2020 IEEE Intelligent Vehicles Symposium (IV)
October 20-23, 2020. Las Vegas, USA

Uncertainty-aware Energy Management of Extended Range Electric Delivery Vehicles with Bayesian Ensemble

Pengyue Wang, Yan Li, Shashi Shekhar, *Fellow*, *IEEE,* William F. Northrop*

***Abstract*—In recent years, deep reinforcement learning (DRL) algorithms have been widely studied and utilized in the area of Intelligent Transportation Systems (ITS). DRL agents are mostly trained with transition pairs and interaction trajectories generated from simulation, and they can achieve satisfying or near optimal performances under familiar input states. However, for relative rare visited or even unvisited regions in the state space, there is no guarantee that the agent could perform well. Unfortunately, novel conditions are inevitable in real-world problems and there is always a gap between the real data and simulated data. Therefore, to implement DRL algorithms in real-world transportation systems, we should not only train the agent learn a policy that maps states to actions, but also the model uncertainty associated with each action. In this study, we adapt the method of Bayesian ensemble to train a group of agents with imposed diversity for an energy management system of a delivery vehicle. The agents in the ensemble agree well on familiar states but show diverse results on unfamiliar or novel states. This uncertainty estimation facilitates the implementation of interpretable postprocessing modules which can ensure robust and safe operations under high uncertainty conditions.**

I. Introduction

In recent years, data-driven deep reinforcement learning (DRL) algorithms are more and more popular in the area of Intelligent Transportation Systems (ITS), they are utilized in applications like autonomous driving [1][2], traffic light cycle control [3][4] or energy management strategies (EMS) of hybrid vehicles [5][6]. The agents are mostly trained on transition pairs and interaction trajectories generated from simulation environments, since it is time-consuming, expensive and even dangerous to train the agents on real transportation systems. The well-trained agents can achieve satisfying or near optimal performances when the input states are very familiar, i.e., the frequently visited region in the state space during training. However, there is no guarantee for the relative less frequent or even unvisited region in the state space. Agents could choose arbitrarily bad actions under these conditions. Unfortunately, it is inevitable to encounter unfamiliar or novel conditions/states in a real-world transportation system, since it is impossible to build a simulator that can cover all possible regions in the underlying state space. Further, there are always gaps between the trajectories generated in the simulator and the real systems. Consequently, to implement DRL algorithms in real-world transportation systems, it is crucial to build uncertainty-aware agents, which not only have the ability to solve sequential decision-making problems, but also have some measure of model uncertainty associated with each action during the task.

A simple but inspiring regression problem is shown in Fig. 1. There is a nonlinear mapping from x to y and the eight red dots represent the available data points. An ensemble of neural networks (NNs) with imposed diversity are trained to capture the nonlinear mapping by learning from the available data [7]. It can be observed that, in the region where exists data, different NNs agree well while in the region there is no data, NNs diverge which indicates high model uncertainty.

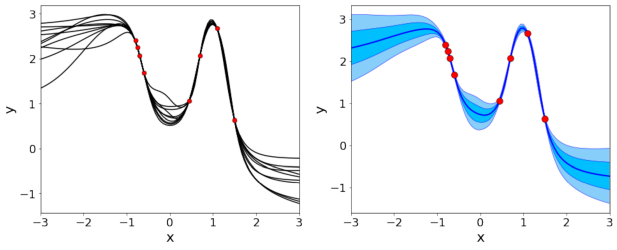

Fig. 1. Ten NNs are trained. The prediction of each individual NN is shown on the left figure. The mean prediction is shown on the right figure with one and two standard deviation represented by the shaded region.

This work adapts the Bayesian ensemble method [7] to train a group of DRL agents for the energy management system (EMS) of an extended range electric vehicle (EREV) for package delivery with two-way Vehicle-to-Cloud (V2C) connectivity. Uncertainty estimation is performed for each action during the task and it facilitates the implementation of interpretable postprocessing modules for unfamiliar or novel conditions in real-world tasks, which provides a more robust and safer system. The general framework is shown in Fig. 2.

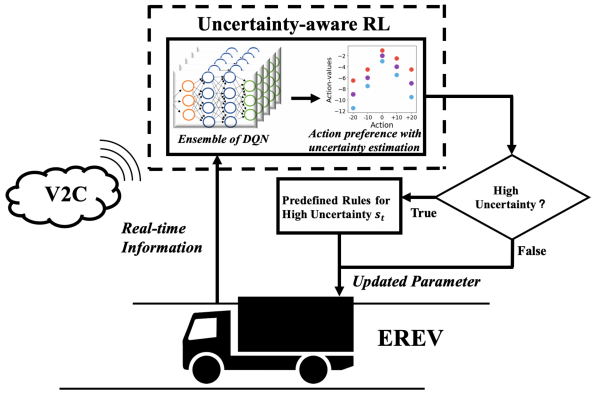

Fig. 2. Illustration of the proposed framework.

