Reinforcement Learning Approaches for Traffic Signal Control under Missing Data

Hao Mei[1] , **Junxian Li**[2] , **Bin Shi**[2] and **Hua Wei**[†1,3]

[1] New Jersey Institute of Technology
[2] Xi'an Jiaotong University
[3] Arizona State University

hm467@njit.edu, ljx201806@stu.xjtu.edu.cn, shibin@xjtu.edu.cn, hwei27@asu.edu

Abstract

The emergence of reinforcement learning (RL) methods in traffic signal control (TSC) tasks has achieved promising results. Most RL approaches require the observation of the environment for the agent to decide which action is optimal for a long-term reward. However, in real-world urban scenarios, missing observation of traffic states may frequently occur due to the lack of sensors, which makes existing RL methods inapplicable on road networks with missing observation. In this work, we aim to control the traffic signals in a real-world setting, where some of the intersections in the road network are not installed with sensors and thus with no direct observations around them. To the best of our knowledge, we are the first to use RL methods to tackle the TSC problem in this real-world setting. Specifically, we propose two solutions: 1) imputes the traffic states to enable adaptive control. 2) imputes both states and rewards to enable adaptive control and the training of RL agents. Through extensive experiments on both synthetic and real-world road network traffic, we reveal that our method outperforms conventional approaches and performs consistently with different missing rates. We also investigate how missing data influences the performance of our model.

1 Introduction

Traffic congestion has been a challenge in modern society and adversely affects economic growth, environmental sustainability, and people's quality of life. For example, traffic congestion costs an estimated $87 billion in lost productivity in the US alone [Tirone, 2022]. Recently, reinforcement learning (RL) has shown superior performance over traditional transportation approaches in controlling traffic signals in dynamic traffic [Arel *et al.*, 2010; El-Tantawy *et al.*, 2013; Oroojlooy *et al.*, 2020; Wei *et al.*, 2019a]. The biggest advantage of RL is that it directly learns to take adaptive actions in response to dynamic traffic by observing the states and feedback from the environment after previous actions.

Although a number of literature has focused on improving RL methods' performance in traffic signal control, RL cannot be directly deployed in the real world where accessible observations are sparse, i.e., some traffic states are missing [Chen *et al.*, 2019; Mai *et al.*, 2019]. In most cities where sensors are only installed at certain intersections, intersections without sensors cannot utilize RL and usually use pre-timed traffic signal plans that cannot adapt to dynamic traffic. Similar situations could happen when installed sensors are not properly functioning, which will lead to missing observations in the collected traffic states [Duan *et al.*, 2016] and the failure to deploy RL methods. Though there have been attempts such as using imitation learning to learn from the experience of human traffic engineers [Li *et al.*, 2020], these methods need manual design and cannot be easily extended to new scenarios. Thus the missing data issue still hinders not only the application of RL, but also the deployment of other adaptive control methods that require observing traffic states like Max-Pressure [Varaiya, 2013].

In this paper, we investigate the traffic signal control problem under the real-world setting, where the traffic condition around certain intersections are never observed. To enable dynamic control over these intersections, we investigate how data imputation could help the control, especially how the imputation on state and reward can remedy the missing data challenge for RL methods. With imputed states, adaptive control methods from transportation could be utilized; with both imputed states and rewards, RL agents could be trained for unobserved intersections. Inspired by model-based RL, we also investigate to use the imaginary rollout with reward model for better performance. The main contributions of this work are summarized as follows:

• To the best of our knowledge, we are the first to adapt RL-based traffic signal control methods under the missing data scenario, hence improving reinforcement learning method's applicability under more realistic settings. We test different kinds of approaches to control the intersections without observations. We propose a two-step approach that firstly imputes the states and rewards to enable the second step of RL training. The proposed approach can achieve better performance than only training RL agents at fully observed intersections or training RL agents on all intersections using only observed data without imputation and also outperforms using pre-timed control methods.

[†]Corresponding author.

• We investigate our methods under synthetic and real-world datasets with different missing rates and whether having neighboring unobserved intersections. And the studies on missing rates and relationships of missing positions show our methods perform better than pre-timed methods and are effective in real-world scenarios.

2 Related Work

Traffic signal control methods. Optimizing traffic signal control to alleviate traffic congestion has been a challenge in the transportation field for a long time. Different approaches have been extensively studied, including rule-based methods [HUNT, 1981; Sims and Dobinson, 1980; Varaiya, 2013] and RL-based methods [Arel *et al.*, 2010; Wei *et al.*, 2019a; Wei *et al.*, 2018] to optimize vehicle travel time or delay. Most of these studies, for example, IDQN method [Wei *et al.*, 2018], have significantly improved compared to pre-designed time control methods. However, to the best of our knowledge, there is no existing work on dealing with the unobserved intersections in dynamic traffic signal control methods.

Traffic data imputation. In real-world scenarios, full observation is not always accessible. An effective way to deal with the missing observations is data imputation, i.e., to infer the missing data to complete traffic observations. Earlier studies typically use historical data collected at each location to predict the values at missing positions of the same site [Gan *et al.*, 2015; Zhong *et al.*, 2004]. Recently, neural network-based methods have been proven effective and extended to be used in the traffic data imputation task [Lv *et al.*, 2015]. These methods could be categorized into Recurrent Neural Networks (RNNs) [Cui *et al.*, 2020; Yao *et al.*, 2018], Graph Neural Network (GNN) [Wang *et al.*, 2022] and Generative Adversarial Networks (GANs) [Zhang *et al.*, 2021]. However, all the methods mentioned above also need the observation of all intersections to train models, while in reality, it is hard to fulfill. Store-and-forward method (SFM) [Aboudolas *et al.*, 2009] is another approach to model traffic state transition and is often used as the base model traffic simulation.

Model-based reinforcement learning. Model-based reinforcement learning (MBRL) methods utilize predictive models of the environment on the immediate reward or transition to provide imaginary samples for RL [Luo *et al.*, 2022]. In MBRL with a reward model, an agent learns to predict the immediate reward of taking action at a certain state, and in this paper, we borrow this idea to train a reward model. In MBRL with a transition model, an agent usually has the direct observation of its own surrounding states, and a transition model is used to simulate the *next states* from current observations. Unlike MBRL with transition models, in this paper, some agents do not have observations of their own surroundings, where imputation methods are utilized to infer the *current states* (rather than next states) for those agents.

