Received 16 January 2024; accepted 14 February 2024. Date of publication 22 February 2024;
date of current version 14 March 2024. The review of this article was coordinated by Editor Marko Beko.

Digital Object Identifier 10.1109/OJVT.2024.3368693

Scalable Reinforcement Learning Framework for Traffic Signal Control Under Communication Delays

AOYU PANG [1], MAONAN WANG [1], YIRONG CHEN [2], MAN-ON PUN [3] (Senior Member, IEEE), AND MICHAEL LEPECH [2]

[1]Future Network of Intelligence Institute (FNii), Chinese University of Hong Kong, Shenzhen 518172, China
[2]Department of Civil and Environmental Engineering, Stanford University, Stanford, CA 94305 USA
[3]School of Science and Engineering, Chinese University of Hong Kong, Shenzhen 518172, China

CORRESPONDING AUTHOR: MAN-ON PUN (e-mail: simonpun@cuhk.edu.cn)

This work was supported by the National Key Research and Development Program of China under Grant 2020YFB1807700.

ABSTRACT Vehicle-to-everything (V2X) technology is pivotal for enhancing road safety, traffic efficiency, and energy conservation through the communication of vehicles with their surrounding entities such as other vehicles, pedestrians, roadside infrastructure, and networks. Among these, traffic signal control (TSC) plays a significant role in roadside infrastructure for V2X. However, most existing works on TSC design assume that real-time traffic flow information is accessible, which does not hold in real-world deployment. This study proposes a two-stage framework to address this issue. In the first stage, a scene prediction module and a scene context encoder are utilized to process historical and current traffic data to generate preliminary traffic signal actions. In the second stage, an action refinement module, informed by human-defined traffic rules and real-time traffic metrics, adjusts the preliminary actions to account for the latency in observations. This modular design allows device deployment with varying computational resources while facilitating system customization, ensuring both adaptability and scalability, particularly in edge-computing environments. Through extensive simulations on the SUMO platform, the proposed framework demonstrates robustness and superior performance in diverse traffic scenarios under varying communication delays. The related code is available at https://github.com/Traffic-Alpha/TSC-DelayLight.

INDEX TERMS Traffic signal control, scalable framework, historical-future data fusion, reinforcement learning.

I. INTRODUCTION

Vehicle-to-everything (V2X) communication is poised to revolutionize the transportation landscape by enabling vehicles to interact with each other and with road infrastructure, thereby orchestrating a more coordinated and efficient traffic flow [1]. At the heart of this transformation lies intelligent Traffic Signal Control (TSC) systems, which are critical for the practical deployment of V2X technologies and the realization of smart road networks [2]. Intelligent TSC systems are instrumental in mitigating one of the primary sources of urban traffic congestion: inefficient traffic signal timing at intersections. Traditional TSC methods, such as the Webster method [3], focus on optimizing traffic light cycles based on fixed traffic volumes, while adaptive systems like SCATS (Sydney Coordinated Adaptive Traffic System) [4] dynamically adjust signals in response to sensor data. These conventional systems, while foundational, struggle to accommodate the inherently dynamic and stochastic nature of traffic flow, often leading to inefficient traffic light timing and more congestion. Moreover, the parameterization of these systems involves extensive expertise, rendering them less adaptable to the unpredictable patterns of real-world traffic conditions [5].

Recent advancements in reinforcement learning (RL) have been directed towards harnessing real-time traffic data to enhance traffic light control, signaling a paradigm shift from static to dynamic TSC systems without expert knowledge [6],

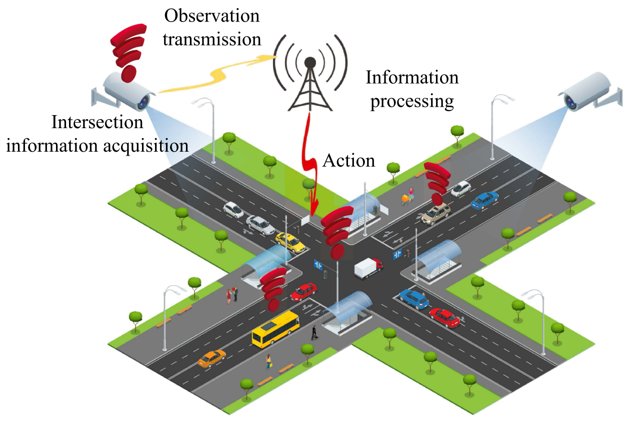


FIGURE 1. Schematic representation of the delay incurred from data acquisition at an intersection to the agent's control action, illustrating the stages of information transmission and processing time.

[7], [8], [9]. The RL-based methodologies diverge in their action space designs, with strategies ranging from setting phase splits [6], determining the duration of the current phase [9], deciding on phase continuation [7], to the more adaptable approach of selecting the subsequent phase [8]. The latter, which is employed in our research for its flexibility in phase selection and duration setting, represents a major improvement from the conventional rigid cycle-based signaling schemes. However, existing RL-based approaches are handicapped by their presumption of instantaneous access to accurate traffic data, disregarding the inevitable transmission and processing delays as depicted in Fig. 1. This oversight can lead to suboptimal control decisions, thereby necessitating a framework capable of accounting for such delays to ensure the reliability and efficiency of TSC systems in real-world scenarios. In this research, the transmission delay is defined as the time for transmitting packets from the traffic lights to the Road-Side Unit (RSU), and vice versa. The processing time is defined as the time for performing the task at the mobile edge computing (MEC) servers [10]. Clearly, the performance of such TSC systems derived from an idealistic assumption is under doubt in practical deployment.

To address these challenges, our study introduces an innovative RL-based framework designed to maintain efficacy amidst such delays, which can be detrimental to traffic management systems. The proposed framework is structured in two stages: The first stage leverages a dynamic model to predict near-term traffic states, incorporating a scene context encoder that processes historical and delayed current traffic data, thus generating preliminary traffic signal actions. Subsequently, the second stage refines the preliminary actions through an action refinement module, which integrates expert-defined traffic rules and real-time data such as lane occupancy and phase duration, to mitigate the adverse effects of observation latency.

Our approach is characterized by its modular design, which not only facilitates deployment on roadside terminal devices with varying computational capacities but also allows for the substitution of individual components to tailor to specific requirements. Such design ensures adaptability and scalability, particularly in edge computing scenarios where resources are limited. The contributions of this work are:

- We present a novel scalable RL-based framework that is inherently designed to address the challenges posed by observation delay in TSC systems.
- The framework is distinguished by its two-stage process, involving sophisticated data fusion for action generation, followed by a rule-informed action refinement to enhance decision accuracy.
- Through rigorous testing on the Simulation of Urban MObility (SUMO) platform, we demonstrate the framework's robustness and superior performance across various road network configurations, validating its potential for real-world applications.

The rest of the paper is organized as follows. Section II provides a review of the relevant literature. In Section III, we define the traffic terminology pertinent to our discussion. The core of our proposed methodology, including the historical-future data fusion module and the action refinement module, is detailed in Section IV. Section V describes the experimental framework, outlines the benchmark methods, and evaluates the performance of our proposed framework against these benchmarks with a focus on the average waiting time of vehicles. Finally, Section VI concludes our findings and offers insights into potential avenues for future research in this domain.

II. RELATED WORKS

Urban areas frequently face the challenge of traffic congestion [11], and by strategically placing and designing traffic signal systems, controlled traffic flow can be facilitated and achieved by granting the right-of-way to conflicting traffic streams [12]. To alleviate traffic congestion, some studies advocate replacing traditional signaling control methods with intelligent and adaptive systems [3], [13], [14]. The Webster algorithm [3] calculates the green time for each phase using parameters such as traffic flow rate and cycle length to achieve optimal traffic flow. While the Webster algorithm is suitable for small intersections, its performance may be suboptimal in handling complex traffic situations. Additionally, Self-Organizing Traffic Light Control (SOTL) [15], [16] is designed based on perception and rules. Specifically, when vehicles on the red light side request a green light and the current green light side exceeds the set threshold time, the green light will switch to the next phase. Conversely, if a vehicle on the green light side requests a green light, it will maintain the current green light.

Despite the significant improvements demonstrated by these methods compared to fixed-time schemes, the emergence of reinforcement learning (RL) signal control algorithms in recent years has shown outstanding performance in alleviating traffic congestion [17], [18], [19], [20], [21]. [22] conducted a comprehensive comparison of the Proximal Policy Optimization (PPO) algorithm with other advanced RL algorithms in terms of traffic efficiency, concluding that PPO exhibits stronger portability and robustness. However, in the high-dimensional traffic state space, traditional RL algorithms face limitations in efficiently computing value functions or policy functions for all states [23]. In this context, the use of deep learning to extract significant features has become a viable solution. This involves training deep neural networks to extract features from complex state spaces, with Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) being prominent choices for feature extraction in high-dimensional traffic state spaces [24]. Nevertheless, most existing studies did not consider the various delays in the real-life transmission of information, which may raise concerns in practical applications.

In practice, delayed environment observation is a common problem hindering the RL deployment in many real-world applications beyond TSC [25], [26], [27]. Various solutions have been devised in the literature. For instance, in the field of robotic control, an improved temporal difference (TD) learning algorithm is used to solve problems caused by feedback delays [28]. Furthermore, it has been proposed for energy management in commercial buildings to cope with the feedback delay by expanding the state space to a larger but partially observed space [29]. Generally speaking, these methods were designed to predict the current state information and subsequently compensate for the delay.

