Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/43/20230804

Traffic light control with reinforcement learning

Taoyu Pan

Cambridge A Level Centre, Hangzhou Foreign Language School, Hangzhou, 310000, China

pty060110@gmail.com

Abstract. Urban traffic signal optimization is important for alleviating congestion in urban transportation systems. This study proposes a real-time traffic light control algorithm based on deep Q learning with a reward function that accounts for queue lengths, delays, travel times, and throughput. The model dynamically decides phase changes based on current traffic conditions. The training of the deep Q network involves an offline stage from pre-generated data with fixed signal timing and an online stage using real-time traffic data. A deep Q network structure with a "phase gate" component is used to simplify the model's learning task under different phases. A "memory palace" mechanism is used to address sample imbalance during the training process. Both synthetic and real-world traffic flow data are used to validate our approach under an urban road intersection scenario in Hangzhou, China. Results demonstrate significant performance improvements of the proposed method in reducing vehicle waiting time (57.1% to 100%), queue lengths (40.9% to 100%), and total travel time (16.8% to 68.0%) compared to traditional fixed signal timing plans.

Keywords: Traffic Light Control, Reinforcement Learning, Deep Q Learning, Traffic Simulation.

1. Introduction

With the rapid process of urbanization, car ownership has been constantly increasing, especially in China. According to the statistics of the Ministry of Public Security of China [1], the number of motor vehicles owned nationwide reached 417 million in 2022. The rapid increase in car ownership causes great traffic congestion, resulting in a waste of time for drivers and commuters.

Traffic light control is crucial for reducing congestion in urban transportation systems. However, traditional traffic control methods typically rely on pre-defined timing schedules and fixed signal patterns to regulate the traffic flow at intersections [2]. While traditional methods have been effective in regulating traffic flow to a certain extent, the limitations of their fixed schedules make them inadequate for addressing the complex and dynamic nature of complicated traffic systems. For example, during peak traffic hours, the green light of the major traffic flow direction may be too short to allow enough vehicles to pass through the intersection, causing long queues and delays. Additionally, traditional methods may not take into account the impact of unexpected events such as accidents, road closures, and sudden increase or decrease in traffic flows, which further exacerbate congestion.

Recent developments in deep reinforcement learning (RL) have shown great promise in improving traffic light control by enabling real-time adaptation to dynamic traffic conditions [3]. RL allows the traffic light to learn from its environment by observing traffic patterns and adjusting its control strategy accordingly. The traffic light is trained using a reward-based system, where the goal is to maximize the traffic flow through the intersection while minimizing delays and reducing congestion. This adaptive approach to traffic control has shown great potential to improve traffic flows and reduce travel time.

This paper proposes a deep Q learning-based RL method for real-time traffic light control. Following the work of Wei et al. [4], the proposed method considers a reward function that takes into account queue lengths, delays, travel time, and throughput. At each time step, the model decides whether to change the phase or not, providing an adaptive solution to dynamic traffic conditions. The training process is divided into two stages: offline and online. The offline training uses pre-generated traffic flow data with fixed time schedules to obtain a good prior for model parameters, while the online training leverages real-time traffic flow data for further adaptive learning of the model. To better capture the dynamics of different traffic light phases, a well-designed deep Q network structure with a "phase gate" component is employed. Additionally, to address the sample imbalance issue in the experience replay of the deep Q learning, a "memory palace" mechanism is used to ensure sufficient sampling of rarely appeared state-action combinations. The proposed approach is validated using both synthetic and real-world traffic flow data, with a road intersection in Hangzhou, China serving as the case study. Results show that our method outperforms traditional fixed signal plan traffic light control in terms of reducing vehicle waiting time, queue lengths, and total travel time in different traffic flow scenarios.

The paper is structured as follows: Section 2 presents the literature review. In Section 3, we provide an overview of our methodology, covering preliminaries, problem formulation, and the proposed RL framework. Section 4 applies the proposed method to a real-world road intersection in Hangzhou as a case study. Section 5 concludes the study and summarizes the key findings.

2. Literature review

2.1. Traditional traffic light control

Traditional traffic light control methods have been widely used for decades and are still prevalent in many cities around the world. These methods are typically based on fixed-time schedules or traffic-responsive strategies that adjust signal timings based on traffic volume or occupancy. Fixed-time schedules use pre-determined signal timings that are set according to historical traffic patterns or the peak traffic volume of a specific area. For example, the fixed-time schedules are set using historical traffic demand to determine the time for each phase [2, 5, 6]. The traffic-responsive strategies use updated time information that is set according to real-world traffic data. For example, Porche and Lafortune [7] and Cools et al. [8] implement self-organizing traffic lights using real-time traffic data. These methods can deal with highly random traffic conditions.

One of the main drawbacks of fixed-time schedules is that they cannot adapt to changing traffic conditions, such as fluctuations in traffic volume or unexpected events. This often results in inefficient traffic flow, with long queues and delays and wasted time and fuel consumption. The problem with traffic-responsive strategies is that they are dependent on man-made rules for current traffic patterns, but do not consider the subsequent traffic conditions. In this way, it is unable to optimally adapt to the stochastic traffic flow dynamics.

2.2. Applications of reinforcement learning

Reinforcement learning (RL) is a field within machine learning that investigates how intelligent agents should make decisions in an environment to maximize their cumulative reward. With the success of Alpha Go [9], RL has gained considerable attention in many fields, including energy [10], civil engineering [11], network system [12], finance [13], logistics [14], and transportation [15]. For a brief survey with recent advances in reinforcement learning, please refer to Arulkumaran et al. [16].

2.3. Reinforcement learning-based traffic light control

Since traditional traffic light methods are not able to solve the multi-direction dynamic traffic light control problems comprehensively, there are more and more attempts using RL to deal with the problem [17, 18]. Traditional RL algorithms designed the state as discrete values of traffic conditions, such as the location of vehicles and the number of cars on the road [19–23]. However, the corresponding paired action-state matrix has a large space demand for storage. In this way, if larger state space problems are considered, the method will not work.

To solve the problem to consider a larger state space, the algorithm of Deep Q learning is applied to take continuous variables into account [3]. Deep Q learning sets up a deep neural network (DNN) to learn the Q function of RL from the traffic state inputs and the corresponding traffic system performance output. In this way, the state and action are associated with reward. The state design considers queue length [3] and average delay [24], etc. The reward design also takes those variables into account. Nevertheless, these methods assume relatively static traffic environments, and hence far from real-world traffic conditions.

Recently, Wei et al. [4] proposed an RL-based traffic light control model. The algorithm is tested with real-world traffic settings. In this paper, we follow the framework of Wei et al. [4], but use the real-world traffic flow data from a road intersection in Hangzhou, China, which none of the existing studies has considered.