3 Preliminaries

In this section, we take the basic problem definition used in the multi-intersection traffic signal control [Wei *et al.*, 2019a] and extend it into the missing data scenario frequently encountered in the real world. An agent controls each intersection in the system. Given that only part of the agents can have their local observation of the total system condition as their state, we would like to proactively decide for all the intersections in the system which phases they should change to so as to minimize the average queue length on the lanes around the intersections. Specifically, the problem is characterized by the following major components $\langle \mathcal{S}, \hat{\mathcal{S}}, \mathcal{A}, r, \Pi, \gamma \rangle$:

• Observed state space $\mathcal{S}$ and imputed state space $\hat{\mathcal{S}}$. We assume that the system consists of a set of intersections $\mathcal{N} = \mathcal{N}_o \cup \mathcal{N}_m$, where $\mathcal{N}_o$ is the set of intersections where the agent can observe part of the system as its state $s \in \mathcal{S}$, and $\mathcal{N}_m$ is the set of intersections where the agent cannot observe the system. We follow setting from past works [Wei *et al.*, 2019b; Wu *et al.*, 2021; Huang *et al.*, 2021], and define s_t^j for agent $j \in \mathcal{N}_o$ at time t, which consists of its current phase (which direction is in green light) and the number of vehicles on each lane at time t. Later we will introduce unobserved agent $k \in \mathcal{N}_m$, and how we can infer its state $\hat{s}_t^k \in \hat{\mathcal{S}}$ at time t.

• Set of actions $\mathcal{A}$. In the traffic signal control problem, at time t, an agent i would choose an action a_t^i from its candidate action set $\mathcal{A}_i$ as a decision for the next Δt period of time. Here, we take acyclic control method, in which each intersection would choose a phase p as its action a_t^i from its pre-defined phase set, indicating that from time t to $t + \Delta t$, this intersection would be in phase p.

• Reward r. Each agent i obtains an immediate reward r_t^i from the environment at time t by a reward function $\mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \to \mathbb{R}$. In this paper, we want to minimize the travel time for all vehicles in the system, which is hard to optimize directly. Therefore, we define the reward for intersection i as $r_t^i = -\Sigma_l u_t^{i,l}$ where $u_t^{i,l}$ is the queue length on the approaching lane l at time t. Specifically, we denote r_t^j as the observed reward for agent $j \in \mathcal{N}_o$ at time t, and the inferred reward $\hat{r}_t^k$ as the reward for agent $k \in \mathcal{N}_m$ at time t.

• Policy set Π and discount factor γ. Intuitively, the joint actions have long-term effects on the system, so we want to minimize the expected queue length of each intersection in each episode. Specifically, at time t, each agent chooses an action following a certain policy $\pi : \mathcal{S} \to \mathcal{A}$.

An RL agent follows policy $\pi_\theta \in \Pi$ parameterized by θ, aiming to maximize its total reward $G_t^i = \Sigma_{\tau=t}^T \gamma^{\tau-t} r_t^i$, where T is total time steps of an episode and $\gamma \in [0, 1]$ differentiates the rewards in terms of temporal proximity. Other rule-based agents are denoted as $\pi_\varnothing \in \Pi$.

Problem 1 (Traffic signal control under missing data). *Given a road network where only part of the intersections is observed with $\mathcal{S}$, the goal of this paper is to find a better Π, no matter whether it consists of π_θ, $\pi_\varnothing$ or mixed policies of previous two kinds, that can minimize the average travel time of all vehicles.*

In the RL framework, training and execution are two decoupled phases: (1) During *execution*, an agent takes actions based on its policy π to roll out trajectories and evaluate their performances. For policies that take the current state as input, the agent can execute adaptive actions as long as the input

states are available. For observed intersections $\mathcal{N}_o$, the input states could directly be observed state $s \in \mathcal{S}$; for unobserved intersections $\mathcal{N}_m$, the input states could be inferred $\hat{s} \in \hat{\mathcal{S}}$ using data imputation. (2) In the *training* phase, agents explore the environment, store experiences in the replay buffer, and update their policies to maximize their long-term rewards. The experiences usually consist of state s_t, reward r_t, action a_t, and next state s_{t+1}. Different from *execution* phase, which only requires the input states, the *training* phase of RL requires reward information. For unobserved intersections $\mathcal{N}_m$, the $\hat{r}_t$ could also be inferred with data imputation on the reward. Later in Sec. 4, we will introduce how the missing data in the training and execution phase would influence the design of methods to tackle the traffic signal control problem.

4 Methods under Missing Data

Adaptive control methods like MaxPressure and RL-based methods dynamically adjust traffic signals based on real-time traffic state observations, which have been proven to work well on traffic signal control tasks. However, in the real-world scenario, these adaptive control methods cannot work properly at intersections $\mathcal{N}_m$ where real-time observations are missing. To adapt dynamic control methods to real-world, we explore the conventional approach and propose two effective imputation approaches to handle the failure of adaptive control at $\mathcal{N}_m$. The overall frameworks are shown in Figure 1.

4.1 Conventional Approaches

Under the missing data scenario, there are three direct approaches for traffic signal control: (1) pre-timed control, which sets fixed timings for all intersections; (2) the mixed control of RL and pre-timed agents, which uses RL agents only at observed intersections $\mathcal{N}_o$ and deploys pre-timed agents at unobserved intersections $\mathcal{N}_m$; (3) neighboring RL control, where agents at unobserved intersections $\mathcal{N}_m$ concatenate states from neighboring observed intersections as their own state and accumulate rewards from their neighboring observed intersections as their reward. This approach follows the general solution to the Partially Observable Markov Decision Process (POMDP) and assumes the traffic condition from observed neighboring intersections could reflect the unobserved traffic condition. This assumption might not hold when the number of missing intersections increases or the traffic is dynamic and complex.

