A Deep Reinforcement Learning Network for Traffic Light Cycle Control

Xiaoyuan Liang, Xunsheng Du, *Student Member, IEEE*, Guiling Wang, *Member, IEEE*, and Zhu Han, *Fellow, IEEE*

***Abstract*—Existing inefficient traffic light cycle control causes numerous problems, such as long delay and waste of energy. To improve efficiency, taking real-time traffic information as an input and dynamically adjusting the traffic light duration accordingly is a must. Existing works either split the traffic signal into equal duration or only leverage limited traffic information. In this paper, we study how to decide the traffic signal duration based on the collected data from different sensors. We propose a deep reinforcement learning model to control the traffic light cycle. In the model, we quantify the complex traffic scenario as states by collecting traffic data and dividing the whole intersection into small grids. The duration changes of a traffic light are the actions, which are modeled as a high-dimension Markov decision process. The reward is the cumulative waiting time difference between two cycles. To solve the model, a convolutional neural network is employed to map states to rewards. The proposed model incorporates multiple optimization elements to improve the performance, such as dueling network, target network, double Q-learning network, and prioritized experience replay. We evaluate our model via simulation on a Simulation of Urban MObility simulator. Simulation results show the efficiency of our model in controlling traffic lights.**

***Index Terms*—Reinforcement learning, deep learning, traffic light control, vehicular network.**

I. Introduction

The intersection management of busy or major roads is primarily done through traffic lights, whose inefficient control causes numerous problems, such as long delay of travelers and huge waste of energy. Even worse, it may also incur vehicular accidents [1], [2]. Existing traffic light control either deploys fixed programs without considering real-time traffic or considering the traffic to a very limited degree [3]. The fixed programs set the traffic signals equal time duration in every cycle, or different time duration based on historical information. Some control programs take inputs from sensors such as underground inductive loop detectors to detect the existence of vehicles in front of traffic lights. However, the inputs are processed in a very coarse way to determine the duration of green/red lights.

In some cases, existing traffic light control systems work, though at a low efficiency. However, in many other cases, such as a football event or a more common high traffic hour scenario, the traffic light control systems become paralyzed. Instead, we often witness an experienced policeman directly manages the intersection by waving signals. In high traffic scenarios, a human operator observes the real time traffic condition in the intersecting roads and smartly determines the duration of the allowed passing time for each direction using his/her long-term experience and understanding about the intersection, which is very effective. This observation motivates us to propose a smart intersection traffic light management system which can take real-time traffic condition as input and learn how to manage the intersection just like the human operator. To implement such a system, we need 'eyes' to watch the real-time road condition and 'a brain' to process it. For the former, recent advances in sensor and networking technology enables taking real-time traffic information as input, such as the number of vehicles, the locations of vehicles, and their waiting time [4]. For the 'brain' part, reinforcement learning, as a type of machine learning techniques, is a promising way to solve the problem. A reinforcement learning system's goal is to make an action agent learn the optimal policy through interacting with the environment to maximize the reward, e.g., the minimum waiting time in our intersection control scenario. It usually contains three components: states of the environment, action space of the agent, and reward from every action [5]. A well-known application of reinforcement learning is AlphaGo [6], followed by AlphaGo Zero [7]. AlphaGo, acting as the action agent in a Go game (environment), first observes the current image of the chessboard (state), and takes the image as the input of a reinforcement learning model to determine where to place the optimal next playing piece 'stone' (action). Its final reward is to win the game or to lose. Thus, the reward may not be obvious during the playing process but becomes clear when the game is over. When applying reinforcement learning to the traffic light control problem, the key point is to define the three components at an intersection and quantify them to be computable.

Some previous works propose to dynamically control the traffic lights using reinforcement learning. Some define the states

Manuscript received January 22, 2018; revised November 21, 2018; accepted November 30, 2018. Date of publication January 3, 2019; date of current version February 12, 2019. This work was supported in part by the National Science Foundation under Grants CMMI-1844238, CNS-1717454, and CNS-1731424, and in part by the Air Force Office of Scientific Research under Grant MURI-18RT0073. The review of this paper was coordinated by Dr. Z. Fadlullah. *(Corresponding author: Xiaoyuan Liang.)*

X. Liang and G. Wang are with the Department of Computer Science, New Jersey Institute of Technology, Newark, NJ 07102 USA (e-mail: xl367@njit.edu; gwang@njit.edu).

X. Du is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA (e-mail: xunshengdu@gmail.com).

Z. Han is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul, South Korea (e-mail: zhan2@uh.edu).

Digital Object Identifier 10.1109/TVT.2018.2890726

by the number of waiting vehicles or the waiting queue length [4], [8]. But real traffic situation cannot be accurately captured by only the number of waiting vehicles or queue length [9]. With the popularization of vehicular networks and sensor networks, more accurate on-road traffic information can be extracted, such as vehicles' speed and waiting time [10]. However, rich information causes the number of states to increase dramatically. When the number of states increases, the complexity in a traditional reinforcement learning system grows exponentially. With the rapid development of deep learning [11], deep neural networks have been employed to deal with the large number of states, which constitutes a deep reinforcement learning model [12]. A few recent studies have proposed to apply deep reinforcement learning in the traffic light control problem [13], [14]. But there are two main limitations in existing studies: (1) the traffic signals are usually split into fixed-time intervals, and the duration of green/red lights can only be a multiple of this fixed-length interval, which is not efficient in many situations; (2) the traffic signals are designed to change in a random sequence, which is not a safe or comfortable way for drivers. In this paper, we study the problem on how to control the traffic light signal duration in a cycle based on the extracted information from vehicular networks or sensor networks.

Our general idea is to mimic an experienced operator to control the signal duration in every cycle based on the information gathered from vehicular networks. To implement such an idea, the operation of the experienced operator is modeled as an Markov Decision Process (MDP). The MDP is a high-dimension model, which contains the time duration of every phase. The system learns the control strategy based on the MDP by trial and error in a deep reinforcement learning model. To fit a deep reinforcement learning model, we divide the whole intersection into grids and build a matrix, each element of which is the vehicles' information in the corresponding grid collected by vehicular networks or extracted from cameras via image processing. The matrix is defined as the states and the reward is the cumulative waiting time difference between two cycles. In our model, a convolutional neural network is employed to match the states and expected future rewards. Note that, every traffic light's action produced from our model affects the environment. When the traffic flow changes dynamically, the environment becomes unpredictable. To solve this problem, we employ a series of state-of-the-art techniques in our model to improve the performance, including dueling network [15], target network [12], double Q-learning network [16], and prioritized experience replay [17].

Our contribution of the paper includes 1) We are the first to combine dueling network, target network, double Q network and prioritized experience replay into one framework to solve the traffic light control problem, which can be easily applied into other problems. 2) We propose a control system to decide the phases' time duration in a whole cycle instead of dividing the time into segments. 3) Extensive experiments on a traffic micro-simulator, Simulation of Urban MObility (SUMO) [18], show the effectiveness and high-efficiency of our model.

The reminder of this paper is organized as follows. The literature review is presented in Section II. The model and problem statement are introduced in Section IV. The background on reinforcement learning is introduced in Section III. Section V details our reinforcement learning model in the traffic light control system. Section VI extends the reinforcement learning model into a deep learning model to handle the complex states in the our system. The model is evaluated in Section VII. Finally, the paper is concluded in Section VIII.