3. Methodology

3.1. Preliminaries

3.1.1. Introduction to reinforcement learning RL stands as a type of machine learning method dedicated to making real-time decisions with the overarching aim of maximizing a predefined long-term objective function [25]. At its core, RL models are structured upon the framework of Markovian Decision Processes (MDP) [26], encompassing the following key components:

- **State Space** $\mathcal{S}$: A state within the RL context represents a collection of variables employed to characterize the current environment. The entire spectrum of feasible states constitutes the state space $\mathcal{S}$, typically having a finite nature. In the MDP framework, a state at time t, denoted as $s_t \in \mathcal{S}$, is pivotal in determining the current action a_t.
- **Action Space** $\mathcal{A}$: The action space $\mathcal{A}$ is defined as the set comprising all potential actions an agent can undertake within an RL model. Given state s_t at time t, the agent's choice of action is dictated by a policy $\pi(\cdot)$ (i.e., $a_t = \pi(s_t) \in \mathcal{A}$). The objective is to ensure that this action selection maximizes the expected long-term system reward.
- **Transition Probability** $\mathbb{P}$: The transition probability is defined for an MDP model. Given a state s_t and action a_t at time t, the transition probability elucidates the distribution governing the system's subsequent state, i.e., $\mathbb{P}(s_{t+1} \mid s_t, a_t), \forall s_{t+1} \in \mathcal{S}$.
- **Reward** R: At each time step t, reward $R_t \in \mathbb{R}$ represents the intermediate reward of the system. People usually need to define the reward function $R_t(a_t, s_t)$ in a proper form so that the agents work as expected. With the definition of R_t, the long-term reward is defined as the total discounted reward starting from the time step t up to time $t + n$ (where n can be infinity):

$$G_t = R_t + \gamma R_{t+1} + ... + \gamma^n R_{t+n} = \sum_{k=0}^{n} \gamma^k R_{t+k} \tag{1}$$

  where γ is the discount factor (range from 0 to 1) to balance the importance between current reward and future reward.

The agent's primary objective lies in the determination of a policy π, a set of guidelines for action selection based on states, to maximize the total discounted reward. To achieve this goal, we employ the Q learning algorithm, which is a model-free, value-based, and off-policy reinforcement learning technique [27]. The Q value function given policy π is defined as follows:

$$Q^\pi(s, a) = \mathbb{E}[G_t \mid s, a, \pi] \tag{2}$$

Denote $Q^*(s, a)$ as the optimal action-value function that yields the maximum expected return given current state s and action a. Consequently, determining the optimal policy π^* involves selecting the action a that maximizes the Q value for the given state s:

$$a^* = \underset{a\in\mathcal{A}}{\arg\max}\, Q^*(s,a) \tag{3}$$

The optimal Q value function is characterized by the Bellman equation:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'\in\mathcal{S}} \mathbb{P}(s' \mid s,a)\cdot Q^*(s',a) \tag{4}$$

For an MDP model, $Q^*(s,a)$ can be learned through an iterative update process, called the Q learning algorithm. Specifically, given the current state s_t and action a_t at time t, suppose the system changes to s_{t+1}, the Q value for the current state-action pair, $Q_t(s_t,a_t)$, is updated by adding a fraction α of the temporal difference (TD) error:

$$Q_{t+1}(s_t,a_t) = Q_t(s_t,a_t) + \alpha\cdot \underbrace{(R_t + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_t(s_{t+1},a_{t+1}) - Q_t(s_t,a_t))}_{\text{TD error}} \tag{5}$$

where $0 < \alpha < 1$ is the learning rate. R_t is the intermediate reward at time t, and γ is the discount factor.

3.1.2. Deep reinforcement learning The conventional Q learning algorithm relies on tabular storage for state-action pair values [27], posing challenges when dealing with high-dimensional or continuous state spaces. To surmount this limitation, deep neural networks offer a viable alternative for approximating the Q value function. These networks excel at handling high-dimensional and continuous state spaces, adeptly capturing intricate state features essential for approximating Q values. The Deep Q Network (DQN) [9] serves as the standard deep reinforcement learning algorithm, comprising an input layer that receives states, a series of hidden layers responsible for feature extraction and transformation, and an output layer dedicated to approximating Q values for state-action pairs.

Throughout the learning process, the DQN employs a neural network parameterized by weights $\boldsymbol{\theta}$ alongside an experience replay memory for storing past encounters. At each time step, a new experience denoted as $e_t = (s_t, a_t, R_t, s_{t+1})$, is added to the memory. To update the Q network effectively, a mini-batch of experiences is randomly sampled from the experience replay memory, facilitating Q learning updates with the utilization of target values y_t as depicted by Eq. 6:

$$y_t = R_t + \gamma \max_{a_{t+1}\in\mathcal{A}} Q_t(s_{t+1},a_{t+1};\boldsymbol{\theta}_t) \tag{6}$$

Here, $\boldsymbol{\theta}_t$ represents the weights of the DQN at time step t. The learning process strives to minimize the loss of function:

$$Loss_t(\boldsymbol{\theta}) = \mathbb{E}\Big[\big(y_t - Q_t(s_t,a_t;\boldsymbol{\theta}_t)\big)^2\Big] \tag{7}$$

The rationale underlying Eq. 7 lies in the aspiration to update the DQN in a manner that the approximated $Q(s,a;\boldsymbol{\theta})$ aligns closely with the Bellman equation (Eq. 4).

3.2. Problem definition

Figure 1 provides an overview of the problem addressed in this study. The environment is a traffic signal intersection, and the deep RL agent receives a state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, and receives a reward R_t from the environment. The goal of the agent is to determine the optimal action a_t for each state s_t, to minimize the average pre-defined cumulative discounted return. To maintain effective decision-making over time in the dynamic traffic signal intersection environment, the agent must continuously learn and

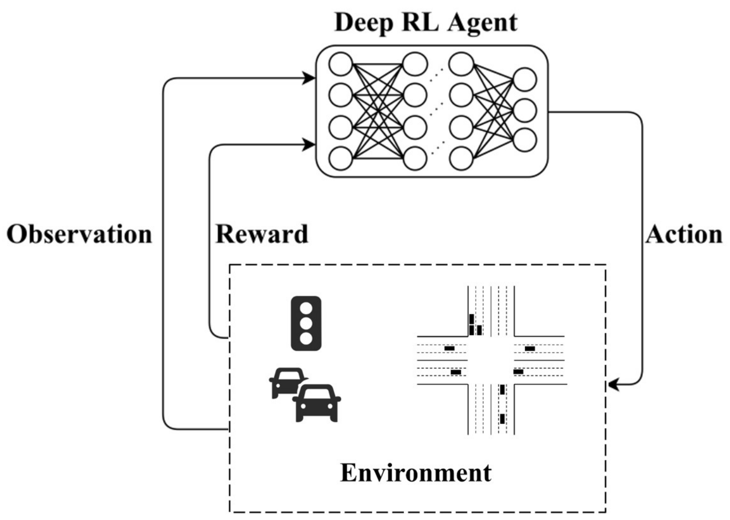


Figure 1: Problem definition

improve its policy. We elaborate on each of the key components of the problem definition in the following.

Phase. In this study, a traffic signal's phase is defined as a specification of signal color for each direction. Two phases are considered: NS and WE, where NS represents green for north and south directions and red for east and west directions and WE represents green for east and west directions and red for north and south directions (see Figure 2). Yellow lights are ignored in this study as they have a fixed time length and can be attached to the end of each phase. It is worth noting that in this study, we only consider two phases (instead of four where there are dedicated left-turn phases) for simplicity. And all left-turn vehicles will yield to straight vehicles when crossing the intersection.