4.2 Remedy 1: Imputation over Unobserved States

To enable $\mathcal{N}_m$ with dynamic control during execution, a natural solution is to impute unobserved states $\hat{s}_t^k$ at $\mathcal{N}_m$ for control methods. After the imputation of the states at unobserved intersections, dynamic control methods can be applied.

Imputation. Since the state information at unobserved intersections is totally missing, it is inapplicable to train a model on data collected from unobserved intersections and recover unobserved states. Therefore, we need to pretrain a state imputation model that will be shared by all the unobserved intersections and apply it during the training of RL. Intuitively, vehicles currently on each lane are aggregated from its up-streaming connected lanes in the previous time step.

Given the states of neighboring intersections $\mathcal{V}^k$ of intersection $k \in \mathcal{N}_m$, the state imputation at k can be formally defined as follows:

$$\hat{s}_t^k = f(\mathcal{V}_{t-1}^k) \tag{1}$$

where f could be any state imputation model. In this paper, we investigate two pre-trained models, a rule-based Store-and-Forward model (SFM) and a neural network model. Their detailed descriptions can be found in Sec. 5.1.

Control. After imputation, we investigate two control approaches that can function during execution.

• *Approach 1: Adaptive control methods in transportation.* Adaptive control methods in transportation usually require observation of the surrounding traffic conditions to decide the action for traffic signals. Without imputation, these methods cannot be applied directly. In this paper, following [Wei *et al.*, 2019a; Chen *et al.*, 2020], we use one widely used adaptive control method, MaxPressure [Varaiya, 2013], to control traffic signals for unobserved intersections after imputation.

• *Approach 2: Transferred RL models.* Another method is to enable RL-based control at missing intersections by training an RL policy π_θ at observed intersections $\mathcal{N}_o$ and later transferring to $\mathcal{N}_m$ during execution. Since all agents share the same policy, we refer to this model-sharing agent as SDQN for later use. During execution, agent $j \in \mathcal{N}_o$ can directly use the states s_t^j observed from the environment to take action $\pi_\theta(a_t^j|s_t^j)$. For agent $k \in \mathcal{N}_m$, it first imputes states $\hat{s}_t^k$ and then takes action $\pi_\theta(a_t^k|\hat{s}_t^k)$ based on the imputed states $\hat{s}_t^k$. In this solution, we use $(s_t^j, r_t^j, a_t^j, s_{t+1}^j)$ from all the observed intersections $j \in \mathcal{N}_o$ as experiences to train an RL model shared by all intersections. This approach can significantly improve sample efficiency, and all training samples can reflect the true state of the environment. However, since the agent is only trained on the experiences from observed intersections $\mathcal{N}_o$, it might not be able to cope with unexplored situations at $\mathcal{N}_m$, which could result in a loss of generality based on the agent's policy.

4.3 Remedy 2: Imputation over Unobserved States and Rewards

To enable agents to learn from experiences on unobserved intersections $\mathcal{N}_m$, it is necessary to impute both state and reward for unobserved intersections $k \in \mathcal{N}_m$. After getting both the imputed state $\hat{s}_t^k$ and inferred reward $\hat{r}_t^k$ at $\mathcal{N}_m$, we can train agents with these imputed experiences $(\hat{s}_t^k, \hat{r}_t^k, a_t^k, \hat{s}_{t+1}^k)$.

Imputation. The process of state imputation is the same as described in Sec. 4.2.

For reward imputation, we use a neural network to infer $\hat{r}_t^k$ for $\mathcal{N}_m$ with state and action as input and pre-train it before RL training starts. In the pre-training phase, we first run with a conventional control approach to collect (s_t^j, r_t^j, a_t^j) as training samples from observed intersections $j \in \mathcal{N}_o$, upon which we train the reward imputation model g_ψ with MSE Loss:

$$\mathcal{L}(\psi) = \frac{1}{n} \sum_n (g_\psi(s^j, a^j) - r^j)^2 \tag{2}$$
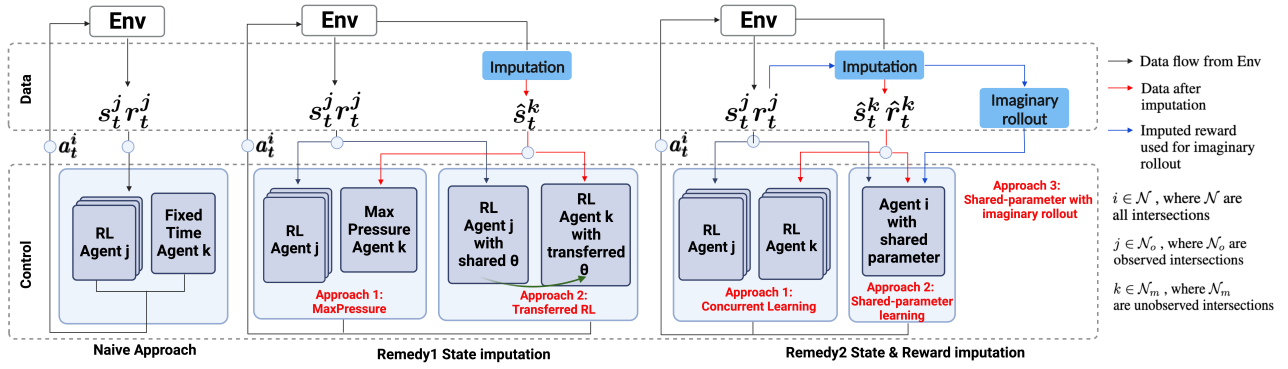
Figure 1: Overall framework of our proposed approaches. Red lines represent how the imputed data flows into unobserved intersections. And blue lines show how reward is imputed and used for imaginary rollout. In Remedy 1, we only impute states at $\mathcal{N}_m$ and enable agents at these intersections to take adaptive actions; In Remedy 2, we impute states and rewards together for training agents. *(a) Remedy 1.1*: RL agents at $\mathcal{N}_o$ take s_t^j, r_t^j for training their policies, and take s_t^j during the execution phase. And MaxPressure agents at $\mathcal{N}_m$ take imputed $\hat{s}_t^k$ together with neighboring intersections' s_t^j during training and execution phases. *(b) Remedy 1.2*: RL agents take s_t^j, r_t^j, and train based on these observable experiences. And at the $\mathcal{N}_m$, agents are transferred from $\mathcal{N}_o$ and take imputed $\hat{s}_t^k$ for taking actions during the execution phase. *(c) Remedy 2.1*: agents at $\mathcal{N}_o$ and $\mathcal{N}_m$ take s_t^j, r_t^j pairs or $\hat{s}_t^k, \hat{r}_t^k$ pairs respectively and optimize their own policies during training. During execution, agents at $\mathcal{N}_o$ take s_t^j from the environment, and agents at $\mathcal{N}_m$ take imputed $\hat{s}_t^k$ for execution. Different from Remedy 2.1, in *(d) Remedy 2.2* and *(e) Remedy 2.3*, all agents share the same policy, while *Remedy 2.3* has an additional imaginary rollout step.