*Corresponding author
The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E) U.S. Department of Energy, under Award Number DE-AR0000795.

P. Wang, Y. Li, S. Shekhar and W. F. Northrop are with University of Minnesota, Minneapolis, MN 55455. (email: wang6609@umn.edu; lixx4266@umn.edu; shekhar@umn.edu; wnorthro@umn.edu)

II. Related Work

There are two kinds of uncertainties that can be modeled in RL problems: risk and model uncertainty [8][9]. Risk is also called Aleatoric uncertainty [8], which represents the intrinsic stochastic property of the studied system or environment. It cannot be eliminated no matter how much data we collect or how long we run the simulation. This can be handled with distributional RL algorithms, which model the full return distribution instead of the expected value. We recommend the reader to refer [10][11] for the algorithms and [2][12][13] for transportation-related applications. Although it is an important topic, it should be distinguished from model uncertainty clearly, which is the focus of this work.

Model uncertainty is also called epistemic uncertainty, and it can be explained away with enough data. If trajectories generated during training process can populate the whole state space that is possible to be encountered in the real-world condition, there is no need to consider the model uncertainty. However, that is usually difficult to achieve as the amount of data needs to fill a space grows exponentially with the dimensionality of the input space [14], and as the transition probability of the real-world system cannot be fully captured by the data generated from simulation.

Model uncertainty can be handled elegantly with the Bayesian framework, if applicable, and Bayesian NNs provide a systematic way to model uncertainties in NNs [9][14]. It keeps posterior distributions for the parameters calculated from the predefined prior distributions and training data. However, there are mainly two challenges that prevent it from being widely used in real-world problems [7][9][15]. First, exact Bayesian inference is only feasible on small-scale problems, and its approximations are also computationally expensive. Second, to implement Bayesian NNs, significant modifications in model building and training procedure are needed compared with standard NNs, which are very easy to use with popular deep learning libraries like Pytorch and Tensorflow. Therefore, researchers have investigated non-Bayesian methods to model the uncertainty, aiming at developing practical and scalable frameworks.

A simple and scalable method to estimate the predictive uncertainty is developed in [15]. An ensemble of NNs is trained, and for each individual NN, the conditional Gaussian distribution given the input is modeled. The predictive uncertainty is obtained by treating the output as a mixture of Gaussian distributions. In [16], ensemble sampling is developed to approximate the posterior distribution. First, M NNs are drawn from a prior distribution. Then, at each timestep during training, only one model in the ensemble will be sampled to generate data. After the data is generated, all the models in the ensemble will be updated. To bridge the gap between Bayesian NNs and the practical ensemble methods, Bayesian ensemble [7] is developed, which regularizes the parameters in the standard NNs to values drawn from a predefined distribution as the prior information for the NNs, which can produce Bayesian behaviors while keeping the algorithm practical.

Energy management problems of hybrid vehicles have been a research focus for a long time. A detailed survey can be found in [18], and for RL-based EMS, a comprehensive discussion can be found in [19]. However, there is little work about the model uncertainty in EMS problems. One of the main reasons is, for most hybrid vehicles, the worst condition is unsatisfying energy efficiencies, which does not lead to serious problems. However, for our EREV for delivery, bad EMS can lead to the condition of running out of battery during the delivery trip, which may cause delays of delivery tasks and can be very dangerous if it happens during highway driving. Consequently, our goal is to build an uncertainty-aware agent that can facilitate a postprocessing module to execute predefined rules when the uncertainty exceeds a predefined threshold. The Bayesian ensemble method is used due to its simplicity in implementation and better theoretical support compared with other ensemble-based nonparametric methods.

III. Reinforcement Learning Problem Formulation

A. Vehicle Configuration and EMS Problem Introduction

The powertrain configuration of the studied EREV is shown in Fig. 3. The vehicle is primarily running as an EV using the energy from the high-capacity battery. The engine serves as a range-extender which is used to charge the battery through a generator according to the EMS. There is no mechanical connection between the engine output shaft and the wheels so that the engine is completely decoupled from the vehicle movement. Ideally, for short trips that does not exceed the all-electric range (AER) of the vehicle, the engine should not be used during the trip. For longer trips that the battery is not enough, the goal is to gradually deplete the battery energy during the trip and reach a target end battery state of charge (SOC) when the trip is finished with the help of the engine. This guarantees the vehicle always have enough electric energy and minimizes the use of fuel, which is widely used in the area of EMS [17][18][22].

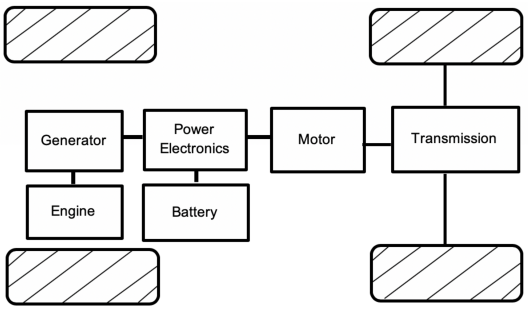

Fig. 3. Powertrain configuration of the studied EREV.

