Deep Reinforcement Learning Based Platooning Control for Travel Delay and Fuel Optimization

Chia-Cheng Yen[1], Hang Gao[2], and Michael Zhang[3]

***Abstract*— Vehicular emissions and traffic congestion have been deteriorated by highly urbanization. The worsen traffic burdens drivers with a higher cost and longer time on driving, and exposures pedestrians to unhealthy emissions such as PM, NO_x, SO_2 and greenhouse gases. In response to these issues, connected autonomous vehicles (CAVs), which enable information sharing between vehicles and infrastructure was proposed. With CAVs and advanced wireless technologies offering extremely low latency, platooning control can be realized to reduce the traffic delay, fuel consumption and emissions by improving traffic efficiency. However, conventional platooning control algorithms require complex computations and hence, are not a perfect candidate when applying to real-time operations. To overcome this issue, this work focuses on designing an innovative learning framework for platooning control capable of reducing the traffic delay and fuel consumption by the four basic platoon manipulations, e.g., split, acceleration, deceleration, and no-op. We integrate reinforcement learning (RL) with neural networks (NNs) to be able to model nonlinear relationships between inputs and outputs for a complex application. The experimental results reveal decreasing trends of the delay and fuel usage and a growing trend of the reward. They demonstrate that the proposed DRL platooning control optimizes the average delay and fuel consumption by fine-tuning speeds and sizes of platoons.**

***Index Terms*— Deep reinforcement learning, Connected autonomous vehicles (CAVs), Platooning control, Arrival timing vector, Travel delay, Fuel optimization**

I. INTRODUCTION

With excessively dense population in urban cities, traffic congestion has been part of our daily life and impacted on our emotion as well as the environment. Highly congested traffic lengthen the average sojourn time for people who need to commute everyday as well as increase the average fuel consumption (cost) for each driver and the average volume of noxious emissions to the environment. According to the report by INRIX [1], the annual congestion cost for each driver was 99 hours and $\$1,377$ in 2019 in the United States (US). Greenhouse gas emissions [2] contributed by transportation were 27% in 2020. Light-duty and Medium-duty (including heavy-duty) vehicles accounted for 57% and 26% of transportation greenhouse gas emissions in 2020, respectively. Undoubtedly, the ever-worsening congestion, fuel consumption, and emission issues attract public attention to address energy consumption and pollutant emissions generated by transportation systems. Thereby, many solutions with novel technologies have sprung up to address these issues.

Connected autonomous vehicle (CAV) [3] is an advanced technology which enables vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications among drivers, road side units (RSUs) and controllers. Research on CAV to improve mobility, safety and sustainability has been thriving in recent years. With an environment that most of vehicles are connected and all information is shared, platooning control can be fulfilled to improve traffic efficiency and safety. Platooning control is a technique that organizes a traffic flow into multiple groups, convoys, or platoons with close-following vehicles also known as road trains in order to increase the overall capacity of roads and reduce fuel consumption and emissions [4] in the traffic network. Platoons adopt cooperative adaptive cruise control (CACC) [5] with wireless communications to manipulate platooning formations and maneuvers. By forming vehicles into road trains, the overall throughput can be improved because gaps among CAVs are minimized. In addition, fuel consumption and emissions can be reduced due to lower air drag and speed variation for each CAV in a platoon [4]. Platooning can be considered as a effective control strategy for heavy-duty vehicles (HDVs) as well as a promising solution to reduce fuel consumption in HDVs [6]–[8].

However, platooning control methods requiring complicated computations are generally time-consuming and not amenable for real-time operations. Such intensive computation problems, on the other hand, are perfect applications for machine learning (ML). Deep reinforcement learning (DRL), one branch of many ML techniques, is an unsupervised learning framework that integrates reinforcement learning (RL) [9] with neural networks. DRL can be applied to an episodic traffic environment where a DRL agent takes predefined actions based on observations, and then receives rewards from the environment. The DRL agent learns to determine the optimal speed and size of each platoon that maximize the expected cumulative reward in order to reduce the delay and fuel consumption. In this research, we propose a DRL-based approach that selects the optimal speed and size for each platoon to manipulate platoon maneuvers so that each platoon can either cross the intersection as a whole without stopping or be split into two sub-platoons that the former sub-platoon can cross without stopping. The goal is to minimize the number of stopping vehicles to reduce the average delay and fuel consumption by vehicles.

