


Article

Model-Based Deep Reinforcement Learning with Traffic Inference for Traffic Signal Control

Hao Wang [†] , Jinan Zhu ^{*,†} and Bao Gu

School of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai 201418, China

* Correspondence: 206141124@mail.sit.edu.cn

† These authors contributed equally to this work.

Abstract: In the modern world, the extremely rapid growth of traffic demand has become a major problem for urban traffic development. Continuous optimization of signal control systems is an important way to relieve traffic pressure in cities. In recent years, with the impressive development of deep reinforcement learning (DRL), some DRL approaches have started to be applied to traffic signal control. Unlike traditional signal control methods, agents trained using DRL approaches continuously receive feedback from the environment to continuously improve the policy. Since current research in the field is more focused on the performance of the agent, data efficiency during training is ignored to some extent. However, in traffic signal control tasks, the cost of trial and error is very expensive. In this paper, we propose a DRL approach based on a traffic inference model. The proposed traffic inference model is based on the future information given based on upstream intersections and data from the environment to continuously learn the changing patterns of the traffic environment in order to make inferences about changes in the traffic environment. In the proposed algorithm, the inference model interacts with the agent instead of the environment. Through comprehensive experiments based on realistic datasets, we demonstrate that our proposed algorithm is superior to other algorithms in terms of its data efficiency and stronger performance.

Keywords: reinforcement learning; deep learning; traffic signal control; traffic inference



Citation: Wang, H.; Zhu, J.; Gu, B. Model-Based Deep Reinforcement Learning with Traffic Inference for Traffic Signal Control. *Appl. Sci.* **2023**, *13*, 4010. <https://doi.org/10.3390/app13064010>

Academic Editor: Vicente Julian Inglada

Received: 11 February 2023

Revised: 16 March 2023

Accepted: 20 March 2023

Published: 21 March 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Since the 1970s, adaptive traffic signal control (ATSC) systems have been used in urban traffic environments in many regions around the world as an effective method of alleviating traffic congestion by controlling traffic signals in response to real-time traffic flow. With adaptive traffic signal control, the aim is to adjust signal timing according to real-time traffic conditions in order to achieve the goal of being closer to the optimal control policy than when using static timing schemes. Early and successful products, such as SCOOT [1] and SCATS [2], treat signal control as an optimization problem in finding effective coordination and control strategies. Various interdisciplinary techniques have been applied to ATSC since the 1990s, such as fuzzy logic [3] and neural networks [4]. The design of ATSC systems is a long-term challenge.

Reinforcement learning is a technique in artificial intelligence based on Markov decision processes (MDPs) with the goal of producing fully autonomous agents [5]. These agents are able to interact with the environment to learn optimal behavioral policies, and through trial and error they can continuously improve, gradually approaching the optimal policy. RL has had some success in the past [6,7], but previous approaches were limited to low-dimensional problems. RL algorithms suffer from the same complexity problems as other algorithms: as the size of the problem increases, the requirement for resources for computation grows exponentially. In recent years, deep learning has gradually emerged [8,9]. It relies on the approximation ability of the neural network to the function to

reduce large-scale sample data, and the generated model can extract the common patterns and features from the samples, so as to achieve the purpose of classification or regression. The generated model has strong generalization, and excellent prediction results can be obtained for homologous distribution data. The rise of deep learning has provided new tools for reinforcement learning.

The combination of deep neural networks and reinforcement learning, or deep reinforcement learning (DRL), has been used in higher dimensional problems, resulting in many exciting achievements in the past few years. Deep Q-learning network (DQN) was the start of a revolution in DRL, learning directly from images how to play a series of Atari 2600 video games, reaching superhuman levels [10]. AlphaGo, based on techniques such as Monte Carlo tree search (MCTS) and DRL, defeated the human world champion in Go [11]. Many new architectures are also emerging, such as rainbow [12], asynchronous methods [13], twin delayed deep deterministic policy gradient [14], and proximal policy optimization [15], among others.