The EMS is based on the measured real-time SOC and a calculated reference value. If the real-time SOC is lower than the reference value, the engine will be turned on to charge the battery at a predefined high-efficiency speed and load condition. The reference value is calculated as:

$$SOC_{ref} = 100\% \times \left(1 - 0.9\frac{d_t}{L_{set}}\right), \quad (1)$$

where d_t is the distance the EREV has traveled on a given trip, and L_{set} is the energy-compensated expected trip

distance. The value of 0.9 comes from the setting that the target end SOC is 10% in this work. In addition, if the calculated SOC_{ref} is higher than 60%, it is set to be 60% to prevent fuel use in short trips.

SOC_{ref} represents how much energy is expected to be left when the vehicle has traveled for d_t given L_{set}. So, to achieve high fuel efficiencies, the value of L_{set} should be optimized so that the EREV can approximate the ideal performance described above.

Due to factors like delivery demand, weather and traffic conditions, even for a vehicle that running in a certain area, the distance and energy intensity vary day-to-day. Consequently, the value of L_{set} should be updated during the delivery trip according to the real-time information. This is a sequential decision-making problem without an available model of the system, i.e., the detailed trip information cannot be used before it is observed. This fits into the framework of RL well [20]. The formulation of the RL problem will be first introduced. Then, in the next section, the uncertainty-aware RL algorithm that used to solve it is shown.

B. Formulating the EMS Problem as a RL Problem

In a RL problem setting, an agent interacts with an environment through state s, action a and reward r [20]. At the beginning of the task, the agent observes the initial state s_0 provided by the environment and takes action a_0 according to its policy $\pi: s \rightarrow a$. Then, the environment outputs the reward r_0 and next state s_1 according to the transition probability $p(s_{t+1}|s_t, a_t)$ and reward function $r(r_t|s_{t+1}, s_t, a_t)$. A series of interactions will give rise to a sequence like:

$$s_0, a_0, r_0, s_1, a_1, r_1, s_2, a_2, r_2 \dots s_T, \tag{2}$$

where s_t represents the terminal states.

The goal of the agent is to maximize its discounted cumulated reward at each timestep which is defined as:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-1-t} r_{T-1}. \tag{3}$$

γ is the discount factor which uses the value of 0.99. The EMS problem is formulated as a RL problem as follows.

1) State space

The state space is a 7-dimensional space consisting of the real-time information that is available during the delivery trip, including travelled time, distance, battery SOC, fuel use, GPS coordinates and current L_{set} setting. It can be represented as a vector at each timestep t:

$$s_t = [t_{travel}, d, SOC, f, x, y, L_{set}]. \tag{4}$$

2) Action space

The action space is a set of predefined actions where the magnitude represents the change of L_{set} at timestep t:

$$a_t \in [-20, -10, 0, +10, +20]. \tag{5}$$

3) Transition probability

The transition probability $p(s_{t+1}|s_t, a_t)$ of the underlying system that generates the real-world data is unknown, and only can be approximated by existing data with a vehicle model that calculates the SOC curve and fuel use with different L_{set} settings. Consequently, the explore of the state space is constrained within the recorded data, especially for the dimension of distance d and GPS coordinates x and y. In this work, 22 recorded delivery trips were used for training and 30 trips were used for testing. The distance and energy intensity distributions are shown in Fig. 4. It can be observed that, as data cumulates, the range of the possible trip distance gets larger as well as the overall energy intensity.

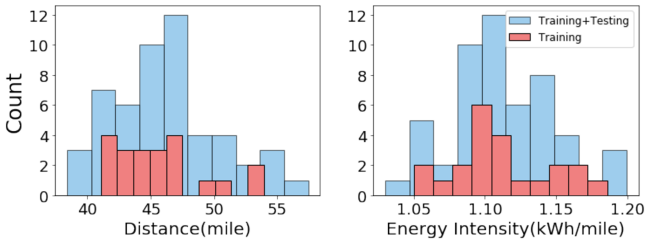

Fig. 4. Distance and energy intensity distribution of training trips and training-testing trips combined.

Furthermore, even for two delivery trips with similar distance, GPS trajectories of a future trip can be significantly different, which will also lead to unfamiliar states. For example, the GPS trajectories of two delivery trips are shown in Fig. 5. It can be observed that, although the delivery region is the same, there are differences in the area covered due to factors like delivery demand.