In the field of TSC, there are some efforts to enhance the performance of RL agents through the prediction of future traffic flows [30], [31], [32]. Notably, [32] proposed a traffic delay-aware feature transformation module, enabling the model to explicitly model the time delay in the propagation of spatial information. However, such prediction models usually require pre-training, incurring high computational complexity. In [33], a rule-based prediction model was introduced to discover and exploit the causal structure in the environment. While this reduces the complexity of pre-training models, relying solely on rule-based predictions may struggle to adapt to the complex and variable traffic trends at intersections. Thus, it is challenging to cope with long unknown delays by prediction alone. Our prior work [34] addressed delay-related challenges through the development of feature extraction modules that utilize multiple historical observations. Despite its many advantages, the approach demonstrated limitations in handling prolonged delays. In the current study, we refine this approach by integrating a scene context encoder with a novel scene prediction module that extracts features from sequential observations. We further introduce a rule-informed action refinement process to improve decision-making accuracy. This enhanced framework presents a more robust solution for managing the observation delays commonly encountered in TSC applications, paving the way for more resilient traffic management systems.

III. PRELIMINARIES

This section presents the TSC-related terminology pertinent to our study.

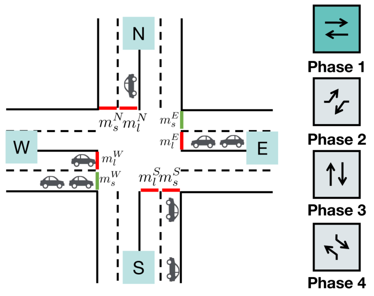


FIGURE 2. Illustration of a standard 4-way intersection, showcasing 8 traffic movements and the corresponding 4 signal phases.

Traffic Movements: In a four-legged road intersection, there are four entrances named East (E), West (W), North (N) and South (S). A traffic movement refers to vehicles moving from an incoming approach to an outgoing approach. Each entrance has two movements to exit, which are going left (l) and going straight (s). Note that this work ignores going right, assuming that going right is always allowed in countries following left-hand traffic laws. As shown in Fig. 2, eight traffic movements in the intersection can be defined using their entrance and direction attributes, namely m_l^E, m_s^E, m_l^W, m_s^W, m_l^N, m_s^N, m_l^S and m_s^S. For notational convenience, they are referred by aliases: m_1, m_2, m_3, m_4, m_5, m_6, m_7 and m_8, respectively in the sequel.

Movement Signals: The movement signal is defined by a traffic movement with the green and red signals indicating that the corresponding movement is allowed and forbidden, respectively. The yellow signal is ignored as a fixed-duration yellow signal is assumed to follow each green signal. Accordingly, eight movement signals are considered in this work.

Phases: A phase is a combination of movement signals that occur simultaneously. Clearly, movements allowed in the same phase should not conflict with each other. Since only green and red signals are considered, a phase can be considered as a set of allowed traffic movements while the others are prohibited. Fig. 2 shows four phases constituted by the eight aforementioned movements, namely $\mathcal{P}_1 = \{m_2,\ m_4\}$, $\mathcal{P}_2 = \{m_1,\ m_3\}$, $\mathcal{P}_3 = \{m_6,\ m_8\}$ and $\mathcal{P}_4 = \{m_5,\ m_7\}$. We denote by $\boldsymbol{\mathcal{P}} = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, \mathcal{P}_4\}$ the feasible phase set. The green lines and red lines represent the allowed traffic movements and the prohibited ones in a phase, respectively.

Signal Plans: A signal plan is an ordered set of traffic light phases, each with an allocated duration, that directs the sequence of movements at an intersection. We represent a signal plan over a period from t_1 to t_T as a series of tuples $\{(p_1, t_1), (p_2, t_2), \ldots, (p_T, t_T)\}$, where each phase p_n is part of a predefined phase set $\boldsymbol{\mathcal{P}}$ and is activated at the corresponding time t_n. The index n ranges from 1 to T, indicating the progression of phases within the signal plan.

IV. THE SCALABLE RL-BASED TSC FRAMEWORK WITH DELAY MITIGATION

In this section, we present a two-stage RL framework to tackle the challenges associated with delayed observations in TSC

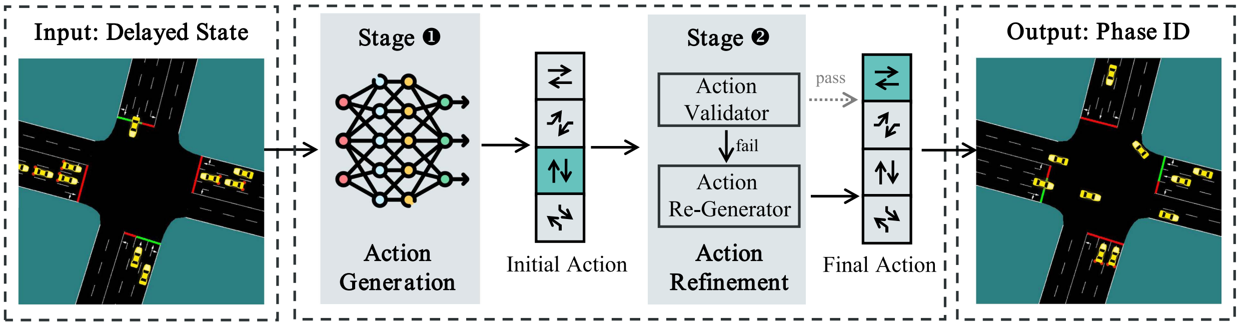


FIGURE 3. Schematic of the two-stage RL-based TSC framework with delay mitigation. Stage 1 illustrates the RL agent proposing an initial traffic signal action based on the current environment state. In Stage 2, this initial action undergoes a validation process against established traffic rules. If the action complies, it is executed. Otherwise, it is adjusted in accordance with the predefined rules before execution. The diagram exemplifies a scenario where the RL agent's initial suggestion for Phase 3 to set a green light is revised in Stage 2, resulting in Phase 1 being the final green light decision for implementation.

system. Fig. 3 illustrates the system architecture, which comprises an initial action generation stage where the RL agent formulates traffic signal control actions by integrating historical trends with projections of future traffic scenarios. This is followed by an action refinement stage that enhances the preliminary decisions by incorporating established traffic regulations and real-time observational data, thereby increasing the system's robustness to significant communication delays. The forthcoming sections will elaborate on the design of the RL agent and provide a detailed description of the action generation and refinement stages.

A. RL DESIGN

State: The following seven state variables (SV) are used to characterize each movement m_i for $i = 1, 2, \ldots, 8$.

1) *Flow* (SV_1): Number of cars passing the stop line;
2) *Mean Occupancy* (SV_2): Occupancy is defined as the ratio between the total length of roads occupied by vehicles and the total length of lanes. This SV is computed by averaging the occupancy over a single time slot;
3) *Maximum Occupancy* (SV_3): The largest occupancy over a time slot;
4) *Is-Straight* (SV_4): A binary variable indicating whether a traffic movement is going straight or not;
5) *Number of Lanes* (SV_5): The number of lanes of the movement under consideration;
6) *Is-Min-Green* (SV_6): A binary variable indicating if a green signal m_i lasts longer than the user-defined duration;
7) *Is-Green* (SV_7): A binary variable indicating whether a movement signal is green or red.

Let $\boldsymbol{m}_t^i = [SV_1^i(t), SV_2^i(t), \ldots, SV_7^i(t)]$ represents the information for movement m_i at time t, where $i \in \{1, 2, \ldots, 8\}$ corresponds to one of eight traffic movements in the intersection. The full intersection state $\boldsymbol{J}_t \in \mathbb{R}^{8\times 7}$ at time t is then defined as:

$$\boldsymbol{J}_t = [\boldsymbol{m}_t^1, \boldsymbol{m}_t^2, \ldots, \boldsymbol{m}_t^8]^T, \tag{1}$$

where $[\cdot]^T$ stands for the transpose operator. Denote by D the observation delay, we utilize the past K delayed observations to form an input state $\boldsymbol{S}_t \in \mathbb{R}^{K\times 8\times 7}$ for feature extraction where $\boldsymbol{S}_t$ takes the following form:

$$\boldsymbol{S}_t = \left[\boldsymbol{J}_{t-K-D+1}, \boldsymbol{J}_{t-K-D+2}, \ldots, \boldsymbol{J}_{t-D}\right]. \tag{2}$$

These extracted features are then used by the agent to determine appropriate actions given the delayed state information as illustrated in Fig. 4.

Action: At the end of each time slot, the RL agent takes an action A_t to choose a phase for the next time slot based on input state $\boldsymbol{S}_t$. More specifically, $A_t = a$, where $a \in \{1, 2, \ldots, N\}$ with N representing the number of phases in the intersection.

