Neural Computing and Applications (2020) 32:14431–14445
https://doi.org/10.1007/s00521-019-04556-4

EXTREME LEARNING MACHINE AND DEEP LEARNING NETWORKS

Energy management strategy for electric vehicles based on deep *Q*-learning using Bayesian optimization

Huifang Kong[1] · Jiapeng Yan[1] · Hai Wang[2] · Lei Fan[1]

Received: 3 January 2019 / Accepted: 5 October 2019 / Published online: 24 October 2019

Abstract
In this paper, a deep *Q*-learning (DQL)-based energy management strategy (EMS) is designed for an electric vehicle. Firstly, the energy management problem is reformulated to satisfy the condition of employing DQL by considering the dynamics of the system. Then, to achieve the minimum of electricity consumption and the maximum of the battery lifetime, the DQL-based EMS is designed to properly split the power demand into two parts: one is supplied by the battery and the other by supercapacitor. In addition, a hyperparameter tuning method, Bayesian optimization (BO), is introduced to optimize the hyperparameter configuration for the DQL-based EMS. Simulations are conducted to validate the improvements brought by BO and the convergence of DQL algorithm equipped with tuned hyperparameters. Simulations are also carried out on both training dataset and the testing dataset to validate the optimality and the adaptability of the DQL-based EMS, where the developed EMS outperforms a previously published rule-based EMS in almost all the cases.

Keywords Energy management strategy (EMS) · Electric vehicle (EV) · Deep *Q*-learning (DQL) · Bayesian optimization (BO)

1 Introduction

Development and deployment of electric vehicles (EVs) have gained tremendous momentum nowadays mainly due to concerns over petroleum shortages and environmental pollution [1]. Nevertheless, the batteries in the EVs suffer severe battery degradation under high-rate charge or discharge operation mode caused by frequent and peak power demands [2–4]. On the other hand, another available energy storage, supercapacitor, is characterized by high power density and exceptionally long cycle lifespan [5], so it is much more robust in handling peak power and current requirements. Therefore, hybrid energy storage systems (HESS), where battery serves as a persistent energy source and supercapacitor is employed to share power load, are widely adopted [6–9]. The introduction of an additional energy source increases the complexity of power flow; therefore, an energy management strategy (EMS) should be carried out to coordinate the power distribution between the two energy sources.

Previously published EMSs can be generally classified into three categories: the rule-based EMSs (such as the deterministic rule-based control strategies [10] and fuzzy logic control strategies [11]), the optimization-based EMSs (such as the dynamic programming algorithm [12] and the Pontryagin's minimum principle [13]) and the learning-based EMSs (such as strategies based on reinforcement learning [14–16]). Rule-based EMSs are easy to implement for real-time applications. Nevertheless, all these strategies cannot obtain the global optimum. In contrast, theoretically optimal control results can be found by optimization-based EMSs. However, these EMSs are often based on awareness of future information in advance, so they cannot be implemented in real time. Although real-time optimization-based EMSs such as model predictive control (MPC) is employed in [17], an accurate estimation on battery degradation is still absent.

EMSs based on reinforcement learning (RL) can be treated as a trade-off between real-time application and

✉ Jiapeng Yan
yanjiapeng@mail.hfut.edu.cn

1 School of Electrical Engineering and Automation, Hefei University of Technology, Hefei 230009, China

2 College of Science, Health, Engineering and Education, Murdoch University, Perth, WA 6150, Australia

solution optimality. Q-learning, as a well-known and effective RL algorithm, has been applied in EMS for HESS recently. For example, an EMS using Q-learning is presented in the literature [14] and shows less energy consumption than a rule-based strategy. In Q-learning, a discretized lookup table is used to represent all the Q-values, which represents the favorable level of control action taken by agent and leads the learning progress into the desired direction. However, it is necessary to adopt continuous state space to accurately represent the complex dynamics of HESS and EV, in which case the growing computational load and poor convergence (i.e., the "curse of dimensionality") can be caused by the discretized table. Therefore, a deep neural network (DNN) is applied to represent the Q-function and deep Q-learning (DQL) algorithm is proposed in [18].

Directly influencing the performance of deep RL, hyperparameters require careful tuning. In existing works on deep reinforcement learning (DRL)-based EMS [15, 16], hyperparameters tuning is often manually done mainly because, as of yet, there is no analytical way to determine the optimal configuration for deep networks [19] and parameters for DRL algorithm. Manually choosing the hyperparameters can be time-consuming, so several strategies are proposed like grid search, random search [20] and gradient-based methods, which are characterized by large execution time and not suitable for most RL tasks [21]. Bayesian optimization, on the other hand, is an efficient strategy in optimizing functions expensive to query, so it is widely applied in hyperparameter tuning tasks [22–24].

In this paper, a DQL-based EMS of a HEES for an EV is developed. Firstly, the DQL algorithm and Bayesian optimization are all introduced and the considered system is modeled. Then the energy management problem is described as a Markov decision process to employ deep reinforcement learning approaches, followed by the details and some theoretical analyses of the method. Furthermore, the hyperparameters of DQL are tuned using Bayesian optimization. Finally, to evaluate the optimality and adaptability of the developed EMS, it is compared with a published rule-based strategy on both the training dataset and the testing dataset.

Our contributions in this paper are summarized as follows:

- A deep Q-learning-based EMS of a HESS for EV is developed;
- Bayesian optimization is introduced in the deep Q-learning process to automatically obtain the best hyperparameter configuration;
- Simulations are conducted to validate the effectiveness of the developed EMS.

The remainder of this paper is organized as follows. In Sect. 2 we introduce the background of this work. The detailed DQL-based EMS is presented in Sect. 3. The simulation results are reported in Sect. 4, followed by conclusion given in Sect. 5.

2 Background

In this section, the background of this work is introduced, including overviews of the deep Q-learning Algorithm in Sect. 2.1, the introduction of Bayesian optimization in Sect. 2.2, and a description of the system modeling in Sect. 2.3.

2.1 Reinforcement learning and deep Q-learning

Reinforcement learning (RL) is a model-free strategy, and it is based on the Markov decision process (MDP). RL is described by a tuple $\langle S,\ A,\ P,\ R\rangle$, where S and A are state space and action space, respectively, $P:\ S \times A \times S \to [0, 1]$ denotes the transition probability among all states and $R:\ S \times A \to \mathbb{R}$ is the reward. The basic objective of RL is to find an optimal policy $\pi:\ S \to A$ which maximizes the long-term accumulative reward $R = \sum_{t=0}^{\infty} \gamma^t r(t)$, where $\gamma \in (0,\ 1)$ is the discount factor.

This objective is realized through the interaction between agent and environment, that is, trials of agent and response from the environment.

The interaction of reinforcement learning is illustrated in Fig. 1. At decision time step t_k, the agent is presented with a state s_k, and after an action a_k is selected and executed, the next state s_{k+1} and a reward r_k are obtained. The optimal Q-value, $Q^*(s_t, a_t)$, represents the maximum accumulative reward, and it is calculated using the Bellman equation:

$$Q^*(s_t,\ a_t) = E\left[r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1},\ a_{t+1})|s_t,\ a_t\right] \qquad (1)$$

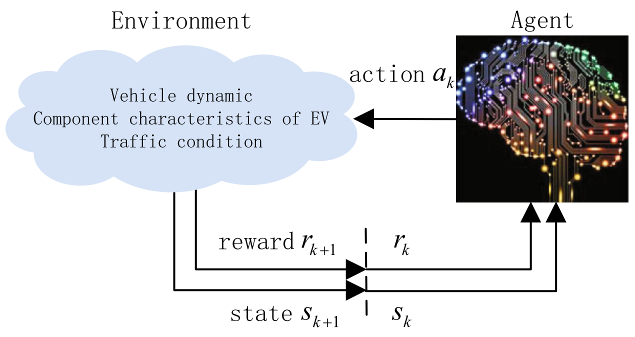


Fig. 1 Interaction between a reinforcement learning agent and the environment [14]

Q-learning is an effective RL algorithm to update the value estimation, and its one-step update rule is formulated as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \eta\left[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \tag{2}$$

where η denotes the learning rate. Such an algorithm converges to the optimal Q-value, i.e., $Q_t \to Q^*$ as $t \to \infty$.

