A Comparative Study of Deep Reinforcement Learning-based Transferable Energy Management Strategies for Hybrid Electric Vehicles

Jingyi Xu1, Zirui Li2, Li Gao1, Junyi Ma1, Qi Liu1 and Yanan Zhao1,*

Abstract—The deep reinforcement learning-based energy management strategies (EMS) has become a promising solution for hybrid electric vehicles (HEVs). When driving cycles are changed, the network will be retrained, which is a time-consuming and laborious task. A more efficient way of choosing EMS is to combine deep reinforcement learning (DRL) with transfer learning, which can transfer knowledge of one domain to the other new domain, making the network of the new domain reach convergence values quickly. Different exploration methods of RL, including adding action space noise and parameter space noise, are compared against each other in the transfer learning process in this work. Results indicate that the network added parameter space noise is more stable and faster convergent than the others. In conclusion, the best exploration method for transferable EMS is to add noise in the parameter space, while the combination of action space noise and parameter space noise generally performs poorly. Our code is available at https://github.com/BIT-XJY/RL-based-Transferable-EMS.git.

I. INTRODUCTION

Hybrid electric vehicles (HEVs) are currently important carriers of self-driving technology [1]. HEVs involve two or more energy sources, which realize efficient power distribution between the motor and engine. Thus there is a strong need for energy management strategies (EMS) to distribute power supplements among several power sources for energy efficiency improvement and emissions reduction [2]. There are mainly three types of EMS for HEVs: rule-based methods, optimization-based techniques, and learning-based approaches [3], [4].

The rule-based approach is the most common method to achieve real-time control of HEVs, the effectiveness of which depends on the intuition and experience of engineers [5]. To reduce the reliance on professional engineers, the optimization-based method is introduced, using optimization algorithms to solve for optimal or sub-optimal solutions in the feasible domain to obtain better fuel economy [6]. According to different optimal control objectives and algorithms, optimization-based EMS can be divided into global optimization and real-time optimization approaches. The classical optimization-based methods include linear programming algorithm (LP) [7], dynamic programming algorithm (DP) [8], [9], [10], equivalent fuel consumption minimization strategy (ECMS) [11], [12], model predictive control (MPC) [13], etc. The above methods improve the real-time and fuel economy of EMS to some extent, but they have more computational cost than the rule-based methods [14].

With the rapid development of machine learning in recent years, the learning-based energy management method has become a promising solution for HEVs, especially those exploiting reinforcement learning (RL). The EMS problem is modeled as a Markov Decision Process in DRL-based approaches. The optimal solution for EMS can be learned through the interaction between agents and the environment. Current studies mainly focus on deep reinforcement learning (DRL) based EMS due to their strong learning ability. [15] used deep Q-learning network (DQN) algorithm for energy management, which solved the dimensional catastrophe problem. Based on this, [16] compared double deep Q learning with DQN for energy management of plug-in hybrid vehicles and demonstrated advantages of the former in terms of convergence and fuel economy. [17] showed that the energy management policy based on deep deterministic policy gradient (DDPG) algorithm has a strong characterization capability of deep neural networks and can improve fuel economy significantly. In addition, [18] showed that asynchronous advantage actor-critic (A3C) and distributed proximal policy optimization (PPO) had improved the learning efficiency of EMS.

Although DRL-based methods have made a significant breakthrough, their limitations are the long time it takes for an agent to learn the optimal solution through trial-and-error interactions with the environment [19]. Besides, the training process must be repeated even when a new but similar task is encountered. Therefore, some works have combined transfer learning with DRL to improve the training efficiency between similar tasks. [20] combined proximal policy optimization (PPO) and transfer learning to effectively reduce time consumption and guarantee control performance. [21] combined DDPG and transfer learning to derive an adaptive energy management controller for hybrid tracked vehicles. Results show that this method has the potential to be applied in real-world environments. [22] incorporated transfer learning into DDPG-based EMS for HEVs to transfer knowledge between three HEVs that have apparently different structures.
In DRL, the agent utilizes exploration methods to acquire knowledge about the environment which may explore better actions. The main approach is to add different types of noise while selecting actions. Comparing the impact of different exploration methods on DRL is implemented by much previous work [23], [24]. However, there are few studies considering exploration methods on the combination of DRL and transfer learning, which improves the real-time performance of the algorithm and reduces the amount of computation.