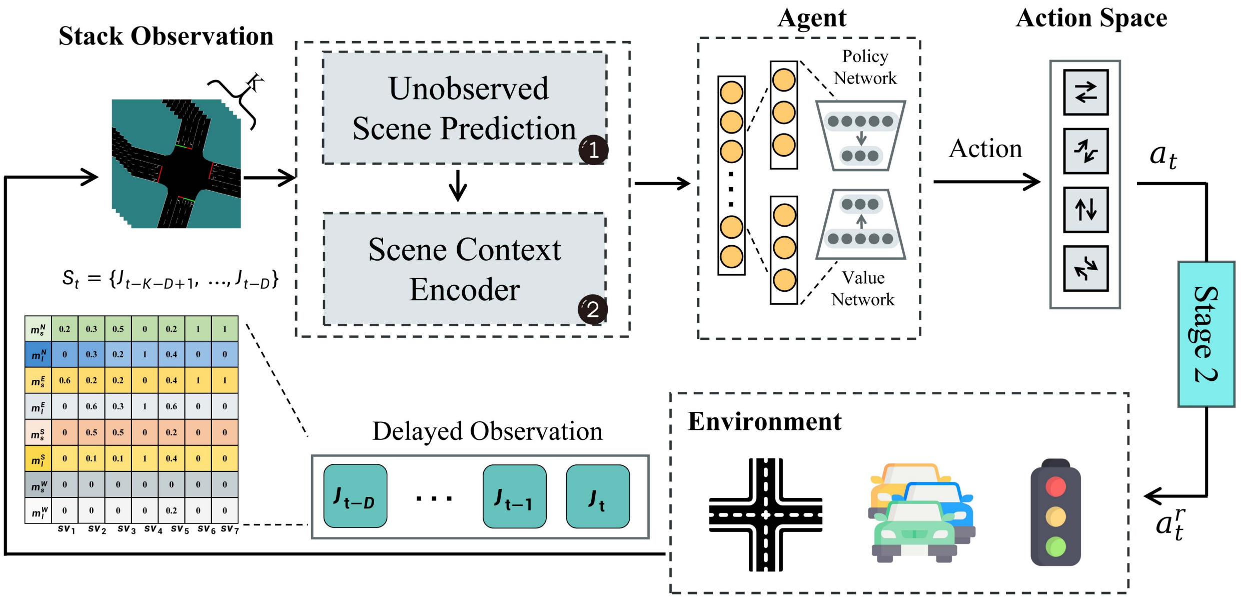
Reward: Even though the goal of the proposed RL-based TSC system is to minimize the total waiting time of all vehicles, it is difficult to obtain the total waiting time from real-world road traffic sensors. To cope with this problem, we propose to use the penalty on the total queue length of all lanes at the intersection as the reward. Denote by $q_\ell^i(t)$ the queue length of the ℓ-th lane for movement m_i at time t. Thus, the proposed reward takes the following form:

$$R_t = -\sum_{i=1}^{8}\sum_{\ell=1}^{L_i} q_\ell^i(t), \tag{3}$$

where L_i is the number of lanes for movement m_i. As the queue length may grow out of control during traffic congestion, the reward is normalized during training.

B. ACTION GENERATION STAGE: UNOBSERVED SCENE PREDICTION

In the action generation stage of our proposed RL framework, showcased in Fig. 4, we introduce the scene prediction module that forecasts imminent traffic conditions, thereby addressing the issues associated with communication delays. This module employs two distinct methodologies: rule-based prediction, which utilizes predetermined traffic patterns and rules to estimate future states, and learning-based prediction, which relies on data-driven techniques to adaptively predict traffic scenarios. These methods are complemented by the scene context encoder, which combines the predicted future conditions



	sv_1	sv_2	sv_3	sv_4	sv_5	sv_6	sv_7
m_s^N	0.2	0.3	0.5	0	0.2	1	1
m_l^N	0	0.3	0.2	1	0.4	0	0
m_s^E	0.6	0.2	0.2	0	0.4	1	1
m_l^E	0	0.6	0.3	1	0.6	0	0
m_s^S	0	0.5	0.5	0	0.2	0	0
m_l^S	0	0.1	0.1	1	0.4	0	0
m_s^W	0	0	0	0	0	0	0
m_l^W	0	0	0	0	0.2	0	0

FIGURE 4. Illustration of the RL training process within the action generation stage (Stage 1). The agent receives information from the past *K* states from the environment with a delay of *D*. Utilizing the scene prediction module, the system forecasts future traffic scenarios based on these *K* states. The scene context encoder then integrates the prediction with historical traffic data, feeding the integrated feature into both the policy network and the value network to generate preliminary traffic signal control actions.

with historical traffic information to form a comprehensive traffic state representation. The dual-method approach within the scene prediction module ensures flexibility and adaptability, allowing for customization based on the computational resources available on the deployment device, which is particularly beneficial for varying edge computing environments. This subsection will clarify the detail of the rule-based and learning-based prediction methods within the scene prediction module.

1) RULE-BASED PREDICTION OF TRAFFIC STATE

Built upon the model proposed by [35], this study develops a state inference model to predict unobserved traffic states, denoted as $\hat{\boldsymbol{J}}_{t+1} = [\widehat{SV}_1^i(t+1), \ldots, \widehat{SV}_7^i(t+1)]$, utilizing time-series data and established traffic flow principles. For the i-th traffic movement, state variables are forecasted according to the following set of rules:

Flow Prediction ($\widehat{SV}_1$): The traffic flow is contingent on the status of the traffic signal. A red signal implies no flow, while a green signal indicates flow based on the average rate observed during previous green intervals within a defined time window K. The predicted flow is given by:

$$\widehat{SV}_1^i(t+1) = \begin{cases} 0, & \text{if } SV_7^i(t) = 0, \\ \frac{\sum_{\tau=t-K+1}^{t} SV_1^i(\tau) \cdot SV_7^i(\tau)}{K_{green}}, & \text{if } SV_7^i(t) = 1, \end{cases} \tag{4}$$

where K_{green} takes the following form, representing the count of time slots within window K during which the signal is green:

$$K_{green} = \sum_{\tau=t-K+1}^{t} SV_7^i(\tau). \tag{5}$$

Mean Occupancy Prediction ($\widehat{SV}_2$): The occupancy level at any given time is intricately linked to the signal phase, as depicted in Fig. 5. When the signal is red, vehicles accumulate at the stop line, leading to an increase in occupancy. Conversely, a green signal facilitates vehicle departure, resulting in a decrease in occupancy. This dynamic is captured in Fig. 5(a), which illustrates the intersection's state at time t, and Fig. 5(b), which visualizes the predicted state at time $t+1$. For instance, if Phase-2 is green at time $t+1$, the occupancy for the corresponding movement is the current occupancy minus the proportion of vehicles that have departed, as shown in Fig. 5(b). For other phases that are red, the occupancy is the sum of the current occupancy and the additional occupancy due to new arrivals. The predicted mean occupancy is thus expressed as:

$$\widehat{SV}_2^i(t+1) = SV_2^i(t) + \Delta SV_2^i(t), \tag{6}$$

where $\Delta SV_2^i(t)$ is the net change in occupancy. This change is calculated based on the signal state as follows:

$$\Delta SV_2^i(t) = \begin{cases} \lambda \cdot k, & \text{if } SV_7^i(t) = 0, \\ -\widehat{SV}_1^i(t+1) \cdot k, & \text{if } SV_7^i(t) = 1, \end{cases} \tag{7}$$

with λ denoting the average rate of vehicle arrivals, which is assumed to follow a Poisson distribution, and k representing a conversion factor that translates the number of vehicles into an occupancy metric based on average vehicle length and lane length.

Maximum Occupancy Prediction ($\widehat{SV}_3$): The maximum occupancy is hypothesized to remain constant during green signal phases and to rise during red phases due to incoming

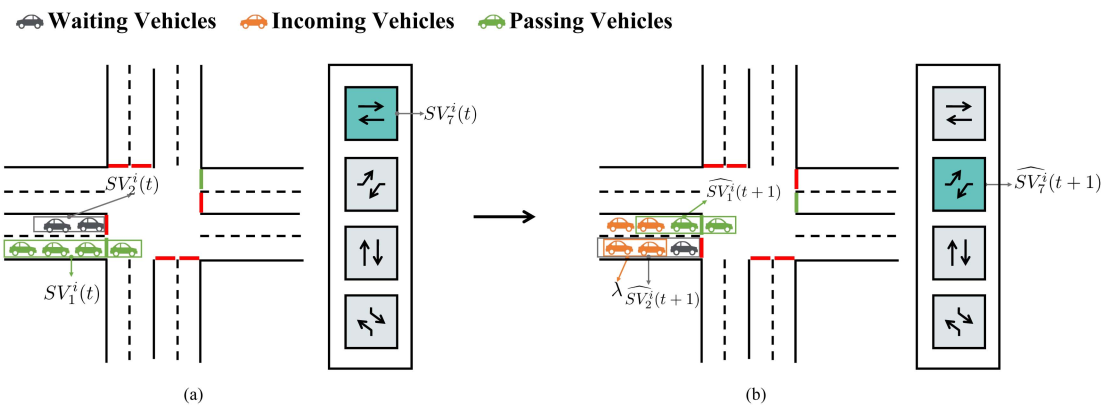


FIGURE 5. Schematic of the rule-based prediction model within the TSC framework. (a) displays the intersection status at time T, highlighting Phase-1 with a green signal, allowing vehicular flow from west to east, denoted as $SV_1^i(t)$, while other directions are on hold. The lane occupancy rate is represented as $SV_2^i(t)$, and the current green phase is indicated by $SV_7^i(t)$. (b) illustrates the prediction of intersection status based on historical data, suggesting a potential shift to Phase-2 due to its high congestion levels. The model anticipates the next action to prioritize Phase-2, and accordingly projects the traffic flow $\widehat{SV}_1$ and occupancy $\widehat{SV}_2$, considering the traffic movements regulated by the activation of Phase-2.

traffic. The prediction is formalized as:

$$\widehat{SV}_3^i(t+1) = \begin{cases} SV_3^i(t), & \text{if } SV_7^i(t) = 1, \\ \max(SV_3^i(t), SV_2^i(t) + \lambda \cdot k), & \text{if } SV_7^i(t) = 0. \end{cases} \tag{8}$$

Static Features Prediction ($\widehat{SV}_4, \widehat{SV}_5, \widehat{SV}_6$): These features, being static, are propagated to the subsequent state without modification:

$$\widehat{SV}_j^i(t+1) = SV_j^i(t), \quad \text{for } j \in \{4, 5, 6\}. \tag{9}$$

Signal State Prediction ($\widehat{SV}_7$): The signal state is determined by a greedy algorithm that favors the traffic phase experiencing the worst congestion:

$$\widehat{SV}_7^i(t+1) = \begin{cases} 1, & \text{if } p_i \in \mathcal{P}^* \\ 0, & \text{otherwise,} \end{cases} \tag{10}$$

where $\mathcal{P}^* \in \boldsymbol{\mathcal{P}}$ with

$$\sum_{m_j \in \mathcal{P}^*} \widehat{SV}_3^j(t+1) \geq \sum_{m_k \in \mathcal{P}_\ell, \mathcal{P}_\ell \neq \mathcal{P}^*} \widehat{SV}_3^k(t+1). \tag{11}$$

2) LEARNING-BASED PREDICTION

In addition to the rule-based traffic intersection prediction method previously discussed, we incorporate a Long Short-Term Memory (LSTM)-based prediction model to forecast future traffic states. This model leverages sequential historical data, denoted as $\boldsymbol{S}_t = [\boldsymbol{J}_{t-K-D+1}, \ldots, \boldsymbol{J}_{t-D}]$. The LSTM is tasked with predicting the subsequent traffic state $\hat{\boldsymbol{J}}_{t+1}$, formulated as:

$$\hat{\boldsymbol{J}}_{t+1} = F^{\text{LSTM}}(\boldsymbol{S}_t; \boldsymbol{\Theta}), \tag{12}$$

where F^{LSTM} and $\boldsymbol{\Theta}$ represent the LSTM mapping function and the set of trainable parameters within the model, respectively.