II. Literature Review

Previous works have been done to dynamically control adaptive traffic lights. But due to the limited computing power and simulation tools, early studies focus on solving the problem by fuzzy logic [19], linear programming [20], etc. In these works, road traffic is modeled by limited information, which cannot be applied in large scale.

With the success of deep learning in artificial intelligence, more and more researchers use deep learning to solve transportation problems. Deep learning includes supervised learning, unsupervised learning and reinforcement learning, which have been applied in network traffic control, such as traffic prediction and routing [21]. In the traffic light control problem, since no labels are available and the traffic scenario is influenced by a series of actions, reinforcement learning is a good way to solve the problem and has been applied in traffic light control since 1990s. El-Tantawy *et al.* [4] summarize the methods from 1997 to 2010 that use reinforcement learning to control traffic light timing. During this period, the reinforcement learning techniques are limited to tabular Q learning and a linear function is normally used to estimate the Q value. Due to the technique limitation at the time in reinforcement learning, they usually make a small-size state space, such as the number of waiting vehicles [8], [22] and the statistics of traffic flow [23], [24]. A signal control system is proposed in [25]. The authors use the queue length and current light time as the state and use a linear function to approximate the Q values. A cooperative traffic light control system based on reinforcement learning is proposed in [22]. The authors propose to cluster vehicles and use a linear function to approximate the Q values; however, only the queue information is used in the states. The complexity in a traffic road system can not be actually presented by such limited information. When much useful relevant information is omitted in the limited states, it seems unable to act optimally in traffic light control [9].

With the development of deep learning and reinforcement learning, they are combined together as deep reinforcement learning to estimate the Q value. Some researchers have applied deep reinforcement learning to control the wireless communication [27], [28], but the systems cannot be directly applied in traffic light control scenarios due to different actions and states. We summarize the recent studies that use the value-based deep reinforcement learning to control traffic lights in Table I. There are three limitations in these previous studies. Firstly, most of them test their models in a simple cross-shape intersection with through traffic only [13], [14]. Secondly, none of the previous works determines the traffic signal timing in a whole cycle. Thirdly, deep reinforcement learning is a fast developing field, where a lot of new ideas are proposed in these two years, such as dueling deep Q network [15], but they have not been applied in traffic control. In this paper, we make the following

TABLE I
LIST OF PREVIOUS STUDIES THAT USE VALUE-BASED DEEP REINFORCEMENT LEARNING TO ADAPTIVELY CONTROL TRAFFIC SIGNALS

Study	State	Action	Reward	Time step	Note
Genders *et al.* (2016) [2]	Position speed	4 phases	Change in cumulative delay	NA	Convolutional neural network
Li *et al.* (2016) [11]	Queue length	2 phases	Difference between flows in two directions	5s	Stacked auto-encoders
Van Der Pol (2016) [12]	Position	2 phases	Teleport, wait time, stop, switch, and delay	1s	Double Q network Prioritized experience replay
Gao *et al.* (2017) [30]	Position speed	4 phases	Change in cumulative staying time	6/10s	Convolutional neural network Experience replay

progress. Firstly, our intersection scenario contains multiple phases, which corresponds a high-dimension action space in a cycle. Secondly, our model guarantees that the traffic signal time smoothly changes between two neighboring actions, which is precisely defined in the MDP model. Thirdly, we employ the state-of-the-art techniques in value-based reinforcement learning algorithms to achieve good performance.

III. BACKGROUND ON DEEP REINFORCEMENT LEARNING

Reinforcement Learning (RL) is a type of algorithms in machine learning. It interacts with the environment to learn better actions to maximize the objective reward function in the long run through trial and error. In reinforcement learning, an agent, the action executor, takes an action and the environment returns a numerical reward based on the action and the current state. A four-tuple $\langle S, A, R, T\rangle$ can be used to define the reinforcement learning model:

- S : the possible state space. s is a specific state ($s \in S$);
- A : the possible action space. a is an action ($a \in A$);
- R : the reward space. $r_{s,a}$ denotes the reward in taking action a at state s;
- T : the transition function space among all states, which represents the probability of the transition from one state to another.

In a deterministic model, T is usually omitted.

A policy is made up of a series of consequent actions. The goal in reinforcement learning is to learn an optimal policy to maximize the cumulative expected rewards starting from the initial state. Generally speaking, the agent at one specific state s takes an action a to reach state s' and gets a reward r, which is denoted by $\langle s, a, r, s'\rangle$. Let t denote the tth step in the policy π. The cumulative reward in the future by taking action a at state s is defined by $Q(s, a)$ in the following equation,

$$Q^{\pi}(s, a) = E\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \mid s_t = s, a_t = a, \pi\right]$$
$$= E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, a_t = a, \pi\right]. \quad (1)$$

In the equation, γ is the discount factor, which is usually in $[0, 1)$. It means the nearest rewards are worthier than the rewards in the future.

The optimal action policy π^* can be obtained recursively. If the agent knows the optimal Q values of the succeeding states, the optimal policy simply chooses the action that achieves the highest cumulative reward. Thus, the optimal $Q(s, a)$ is calculated based on the optimal Q values of the succeeding states. It can be expressed by the Bellman optimality equation to calculate $Q^{\pi^*}(s, a)$,

$$Q^{\pi^*}(s,a) = E_{s'}\left[r_t + \gamma \max_{a'} Q^{\pi^*}(s', a') \mid s, a\right]. \quad (2)$$

The intuition is that the cumulative reward is equal to the sum of the immediate reward and optimal future reward thereafter. If the estimated optimal future reward can be obtained, the cumulative reward since now can be calculated. This equation can be solved by dynamic programming, but it requires that the number of states is finite to make the computing complexity manageable. When the number of states becomes large, a function θ is needed to approximate the Q value.

IV. PROBLEM STATEMENT

This paper targets on controlling the traffic lights at road intersections. A traffic light at an intersection has three signals: green, yellow and red. When there are vehicles from multiple directions at an intersection, one traffic light may not be enough to manage all the vehicles and multiple traffic lights need to cooperate at a multi-direction intersection. A status is defined as one of all the legal combinations of all traffic lights' red and green signals omitting the yellow signals. At an intersection, the traffic signal guides vehicles from non-conflicting directions at one time by changing the traffic lights' statuses. The time duration staying at one status is called one phase. The number of phases is decided by the number of legal statuses at an intersection. All the phases cyclically change in a fixed sequence to guide vehicles to pass the intersection. It is called a cycle when the phases repeat once. The sequence of phases in a cycle is fixed, but the duration of every phase is adaptive based on the current traffic condition. If one phase needs to be skipped, its duration can be set 0 second. In our problem, we dynamically adjust the duration in every phase to deal with different traffic situations at an intersection to minimize the delay.