This work is a comparative study, which focuses on effects of different exploration methods of DDPG for transferable EMS. DDPG combines advantages of DQN and the actor-critic architecture, which leads to stability and efficiency. Thus, DDPG is appropriate for evaluating the strategies from the network parameters transferring. In this work, several types of noise are added to DDPG networks which are trained by multiple driving cycles. Then, training weights are saved to initialize a new DDPG network. The second training process is performed with noise to acquire the optimal transferable EMS.

In sum, the main contributions of this work are: Different types of noise, i.e., action space noise and parameter space noise, are added to the DDPG algorithm to explore in actions selection. Parameters of networks with different exploration methods are used to initialize a new network. The method of exploration that works best for DDPG-based EMS and the most suitable for transfer learning in the real-time performance and final reward values is given by comparative study.

The remainder of this work is organized as follows: Section II introduces the proposed method in comparing effects of different exploration approaches of DDPG-based EMS and the performance of the transferred new network; Section III details experiment results, and the conclusion is depicted in Section IV.

II. DEEP REINFORCEMENT LEARNING-BASED TRANSFERABLE ENERGY MANAGEMENT STRATEGIES

A DRL-based transferable EMS is used to evaluate performances of different exploration methods. The sketch map of DRL-based transferable EMS is shown as Fig.1. This section describes the HEV model, the DRL-based EMS formulation, different types of noise added to DRL networks, and effects of transferred new domain network using different kinds of noise. In this process, different types of noise for exploration in DDPG are compared, which details on effects for deep transfer learning are introduced in Section III.

A. Hybrid Electric Vehicle Model

As one of the most classical HEVs, Prius is the second generation of Toyota hybrid system [25]. The EMS for Prius has been extensively studied.

1) Prius configuration: Prius is equipped with the Hybrid Synergy Drive system, which consists 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.2 are integrated with a power splitting planetary gear, which provides various power flow configurations for different modes of operation.

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 Ff, aerodynamic drag Fw, gradient resistance Fg, and inertial force Fa [26]:

\[
\begin{align*}
F &= F_f + F_w + F_g + F_a \\
F_f &= mg \cdot f \\
F_w &= \frac{1}{2} \rho \cdot A_f \cdot C_D \cdot v^2 \\
F_g &= mg \cdot i \\
F_a &= m \cdot a
\end{align*}
\]

where m is the curb weight, g is the gravitational constant, f is the rolling friction coefficient, \(\rho\) is the air density, \(A_f\) is the frontal area, \(C_D\) is the aerodynamic coefficient, v is the speed in regard to a certain driving cycle, i is road slope (not considered in this paper), and a is the acceleration.

3) Powertrain system model: The ICE, 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 the temperature change and battery aging:

\[
\begin{align*}
P(t) &= I(t) \cdot V_{oc}(t) - R_0 \cdot I(t) \\
I(t) &= V_{oc}(t) - \sqrt{V_{oc}^2(t) - 4 \cdot R_0 \cdot P(t)} \\
SoC(t) &= \frac{Q_0 - \int_{t_0}^{t} I(t) \, dt}{Q}
\end{align*}
\]
where $P$ is the output power, $I$ denotes the current, $V_{\text{soc}}$ is the open-circuit voltage, $Q_0$ 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 I.

### Table I
**Parameters of Prius**

| Components | Parameters | Values |
|------------|------------|--------|
| Engine     | Maximum power, $P_e$ | 56 kW |
|            | Maximum torque, $T_e$ | 120 Nm |
| Motor      | Maximum power, $P_m$ | 50 kW |
|            | Maximum torque, $T_m$ | 400 Nm |
| Battery    | Capacity, $Q$ | 1.54 kWh |
|            | Voltage, $V_{soc}$ | 237 V |
|            | Roll resistance coefficient, $f$ | 0.013 |
| Vehicle    | Air resistance coefficient, $f_a$ | 0.26 |
|            | Frontal area, $A_f$ | 2.23 m² |
|            | Wheel radius, $r$ | 0.287 m |
| Transmission | Final gear ratio, $i_p$ | 3.93 |
|            | Characteristic parameter, $C$ | 2.6 |

### B. DRL Formulation

A DRL problem that satisfies the Markov property can generally be modeled in terms of a Markov Decision Process (MDP), which can be characterized as $(S, A, P, R, \gamma)$. $S$ represents a set of state spaces. $A$ is a set of action spaces. $P$ denotes a state transition probability matrix. $R$ represents a reward function. $\gamma$ denotes a discount factor.

