PMLR; 2016. p. 1995–2003.

[25] 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.

[26] Tang X, Chen J, Liu T, Qin Y, Cao D. Distributed deep reinforcement learning-based energy and emission management strategy for hybrid electric vehicles. IEEE Trans Veh Technol 2021;70(10):9922–34.

[27] Liu T, Wang B, Tan W, Lu S, Yang Y. Data-driven transferred energy management strategy for hybrid electric vehicles via deep reinforcement learning. 2020. arXiv preprint arXiv:2009.03289.

[28] Guo X, Liu T, Tang B, Tang X, Zhang J, Tan W, et al. Transfer deep reinforcement learning- enabled energy management strategy for hybrid tracked vehicle. IEEE Access 2020;8. 165837165848.

[29] 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 Veh Technol 2020;69(8):8367–80.

[30] Zhou Q, Zhao D, Shuai B, Li Y, Williams H, Xu H. Knowledge implementation and transfer with an adaptive learning network for real-time power management of the plug-in hybrid vehicle. IEEE Transact Neural Networks Learn Syst 2021;32(12):5298–308.

[31] Colas C, Sigaud O, Oudeyer P-Y. Gep-pg: decoupling exploration and exploitation in deep reinforcement learning algorithms. In: International conference on machine learning. PMLR; 2018. p. 1039–48.

[32] Liu T, Hu X, Hu W, Zou Y. A heuristic planning reinforcement learning-based energy management for power-split plug-in hybrid electric vehicles. IEEE Trans Ind Inf 2019;15(12):6436–45.

[33] Xu J, Li Z, Gao L, Ma J, Liu Q, Zhao Y. A comparative study of deep reinforcement learning-based transferable energy management strategies for hybrid electric vehicles. 2022. arXiv preprint arXiv:2202.11514.

[34] Prokhorov DV. Toyota prius hev neurocontrol and diagnostics. Neural Network 2008;21(2–3):458–65.

[35] Lian R, Peng J, Wu Y, Tan H, Zhang H. Rule-interposing deep reinforcement learning based energy management strategy for power-split hybrid electric vehicle. Energy 2020;197:117297.

[36] Zhu Z, Lin K, Zhou J. Transfer learning in deep reinforcement learning: a survey. 2020. arXiv preprint arXiv:2009.07888.

[37] Yang N, Han L, Xiang C, Liu H, Hou X. Energy management for a hybrid electric vehicle based on blended reinforcement learning with backward focusing and prioritized sweeping. IEEE Trans Veh Technol 2021;70(4):3136–48.