2017 IEEE Intelligent Vehicles Symposium (IV)
June 11-14, 2017, Redondo Beach, CA, USA

Deep Reinforcement Learning-Based Vehicle Energy Efficiency Autonomous Learning System

Xuewei Qi, *Member, IEEE*, Yadan Luo, Guoyuan Wu, Senior *Member, IEEE*, Kanok Boriboonsomsin, *Member, IEEE*, Matthew J. Barth, *Fellow, IEEE*

***Abstract*—To mitigate air pollution problems and reduce greenhouse gas emissions (GHG), plug-in hybrid electric vehicles (PHEV) have been developed to achieve higher fuel efficiency. The Energy Management System (EMS) is a very important component of a PHEV in achieving better fuel economy and it is a very active research area. So far, most of the existing EMS strategies just simple follow predefined rules that are not adaptive to changing driving conditions; other strategies as starting to incorporate accurate prediction of future traffic conditions. In this study, a deep reinforcement learning based PHEV energy management system is designed to autonomously learn the optimal fuel use from its own historical driving record. It is a fully data-driven and learning-enabled model that does not rely on any prediction or predefined rules. The experiment results show that the proposed model is able to achieve 16.3% energy savings comparing to conventional binary control strategies.**

I. Introduction

Transportation activities are responsible for a significant amount of energy consumption as well as criteria pollutant and greenhouse gas (GHG) emissions, which has attracted increasing public concerns in recent years. As reported in 2014 [1], there was approximately 24.9 Quadrillion BTUs total amount of energy consumed, and around 27% GHG emissions [2] are contributed by transportation sectors in United States. Therefore, reducing transportation-related fuel consumption and emissions has been a very active research area for years. Transportation electrification is one promising way to significantly reduce fossil fuel consumption and emissions from the transportation sector. However, the current mass adoption of battery electric vehicles has not been realized due to the limited availability of charging infrastructure, long charging times, and limited travel range per charge. Plugin hybrid electric vehicles (PHEV) is an effective way to mitigate the so called "range anxiety" [3], due to the hybrid power source.

For a PHEV, the energy management system (EMS) is very critical to achieve higher fuel efficiency, which has been a subject of active research for decades. Existing EMS strategies are generally classified into the following three different categories (see Fig.1): a) rule-based strategies that rely on a set of simple rules without *a priori* knowledge of driving conditions [4, 5]. A typical example is binary control strategy which controls the vehicle to use power from battery and switches to engine when the battery reaches the bottom state-of-charge (SOC); b) optimization-based strategies which optimize the control for the entire trip based the known or predicted future driving conditions [6, 7, 8, 9, 10]. The obvious downside of this type of models are that it is difficult to be implemented due to the lack of accurate trip information known beforehand; and a more recent category 3) learning-based which is able to learn the optimal control strategy from the recorded historical driving conditions [11]. These type of methods rely neither on the predefined simple control rules nor the predicted trip information. However, these type of strategies have yet to be fully developed.

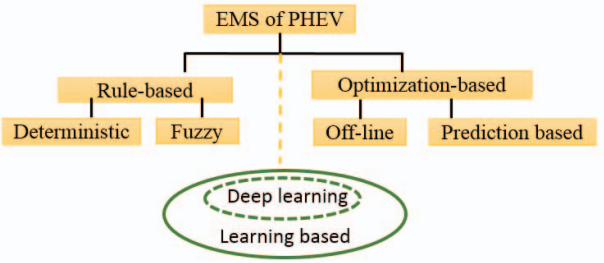

Fig. 1. Classification of existing EMS

In our previous work [11], a reinforcement learning-based EMS model was designed and tested with real-world driving data. The model is built upon the discretized environmental states and actions and the optimal control policy is actually represented by a multi-dimensional table, which impedes its application when the environment state dimension increases or the discretization becomes finer due to the exponentially increased search-space. To alleviate this so-called "curse of dimensionality", a deep reinforcement learning based EMS is proposed to autonomously learn the optimal control strategy from the historical driving data in a continuous state space. The relationship between the continuous environment states and optimal control decision is represented and captured by a neural network whose inputs are the continuous state variables (e.g., continues SOC and power-demand) and outputs are continuous action variables (e.g., engine power supply). To the best of our knowledge, this is the first to apply deep-reinforcement learning to PHEV EMS.