However, the above method is impractical due to the lack of generalization [25]. In deep Q-learning algorithm, a deep neural network called deep Q-network (DQN) with parameters θ is introduced to estimate Q-value, $Q(s_t, a_t; \theta) \approx Q^*(s_t, a_t)$. The training process of DQN aims to minimize the loss function $L(\theta)$, which is defined as follows:

$$L(\theta) = E\left[\left(r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^-) - Q(s_t, a_t; \theta)\right)^2\right] \tag{3}$$

where the target Q-value is represented by $r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta^-)$, calculated by previous network parameters θ^-, $Q(s_t, a_t, \theta)$ denotes the Q-value estimated by DQN parameterized by θ. Parameters θ are updated using a gradient descent performed on $L(\theta)$:

$$\nabla_\theta L(\theta) = E\left[\left(r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^-) - Q(s_t, a_t; \theta)\right)\nabla_\theta Q(s_t, a_t; \theta)\right] \tag{4}$$

2.2 Bayesian optimization and tree-structured Parzen estimator

Consider $f : \mathbf{X} \to \mathbb{R}$ as an implicit function to be optimized, where $\mathbf{X} \subset \mathbb{R}^d$ is a compact and convex set [24], d represents the dimension of the input space. The objective of Bayesian optimization is to find an optimal solution of f through queries $\mathbf{x}_1, \mathbf{x}_2, \ldots \in \mathbf{X}$. To achieve this, one surrogate model M of f is built and the optimal solution is selected according to M.

In this work, the query $\mathbf{x}_i$ represents the hyperparameter configuration at ith iteration and $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^{\mathrm{T}}$ where x is one of the hyperparameters. $y_i = f(\mathbf{x}_i)$ is selected as the final episodic training reward of DQL equipped with hyperparameters $\mathbf{x}_i$. Tree-structured Parzen estimator approach (TPE) [26] algorithm is employed as hyperparameter tuning method. In TPE, expected improvement (EI) [27] is selected as the optimization objective, which represents the expectation under model M that y will exceed a threshold y^*:

$$\begin{aligned} EI_{y^*}(x) &= \int_{-\infty}^{\infty} \max(y^* - y, 0) p(y|x) \mathrm{d}y \\ &= \int_{-\infty}^{y^*} (y^* - y) p(y|x) \mathrm{d}y \\ &= \int_{-\infty}^{y^*} (y^* - y) \frac{p(x|y)p(y)}{p(x)} \mathrm{d}y \end{aligned} \tag{5}$$

where $p(x)$ and $p(y)$ denote the probability of x and y having specific values, respectively, $p(x|y)$ is the surrogate model representing the relationship between y and every hyperparameter x. $p(x|y)$ is built as an alternative density estimate which is conditional on the value of y relative to a threshold y^*:

$$p(x|y) = \begin{cases} l(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases} \tag{6}$$

where $l(x)$ is the density built from all the x that its corresponding y is less than y^*, $g(x)$ is the density formed by the remaining x and y^* is chosen as the γ-quantile of all the corresponding y obtained so far. Intuitively speaking, this creates two probabilistic density estimators $l(x)$ and $g(x)$ for hyperparameters doing "good" and "poor," respectively [28].

On the other hand, $EI_{y^*}(x)$ is proved to satisfy the following expression [26]:

$$EI_{y^*}(x) \propto \left(\gamma + \frac{g(x)}{l(x)}(1 - \gamma)\right)^{-1}. \tag{7}$$

Therefore, $EI_{y^*}(x)$ can be maximized by generating many candidate queries x randomly and picking one query minimizing the ratio $\frac{g(x)}{l(x)}$.

For every hyperparameter x, $l(x)$ and $g(x)$ are estimated using Parzen estimator (also known as kernel density estimator), which provides the probability density function (PDF) of a random variable in a nonparametric way. Let $(z_1, z_2, \ldots, z_n)$ represents the samples of one hyperparameter drawn from an unknown distribution; the Parzen estimator uses samples to estimate the function f by:

$$\hat{f_h}(z) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{z - z_i}{h}\right) \quad i = 1, \ldots, n \tag{8}$$

where $K(\cdot)$ is the kernel function and h is the bandwidth.

In addition, in TPE, the hyperparameter configuration space is defined in a form of a tree structure, where some leaf node variables (e.g., number of neurons of second layer of DQN) are valid only if roof node variables (e.g., number of layers) take particular values. In this case, we call leaf node variables are conditional on roof node variables, and other leaf nodes can be conditional on these leaf nodes, causing a tree-structured space [26]. For each node in the tree structure, a 1-D Parzen estimator is built to estimate PDF. For a given hyperparameter configuration $\mathbf{x}$

that is added to either l or g, only the 1-D estimators of valid hyperparameters in $\mathbf{x}$ are updated.

For continuous hyperparameters, firstly, the hyperparameter is specified by a prior distribution. Secondly, these estimators are constructed by assuming the density as Gaussian (i.e., Gaussian kernel $K\left(\frac{z-z_i}{h}\right) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{z-z_i}{h}\right)^2\right)$ [29], with the bandwidth h set to the greater of the distances to the left and right neighbors of each data point [26]. Finally, the distribution substitutes an equally weighted mixture of these Gaussians and the prior distribution.

For discrete hyperparameters, the prior is a vector of N probabilities p_j where N is the number of choices of the hyperparameter's value and the posterior vector elements are proportional to $p_j + C_j$, where C_j counts the occurrences of choice j. The leaf nodes are mutually independent; so, the dependence of y on hyperparameter configuration $\mathbf{x}$ as well as the joint density function $f(\mathbf{x})$ can be computed by multiplying the individual density estimations of every hyperparameter in $\mathbf{x}$.

2.3 System modeling

The structure of considered EV and its powertrain configuration is sketched in Fig. 2. As shown, the vehicle mainly consists of a single-speed transmission, an electrical bus, a HESS and an electric drive system containing a bidirectional DC–AC inverter and a motor. In the HESS, the supercapacitor is connected to the electrical bus via a bidirectional DC–DC converter while the battery is directly connected. Both the energy sources can power the vehicle or receive regenerative energy either separately or together. In this study, we similarly adopt the configuration of EV discussed in [30].

The powertrain of the HESS can be in one of the six power flow modes listed in Table 1, where P_{dem} is the total power demand drawn from the wheel, A and P_{sc} are output power of the battery and supercapacitor, respectively. A positive value of power represents that the vehicle is being propelled or the corresponding energy source is providing power, while a negative one means the vehicle is in regenerative braking mode or the energy source is being charged. In modes 1 and 6, both the energy storages are powering the vehicle or being charged simultaneously. In the other four modes, one storage is providing power to or being charged by the other.

2.3.1 Vehicle dynamics

In this paper, the longitudinal dynamics of the vehicle are formulated as follows [15]:

$$P_{dem} = (F_a + F_r + F_w)v$$
$$F_a = ma,\ F_r = mgf,\ F_w = \frac{C_d A}{21.15}v^2 \tag{9}$$

where P_{dem} is the power demanded to drive the vehicle, F_a is the inertial force, F_r is the rolling resistance, F_w is the aerodynamic drag, v is the vehicle speed, C_d is the aerodynamic coefficient, A is the fronted area of HEV, v is the speed and a is the acceleration, m is the curb weight of vehicle, f is the rolling resistance coefficient and g is the acceleration of the gravity.

The relationship between total power demand and the electric drive power is formulated as follows, during propelling:

$$P_e = \frac{1}{\eta_m \eta_T \eta_{inv}} P_{dem} \tag{10}$$

for generating:

$$P_e = \eta_r P_{dem} \tag{11}$$

where P_e is the electric drive power, η_m and η_T denote the efficiency of motor and transmission, respectively, η_{inv}

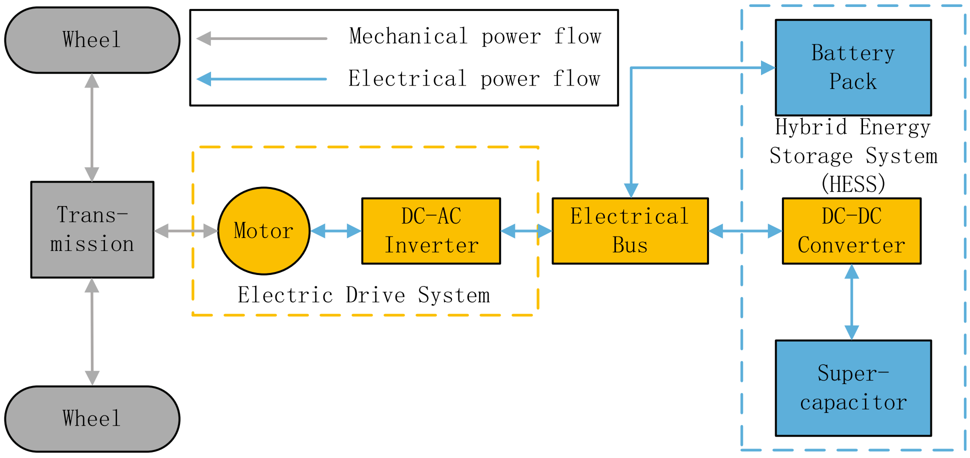


Fig. 2 Configuration and structure of EV

Table 1 Operation mode options of the HESS [31]