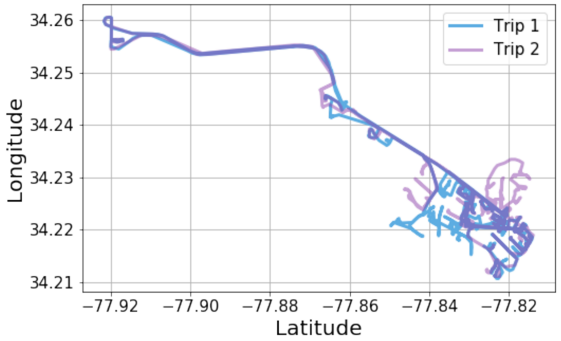

Fig. 5. Comparison of two GPS trajectories with 43.1 and 44.3 miles.

Besides distance and GPS coordinates that can lead to novel and unseen states, there is another reason corresponding to the dimensionality of the state space, which is the "curse of dimensionality" [14]. The amount of data that needs to fill the state space grows exponentially with the dimensionality of the state space. Consequently, for our defined 7-dimensional state space, even for some of the visited states, data and interaction trajectories can be sparse in that region due to relative less frequent visitation during the training process, which is highly dependent on the statistics of the recorded trips. Consequently, uncertainty estimation is of great importance when dealing with real-world problems as it is not feasible to build a model that is well trained on all parts of the possible state space.

To make the above discussion more straightforward, a simplified state space with two dimensions are shown in Fig. 6. For each episode of interaction, a trajectory from the initial state to the region of possible terminal states will be made. After recording some real-world trajectories, simulations can

be performed to train the agent based on the recorded trajectory. During the training process, the region that is reachable from the recorded trajectories by choosing different actions can be visited, and the frequency of visitation depends on the statistics of the recorded trajectories. Region A represents a region which is visited frequently, region B represents a region with sparse data and region C represents an unvisited part of the state space, which is not reachable from changing actions from any of the recorded trajectories. The goal of the uncertainty-aware RL algorithm is to measure whether the encountered state is novel during testing.

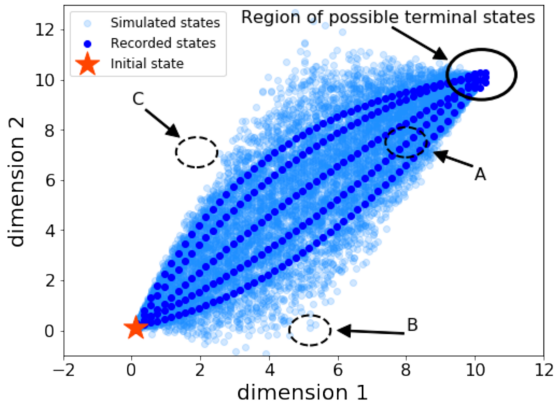

Fig. 6. Illustration of frequently visited states, infrequently visited states and unvisited states in a two-dimensional state space.

4) Reward function

The reward function is defined as:

$$r_t = r_f t_{f,t} + r_{SOC} t_{SOC,t} + r_{a,t} + r_c. \quad (6)$$

The first term penalizes fuel use according to the engine running time $t_{f,t}$ with a coefficient r_f set to -0.001. The second term penalizes the condition of SOC lower than 10% according to the time under that condition $t_{SOC,t}$, with a coefficient r_{SOC} set to be -0.060. The third term is used to encourage the agent learn a more efficient policy. It equals to 0 if the agent chooses the action of not changing L_{set} and equals to -0.020 otherwise. The last term is used to compensate for the negative reward caused by the necessary fuel use. It is only given at the end of the trip, as only with full information of the trip, the minimal amount of fuel to keep the SOC always high than 10% can be calculated.

IV. Uncertainty-Aware RL Algorithm

The uncertainty-aware RL algorithm is based on an ensemble of value-based DRL agents. Deep Q-network (DQN) is used as the base learners [21]. The core concept behind it is the action-value which is also called Q-value:

$$Q_\pi(s,a) = E_\pi[G_t | s_t = s, a_t = a], \quad (7)$$

It represents the expected long-term return that can be achieved if action a is taken at state s and then following policy π thereafter. The optimal action-value is defined as:

$$Q^*(s,a) = max_\pi Q_\pi(s,a), \quad (8)$$

which represents the highest action-value that can be obtained for all possible state-action pairs under all possible policies. Therefore, given the optimal action-value, the optimal control policy can be derived by acting greedily with respect to it:

$$\pi^*(s) = argmax_a Q^*(s,a). \quad (9)$$

The core idea of DQN and most value-based DRL methods is to use NNs to approximate the optimal action-value function. The parametrized action-value function $Q(s,a;\theta)$ is also called Q-network. θ is a vector that contains all the parameters in the NN. For a given problem with stationary underlying transition probabilities, the $Q^*(s,a)$ satisfies the following Bellman optimality equation:

$$Q^*(s,a) = E_{s'}[r + \gamma max_{a'} Q^*(s',a') | s,a]. \quad (10)$$

The function approximator $Q(s,a;\theta)$ is trained according to this relation. As the probability distribution $p(s_{t+1}|s_t,a_t)$ is unknown, the right-hand side of (10) is approximated by a sample generated by a transition pair (s_t, a_t, r_t, s_{t+1}):

$$Q_{target} = r_t + \gamma max_{a'} Q(s_{t+1}, a'; \theta). \quad (11)$$

The parameters θ are trained at each timestep by minimizing the error:

$$L = \left(Q_{target} - Q(s_t, a_t; \theta)\right)^2. \quad (12)$$

The visited state s_t is determined by the interaction between the agent and the simulated environment. For the frequently visited region in the state space, the action-values for the corresponding state-action pairs are well estimated. However, for the less visited region where the data is sparse and unvisited region, θ are not well determined so that the output action-value is not reliable. Consequently, having a measure of uncertainty can detect whether the output of Q-network is trustworthy.