[1]Chia-Cheng Yen is with the Department of Computer Science, University of California, Davis, CA 95616, USA ccyen@ucdavis.edu

[2]Hang Gao is with Institute of Transportation Studies and Department of Statistics, University of California, Davis, CA 95616, USA hangao@ucdavis.edu

[3]Michael Zhang is with the Department of Civil and Environmental Engineering, University of California, Davis, CA 95616, USA hmzhang@ucdavis.edu

The key contributions of this paper are as follows:

- We propose a DRL framework that integrates the deep learning, dueling architecture, and experience replay memory for platooning control to minimize the average delay and fuel consumption by vehicles.
- We consider different arrival times of vehicles in a platoon and adopt a vector of arrival timings of vehicles as a state to emphasize differences among platoons.
- We apply appropriate actions to individual platoons based on their vectors of arrival timings (states) instead of using the same action for all platoons.

II. PROBLEM STATEMENT AND SYSTEM MODEL

A. Problem Statement

CAVs provide great controllability that can lead to lower fuel use and traffic delay in transportation networks. Prior work has demonstrated that properly optimized CAV platoons can reduce both travel time and emissions by as much as 40% when traveling through an isolated intersection. But the method used to achieve this result, optimal trajectory control, is complex and time-consuming to solve; hence, it is not amenable for real-time operations. Such a complex problem, on the other hand, can be a perfect application for ML. In this research, we apply DRL-based platooning control to address the long delay as well as high fuel consumption and emissions by improving the traffic efficiency. To avoid unnecessary accelerations/decelerations that result in longer delay and higher fuel consumption, a DRL agent learns to apply proper platoon manipulations (actions) to platoons based on their arrival timing vectors (states).

B. System Model

We conduct experiments on a corridor where four consecutive intersections are signalized and traffic movements with fixed routes, e.g., $W-E$ and $N-S$ are controlled by traffic lights with fixed-time signals as shown in Fig. 1. The assumption of a high penetration ratio of CAVs is made; thus, information such as velocity, acceleration/deceleration, and headway can be shared with extremely low latency by 5G wireless technologies. Platoons with fixed routes and close-following vehicles are generated periodically throughout an experiment. Their platoon maneuvers are controlled by a DRL-based platooning controller aiming at reducing the average delay and fuel consumption. Note that the corridor is designed to study the impact of applying platooning control on time and fuel savings as well as examine whether the proposed DRL-based platooning controller is capable of cleverly splitting a platoon into two when the remaining green time is not enough for all vehicles in a platoon to pass. The impact of merging any two platoons is not considered in this work.

III. THE PROPOSED DEEP REINFORCEMENT LEARNING BASED PLATOONING CONTROL

To deal with the dynamic of traffic flow and properly select the optimal platooning manipulations adaptive to various remaining green time that minimize the average delay and fuel consumption, we adopt an unsupervised learning approach that is able to learn without human intervention. DRL is an unsupervised learning framework that integrates RL with neural networks as shown in Fig. 2. We implement two control modules in the proposed framework: traffic light control (TLC) and platooning control. TLC is responsible for controlling traffic lights using fixed-time signals; for example, 20s of green interval plus 3s of yellow change interval for $N-S$ and 30s of green interval plus 3s of yellow change interval for $E-W$. Platooning control is responsible for managing the speeds and sizes of the platoons in the environment based on the selected actions by the DRL agent. A DRL agent learns to select actions that maximize the expected cumulative reward by trial and error manner. In this work, we propose a DRL-based algorithm which trains a DRL agent to determine an optimal action every 2s (a time step) to optimize the average delay and fuel consumption. For each time step, the DRL agent interacts with the traffic environment where it first chooses an action (one of the four platooning manipulations), observes some changes from the environment (the next state), and receives a reward. Given the observations and reward, the DRL agent computes a difference value (i.e., loss) from the Q-network and target network and updates the neural networks by the gradient of the loss value.

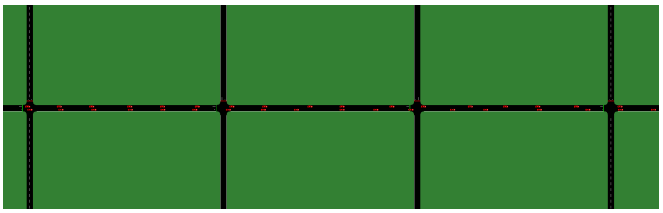

Fig. 1: An illustration shows a row of signalized intersections where traffic flows move towards fixed routs.