Mode	Power demand P_{dem}	Battery output powerP_{bat}	Supercapacitor output power P_{sc}
1	$P_{dem} \geq 0$	$P_{bat} \geq 0$	$P_{sc} \geq 0$
2	$P_{dem} \geq 0$	$P_{bat} < 0$	$P_{sc} \geq 0$
3	$P_{dem} \geq 0$	$P_{bat} \geq 0$	$P_{sc} < 0$
4	$P_{dem} < 0$	$P_{bat} < 0$	$P_{sc} \geq 0$
5	$P_{dem} < 0$	$P_{bat} \geq 0$	$P_{sc} < 0$
6	$P_{dem} < 0$	$P_{bat} < 0$	$P_{sc} < 0$

Table 2 Parameters of the considered electric vehicle [30]

Items	Parameters
Vehicle	Curb weight m: 2184 kg
	Frontal area A: 2.35 m^2
	Rolling resistance coefficient f: 0.0092
	Aerodynamic coefficient C_d: 0.24
Battery	420 Saft 10.8 V, 12 Ah Li-ion modules, 12 strings of 35 modules in series
	capacity: 59.6 kWh
Supercapacitor	139 Maxwell BCAP1200 in series
	capacity: 0.168 kWh

denotes the DC–AC inverter efficiency and η_r is the average efficiency of the regenerative braking.

As the considered HESS in EV contains two energy sources, the electric drive power should be provided by them together; when supercapacitor is discharging:

$$P_e = P_{bat} + P_{sc}\eta_{conv} \tag{12}$$

and when supercapacitor is charging:

$$P_e = P_{bat} + \frac{P_{sc}}{\eta_{conv}} \tag{13}$$

where P_{sc} and P_{bat} are the output power from supercapacitor and battery, respectively. η_{conv} is the DC–DC converter efficiency.

Basic parameters of the vehicle are given in Table 2.

2.3.2 Supercapacitor model

The supercapacitor can be modeled as a series combination of an ideal capacitor and resistance R_{sc} [32]. The state of charge (SOC) of supercapacitor is determined by [33]:

$$SOC_{sc} = \frac{V_{sc}^{oc}}{V_{sc\max}^{oc}} \tag{14}$$

where V_{sc}^{oc} and $V_{sc\max}^{oc}$ are the open-circuit voltage across the capacitor and the maximum voltage across the capacitor, respectively. According to the dynamic characteristic of the ideal capacitor, V_{sc}^{oc} satisfies the following equation:

$$\dot{V}_{sc}^{oc} = -\frac{I_{sc}}{C_{sc}} \tag{15}$$

where I_{sc} and C_{sc} are the output current and capacitance of the supercapacitor, respectively. I_{sc} can be calculated by:

$$I_{sc} = \frac{V_{sc}^{oc} - \sqrt{(V_{sc}^{oc})^2 - 4R_{sc}P_{sc}}}{2R_{sc}} \tag{16}$$

where V_{sc}^{oc}, R_{sc} and P_{sc} are the supercapacitor's voltage, equivalent resistance and charging/discharging power, respectively. Hence, the variation of SOC_{sc} is given by:

$$\dot{SOC}_{sc} = -\frac{SOC_{sc}V_{sc\max}^{oc} - \sqrt{(SOC_{sc}V_{sc\max}^{oc})^2 - 4R_{sc}P_{sc}}}{2R_{sc}C_{sc}V_{sc\max}^{oc}} \tag{17}$$

2.3.3 Battery model

A Li-ion battery is considered as the traction battery for electric vehicles, and it is modeled as an ideal open-circuit voltage module connecting an internal resistance in series. The battery's state of charge (SOC_{bat}) is determined by:

$$\dot{SOC}_{bat} = -\frac{I_{bat}}{Q_{bat}} \tag{18}$$

where Q_{bat} is the nominal capacity of battery and I_{bat} is the output current of battery and can be calculated by [10]:

$$I_{bat} = \frac{V_{bat}^{oc} - \sqrt{\left(V_{bat}^{oc}\right)^2 - 4R_{bat}P_{bat}}}{2R_{bat}} \tag{19}$$

where V_{bat}^{oc}, R_{bat} and P_{bat} are the battery's open-circuit voltage, internal resistance and charging/discharging power, respectively. Considering the different battery dynamics of charging and discharging, we have:

$$R_{bat} = R_{bat,ch} \tag{20}$$

for charging and:

$$R_{bat} = R_{bat,disch} \tag{21}$$

for discharging, where $R_{bat,ch}$ and $R_{bat,disch}$ are internal resistances of battery while charging and discharging, respectively. From Eq. (18) and Eq. (19), we can obtain that:

$$\dot{SOC}_{bat} = -\frac{V_{bat}^{oc} - \sqrt{\left(V_{bat}^{oc}\right)^2 - 4R_{bat}P_{bat}}}{2R_{bat}Q_{bat}}. \tag{22}$$

Also, a dynamic semi-empirical battery degradation model in [34] is introduced to represent battery capacity loss. It is assumed that the battery degradation model in [34] can also be applied to describe dynamic battery capacity loss on the basis of cumulative damage theory [35]. Based on the original battery degradation model, a discrete version is adopted [10]:

$$Q_{loss,p+1} - Q_{loss,p} = \Delta A_h z B^{\frac{1}{z}} e^{\frac{-E_a + A \cdot C_{rate}}{zRT_{bat}}} Q_{loss,p}^{\frac{z-1}{z}} \tag{23}$$

where $Q_{loss,p+1}$ and $Q_{loss,p}$ are total accumulated battery capacity loss at instants t_{p+1} and t_p ranging from 0 to 1, B is the pre-exponential factor, E_a is the activation energy ($J\ mol^{-1}$), C_{rate} is the battery discharge rate, while A is the compensation factor of C_{rate}, R is the gas constant in $\mathrm{J}\,(\mathrm{mol}^{-1}/\mathrm{k})^{-1}$, T_{bat} is the battery absolute temperature in K, z is the power law factor, and ΔA_h is defined as:

$$\Delta A_h = \frac{1}{3600}\int_{t_p}^{t_{p+1}} |I_{bat}|\,\mathrm{d}t. \tag{24}$$

Values of the parameters in this model are provided in [34].

2.3.4 Energy management model

The central objective of the EMS in this work is to minimize the cost function J involving the accumulated battery capacity loss and the electricity consumption for a complete driving cycle, and it can be expressed as follows:

$$\begin{aligned} J &= \sum_{k=1}^{N} J(k) = \sum_{k=1}^{N} \left(\varphi J_{bat,\,loss}(k) + \psi J_{energy}(k)\right) \\ J_{bat,\,loss}(k) &= \frac{|I_{bat}(k)|T_s}{3600} z B^{\frac{1}{z}} e^{\frac{-E_a + A \cdot C_{rate}}{zRT_{bat}}} Q_{loss,\,k-1}^{\frac{z-1}{z}} \\ J_{energy}(k) &= \frac{1}{3600}\left(P_{bat}(k) + P_{sc}(k) + P_{loss}(k)\right) \\ k &= 1,\ 2,\ \ldots,\ N \end{aligned} \tag{25}$$

where $J_{bat,loss}(k)$ denotes the battery capacity loss occurring at discrete k th time step and T_s is the simulation time step size set as 1 s, $J_{energy}(k)$ represents the energy consumption during k th time step, while P_{loss} is the power loss of the HESS, including resistance loss of battery and supercapacitor and DC/DC efficiency loss, φ and ψ are weighting coefficients, $\pi : S \rightarrow A$ is the number of time steps during one driving cycle. Then the energy management problem can be considered as a temporally limited optimization problem with the following constraints:

$$\begin{aligned} &0.5 \le SOC_x(k) \le 0.9 \\ &-247 \le P_{bat}(k) \le 247\ (\mathrm{kW}) \\ &-208 \le P_{sc}(k) \le 208\ (\mathrm{kW}) \end{aligned} \tag{26}$$

for $x = bat,\ sc$.

3 Energy management strategy

In this section, key concepts of DQL-based EMS are defined by considering the system dynamics in 3.1, followed by the algorithm design of the EMS in 3.2 and theoretical analyses on the algorithm in 3.3.

3.1 Key concepts of DQL-based EMS

In this part, key concepts of the DQL-based EMS are defined. To describe a reinforcement problem as a MDP, the problem statement has to satisfy Markov property [36]. The state signal retaining all the important information for the agent's behavior in the future is said to have Markov property. In other words, in this case, the agent's behavior should be independent of the history. Here, we define the state space by considering the system dynamics.