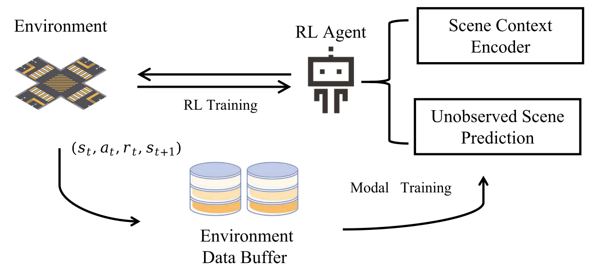


FIGURE 6. Predict-LSTM training process within the RL framework. The diagram depicts the storage of interaction tuples (s_t, a_t, r_t, s_{t+1}) in a data buffer following agent-environment exchanges. The LSTM model subsequently accesses this buffer to retrieve data for continuous parameter optimization, enhancing its traffic state prediction accuracy over iterative training cycles.

The training of the prediction model is integrated with the RL process, drawing inspiration from the Dyna-Q framework [36] to efficiently utilize the data generated from agent-environment interactions. As illustrated in Fig. 6, the interaction tuple (s_t, a_t, r_t, s_{t+1}) is stored in a data buffer. The LSTM model continuously retrieves interaction data from this buffer to update its parameters, thereby refining its predictive capability over time. This approach, referred to as Predict-LSTM, allows for the concurrent evolution of the RL policy and the predictive model, ensuring that both components benefit from the most current data and insights gleaned from the environment.

C. ACTION GENERATION STAGE: SCENE CONTEXT ENCODER

After the scene prediction module, we obtain the predicted unobserved state $\hat{\boldsymbol{J}}_{t-D+1}$. We then form an augmented state set that includes the predicted state and the original delayed observations $\boldsymbol{S}_t$:

$$\hat{\boldsymbol{S}}_t = \left\{\boldsymbol{S}_t, \hat{\boldsymbol{J}}_{t-D+1}\right\} = \left\{\boldsymbol{J}_{t-K-D+1}, \ldots, \boldsymbol{J}_{t-D}, \hat{\boldsymbol{J}}_{t-D+1}\right\}. \tag{13}$$

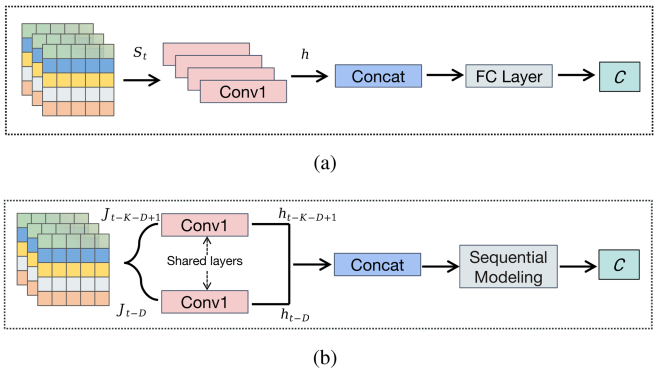


FIGURE 7. Two different architectures for scene context encoder: (a) Stacked-based encoder, (b) embedding-based encoder. The structures of CNN, LSTM, and Transformer are used in the subsequent process to extract time series features.

This augmented state combines current delayed observations with the prediction of the unobserved state.

The scene context module is then used to extract spatial-temporal features $\boldsymbol{C}$ from the intersection scene based on this augmented state input $\hat{\boldsymbol{S}}_t$. As illustrated in Fig. 7, two different encoder architectures are proposed for the scene context module: a stacked-based encoder and an embedding-based encoder. The stacked-based encoder processes the augmented state sequentially, while the embedding-based encoder processes each input in parallel and combines them through embedding. Both encoders output the scene context feature $\boldsymbol{C}$, capturing spatial and temporal information about the intersection.

1) STACKED-BASED ENCODER

As illustrated in Fig. 7(a), the stacked-based encoder processes the intersection information $\boldsymbol{J}_i$ from the augmented state inputs $\hat{\boldsymbol{S}}_t$ individually. Specifically, we stack the intersection information $\boldsymbol{J}$ from time steps $t-K-D+1$ to $t-D$ along with the predicted unobserved state $\hat{\boldsymbol{J}}_{t-D+1}$ and input this to a multi-channel CNN module. This method is referred to as the observation-stacking CNN-based (SCNN). The SCNN encoder allows the CNN to independently extract features $\boldsymbol{h}_i$ from each input:

$$\boldsymbol{h}_i = \phi\left(\boldsymbol{F}_i^{\text{SCNN}}(\boldsymbol{J}_i)\right), \quad \forall \boldsymbol{J}_i \in \hat{\boldsymbol{S}}_t, \tag{14}$$

where $\phi(\cdot)$ stands for the ReLU function while $\boldsymbol{F}_i^{\text{SCNN}}(\cdot)$ is the i-th channel of a CNN model.

The extracted features $\{\boldsymbol{h}_i\}$ are then concatenated and passed through an FC layer to produce the scene context vector $\boldsymbol{C}^{\text{SCNN}}$ as follows:

$$\boldsymbol{C}^{\text{SCNN}} = \mathcal{M}\left(\left[\boldsymbol{h}_{t-K-D+1}, \ldots, \boldsymbol{h}_{t-D}, \boldsymbol{h}_{t-D+1}\right]\right), \tag{15}$$

where $\mathcal{M}(\cdot)$ is a multilayer perceptron network. By processing each time step's observation and the predicted state individually, the SCNN encoder can capture temporal dependencies and extract informative spatial-temporal features.

In the sequel, this observation-stacking CNN-based model is referred to as SCNN-K. Note that setting $K=1$, i.e. only one delayed observation is employed, will degenerate (15) to the case considered in [9].

2) EMBEDDING-BASED ENCODER (EBE)

In contrast to the stacked-based encoder, the embedding-based encoder processes each observation in parallel using a shared CNN layer before fusing the resulting latent features, as shown in Fig. 7. Mathematically, the latent feature vector $\boldsymbol{h}_i$ for each observation $\boldsymbol{J}_i \in \hat{\boldsymbol{S}}_t$ is extracted by:

$$\boldsymbol{h}_i = \phi\left(\boldsymbol{F}^{\text{EBE}}(\boldsymbol{J}_i)\right), \quad \forall \boldsymbol{J}_i \in \hat{\boldsymbol{S}}_t \tag{16}$$

where $\boldsymbol{F}^{\text{EBE}}(\cdot)$ is a CNN model applied to each observation $\boldsymbol{J}_i$ for feature extraction. After obtaining $\boldsymbol{h}_i$, we explore three different backbone architectures, namely CNN, LSTM, and Transformer, to fuse the latent features across time steps.

Embedding-based CNN (ECNN): The stacked-based encoder passes the resulting sequence of latent feature vectors $\boldsymbol{h}_i$ through a 1D convolutional neural network. The CNN processes the latent features sequentially, using convolutional and pooling layers to extract spatial patterns and temporal relationships from the sequence. This captures informative features across the time dimension. Mathematically, the resulting $\boldsymbol{C}^{\text{ECNN}}$ can be given as:

$$\boldsymbol{C}^{\text{ECNN}} = \boldsymbol{F}^{\text{ECNN}}\left(\left[\boldsymbol{h}_{t-K-D+1}, \ldots, \boldsymbol{h}_{t-D}, \boldsymbol{h}_{t+1}\right]\right) \tag{17}$$

where $\boldsymbol{F}^{\text{ECNN}}(\cdot)$ represents a CNN model. In the sequel, this observation-encoding CNN-based model is referred to as ECNN-K.

Embedding-based LSTM (ELSTM): We replace CNN with LSTM to aggregate the latent features, aiming at capturing the long-term temporal dependencies between the latent features. Mathematically, the resulting $\boldsymbol{C}^{\text{ELSTM}}$ takes the following form:

$$\boldsymbol{C}^{\text{ELSTM}} = \boldsymbol{s}_{t-i+1} = \mathcal{L}^{\text{ELSTM}}\left(\boldsymbol{h}_{t-i}, \boldsymbol{s}_{t-i}\right), \tag{18}$$

where $\mathcal{L}^{\text{ELSTM}}(\cdot)$ stands for an LSTM model and $i \in \{K-D+1, \ldots, D, D+1\}$. In the sequel, this observation-encoding LSTM-based model is referred to as ELSTM-K.

Embedding-based Transformer (ETrans): Finally, the Transformer model [37] is employed to fuse the sequence of latent feature vectors. The Transformer is capable of encoding the entire input sequence simultaneously through the multi-head self-attention mechanism, allowing it to directly model interactions between all time steps. This parallel processing captures both local temporal patterns and global relationships in the sequence. More specifically, we first inject positional information into the latent features through fixed sinusoidal position encodings:

$$\tilde{\boldsymbol{h}}_i = \boldsymbol{h}_i + \text{PE}(i), \tag{19}$$

where PE(i) represents the position encoding [38] for the i-th time step and $i \in \{t-K-D+1, \ldots, t-D, t-D+1\}$.