Our problem is to optimize the efficiency of the intersection usage by dynamically changing every phase's duration of a traffic light via learning from historical experiences. The duration of a phase should be extended strategically if there are more vehicles in that direction. In this paper, we build a deep Q learning network to learn the timing strategy of every phase to optimize

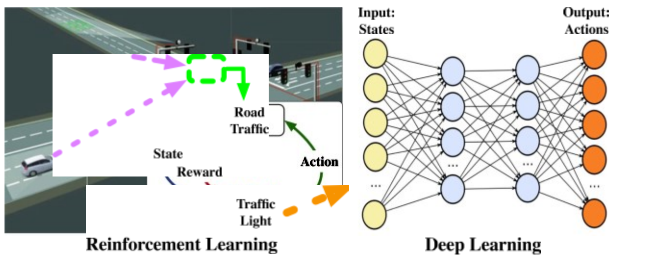


Fig. 1. The traffic light control model in our system. The left side shows the intersection scenario where the traffic light controller gathers road traffic information first and it is controlled by the reinforcement learning model; the right side shows a deep neural network to help the traffic light controller choose an action.

the traffic management. Our network self-updates by continuously receiving states and rewards from the environment. The model is shown in Fig. 1. The left side shows the structure in a traffic light. The traffic light first gathers road traffic information via a vehicular network [10] or other tools, which is presented by the dashed purple lines in the figure. The traffic light processes the data to obtain the road traffic's state and reward, which has been assumed in many previous studies [9], [14], [26]. The traffic light chooses an action based on the current state and reward using a deep neural network shown in the right side. The left side is the reinforcement learning part and the right side is the deep learning part.

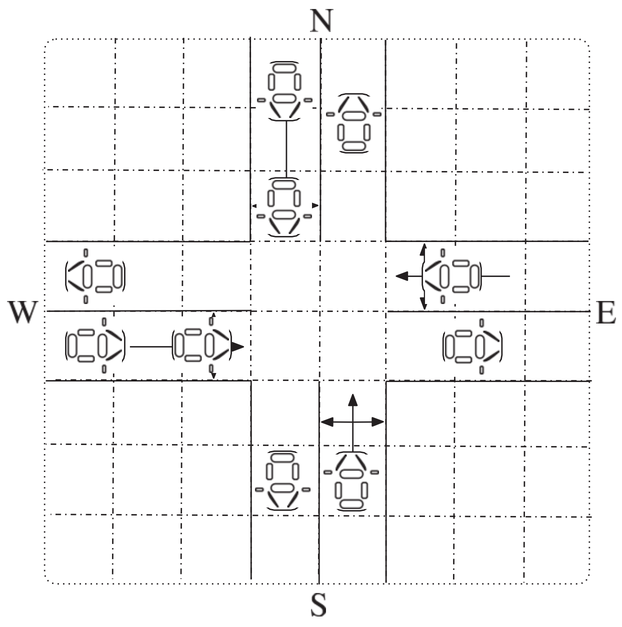
V. Our Reinforcement Learning Model

In this section, we define the three elements of our RL model: states, actions and rewards.

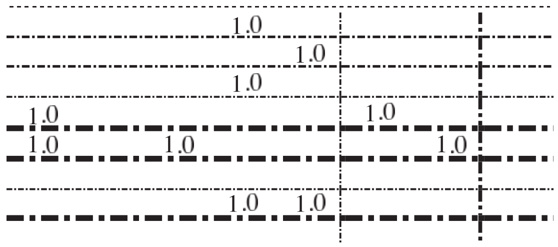
A. States

We define the states based on the position and speed of vehicles at an intersection. Through a vehicular network or other tools, vehicles' position and speed can be obtained [10]. The traffic light can extract a virtual snapshot image of the current intersection. The whole intersection is divided into same-size small square-shape grids. The length of grids, c, should guarantee that no two vehicles can be held in the same grid and one entire vehicle can be put into a grid to reduce computation. In every grid, the state value is a two-value vector $< position, speed >$ of the inside vehicle. The position dimension is a binary value, which denotes whether there is a vehicle in the grid. If there is a vehicle in a grid, the value in the grid is 1; otherwise, it is 0. The value in the speed dimension is an integer, denoting the vehicle's current speed in m/s.

Fig. 2 is an example to show how to set up the state values. Fig. 2(a) shows a snapshot of the traffic status at a simple one-lane four-way intersection, which is divided into square-shape grids. The position matrix has the same size of the grids, which is shown in Fig. 2(b). In the matrix, one cell corresponds to one grid in Fig. 2(a). The blank cells mean no vehicle in the corresponding grid, which are 0. The other cells with vehicles inside are set 1.0. The value in the speed dimension is built in a similar way. If there is a vehicle in the grid, the corresponding value is the vehicle's speed; otherwise, it is 0.



(a) The snapshot of traffic on a road at one moment



(b) The corresponding position matrix on this road

Fig. 2. The process to build the state matrix.

B. Actions

In our model, the actions' space is defined by how to update the duration of every phase in the next cycle. Considering the system may become unstable if the duration change between two cycles is too large, we specify a change step. In this paper, we set it to be 5 seconds. We model the duration changes of two phases between two neighboring cycles as a high-dimension MDP. In the model, the traffic light changes only one phase's duration by 5 seconds if there is any change.

We take the intersection in Fig. 2(a) as an example. At the intersection, there are four phases, north-south green, north-east&south-west green, east-west green, and east-south&west-north green. The other unmentioned directions are red by default. The yellow signals are omitted here and will be presented later. Let a four-tuple $< t_1, t_2, t_3, t_4 >$ denote the duration of the four phases in current cycle. The legal actions in the next cycle is shown in Fig. 3. In the figure, one circle means the durations of the four phases in one cycle. Note that the duration change from the current cycle to the succeeding cycle is 5 seconds. The duration of one and only one phase in the next cycle is the current duration added or subtracted by 5 seconds. After choosing the phases' duration in the next cycle, the current duration becomes the chosen one. The traffic light can

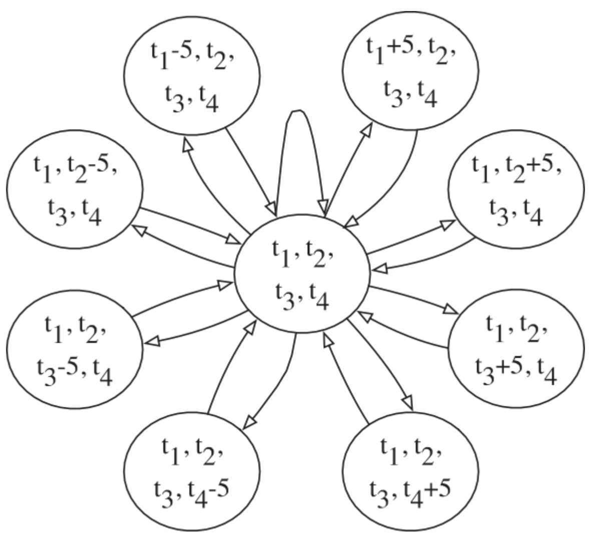


Fig. 3. Part of the Markov decision process in a multiple traffic lights scenario.

select an action in a similar way as the previous procedure. In addition, we set the max duration of a phase as 60 seconds and the minimal as 0 second.

The MDP is a flexible model. It can be applied into a more complex intersection with more traffic lights, such as an irregular intersection with five or six ways, which needs more phases. When there are more phases at an intersection, they can be added in the MDP model as a higher-dimension value. The dimension of the circle in the MDP is equal to the number of phases at the intersection.

The phases in a traffic light cyclically change in sequence. Yellow signal is required between two neighboring phases to guarantee safety, which allows running vehicles to stop before signals become red. The yellow signal duration T_{yellow} is defined by the maximum speed $v_{\max}$ on that road divided by the most commonly-seen decelerating acceleration a_{dec}.

$$T_{\text{yellow}} = \frac{v_{\max}}{a_{dec}}. \quad (3)$$

It means the running vehicle needs such a length of time to firmly stop in front of the intersection.