Xuewei Qi, Yadan Luo, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew J. Barth are with the CE-CERT, University of California Riverside, Riverside, CA 92507. email xqi001@ucr.edu, lyadanluol@gmail.com; gywu@cert.ucr.edu, kanok@cert.ucr.edu, barth@ece.ucr.edu

II. BACKGROUND

A. PHEV Modeling & Energy Management Formulation

This study is focused on the power-split PHEV and the vehicle model was developed in our previous work [12]. For the dynamic equations of the vehicle mechanical path and details about model derivation including the electrical path and parameter selection, please refer to [12]. In the control of series-parallel PHEV, The decision-making on the power-split ratio between an internal combustion engine (ICE) and battery pack is called the power-split control problem. Mathematically, the optimal energy management (i.e., power-split control) for PHEVs can be defined as a nonlinear constrained optimization problem. With discretized ICE supply power, the optimal PHEV power-split control problem in a discretized time space can be formulated as follows:

$$min \sum_{t=1}^{M} \sum_{i=1}^{N} x(t,i)\, P_i^{eng} / \eta_i^{eng} \tag{1}$$

subject to:

$$\sum_{t=1}^{j} f\left(P_t - \sum_{i=1}^{N} x(t,i) P_i^{eng}\right) \le C \quad \forall j = 1, \dots, T \tag{2}$$

$$\sum_{i=1}^{N} x(t,i) = 1 \qquad \forall t \tag{3}$$

$$x(t,i) = \{0,1\} \qquad \forall t, i \tag{4}$$

where M is the entire trip time span; N is number of discretized engine supply power level; t is the time step index; i is the index of ICE power level; C is the gap of the battery pack's state of charge (SOC) between the initial and the minimum; P_i^{eng} is the i-th discretized engine power level and η_i^{eng} is the corresponding engine efficiency; and P_t is the driving demand power at time step t. The objective of the energy management problem is to find the optimal action (i.e. selection of the optimal ICE power level) for each time step to achieve the best fuel efficiency along the entire trip. Intuitively, when the power-demand for each second is known beforehand, then this optimization problem can be easily solved by many mathematical methods. However, in most of the practical application cases, this information is unknown; therefore, reinforcement learning is adopted herein to design an EMS model that does not rely on any future trip information.

B. Reinforcement Learning Background

Reinforcement learning (RL) is originated from approximate dynamic programming and is designed for incrementally learning optimal control strategy that can maximize a reward function [12]. There are two important elements in the definition of RL: a *learning agent;* and *environment*, which interacts with each other in a discretized time space (see Fig.2).

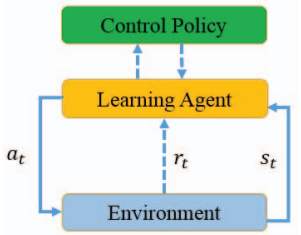

Fig. 2. Reinforcement Learning process

At each time step t (t = 0, 1, 2, 3, 4…), the *learning agent* observes the environment *state* s_t ($s_t \epsilon\, S$),where S is the set of all the possible *environment states*. Then an *action* a_t is selected from the set of available actions upon state s_t where $a_t \epsilon\, A_{(s_t)} \subseteq A$, and A is the set of all possible actions the agent can take. *Action* a_t is then implemented to the environment. The environment then moves to a new state s_{t+1} due to the action and an immediate reward $r_{t+1} \in R$ associated with the transition(s_t, a_t, s_{t+1}) is determined and fed back to the *learning agent*. At each state transition, the *agent* receives an immediate reward, which is then used to update and form a *control policy* that maps the current state to an optimal control action.

In addition, this *control policy* is based on *action-value function*, which is defined as the expected future total reward starting from that state by taking this action. This function is to estimate "*how good*" it is to perform a given action in a given state in terms of the expected return.