These position-encoded vectors are then fed into the multi-head self-attention layer to model interactions between all time steps. For the l-th self-attention layer for $l = 1, \ldots, L-1$, the corresponding self-attention output $\boldsymbol{Z}'_t$ takes the following form:

$$\boldsymbol{Z}'_{l+1} = \text{MultiHeadAttn}\left(\boldsymbol{Z}^Q_l \boldsymbol{W}^Q_l, \boldsymbol{Z}^K_l \boldsymbol{W}^K_l, \boldsymbol{Z}^V_l \boldsymbol{W}^V_l\right), \quad (20)$$

where $\boldsymbol{W}^Q_l$, $\boldsymbol{W}^K_l$ and $\boldsymbol{W}^V_l$ denote the learned projections for the query, key, and value respectively. The self-attention output $\boldsymbol{Z}'_{l+1}$ is then processed by a position-wise feedforward layer and layer normalization:

$$\boldsymbol{Z}_{l+1} = \mathcal{M}\left(\text{LN}(\boldsymbol{Z}'_{l+1})\right) + \boldsymbol{Z}'_{l+1}. \quad (21)$$

Finally, the context vector $\boldsymbol{C}^{\text{ETrans}}$ is produced by the output of the last multi-attention layer as follows:

$$\boldsymbol{C}^{\text{ETrans}} = \text{MLP}(\text{LN}(\boldsymbol{Z}_L)). \quad (22)$$

In the sequel, the feature extractor model based on the encoded attention mechanism is referred to as ETrans-K.

D. ACTION REFINEMENT STAGE

In the scalable RL framework proposed for the TSC system, the action refinement stage constitutes a cornerstone to ensure the resilience of the traffic management system against fluctuation of data transmission delays. This stage is designed to complement the insights provided by the scene prediction and scene context encoder modules, thereby addressing the limitations that arise when these predictions are rendered less effective due to substantial delays. It serves as a robust backstop, invoking a set of pre-defined traffic rules to recalibrate the initial strategies proposed by the RL agent.

The methodology detailed in this subsection delineates the operational mechanics of the action refinement stage. It elucidates the process by which the framework reconciles the initial RL-derived actions with the real-time traffic context, employing a dual-threshold mechanism. This mechanism intervenes when the green light duration exceeds a specified threshold θ_t or when the occupancy rate falls below a predefined level θ_o, ensuring that traffic signals remain responsive to current conditions and historical congestion patterns. This strategic integration of rule-based adjustments is what imparts the framework with enhanced robustness, allowing it to adaptively recalibrate actions in the face of unpredictable communication delays. Algorithm 1 outlines the algorithmic structure of the action refinement stage.

Algorithm 1: Traffic Light Control Algorithm With Action Refinement.

```
Initialization : All phases: $\boldsymbol{\mathcal{P}} = \{\mathcal{P}_1, \ldots, \mathcal{P}_N\}$;
Time since last execution: $\boldsymbol{T}_p = [0, \ldots, 0] \in \mathbb{R}^N$;
Longest unreached time threshold: $\theta_t$;
Occupancy threshold: $\theta_o$;
Time between actions: $\Delta T$;
Input : Action output by stage 1: $a_t$ at time $t$;
Occupancy values: $\boldsymbol{O}_t = [o_1, \ldots, o_N]$ for each phase at time $t$;
Output : Action after refinement $a^r_t = a_t$;
// Refinement original agent action $a_t$
for each $\mathcal{P}_i$ in $\boldsymbol{\mathcal{P}}$ do
    if $\boldsymbol{T}_p[i] \geq \theta_t$ then
        $a^r_t \leftarrow i$;
    end
    else
        if $\boldsymbol{O}_t[a_t] \geq \theta_o$ then
            $a^r_t \leftarrow a_t$;
        end
        else
            $\mathcal{P}_{\text{congested}} \leftarrow \arg\max_{\mathcal{P}_j \in \boldsymbol{\mathcal{P}}} \boldsymbol{O}_t[j]$;
            $a^r_t \leftarrow \mathcal{P}_{\text{congested}}$;
        end
    end
end
// Update $\boldsymbol{T}_p[i]$ since last execution
for each $\mathcal{P}_i$ in $\boldsymbol{\mathcal{P}}$ do
    if $\mathcal{P}_i = a^r_t$ then
        $\boldsymbol{T}_p[i] \leftarrow 0$;
    end
    else
        $\boldsymbol{T}_p[i] \leftarrow \boldsymbol{T}_p[i] + \Delta t$;
    end
end
Wait for $\Delta T$ seconds;
```

E. POLICY AND LEARNING-BASED PREDICTION MODEL UPDATE

In this study, the PPO algorithm [39] is utilized to train the TSC agent, employing dual neural networks: a policy network $\pi_{\boldsymbol{\Psi}}$ and a value network $V_{\boldsymbol{\Phi}}$, parameterized by $\boldsymbol{\Psi}$ and $\boldsymbol{\Phi}$ respectively. Post-processing by the scene prediction and scene context encoder modules yields a context vector $\boldsymbol{C}$, which informs the policy network's action distribution and the value network's expected return estimation. The PPO objective is optimized through a clipped surrogate objective $L^{CLIP}(\boldsymbol{\Psi})$, enhancing reward acquisition, coupled with a value function loss $L^{VF}(\boldsymbol{\Phi})$, which refines state value predictions, thereby addressing the latency challenges inherent in traffic data acquisition and processing.

$$J(\boldsymbol{\Psi}, \boldsymbol{\Phi}) = \hat{\mathbb{E}}_t\left[L^{\text{CLIP}}(\boldsymbol{\Psi}) - c_1 L^{\text{VF}}(\boldsymbol{\Phi})\right], \quad (23)$$

where c_1 balances the two loss terms, and the PPO clip loss function $L^{\text{CLIP}}(\boldsymbol{\Psi})$ can be represented as:

$$L^{\text{CLIP}}(\boldsymbol{\Psi}) = \hat{\mathbb{E}}_t\left[\min(\rho_t(\boldsymbol{\Psi})A_t, \text{clip}(\rho_t(\boldsymbol{\Psi}), 1-\epsilon, 1+\epsilon)A_t)\right], \quad (24)$$

where $A_t = r_{t+1} + \gamma V(\boldsymbol{C}_{t+1}) - V(\boldsymbol{C}_t)$ is used to mean dominance. $\rho_t(\theta)$ denotes the ratio of the current strategy to the

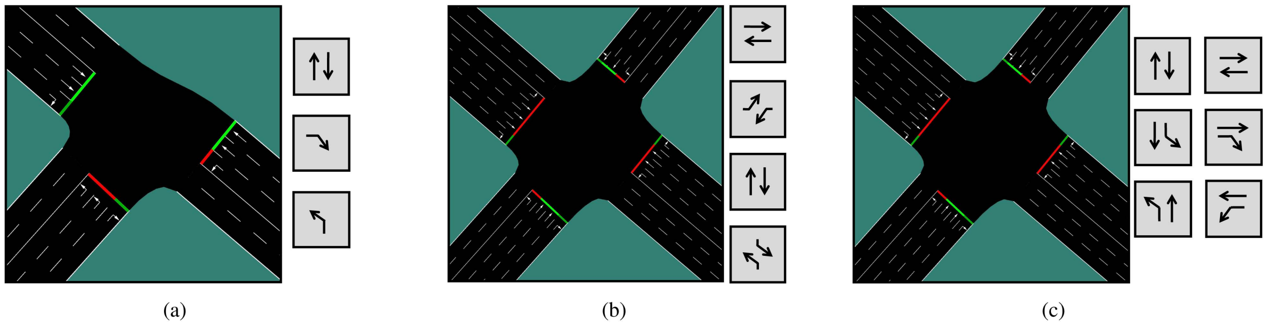


FIGURE 8. Three SUMO junctions used in this study. (a) T-Junction with three phases; (b) 4-Way Junction with four phases; (c) 4-Way Junction with six phases.

past strategy, as expressed in (25).

$$\rho_t(\Psi) = \frac{\pi_\Psi(a_t|\boldsymbol{C}_t)}{\pi_{\Psi_{old}}(a_t|\boldsymbol{C}_t)}, \tag{25}$$

where $\pi_\Psi(a_t|\boldsymbol{C}_t)$ denotes the current strategy and $\pi_{\Psi_{old}}(a_t|\boldsymbol{C}_t)$ denotes the past strategy.

The loss of the value function $L^{VF}(\Phi)$ is typically calculated as the mean square error between the predicted state values $V_\Phi(\boldsymbol{C})$ and the actual discounted returns:

$$L^{VF}(\Phi) = \frac{1}{N}\sum_{\boldsymbol{C}}\left(V_\phi(\boldsymbol{C}) - \sum_{t=0}^{T}\gamma^t r_t\right)^2, \tag{26}$$

where N is the number of sampled states, T is the time horizon, γ is the discount factor, and r_t is the reward received at timestep t. Minimizing this loss encourages the value network V_ϕ to accurately predict state values, which helps the agent estimate long-term rewards.

To update the LSTM network $\boldsymbol{\Theta}$ mentioned in (12), we calculate the mean square loss between the predicted traffic state and the actual observed state. It can be formulated as follows:

$$\mathcal{L}(\boldsymbol{\Theta}) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{\boldsymbol{S}}_{t+1}^{(i)} - \boldsymbol{S}_{t+1}^{(i)}\right)^2, \tag{27}$$

where N is the number of samples in the data buffer, $\hat{\boldsymbol{S}}_{t+1}^{(i)}$ is the predicted traffic state for the i-th state, and $\boldsymbol{S}_{t+1}^{(i)}$ is the corresponding actual observed traffic state.