C. Rewards

The role of rewards is to provide feedback to a reinforcement learning model about the performance of the previous actions. It is important to define the reward appropriately so to correctly guide the learning process, which accordingly helps take the best action policy.

In our system, the main goal is to increase the efficiency of an intersection and reduce the waiting time of vehicles. Thus, we define the rewards as the change of the cumulative waiting time between two neighboring cycles. Let i_t denote the ith observed vehicle from the starting time to the starting time point of the tth cycle and N_t denote the corresponding total number of vehicles till the tth cycle. The waiting time of vehicle i till the tth cycle is denoted by $w_{i_t,t}, (1 \le i_t \le N_t)$. The reward in the tth cycle

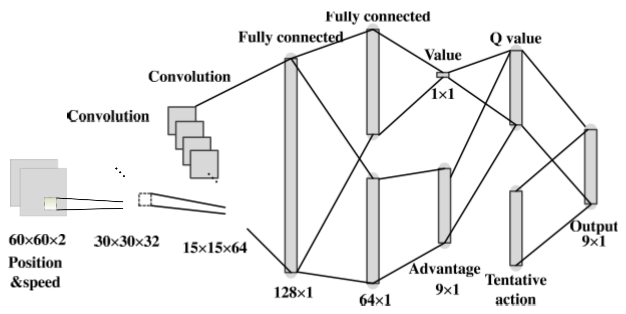


Fig. 4. The architecture of the deep convolutional neural network to approximate the Q value.

is defined by the following equation,

$$r_t = W_t - W_{t+1}, \quad (4)$$

where

$$W_t = \underset{i_t\,=\,1}{\overset{N_t}{}}\; w_{i_t,t}. \quad (5)$$

It means the reward is the increment in cumulative waiting time between before taking the action and after the action. If the reward in the current cycle becomes larger than before, the waiting time increases less than before. Considering the delay is non-decreasing with time, the overall reward is always negative. We aim to maximize the reward so to reduce the waiting time.

VI. Double Dueling Deep Q Network

In the traffic light control system in vehicular networks, the number of states are very large, and thus it is challenging to directly solve equation (2). In this paper we propose to use a Convolutional Neural Network (CNN) [11] to approximate the Q value. Combining with the state-of-the-art techniques, the proposed whole network is called Double Dueling Deep Q Network (3DQN).

A. Convolutional Neural Network

The architecture of the proposed CNN is shown in Fig. 4. It is composed of three convolutional layers and several fully-connected layers. In our system, the input is the small grids including the vehicles' position and speed information. The number of grids at an intersection is 60×60. The input data become 60×60×2 with both position and speed information. The data are first put through three convolutional layers. Each convolutional layer includes three parts, convolution, pooling and activation. The convolutional layer includes multiple filters. Every filter contains a set of weights, which aggregates local patches in the previous layer and shifts a fixed length of step defined by the stride each time. Different filters have different weights to generate different features in the next layer. The convolutional operation makes the presence of a pattern more important than the pattern's position. The pooling layer selects the salient values from a local patch of units to replace the whole patch. The pooling process removes less important information and reduces the dimensionality. The activation function is to decide how a unit is activated. The most common way is to apply a non-linear function on the output. In this paper, we employ the leaky ReLU [29] as the activation function with the following form (let x denote the output from a unit),

$$f(x) = \begin{cases} x, & \text{if } x > 0, \\ \beta x, & \text{if } x \leq 0. \end{cases} \tag{6}$$

β is a small constant to avoid zero gradient in the negative side. The leaky ReLU can converge faster than other activation functions, such as tanh and sigmoid, and prevent the generation of 'dead' neurons from regular ReLU.

In the architecture, three convolutional layers and full connection layers are constructed as follows. The first convolutional layer contains 32 filters. Each filter's size is 4×4 and it moves 2×2 stride every time through the full depth of the input data. The second convolutional layer has 64 filters. Each filter's size is 2×2 and it moves 2×2 stride every time. The size of the output after two convolutional layers is $15 \times 15 \times 64$. The third convolutional layer has 128 filters with the size of 2×2 and the stride's size is 1×1. The third convolutional layer's output is a $15 \times 15 \times 128$ tensor. A fully-connected layer transfers the tensor into a 128×1 matrix. After the fully-connected layer, the data are split into two parts with the same size 64×1. The first part is then used to calculate the value and the second part is for the advantage. The advantage of an action means how well it can achieve by taking an action over all the other actions. Because the number of possible actions in our system is 9 as shown in Fig. 3, the size of the advantage is 9×1. They are combined again to get the Q value, which is the architecture of the dueling Deep Q Network (DQN).

With the Q value corresponding to every action, we must highly penalize illegal actions, which may cause accidents or reach the max/min signal duration. The output combines the Q value and tentative actions to force the traffic light to take a legal action. Finally we get the Q values of every action in the output with penalized values. The parameters in the CNN is denoted by θ. $Q(s, a)$ now becomes $Q(s, a; \theta)$, which is estimated under the CNN θ. The details in the architecture are presented in the next subsections.

B. Dueling DQN

As mentioned before, our network contains a dueling DQN [15]. In the network, the Q value is estimated by the value at the current state and each action's advantage compared to other actions. The value of a state $V(s; \theta)$ denotes the overall expected rewards by taking probabilistic actions in the future steps. The advantage corresponds to every action, which is defined as $A(s, a; \theta)$. The Q value is the sum of the value V and the advantage function A, which is calculated by the following equation,

$$Q(s, a; \theta) = V(s; \theta) + \left(A(s, a; \theta) - \frac{1}{|A|} \sum_{a'} A(s, a'; \theta) \right). \tag{7}$$

$A(s, a; \theta)$ shows how important an action is to the value function among all actions. If the A value of an action is positive, it means the action shows a better performance in numerical rewards compared to the average performance of all possible actions; otherwise, if the value of an action is negative, it means the action's potential reward is less than the average. It has been shown that the subtraction from the mean of all advantage values can improve the stability of optimization compared to using the advantage value directly. The dueling architecture is shown to effectively improve the performance in reinforcement learning.

C. Target Network

To update the parameters in the neural network, a target value is defined to help guide the update process. Let $Q_{target}(s, a)$ denote the target Q value at the state s when taking action a. The neural network is updated by the Mean Square Error (MSE) in the following equation,

$$J = \sum_{s} P(s)[Q_{target}(s, a) - Q(s, a; \theta)]^2, \tag{8}$$

where $P(s)$ denotes the probability of state s in the training mini-batch. The MSE can be considered as a loss function to guide the updating process of the primary network. To provide stable update in each iteration, a separate target network θ^-, the same architecture as the primary neural network but different parameters, is usually employed to generate the target value. The calculation of the target Q value is presented in the double DQN part.

The parameters θ in the primary neural network are updated by back propagation with (8). θ^- is updated based on the θ in the following equation,

$$\theta^- = \alpha\theta^- + (1 - \alpha)\theta. \tag{9}$$

α is the update rate, which presents how much the newest parameters affect the components in the target network. A target network can help mitigate the over optimistic value estimation problem.