During the RL training phase, the original s_t^j at $\mathcal{N}_o$ returned from the environment will first pass through state imputation model and get the recovered data $\hat{s}_t^k$ at $\mathcal{N}_m$. The imputed $\hat{s}_t^k$ combining a_t^k will be fed into g_ψ which could be described as:

$$\hat{r}_t^k = g_\psi(\hat{s}_t^k, a_t^k) \qquad (3)$$

Combining r_t^j from $j \in \mathcal{N}_o$, experiences at all intersections are now available.

Control. After state and reward imputation, the problem of traffic signal control under the missing data could be transformed into the regular traffic signal control problem. In the following, we investigate three approaches:

• *Approach 1: Concurrent learning.* In concurrent learning, each agent has its own policy and learns from its own experiences. We adopt this imputation over state and reward approach to enable the training of RL method. We use experiences returned from the environment to train RL agents at $\mathcal{N}_o$ and imputed experiences to train agents at $\mathcal{N}_m$. This training approach concurrently trains agents over all intersections and potentially makes each agent achieve its local optimality if the evaluation metric could converge at the training end. The concurrent training process could be problematic when the imputation at missing intersections is inaccurate, and training on such imputed experiences can bring additional uncertainties and make it hard to get stable RL models.

• *Approach 2: Parameter sharing.* To improve the sample efficiency and reduce the instability during training, we investigate the shared-parameter learning approach as Sec. 4.2 did. During training, we collect observed experiences $(s_t^j, r_t^j, a_t^j, s_{t+1}^j)$ from observed intersections $j \in \mathcal{N}_o$ and use the imputation models to impute $(\hat{s}_t^k, \hat{r}_t^k)$ and get the imputed experiences $(\hat{s}_t^k, \hat{r}_t^k, a_t^k, \hat{s}_{t+1}^k)$ for intersections $k \in \mathcal{N}_m$. Then a shared RL policy is trained with both observed and imputed experiences. During execution, the

trained RL policy is shared by all the intersections. This parameter-sharing approach aims to expose the shared agent to the experiences from both $\mathcal{N}_o$ and $\mathcal{N}_m$ and make policy stable and easy to converge.

• *Approach 3: Parameter sharing with the imaginary rollout.* In all imputation approaches, we use a rule-based SFM and pre-trained neural network to impute states or states and rewards. However, the sample distribution shifting caused by different policies could be detrimental to the performance of the pre-trained model [Chen and Jiang, 2019]. Thus we combine the model-based reinforcement learning (MBRL) with the reward model and train a shared policy in the Dyna-Q style framework [Sutton, 1991; Zhao *et al.*, 2020].

In this approach, the shared-parameters agent updates the Q function with both observed experiences $(s_t^j, r_t^j, a_t^j, s_{t+1}^j)$ from observed intersections and imputed experiences from state and reward imputation models. At each simulation step, the reward imputation model g_ψ infers $\hat{r}^j, j \in \mathcal{N}_o$, which will be used in training g_ψ by calculating the loss between $\hat{r}^j$ and r^j returned from the environment with Eq. (3). Each round of imaginary rollout samples a batch of $(s_c^j, a_c^j, s_c'^j)$ and $(\hat{s}_c^k, a_c^k, \hat{s}_c'^k)$, where $k \in \mathcal{N}_m$ and $j \in \mathcal{N}_o$. For $k \in \mathcal{N}_m$, the updated reward imputation model will infer the new $\hat{r}_c^j, \hat{r}_c^k$ to apply additional updates the Q function:

$$Q_\theta(s_c^i, a_c^i) := \hat{r}_c^i + \gamma \max_{a'^i} Q_\theta(s_c'^i, a'^i) \qquad (4)$$

where, $\hat{r}_c^i \in \{\hat{r}_c^j, \hat{r}_c^k\}$. Details are shown in Algorithm 1.

5 Experiments

5.1 Experimental Setup

Datasets. We testify our two approaches on TSC task under missing data on a synthetic dataset and two real-world datasets [1]. $\boldsymbol{D_{SYN}}$ is a synthetic dataset generated by

---

[1]The code and dataset can be found on the authors' website.

Algorithm 1: Algorithm for Remedy 2.3 - *SDQN-SDQN* (model-based) with imaginary rollout

---

Input: Observed intersections $j \in \mathcal{N}_o$, unobserved intersections $k \in \mathcal{N}_m$, pre-trained reward function $g_\psi(s,a)$, initial $Q_\theta(s,a)$, state imputation function $f(\mathcal{V}_{t-1}^k)$