Firstly, according to Eq. (9), the power needed to drive the vehicle P_{dem} is determined by the velocity and the acceleration, which are strongly dependent on the previous moving state. Therefore, current P_{dem} is related to previous P_{dem}, so P_{dem} should be concluded in the state space.

Algorithm 1: Deep Q-learning using BO with TPE

```
Initialization of DQL: replay memory $D = \varnothing$ with capacity $N_D$, Q-function network $Q$ and target Q-function network $\hat{Q}$ with random weights $\theta$, number of episodes $M$, number of iterations in a deep Q-learning episode $T_{ep}$;
Initialization of BO with TPE: an original history of data points $H = \varnothing$, prior distribution of $g(x)$ and $l(x)$, number of iterations in hyperparameter optimization $T_{BO}$.
1  For $t_{BO} = 1$: $T_{BO}$ do
2      Find $x^*$ minimizing $\frac{g(x)}{l(x)}$ for each hyperparameter $x$ in $\mathbf{x}_T$
3      For episode = 1: M do
4          Reset environment: $s_0 = [SOC_{sc}(0), P_{dem}(0)]$
5          For t = 1: $T_{ep}$ do
6              With probability $\varepsilon$ select a random action $a_t$
               otherwise select $a_t = \max_{a_t} Q(s_t, a; \theta)$
7              Execute $a_t$ and observe reward $r_t$ and next state $s_{t+1}$
8              Store $(s_t, a_t, r_t, s_{t+1})$ in memory $D$
9              Sample random minibatch of transitions $\bar{D} = (s_t, a_t, r_t, s_{t+1})_{t=1}^{n}$
10             if episode terminates at step $i+1$: set $y_i = r_i$
               else set $y_i = r_i + \gamma \max_{a_{i+1}} \hat{Q}(s_{i+1}, a_{i+1}; \theta^-)$
11             Perform gradient descent on $(y_i - Q(s_i, a_i; \theta))^2$ with respect to Q-network parameters $\theta$
12             Every $C$ steps reset $\hat{Q} = Q$
13         end for
14     end for
15     $y^* = -r_{T_{ep}}$
16     $H = H \cup (\mathbf{x}_T, y^*)$
17     Split $H$ into two groups: the best performing one and the rest, define $y^*$ as the splitting value
18     Build the surrogate model $M$ for $l(x)$ and $g(x)$
19 end for
```

Secondly, SOC_{bat} and SOC_{sc} are calculated based on the previous remaining energy of battery and supercapacitor, and they are also used to judge the energy consumption, which is one of the optimization objectives and directly influences the immediate reward. Therefore, SOC_{bat} and SOC_{sc} should be concluded in the state space.

Thirdly, according to Eq. (23) of the battery degradation model, the degradation happened in the next time step is related to the current aging state of the battery, so $J_{bat,\,loss}$ should also be concluded.

Hence, the definition of state variables can be determined as:

$$s(k) = [P_{dem}(k),\ SOC_{bat}(k),\ SOC_{sc}(k),\ J_{bat,\,loss}(k)]^{\mathrm{T}} \tag{27}$$

Since the basic idea of EMS for HESS is to properly allocate power in demand between energy sources to alleviate battery degradation, we select output power of battery as the action variable, and it is denoted as:

$$a(k) = P_{bat}(k) \tag{28}$$

The action space includes 17 options representing discretized $P_{bat}(k)$ in kW, and it is defined as:

$$A = \{-60,\ -42,\ -30,\ -20,\ -10,\ -7,\ -4,\ -2,\ 0,\ \ldots, 2,\ 4,\ 7,\ 10,\ 20,\ 30,\ 42,\ 60\} \tag{29}$$

The immediate reward is defined as:

$$r_{ss'}^{a} = \alpha J(k) \tag{30}$$

where α is the weight factor of the optimization objective J defined in Eq. (25) with $\alpha < 0$.

3.2 Algorithm design of DQL-based EMS

The details of DQL algorithm are described in Algorithm 1. In this algorithm, the hyperparameter optimization process using BO and TPE is set as the outer loop [26]. The second outer loop is performed during one training episode, while the inner loop represents the control and learning process at each time step. Inspired by [25], a target Q-network $\hat{Q}$ with weights θ^- is used to output the Q-value. For the balance between exploration and exploitation, $\varepsilon -$ *greedy* policy is adopted in step 4, i.e., the action with the maximum Q-value is chosen with probability $1 - \varepsilon$ and a random action is selected with probability ε. To get rid of the correlations between the training samples, experience replay [25] is applied to store the experience of time step t_k as $(s_k,\ a_k,\ r_k,\ s_{k+1})$ in an experience memory D in step 6. For each training time step, D is randomly sampled as $(s_j,\ a_j,\ r_j,\ s_{j+1})$ to form a minibatch to update the weights of deep Q-network.

The complete process of DQL-based EMS is illustrated in Fig. 3, which demonstrates that the developed EMS can autonomously learn the optimal control policy only based on real-time data, without any prior knowledge.

3.3 Theoretical analyses on the algorithm of DQL-based EMS

By describing the problem in the above way, the deep Q-network in DQL algorithm is supposed to be able to finally map all the states into optimal Q-values. The ability to deal with states with various dynamics (e.g., the internal resistance of battery varying with operating modes and SOC_{bat}, the battery degradation having an exponential relationship with energy loss in the battery and various real-time