D. Double DQN

The target Q value is generated by the double Q-learning algorithm [16]. In the double DQN, the target network is to generate the target Q value and the action is generated from the primary network. The target Q value can be expressed in the following equation,

$$Q_{target}(s, a) = r + \gamma Q(s', \arg\max_{a'}(Q(s', a'; \theta)), \theta^-). \tag{10}$$

It is shown that the double DQN effectively mitigates the over estimations and improves the performance [16].

In addition, we also employ the ϵ-greedy algorithm to balance the exploration and exploitation in choosing actions. With the increasing steps of training process, the value of ϵ decreases gradually. We set a starting and ending values of ϵ and the number of steps to reach the ending value. The value of ϵ linearly decreases to the ending value. When ϵ reaches the ending value, it keeps the value in the following procedure.

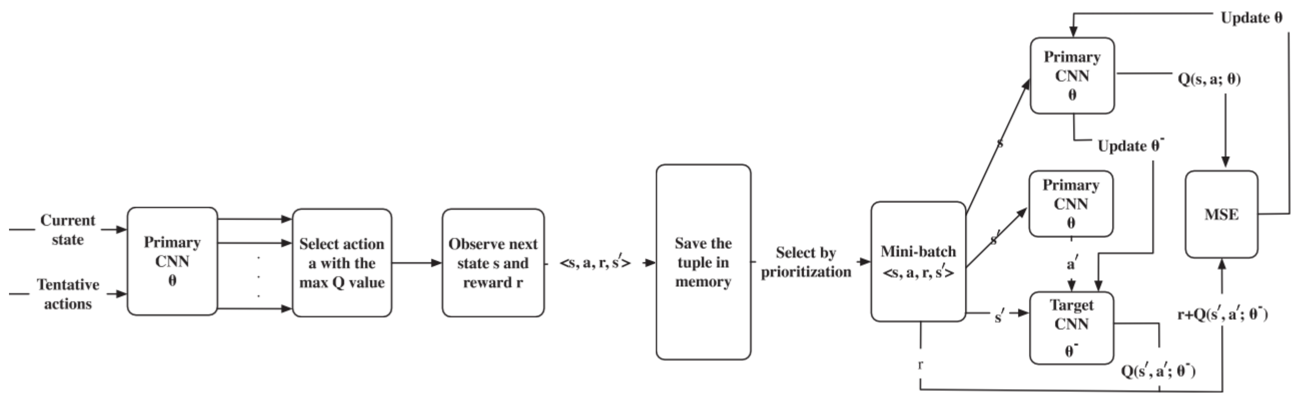


Fig. 5. The architecture of the reinforcement learning model in our system.

E. Prioritized Experience Replay

During the updating process, the gradients are updated through the experience replay strategy. A prioritized experience replay strategy chooses samples from the memory based on priorities, which can lead to faster learning and to better final policy [17]. The key idea is to increase the replay probability of the samples that have a high temporal difference error. There are two possible methods estimating the probability of an experience in a replay, proportional and rank-based. Rank-based prioritized experience replay can provide a more stable performance since it is not affected by some extreme large errors. In this system, we take the rank-based method to calculate the priority of an experience sample. The temporal difference error δ of an experience sample i is defined in the following equation,

$$\delta_i = |Q(s, a; \theta)_i - Q_{target}(s, a)_i|. \tag{11}$$

The experiences are ranked by the errors and then the priority p_i of experience i is the reciprocal of its rank. Finally, the probability of sampling the experience i is calculated in the following equation,

$$P_i = \frac{p_i^\tau}{\sum_k p_k^\tau}. \tag{12}$$

τ presents how much prioritization is used. When τ is 0, it is random sampling.

F. Optimization

In this paper, we optimize the neural networks by the ADAptive Moment estimation (Adam) [30]. The Adam is evaluated and compared with other back propagation optimization algorithms in [31], which concludes that the Adam attains satisfactory overall performance with a fast convergence and adaptive learning rate. The Adam optimization method adaptively updates the learning rate considering both first-order and second-order moments using the stochastic gradient descent procedure. Specifically, let $\boldsymbol{\theta}$ denote the parameters in the CNN and $J(\boldsymbol{\theta})$ denote the loss function. Adam first calculates the gradients of the parameters,

$$\mathbf{g} = \nabla_\theta J(\boldsymbol{\theta}). \tag{13}$$

It then respectively updates the first-order and second-order biased moments, $\mathbf{s}$ and $\mathbf{r}$, by the exponential moving average,

$$\mathbf{s} = \rho_s \mathbf{s} + (1 - \rho_s)\mathbf{g},$$
$$\mathbf{r} = \rho_r \mathbf{r} + (1 - \rho_r)\mathbf{g}, \tag{14}$$

where ρ_s and ρ_r are the exponential decay rates for the first-order and second-order moments, respectively. The first-order and second-order biased moments are corrected using the time step t through the following equations,

$$\hat{\mathbf{s}} = \frac{\mathbf{s}}{1 - \rho_s^t},$$
$$\hat{\mathbf{r}} = \frac{\mathbf{r}}{1 - \rho_r^t}. \tag{15}$$

Finally the parameters are updated as follows,

$$\boldsymbol{\theta} = \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$$
$$= \boldsymbol{\theta} + \left(-E_r \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}} + \delta}}\right), \tag{16}$$

where E_r is the initial learning rate and δ is a small positive constant to attain numerical stability.

G. Overall Architecture

Our deep learning architecture is illustrated in Fig. 5. The current state and the tentative actions are fed to the primary convolutional neural network to choose the most rewarding action. The current state and action along with the next state and received reward are stored into the memory as a four-tuple $\langle s, a, r, s' \rangle$. The data in the memory are selected by the prioritized experience replay to generate mini-batches and they are used to update the primary neural network's parameters. The target network θ^- is a separate neural network to increase stability during the learning. We use the double DQN [16] and dueling DQN [15] to reduce the possible overestimation and improve performance. Through this way, the approximating function can be trained and the Q value at every state to every action can be calculated. The optimal policy can then be obtained by choosing the action with the max Q value.

The pseudocode of our 3DQN with prioritized experience replay is shown in Algorithm 1. Its goal is to train a mature

Algorithm 1: Dueling Double Deep Q Network with Prioritized Experience Replay Algorithm on a Traffic Light.

```
Input: replay memory size $M$, minibatch size $B$, greedy $E$,
pre-train steps $tp$, target network update rate $\alpha$, discount
factor $\gamma$.
Notations:
$\theta$: the parameters in the primary neural network.
$\theta^-$: the parameters in the target neural network.
$m$: the replay memory.
$i$: step number.
Initialize parameters $\theta$, $\theta^-$ with random values.
Initialize $m$ to be empty and $i$ to be zero.
Initialize $s$ with the starting scenario at the intersection.
while there exists a state $s$ do
  Choose an action $a$ according to the $E$ greedy.
  Take action $a$ and observe reward $r$ and new state $s^t$.
  if the size of memory $m > M$ then
    Remove the oldest experiences in the memory.
  end if
  Add the four-tuple $\times s, a, r, s^t \ast$ into $M$.
  Assign $s^t$ to $s$: $s \leftarrow s^t$.
  $i \leftarrow i + 1$.
  if $|M| > B$ and $i > tp$ then
    Select $B$ samples from $m$ based on the sampling
    priorities.
    Calculate the loss $J$:
      $J = \sum_{s} \frac{1}{B} [r + \gamma Q(s^t, \arg\max_{a^t}(Q(s^t, a^t; \theta)), \theta^-) -$
        $Q(s, a; \theta)]^2$.
    Update $\theta$ with $\nabla J$ using Adam back propagation.
    Update $\theta^-$ with $\theta$:
      $\theta^- = \alpha\theta^- + (1 - \alpha)\theta$.
    Update every experience's sampling priority
    based on $\delta$.
    Update the value of $E$.
  end if
end while
```

adaptive traffic light, which can change its phases' duration based on different traffic scenarios. The agent first chooses actions randomly till the number of steps is over the pre-train steps and the memory has enough samples for at least one mini-batch. Before the training, every samples' priorities are the same. Thus, they are randomly selected into a mini-batch to train. After training once, the samples' priorities change and they are selected by different probabilities. The parameters in the neural network is updated by the Adam back propagation [31]. The agent chooses actions based on the E and the action that has the max Q value. The agent finally learns to get a high reward by reacting on different traffic scenarios.