Output: Q_θ, g_ψ

1 **for** *e = 1,2, . . .* **do**
2   Reset simulator environment
3   **for** *t = 1,2 , . . .* **do**
4   Use states from observed intersections to impute $\hat{s}^k$ with Eq. (1)
5   Take actions with $a^j = \arg\max Q_\theta(s^j, a^j)$, and $a^k = \arg\max Q_\theta(\hat{s}^k, a^k)$
6   Infer $\hat{r}_k$ with $g_\psi(\hat{s}^k, a^k)$, and $\hat{r}_j$ with $g_\psi(s^j, a^j)$
7   Record r^j returned from environment
8   Update $Q_\theta(s,a)$ with (s^j, r^j, a^j, s'^j) and $(\hat{s}^k, \hat{r}^k, a^k, \hat{s}'^k)$ using Eq. (4)
9   ***# Reward function update step***
10   Optimizing $g_\psi(s,a)$ with Eq. (2)
11   ***# Imaginary rollout step***
12   **for** *c = 1,2 , . . .* **do**
13   Randomly sample experiences, including imputated state-action pairs $(\hat{s}^k, a^k, \hat{s}'^k)$ from $\mathcal{N}_m$ and observed experiences (s^j, a^j, s'^j)
14   Infer $\hat{r}^j$ with $g_\psi(s^j, a^j)$, $\hat{r}^k$ with $g_\psi(\hat{s}^k, a^k)$
15   Update $Q_\theta(s,a)$ with $(s^j, \hat{r}^j, a^j, s'^j)$ and $(\hat{s}^k, \hat{r}^k, a^k, \hat{s}'^k)$
16   **end**
17   **end**
18 **end**

CityFlow [Zhang *et al.*, 2019], an open-source microscopic traffic simulator. The traffic road network is 4×4 grid structured, and traffic flow is randomly generated following Gaussian distribution. $\mathbf{D_{HZ}}$ is a public traffic dataset that recorded a 4×4 network at Hangzhou in 2016. All the dataset is collected from surveillance cameras nearby. $\mathbf{D_{NY}}$ is a public traffic dataset collected in New York City within 16×3 intersections. Three datasets contain every vehicle's position and speed at each second and the trajectory within the road network.

Implementation. We introduce the details of reinforcement learning , state, and reward imputation models:

• *RL settings.* We follow the past work [Wei *et al.*, 2019b; Wu *et al.*, 2021; Huang *et al.*, 2021] to set up the RL environment, and details on the state, reward, and action definition can be found in Sec. 3. We take exploration rate $\epsilon = 0.1$, discount factor $\gamma = 0.95$, minimum exploration rate $\epsilon_{min} = 0.01$, exploration decay rate $\epsilon_{decay} = 0.995$, and model learning rate $r = 0.0001$.

• *State imputation model.* SFM model is a rule-based method often used in past traffic signal control design avenues. In this work, we model the current state as: $f(\mathcal{V}_{t-1}^k) = \frac{1}{|\mathcal{V}_{t-1}^k|} \Sigma_l s_{t-1}^l$, where $l \in \mathcal{V}_{t-1}^k$ and $|\mathcal{V}_{t-1}^k|$ is the number of k's neighboring intersections.

• *Reward imputation model.* To pre-train the reward imputation model, we use a four-layer feed-forward neural network and simulate 100 epochs to collect the training data with traffic signals controlled by the conventional approach 2 described in Sec. 4.1. The training samples are collected from observed intersections and divided into 80% and 20% for training and testing. In the RL framework, we train agents for 100 epochs and take the average travel time for agents' performance evaluation.

Compared methods To describe different control methods without misunderstanding, we use the kind of agents at observed and unobserved intersections to denote these methods. For example, in *IDQN-Fix*, the first term represents that $\mathcal{N}_o$ uses IDQN [Wei *et al.*, 2018], and the second term represents that $\mathcal{N}_m$ uses fixed timing:

• *Conventional 1: Fix-Fix.* This is a ruled-based method with fixed timings for all phases. We use Webster's method [Koonce and Rodegerdts, 2008] to calculate the fixed timing and fine-tune it with a grid search to ensure the fixed time method had its best results.

• *Conventional 2: IDQN-Fix.* In this method, intersections in $\mathcal{N}_o$ use their own model trained by Deep Q-Learning (DQN) [Wei *et al.*, 2018] and intersections in $\mathcal{N}_m$ use fine-tuned fixed timings.

• *Conventional 3: IDQN-Neighboring.* This is a method where both $\mathcal{N}_m$ and $\mathcal{N}_o$ use IDQN. At $\mathcal{N}_o$, agents take in state and reward from the environment, and at $\mathcal{N}_m$, agents take states and rewards from neighboring intersections. Unobserved neighboring intersections are zero-padded.

• *Remedy 1.1: IDQN-MaxP.* In this method, intersections in $\mathcal{N}_o$ uses the same IDQN agents as *IDQN-Fix*. For intersections in $\mathcal{N}_m$, a ruled-based control approach MaxPressure [Varaiya, 2013] is used after the imputation of $\hat{s}_t^k$. Different from the conventional methods, this method has a predefined SFM model for state imputation, which is shared by all the intersections in $\mathcal{N}_m$.

• *Remedy 1.2: SDQN-SDQN (transferred).* Similar to *IDQN-MaxP*, this method also imputes the states with SFM model. Different from *IDQN-MaxP*, all the agents share one policy which is trained by collecting data from intersections in $\mathcal{N}_o$ and then transferred to intersections in $\mathcal{N}_m$.

• *Remedy 2.1: IDQN-IDQN.* Unlike Remedy 1, in addition to state imputation model, this method has a pretrained reward imputation model shared by the intersections in $\mathcal{N}_m$. Each intersection has its individual RL policy to control the actions trained from the observed data (for intersections in $\mathcal{N}_o$) or the imputed data (for intersections in $\mathcal{N}_m$).

• *Remedy 2.2: SDQN-SDQN (all).* Similar to *IDQN-IDQN*, this method also has a state imputation model and a reward imputation model, while it only trains one shared policy using the observed data from $\mathcal{N}_o$ and imputed data from $\mathcal{N}_m$.