Here we define $Q^{\pi}(s,a)$ the value of taking *action a* in *state s* under a *control policy* π(i.e.,a series of optimal action at each time step), which is also the expected return starting from *s*, taking the action *a*, and thereafter following policy π:

$$Q^{\pi}(s,a) = E_{\pi}\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots | s_t = s, a_t = a\}$$
$$= E_{\pi}\{\sum_{k=1}^{\infty} \gamma^k * r(s_{t+k}, a_{t+k}) | s_t = s, a_t = a\} \tag{5}$$

where s_t is the state at time step *t*; γ is a discount factor in (0,1) to guarantee the convergence; $r(s_{t+k}, a_{t+k})$ is the immediate reward upon the state *s* and action *a* at a given time step (*t*+*k*). The goal of RL agent is to identify the best (optimal) *control policy* that maximizes the above action-value function for all the state-action pairs. The *optimal control policy* π^* is the policy that maximize the total experceed reward, can be obtained by the following equations:

$$Q^{\pi^*}(s,a) = \max_{\pi} E\{\sum_{k=1}^{\infty} \gamma^k * r(s_{t+k}, a_{t+k}) | s_t = s, a_t = a\}$$
$$= \mathrm{E}\{\sum_{k=1}^{\infty} \gamma^k * \pi^*(s_{t+k}, a_{t+k}) | s_t = s, a_t = a\}$$
$$\forall s \epsilon\, S, a \epsilon\, A_{(s)} \tag{6}$$

If we assume the state transition probability is known for the environment, π^* can be obtained explicitly by solving a Dynamic Programming problem and Bellman equations [13]. The existence of π^* has been proved mathematically in [13]:

$$Q^{\pi^*}(s,a)=\max_a \sum_{k=1}^{\infty} Pr_{ss'}^{a} * [r_{ss'}^{a} + \gamma * Q^{\pi^*}(s',a')] \quad (7)$$

$$\pi^*(s,a)=arg \max_a \sum_{k=1}^{\infty} Pr_{ss'}^{a} * [r_{ss'}^{a} + \gamma * Q^{\pi^*}(s',a')] \quad (8)$$

where $Pr_{ss'}^{a}$ is the probability of transitioning from s to s' upon action a; $r_{ss'}^{a}$ is the immediate reward corresponding to this state transition. However, it is always difficult to solve the above equations in real time. In the following sections, a Deep Q-learning algorithm is designed to continuously evaluate and improve the control policy.

III. DEEP Q-LEARNING BASED EMS

In this study, a deep reinforcement-learning model is built by combining a neural network and a conventional reinforcement learning to form a *real-time* controller since it makes decision based on only the current system state. It is called deep Q-network or DQN. In conventional RL, the *control policy* is represented by a table but there are only limited number of states and actions that can be taken. However, in the proposed DQN, a neural network is adopted to capture the non-linear relationship between the *environment states* and optimal *actions.* Therefore, the *environment states* and *actions* are continues variable in this model and it is impossible to be represented by a table due to the infinite number of combinations of *states* and *actions*. The *actions*, *states*, and *reward* for this specific problem are defined in the following.

A. States and Immediate Reward

In this study, power demand at wheel (P_{whl}) and the battery pack's state-of-charge (B_{soc}) are selected to form a two-dimensional *state* space for this specific optimization problem. The state space is defined as:

$$\boldsymbol{S}=\{s=[P_{whl}, B_{soc}\ ]^T \mid P_{whl}\epsilon[P_l,P_u], B_{soc}\epsilon\ [B_l,B_u]\} \quad (9)$$

where P_l, P_u are the lower and upper bound of the continuous power-demand state. The value of P_l, P_u are calibrated from the real-world commuting trips data; B_l, B_u are the lower and upper bound of battery state-of-charge (SOC) levels, which are 0.2 and 0.8 respectively in this work.

In real-world RL applications, the appropriate definition of an *immediate reward* is very important. RL *agent* is always trying to maximize the reward it can obtain by taking the best actions at each time step. Therefore, the *immediate reward* should be defined in accordance to the optimization objective. In this work, the reciprocal of the resultant ICE energy consumption at each time step is defined as the *immediate reward*. A penalty value is introduced to penalize the situation where the SOC is beyond the predefined SOC threshold. *Immediate reward is* defined by the following equations*:*