The intuition behind ensemble-based uncertainty estimation methods is for different function approximators, the value will agree well on frequently visited states and diverge on the rest of the regions, and the variance of the output values indicates the uncertainty. The diversity of different Q-networks mainly comes from the following aspects. First, each of them is randomly initialized with different parameters so that the target network that each Q-network is trained on is different. Second, stochastic gradient descent-based optimization provides another source of randomness. Third, the exploration process like the $\varepsilon-$greedy chooses actions randomly with some probability controlled by the hyperparameter ε. In addition, for the Bayesian ensemble method [7], the parameters in each Q-network is regularized to some anchored values sampled from a predefined distribution. Therefore, the loss function (12) is modified as:

$$L_{anchor} = \left(Q_{target} - Q(s_t, a_t; \theta)\right)^2 + \lambda \|\theta - \theta_0\|^2, \quad (13)$$

where λ represents the regularization strength and θ_0 is the anchored value randomly sampled from a distribution which represents the prior information of the Q-network.

Assume M Q-networks $Q(s,a;\theta_j)$ are trained individually with the loss function in (13) with corresponding anchored values $\theta_{0,j} \sim Normal(\mu_{prior}, \Sigma_{prior})$, the M outputs of the ensemble given an input state-action pair can be considered as M samples from the posterior distribution [7].

Therefore, the variance of the ensemble's predictions can be interpreted as its uncertainty.

Although this method does not perform exact or approximate Bayesian inference over NN parameters [14], it keeps an explicit notion of prior which does not vanish during training, resulting in Bayesian behavior. Also, it is built on the fact that adding a regularization term to a loss function returns maximum a posterior (MAP) estimates of parameters [23]. Therefore, the Bayesian ensemble method gets better theoretical support compared with other ensemble-based methods without compromising the advantage of being easy to implement. The training procedure is briefly summarized in Algorithm I.

Algorithm 1 Bayesian Ensemble

For $Q(s,a;\theta_j), j = 1$ to M **do**
. **Initialize** θ_j randomly
. **Sample** $\theta_{0,j} \sim Normal(\mu_{prior}, \Sigma_{prior})$
. **Initialize** the target network as $\theta_j^- = \theta_j$
. **Initialize** the replay buffer B_j to capacity 5×10^4 with no transitions
. **For** epoch = 1 to N **do**
. . **For** trip = 1 to K **do**
. . . Get initial state s_0 from current trip data
. . . $t = 0$
. . . **While** $t < T$, **do**
. . . . With probability ε, randomly select an action a_t
. . . . Otherwise select $a_t = argmax_a Q(s_t, a; \theta_j)$
. . . . Update current L_{set} according to a_t
. . . . Run the vehicle model for 2000s with updated L_{set}
. . . . Return state s_{t+1} and r_t
. . . . Store the transition (s_t, a_t, r_t, s_{t+1}) in the replay buffer B_j
. . . . Sample random n transitions (s_j, a_j, r_j, s_{j+1}) from B_j
. . . . Perform a gradient descent step on loss shown in (12)
. . . . Perform soft update of the target network θ^-
. . . . $s_t = s_{t+1}$
. . **End For**
. **End For**
. **Save** $Q(s,a;\theta_j)$
End For

TABLE I
NEURAL NETWORK STRUCTURES AND HYPERPARAMETERS

Pytorch Implementation of Q-network		
Hidden layer 1	Number of units	64
	Activation function	ReLU
Hidden layer 2	Number of units	32
	Activation function	ReLU
Output layer	Number of units	5
	Activation function	Linear
Hyperparameters	Optimizer	Adam
	Initial learning rate	5×10^{-4}
	Batch size	48
	Number of epochs	1000
	Regularization coefficient λ	1×10^{-3}

V. SIMULATION RESULTS

Feedforward NNs [14] were used to build the Q-networks and the implementation details are summarized in Table I. An ensemble of five Q-networks was trained. For each Q-network $Q(s,a;\theta_j)$, both of the model parameters θ_j and the anchored values $\theta_{0,j}$ were sampled from the distribution of $Normal(\mu_{prior}, \Sigma_{prior})$. All the components in μ_{prior} are 0 and Σ_{prior} is a diagonal matrix with equal variance. The variance is set to be 1 and it controls the degree of diversity among the models in the ensemble. In other words, the initial value of each parameter in the Q-networks is sampled from a one-dimensional normal distribution with mean of 0 and variance of 1. It should be noted that although the initial values of θ_j and $\theta_{0,j}$ comes from the same distribution, they were not the same values as recommended in the original paper [7].