DDPG is one of the most typical actor-critic DRL methods, which is an off-policy and model-free algorithm. As illustrated in Fig. 3, DDPG has an actor network $\mu(s|\theta^\mu)$, a critic network $Q(s,a|\theta^Q)$, an actor target network $\mu'(s'|\theta'^\mu)$, and a critic target network $Q'(s',a'|\theta'^Q)$. 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 actor network and critic network input. $a$ is the agent action as actor network output and critic network input. $\theta$ represents parameters of the corresponding network. $s'$ and $a'$ are defined in the same way.

The DDPG algorithm is used to learn the optimal policy of Prius EMS in this work. The neural network has a pyramid-like architecture, with the number of neurons in hidden layers decreasing layer by layer. The optimal EMS utilizes the DDPG algorithm, which is trained with different driving cycles.

The DDPG-based EMS is formulated according to the following Markov Decision Process.

1) **State space, $S$**: The state of the system, 
   $$s = \{SoC, v, acc\}$$  
   which consists of $SoC$, the velocity of Prius $v$ and the acceleration $acc$.

2) **Action space, $A$**: At every episode, the agent can select actions in continuous engine power $T_{\text{eng}}$: 
   $$a = \{T_{\text{eng}}\}$$

3) **Reward function, $R$**: Multi-objectives of the agent are to minimize the energy consumption and retain the battery SoC at an appropriate range:
   $$r = -\{\alpha[\text{fuel}(t) + \text{elec}(t)] + \beta[\text{SoC}_{\text{ref}} - \text{SoC}(t)]^\mu\}$$  
   where $\alpha$ is the weight of fuel consumption $\text{fuel}$ and electricity consumption $\text{elec}$, $\beta$ is the weight of battery charge-sustaining, and $\text{SoC}_{\text{ref}}$ represents the SoC reference value.

### C. Transfer Learning

Traditional DRL algorithms are used to solve the problem with training and test data in the same domain. Otherwise, once the domain is changed, the network need to be re-trained, which is quite expensive and inconvenient. Transfer learning is extremely useful in solving this problem. When two domains are similar, network parameters can be stored and reused in the new one along with transfer learning approaches.

Given a source domain $M_s$ and a target domain $M_t$, transfer learning aims to learn an optimal policy $\pi^*$ from the source domain $M_s$ for the target domain $M_t$. The source domain $M_s$ provides prior knowledge $D_s$ that is accessible for the target domain $M_t$. Thus, by leveraging the information from $D_s$, the target agent learns better and faster in the target domain $M_t$ [27].

A network that specializes in obtaining source EMS is used in our work. Since driving cycles of the source domain and the target domain have the same feature space and are correlated with each other, source domain knowledge can be transferred to the novel, but related 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 DDPG architecture shown in Fig. 3, and weights of the source network except for the last layer are used to initialize the target network that will be trained on new driving cycles. Further details about the hyperparameters of DDPG in the source domain and target domain are given in Table II.

In DDPG, the agent utilizes exploration to acquire knowledge about the environment and applies exploitation to achieve a control action based on current knowledge [29]. Thus, the coordination between exploitation and exploration
is essential. This subsection provides a description of exploration methods used in the DDPG-based transferable EMS.

The primary purpose of exploration is to avoid local optimum for agent’s behaviors [23]. Thus, to realize efficient and effective exploration, random noise is frequently added to perturb selected actions, which is the engine power in our work. Main approaches can be classified into two groups: adding noise in the action space and adding noise directly to agent’s parameters.

1) **Action space noise**: When the agent selects actions through the actor network, the noise \( N \) is added to the action space. The final selected action \( a_t \) at every step satisfies:

\[
a_t = \mu(s|\theta^\mu) + N
\]

Action space noise could be a simple Gaussian noise or a more advanced Ornstein-Uhlenbeck (OU) correlated noise process [24]. Gaussian noise satisfies \( N \sim N(0, \sigma^2 I) \), where \( \sigma^2 \) denotes variance and the expected value is set as 0. An OU process [30] can be used as a temporally correlated noise. Just like the Gaussian noise mentioned above, the expected value of OU noise \( N \sim OU(0, \sigma^2) \) is set as 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, which can lead to inconsistent exploration. The parameter space noise solves this problem and directly perturbs actor network parameters to get a rich set of behaviors. The final selected action \( a_t \) at every step satisfies:

\[
\begin{align*}
\bar{a}_t &= \mu(s|\tilde{\theta}^\mu) \\
\tilde{\theta} &= \theta + N(0, \sigma^2 I)
\end{align*}
\]

### III. Experiments

The purpose of our work is to compare effects of different exploration methods of DDPG-based EMS and the transferred new network in terms of transfer efficiency. Driving cycles are selected for the source domain and the target domain, which are different but similar. Then, networks with different exploration methods are trained in the source domain, of which parameters are saved. Finally, the adaptation of target domain networks, which parameters are initialized using saved parameters, is evaluated.