$$r_{ss'}^{a}=\begin{cases} \frac{1}{P_{ICE}} & if\ P_{ICE}\neq 0 \cap\ \ 0.2\le SOC\le 0.8 \\ \frac{1}{P_{ICE}+P} & if\ P_{ICE}\neq 0 \cap SOC\le 0.2\ or\ SOC\ge 0.8 \\ \frac{2}{Min_{P_{ICE}}} & if\ P_{ICE}=0 \cap\ \ 0.2\le SOC\le 0.8 \\ \frac{1}{2*P} & if\ P_{ICE}=0 \cap\ \ SOC\le 0.2 \end{cases} \quad (10)$$

where $r_{ss'}^{a}$ is *immediate reward* generated when state change from s to s' by taking action a; P_{ICE} is the power supply from ICE; P is the numerical penalty and is given as the maximum power supply from ICE; Min_P_{ICE} is the minimum nonzero value of ICE power supply. This definition is used to guarantee the lower ICE power supply (*action*) that satisfies the SOC constraints is favored by giving it larger numerical reward.

B. DQN Structure and Experience Replay

In this study, a network with multi-layer perceptron (see Fig 3) is designed to learn the relationship between environment *states* and optimal *actions*. The network is trained within the iterations of a conventional RL. The inputs of the neural network are the continuous *state* variables, and the output are the continuous *actions* variable. Since the PHEV model used in this study is a discretized model, hence we choose to discretize the *actions* into different levels (here we use 24 ICE power output level). Therefore, the second hidden layer are the Q values of each *action* level and the output layer is the outputted action with largest Q-value. The *Loss* function of network training is defined as:

$$L(\boldsymbol{w})=\mathrm{E}[\underbrace{(r+\gamma \max_{a'} Q(s',a',\boldsymbol{w})}_{Target} - Q(s,a,\boldsymbol{w}))^2] \quad (11)$$

where w are the weights that obtained by training. However, there will be issues if we train the network at each iteration of RL since learning directly from consecutive samples is inefficient, owing to the strong correlations between the samples in a short time period. To avoid such issues, experience replay is adopted in which we store the experience (i.e., a batch of *states*, *action* and reward) at each time-step in a data experience pool. For each certain time number of Q-learning updates, random samples of experience are drawn from the experiment pool and used to train the Q-network. In such way, every experience that learning agent experienced can be used for training multiple times which make the

use of data more efficient. The pseudocode of the proposed deep Q-learning is given in Algorithm 1.

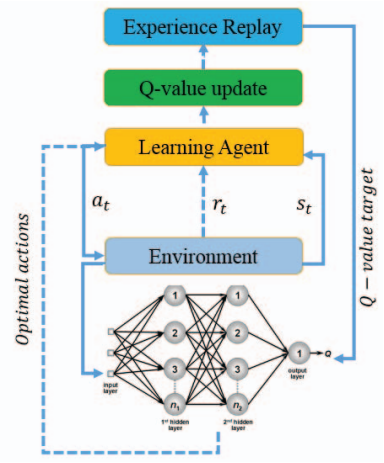

Fig. 3. Deep Reinforcement Learning

Algorithm 1: Deep Q-learning with Experience Replay

Inputs: Training States $s = (SOC, P_d)$; Action Table a; Discount factor γ = 0.9; Exploration probability ε ϵ (0,1); SOC_{INTIAL} = 0.8;

Outputs: Optimal Control Policy π^* ;

1: **for** episode = 1, M **do**
2: Reset environment: $s_0 = (SOC_{INTIAL}, P_0)$
3: **for** t = 1, M **do**
4: With probability ε select a random action a_t
otherwise select $a_t = \max_{a_t} Q^*(s_t, a; \theta)$
5: Choose action a_t and observe reward r_t
6: Set $s_{t+1} = (SOC_t - a_t, P_{t+1})$
7: Store (s_t, a_t, r_t, s_{t+1}) in memory D
8: Sample random minibatch of (s_t, a_t, r_t, s_{t+1}) from D:
9: **if** terminal s_{j+1}: Set $y_j = r_j$
else set $y_j = r_j + \gamma \max_{a'} Q^*(s_{t+1}, a'; \theta)$
10: Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$
11: **end for**
12: **end for**