A. State Representation

We formulate a state as a vector of arrival timings for an individual platoon. The arrival timing vector (ATV) includes arrival timings for all the vehicles in a platoon. An arrival timing of a vehicle is determined by three arrival timing variables proposed in [10]. The three arrival timing variables are 1) cruising time-to-arrival t_c, 2) earliest time-to-arrival t_e, and 3) latest time-to-arrival t_l. They can be calculated by

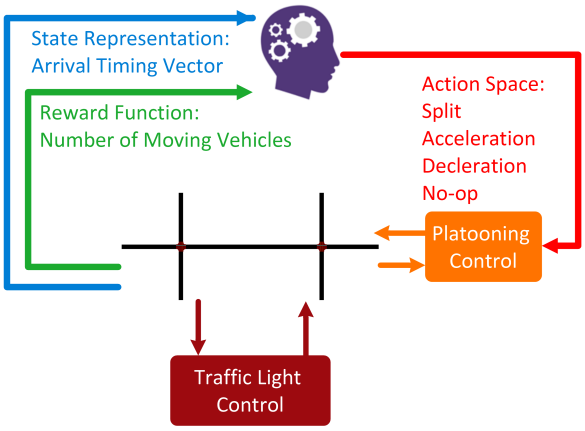

Fig. 2: The proposed DRL platooning control framework from which The DRL agent learns to select the actions that return the maximum expected cumulative reward. There are two control modules in the proposed framework: traffic light control (TLC) and platooning control. Platooning control is responsible for managing platoons and TLC is responsible for controlling traffic lights using fixed-time signals; for example, 20s of green interval plus 3s of yellow change interval for $N-S$ and 30s of green interval plus 3s of yellow change interval for $E-W$

$$t_c = \frac{d_1}{v_1} \tag{1}$$

$$t_e = \frac{d_1 - v_1 \cdot \frac{\pi}{2\alpha}}{v_{lim}} + \frac{\pi}{2\alpha} \tag{2}$$

$$t_l = \frac{d_1 - v_1 \cdot \frac{\pi}{2\beta}}{v_{coast}} + \frac{\pi}{2\beta} \tag{3}$$

$$\alpha = min\left\{\frac{2 \cdot a_{max}}{v_{lim} - v_1}, \sqrt{\frac{2 \cdot jerk_{max}}{v_{lim} - v_1}}\right\} \tag{4}$$

$$\beta = min\left\{\frac{2 \cdot a_{max}}{v_1 - v_{coast}}, \sqrt{\frac{2 \cdot jerk_{max}}{v_1 - v_{coast}}}\right\} \tag{5}$$

where d_1 is the distance from the current position to the intersection, v_1 is the current speed of the vehicle, α and β are coefficients to calculate t_e and t_l, $jerk_{max}$ is a pre-defined constant denoting the maximum changing rate of acceleration or deceleration, v_{lim} is a pre-defined constant denoting the speed restriction of the current roadway, and v_{coast} is a pre-defined constant denoting the coasting speed.