• *Remedy 2.3: SDQN-SDQN (model-based).* This method integrates *SDQN-SDQN* (all) approaches into MBRL frame-

Dataset	Missing rate	Method							
		Fix-Fix	IDQN-Neighboring	IDQN-Fix	IDQN-MaxP	SDQN-SDQN (transferred)	IDQN-IDQN	SDQN-SDQN (all)	SDQN-SDQN (model-based)
D_{HZ}	6.25%	609.13	433.67±26.75	337.07±4.54	334.41±2.42	331.16±2.28	424.81±15.36	330.85±2.61	**330.23**±1.04
	12.5%		-	362.89±6.03	339.71±1.86	330.84±1.85	497.21±59.43	**329.11**±0.30	331.35±1.63
	18.75%		-	370.18±2.58	342.57±1.82	332.20±4.55	537.85±56.67	**330.28**±1.99	358.55±35.78
	25%		-	396.36±1.79	382.93±5.60	331.57±1.81	653.09±71.44	333.87±3.06	**330.51**±1.48
D_{SYN}	6.25%	713.69	767.54±14.63	640.68±27.11	577.92±31.58	350.85±7.51	683.41±95.21	368.76±4.43	**325.64**±13.31
	12.5%		-	600.45±15.53	699.26±54.01	399.03±15.75	727.64±83.78	440.68±43.54	**361.07**±9.64
	18.75%		-	637.50±32.93	673.56±42.76	808.25±52.81	794.89 ± 51.09	584.03±37.95	**568.21**±7.29
	25%		-	574.01±18.42	719.52±28.91	660.59±17.09	877.44±101.36	**538.42**±32.09	540.13±20.17
D_{NY}	6.25%	1099.67	519.95±259.09	286.43±124.59	334.41±2.42	200.72±9.1	279.25±40.34	**191.06**±1.21	192.53±4.26
	12.5%		-	726.68±163.72	502.56±285.56	215.54±3.39	336.9±26.28	513.08±273.82	**210.4**±5.36
	18.75%		-	913.48±31.77	820.91±82.97	240.98±33.12	415.31±83.85	228.39±2.73	**220.49**±1.06
	25%		-	1012.91±44.25	1218.71±32.86	414.79±147.33	1331.60±58.94	507.67±181.38	**316.56**±37.97

Table 1: Overall performance of two imputation approaches and two baseline approaches w.r.t. the average travel time. The lower, the better. The **best** and second best performance are highlighted. '-' means this method does not converge after 100 epochs of training.
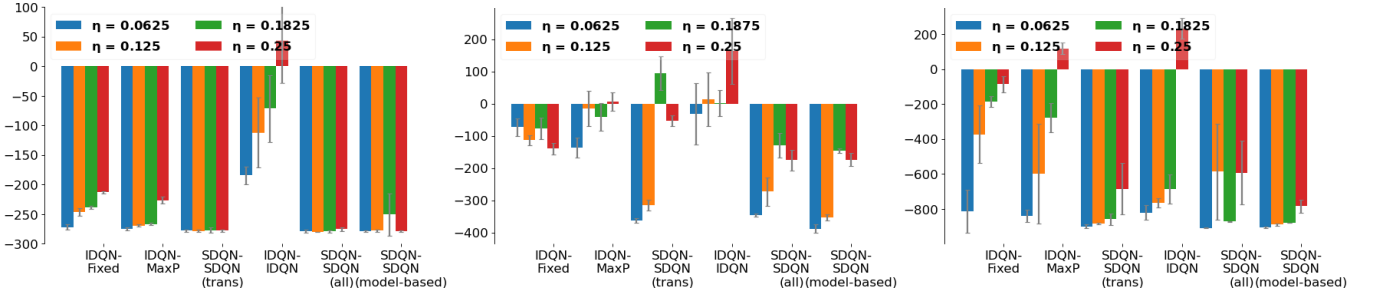

Figure 2: The decrease of average travel time for six control approaches over *Fix-Fix* control method (conventional 1) with missing data at non-neighboring intersections w.r.t. D_{HZ} (left) D_{SYN} (middle) D_{NY} (right). The more negative the value is, the better. Each column in one group uses the same control method. η represents the missing rate of intersections in the road network. All dynamic control methods achieve better performance than *Fix-Fix* control. *IDQN-MaxP*, *SDQN-SDQN* (transferred), *SDQN-SDQN* (all), and *SDQN-SDQN* (model-based) outperform *IDQN-Fix* (conventional 2) method on all three datasets.

work. During training, the SDQN agents learn from experiences at all intersections and, at the same time, the reward inference model predicts reward at $\mathcal{N}_m$. Different from *SDQN-SDQN* (all), the reward model is updated upon new experiments returned from the environment; it also has an imaginary rollout phase, during which SDQN agents learn from both states s_t^j at $\mathcal{N}_o$ and imputed states $\hat{s}_t^k$ at $\mathcal{N}_m$ and inferred rewards $\hat{r}_t^i$ from the updated reward model.

5.2 Overall Performance

We perform experiments to investigate how different approaches perform under different missing rates. The results can be found in Table 1. We have the following observations: (1) Compared with *Fix-Fix*, optimizing IDQN agents at $\mathcal{N}_o$ can significantly reduce the average travel time. This validates the effectiveness of RL agents over pre-timed agents. (2) Compared with *IDQN-Fix*, *IDQN-Neighboring* works worse even at the ideal settings and cannot converge at higher missing rates, which proves optimizing IDQN agents with no imputation cannot solve the missing data problem. (3) *IDQN-IDQN* method in Remedy 2 does not outperform the original naive method since all agents on $\mathcal{N}_m$ are only trained with imputed data, which could bring in large uncertainty. (4) *SDQN-SDQN* (model-based), *SDQN-SDQN* (all), *SDQN-SDQN* (transferred), and *IDQN-MaxP* approaches achieve

better performances than naive *IDQN-Fix* approach under all three datasets. This proves the effectiveness of our two-step imputation and control method. As the missing rate increases, their performance decreases. (5) For shared-parameter methods, when the missing rate is moderate, the overall performance is not greatly affected. This is because the shared agent can learn from the experience in both $\mathcal{N}_o$ and $\mathcal{N}_m$ and make policy stable and easy to converge.