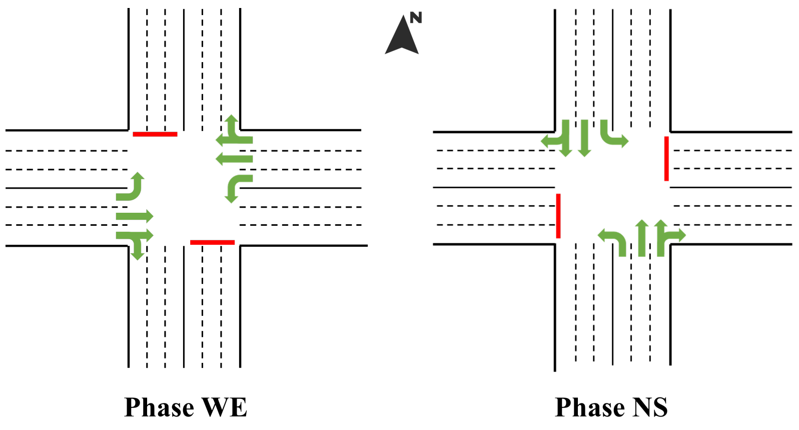


Figure 2: Explanation of phases

Environment. The intersection environment in this study consists of four directions: East (E), West (W), North (N), and South (S). Each direction has a specific lane layout (e.g., three lanes as shown in Figure 2). The environment also includes pre-defined vehicle movements for each phase, such as straight movements for East-West and left turns for East-West in the WE phase.

Agent. The agent plays a crucial role in this study. It observes the state of the environment (traffic signal intersection), selects an action based on its policy, and receives an immediate reward at each time step. Fig. 1 illustrates the general structure of the interaction between the agent and the environment

3.3. Model design

In this section, we describe the specific design for the model, including state, action, and reward functions.

3.3.1. State description The state is defined as a snapshot of the environment (i.e., intersection). Denote $\mathcal{I}$ as the set of all lanes for four directions. The state at time t includes:

- Queue length: $q_t = (q_{i,t})_{i \in \mathcal{I}}$, where $q_{i,t}$ indicates the queue length for lane i at time t.
- Number of vehicles: $v_t = (v_{i,t})_{i \in \mathcal{I}}$, where $v_{i,t}$ is the number of vehicles lane i at time t.
- Total waiting time: $w_t = (w_{i,t})_{i \in \mathcal{I}}$, where $w_{i,t}$ is the total waiting time (i.e., from the most recent vehicle stop up to time t) of all vehicles in lane i at time t.
- Phase: (P_t, P_{t+1}), where P_t is the current phase and P_{t+1} is the next phase.

Besides the numerical environmental information above, the state also includes vehicles' positions which are represented by an image. As shown in Figure 3, vehicle locations at time t are mapped to an image with grids (denoted as M_t). In each pixel of the image, value 1 represents there is a vehicle in that grid, otherwise 0. The image is encoded by a convolutional neural network (CNN) to get a latent vector l_t:

$$l_t = \text{CNN}(M_t;\ \boldsymbol{\theta}_t^{\text{CNN}}) \tag{8}$$

where $\boldsymbol{\theta}_t^{\text{CNN}}$ is the network weights of the CNN at time t.

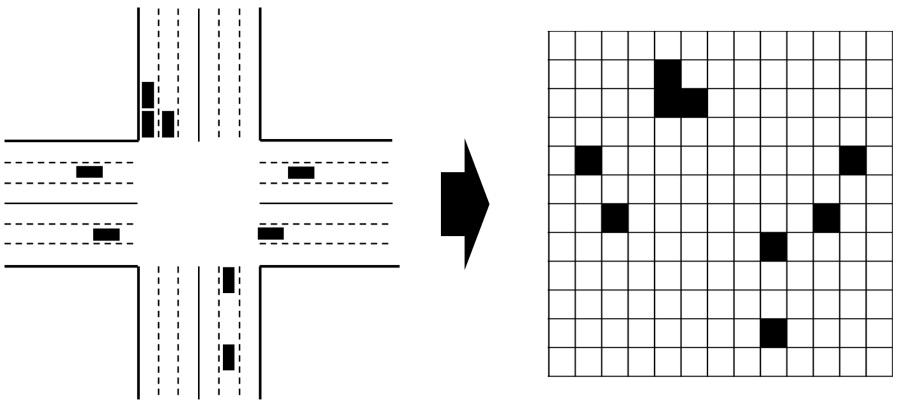


Figure 3: Representation of vehicle's position

Therefore, the final state of the model is

$$s_t = \text{Concat}(q_t, v_t, w_t, P_t, P_{t+1}, l_t) \tag{9}$$

where Concat$(\cdot)$ represents concatenate of all vectors.

3.3.2. Action set There are two actions considered in this study: 1) change to the next phase and 2) keep the current phase. That is,

$$\mathcal{A} = \{\text{Change to next phase, Keep the current phase}\} \tag{10}$$

With this action set, the agent can use current traffic conditions to dynamically determine the cycle length at each time t. Note that it does not make sense to switch phases frequently. Therefore, there will be a cost imposed for taking the action of "change to the next phase".

3.3.3. Reward function Reward is the key component of this study. Recall that the objective of the paper is to maximize the flow of traffic through the intersection while minimizing delays and reducing congestion, which is typically used in many transportation-related studies [28–30]. To accomplish this goal, the reward function at time t is formulated as a weighted sum of multiple factors, including:

- Total delays for lane i at time t (denoted as $d_{i,t}$):

$$d_{i,t} = 1 - \frac{\text{ALS}_{i,t}}{\text{SL}_i} \tag{11}$$

  where $\text{ALS}_{i,t}$ is the average lane speed of all vehicles at lane i and time t. SL_i is the pre-determined lane speed limit for lane i.
- Total waiting time for lane i (denoted as $w_{i,t}$): The waiting time for vehicle j in lane i at time t (denoted as $w_{i,t}^{(j)}$) is defined as the time from its most recent stop (speed < 0.1 m/s) to time t. Hence, $w_{i,t}^{(j)}$ will be recounted every time the vehicle moves (i.e., speed ≥ 0.1 m/s). Then the total waiting time for lane i is $w_{i,t} = \sum_j w_{i,t}^{(j)}$.
- Total queue length of lane i at time t (denoted as $q_{i,t}$): number of vehicles of speed equal to zero at lane i time t
- Change of phase (denoted as C_t): $C_t = 1$ if the a_t is changing to the next phase.
- Total number of vehicles that traversed the intersection within time interval t (denoted as V_t).
- Total travel time of all vehicles that traversed the intersection within time interval t (denoted as T_t).

Given the definition of these factors, the reward function at time t is defined as:

$$R_t = \beta_1 \sum_{i \in \mathcal{I}} d_{i,t} + \beta_2 \sum_{i \in \mathcal{I}} w_{i,t} + \beta_3 \sum_{i \in \mathcal{I}} q_{i,t} + \beta_4 C_t + \beta_5 V_t + \beta_6 T_t \tag{12}$$

where $\beta_1, ..., \beta_6$ are weights determining the importance of each factor in the reward.

3.4. Deep Q Network structure

The structure of the deep Q network is shown in Figure 4. All state information (see Eq. 9 for details) is concatenated as an input vector before being put into the deep Q network.