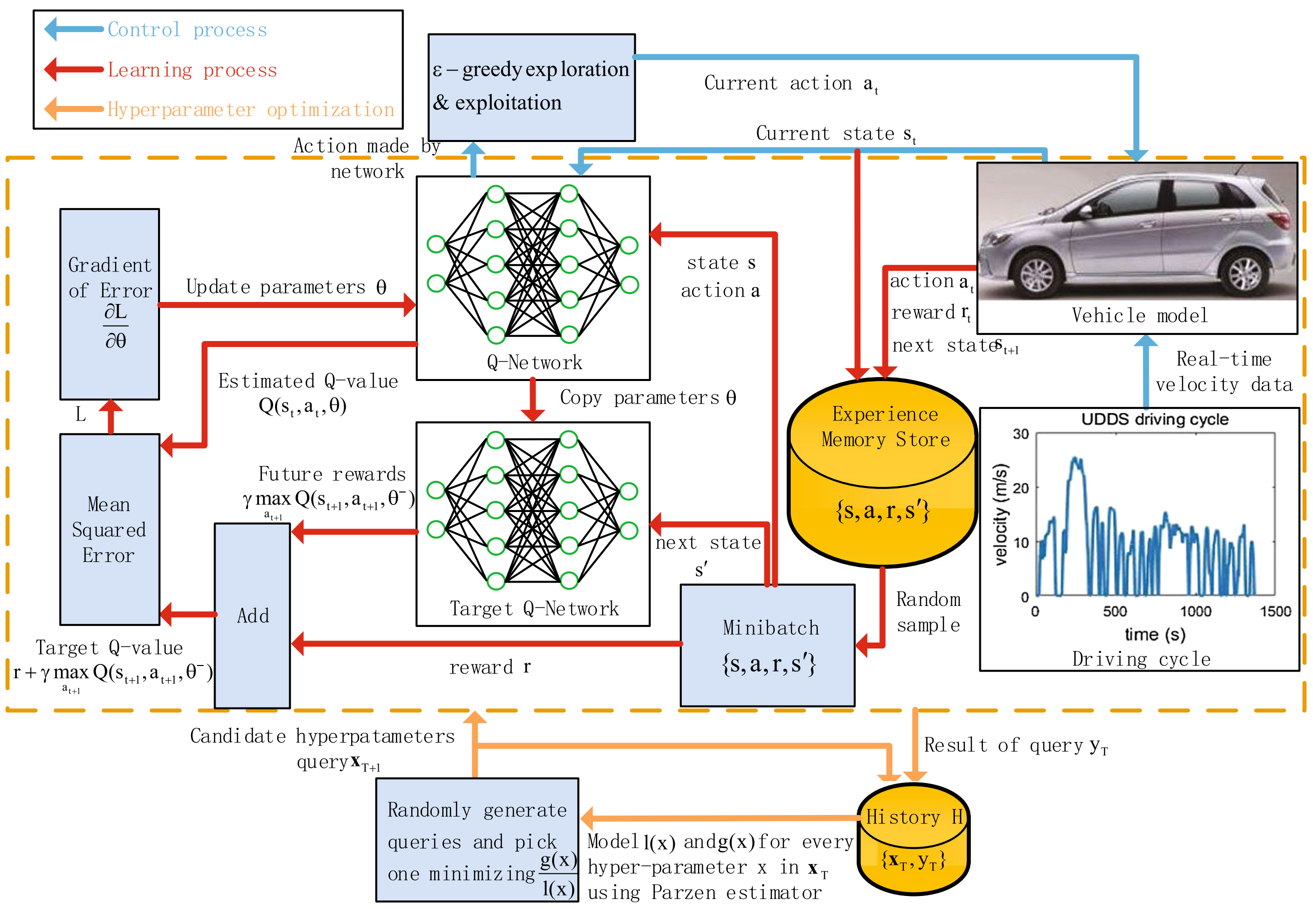

Control process
Learning process
Hyperparameter optimization
ε – greedy exploration & exploitation
Current action a_t
Current state s_t
Action made by network
Gradient of Error $\frac{\partial L}{\partial \theta}$
Update parameters θ
Q-Network
state s
action a
action a_t
reward r_t
next state s_{t+1}
Vehicle model
Real-time velocity data
L
Estimated Q-value $Q(s_t, a_t, \theta)$
Copy parameters θ
Experience Memory Store
{s,a,r,s′}
UDDS driving cycle
velocity (m/s)
time (s)
Driving cycle
Mean Squared Error
Future rewards $\gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta^-)$
Add
Target Q-Network
next state s′
Minibatch {s,a,r,s′}
Random sample
Target Q-value $r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta^-)$
reward r
Candidate hyperpatameters query $\mathbf{x}_{T+1}$
Result of query y_T
Randomly generate queries and pick one minimizing $\frac{g(x)}{l(x)}$
Model l(x) and g(x) for every hyper-parameter x in $\mathbf{x}_T$ using Parzen estimator
History H
$\{\mathbf{x}_T, y_T\}$

Fig. 3 Process of DQL-based energy management strategy

velocity of driving cycles representing complicated traffic conditions, etc.) is called adaptability.

The reason behind this ability is the "experience replay" mechanism introduced to achieve stability in DQL [25]. In this mechanism, a "replay buffer" is created to store historical experience tuples, and at every time step, a group of independent transitions are sampled from the buffer as training samples for DQN.

By doing this, the training process of agent involves more efficient use of previous experience, because theoretically, all the experience can be used for training the Q-network. Therefore, the network is supposed to deal with any possible, various dynamics of the system, as long as training iterations are sufficient. So we can conclude that:

Remark 3.1 It is the "experience replay" mechanism that empowers the network with the adaptability to generate optimal Q-values under various states, representing different traffic conditions and complicated dynamics of vehicle and battery.

In addition, as mentioned in Sect. 2.1, the updating process of Q-network is led by the error between the real Q-value $Q(s, a, \theta_i)$ and the target Q-value $y^i_{s,a}$ in i th iteration, where $y^i_{s,a} = r + \gamma \max_{a'} Q(s', a', \theta^-_i)$. This error is called target approximation error (TAE), denoted by $Z^i_{s,a}$, which is supposed to be a random process with $E[Z^i_{s,a}] = 0$ and $Var[Z^i_{s,a}] = \sigma^2_s$, and it's defined as:

$$Z^i_{s,a} = Q(s, a, \theta_i) - y^i_{s,a}. \tag{31}$$

In [37], the considered reinforcement learning problem is first assumed to be an M-state unidirectional MDP, where the agent starts at s_0 and ends at s_{M-1}, and then for $i > M$, the variance of DQN is analyzed. Inspired by [37], we analyze the convergence of the EMS in the final learning episodes, where the choice of action of Q-network does not involve exploration and a single episode can be seen as an M-state unidirectional MDP. Here the Q-value $Q(s_0, a_0; \theta_i)$ is expressed as follows:

$$\begin{aligned} Q(s_0, a_0; \theta_i) &= Z^i_{s_0,a_0} + y^i_{s_0,a_0} \\ &= Z^i_{s_0,a_0} + \gamma Q(s_1, a_1; \theta^-_i) \end{aligned}. \tag{32}$$

It is worth noting that, firstly, we consider the reward to be zero $(r = 0)$ everywhere since it has no effect on variance [37]; secondly, since there is no exploration in action,

a_1 is exactly the action to maximize target Q-value. Then, we have:

$$\begin{aligned} Var[Q(s_0, a_0; \theta_i)] &= Var[Z^i_{s_0,a_0}] + Var[y^i_{s_0,a_0}] \\ &= \sigma^2_{s_1} + \gamma Var[Q(s_1, a_1; \theta_i^-)] \end{aligned}. \quad (33)$$

That is, the variance of Q-value consists of variance of TAE and variance of target Q-value. When $\gamma = 1$, we can deduce that the variance of Q-value in s_1 is smaller than that in s_0, and this can easily be extended to other states in the same episode. Then we have:

Remark 3.2 Assume the discount factor γ to be 1, the variance of Q-value decreases with the agent passing along every state in a single learning episode without exploration, which would benefit DQN's convergence in the last few stages of the learning process.

4 Simulation results and discussion

In this section, the effectiveness of the DQL-based EMS is evaluated on the considered HESS and electric vehicle. Firstly, the effectiveness of the hyperparameter tuning method is validated by comparison with random search. Then, DQL-based EMS is compared with a previously published rule-based EMS [10] under both the training dataset and the testing dataset, respectively.

Urban Dynamometer Driving Schedule (UDDS) cycle is applied as the training dataset. New York City Cycle (NYCC), West Virginia University Suburban (WVUSUB) and Highway Fuel Economy Test (HWFET), representing urban, suburban and highway driving cycle, respectively, are employed to constitute a combined testing driving cycle.

The combined cycle is employed as the testing dataset. The combined cycle is depicted in Fig. 4. In addition, some characteristics of UDDS and combined cycle are compared in Table 3. Differences in velocity and acceleration of two cycles indicate that they represent different driving

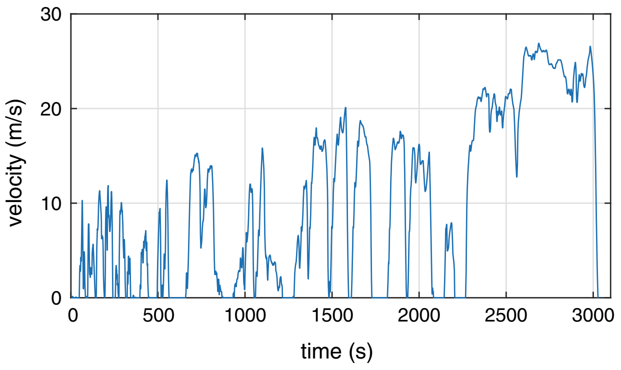


Fig. 4 Combined testing driving cycle

Table 3 Statistic comparison between UDDS and combined testing driving cycle

Driving cycle	UDDS	Combined cycle
Maximum velocity (m/s)	25.458	26.895
Average velocity (m/s)	8.790	10.069
Maximum acceleration (m/s^2)	1.482	2.694

conditions; therefore, the combined testing driving cycle is suitable for validating the adaptability of the developed strategy.

We employ GTX 1080 as well as AMD Ryzen 7 1700 processor running at 3.0 GHz as our hardware platform. The DQL-based EMS is implemented using Tensorflow 1.11.0. As for the hyperparameter tuning process, we apply hyperopt library [38] to implement TPE algorithm. We set all the hidden layers as fully connected tanh layers, and the output layer is chosen as a fully connected sigmoid layer with 17 neurons representing Q-values of all possible options of discrete P_{bat}. The exploration rate ε is set as initial exploration at the beginning of the training process, and then it decreases at a constant speed with iterations and finally reaches the final exploration rate.

Then both the architecture of the deep network and parameters of the learning process are all optimized. The search spaces for hyperparameters, i.e., the intervals that the hyperparameters fall into or options the hyperparameters choose, are shown in Table 4, where the top 5 best trails of hyperparameters of BO with TPE are also exhibited.

Despite the fact that it is tough to figure the relationship between settings and the performance of a neural network, the five results are sharing some features following some laws or previous studies on hyperparameters.

Firstly, the discount factor γ determines to what extent future rewards are taken into consideration by the agent. We can find that in Table 4, all the γ are more than 0.6 and close to 1, which is beneficial to the optimization in the long run. Secondly, compared with RMSProp optimizer, the gradient descent optimizer is characterized by higher training time and more likely to fall into a local minimum, and it is shown in Table 4 that RMSProp is applied in all the results. Next, the best learning rate is neither valued too high nor too low but at around 0.04–0.05 in order not to skip over the global minimum or fall into a local minimum. Finally, the network is not oversized and its depth and width are similar to the network sizes in other works using reinforcement learning [16], whose performance has been validated.