For each vehicle in a platoon, its optimal arrival timing t_{arr} can be determined by the following rules: 1) assign t_c to be its t_{arr} if t_c is within a green interval T, 2) assign $\min\{t_e, t_c\}$ to be its t_{arr} if the intersection of $[t_e, t_c]$ and a green interval T is not empty, 3) assign $\min\{t_c, t_l\}$ to be its t_{arr} if the intersection of $[t_c, t_l]$ and a green interval T is not empty, and 4) assign the beginning time of the next green interval to be its t_{arr} if the vehicle has no choice and must stop and wait for the next green interval. Note that the rules are ranked by priority; that is to say, no need to move to rule 2, 3, and 4 if rule 1 is applicable. If rule 1 is not applicable and rule 2 is applicable, then no need to go to rule 3, and 4 and so forth. An example of different arrival timings assigned to vehicles in the same platoon is shown in Fig. 3 where t_e^n is earliest time-to-arrival for the n-th vehicle, t_c^n denotes the cruising time-to-arrival for the n-th vehicle, and t_l^n means latest time-to-arrival for the n-th vehicle. ATV can be represented by $[t_{arr}^1, t_{arr}^2, t_{arr}^3, \ldots, t_{arr}^n]$. ATV with arrival timings of all the member vehicles can be meaningful state representation that encodes information of whether a platoon has to be split or not. For example, in Fig. 3, there are 6 vehicles in a given platoon. To help the readers to understand the concept of ATV, we mark the three timing variables for the 1-st and 4-th vehicles as indicators to show feasible trajectories. As can be observed, both t_e^1 and t_c^1 are in the current green interval (51s to 80s). Based on the four rules, instead of t_e^1, t_c^1 is assigned to t_{arr}^1 because there is no need to accelerate and consume extra fuel if the vehicle can pass an intersection at the current speed. But, with respect to the 4-th vehicle, t_e^4 and t_c^4 are in the red interval and only t_l^4 is in the next green interval (101s to 130s). In this case, t_l^4 is assigned to t_{arr}^4 because the vehicle needs to decelerate to avoid a fully stop before a traffic light turns green. After assigning timings to the rest of the vehicles, the former three vehicles can pass the intersection within the current green interval if they keep the current speed but the latter three vehicles must stop and wait for the next green interval. Hence, the ATV of the platoon has distinct arrival timings among member vehicles and this pattern helps the DRL agent to recognize if a platoon needs to be split to avoid fully stops and unnecessary accelerations/decelerations. Fully stops and unnecessary accelerations/decelerations are platooning manipulations which lead to extra delay and fuel usage. The number of these misbehaviors has to be minimized to achieve optimal delay and fuel consumption. The trajectories as shown in Fig. 3 demonstrates that the whole platoon can avoid fully stops if it can be split into two subplatoons at t_1. The former can keep the current speed and pass the intersection at t_2 while the latter can first decelerate and then accelerate to pass the intersection later at t_3 without any stop.

B. Action Space

Recent research [11], [12] has demonstrated that RL can be trained by using images as states and the others [13]–[17] had formulated a matrix for an intersection by marking 0 and 1 based on coordinates of vehicles to describe a state. In these previous papers, one characteristic they have in common is that one state is mapped to one action. Namely, they considered a fully observable environment as a state and apply an action to an environment. However, in our case, it is unrealistic to map a state to an action and then, apply the same action to all platoons in the environment

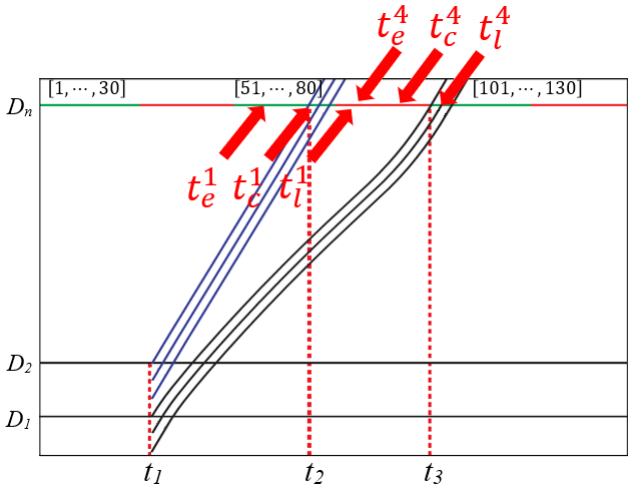

Fig. 3: An example of different arrival timings for vehicles in the same platoon

because their profiles such as positions, speeds and sizes are different; thus, different actions are required for platoons with different profiles. In response to this issue, different actions are applied to individual platoons according to their own states (ATVs). A large action space would impede convergence of the learning process. Thus, we only consider four actions, namely, 1) split, 2) acceleration, 3) deceleration, and 4) no-op which stands for *no operation*. In the no-op action, the agent will not apply any change to the platoons. These manipulations are sufficient to accomplish all platoon maneuvers in our simulations.

C. Reward Function

As mentioned, the goal is to minimize the number of fully stops and unnecessary accelerations/decelerations in order to reduce the delay and fuel consumption. In other words, the number of moving vehicles has to be maximized. Thus, we choose the number of moving vehicles as the reward function. In particular, the vehicles with any speed equal to or larger than the eco-speed (i.e., 15m/s or 34 mph) are considered as the moving vehicles. The purpose of setting the speed limit (15m/s) for moving vehicles is to avoid the circumstances in which all vehicles enter and nearly stop (1m/s) during the simulations. Moreover, the result of setting a higher speed limit (20m/s or 45mph) is consistent with the eco-speed one.