IV. Case Study

A. Data Collection

For evaluating the proposed system, we collected real-world traffic data to synthesize speed trajectory (second-by-second velocity trajectories) for a commute trip by applying the technique proposed in our previous work [12]. Data are from the inductive loops detector (ILD) data archived in the California Freeway Performance Measurement System (PeMS) [23]. The detailed description of how the trajectory is synthesized can be found in one of our previous work [12]. The commuting trips data are collected from I-210 freeway segment between I-605 and Day Creek Blvd in Southern California. The data collection covers the time period starting at 8:00 a.m. in the morning (westbound) and returning at 4:00 p.m. in the afternoon on January 9th, 2012 (see Fig.4). The collected data only covers the velocity trajectory on the freeway portion due to the lack of traffic information along arterials. However, the proposed EMS can be well applied to arterial driving once the trip data are available. Moreover, the route gradient information is also synchronized with the trip data to extract the second-by-second power demand. Fig 5 and 6 provide the velocity and the according power-demand of the trip respectively.

Fig. 4. Example trip with O-D location from Google Map

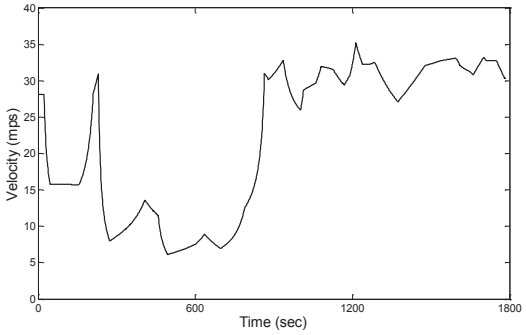

Fig. 5. Synthesized velocity profile of the example trip.

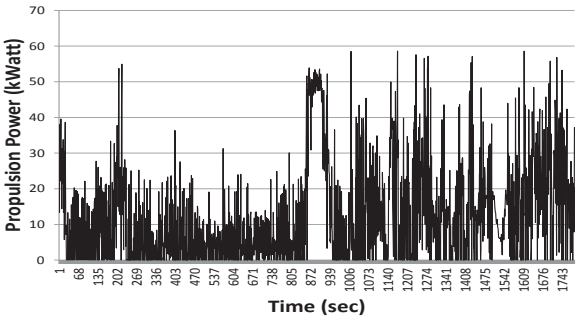

Fig. 6. Power demand along the example trip.

B. DQN training and Convergence

To evaluate the performance of the above-mentioned DQN network, the collected real-world commute data is used for training and testing. The parameters are fixed as following: discount factor γ=0.9, initial =0.5, final =0.1, replay size D = 50. In these experiments, we trained the model utilizing stochastic optimization with the learning rate 0.001 and mini-batches of size 32. The behavior

policy during training was ε-greedy with annealed linearly from 1 to 0.1 and fixed at 0.1 thereafter. Two GeForce GTX 1080 GPUs are used to assist accelerating training phase.

The network is trained with 300 episodes and each episode means a trip (1780 sec data) that is described in previous section. The track of *Loss* (see eq. 11) is recorded at two different learning rate (0.0001 ad 0.000025) in Fig.7. To avoid the overlap two curves, the y-axis is designed in symmetric to the zero at the center. It is quite clear that *Loss* is decreasing along the training process and the convergence with higher learning rate is much faster than that of lower learning rate. In addition, when looking at the zoomed-in figure of a local region, there are periodical jumps of *Loss* which is due to that fact that the at the end of each training episode, the SOC state and the power-demand state are changed abruptly and the environment states are initialized. However, the overall trend of *Loss* is decreasing which indicates the improvement of Q-network. Fig. 8 presents the track of average reward along the training process. There are periodical reward drop the early stage of training process. This is because the adding of big penalty when the agent selects an action that results in the violation of SOC constraint (see eq. (10)). These drops disappeared after a certain number of training iterations (e.g., $1.5x10^5$) since the agent learned from these penalties and avoids such actions in the later stage of training. This is also a convincing evidence of performance improvement from learning from the historical driving records.