V. EXPERIMENTS AND RESULT ANALYSIS

In this section, we introduce the experimental setup, benchmark methods, and performance of our two-stage RL-based TSC framework. A series of comprehensive experiments were conducted to assess the robustness of our model under varying levels of communication delay. Comparative analyses reveal that stage 1 of our framework exhibits enhanced performance in scenarios with minimal delay, while stage 2 demonstrates superior efficacy when confronted with more substantial delays. These findings underscore the individual modules' contributions to the system's overall adaptability and effectiveness in delay-prone environments.

A. EXPERIMENT SETTING

Three road network configurations were chosen to evaluate the proposed RL-based TSC system: a three-phase T-junction (ENV-1), a four-phase intersection (ENV-2), and a six-phase intersection (ENV-3), as illustrated in Fig. 8. These setups capture a range of complexity within traffic management tasks, facilitating a robust assessment of our framework. Simulations were conducted on the SUMO platform [40], with the acknowledgment that this methodology is extendable to other intersection types and traffic signal phases.

Control intervals for the traffic agents were set at 5 s, with each green light phase transitioning to red after a 3 s yellow light. Delays were systematically introduced at $D = 0, 5, 10, 15, 20, 40, 60, 80, 100$ seconds to determine the impact on system performance, measured by the vehicles' average waiting time. The simulation parameters included a detection range of 100 meters and an average vehicle occupancy length of 7 meters within a lane. Policy updates were calibrated with parameters $c_1 = 1$, discount factor $\gamma = 0.99$, and mini-batch size $M = 256$.

B. BENCHMARK METHODS

Our evaluation encompasses two benchmark methods: three traditional TSC algorithms and one RL-based approach, detailed as follows:

1) Fixed duration control: Fixed duration signal control algorithm is the most traditional method of traffic signal control by setting the traffic signal phase sequence, phase duration, and cycle time in advance;
2) The Webster model [17]: The Webster model is designed to minimize the average waiting time for vehicles. By estimating vehicle delays at traffic intersections, this model derives a set of timing parameters by optimizing the calculation of the phase cycle length.
3) The SOTL model [16]: Self-organizing traffic lights (SOTL) algorithms utilize straightforward rules and indirect communication to enable traffic lights to autonomously organize and adjust to dynamic traffic conditions, leading to a reduction in waiting times, a decrease in the number of stops, and an increase in average speeds.

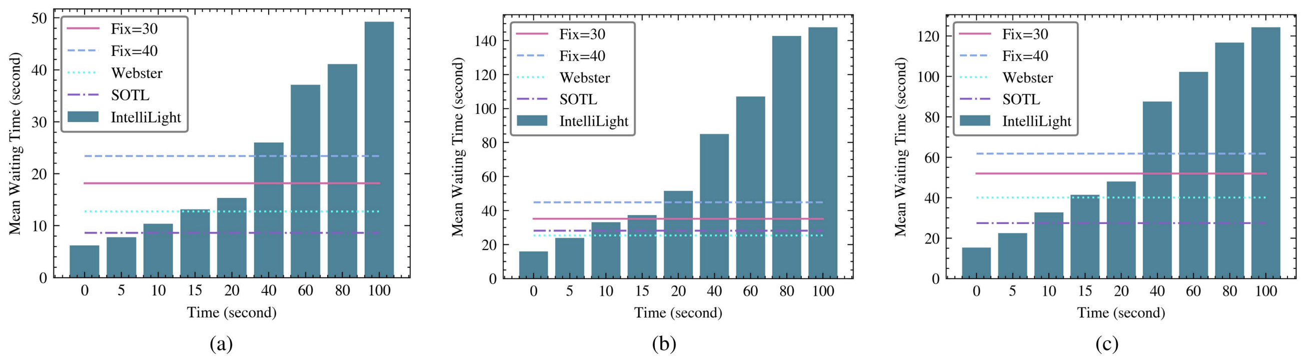


FIGURE 9. Performance comparison of the benchmark model with Fix time = 30 s, Fix time = 40 s, Webster model, SOTL model, and RL model for the three environments. (a) T-junction. (b) Four phases at the intersection. (c) Six phases at the intersection.

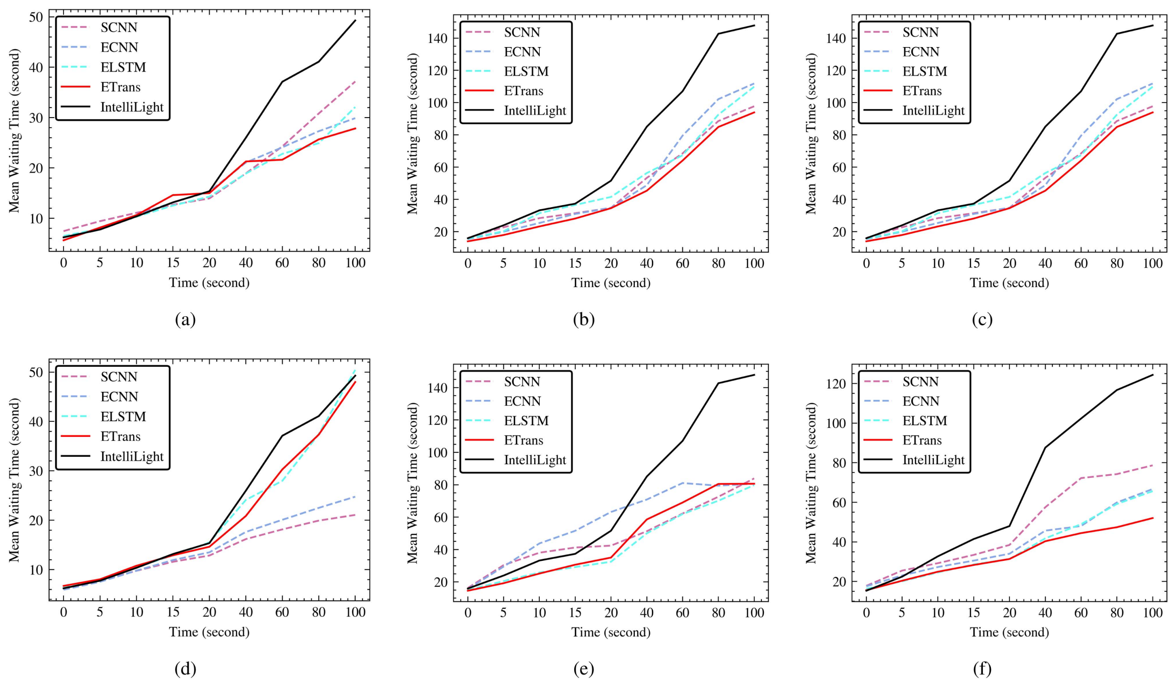


FIGURE 10. Framework experiment results. (a)–(c) Model performance with rule-based prediction model in three environments. (d) and (e) Model performance with learning-based prediction model in three environments.

4) IntelliLight [7]: This RL-based TSC method incorporates queue lengths, vehicle counts, and updated waiting times for each lane into its state representation. To ensure comparability, we align the agent's action space with that of our proposed framework, allowing selection from all available signal phases.

C. EXPERIMENTAL RESULTS

1) TSC SYSTEM PERFORMANCE DETERIORATION UNDER OBSERVATION DELAY

The experimental results of the four baseline algorithms under observation delay are depicted in Fig. 9. Across all three environments, the average waiting time grows with increasing observation delay for the baseline RL approach. At $D = 40$ s, the RL model performs worse than traditional models, with an average waiting time of 66.65 s, which is five times higher than the average waiting time in the absence of observation delay. The experiments indicate that, without observation delay, the performance of the reinforcement learning algorithm is better than fixed-time algorithms and shows an improvement of over 10% compared to the SOTL. However, under observation delay, the performance of the RL approach degrades significantly. In practical applications, the observation delay in the TSC system can have a substantial impact on the performance of the RL model, highlighting the need to address this issue.

2) ROBUST RL-BASED TSC SYSTEM

Fig. 10(a), (b), (c), and (d), (e), (f) respectively illustrate the performance of four scene context encoder modules combined with rule-based and learning-based predictions in three environments. In environments with observation

TABLE 1. Model Ablation Experiments With Rule-ETrans Model and Action Refinement

Stage-1		Stage-2	ENV-1			ENV-2			ENV-3		
Rule-P	ETrans	Action Refinement	D=0s	D=20s	D=100s	D=0s	D=20s	D=100s	D=0s	D=20s	D=100s
✓			6.84	13.42	31.78	15.33	40.62	117.11	15.25	34.19	148.99
	✓		6.47	16.34	49.22	14.80	38.14	93.60	14.13	39.81	137.85
✓	✓		6.52	14.82	40.59	14.40	39.59	147.30	14.16	35.94	126.80
✓		✓	7.46	**13.95**	37.16	16.03	34.75	97.73	16.95	35.75	130.35
	✓	✓	6.35	15.43	35.62	14.82	38.45	**90.72**	14.34	38.29	103.25
✓	✓	✓	**5.63**	14.99	**27.83**	**14.22**	**34.54**	93.93	**14.1**	**35.67**	**79.34**

delays, the proposed framework generally outperforms the RL baseline method. At $D = 20$ s, the proposed framework has an average improvement in performance of 20.6%. When $D = 100$ s, the average performance improvement reaches an impressive 38.2%.

Compared to learning-based predictions, rule-based predictions exhibit better stability. According to experimental data, at $D = 100$ s, the performance of the rule-based prediction model averages 2% higher than that of the learning-based prediction model. This may be attributed to two factors: First, in traffic prediction, rule-based prediction has unique advantages, as it can accurately infer future states based on existing traffic rules. Second, training in learning-based prediction is more complex, and here, we only employed the commonly used LSTM model. Achieving better results may require more sophisticated models.