After training, for each s_t encountered during testing, the mean and standard deviation (SD) for the action-values for the state-action pairs are calculated:

$$Q_{mean}(s_t, a) = \frac{1}{M}\sum_{j=1}^{M} Q(s_t, a; \theta_j),$$

$$Q_{SD}(s_t, a) = \sqrt{\frac{\sum_{j=1}^{M}\left[Q(s_t, a; \theta_j) - Q_{mean}(s_t, a)\right]^2}{M-1}}. \quad (14)$$

Then the corresponding policy is:

$$\pi(s_t) = argmax_a Q_{mean}(s_t, a). \quad (15)$$

The mean is used to select actions and the values of SD indicate the uncertainty corresponding to each action. Next, the detailed performance of the Bayesian ensemble on three delivery trips are shown, demonstrating the three kinds of conditions discussed in Fig. 6 respectively.

The frequently visited states where the model parameters are well-determined and the infrequently visited states where the model parameters are less well-determined are shown on two training trips with 43.1 and 52.9 miles from Fig. 7-Fig. 10. For the shorter trip, it can be observed from Fig.7 that for all states encountered during the trip, the SD of action-values for all the 5 possible actions are lower than 1, which indicates low model uncertainties. Taking state *A* annotated in Fig.7 as an example, the mean and one standard deviation of action-values are plotted on the left part of Fig. 8 and the individual predictions from each agent are plotted on the right. Although there are imposed diversities among the agents, it is obvious that all the agents agree well on the action-values for each possible action. All the states encountered in this trip can be considered as locating in the frequently visited region in the state space. This can be partly explained by looking at Fig. 4. The distance of most of the 22 training trips are concentrated near 43 miles, and energy use is highly correlated with distance so that states with similar distance, SOC, fuel use as well as the L_{set} setting will be generated frequently when the simulation is run on any of these trips with similar distance, which leads to a frequently visited region near trajectories generated from these training trips in the state space, like region A in Fig. 6.

The simulation results for the longer trip which is also the second longest trip in the training trips are shown in Fig 9. It requires the highest fuel use among the 22 training trips, the states encountered at the last part of this trip can only be generated when the simulation is run on the longest two trips, which leads to relative less frequent visitations, corresponding to region B in Fig. 6. Consequently, the uncertainties for part

of the action-values are higher than 1, indicating the parameters in the agents are not well-determined for the corresponding state-action pairs.

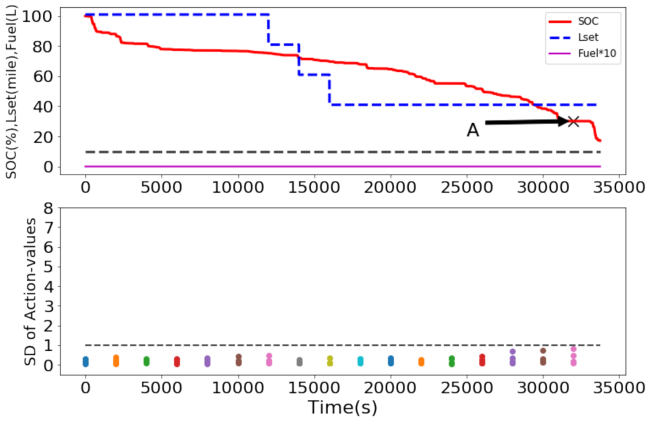

Fig. 7. Performance of the trained algorithm on a training trip with a distance of 43.1 miles (upper) and the estimated uncertainties along the trip (lower).

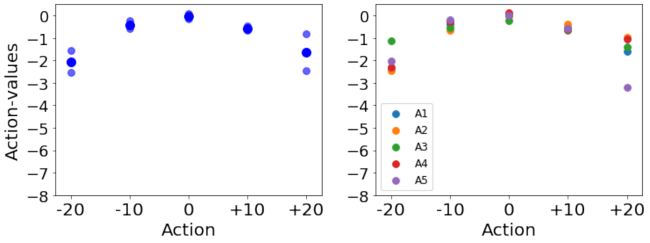

Fig. 8. The mean action-values for each action with one standard deviation (left) and the predictions from each agent (right) for state A.