In recent years, modern DRL methods have started to be applied to ATSC. Similar to other DRL domains, traffic signal control tasks are converted into MDP-based forms. The agent controls traffic signals in a training environment by trial and error to collect data. The collected data will be saved into the experience replay [16,17] or used directly [13,15] for policy improvement of the agent. An intuitive experience is that the greater the amount and diversity of data, the more effective it is in policy improvement [17]. Therefore, enhancing the diversity and amount of data obtained by giving agents more opportunities to explore has become a common trend in high-performance DRL algorithms nowadays. However, in traffic signal control tasks, it is very costly for the agent to directly explore the environment. During the exploration process, the agent needs to maintain a trial-and-error process in the environment to obtain reward data to support policy improvement. However, unlike the Atari game, Go, and other tasks where the agent can perform a trial-and-error process at almost no cost, the agent in the traffic signal control task can directly cause traffic congestion during the trial-and-error process, which is very costly. Giving agents more opportunities to directly explore the environment leads to costly enhancements that are not applicable under traffic signal control tasks. Model-based reinforcement learning aims to solve this problem by first learning a model of the environment dynamics and then reducing the interaction between the agent and the environment based on the learned model.

Therefore, inspired by many successful cases of model-based DRL algorithms, we propose a model-based deep reinforcement learning method for the problem of data efficiency of model-free DRL algorithms that are currently widely used in ATSC. Previous model-based work [18] is still far from representing the state of the art under the task of traffic signal control. In contrast, the most successful approaches are based on model-free RL [19–21]; i.e., they estimate the optimal policy and/or state value function directly from the interaction with the environment. In this paper, we present a DRL algorithm based on future traffic flow information originating from upstream intersections. This is a new method of model-based deep reinforcement learning that achieves superior levels under a single-intersection traffic signal control task. Specifically, this approach uses a traffic inference model based on future information to enhance data efficiency. In addition, this deep neural network based traffic inference model is continuously optimized based on the data collected from the environment.

In summary, the main contributions of this paper are as follows:

- We propose a traffic inference model based on future information, which is able to infer the subsequent traffic state based on the current traffic state and the future traffic flow information given by the upstream intersection.
- Based on our proposed traffic inference model, we propose a model-based DRL algorithm. The traffic inference model is embedded into this algorithm to enhance data efficiency and accelerate strategy improvement.

- Through comprehensive experiments based on real-world data, we demonstrate that our proposed algorithm outperforms existing model-free deep reinforcement learning methods, with enhanced data efficiency and improved final optimal performance.

The remainder of this paper is structured as follows. In Section 2, we review previous related work and present the focus of our research. In Section 3, we present the relevant background of the techniques used. In Section 4, we propose the traffic inference model. In Section 5, we propose a model-based reinforcement learning approach based on the traffic inference model. In Section 6, we conduct comprehensive experiments. In Section 7, we discuss the advantages and disadvantages of our proposed approach. Finally, we present our conclusion and the outlook for future work.

2. Related Work

Deep reinforcement learning has emerged as a promising approach for improving the performance of adaptive traffic signal control systems. In recent years, many articles have explored the application of DRL in ATSC [22–25], especially in improving the efficiency and performance of signal control systems. These articles are broadly focused on two research areas.

One area of research in this field focuses on the problem of cooperation between multiple DRL agents operating at different intersections. These studies have sought to optimize the overall policy by improving the coordination and collaboration between the agents [26–31]. The aim of these efforts is to improve the algorithm for communicating information between agents and to improve the overall performance of the policy.

In another area of research, the focus is on the performance of DRL agents under single intersections [19–21,32,33]. The associated studies have sought to improve the optimality of the policy by developing more advanced methods, such as improving deep neural networks or improving the architecture of the algorithms. These techniques can help to improve the decision-making accuracy and performance of traffic signal control agents, resulting in more efficient and safer traffic flow.

In 2016, DQN was applied to traffic signal control [19], demonstrating its feasibility for converting traffic states into an MDP-based form to find the optimal control policy, and also demonstrating once again that the approximation capabilities of deep neural networks result in the substantially improved performance of reinforcement learning algorithms.