In the real-world scenario, traffic conditions can be very different under different phases. For instance, when the system is in phase NS, more traffic on the WE direction should make the light tend to change. However, when the system is in phase WE, the traffic in the WE direction should make the light tend to keep. This means that the traffic on WE directions has two different roles under different phases. The agent should be able to intelligently differentiate this. Simply adding phase information into the state may not be enough. In this research, we employ a deep Q network architecture that explicitly accounts for various scenarios, referred to as "phase gate"

As illustrated in Figure 4, the concatenated features of state information are fed into multiple fully connected (FC) layers. The role of these layers is to map from traffic conditions to potential Q values. It is worth noting that separate FC layers (red square and blue square in Figure 4) are employed for each phase to enable a distinct learning process. A phase selector gate is used to determine which branch of the FC layers to activate based on the current phase P_t. When P_t = NS, the NS phase selector is set to 1 while the WE phase selector is set to 0, activating the NS branch. Similarly, when P_t = WE, the WE branch is activated. This approach ensures that the decision-making process is tailored to the specific phase, avoids bias towards certain actions, and improves the network's fitting capability [4].

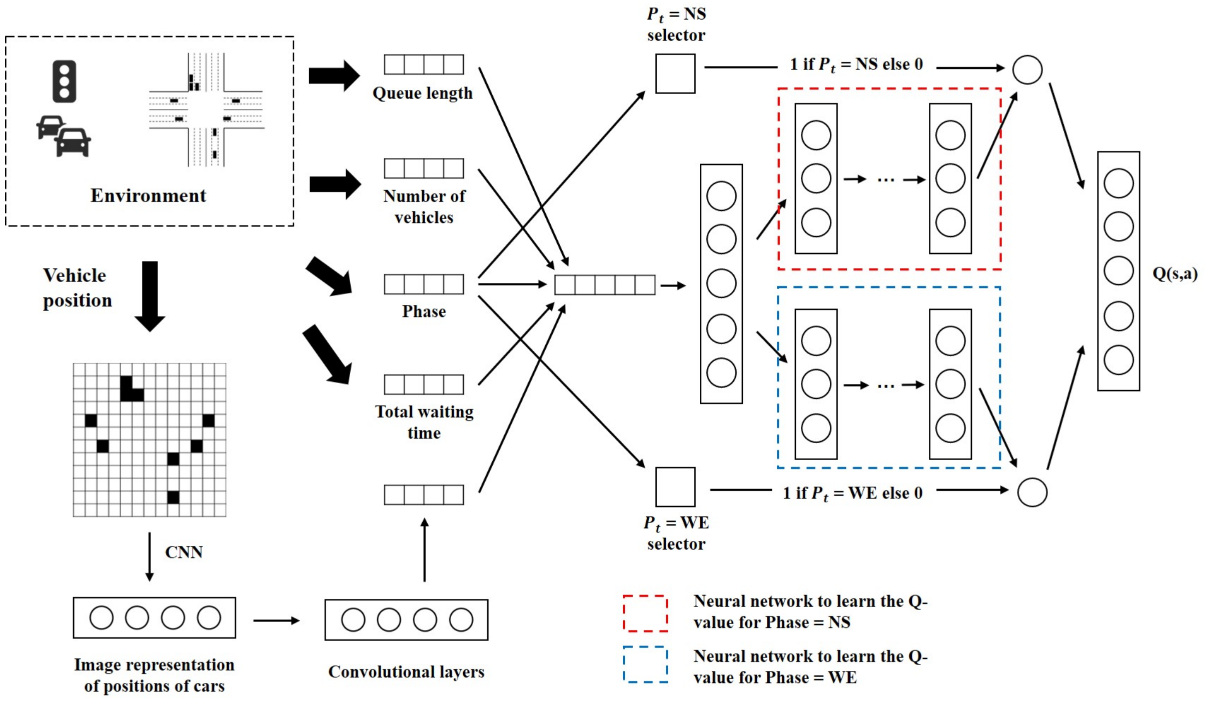


Figure 4: Structure of the deep Q network

3.5. Algorithm training

3.5.1. Offline pre-train and online training Our model is composed of two training steps (as shown in Figure 5): offline and online. In the offline stage, we set up several predefined traffic light schedules. We run the traffic simulation using these fixed schedules, allowing traffic to flow through the system to collect data samples. Compared to the typical online deep Q learning, the offline training part decides the action based on the pre-determined fixed timetable:

$$a_t^{\text{Offline}} = \text{Timetable}^{\text{Offline}}(t) \tag{13}$$

where $\text{Timetable}^{\text{Offline}}(\cdot)$ is a function that returns changing the phase or not according to the current time step t. For example, if the given fixed timetable is phase NS = 20 seconds, phase WE = 10 seconds, and $t = 0$ is phase WE. Then we have a_t^{Offline} equals to "Change to the next phase" for all $t = 10, 30, 40, 60, 70, 90, ...$ (i.e., every time point when we switch the phase according to the 20/10 timetable). And otherwise a_t^{Offline} ="Keep the current phase".

At each step t of the offline stage, the collected sample (experience) is $e_t^{\text{Offline}} = (s_t, a_t^{\text{Offline}}, R_t, s_{t+1})$. After collecting samples for several timetables, we use all collected samples to pre-train the DQN using the same loss function in Eq. 7.

After completing offline training, the pre-trained model transitions to the online stage. At each time step t, the traffic light agent observes the current state s_t from the environment and makes a decision a_t^{Online} (i.e., whether to change the traffic signal to the next phase or not) using an ϵ-greedy strategy (Eq. 14). This strategy combines exploration (i.e., randomly selecting an action with probability ϵ) and exploitation (i.e., choosing the action with the highest estimated reward), enabling the agent to strike a balance between exploring new actions and leveraging learned knowledge to make optimal real-time decisions in varying traffic scenarios.

$$a_t^{\text{Online}} = \begin{cases} \text{Randomly pick an action from } \mathcal{A} & \text{with probability } \epsilon \\ \arg\max_{a \in \mathcal{A}} Q_t(s_t, a;\ \boldsymbol{\theta}_t) & \text{otherwise} \end{cases} \tag{14}$$