Summary of Different Approaches. We analyze and summarize all approaches in the Table 2, Generally, fixed-time methods (e.g., Fix-Fix, IDQN-Fix) do not rely on observations and cannot adapt well to dynamic traffic; shared RL methods (e.g., SDQN-SDQN) using Centralized Training and Decentralized Execution (CTDE) usually perform better than individual RL (e.g., IDQN), particularly with more agents, as validated by [Chen *et al.*, 2020]. SDQN-SDQN (transferred) in Remedy 1 uses control models only trained on observed intersections, which may underperform when deployed on unobserved intersections. SDQN-SDQN (all) in Remedy 2 pre-trains an additional reward model, enabling the update of the control model for unobserved intersections, thus mitigating performance issues. SDQN-SDQN (model-based) further refines the reward model training, enhancing the control model's performance.
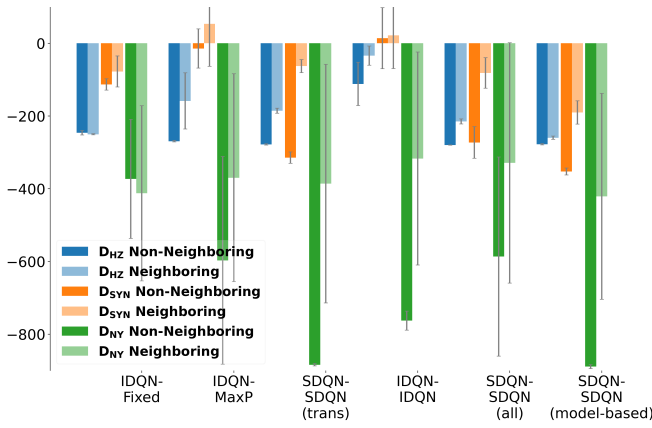
Figure 3: The decrease of average travel time for six control approaches over *Fix-Fix* (conventional 1) with missing data at non-neighboring and neighboring intersections under all three datasets. The more negative the value is, the better. Each group on X-axis represents a method. Within each group, three colors represent three datasets, where light- and dark-colored bars indicate scenarios with and without adjacent unobserved intersections, respectively. Our proposed method outperforms the *Fix-Fix* method in most cases.

5.3 Data Sparsity Analysis

We investigate how the missing rate and unobserved locations influence the performance of the proposed methods.

Influence of missing rates. We randomly sample 1,2,3,4 intersections from 16 intersections in the D_{HZ} and D_{SYN} and 3,6,9,12 intersections from 48 intersections in the D_{NY} as unobserved intersections. The results in Table 1 and Figure 2 show that when there are no neighboring unobserved intersections, *IDQN-MaxP*, *SDQN-SDQN* (transferred), and *SDQN-SDQN* (all), *SDQN-SDQN* (model-based) achieve consistent better performances than *Fix-Fix*. The performances of the four approaches decrease as the number of unobserved intersections increases. Moreover, the three shared-parameter methods are more stable in performance improvement when the missing rates increase.

Influence of unobserved locations. In the previous experiments, the unobserved intersections are not adjacent to each other. Here we investigate how the locations of unobserved intersections influence the performance. We conduct experiments on situations where adjacent intersections are unobserved. We randomly sample missing intersections and make sure the network has two unobserved intersections adjacent. The result is shown in Figure 3. We have the following observations: (1) When there are adjacent unobserved intersections, our proposed method still outperforms the *Fix-Fix* method in most cases. Specifically, *SDQN-SDQN* (transferred), *SDQN-SDQN* (all), and *SDQN-SDQN* (model-based) perform consistently better than other baseline methods. (2) Except for *IDQN-Fix*, the performance of all other methods drops from non-neighboring scenarios to neighboring scenarios. This is likely because the performance of the control method relies on the imputation method, and missing data at neighboring intersections could negatively affect the performance of the imputation.

Remedy	Approach	Advantage
Convention	*Fix-Fix*	The control model does not rely on imputation.
	IDQN-Fix	1. All the control models do not rely on imputation. 2. IDQN is adaptive in observed intersections.
Remedy 1	*IDQN-MaxP*	1. Remedy 1 approaches enable adaptive control in unobserved intersections. 2. Unlike Remedy 2, Remedy 1 approaches do not require a reward imputation model.
	SDQN-SDQN (transferred)	1. Remedy 1 approaches enable adaptive control in unobserved intersections. 2. Unlike Remedy 2, Remedy 1 approaches do not require a reward imputation model. 3. SDQN uses CTDE and converges faster in training the control models.
Remedy 2	*IDQN-IDQN*	Remedy 2 approaches enable the training of the control model on unobserved intersections.
	SDQN-SDQN (all)	1. Remedy 2 approaches enable the training of the control model on unobserved intersections. 2. SDQN uses CTDE and converges faster in training the control models.
	SDQN-SDQN (model-based)	1. Remedy 2 approaches enable the training of the control model on unobserved intersections. 2. SDQN uses CTDE and converges faster in training the control models. 3. This method can alleviate the data-shifting problem by updating the reward model.
		Disadvatage
Convention	*Fix-Fix*	Fixed-time agents are not adaptive to dynamic traffic.
	IDQN-Fix	Fixed-time agents are not adaptive to dynamic traffic.
Remedy 1	*IDQN-MaxP*	1. Remedy 1 approaches require additional training on the state imputation model. 2. MaxPrusure is rule-based and has limited performance under complex traffic. 3. IDQN is hard to converge with multiple agents.
	SDQN-SDQN (transferred)	1. Remedy 1 approaches require additional training on the state imputation model. 2. Control models are not trained on imputed unobserved intersections.
Remedy 2	*IDQN-IDQN*	1. Remedy 2 approaches require additional training on reward imputation models. 2. IDQN is hard to converge with multiple agents. 3. Pretrained reward models might be biased between observed and unobserved intersections.
	SDQN-SDQN (all)	1. Remedy 2 approaches require additional training on reward imputation models. 2. Pretrained reward models might be biased between observed and unobserved intersections.
	SDQN-SDQN (model-based)	Remedy 2 approaches require additional training on reward imputation models.

Table 2: Summary of advantages and disadvantages of approaches

6 Conclusion and Future Work

We investigate the traffic signal control problem in a real-world setting where the traffic condition around certain locations is missing. To tackle the missing data challenge, we propose two solutions: 1) impute the state at missing intersections and directly use them to help agents at missing intersections make decisions; 2) impute both state and reward and use the imputed experiences to train agents. We conduct extensive experiments using synthetic and real-world data and demonstrate the superior performance of our proposed methods over conventional methods. In addition, we show in-depth case studies and observations to understand how missing data influences the final control performance.

We would also like to point out several future works. First, missing data at neighboring intersections brings a considerable challenge to the imputation model. Future work could explore different imputation methods to improve imputation accuracy. Another direction is to combine our imputation model with more external data like speed data to make the imputation more accurate and improve control performance.