For comparison, random search (RS) [20] is implemented and allocated the same time budget with BO, and the best result of RS is also shown in Table 4. It can be

Table 4 Results of hyperparameter optimization using two methods

Hyperparameters	Search space	BO with TPE					Random Search
		Trial 1	Trail 2	Trial 3	Trail 4	Trail 5	Trail 1
Discount factor γ	[0, 1]	0.65	0.62	0.81	0.73	0.73	0.90
Optimizer	RMSProp, gradient descent	RMSProp	RMSProp	RMSProp	RMSProp	RMSProp	RMSProp
Learning rate	[0.00001, 0.1]	0.0721	0.0838	0.0434	0.0441	0.0453	0.097
Memory size N_D	[50, 10,000]	4386	3476	1087	1492	1398	9692
Minibatch size	[1, 50]	26	44	30	25	26	18
Initial exploration ε	[0.5, 1]	0.89	0.72	0.87	0.87	0.89	0.93
Final exploration ε	[0, 0.5]	0.1	0	0.13	0.07	0.07	0.05
Replay period C	[100, 500]	320	259	499	285	286	367
Hidden layers' number	[1, 5]	3	2	4	3	3	2
Neurons in layer 1	[1, 100]	16	28	72	17	21	5
Neurons in layer 2	[1, 100]	19	19	69	23	19	5
Neurons in layer 3	[1, 100]	91		41	88	86	
Neurons in layer 4	[1, 100]			45			
Neurons in layer 5	[1, 100]						
Final reward of this configuration		− 1168	− 1165	− 1162	− 1151	− 1137	− 1304

seen that the best performer of BO with TPE (trail 5 in bold type) achieves reward at -1137 when the learning process ends, while the best reward of RS is -1304. Although RS performs very well especially in some complicated search problems [39], the hyperparameter tuning method outperforms RS in this problem. A further comparison between the best results of both the hyperparameter tuning approaches is made during the learning process.

4.1 Learning performance

In this part, the Q-network with hyperparameters tuned by both BO and RS algorithm is trained with training dataset. A total of 300 episodes are used during the training process, and each episode means a complete UDDS cycle. The track of average loss defined in Eq. (3) in every episode is demonstrated in Fig. 5. It is apparent that the average loss with hyperparameters tuned by BO continuously decreases at first and then fluctuates around 0.055, which indicates that the training loss has been reduced effectively. In contrast, the counterpart using RS is often less satisfying. Figure 6 records the trend of total reward of each episode. In both tracks, violent fluctuations at the beginning indicate that the RL agent cannot make favorable decisions and exploration is frequently used. However, the track optimized by BO ascends with training episodes, and after about 100 episodes the value of total reward becomes significantly improved and stabilized at a historical high position, representing the effectiveness of learning. In contrast, fluctuations are not relieved in the result of RS, indicating a poor convergence performance.

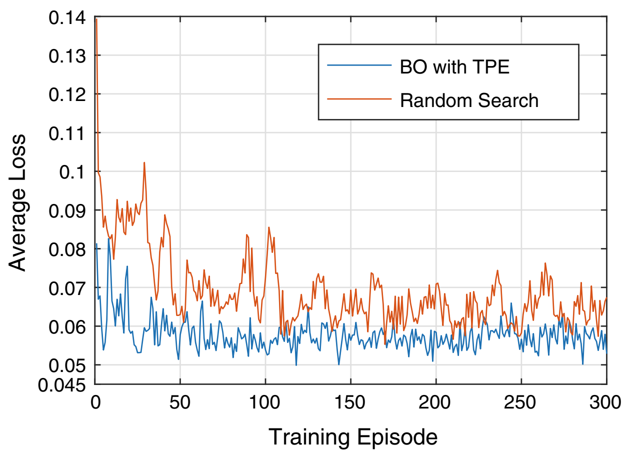


Fig. 5 Average loss of DQL

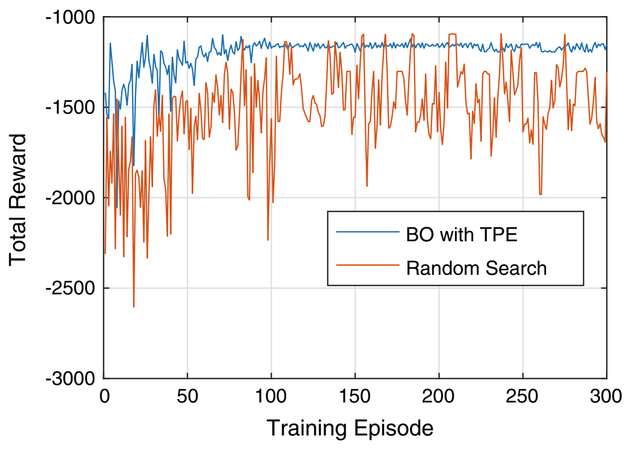


Fig. 6 Total reward of DQL

4.2 Optimality

In this part, the DQL-based algorithm is compared with a previous work [10] to validate both optimality and adaptability. The rule-based strategy proposed in [10] is controlled by four fixed parameters, i.e., P_{hd} and $P_{\min}$ are two values deciding the operation mode of HESS, P_{high} is the maximum output power of the battery and finally P_{ch} is the charging power from battery to supercapacitor in some cases. It is worth noting that in [10], P_{dem} denotes powertrain power demand draw from the inverter, so it is equal to P_e in this work. These parameters should be adjusted well to fit specific vehicle model and driving environment in this study. Motivated by [40], we set the parameters with all possible values to search for the best performance. The determined values are listed in Table 5.

In practice, the two EMSs finish UDDS with different terminal SOC_{sc}, which may indicate a diverse amount of battery electricity used. For a fair comparison, equivalent battery capacity loss Q_{eq_loss}, as a measure of efficiency of prolonging battery life, is defined as battery capacity loss per kilowatt battery power and it is formulated as:

$$Q_{eq\_loss} = \frac{Q_{loss,\,N}}{\sum_{k=1}^{N} |P_{bat}(k)|} \qquad k = 1, \ldots, N \tag{34}$$

The comparison results between the two EMSs are presented in Table 6. The rule-based EMS achieves 97.62% battery life prolonging and 99.14% energy economy when the DQL-based one is seen as a 100% benchmark, that is, DQL-based EMS performs 2.38% and 0.86% better than the rule-based EMS, respectively. It is worth noting that the rule-based EMS is extracted from DP results and it can be seen as a near-optimal strategy [10], so the optimality of DQL-based EMS is validated.

The detailed power allocations of the DQL-based and the rule-based strategy under training dataset are exhibited in Fig. 7a, b, respectively, and SOC_{sc} trajectories of both EMS are shown in Fig. 7c. As shown, whether in instantaneous peak power demand or in long-term normal use, the rule-based EMS obtains higher battery peak power and severe fluctuation of battery power than the DQL-based EMS.

Table 5 Values of parameters in rule-based EMS after adjustment

Parameters of rule-based EMS	Training dataset	Testing dataset
P_{ch}	1	1
$P_{\min}$	3	7
P_{high}	11	11
P_{hd}	13	15

For example, when the power demand reaches the highest value at around 200 s, the supercapacitor cannot afford positive output due to low remaining electricity, so the battery has to fully provide all the driving power. In contrast, the DQL-based method successfully protects battery from the highest power. Another example appears in 800–1000 s, where the DQL-based EMS keeps battery power load in a smaller range compared with the rule-based EMS. The reason of the improvements is that in DQL-based EMS, agent tends to charge supercapacitor from battery when the SOC_{sc} is in a low level and the electric drive system is either idle or coping with low power load; for example, when it is 0–25 s in Fig. 7a, c, battery charges the supercapacitor when the vehicle is stationary. Therefore, SOC_{sc} is kept in a relatively high position to share possible peak power in the future. In this way, battery life is extended in the developed method.

4.3 Adaptability

Although the optimality of the DQL-based EMS has been validated, it could be faced with uncertainty in an unfamiliar scenario or untrained environments. Therefore, in this part, both EMSs are implemented on the testing dataset to validate the adaptability. Comparisons are made in Fig. 8, where the distributions of the battery power P_{bat} of different cases are plotted. The distribution of P_{bat} in EV equipped with the same battery considered by the DQL-based EMS as the single energy source, i.e., the battery-only configuration, is also plotted. The vertical axis represents the frequency that P_{bat} appears with in defined intervals.

Firstly, it is seen that compared with battery-only configuration, P_{bat} of both EMSs avoid intervals on both sides and gather in the central area which is characterized by low power load and alleviative current stress on the battery. In addition, compared with rule-based EMS, frequencies that battery power of DQL-based EMS falls in intervals $(21, +\infty)$,$(12, 21]$ and $(-1, 1]$ in Fig. 8a, b are smaller than those of the rule-based EMS. This phenomenon indicates that DQL-based strategy tends to spread the power load on daily operation and peak power demands can be shared more equally, and this character is kept under distinct conditions.