D. The Overall Architecture

In this section, we elaborate more on the underlying architecture of the framework and introduce detailed deep learning techniques utilized to train a DRL model. The DRL architecture integrating several components that facilitate the learning process is shown in Fig. 4. First, usually, there is strong correlations among several consecutive states especially in time-series data. The experience memory replay component proposed in [18] is applied to break the strong temporal correlations and speeds up the learning process for the DRL agent. The second component included in the architecture is the two separated neural networks, Q-network and target network [19]. In DRL, neural networks are applied as non-linear approximation function to map a state to an action that returns the maximum Q-value. Q-network and target network share exactly the same number of neurons and architecture. The set of neural parameters θ is copied over to the target network every τ steps. The current state and action are first fed into the Q-network for approximating a Q-value. Previous experiences sampled from the replay memory buffer are inputs to the target network for approximating a target Q-value. The loss value, L, calculated by the Q-value and target Q-value will be partially differentiated to get the gradient in which the agent will have a sense and know how to update the Q-network by backwardpropagation to fine-tune the set of neural parameters θ. The dueling architecture [20] is the third component that helps the selection of action by separating the estimations of state-value and action advantage. Instead of using one single sequence for value estimations, a Q-network with two separated sequences is implemented, one for value (blue cycle) and the other for action advantage (brown cycles). We include this component because in some states, it matters which action needs to be taken, but most states, the selection of action has no influence on what happens. Hence, we separate the estimations of state-value and action advantage in order to learn which state-values are higher without going through and learn every single state-action pair.

IV. RESULTS AND DISCUSSIONS

In this section, we will elaborate on the hyperparameters in DRL framework and experimental results regarding the learning performance of the DRL agent, average delay and average fuel consumption.

A. Hyperparameters

As shown in Table I, a set of hyperparameters used in the proposed DRL framework is pre-defined. Traffic arrival ratio for each intersection is 900 vehicles/hour/lane. The ϵ-greedy function determines the probability of doing exploration or exploitation where ϵ_i as the initializing value is 1.0, $\epsilon_f = 0.01$ is the finalizing value, and the decay ratio is 300. Learning ratio α determines the weight of newly learned knowledge. α approaching 0 makes the agent exploit prior knowledge instead of learning something new while α approaching 1 makes it explore new potential rather than using prior knowledge. α is set to 0.0001 in the simulations. Discount factor γ determines the importance of future rewards. γ closer to 0 makes the agent behave myopically by only considering immediate rewards while γ closer to 1 makes it seek a long-term outcome by weighting future rewards with a higher value. γ is set to 0.99 in the simulations. The replay memory size M used to store previous experiences is set to 30000, and mini-batch size B used to sample experiences from the memory is set to 1024. The number of training episode $E = 700$ is selected by trial and error.