Among the four models of scene context encoders proposed, ETrans demonstrates superior performance. At $D = 100$ s, the scene context encoder using the ETrans model shows a remarkable performance improvement of 38.9% compared to the baseline model. However, since the models are trained under unified parameters, and the second stage also influences the final results, ETrans does not exhibit a consistently stable performance improvement in the presented results.

D. ABLATION EXPERIMENT

Conducting ablation experiments allows us to study the effectiveness of each component in the proposed scalable RL framework. To effectively carry out these experiments, based on prior trials, we selected the best-performing models, namely the rule-based prediction for unobserved scene prediction and the ETrans model for scene context encoder. The results are shown in Table 1. The overall performance of the framework is expected to surpass the individual performance of each module. A detailed analysis of each module is provided below.

Effects of the scene prediction module. As shown in Table 1, the ETrans model with rule-based prediction demonstrates a 13.8% performance improvement at $D = 100$ s. Particularly in ENV-3, there is a 23.2% performance enhancement at $D = 100$ s. This indicates that the scene prediction module can enhance the overall performance of the framework. However, in individual cases, performance degradation is observed, possibly due to the lack of specific adjustments to the prediction module for different environments. This leads to a decrease in module performance at high delays. Nevertheless, in the majority of cases, unobserved scene prediction module contributes to the overall improvement in framework performance.

Effects of the scene context encoder module. The scene context encoder module excels in extracting features from time-series observations, thereby mitigating the impact of observation delay. As shown in Table 1, our proposed ETrans model demonstrates a significant enhancement in the performance of the RL-based model. With the incorporation of ETrans, performance improves by 26.6% compared to the baseline at $D = 20$ s. The scene context encoder module contributes to a general performance improvement of over 10% in the framework.

Although unobserved scene prediction module can demonstrate excellent performance, having only the scene context encoder module can already lead to a significant improvement. As shown in Table 1, at $D = 100$ s, having only the scene context encoder module can enhance the overall performance by 27.8%, with the overall framework performance improving by 38.6%. Not using the scene prediction module can also achieve 72% of the overall performance of the framework. Having only the scene context encoder in the first stage can significantly reduce model complexity, as the scene context encoder can be trained together with the RL agent.

Effects of the action refinement. The action refinement is used to assess the actions of the RL agent. It can be observed that at $D = 100$ s, this module plays a crucial role, as having the action refinement module further improves the overall performance by 34.8%. However, action refinement does not lead to a noticeable performance improvement at lower delays because when the observation delay is low, the RL Agent's performance can remain stable, and most strategies are still effective. In contrast, when the delay is significant, the observations obtained by the RL agent deviate significantly from the actual scenario, making it unable to derive effective strategies. In such cases, the system might collapse, and action refinement is needed to maintain system stability.

E. PARAMETRIC ANALYSIS

1) OBSERVATION FRAME LENGTH

The length of the observations used by the RL agent significantly impacts the model's performance. It is generally believed that choosing a longer observation time allows the model to learn more temporal features, resulting in better performance. For the observation frame length test, we choose the ETrans model, which demonstrates superior performance

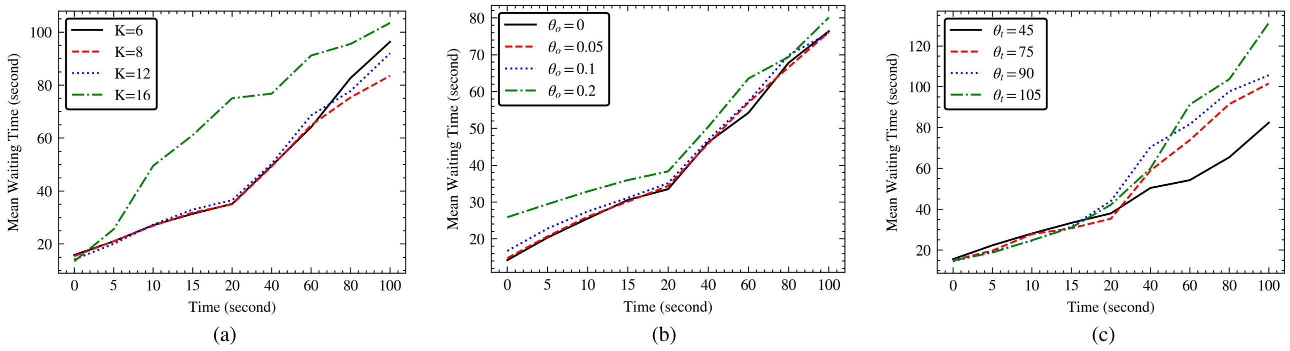


FIGURE 11. Parametric analysis results. (a) Model performance of ETrans with different observation frames. (b) Model performance for different values of θ_o. (c) Model performance for different values of θ_t.

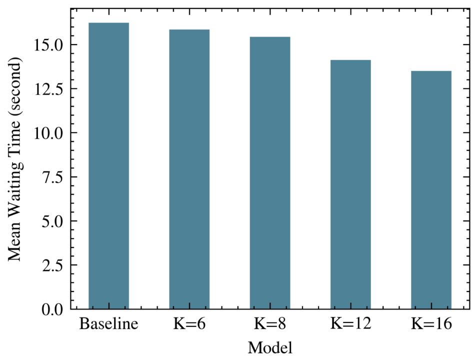


FIGURE 12. Impact of observation length on the performance of the scene context encoder in the absence of observation delay.

in the experiment presented earlier. The experimental results are illustrated in Fig. 12. With the increase in the number of observation frames acquired by the RL agent, the model's performance gradually improves. Particularly, at $D = 0$ s, the performance with $K = 16$ is enhanced by 14.7% compared to $K = 6$, and when compared to the baseline model, there is an approximately 16.8% performance improvement. This performance is significantly superior to traditional methods.

However, with the increase in observation delay, ETrans-16 fails to sustain its optimal performance, as depicted in Fig. 11(a). Notably, ETrans-8 exhibits the most robust performance, achieving a 6.2% enhancement over $K = 12$ and a significant 21.3% improvement over $K = 16$ at $D = 100$ s. The underlying reason resides in the escalating dimensionality of the feature data as the number of observed frames increases. Training the RL model becomes challenging in a high-dimensional feature space. Although ETrans-16 shows superior performance without delay time, the agent encounters numerous unprecedented situations with increasing delay time, hindering effective command decisions. Therefore, in the case of observation delays, the model's performance does not improve with the increase in observed frames.

2) ACTION REFINEMENT PARAMETER θ_o

The hyperparameters of action refinement include $\boldsymbol{\theta}_t$ and $\boldsymbol{\theta}_o$, where $\boldsymbol{\theta}_t$ represents a time threshold for the longest unselected phase, and $\boldsymbol{\theta}_o$ denotes the lane occupancy threshold for phase selection. In this context, an analysis is conducted on $\boldsymbol{\theta}_o$ to assess its impact on the framework's performance. $\boldsymbol{\theta}_o$ is pivotal in determining the accuracy of actions; if set too low, it may hinder the effectiveness of the safety module, while setting it too high may lead to misjudgments in action generation. The results are presented in Fig. 11(b).

When there is no observation delay, increasing $\boldsymbol{\theta}_o$ significantly contributes to an increase in average waiting time. This is because the decision-making accuracy of the ETrans model is optimal when there is no delay. However, when $\boldsymbol{\theta}_o$ is excessively large, it leads to miscalculations, causing the Action Refinement model to make misjudgments. For instance, the model's performance severely degrades when $\boldsymbol{\theta}_o = 0.2$ compared to when $\boldsymbol{\theta}_o = 0$, resulting in an 80% increase in average waiting time. On the other hand, setting $\boldsymbol{\theta}_o = 0$ is also suboptimal. For example, at $D = 80$ s, $\boldsymbol{\theta}_o = 0.05$ improves performance by 1.5% compared to $\boldsymbol{\theta}_o = 0$.

3) ACTION REFINEMENT PARAMETER θ_t

$\boldsymbol{\theta}_t$ represents the time threshold for the longest unselected phase, and its impact on model performance is closely related to the extent of observation delay. In situations with small observation delays, the RL agent exhibits effective decision-making. However, setting $\boldsymbol{\theta}_t$ too small can impact the RL agent's strategy and, consequently, its overall performance. Illustrated in Fig. 11(c), when the observation delay is $D = 10$ s, configuring $\boldsymbol{\theta}_t = 105$ s yields a 13% performance improvement compared to $\boldsymbol{\theta}_t = 45$ s. Conversely, in scenarios with significant delays, the RL agent's strategy may no longer align effectively with the intersection state, diminishing its effectiveness. A smaller $\boldsymbol{\theta}_t$ enhances the overall responsiveness of the framework, thereby alleviating the impact of high delays. For instance, when $D = 100$ s, configuring $\boldsymbol{\theta}_t = 45$ s results in a substantial 46% performance improvement compared to $\boldsymbol{\theta}_t = 105$ s.

VI. CONCLUSION

In this study, we investigated a scalable RL traffic signal control framework, with explicit consideration of the impact of observation delays. More specifically, a novel two-stage reinforcement learning control framework has been developed, consisting of an action generation stage and an action refinement stage. In the action generation stage, two modules are designed: a scene predictor and a scene context encoder. In the action refinement stage, the initial RL decisions are calibrated based on general traffic rules, effectively avoiding disastrous decision recommended by the RL agent when facing delayed observations, which is shown to enhance the overall robustness of the framework. Extensive experiments on a four-phase intersection environment have confirmed that the proposed RL-based intelligent traffic light control system exhibits a significant reduction of 43.03% in average waiting time as compared to the baseline, with an observation delay of $D = 100$ s. Notably, the model performs well even without explicit knowledge of the observation delay time, which is of practical significance.