**TABLE II**

| Parameters                      | Source Domain | Target Domain |
|---------------------------------|---------------|---------------|
| Number of Episodes, \( K \)     | 1000          | 300           |
| Replay memory size, \( M \)     | 50000         | 50000         |
| Learning rate of actor network, \( lr_a \) | 0.001        | 0.0009        |
| Learning rate of critic network, \( lr_c \) | 0.01        | 0.009         |
| Discount factor, \( \gamma \)  | 0.9           | 0.9           |
| Target network update frequency, \( \tau \) | 0.01        | 0.01          |
| Mini-batch size, batch          | 64            | 64            |

**TABLE III**

| Space Added Noise | Noise Type | Variance |
|-------------------|------------|----------|
| Action space      | Gaussian   | 0.02, 0.03, 0.04, 0.05, 0.06 |
| Action space & parameter space | OU & Gaussian | 0.08, 0.09, 0.10, 0.11, 0.13 |
| Parameter space   | Gaussian   | 0.05, 0.09 |
| Action & parameter space | Gaussian & OU | 0.06 & 0.03 |

**A. Driving Cycles**

In this work, driving cycles are all selected from standard data. Source tasks are performed over multiple cycles, including Urban Dynamometer Driving Schedule (UDDS) [31], FTP75 [26], etc. Target tasks are conducted on New European Driving Cycle (NEDC) [32], which differs from the driving cycles used in the source domain but similar. Using multiple 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 high similarity to the source driving cycles is chosen for the target domain, as similarity is a necessary factor for transfer learning.

**B. Training in the Source Domain**

To ensure the validity of weights to be transferred, networks with different exploration methods are firstly trained on the source domain. Settings of networks added different types of noise are shown in Table III. By comparative studying, suitable networks, of which parameters are used to initiate weights of target networks, are selected according to training results.

As described in Fig.4, Gaussian noise added in the action space, OU noise added in the action space, Gaussian noise added in the parameter space, and their mixture are used to explore in DDPG-based EMS. In Fig.4(a), different variance values \( \sigma^2 \) of Gaussian noise added in the action space are set as 0.02, 0.03, 0.04, 0.05, 0.06, respectively. The reward fluctuates the most when the variance is set to 0.02. Only the network with variance set to 0.05 shows considerable oscillations once the network almost reaches the converge. Thus, the network with a variance of 0.06 is the most stable.

Similarly, Fig.4(b) and 4(c) illustrates that the network’s variance of 0.09 with OU noise added in the action space and the network’s variance of 0.03 with Gaussian noise added in the parameter space are the most stable, respectively. Results of other noise configurations, which converge to a local optimum or fluctuate too much, are not shown here. Besides, different effects of a single noise and a mixed noise are compared in Fig.4(d). Results using the Gaussian noise added in the action space and parameter space are very unstable, as shown by the yellow line. The noise of multiple Gaussian processes allows the agent to explore in selecting actions too much, leading to a wide range of reward values for each episode. The most stable is the Gaussian noise added in the action space, followed by the Gaussian noise added in the both action space and parameter space.
Above all, the most stable network is the one that utilizes the Gaussian noise added in the action space with 0.06 variance to explore, followed by the network with the combination of OU noise added in the action space and Gaussian noise added in the parameter space to explore. Chosen networks which are transferred to a new domain are as shown in Fig. 4(e).

C. Adaptation of Transfer Learning

Results of the transferred DDPG in the target domain using different exploration methods in EMS are discussed in this subsection. A new network is trained on the new driving cycle, learning from scratch or initializing the network parameters using prior ones.

As shown in Table IV, following criteria are adopted to evaluate the adaptation of the transferable EMS [33]:

1) Jumpstart Performance (JP): The initial performance (reward) of the agent. The mean reward of first 50 episodes is used to evaluate it.