VII. Evaluation

A. Evaluation Methodology and Parameters

1) Evaluation Metrics: Our model's objective is to maximize the defined reward, which is to reduce the cumulative delay of all vehicles. Thus, we evaluate the performance of our model using the following two metrics: cumulative reward and average waiting time. The cumulative reward is measured by adding up the rewards of all cycles in every episode within one hour period. The average waiting time is measured by dividing the total waiting time by the number of vehicles in an episode.

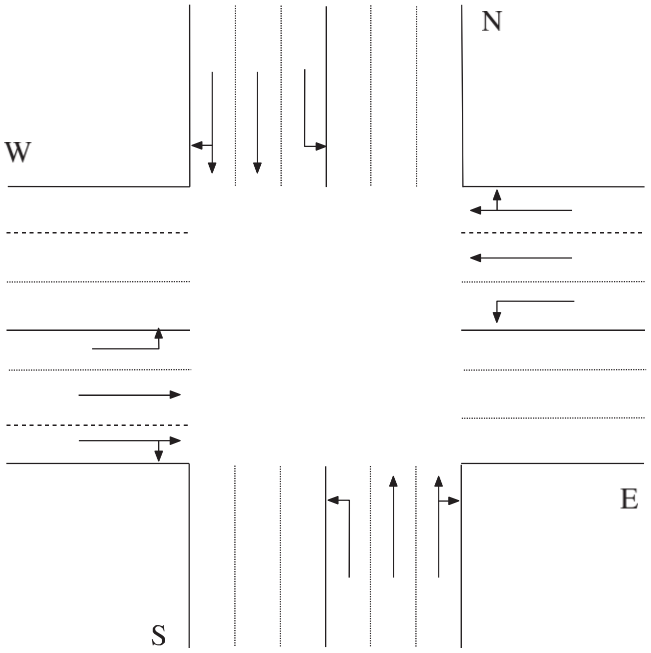


Fig. 6. The intersection scenario tested in our evaluation.

2) Traffic Parameters: The evaluation is conducted in SUMO [18], which provides real-time traffic simulation. We use the Python APIs provided by SUMO to obtain the intersection's information and to send orders to change the traffic light's timing. The intersection is composed of four perpendicular roads, as shown in Fig. 6. Each road has three lanes. The right-most lane allows right-turn and through traffic, the middle lane only allows through traffic, and the left inner lane allows only left-turn traffic. The simulated intersection is a 300 m $\times$ 300 m area. The grid length c is 5 meters, which means the total number of grids is 60×60. The lane length is 150 meters. Vehicles are 5 meters long and the minimal gap between two vehicles is 2 meters. Vehicles arrive at the intersection following a random process. The average vehicle arrival rate in every lane is 1/10 per second (i.e., on average, there is one vehicle arriving every 10 seconds). Two lanes allow for through traffic, so the flow rate of all through traffic (west-to-east, east-to-west, north-to-south, south-to-north) is 2/10 per second (i.e., on average, there are two vehicles every 10 seconds). Turning traffic (east-to-south, west-to-north, south-to-west, north-to-east) is 1/10 per second. Krauss following model [32] is used for Vehicles on the road, which guarantees safe driving. The max speed of a vehicle is 13.9 m/s (50 km/h). The max accelerating acceleration is 1.0 m/s^2 and the decelerating acceleration is 4.5 m/s^2. The duration of yellow signals T_{yellow} is set 4 to be seconds.

3) Model Parameters: The model is trained in iterations. One iteration is an episode in an hour. The reward is accumulated in an episode. The simulation results are the average of 50

TABLE II
PARAMETERS IN THE REINFORCEMENT LEARNING NETWORK

Parameter	Value
Replay memory size M	20000
Minibatch size B	64
Starting ϵ	1
Ending ϵ	0.01
Steps from starting ϵ to ending ϵ	10000
Pre-training steps tp	2000
Target network update rate α	0.001
Discount factor γ	0.99
Learning rate ϵ_r	0.0001
Leaky ReLU β	0.01

iterations. The development environment is built on the top of Tensorflow [33]. The parameters in the deep learning network are shown in Table II.

4) Comparison Study: The performance of our system is compared with three strategies: The first one is the simplest setting: the traffic light's duration is fixed. We set it to be 30 seconds and 40 seconds for every phase. The second one is a conventional method called Adaptive Traffic Signal Control (ATSC) [20], which set the traffic lights with fixed-time signals. The third one is a state-of-the-art method, Deep Q Network (DQN) [13]. In ATSC, the authors propose a Webster's method to estimate the optimal light time duration based on the most recent cycles' saturation. In DQN, the authors propose to use reinforcement learning with an auto-encoder. They use the queue length as the state to control traffic lights. The authors show that one single Q network can learn good control strategy in a two-phase intersection. Regarding to our own framework, we also conduct the ablation studies with different reinforcement learning architectures and different parameters to present our model's good performance.

B. Experimental Results

1) Cumulative Reward: The cumulative reward in every episode is first evaluated under the same traffic flow rate from every lane. We make all strategies have the same rewards as our work. Note that we aim to maximize the the rewards, which is to minimize the cumulative waiting time, represented as a negative number. The simulation results are shown in Fig. 7. From this figure, we can see that our 3DQN outperforms the other strategies. Specifically, the cumulative reward in 3DQN is greater than -50000 (note that the reward is negative since the vehicles' delay is positive) while that in the two fixed-time strategies is less than -6000. The fixed-time traffic signals always obtains a low reward even after more iterations while our model can learn to achieve a higher reward with more iterations. This is because the fixed-time traffic signals do not change the signals' time under different traffic scenarios. DQN's performance is very unstable, which cannot accurately capture the whole information in a complex intersection by a deep neural

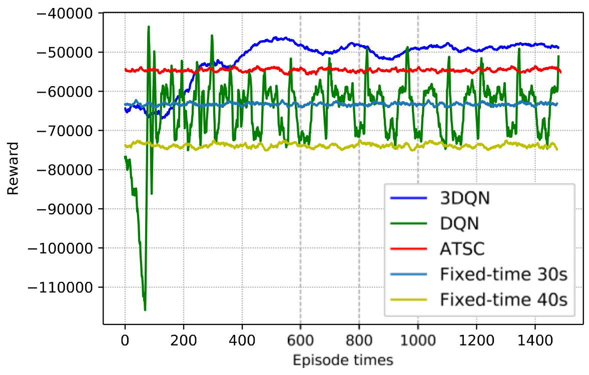


Fig. 7. The cumulative reward during all the training episodes.