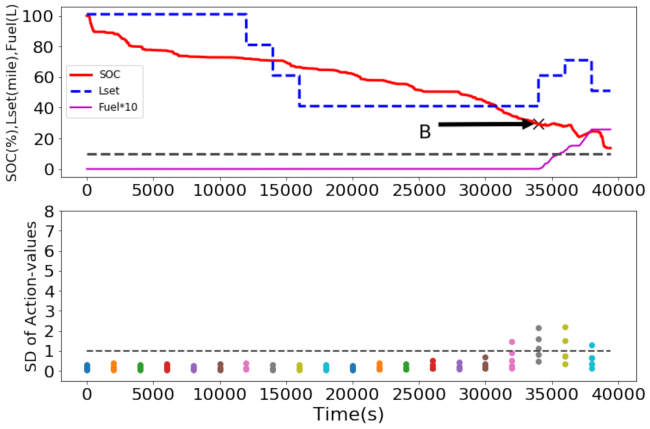

Fig. 9. Performance of the trained algorithm on a training trip with a distance of 52.9 miles (upper) and the estimated uncertainties along the trip (lower).

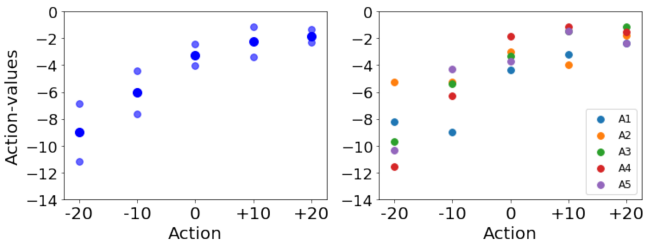

Fig. 10. The mean action-values for each action with one standard deviation (left) and the predictions from each agent (right) for state B.

For a test trip with a distance longer than any of the training trips, it is obvious from Fig. 11 that the algorithm does not perform well as the battery SOC drops below 10% for a significant amount of time. Nevertheless, the model outputs very high uncertainties for all the actions from 34000s, indicating the states are novel and significantly different from the states that it experienced during the training process. It can be observed from Fig. 12 that although the ranking of the action-values for state C is correct, the agents cannot agree on the long-term return after this state, i.e., the agents are not sure whether the EMS can prevent the EREV from running out of battery. The states that encountered at the last part of this trip can be considered as locating in the region that is never visited during training, just like region C in Fig. 6. Therefore, the predictions from different agents are significant different, just like the NNs diverge on the region there is no data in Fig. 1.

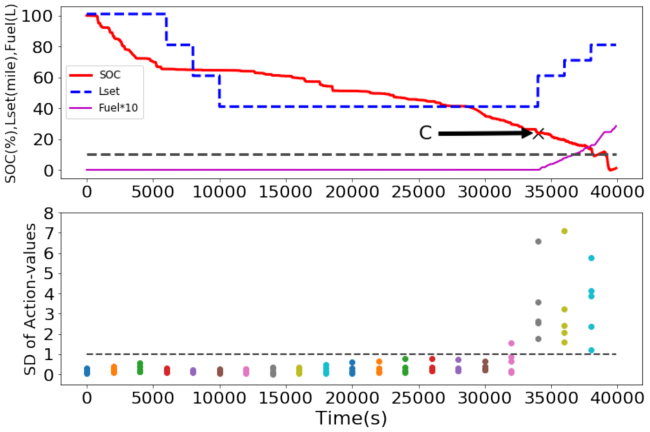

Fig. 11. Performance of the trained algorithm on a test trip with a distance of 57.4 miles (upper) and the estimated uncertainties along the trip (lower).

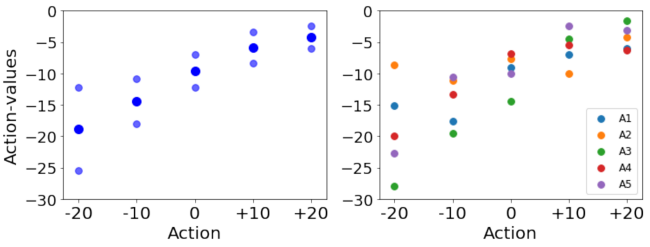

Fig. 12. The mean action-values for each action with one standard deviation (left) and the predictions from each agent (right) for state C.

With the uncertainty-aware RL algorithm, an interpretable white-box postprocessing module can be integrated into the black-box RL system to include predefined explicit rules for unfamiliar or novel states. The threshold to define unfamiliar or novel states can also be controlled. If the threshold is set to be 0, then the RL-based EMS becomes the widely used interpretable rule-based EMS. For example, for our EREV application, if the model uncertainty is higher than the predefined threshold value, the postprocessing module can overwrite the L_{set} with a high value, or it can notify the driver to stop the EREV until the SOC is charged back to some value. The focus of this work is to develop the uncertainty-

aware RL algorithm that can identify the novel states so that the design of postprocessing module is not discussed here.