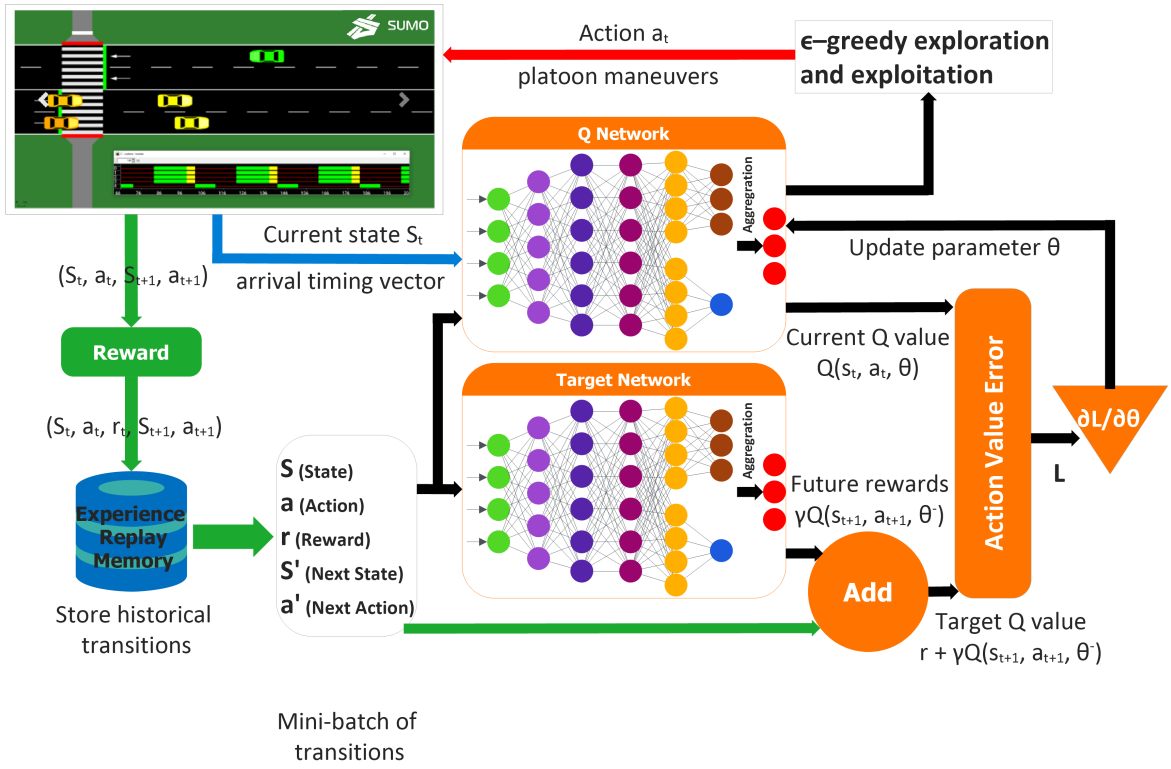

Fig. 4: An overview of the DRL architecture for training a platooning control agent.

TABLE I: Summary of hyperparameters in the proposed DRL framework

Parameter	Description
Traffic arrival ratio:	900 v/h/l
ϵ-greedy decay function:	calculate the probability of exploration
$\epsilon_f + (\epsilon_i - \epsilon_f) \times e^{-\frac{episode}{decay\ ratio}}$	and exploitation where decay ratio is 300
Initializing $\epsilon_i = 1.0$	the beginning value of ϵ
Finalizing $\epsilon_f = 0.01$	the final value of ϵ
Learning ratio $\alpha = 0.0001$	learning ratio for updating the neural network
Discount ratio $\gamma = 0.99$	discount for future rewards
Memory size $M = 30000$	the size of entire replay memory buffer
Mini-batch size $B = 1024$	the size of samples that will be reused
Maximum number of episodes $E = 700$	the total number of training episodes

B. *Learning Performance*

We discuss the impact of platooning control on the average fuel consumption by each vehicle and the average delay of each vehicle. In the simulations, the agent is allowed to explore more than exploiting at a pre-training stage. The pre-training stage is set to go off in 100 episodes. During the pre-training stage, the agent randomly selects an action instead of using the optimal one in order to discover potential actions that would return a better reward. In Fig. 5, the trend of the reward shows that the DRL platooning agent keeps obtaining better rewards after the pre-training stage. Namely, it had learned how to either split platoons or control the cruise speeds of the platoons to reduce fuel usage by vehicles during the previous stage (pre-training) and applies the knowledge to manipulate platoons in the current stage (training). As more and more training episodes had been learned (close to the 700-th episode), the agent learns to maximize rewards by controlling platoons to achieve lower fuel consumption.

On the other hand, as shown in Fig 6, the average delay reveals a decreasing trend when the reward gets higher. In the simulation, we observe two reasons to explain the decreasing trend. First, we observe that the average delay decreases because the speeds of the platoons can be manipulated to avoid stopping in front of an intersection. A platoon is able

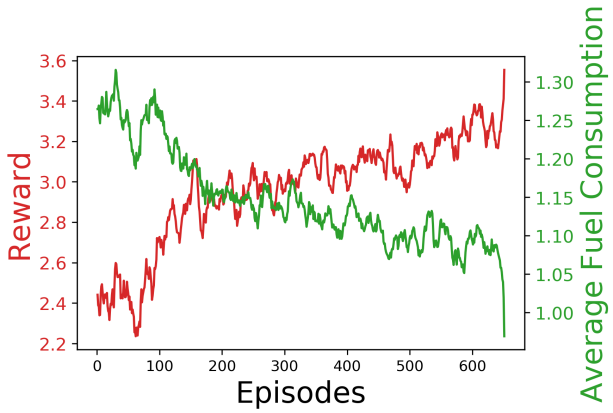

Fig. 5: The episode reward and average fuel consumption by vehicles.

to pass an intersection through acceleration before the end of a green interval if earliest time-to-arrivals for all the vehicles in the platoon can be in the same green interval. Hence, acceleration that helps to reduce the delay could be one contributing factor to the decreasing trend of the delay. Second, platoons which can not pass an intersection as a whole could be split into two where the first half of the platoons keep the same speed and pass but the second half of them have to fully stop and wait; thus, the first half of them could be the other contributing factor.