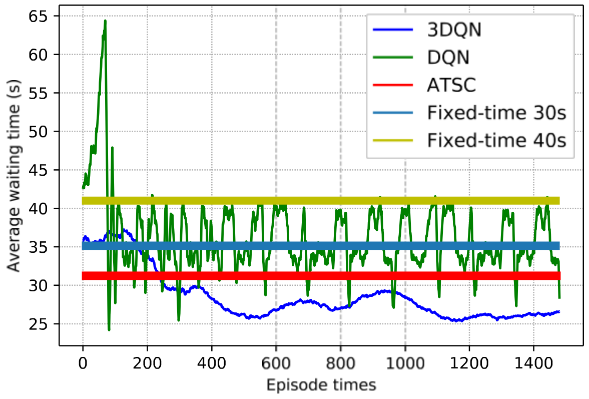


Fig. 8. The average waiting time during all the training episodes.

network with the queue length only. Because the normal traffic scenario is much more complex than a simple 2-phase intersection, the traffic information represented by queue length is inaccurate, which makes DQN choose false actions when two traffic scenarios are different but the queue length is the same. In addition, one network in DQN is easy to overfit the training data. ATSC only chooses the phases' time duration based on several previous cycles, which is inaccurate to predict the future traffic scenarios. In 3DQN, the signals' time changes to achieve the best expected rewards, which learns a more general strategy to handle different traffic states. When the training process iterates over 1000 times, the cumulative rewards become more stable than previous iterations. It means 3DQN has learnt how to handle different traffic scenarios to get the most rewards after 1000 iterations.

2) Average Waiting Time: We calculate the average waiting time of vehicles in every episode, which is shown in Fig. 8. From this figure, we can see that 3DQN outperforms the other four strategies. Specifically, the average waiting time in the fixed-time signals is always over 35 seconds. Our model can learn to reduce the waiting time to about 26 seconds after 1200 iterations from over 35 seconds, which is at least 25.7% less than the fixed-time strategies. ATSC can get better performance than the fixed-time strategy, but it only uses several most recent cycles' information, which cannot well represent future traffic. DQN's performance is very unstable, which means one neural network with the queue length cannot accurately capture the real traffic

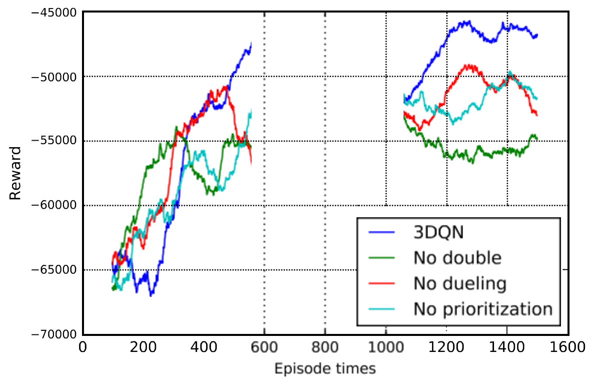


Fig. 9. The cumulative reward during all the training episodes in different network architecture.

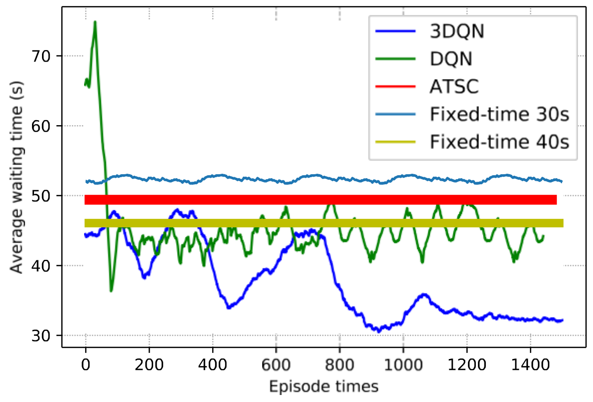


Fig. 10. The average waiting time in all the training episodes during the rush hours with unbalanced traffic from all lanes.

information. The results show that our model can obtain the most stable and best performance in vehicles' average waiting time among all the methods.

3) Ablation Studies: In this part, we evaluate our model by comparing to others with different parameters and different architectures. In our model, we use a series of techniques to improve the performance of deep Q networks. For comparison, we remove one of these techniques each time to see how every technique influences the performance. The techniques include double network, dueling network and prioritized experience replay. The reward changes in all methods are shown in Fig. 9. We can see that our model can learn fastest among the four models. It means our model reaches the best policy faster than others. Specifically, even there is some fluctuation in the first 400 iterations, our model still outperforms the other three after 500 iterations. Our model can achieve greater than −47000 rewards while the others have less than −50000 rewards.

4) Average Waiting Time Under Rush Hours: In this part, we evaluate our model by comparing the performance under the rush hours. The rush hour means the traffic flows from all lanes are not the same, which is usually seen in the real world. During the rush hours, the traffic flow rate from one direction doubles, and the traffic flow rates in the other lanes keep the same as normal hours. Specifically, in our experiments, the arrival rate of vehicles on the lanes from the west to east becomes $2/10$ per second and the arrival rates of vehicles on the other lanes are still $1/10$ per second. The experimental results are shown in Fig. 10. From the figure, we can see that the best policy becomes harder to be learnt than the previous scenario. This is because the traffic scenario becomes more complex, which contains more uncertain factors. But after trial and error, our model can still learn a good policy to reduce the average waiting time. Specifically, the average waiting time in 3DQN is about 33 seconds after 1000 iterations while the average waiting time in the other two fixed-time methods is over 45 seconds. Our model reduces about 26.7% of the average waiting than the fixed-time methods. ATSC can achieve better results than one fixed-time method and worse than the other because the optimal phases' time duration in the most recent cycles does not work in the future traffic considering the traffic scenario becomes very complex. DQN's performance becomes more unstable than that in the previous scenario. In summary, 3DQN can achieve the best performance under the rush hours.

VIII. Conclusion

In this paper, we propose to solve the traffic light cycle control problem using a deep reinforcement learning model. The traffic information can be gathered from vehicular networks or sensors and used as the input of our model. The states are two-dimension values with vehicles' position and speed information. The actions are modeled as a Markov decision process and the rewards are the cumulative waiting time difference between two cycles. To handle the complex traffic scenario in our problem, we propose a double dueling deep Q network (3DQN) with prioritized experience replay. The model can learn a good policy under both rush hours and normal traffic flow. It can reduce the average waiting timing by over 20% from the start of the training. The proposed model also outperforms other comparison peers in learning speed and other metrics.

References

[1] S. S. Mousavi, M. Schukat, P. Corcoran, and E. Howley, "Traffic light control using deep policy-gradient and value-function based reinforcement learning," *IET Intell. Transp. Syst.*, vol. 11, no. 7, pp. 417–423, Sep. 2017.

[2] X. Liang, T. Yan, J. Lee, and G. Wang, "A distributed intersection management protocol for safety, efficiency, and driver's comfort," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1924–1935, Jun. 2018.