2) Asymptotic Performance (AP): The ultimate performance (reward) of the agent. The mean reward of the convergence interval is used to evaluate it.

3) Time to Threshold (TT): The learning time (iterations) needed for the target agent to reach a certain performance threshold. The iteration number of convergence is used to evaluate it.

In Table IV, TFS means the source network trained from scratch, Gaussian AS, OU AS, Gaussian PS and APS mean source networks with Gaussian noise added in the action space, OU noise added in the action space, Gaussian noise added in the parameter space, and both OU noise added in the action space and Gaussian noise added in the parameter space, respectively. While target networks use the same type of noise, the maximum reward value obtained from first 50 episodes is larger than the maximum convergence value in the convergence interval, which means that the DDPG network falls into a local optimum during the exploration process. Besides, the initialization of parameters of the target network with parameter space noise generally works well in terms of convergence speed and mean reward value, except for target networks that use the mixture of action space and parameter space noise for exploration. After the target network has converged, convergence values of different exploration methods do not differ significantly.

As shown in Fig. 5(a), no noise is added on the target network. In first 50 episodes, the network which is trained from scratch (gray line) starts out with a small reward value, which means that its JP is poor. The green, blue, yellow, and orange lines represent parameters of networks with different exploration methods to initialize the current target network. Weights of these networks are perturbed by Gaussian noise added in the action space ($\sigma^2 = 0.06$), OU noise added in the action space ($\sigma^2 = 0.09$), Gaussian noise added in the parameter space ($\sigma^2 = 0.03$), both OU noise added in the action space noise and Gaussian noise added in the parameter space to explore, respectively. The yellow line fluctuates the least. In the convergence interval (To unify the variables, the convergence interval is considered between 50 and 300 episodes in all target cases), the green line is the most stable,
| Exploration Method (Target Network) | Transferred Network Parameter (Source Network) | Mean Return (First 50 Episodes) | Mean Return (Convergence Interval) | Iteration Number |
|-----------------------------------|-----------------------------------------------|--------------------------------|-----------------------------------|-----------------|
| No noise                          | TFS                                           | -21.6386                       | -1.2101                           | 30              |
|                                   | Gaussian_AS                                   | -0.9186                        | -1.0540                           | 20              |
|                                   | OU_AS                                         | -0.9451                        | -1.2019                           | 29              |
|                                   | Gaussian_PS                                   | -0.9009                        | -1.4133                           | 21              |
|                                   | APS                                           | -2.0715                        | -1.1878                           | 29              |
|                                  | TFS                                           | -20.9453                       | -0.8766                           | 36              |
| Gaussian noise added in the action space | Gaussian_AS                                   | -0.8780                        | -0.8776                           | 35              |
|                                  | OU_AS                                         | -0.9748                        | -0.8908                           | 30              |
|                                  | Gaussian_PS                                   | -0.7987                        | -0.9071                           | 25              |
|                                  | APS                                           | -1.7628                        | -0.9152                           | 32              |
| OU noise added in the action space | TFS                                           | -23.3330                       | -1.1327                           | 36              |
|                                  | Gaussian_AS                                   | -1.1239                        | -1.0585                           | 24              |
|                                  | OU_AS                                         | -0.9035                        | -1.0417                           | 22              |
|                                  | Gaussian_PS                                   | -0.8720                        | -1.1611                           | 31              |
|                                  | APS                                           | -2.3262                        | -1.0474                           | 23              |
| Gaussian noise added in the parameter space | TFS                                           | -25.009                        | -1.1523                           | 35              |
|                                  | Gaussian_AS                                   | -1.0250                        | -1.1132                           | 21              |
|                                  | OU_AS                                         | -1.1109                        | -1.1652                           | 22              |
|                                  | Gaussian_PS                                   | -1.5669                        | -1.2466                           | 34              |
|                                  | APS                                           | -4.4550                        | -1.2734                           | 30              |
| Noise added in action space and parameter space | TFS                                           | -21.1288                       | -1.0919                           | 35              |
|                                  | Gaussian_AS                                   | -1.4819                        | -1.0066                           | 31              |
|                                  | OU_AS                                         | -0.9408                        | -1.1473                           | 27              |
|                                  | Gaussian_PS                                   | -13.8613                       | -0.9998                           | 24              |
|                                  | APS                                           | -2.6120                        | -1.1589                           | 30              |

Fig. 5. Comparison of different exploration methods in the target domain.
gray and blue lines are the second most unstable, and the most unstable is the yellow line. In general, they do not differ much. This means that it is the most stable to initialize the target network with parameters of the original network with Gaussian action space noise.