Detailed statistic results are recorded in Table 7. It is seen that the DQL-based EMS performs almost the same as the rule-based EMS in battery life prolonging, and the electricity economy improvement brought by DQL-based method reaches 10.78% (from 89.22 to 100%). Since the rule-based strategy is adjusted again to obtain the best performance, it can be concluded that the optimality of DQL-based EMS is still guaranteed under testing environments and the adaptability is also confirmed.

Table 6 Comparison between DQL-based EMS and rule-based EMS under training dataset

Energy management strategy	Equivalent battery capacity loss (%/kW)	Battery life prolonging (%)	Energy consumption (kWh)	Energy economy (%)
DQL-based	0.1436	100	1.3197	100
Rule-based	0.1471	97.62	1.3312	99.14

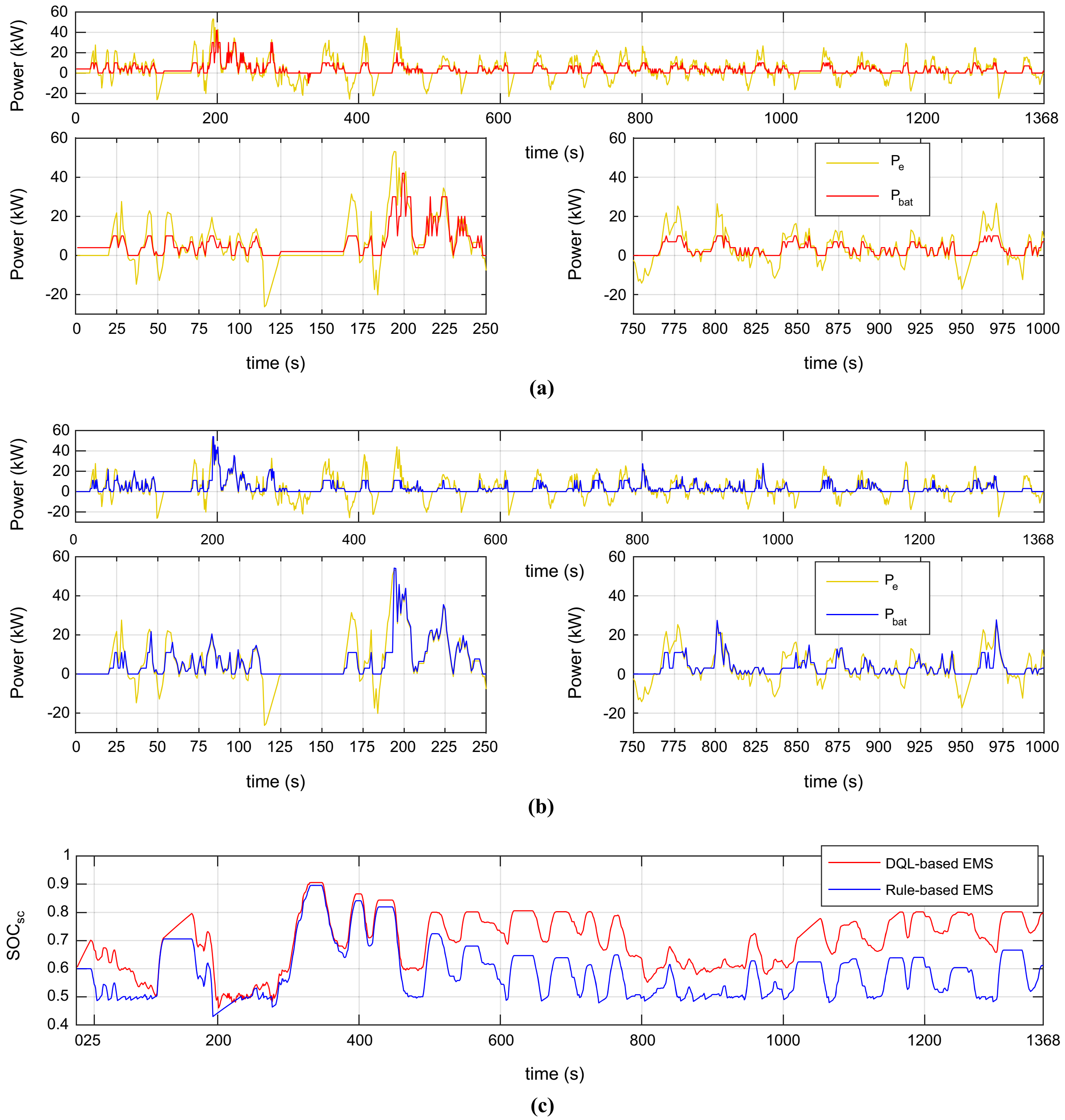
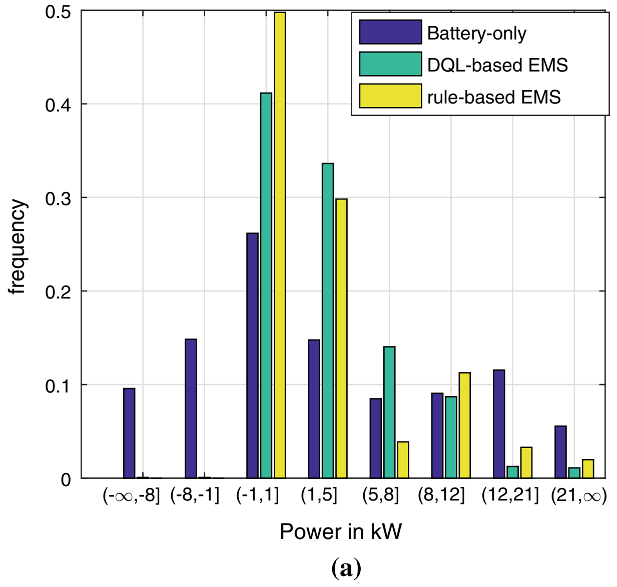


Fig. 7 Power split results of **a** DQL-based EMS, **b** rule-based EMS and **c** SOC_{sc} trajectory under UDDS, **a**, **b** also provide details with enlarged scale



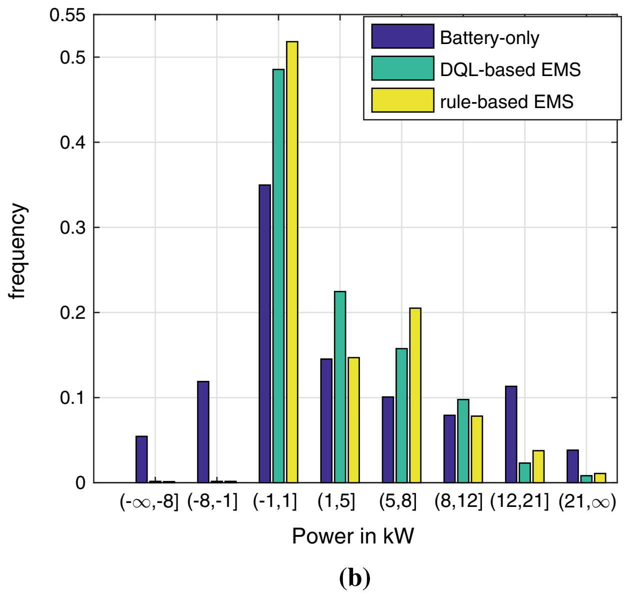


Fig. 8 Battery power distribution of both EMSs and battery-only configuration under **a** training dataset and **b** testing dataset

4.4 Time consumption

Time consumption of DQL-based EMS can be classified into two parts: training time and inference time. Firstly, it takes 2 h and 57 min to optimize hyperparameters for DQL algorithm because every query of Bayesian optimization is time-consuming. Secondly, inference times of DQL-based EMS are 5.661089 s on training dataset and 7.91536 s on testing dataset, respectively.

As for rule-based EMS, it costs 1 h and 3 min to adjust parameters shown in Table 5, and the inference time is 0.009772 s on training dataset and 0.011598 s on training dataset, respectively. The cause of the difference between the two methods is that computations on neural networks are much more complicated than those on simple rule-based programs.

It seems that the rule-based EMS is more time-saving; however, it is worth noting that rules in the rule-based method are extracted from dynamic programming results, which requires knowledge on the environment in advance and human expertise, so it is tough to realize in real applications. In contrast, the DQL-based EMS is an efficient autonomous learning method without prior knowledge and manual intervention.