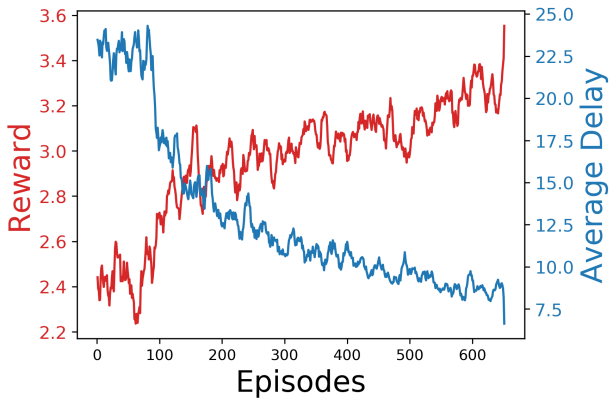

Fig. 6: The episode reward and average delay by vehicles.

V. CONCLUSION

With the extremely low latency guaranteed by 5G NR, information sharing among CAVs within a few milliseconds becomes possible, which facilitates research on platooning control to move further. Previous work has demonstrated feasibility of improving latency and fuel usage by optimizing platooning control but tremendous mathematical efforts are required to complete the computation and hence, the computation complexity is high and cannot be applicable to real-world applications. In this work, we have demonstrated that the reduction of the delay and fuel usage is feasible by using DRL. The DRL agent learns how to deal with highly changeable traffic flow and select the optimal action (e.g., change the current speed or re-size a platoon) to minimize the delay and fuel consumption. The trends of the reward, average delay, and average fuel consumption show promising results if DRL-based platooning control can be applied. In our future work, we plan to conduct more complicated scenarios such as considering multi-models traffic and a larger traffic network.

REFERENCES

[1] G. Cookson, "Inrix global traffic scorecard," 2020.

[2] "Fast facts: U.s. transportation sector greenhouse gas emissions 1990-2020," https://www.epa.gov/greenvehicles/fast-facts-transportation-greenhouse-gas-emissions.

[3] E. Uhlemann, "Introducing connected vehicles [connected vehicles]," *IEEE vehicular technology magazine*, vol. 10, no. 1, pp. 23–31, 2015.

[4] M. Michaelian and F. Browand, "Field experiments demonstrate fuel savings for close-following," 2000.

[5] L. Güvenç, I. M. C. Uygan, K. Kahraman, R. Karaahmetoglu, I. Altay, M. Sentürk, M. T. Emirler, A. E. H. Karci, B. A. Guvenc, E. Altug *et al.*, "Cooperative adaptive cruise control implementation of team mekar at the grand cooperative driving challenge," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1062–1074, 2012.

[6] A. Al Alam, A. Gattami, and K. H. Johansson, "An experimental study on the fuel reduction potential of heavy duty vehicle platooning," in *13th international IEEE conference on intelligent transportation systems*. IEEE, 2010, pp. 306–311.

[7] F. Browand, J. McArthur, and C. Radovich, "Fuel saving achieved in the field test of two tandem trucks," 2004.

[8] S. Tsugawa, S. Kato, and K. Aoki, "An automated truck platoon for energy saving," in *2011 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2011, pp. 4109–4114.

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: An Introduction, 2nd edition*. MIT press, Oct. 2018.

[10] Z. Wang, G. Wu, and M. J. Barth, "Cooperative eco-driving at signalized intersections in a partially connected and automated vehicle environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 2029–2038, 2019.

[11] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, "Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction," *Sensors*, vol. 17, no. 4, p. 818, 2017.

[12] C.-C. Yen, D. Ghosal, M. Zhang, and C.-N. Chuah, "A deep on-policy learning agent for traffic signal control of multiple intersections," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[13] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network," *arXiv preprint arXiv:1705.02755*, 2017.

[14] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142*, 2016.

[15] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1243–1253, 2019.

[16] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, 2016.

[17] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2496–2505.

[18] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[20] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.