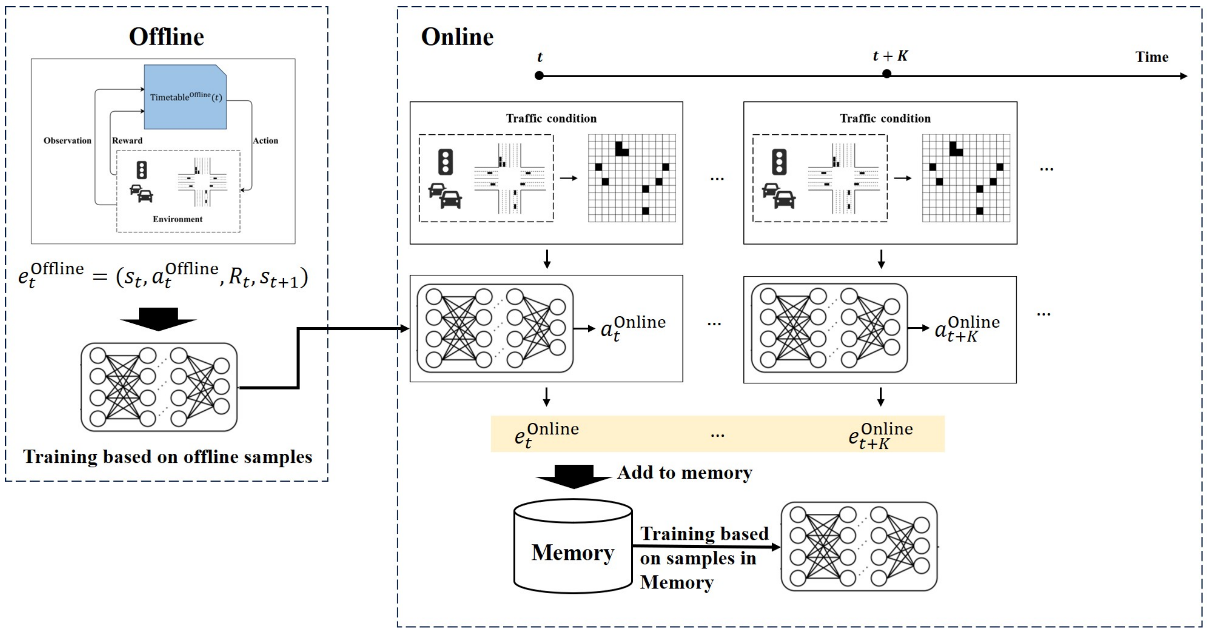

Offline
Timetable$^{\text{Offline}}(t)$
Observation
Reward
Action
Environment
$e_t^{\text{Offline}} = (s_t, a_t^{\text{Offline}}, R_t, s_{t+1})$
Training based on offline samples
Online
t
$t+K$
Time
Traffic condition
Traffic condition
a_t^{Online}
a_{t+K}^{Online}
e_t^{Online}
e_{t+K}^{Online}
Add to memory
Memory
Training based on samples in Memory

Figure 5: Illusration of offline line pre-train and online training

Upon selecting action a_t^{Online}, the agent proceeds to observe the environment with state s_{t+1} and receives the reward R_t in response. Subsequently, the tuple $e_t^{\text{Online}} = (s_t, a_t^{\text{Online}}, R_t, s_{t+1})$ is recorded and stored in memory. This process continues for K timestamps, resulting in a collection of newly acquired samples $(e_t^{\text{Online}}, \ldots, e_{t+K}^{\text{Online}})$. Then the agent updates the neural network based on the information stored in memory.

3.5.2. Phase-action dependent replay memory and Balanced sampling In deep Q learning, the agent collects samples at every time step and then uses these samples to update the deep Q network. Typically, a memory buffer is used to store all these samples. New samples are added while old samples are removed to maintain a constant sample size. This technique, known as experience replay [9], is commonly employed in RL models. However, real-world traffic conditions often exhibit imbalanced traffic patterns across different directions. Previous studies [3, 24, 31, 32] usually stored all state-action-reward training samples in a single memory buffer. In cases of significant imbalance, this memory buffer can become skewed towards samples from frequently occurring phases and actions. For instance, if a road intersection primarily experiences North-South (NS) traffic flows with fewer East-West (WE) flows, the memory buffer may be predominantly filled with samples corresponding to $a_t =$ “Keep the phase” and $P_t =$ NS (together with their associated s_t and R_t values). In contrast, less frequent phase-action combinations like $a_t =$ “Change the phase” and $P_t =$ WE may be underrepresented or ignored for experience replay sampling. Consequently, the Q values learned for these infrequent phase-action combinations may be inaccurate, potentially resulting in suboptimal decision-making.

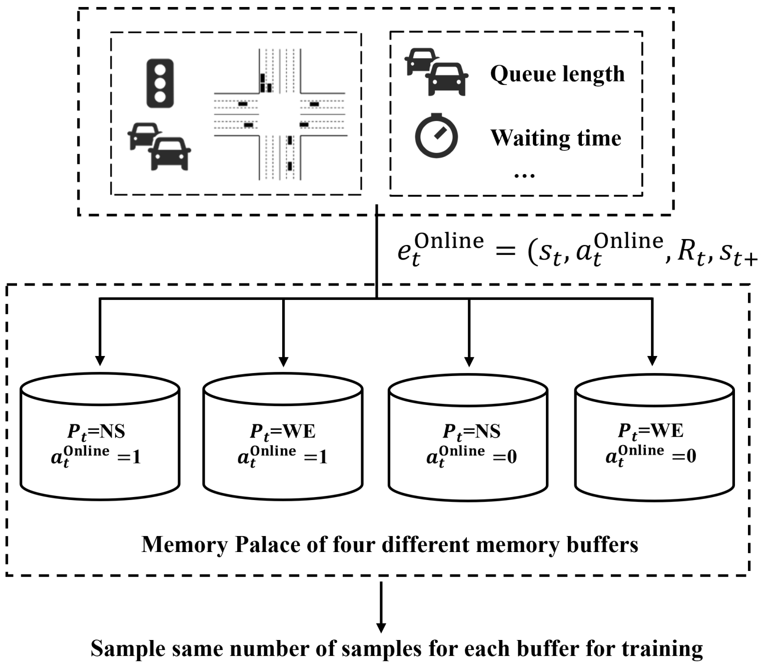
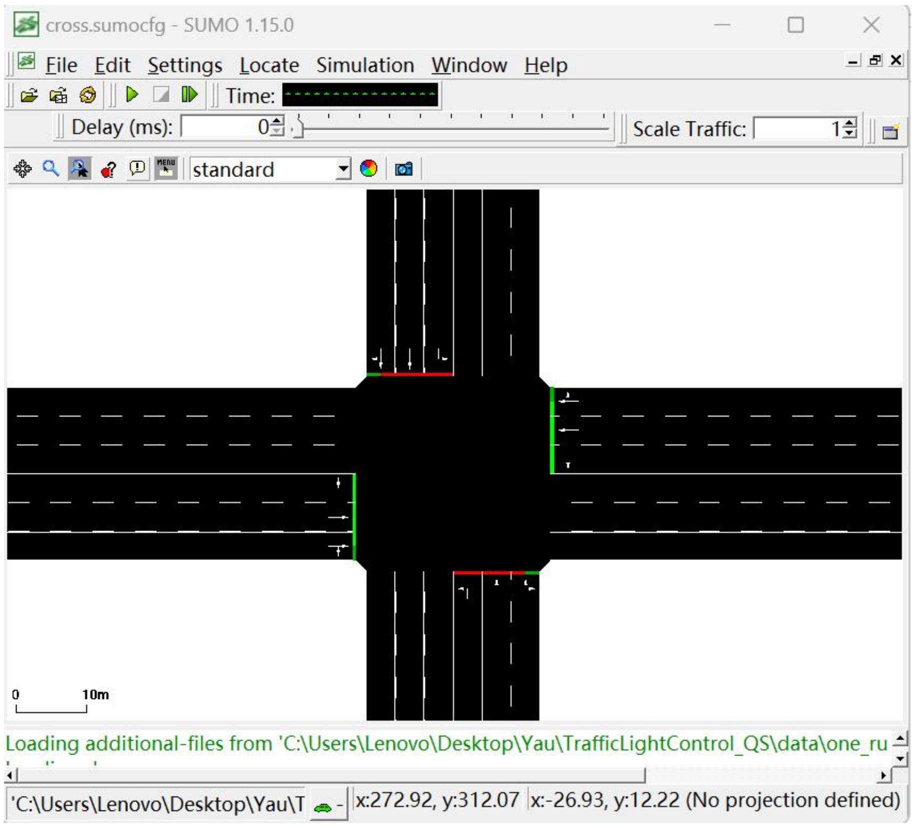

Queue length
Waiting time
...
$e_t^{\text{Online}} = (s_t, a_t^{\text{Online}}, R_t, s_{t+1})$
P_t=NS a_t^{Online} =1
P_t=WE a_t^{Online} =1
P_t=NS a_t^{Online} =0
P_t=WE a_t^{Online} =0
Memory Palace of four different memory buffers
Sample same number of samples for each buffer for training

Figure 6: Illustration of memory palace

To tackle this challenge, our study employs a Memory Palace mechanism proposed by Wei et al. [4]. Unlike the conventional Deep Q learning approach that relies on a single memory buffer for all samples, we introduce distinct memory buffers for different phase-action combinations, as depicted in Figure 6. Training samples of various phase-action combinations are stored within their respective memory buffers. During each training iteration, we randomly select an equal number of samples from these different memory buffers. This strategy ensures a balanced distribution of training samples for different phase-action combinations. Consequently, this approach enhances the network's capacity to accurately predict Q values for each specific phase-action combination.