In 2018, some improvements of the DQN algorithm (e.g., dueling, doubling) were integrated and applied to traffic signal control, resulting in 3DQN [20]. Dueling is improvement of the model by introducing the concept of advantage and dividing the Q values into value and advantage for approximation [34]. In contrast, double is an improvement of the training method in which the model that performs policy improvement is separated from the model that executes the policy through model backup, and the model that executes the policy is fixed and replaced only after confirming that the model subject to policy improvement has indeed improved optimality, effectively alleviating the problem of RL algorithm instability based on approximation [35]. Both dueling and doubling can effectively improve the stability of the DQN algorithm.

In 2019, a very novel neural network structure was proposed, called FRAP [21]. This structure is built with same-direction-phase lane groups as the basic unit, and the lane groups without conflicts are arranged and combined. Based on such a structure, the traffic states that are rotated, flipped, etc., and arranged and combined in groups of phase lanes in the same direction are exactly the same for the agent. On the flip side, the collected data are expanded in such a way that FRAP possesses a much faster convergence rate than DQN. In the same year, a demonstration-based DRL training method was proposed and applied to traffic signal control [32]. This approach allows the agent to first learn the SOTL [36] decisions by imitation and then train them in the environment and improve the policy.

In 2020, a framework for initializing DRL agent parameters for traffic environments and based on meta-reinforcement learning was proposed, called Metalight [33]. This framework, through an improved meta-reinforcement learning framework, directly gen-

erates agents that have been improved, and since this generation is independent of the environment dynamics, the agents need to continue training in the environment to improve optimality.

In 2022, a method based on the representation of pressure and demand was proposed [37]. By combining the proposed Advanced-MP with a reinforcement learning algorithm, the algorithm performance reaches an advanced level.

These novel approaches involve either improving training methods [19–21], improving models [20,21], or introducing methods from other domains [32,33], such as imitation learning, meta-reinforcement learning, etc., to allow agents to more rapidly converge. At the same time, experience replay methods have been introduced, such as the introduction of prioritized experience replay for 3DQN [20] and Ape-X for FRAP [21]. The experience replay method is a training method general to the DRL algorithm and is a crucial improvement for DQN [12].

However, to the best of our knowledge, the previous studies still have limitations. To improve the efficiency of agents in policy improvement, algorithms such as Ape-X use multi-environment training methods, which significantly increase the frequency of interaction between the agent and the environment, thus increasing the diversity of the acquired data and helping to improve the performance of the agent. However, as mentioned before, the cost of making mistakes in traffic signal control tasks is very expensive. Simply significantly increasing the interaction of agents with the environment can simultaneously increase the number of agent errors and add significant costs. Therefore, for traffic signal control tasks, an important task is to improve the efficiency of data utilization and reduce the interaction between agents and the environment.

In order to improve the data utilization efficiency of the algorithm, we propose a model-based DRL algorithm that embeds a traffic inference model. This model is capable of making inferences about the evolution of the ensuing traffic state based on the current traffic state and information about the future traffic flow provided by the upstream intersections. Using this model, the agent can significantly reduce the number of actual interactions with the environment and increase the efficiency of utilization of the data collected from the environment.

This proposed model is distinct from those that use information such as weather and time of day to predict traffic flow in order to adapt the agent to a state change closer to reality [38] and the traffic inference systems that use correlation analysis of various predicted big data [39]. We are inspired by multi-agent studies on traffic signal control and directly use the traffic flow information given by upstream intersections to infer the next state. Moreover, unlike the multi-agent methods, we do not provide the traffic flow information directly to the agents but use our proposed inference model to infer. The idea of using models to improve the utilization of data is not entirely new. However, the model design used in previous studies was crude, using only simple fully connected neural networks that neither considered the effect of signal changes on traffic state changes nor integrated the model with the DRL algorithm [18]. In addition, our model not only has a novel neural network architecture but also takes into account the impact that changes in traffic flow at upstream intersections can have on changes in traffic state.

3. Background

Reinforcement learning algorithms use trial and error to learn which actions yield the most reward. The agent begins by taking random actions, and as it gathers more information about the environment, it adjusts its actions based on the rewards it receives. This process of learning through interaction with the environment is known as exploration.

As the agent explores the environment, it builds up a model of the relationships between its actions, observations, and rewards. This model, known as the agent's policy, guides the agent's decision-making process as it selects actions to take in different situations. The goal of the agent is to optimize its policy by choosing actions that maximize the expected reward over time.