5 Conclusion

In this paper, a deep *Q*-learning-based energy management strategy of the hybrid energy storage system has been developed for electric vehicles. In addition, in order to properly tune the hyperparameters of the DQL algorithm, we have employed Bayesian optimization (BO) with tree-structured Parzen estimator (TPE) algorithm as the hyperparameter optimization approach. Simulation results have shown that the DQL algorithm with hyperparameters tuned by Bayesian optimization has a better learning performance than the DQL using random search. It has also been validated that the developed EMS outperforms a previously published near-optimal rule-based EMS in terms of both battery life prolonging and energy economy on both the training dataset and the testing dataset. For future development, the application of more advanced reinforcement learning technologies that are trained with higher efficiency and can reduce both training time and inference time will be accomplished.

Table 7 Comparison between DQL-based EMS and rule-based EMS on the testing dataset

Energy management strategy	Equivalent battery capacity loss (%/kW)	Battery life prolonging (%)	Energy consumption (kWh)	Energy economy (%)
DQL-based	0.1006	99.90	2.6291	100
Rule-based	0.1005	100	2.9468	89.22

Acknowledgements This research was supported by the National Science and Technology Support Program under grant No 2014BAG06B02, Fundamental Research Funds for the Central Universities under grant No 2014HGCH0003 and the National Natural Science Foundation of China under Grant 61771178.

Compliance with ethical standards

Conflict of interest The authors declare that they have no competing interests in the present work.

References

1. Chan CC, Wong YS, Bouscayrol A, Chen K (2009) Powering sustainable mobility: roadmaps of electric, hybrid, and fuel cell vehicles [point of view]. Proc IEEE 97(4):603–607
2. Wang B, Xu J, Cao B, Ning B (2017) Adaptive mode switch strategy based on simulated annealing optimization of a multi-mode hybrid energy storage system for electric vehicles. Appl Energy 194:596–608
3. Song Z, Li J, Han X, Xu L, Lu L, Ouyang M et al (2014) Multi-objective optimization of a semi-active battery/supercapacitor energy storage system for electric vehicles. Appl Energy 135:212–224
4. Cao J, Emadi A (2012) A new battery/ultracapacitor hybrid energy storage system for electric, hybrid, and plug-in hybrid electric vehicles. IEEE Trans Power Electron 27(1):122–132
5. Zhang L, Hu X, Wang Z, Sun F, Dorrell DG (2015) Experimental impedance investigation of an ultracapacitor at different conditions for electric vehicle applications. J Power Sources 287:129–138
6. Wang B, Xu J, Cao B, Qiyu L, Qingxia Y (2015) Compound-type hybrid energy storage system and its mode control strategy for electric vehicles. J Power Electron 15(3):849–859
7. Trovão JP, Pereirinha PG, Jorge HM, Antunes CH (2013) A multi-level energy management system for multi-source electric vehicles—an integrated rule-based meta-heuristic approach. Appl Energy 105:304–318
8. Blanes JM, Gutierrez R, Garrigos A, Lizan JL, Cuadrado JM (2013) Electric vehicle battery life extension using ultracapacitors and an FPGA controlled interleaved buck–boost converter. IEEE Trans Power Electron 28(12):5940–5948
9. Hu X, Johannesson L, Murgovski N, Egardt B (2015) Longevity-conscious dimensioning and power management of the hybrid energy storage system in a fuel cell hybrid electric bus. Appl Energy 137:913–924
10. Song Z, Hofmann H, Li J, Han X, Ouyang M (2015) Optimization for a hybrid energy storage system in electric vehicles using dynamic programing approach. Appl Energy 139:151–162
11. Sisakht ST, Barakati SM (2016) Energy management using fuzzy controller for hybrid electrical vehicles. J. Intell. Fuzzy Syst. 30:1411–1420
12. Santucci A, Sorniotti A, Lekakou C (2014) Power split strategies for hybrid energy storage systems for vehicular applications. J Power Source 258:395–407
13. Vinot E, Trigui R (2013) Optimal energy management of HEVs with hybrid storage system. Energy Convers Manag 76:437–452
14. Xiong R, Cao J, Yu Q (2018) Reinforcement learning-based real-time power management for hybrid energy storage system in the plug-in hybrid electric vehicle. Appl Energy 211(C):538–548
15. Wu J, He H, Peng J, Li Y, Li Z (2018) Continuous reinforcement learning of energy management with deep q network for a power split hybrid electric bus. Appl Energy 222:799–811
16. Hu Yue, Li Weimin, Xu Kun, Zahid Taimoor, Qin Feiyan, Li Chenming (2018) Energy management strategy for a hybrid electric vehicle based on deep reinforcement learning. Appl Sci 8:187. https://doi.org/10.3390/app8020187
17. Hredzak B, Agelidis VG, Jang M (2014) Model predictive control system for a hybrid battery-ultracapacitor power source. IEEE Trans Power Electron 29:1469–1479
18. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D et al (2013) Playing Atari with deep reinforcement learning. Technical report. Deepmind Technologies, arXiv:1312.5602 [cs.LG]
19. Neary P (2018) Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning. In: 2018 IEEE international conference on cognitive computing (ICCC), San Francisco, CA, pp 73–77
20. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13:281–305
21. Barsce JC, Palombarini JA, Martínez EC (2018) Towardsautonomous reinforcement learning: automatic setting of hyper-parameters using bayesian optimization. CoRR, vol. abs/1805.04748. http://arxiv.org/abs/1805.04748
22. Jones DR, Schonlau M, Welch WJ (1998) Efficient global optimization of expensive black-box functions. J Glob Optim 13(4):455–492
23. Srinivas N, Krause A, Kakade SM, Seeger M (2012) Information-theoretic regret bounds for gaussian process optimization in the bandit setting. IEEE Trans Inf Theory 58(5):3250–3265
24. Contal E, Perchet V, Vayatis N (2014) Gaussian process optimization with mutual information. In: International conference on machine learning (ICML)
25. Mnih V, Kavukcuoglu K, Silver D, Rusu A, Veness J, Bellemare G, Marc MG, Graves A, Riedmiller M, Fidjeland K, Andreas, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518:529–533. https://doi.org/10.1038/nature14236
26. Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. In: Proceedings of neural information and processing systems
27. Jones DR (2001) A taxonomy of global optimization methods based on response surfaces. J Glob Optim 21:345–383
28. Thornton C, Hutter F, Hoos HH, Leyton-Brown K (2013) Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the Knowledge discovery and data mining, pp 847–855
29. Xia Y, Liu C, Li YY, Liu N (2017) A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring. Expert Syst Appl 78:225–241
30. Meyer RT, DeCarlo RA, Pekarek S (2016) Hybrid model predictive power management of a battery-supercapacitor electric vehicle. Asian J Control 18(1):150–165
31. Yue SA, Wang YA, Xie QA, Zhu DA, Pedram MA, Chang NB (2015) Model-free learning-based online management of hybrid electrical energy storage systems in electric vehicles. In: Conference of the IEEE Industrial Electronics Society. IEEE
32. Zhang S, Xiong R, Sun F (2015) Model predictive control for power management in a plug-in hybrid electric vehicle with a hybrid energy storage system ☆. Appl Energy 185:1654–1662
33. Golchoubian P, Azad NL (2015) An optimal energy management system for electric vehicles hybridized with supercapacitor. In: ASME 2015 Dynamic Systems and Control Conference. American Society of Mechanical Engineers, p V001T10A004
34. Wang J, Liu P, Hicks-Garner J, Sherman E, Soukiazian S, Verbrugge M et al (2011) Cycle-life model for graphite-lifepo4 cells. J Power Sources 196(8):3942–3948

35. Safari M, te Morcret M, Teyssot A, Delacourt C (2010) Life-prediction methods for Lithium-ion batteries derived from a fatigue approach. J Electrochem Soc 157:A713–A720
36. Barto AG, Sutton RS (1998) Reinforcement learning: an introduction. MIT Press, Cambridge
37. Anschel O, Baram N, Shimkin N (2017) Averaged-dqn: variance reduction and stabilization for deep reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning, vol 70. JMLR. org, pp 176–185
38. Bergstra J, Komer B, Eliasmith C, Yamins D, Cox DD (2015) Hyperopt: a python library for model selection and hyperparameter optimization. Comput Sci Discov 8:014008
39. Levesque J-C, Durand A, Gagne C, Sabourin R (2017) Bayesian optimization for conditional hyperparameter spaces. In: Int. Joint Conf. Neural Networks, Anchorage, Alaska, USA, pp 286–293
40. Song Z, Hofmann H, Li J, Hou J, Han X, Ouyang M (2014) Energy management strategies comparison for electric vehicles with hybrid energy storage system. Appl Energy 134(C):321–331