3.6. Simulation tool

The SUMO (Simulation of Urban Mobility) tool [33] was used to simulate traffic in this study. SUMO is a widely recognized open-source traffic simulator that offers useful application programming interfaces (APIs) and a graphical user interface (GUI) for modeling large road networks and handling them effectively (see Figure 7). It supports dynamic routing based on the right-side driving rules of road intersections and provides a visual graphical interface for designing various road network layouts in multiple grid formats. Additionally, SUMO allows for controlling the traffic lights at each intersection according to user-defined policies. It also enables capturing snapshots of each simulation step, allowing us to obtain the state information s_t for our study.


cross.sumocfg - SUMO 1.15.0
File Edit Settings Locate Simulation Window Help
Time:
Delay (ms): 0
Scale Traffic: 1
standard
0 10m

Loading additional-files from 'C:\Users\Lenovo\Desktop\Yau\TrafficLightControl_QS\data\one_ru

'C:\Users\Lenovo\Desktop\Yau\T x:272.92, y:312.07 x:-26.93, y:12.22 (No projection defined)

Figure 7: Example of SUMO simulator

4. Case study design

4.1. Traffic intersection

This paper presents a case study of a real-world traffic intersection located at Xueyuan Road and Wensan Road in Hangzhou City. The traffic flows are generated using camera data from Hangzhou. Since there is no available information on turning vehicles, we have set fixed turning ratios for each data set, with 10% turning left, 60% going straight, and 30% turning right. Additional details can be found in Li et al. [34]. The layout of the intersection is depicted in Figure 8. The intersection features four directions, each with three lanes. The rightmost lane is designated for right turns and going straight, the middle lane is exclusively for going straight, and the leftmost lane is reserved for left turns.

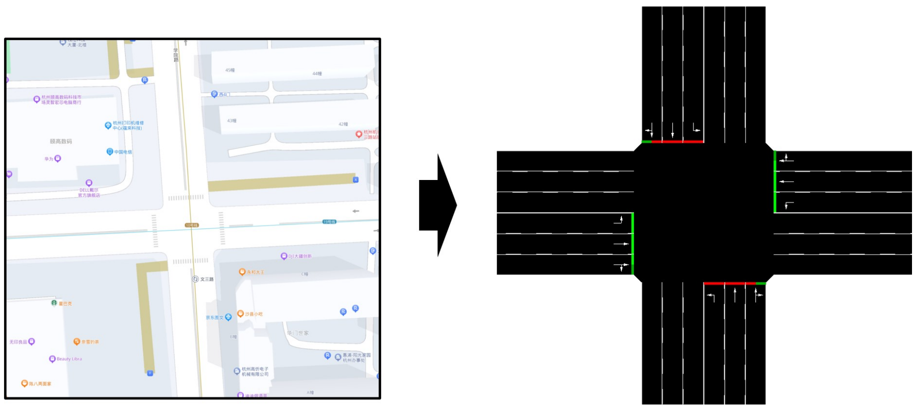


Figure 8: Layout of the case study intersection

4.2. Experiment design

To evaluate the effectiveness of the proposed algorithm, we conducted experiments on four distinct traffic conditions as outlined in Table 1. The first three scenarios were synthetic, while the last one was based on actual traffic flow data collected from surveillance cameras in Hangzhou. The first scenario represented a balanced traffic situation, where both NS and WE directions had equal vehicle arrival rates of 720 vehicles per hour. The second scenario simulated an imbalanced traffic situation with significantly higher flow rates in the WE direction compared to the NS direction. The third scenario represented an extreme switching situation, where traffic flows were present in either the WE or NE direction for different halves of the simulation time. These synthetic scenarios were designed to assess the model's performance under varying and complex traffic conditions. The last scenario utilized actual flow rate data in Hangzhou from previous studies [35–37], which originally only covered the morning peak hour from 7:00 AM to 8:00 AM. To allow for longer training time, we duplicated the data to span a 20-hour simulation period. The codes and data of this paper are available in `https://github.com/OscarTaoyuPan/TrafficLightControl_QS`.

Table 1: Experiment design

Scenario	**Directions**	**Arrival rate (# veh/hr)**	**Start time (hr)**	**End time (hr)**
Balanced	WE	720	0	20
	NS	720	0	20
Imbalanced	WE	1440	0	20
	NS	240	0	20
Switch	WE	1440	0	10
	NS	1440	10	20
Hangzhou	WE	716	0	20
	NS	1132	0	20

4.3. Parameter setting

The parameter settings in this study are similar to Wei et al. [4]. The time interval between two consecutive time steps (t and $t+1$) is set as 5 seconds. The model update interval is 300 seconds. The discount factor γ for future reward is set as 0.8. $\epsilon = 0.05$ is used for ϵ-greedy exploration. The batch size for each training is 300. The memory length for each phase-action-based replay buffer is 1000. As the total experiment time is 20 hours, the first 2 hours are used for offline training. The coefficients for the reward function are summarized in Table 2.

Table 2: Coefficients for the reward function

β_1	β_2	β_3	β_4	β_5	β_6
-0.25	-0.25	-0.25	-5	-1	-1

4.4. Benchmark methods

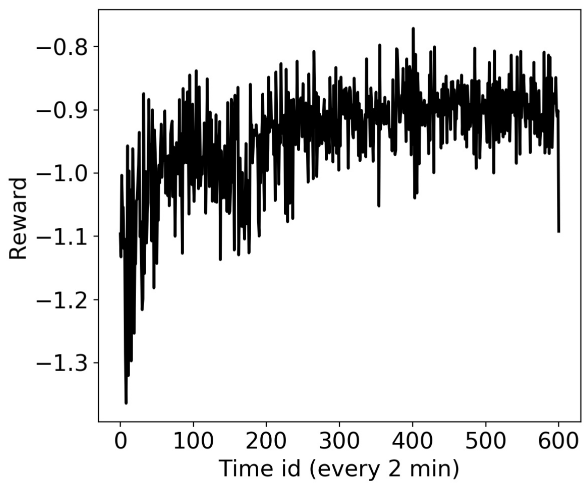
For comparison, we use the well-known Webster's Equation [38] to calculate fixed traffic signal plans for each of the four scenarios in Table 1. The green phase time for each direction is divided by proportional to the traffic flow volume. The resulting fixed signal plans are shown in Table 3