The Gaussian noise added in the action space is added on the target network, of which the training process is shown in Fig. 5(b). In first 50 episodes, the gray line fluctuates the most, and the orange line fluctuates the least, while in the convergence interval, the blue line fluctuates the least. This means that it is the most stable to initialize the target network, which uses Gaussian action space noise to explore, with parameters of the original network with OU action space noise.

The action space noise with OU process is added on the target network, of which the training process is shown in Fig. 5(c). The gray line has the most considerable fluctuation, followed by the orange line in first 50 episodes. The blue line has the slightest fluctuation. In the convergence interval, the most stable is still the blue line, which means that it has a better learning effect to initialize the target network, which uses OU action space noise to explore, with parameters of the original network with OU action space noise.

The Gaussian noise added in the parameter space is added on the target network, of which the training process is shown in Fig. 5(d). In first 50 episodes, the gray line has a small initial value because it does not have any prior knowledge. The orange line fluctuates the second most, which means parameters of the network with action space noise and parameter space noise are not suitable for initializing the new target network with parameter space noise. In the convergence interval, the most stable is still the blue line, followed by the green line, and the most unstable one is the yellow line. The blue line is the most stable throughout the training process.

The action space noise and the parameter space noise are added on the target network, of which the training process is shown in Fig. 5(e). The gray line has the smallest initial value, while the yellow line has the largest fluctuation in first 50 episodes. It means that parameters of the network with parameter space noise are not suitable for initializing the target network with action space noise and parameter space noise. In the convergence interval, the fluctuation in each of these methods is negligible.

Combining results expressed in Fig. 4 and Fig. 5, the network with the parameter space noise is the most stable, while the network with the multiple noises of action space and parameter space has poor initial performance.

IV. CONCLUSION

In this paper, choosing an optimal and real-time EMS is formulated as a deep reinforcement learning-based transfer learning problem. We compared different exploration approaches for deep reinforcement learning and transfer learning to find out the best energy management strategy. Effects of action space noise and parameter space noise which are added to the DDPG algorithm, are presented. Experimental results show that the method of parameter space noise exploration works best for DDPG-based EMS and is the most suitable approach for transfer learning.

Despite these encouraging comparison results, there are several avenues for future research. First, we want to investigate other exploration modalities, not only adding noise for action selecting. We furthermore plan to compare different exploration methods based on more robust DRL algorithms, such as twin delayed DDPG (TD3) and soft actor-critic (SAC).