To track the performance improvement, for every ten episodes, the model obtained by that time step is evaluated with one stranded trip for 10 times, and the average fuel consumption is recorded as shown in Fig. 9. As we can see in the figure, there is a clear decreasing trend in average fuel consumption along the training process and there is no significant improvement after 260 episodes, which indicates the convergence of performance within 300 episodes. What is noteworthy is that at the beginning of the training process, the fuel efficiency is extremely low, which is due to the almost random selection of engine power supplies. Fig. 10 provides more details on the distribution of the fuel consumption during a sub-period of the training process that is marked with a dashed red box in Fig. 9. The proposed system is able to achieve lower fuel consumption than binary control strategy (0.404 gallon marked by the dashed line in Fig. 10) only after 260 episodes. The lowest fuel consumption on the standard trip is 0.338 gallon.

C. Performance Comparison

To further validate the performance of the proposed DQN based EMS, the fuel efficiency is compared with that of a conventional reinforcement learning based EMS as well as a binary control strategy as baseline. The track of SOC for each of that strategy on the same standard trip is

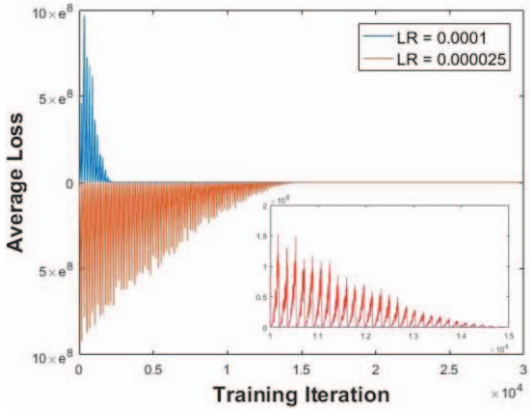

Fig. 7. Track of *Loss*

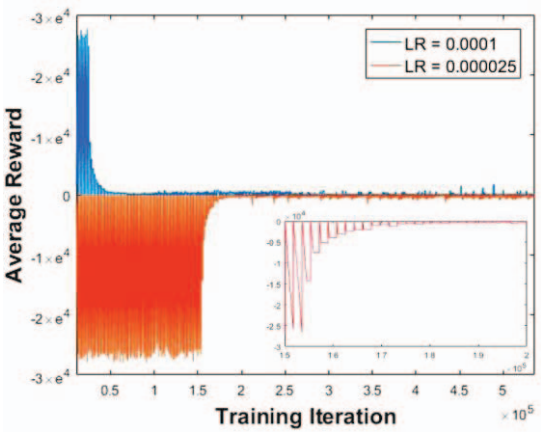

Fig. 8. Track of reward

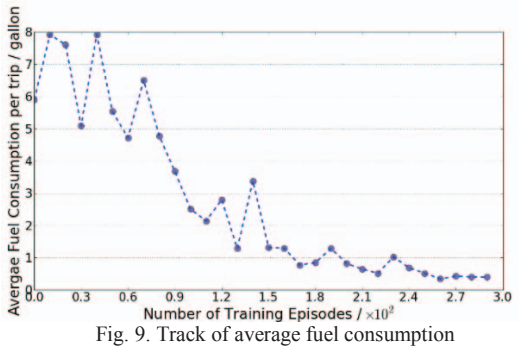

Fig. 9. Track of average fuel consumption

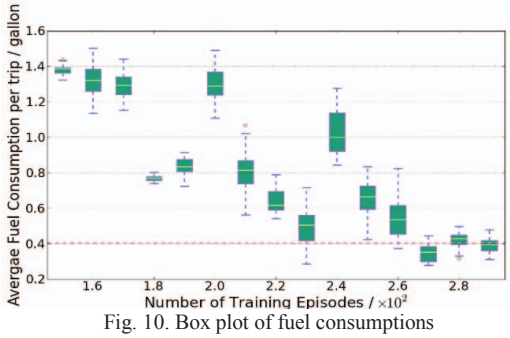

Fig. 10. Box plot of fuel consumptions

given in Fig. 11. It can be seen that deep Q-learning based EMS achieves the lowest fuel consumption and it could be explained by comparing the resulted SOC track to the power-demand depicted in Fig.6. As shown in Fig. 6, there are two power-demand peaks at around time step 200 and 800. The DQN based model consumes more battery energy at these time steps with significant SOC drops to avoid unnecessary engine energy consumption. This result also shows that a smoothed use of battery energy along the trip is not necessarily the optimal solution. In other words, an optimal EMS controller should be always adaptive to the changing driving conditions in order to achieve maximized fuel efficiency.