Table 3: Fixed signal plans for different scenarios

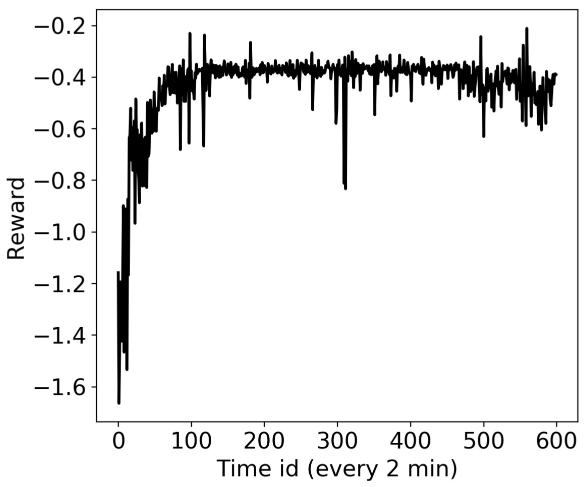
Secnario	**WE (second)**	**NS (second)**	**Cycle length (second)**
Balanced	18	18	36
Imbalanced	33	6	39
Switch	28	28	56
Hangzhou	16	25	41

4.5. Results

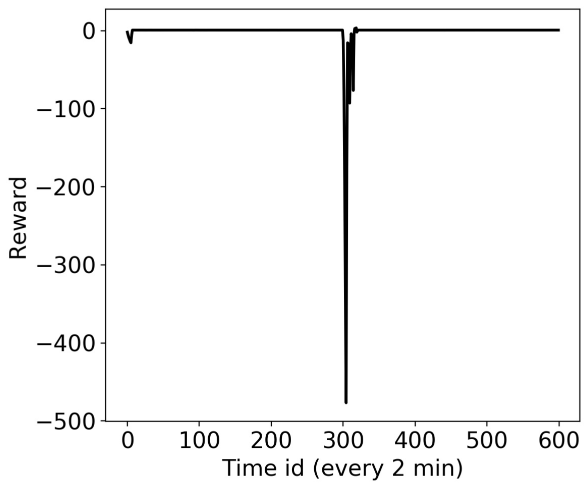
Figure 9 shows the reward function change during the training process of RL models for four different scenarios. Similar to previous DQN work [9], the training process has fluctuations but the overall reward function keeps increasing, showing that the algorithm keeps learning better traffic light control strategies. It is worth noting that in the switching scenario, where one-directional traffic flows change between the first and second half of the simulation period, the strategy is relatively straightforward to learn (i.e., always green for the current direction). As a result, the reward function remains largely unchanged for the majority of the simulation time. However, at the switching point, when the old strategy needs to be reversed, the system experiences a significant drop in the reward function. Fortunately, the RL method quickly adapts and adjusts its strategies to accommodate the dynamic traffic conditions in real time (i.e., changes the direction of the green light). This is evident from the prompt recovery of the reward function. These results demonstrate the capability of the RL approach to dynamically adapt its control strategies in response to changing traffic patterns, ensuring efficient traffic flow management.



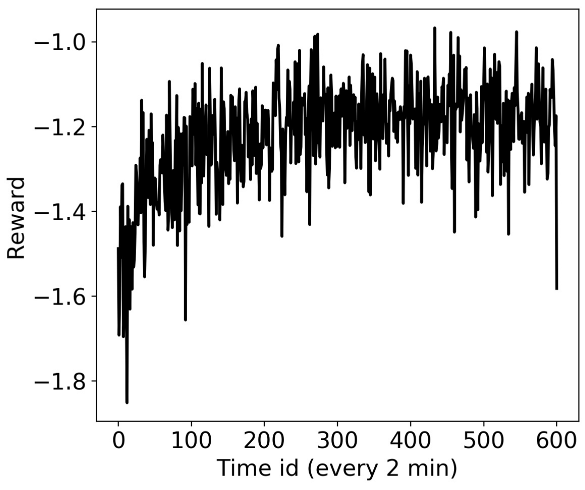
(a) Balanced



(b) Imbalanced



(c) Switch



(d) Hangzhou

Figure 9: Reward function during the training process

The final results of all models and scenarios are summarized in Table 4, with four metrics selected for comparison: average waiting time of all vehicles, average travel time of all vehicles, average queue length of all lanes, and the reward. These metrics are aggregated over a 1-hour interval. The RL-based method consistently outperforms the fixed signal method across all conditions and metrics, with notable improvements observed, particularly in imbalanced and switch scenarios. This highlights the limitations of traditional static methods in effectively addressing unconventional traffic conditions. It is worth noting that the 0 wait time and queue length for "switch" traffic conditions are due to its one-directional traffic flows during a period, where the RL model can learn that and sets the corresponding traffic signal to be green for the directions with flows. The superiority of the RL approach demonstrates its ability to adapt and optimize traffic control strategies, leading to reduced waiting times, improved travel times, shorter queue lengths, and overall enhanced performance.

Table 4: Model results

Models	Scenario	Performance metrics			
		Wait time (sec)	Travel time (sec)	Queue length	Reward
Fixed signal	Balanced	14.5	204.2	1.988	-1.139
	Imbalanced	13.8	200.5	1.383	-0.822
	Switch	80.3	697.2	7.717	-1.691
	Hangzhou	22.6	270.9	2.720	-1.488
RL	Balanced	6.2 (-57.2%)	169.3 (-17.1%)	1.180 (-40.6%)	-0.903 (+20.7%)
	Imbalanced	1.5 (-89.1%)	162.6 (-18.9%)	0.472 (-65.9%)	-0.426 (+48.2%)
	Switch	0.0 (-100%)	222.9 (-68.0%)	0.000 (-100%)	0.721 (+142.6%)
	Hangzhou	9.7 (-57.1%)	226.2 (-16.5%)	1.697 (-37.6%)	-1.164 (+21.8%)

The numbers in the parentheses represent percentage change compared to the fixed signal timing

5. Conclusion and Discussion

In this paper, we present an RL approach, specifically deep Q learning, to tackle the challenging problem of traffic light control. The input data of the model is the traffic arrival rates of each direction (e.g., Table 1). Our proposed method incorporates a comprehensive reward function that considers queue lengths, delays, travel time, and throughput, enabling an adaptive solution to varying traffic conditions. At each time step, the model intelligently determines whether to change the traffic light phase, allowing for a dynamic response to the evolving traffic environment.

The training process consists of two stages: offline and online. During offline training, we utilize pre-generated traffic flow data with fixed time schedules to establish a strong initial set of model parameters. This offline training phase provides a solid foundation for subsequent adaptive learning during the online training stage, where real-time traffic flow data is leveraged to continually refine the model's performance.

To effectively capture the dynamics associated with different traffic light phases, we employ a well-designed deep Q network structure featuring a unique "phase gate" component. This component ensures that the model focuses on the appropriate information for each specific phase, enhancing its decision-making capabilities. Furthermore, to address the issue of sample imbalance in the experience replay process of deep Q learning, we introduce a novel "memory palace" mechanism. This mechanism guarantees sufficient sampling of rarely encountered state-action combinations, thus improving the model's ability to accurately estimate Q values for all possible phase-action pairs.