REFERENCES

[1] T. Liu, W. Tan, X. Tang, J. Zhang, Y. Xing, and D. Cao, “Driving conditions-driven energy management strategies for hybrid electric vehicles: A review,” Renewable and Sustainable Energy Reviews, vol. 151, p. 115521, 2021.
[2] T. Liu and X. Hu, “A bi-level control for energy efficiency improvement of a hybrid tracked vehicle,” IEEE Transactions on Industrial Informatics, vol. 14, no. 4, pp. 1616–1625, 2018.
[3] X. Guo, T. Liu, B. Tang, X. Tang, J. Zhang, W. Tan, and S. Jin, “Transfer deep reinforcement learning-enabled energy management strategy for hybrid tracked vehicle,” IEEE Access, vol. 8, pp. 165 837–165 848, 2020.
[4] J. Peng, H. He, and R. Xiong, “Rule based energy management strategy for a series-parallel plug-in hybrid electric bus optimized by dynamic programming,” Applied Energy, vol. 185, pp. 1633–1643, 2017.
[5] H. Guo, X. Wang, and L. Li, “State-of-charge-constraint-based energy management strategy of plug-in hybrid electric vehicle with bus route,” Energy Conversion and Management, vol. 199, p. 111972, 2019.
[6] T. Hofman, M. Steinbuch, R. Van Druten, and A. Serrarens, “Rule-based energy management strategies for hybrid vehicles,” International Journal of Electric and Hybrid Vehicles, vol. 1, no. 1, pp. 71–94, 2007.
[7] F. Z. Kadda, S. Zouggar, and M. L. Elhafyani, “Optimal energy management of an autonomous hybrid system by using the linear programming method,” in 2013 International Renewable and Sustainable Energy Conference (IRESCE). IEEE, 2013, pp. 420–425.
[8] L. Li, C. Yang, Y. Zhang, L. Zhang, and J. Song, “Correctional dp-based energy management strategy of plug-in hybrid electric bus for city-bus route,” IEEE Transactions on Vehicular Technology, vol. 64, no. 7, pp. 2792–2803, 2014.
[9] C.-C. Lin, H. Peng, J. W. Grizzle, and J.-M. Kang, “Power management strategy for a parallel hybrid electric truck,” IEEE transactions on control systems technology, vol. 11, no. 6, pp. 839–849, 2003.
[10] X. Wang, H. He, F. Sun, and J. Zhang, “Application study on the dynamic programming algorithm for energy management of plug-in hybrid electric vehicles,” Energies, vol. 8, no. 4, pp. 3225–3244, 2015.
[11] A. Rezaei, J. B. Burl, and B. Zhou, “Estimation of the ecms equivalent factor bounds for hybrid electric vehicles,” IEEE Transactions on Control Systems Technology, vol. 26, no. 6, pp. 2198–2205, 2017.
[12] Q. Jiang, F. Ossart, and C. Marchand, “Comparative study of real-time hev energy management strategies,” IEEE Transactions on Vehicular Technology, vol. 66, no. 12, pp. 10875–10888, 2017.
[13] H. Borhan, A. Vahidi, A. M. Phillips, M. L. Kuang, I. V. Kolmanovsky, and S. Di Cairano, “Mpc-based energy management of a power-split hybrid electric vehicle,” IEEE Transactions on Control Systems Technology, vol. 20, no. 3, pp. 593–603, 2011.
[14] R. Johri and Z. Filipi, “Optimal energy management of a series hybrid vehicle with combined fuel economy and low-emission objectives,” Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering, vol. 228, no. 12, pp. 1424–1439, 2014.
[15] J. Wu, H. He, J. Peng, Y. Li, and Z. Li, “Continuous reinforcement learning of energy management with deep q network for a power split hybrid electric bus,” Applied Energy, vol. 222, pp. 799–811, 2018.
[16] X. Qi, Y. Luo, G. Wu, K. Boriboonsomsin, and M. Barth, “Deep reinforcement learning enabled self-learning control for energy efficient driving,” Transportation Research Part C: Emerging Technologies, vol. 99, pp. 67–81, 2019.
[17] Y. Wu, H. Tan, J. Peng, H. Zhang, and H. He, “Deep reinforcement learning of energy management with continuous control strategy and traffic information for a series-parallel plug-in hybrid electric bus,” Applied energy, vol. 247, pp. 454–466, 2019.

[18] X. Tang, J. Chen, T. Liu, Y. Qin, and D. Cao, “Distributed deep reinforcement learning-based energy and emission management strategy for hybrid electric vehicles,” IEEE Transactions on Vehicular Technology, vol. 70, no. 10, pp. 9922–9934, 2021.

[19] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26–38, 2017.

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

[21] X. Guo, T. Liu, B. Tang, X. Tang, J. Zhang, W. Tan, and S. Jin, “Transfer deep reinforcement learning-enabled energy management strategy for hybrid tracked vehicle,” IEEE Access, vol. 8, pp. 165837–165848, 2020.

[22] R. Lian, H. Tan, J. Peng, Q. Li, and Y. Wu, “Cross-type transfer for deep reinforcement learning based hybrid electric vehicle energy management,” IEEE Transactions on Vehicular Technology, vol. 69, no. 8, pp. 8367–8380, 2020.

[23] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, “Parameter space noise for exploration,” arXiv preprint arXiv:1706.01905, 2017.

[24] C. Colas, O. Sigaud, and P.-Y. Oudeyer, “Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms,” in International conference on machine learning, PMLR, 2018, pp. 1039–1048.

[25] D. V. Prokhorov, “Toyota prius hev neurocontrol and diagnostics,” Neural Networks, vol. 21, no. 2-3, pp. 458–465, 2008.

[26] R. Lian, J. Peng, Y. Wu, H. Tan, and H. Zhang, “Rule-interposing deep reinforcement learning based energy management strategy for power-split hybrid electric vehicle,” Energy, vol. 197, p. 117297, 2020.

[27] Z. Zhu, K. Lin, and J. Zhou, “Transfer learning in deep reinforcement learning: A survey,” arXiv preprint arXiv:2009.07888, 2020.