REFERENCES

[1] M. Noor-A-Rahim et al., "6G for vehicle-to-everything (V2X) communications: Enabling technologies, challenges, and opportunities," *Proc. IEEE*, vol. 110, no. 6, pp. 712–734, Jun. 2022.

[2] M. A. Salman et al., "Traffic signal control with vehicle-to-everything communication," in *Proc. Int. Conf. Comput. Sci. Inf. Technol.*, 2017, pp. 57–69.

[3] P. Koonce and L. Rodegerdts, "Traffic signal timing manual," United States. Federal Highway Administration, Washington, DC, USA, Tech. Rep. FHWA-HOP-08-024, 2008.

[4] P. Lowrie, "Scats-a traffic responsive method of controlling urban traffic," in *Sales Information Brochure*, Sydney, Australia: Roads & Traffic Authority, 1990.

[5] N. Nigam, D. P. Singh, and J. Choudhary, "A review of different components of the intelligent traffic management system (ITMS)," *Symmetry*, vol. 15, no. 3, 2023, Art. no. 583.

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

[7] H. Wei, G. Zheng, H. Yao, and Z. Li, "IntelliLight: A reinforcement learning approach for intelligent traffic light control," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2496–2505.

[8] S. M. A. Shabestary and B. Abdulhai, "Adaptive traffic signal control with deep reinforcement learning and high dimensional sensory inputs: Case study and comprehensive sensitivity analyses," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 20021–20035, Nov. 2022.

[9] M. Wang, Y. Xu, X. Xiong, Y. Kan, C. Xu, and M.-O. Pun, "ADLight: A universal approach of traffic signal control with augmented data using reinforcement learning," in *Proc. Transp. Res. Board Annu. Meeting*, 2023.

[10] P. L. Nguyen, R.-H. Hwang, P. M. Khiem, K. Nguyen, and Y.-D. Lin, "Modeling and minimizing latency in three-tier v2x networks," in *zproc. IEEE Glob. Commun. Conf.*, 2020, pp. 1–6.

[11] P. Zhao and H. Hu, "Geographical patterns of traffic congestion in growing megacities: Big data analytics from Beijing," *Cities*, vol. 92, pp. 164–174, 2019.

[12] Ž. Majstorović, L. Tišljarić, E. Ivanjko, and T. Carić, "Urban traffic signal control under mixed traffic flows: Literature review," *Appl. Sci.*, vol. 13, no. 7, 2023, Art. no. 4484.

[13] S. Samadi et al., "Performance evaluation of intelligent adaptive traffic control systems: A case study," *J. Transp. Technol.*, vol. 2, no. 3, pp. 248–259, 2012.

[14] A. L. Bazzan and F. Klügl, *Introduction to Intelligent Systems in Traffic and Transportation*. Berlin, Germany: Springer, 2022.

[15] C. G. Garcia, "Self-organizing traffic lights," *Complex Syst.*, vol. 1, no. 16, pp. 29–53, 2005.

[16] S.-B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," in *Advances in Applied Self-Organizing Systems*, Berlin, Germany: Springer, 2013, pp. 45–55.

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

[18] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1243–1253, Feb. 2019.

[19] M. Noaeen et al., "Reinforcement learning in urban network traffic signal control: A systematic literature review," *Expert Syst. Appl.*, vol. 199, 2022, Art. no. 116830.

[20] M. Wang, X. Xiong, Y. Kan, C. Xu, and M.-O. Pun, "UniTSA: A universal reinforcement learning framework for V2X traffic signal control," 2023, *arXiv:2312.05090*.

[21] M. Yazdani, M. Sarvi, S. A. Bagloee, N. Nassir, J. Price, and H. Parineh, "Intelligent vehicle pedestrian light (IVPL): A deep reinforcement learning approach for traffic signal control," *Transp. Res. Part C: Emerg. technol.*, vol. 149, 2023, Art. no. 103991.

[22] Y. Li, J. He, and Y. Gao, "Intelligent traffic signal control with deep reinforcement learning at single intersection," in *Proc. 7th Int. Conf. Comput. Artif. Intell.*, 2021, pp. 399–406.

[23] S. Wang, X. Xie, K. Huang, J. Zeng, and Z. Cai, "Deep reinforcement learning-based traffic signal control using high-resolution event-based data," *Entropy*, vol. 21, no. 8, 2019, Art. no. 744.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[25] G. Dulac-Arnold et al., "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Mach. Learn.*, vol. 110, no. 9, pp. 2419–2468, 2021.

[26] C. Gulcehre et al., "Rl unplugged: A suite of benchmarks for offline reinforcement learning," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 7248–7259, 2020.

[27] M. Agarwal and V. Aggarwal, "Blind decision making: Reinforcement learning with delayed observations," *Pattern Recognit. Lett.*, vol. 150, pp. 176–182, 2021.

[28] E. Schuitema, L. Buşoniu, R. Babuška, and P. Jonker, "Control delay in reinforcement learning for real-time dynamic systems: A memoryless approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2010, pp. 3226–3231.

[29] H. Zhao, B. Wang, H. Liu, H. Sun, Z. Pan, and Q. Guo, "Exploiting the flexibility inside park-level commercial buildings considering heat transfer time delay: A memory-augmented deep reinforcement learning approach," *IEEE Trans. Sustain. Energy*, vol. 13, no. 1, pp. 207–219, Jan. 2022.

[30] H. Peng et al., "Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning," *Inf. Sci.*, vol. 578, pp. 401–416, 2021.

[31] M. Abdoos and A. L. Bazzan, "Hierarchical traffic signal optimization using reinforcement learning and traffic prediction with long-short term memory," *Expert Syst. Appl.*, vol. 171, 2021, Art. no. 114580.

[32] J. Jiang, C. Han, W. X. Zhao, and J. Wang, "PDFormer: Propagation delay-aware dynamic long-range transformer for traffic flow prediction," 2023, *arXiv:2301.07945*.

[33] N. Zhang, X. Yang, H. Guo, H. Dong, and W. Ma, "Approximate inference of traffic flow state at signalized intersections using a Bayesian learning framework," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 4765–4776, May 2023.

[34] A. Pang, Z. Xu, M. Wang, M.-O. Pun, and Y. Kan, "Reinforcement learning-based traffic signal control using delayed observations for V2X," in *Proc. IEEE Int. Conf. Commun.*, 2023, pp. 4020–4025.

[35] H. Wang, J. Zhu, and B. Gu, "Model-based deep reinforcement learning with traffic inference for traffic signal control," *Appl. Sci.*, vol. 13, no. 6, 2023, Art. no. 4010.

[36] B. Peng, X. Li, J. Gao, J. Liu, K.-F. Wong, and S.-Y. Su, "Deep Dyna-Q: Integrating planning for task-completion dialogue policy learning," in *Proc. 56th Annu. Meeting Assoc. Computat. Linguistics*, 2018, pp. 2182–2192.

[37] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2020.

[38] P. Dufter, M. Schmitt, and H. Schütze, "Position information in transformers: An overview," *Comput. Linguistics*, vol. 48, no. 3, pp. 733–763, 2022.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[40] P. A. Lopez et al., "Microscopic traffic simulation using sumo," in *Proc. 21st Int. Conf. Intell. Transp. Syst.*, 2018, pp. 2575–2582.

AOYU PANG received the B.Eng. degree in automatization from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2023. He is currently working toward the Ph.D. degree with the Computer and Information Engineering, Chinese University of Hong Kong, Shenzhen, China. His research interests include V2X communications, intelligent transportation, and reinforcement learning.

MAONAN WANG received the B.Eng. degree in mathematics and applied mathematics from the Beijing University of Posts and Telecommunications, Beijing, China, in 2018, and the M.S. degree in cyberspace security in 2021. He is currently working toward the Ph.D. degree in computer and information engineering, Chinese University of Hong Kong, Shenzhen, China. His research interests include intelligent transportation system, internet of vehicles, and reinforcement learning.

YIRONG CHEN is currently working toward the Ph.D. degree with the Civil and Environmental Engineering Department, Stanford University, Stanford, CA, USA. His research interests include sustainable urban developments and innovative and multidisciplinary methods to model and improve urban traffic problems, including traffic signal control and traffic forecasting.

MAN-ON PUN (Senior Member, IEEE) received the B.Eng. degree in electronic engineering from The Chinese University of Hong Kong, Hong Kong, in 1996, the M.Eng. degree in computer science from the University of Tsukuba, Tsukuba, Japan, in 1999, and the Ph.D. degree in electrical engineering from the University of Southern California (USC) at Los Angeles, Los Angeles, CA, USA, in 2006. He was a Posdoctoral Research Associate with Princeton University, Princeton, NJ, USA, from 2006 to 2008. He held research positions with Huawei, NJ, USA, the Mitsubishi Electric Research Labs (MERL), Boston, MA, USA, and Sony, Tokyo, Japan. He is currently an Associate Professor with the School of Science and Engineering, CUHKSZ. His research interests include artificial intelligence (AI) Internet of Things (AIoT) and applications of machine learning in communications and satellite remote sensing. Prof. Pun was the recipient of the best paper awards from the IEEE Vehicular Technology Conference 2006 Fall, the IEEE International Conference on Communication 2008, and the IEEE Infocom'09. He is the Founding Chair of the IEEE Joint Signal Processing Society-Communications Society Chapter, Shenzhen. He is an Associate Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS from 2010 to 2014.

MICHAEL LEPECH is currently a Professor of the Department of Civil & Environmental Engineering with the Stanford University, Stanford, CA, USA. His research interests include integration of sustainability indicators into engineering design and the design of sustainable high performance fiber-reinforced cementitious composites and fiber-reinforced polymers (FRPs). Along with these, his research also focuses on creating multiscale, fundamental engineering tools that integrate with sustainability assessment and facilitate setting and meeting sustainability targets throughout the life cycle of constructed facilities.