VI. Conclusion

In this work, we demonstrated the importance of uncertainty estimation for DRL algorithms in real-world applications by applying it on an EMS problem. It is inevitable to encounter novel states as it is impossible to build a simulator that can generate all possible states during training. Also, even for part of the visited states, due to the statistics of the recorded trajectories and the "curse of dimensionality", the parameters in the agent might not be well determined due to the sparse data. The method of Bayesian ensemble is used to perform the uncertainty estimation, which is practical to implement and keeps an explicit notion of prior information compared with other ensemble-based methods. Simulation results show that for the frequently visited states, uncertainties are low for each action, while for states that are relatively less frequently visited and novel states, the uncertainties are high, indicating the model is not well-trained for these states. The uncertainty estimation is highly beneficial for applications in ITS, especially for the safety-critical problems. Interpretable postprocessing modules can be built based on the estimated uncertainty, providing a more robust and safer system.

Acknowledgment

The information, data, or work presented herein was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E) U.S. Department of Energy, under Award Number DE-AR0000795. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

[1] Lötjens, Björn, Michael Everett, and Jonathan P. How. "Safe reinforcement learning with model uncertainty estimates." In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8662-8668.

[2] K. Min, H. Kim and K. Huh, "Deep Distributional Reinforcement Learning Based High-Level Driving Policy Determination," in *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 416-424, Sept. 2019.

[3] Y. Du, W. ShangGuan, D. Rong and L. Chai, "RA-TSC: Learning Adaptive Traffic Signal Control Strategy via Deep Reinforcement Learning," *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand, 2019, pp. 3275-3280.

[4] X. Liang, X. Du, G. Wang and Z. Han, "A Deep Reinforcement Learning Network for Traffic Light Cycle Control," in *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1243-1253, Feb. 2019.

[5] P. Wang, Y. Li, S. Shekhar and W. F. Northrop, "A Deep Reinforcement Learning Framework for Energy Management of Extended Range Electric Delivery Vehicles," *2019 IEEE Intelligent Vehicles Symposium (IV)*, Paris, France, 2019, pp. 1837-1842.

[6] P. Wang, Y. Li, S. Shekhar and W. F. Northrop, "Actor-Critic based Deep Reinforcement Learning Framework for Energy Management of Extended Range Electric Delivery Vehicles," *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Hong Kong, China, 2019, pp. 1379-1384.

[7] Pearce, Tim, Mohamed Zaki, Alexandra Brintrup, and Andy Neel. "Uncertainty in neural networks: Bayesian ensembling." *arXiv preprint arXiv:1810.05546* (2018).

[8] Osband, Ian. "Risk versus uncertainty in deep learning: Bayes, bootstrap and the dangers of dropout." In *NIPS Workshop on Bayesian Deep Learning*. 2016.

[9] Gal, Yarin. "Uncertainty in deep learning." PhD diss., PhD thesis, University of Cambridge, 2016.

[10] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 449-458. JMLR. org, 2017.

[11] Dabney, Will, Mark Rowland, Marc G. Bellemare, and Rémi Munos. "Distributional reinforcement learning with quantile regression." In *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[12] P. Wang, Y. Li, S. Shekhar and W. F. Northrop, "Uncertainty Estimation with Distributional Reinforcement Learning for Applications in Intelligent Transportation Systems: A Case Study," *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand, 2019, pp. 3822-3827.

[13] J. Bernhard, S. Pollok and A. Knoll, "Addressing Inherent Uncertainty: Risk-Sensitive Behavior Generation for Automated Driving using Distributional Reinforcement Learning," *2019 IEEE Intelligent Vehicles Symposium (IV)*, Paris, France, 2019, pp. 2148-2155.

[14] C. M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[15] Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles." In *Advances in Neural Information Processing Systems*, pp. 6402-6413. 2017.

[16] Lu, Xiuyuan, and Benjamin Van Roy. "Ensemble sampling." In *Advances in neural information processing systems*, pp. 3258-3266. 2017.

[17] S. G. Wirasingha and A. Emadi, "Classification and Review of Control Strategies for Plug-In Hybrid Electric Vehicles," in *IEEE Transactions on Vehicular Technology*, vol. 60, no. 1, pp. 111-122, Jan. 2011.

[18] C. Martinez, X. Hu, D. Cao, E. Velenis, B. Gao and M. Wellers, "Energy Management in Plug-in Hybrid Electric Vehicles: Recent Progress and a Connected Vehicles Perspective", *IEEE Transactions on Vehicular Technology*, vol. 66, no. 6, pp. 4534-4549, 2017

[19] Hu, Xiaosong, Teng Liu, Xuewei Qi, and Matthew Barth. "Reinforcement Learning for Hybrid and Plug-In Hybrid Electric Vehicle Energy Management: Recent Advances and Prospects." *IEEE Industrial Electronics Magazine* 13, no. 3 (2019): 16-25.

[20] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Cambridge MA: MIT Press, 2018.

[21] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." *Nature* 518, no. 7540 (2015): 529.

[22] Y. Li, H. He, J. Peng and H. Wang, "Deep Reinforcement Learning-Based Energy Management for a Series Hybrid Electric Vehicle Enabled by History Cumulative Trip Information," in *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7416-7430, Aug. 2019.

[23] MacKay, David JC, and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.