[3] N. Casas, "Deep deterministic policy gradient for urban traffic light control," unpublished paper, 2017. [Online]. Available: https://arxiv.org/abs/1703.09035v1

[4] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Design of reinforcement learning parameters for seamless application of adaptive traffic signal control," *J. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 227–245, Jul. 2014.

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, no. 1. Cambridge, MA, USA: MIT Press, Mar. 1998.

[6] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[7] D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

[8] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Holonic multi-agent system for traffic signals control," *Eng. Appl. Artif. Intell.*, vol. 26, no. 5, pp. 1575–1587, May/Jun. 2013.

[9] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," unpublished paper, 2016. [Online]. Available: https://arxiv.org/abs/1611.01142

[10] H. Hartenstein and L. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Commun. Mag.*, vol. 46, no. 6, pp. 164–171, Jun. 2008.

[11] X. Liang and G. Wang, "A convolutional neural network for transportation mode detection based on smartphone platform," in *Proc. IEEE 14th Int. Conf. Mobile Ad Hoc Sensor Syst.*, Oct. 2017, pp. 338–342.

[12] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[13] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA J. Automatica Sinica*, vol. 3, no. 3, pp. 247–254, Jul. 2016.

[14] E. van der Pol, "Deep reinforcement learning for coordination in traffic light control," Master's thesis, Dept. Artif. Intell., Univ. Amsterdam, Amsterdam, The Netherlands, Aug. 2016.

[15] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. 33rd Int. Conf. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

[16] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, Feb. 2016, pp. 2094–2100.

[17] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. 4th Int. Conf. Learn. Representations*, May2016.

[18] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of sumo-simulation of urban mobility," *Int. J. Adv. Syst. Meas.*, vol. 5, no. 3/4, pp. 128–138, Dec. 2012.

[19] S. Chiu and S. Chand, "Adaptive traffic signal control using fuzzy logic," in *Proc. 1st IEEE Regional Conf. Aerosp. Control Syst.*, Apr. 1993, pp. 1371–1376.

[20] K. Pandit, D. Ghosal, H. M. Zhang, and C.-N. Chuah, "Adaptive traffic signal control with vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 4, pp. 1459–1471, May 2013.

[21] Z. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surveys Tut.*, vol. 19, no. 4, pp. 2432–2455, May 2017.

[22] W. Liu, G. Qin, Y. He, and F. Jiang, "Distributed cooperative reinforcement learning-based traffic signal control that integrates v2x networks' dynamic clustering," *IEEE Trans. Veh. Technol.*, vol. 66, no. 10, pp. 8667–8681, Oct. 2017.

[23] I. Arel, C. Liu, T. Urbanik, and A. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intell. Transp. Syst.*, vol. 4, no. 2, pp. 128–135, Jun. 2010.

[24] P. Balaji, X. German, and D. Srinivasan, "Urban traffic signal control using reinforcement learning agents," *IET Intell. Transp. Syst.*, vol. 4, no. 3, pp. 177–188, Sep. 2010.

[25] L. Prashanth and S. Bhatnagar, "Threshold tuning using stochastic optimization for graded signal control," *IEEE Trans. Veh. Technol.*, vol. 61, no. 9, pp. 3865–3880, Nov. 2012.

[26] 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," unpublished paper, 2017. [Online]. Available: https://arxiv.org/abs/1705.02755

[27] L. Zhu, Y. He, F. R. Yu, B. Ning, T. Tang, and N. Zhao, "Communication-based train control system performance optimization using deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10705–10717, Dec. 2017.

[28] F. Tang, *et al.*, "On removing routing protocol from future wireless networks: A real-time deep learning approach for intelligent traffic control," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 154–160, Feb. 2018.

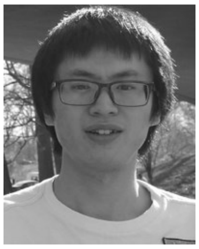
[29] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vision*, Dec. 2015, pp. 1026–1034.

[30] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Representations*, May 2015.

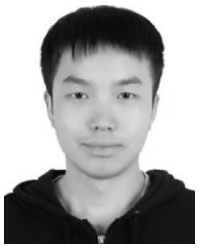
[31] S. Ruder, "An overview of gradient descent optimization algorithms," unpublished paper, 2016. [Online]. Available: https://arxiv.org/abs/1609.04747v1

[32] S. Krauß, "Towards a unified view of microscopic traffic flow theories," *IFAC Transp. Syst.*, vol. 30, no. 8, pp. 901–905, Jun. 1997.

[33] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," in *Proc. 12th USENIX Conf. Oper. Syst. Des. Implementation*, Nov. 2016, pp. 265–283.

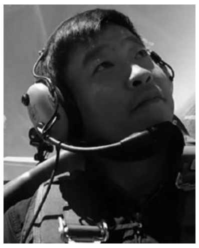


Xiaoyuan Liang received the B.S. degree in computer science and technology from Harbin Institute of Technology, Harbin, China, in June 2013. He is currently working toward the Ph.D. degree at the Computer Science Department, New Jersey Institute of Technology, Newark, NJ, USA. His research interests include deep learning, data mining, and vehicular networks.



Xunsheng Du (S'17) received the B.S. degree from Huazhong University of Science and Technology, Wuhan, China, in June 2015. He is currently working toward the Ph.D. degree in electronic and computer engineering at the University of Houston, Houston, TX, USA. He has been a Reviewer for IEEE SIGNAL PROCESSING MAGAZINE, IEEE International Conference on Communications, IEEE Special issue on Artificial Intelligence and Machine Learning for Networking and Communications (AI4NET), IEEE Transactions on Network Science and Engineering (TNSE), and IEEE INTERNET OF THINGS JOURNAL. His research interests include wireless communication, vehicular networks, image processing, deep learning, and reinforcement learning.

Guiling Wang received the B.S. degree in software from Nankai University, Tianjin, China, and the Ph.D. degree in computer science and engineering and a minor in statistics from The Pennsylvania State University, State College, PA, USA, in May 2006. She is currently a Professor with the Yingwu College of Computing Sciences. She also holds a joint appointment at the Martin Tuchman School of Management. She joined New Jersey Institute of Technology in July 2006 as an Assistant Professor and was promoted to an Associate Professor with tenure in June 2011. She was promoted to a Full Professor in June 2016 in her 30s. Her research interests include deep learning applications, blockchain technologies, intelligent transportation, and mobile computing.



Zhu Han (S'01–M'04–SM'09–F'14) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1997, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Maryland, College Park, MD, USA, in 1999 and 2003, respectively. From 2000 to 2002, he was an R&D Engineer with JDSU, Germantown, MD, USA. From 2003 to 2006, he was a Research Associate with the University of Maryland. From 2006 to 2008, he was an Assistant Professor with Boise State University, Boise, ID, USA. His research interests include wireless resource allocation and management, wireless communications and networking, game theory, big data analysis, security, and smart grid. He is the recipient of an NSF Career Award in 2010, the Fred W. Ellersick Prize of the IEEE Communication Society in 2011, the EURASIP Best Paper Award for the Journal on Advances in Signal Processing in 2015, IEEE Leonard G. Abraham Prize in the field of Communications Systems (Best Paper Award in IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS) in 2016, and several Best Paper Awards in IEEE conferences. He is currently an IEEE Communications Society Distinguished Lecturer.