Acknowledgments

References

[Aboudolas *et al.*, 2009] K. Aboudolas, M. Papageorgiou, and E. Kosmatopoulos. Store-and-forward based methods for the signal control problem in large-scale congested urban road networks. *Transportation Research Part C: Emerging Technologies*, 17(2):163–174, 2009. Selected papers from the Sixth Triennial Symposium on Transportation Analysis (TRISTAN VI).

[Arel *et al.*, 2010] Itamar Arel, Cong Liu, Tom Urbanik, and Airton G Kohls. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135, 2010.

[Chen and Jiang, 2019] Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pages 1042–1051. PMLR, 2019.

[Chen *et al.*, 2019] Xinyu Chen, Zhaocheng He, and Lijun Sun. A bayesian tensor decomposition approach for spatiotemporal traffic data imputation. *Transportation Research Part C: Emerging Technologies*, 98:73–84, 2019.

[Chen *et al.*, 2020] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3414–3421, 2020.

[Cui *et al.*, 2020] Zhiyong Cui, Ruimin Ke, Ziyuan Pu, and Yinhai Wang. Stacked bidirectional and unidirectional lstm recurrent neural network for forecasting network-wide traffic state with missing values. *Transportation Research Part C: Emerging Technologies*, 118:102674, 2020.

[Duan *et al.*, 2016] Yanjie Duan, Yisheng Lv, Yu-Liang Liu, and Fei-Yue Wang. An efficient realization of deep learning for traffic data imputation. *Transportation Research Part C: Emerging Technologies*, 72:168–181, 2016.

[El-Tantawy *et al.*, 2013] Samah El-Tantawy, Baher Abdulhai, and Hossam 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*, 14(3):1140–1150, 2013.

[Gan *et al.*, 2015] Min Gan, C. L. Philip Chen, Han-Xiong Li, and Long Chen. Gradient radial basis function based varying-coefficient autoregressive model for nonlinear and nonstationary time series. *IEEE Signal Processing Letters*, 22(7):809–812, 2015.

[Huang *et al.*, 2021] Xingshuai Huang, Di Wu, Michael Jenkin, and Benoit Boulet. Modellight: Model-based meta-reinforcement learning for traffic signal control. *arXiv preprint arXiv:2111.08067*, 2021.

[HUNT, 1981] PB HUNT. Scoot-a traffic responsive method of coordinating signals. *TRRL Laboratory Report*, 1014, 1981.

[Koonce and Rodegerdts, 2008] Peter Koonce and Lee Rodegerdts. Traffic signal timing manual. Technical report, United States. Federal Highway Administration, 2008.

[Li *et al.*, 2020] Xiaoshuang Li, Zhongzheng Guo, Xingyuan Dai, Yilun Lin, Junchen Jin, Fenghua Zhu, and Fei-Yue Wang. Deep imitation learning for traffic signal control and operations based on graph convolutional neural networks. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[Luo *et al.*, 2022] Fan-Ming Luo, Tian Xu, Hang Lai, Xiong-Hui Chen, Weinan Zhang, and Yang Yu. A survey on model-based reinforcement learning. *arXiv preprint arXiv:2206.09328*, 2022.

[Lv *et al.*, 2015] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[Mai *et al.*, 2019] Tien Mai, Quoc Phong Nguyen, Kian Hsiang Low, and Patrick Jaillet. Inverse reinforcement learning with missing data. *arXiv preprint arXiv:1911.06930*, 2019.

[Oroojlooy *et al.*, 2020] Afshin Oroojlooy, Mohammadreza Nazari, Davood Hajinezhad, and Jorge Silva. Attendlight: Universal attention-based reinforcement learning model for traffic signal control. *Advances in Neural Information Processing Systems*, 33:4079–4090, 2020.

[Sims and Dobinson, 1980] Arthur G Sims and Kenneth W Dobinson. The sydney coordinated adaptive traffic (scat) system philosophy and benefits. *IEEE Transactions on vehicular technology*, 29(2):130–137, 1980.

[Sutton, 1991] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[Tirone, 2022] Onathan Tirone. Congestion pricing, the route more cities are taking. *Bloomberg*, 2022.

[Varaiya, 2013] Pravin Varaiya. The max-pressure controller for arbitrary networks of signalized intersections. In *Advances in dynamic network modeling in complex transportation systems*, pages 27–66. Springer, 2013.

[Wang *et al.*, 2022] Peixiao Wang, Tong Zhang, Yueming Zheng, and Tao Hu. A multi-view bidirectional spatiotemporal graph network for urban traffic flow imputation. *International Journal of Geographical Information Science*, 36(6):1231–1257, 2022.

[Wei *et al.*, 2018] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui 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, pages 2496–2505, 2018.

[Wei *et al.*, 2019a] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, pages 1290–1298, 2019.

[Wei *et al.*, 2019b] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1913–1922, 2019.

[Wu *et al.*, 2021] Libing Wu, Min Wang, Dan Wu, and Jia Wu. Dynstgat: Dynamic spatial-temporal graph attention network for traffic signal control. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2150–2159, 2021.

[Yao *et al.*, 2018] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. Deep multi-view spatial-temporal network for taxi demand prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[Zhang *et al.*, 2019] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The world wide web conference*, pages 3620–3624, 2019.

[Zhang *et al.*, 2021] Weibin Zhang, Pulin Zhang, Yinghao Yu, Xiying Li, Salvatore Antonio Biancardo, and Junyi Zhang. Missing data repairs for traffic flow with self-attention generative adversarial imputation net. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[Zhao *et al.*, 2020] Yangyang Zhao, Zhenyu Wang, Kai Yin, Rui Zhang, Zhenhua Huang, and Pei Wang. Dynamic reward-based dueling deep dyna-q: Robust policy learning in noisy environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9676–9684, 2020.

[Zhong *et al.*, 2004] Ming Zhong, Pawan Lingras, and Satish Sharma. Estimation of missing traffic counts using factor, genetic, neural, and regression techniques. *Transportation Research Part C: Emerging Technologies*, 12(2):139–166, 2004.