To validate our approach, we conduct experiments using both synthetic and real-world traffic flow data, with a road intersection in Hangzhou, China serving as our case study. The results demonstrate that our proposed method outperforms traditional fixed signal plan traffic light control in terms of reducing vehicle waiting time (by 57.1%∼100%), queue lengths (by 40.9%∼100%), and total travel time (by 16.8%∼68.0%) in different traffic flow scenarios. Importantly, since our trained model can make real-time decisions, our approach has the potential to be implemented for real-world traffic control scenarios, leveraging up-to-date traffic flow information as input.

Future studies can be pursued in several directions to further advance the field of traffic light control. 1) Extending the model to the multi-intersection case: While this paper focuses on a single intersection, it is crucial to acknowledge that real-world road networks are significantly more complex. Future studies could delve into the interactions between different intersections and explore the application of multiple RL agents for network-level control. By considering the collective behavior of multiple intersections, we can develop more comprehensive and efficient traffic control strategies that optimize the overall network performance. 2) Developing enhanced network structures for information extraction: The current Deep Q network employs a phase gate mechanism, allowing different components to specialize in specific conditions. However, future research can investigate the integration of multiple phases or conditions to further streamline the learning process. For instance, incorporating distinct network components dedicated to peak and off-peak traffic conditions could facilitate more accurate and efficient decision-making. By tailoring the network structure to specific traffic scenarios, we can enhance the model's adaptability and performance in varying traffic conditions. 3) Comparing the proposed approach with more benchmark models, such as simulation-based optimization method [39–41] and other state-or-the-art RL methods.

Acknowledgement

I would like to express my gratitude to Dr. Xiqun (Michael) Chen for giving me academic support, reviewing my essay, and giving me feedback. I would also like to thank Dr. Baichuan Mo for his insightful guidance on my fundamental knowledge-learning process, coding process, and essay-writing process.

References

[1] Chinese Ministry of Public Security, "The number of motor vehicles in the country has reached 417 million, with more than 500 million drivers," https://www.gov.cn/xinwen/2023-01/

11/content_5736278.htm, 2023, accessed: 2023-08-13.

[2] A. J. Miller, "Settings for fixed-cycle traffic signals," *Journal of the Operational Research Society*, vol. 14, no. 4, pp. 373–386, 1963.

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

[4] 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.

[5] F. Dion, H. Rakha, and Y.-S. Kang, "Comparison of delay estimates at under-saturated and over-saturated pre-timed signalized intersections," *Transportation Research Part B: Methodological*, vol. 38, no. 2, pp. 99–122, 2004.

[6] F. V. Webster, "Traffic signal settings," Tech. Rep., 1958.

[7] I. Porche and S. Lafortune, "Adaptive look-ahead optimization of traffic signals," *Journal of Intelligent Transportation System*, vol. 4, no. 3-4, pp. 209–254, 1999.

[8] S.-B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," *Advances in applied self-organizing systems*, pp. 45–55, 2013.

[9] 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.

[10] A. Perera and P. Kamalaruban, "Applications of reinforcement learning in energy systems," *Renewable and Sustainable Energy Reviews*, vol. 137, p. 110618, 2021.

[11] Q. Fu, Z. Han, J. Chen, Y. Lu, H. Wu, and Y. Wang, "Applications of reinforcement learning for building energy efficiency control: A review," *Journal of Building Engineering*, vol. 50, p. 104165, 2022.

[12] H. A. Al-Rawi, M. A. Ng, and K.-L. A. Yau, "Application of reinforcement learning to routing in distributed wireless networks: a review," *Artificial Intelligence Review*, vol. 43, pp. 381–416, 2015.

[13] B. Hambly, R. Xu, and H. Yang, "Recent advances in reinforcement learning in finance," *Mathematical Finance*, vol. 33, no. 3, pp. 437–503, 2023.

[14] Y. Yan, A. H. Chow, C. P. Ho, Y.-H. Kuo, Q. Wu, and C. Ying, "Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities," *Transportation Research Part E: Logistics and Transportation Review*, vol. 162, p. 102712, 2022.

[15] A. Haydari and Y. Yılmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 11–32, 2020.

[16] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[17] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis, "Multiagent reinforcement learning for urban traffic control using coordination graphs," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I 19*. Springer, 2008, pp. 656–671.

[18] P. Mannion, J. Duggan, and E. Howley, "An experimental review of reinforcement learning algorithms for adaptive traffic signal control," *Autonomic Road Transport Support Systems*, pp. 47–66, 2016.

[19] M. A. Wiering *et al.*, "Multi-agent reinforcement learning for traffic light control," in *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, 2000, pp. 1151–1158.

[20] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[21] B. Bakker, S. Whiteson, L. Kester, and F. C. Groen, *Traffic Light Control by Multiagent Reinforcement Learning Systems*. Springer, 2010.

[22] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Holonic multi-agent system for traffic signals control," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5-6, pp. 1575–1587, 2013.

[23] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown Toronto," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1140–1150, 2013.

[24] 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)*, vol. 8, pp. 21–38, 2016.

[25] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 4, no. 1, pp. 1–103, 2010.

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

[27] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[28] B. Mo, H. N. Koutsopoulos, Z.-J. M. Shen, and J. Zhao, "Individual path recommendation under public transit service disruptions considering behavior uncertainty," *arXiv preprint arXiv:2301.00916*, 2023.

[29] ——, "Robust path recommendations during public transit disruptions under demand uncertainty," *Transportation Research Part B: Methodological*, vol. 169, pp. 82–107, 2023.

[30] B. Mo, Z. Ma, H. N. Koutsopoulos, and J. Zhao, "Ex post path choice estimation for urban rail systems using smart card data: An aggregated time-space hypernetwork approach," *Transportation Science*, vol. 57, no. 2, pp. 313–335, 2023.

[31] 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.

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

[33] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[34] Z. Li, H. Wei, G. Zheng, C. Chen, X. Zang, Y. Xiong, N. Xu, and H. Zhang, "Reinforcement learning for traffic signal control," https://traffic-signal-control.github.io/, 2018, accessed: 2023-09-4.

[35] G. Zheng, Y. Xiong, X. Zang, J. Feng, H. Wei, H. Zhang, Y. Li, K. Xu, and Z. Li, "Learning phase competition for traffic signal control," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1963–1972.

[36] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, and Z. Li, "Colight: Learning network-level cooperation for traffic signal control," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1913–1922.

[37] H. Wei, G. Zheng, V. Gayah, and Z. Li, "A survey on traffic signal control methods," *arXiv preprint arXiv:1904.08117*, 2019.

[38] Federal Highway Administration Department of Transportation, "Signal timing on a shoestring," https://ops.fhwa.dot.gov/publications/signal_timing/03.htm, 2022, accessed: 2023-05-20.

[39] C. Osorio and K. Nanduri, "Urban transportation emissions mitigation: Coupling high-resolution vehicular emissions and traffic models for traffic signal optimization," *Transportation Research Part B: Methodological*, vol. 81, pp. 520–538, 2015.

[40] B. Mo, Z. Ma, H. N. Koutsopoulos, and J. Zhao, "Calibrating path choices and train capacities for urban rail transit simulation models using smart card and train movement data," *Journal of Advanced Transportation*, vol. 2021, pp. 1–15, 2021.

[41] ——, "Capacity-constrained network performance model for urban rail systems," *Transportation Research Record*, vol. 2674, no. 5, pp. 59–69, 2020.