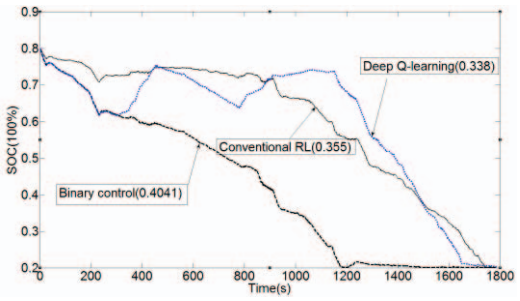

Fig.11. SOC track and fuel consumption of different EMS models.

V. CONCLUSIONS

In this paper, a deep reinforcement learning based real-time energy management system is designed and tested with data from a real-world commute trip in Southern California. The proposed model combines a Q-learning and a deep neural network to form a deep Q-network which is capable of learning and providing the optimal control decisions in continuous environment and actions states. The evaluation with real-world trip data shows that an average 16.3% fuel savings can be achieved comparing to conventional binary control strategy. The future work would be focused on the further testing on vehicle platforms with more real-world driving data.

REFERENCES

[1] Bureau of Transportation Statistics (BTS). Available at: http://www.bts.gov/publications/national_transportation_statistic.

[2] U.S. Environmental Protection Agency (EPA). "DRAFT Inventory of U.S. Greenhouse Gas Emissions and Sinks: 1990 – 2013". Final report, Feb, 2015.

[3] M. Faraj and O. Basir, "Range anxiety reduction in battery-powered vehicles," 2016 IEEE Transportation Electrification Conference and Expo (ITEC), Dearborn, MI, 2016, pp. 1-6

[4] Denis, N.; Dubois, M.R.; Desrochers, A., Fuzzy-based blended control for the energy management of a parallel plug-in hybrid electric vehicle, Intelligent Transport Systems, IET , vol.9, no.1, pp.30,37, 2 2015

[5] Wang X., He, H. Sun, F., Sun, X., Tang,H., Comparative Study on Different Energy Management Strategies for Plug-In Hybrid Electric Vehicles, Energies 2013, 6, 5656-5675

[6] Tribioli, L.; Onori, S., Analysis of energy management strategies in plug-in hybrid electric vehicles: Application to the GM Chevrolet Volt, American Control Conference (ACC), 2013 , vol., no., pp.5966,5971, 17-19 June 2013

[7] Qiuming Gong; Yaoyu Li; Zhong-Ren Peng, Trip based optimal power management of plug-in hybrid electric vehicles using gas-kinetic traffic flow model, American Control Conference, 2008 , vol., no., pp.3225,3230, 11-13 June 2008

[8] Larsson, V.; Johannesson Mårdh, L.; Egardt, B.; Karlsson, S., Commuter Route Optimized Energy Management of Hybrid Electric Vehicles, Intelligent Transportation Systems, IEEE Transactions on , vol.15, no.3, pp.1145,1154, June 2014

[9] Zheng Chen, Chris Chunting Mi, Rui Xiong, Jun Xu, Chenwen You, Energy management of a power-split plug-in hybrid electric vehicle based on genetic algorithm and quadratic programming, Journal of Power Sources, Volume 248, 15 February 2014, Pages 416-426.

[10] X. Qi; G. Wu; K. Boriboonsomsin; M. J. Barth, "Development and Evaluation of an Evolutionary Algorithm-Based Online Energy Management System for Plug-In Hybrid Electric Vehicles," in IEEE Transactions on Intelligent Transportation Systems , vol.PP, no.99, pp.1-11

[11] Xuewei Qi, Guoyuan Wu, Boriboonsomsin Kanok., Barth Matthew 1., Jeffery Gonder. Data-Driven Reinforcement Learning-Based Real-Time Energy Management System for Plug-in Hybrid Electric Vehicles. Transportation Research Record 2016; 2572:1-8.

[12] G. Wu, K. Boriboonsomsin, M. Barth. "Development and Evaluation of an Intelligent Energy-Management Strategy for Plug-in Hybrid Electric Vehicles". IEEE Transactions on Intelligent Transportation Systems, Vol.15, No.3, June 2014, pp. 1091 – 1100

[13] R. S. Sutton and A. G. Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge MA, 1998.