The goal of reinforcement learning is to develop intelligent agents that can act independently and adapt their behavior to maximize reward in dynamic environments. The agent interacts with the environment to form sequences:

$$H_t = A_1, O_1, R_1, \dots, A_t, O_t, R_t, \quad (1)$$

where A represents the agent's behavior, O represents the agent's observation of the environment, and R represents the reward received by the agent.

One of the key ideas behind reinforcement learning is the concept of the Markov decision process (MDP). An MDP is a mathematical model used to represent a dynamic, uncertain environment in which an agent can interact with and make decisions.

Since history is serial data, it contains too much noisy information to be used directly. In an MDP, the agent's current state, denoted as $S_t = f(H_t)$, represents all of the relevant information the agent has about its environment at a given time. The state includes everything the agent needs to know in order to make an informed decision. Using the state S_t instead of the full history H_t can be beneficial because it reduces the amount of noise and irrelevant information the agent needs to consider when making a decision. By using a function f to extract the relevant information from the history, the agent can focus on the most important factors and ignore the rest. This can help the agent make more efficient and accurate decisions. The agent's decision, or action, at each time step is based on its current state and is denoted as A_t .

One important property of an MDP is the Markov property, which states that the future state of the system is dependent only on the current state and not on the past states. In other words, the state at time $t + 1$ is only influenced by the state at time t and the action taken at time t . This means that given the current state, the agent does not need to know the full history of the system in order to make an informed decision [5], defined as follows:

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]. \quad (2)$$

The Markov decision process based on Markov states can be formulated as $\langle S, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where:

- S is the set of all states
- \mathcal{A} is the set of all actions
- \mathcal{P} is the state transfer matrix, which describes the probability of transitions between states, specifically, $\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$
- \mathcal{R} is the reward function, which describes the expected reward in the state, specifically, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$
- γ is a discount factor used to discount rewards further away from the current time when calculating cumulative rewards, where $\gamma \in [0, 1]$.

Unlike traditional classification or regression tasks, reinforcement learning does not have a supervisor; i.e., there is no sample label, but instead a reward function to evaluate the behavior, which is by nature a trial-and-error process. The agent's behavior affects the environmental changes and the rewards obtained, while the algorithm's evaluation of the behavior is delayed and is gradually updated and changed. The goal of the agent is to maximize the sum of rewards. The agent aims to learn a policy $\pi(A_t = a | S_t = s)$, which determines the best action a to take given state s , so that the following total discounted return is maximized.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \quad (3)$$

The value function is a forecast of future returns under the current policy and is used to evaluate the goodness of a state and thus determine behavior [5]:

$$V_\pi(s) = \mathbb{E}_\pi[R_t + \gamma V_\pi(S_{t+1}) | S_t = s]. \quad (4)$$

Meanwhile, the action value function is similar but is used to evaluate how good the action is in the target state [5]:

$$Q_{\pi}(s, a) = \mathbb{E}_{\pi}[R_t + \gamma Q_{\pi}(S_{t+1}, A_{t+1}) | S_t = s, A_t = a]. \tag{5}$$

The value function and action value function are interconvertible with each other, and the action value function is usually used in practical applications of reinforcement learning algorithms to avoid the need for a state transfer matrix.

A general method for finding the optimal policy in a known environment is policy iteration. Policy iteration can be divided into two steps. The first step is policy evaluation, i.e., finding the value function of the current policy (which may be randomly initialized), and the second step is policy improvement, i.e., greedy improvement of the policy based on the value function [5]:

$$Q_{\pi}(s, \pi'(s)) = \max_{a \in A} Q_{\pi}(s, a) \geq Q_{\pi}(s, \pi(s)) = V_{\pi}(s). \tag{6}$$

A more concise and efficient version of the policy iteration is Q-learning [5]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha(r + \underset{a'}{\operatorname{argmax}} \gamma Q(s', a') - Q(s, a)). \tag{7}$$

4. Traffic Inference Model

4.1. Model Overview

As shown in Figure 1, the task of the traffic inference model is to infer changes in the traffic state based on the current traffic state and other necessary information, aiming to predict and understand the dynamic changes occurring in the traffic situation. Our proposed traffic inference model is based on neural networks, which have powerful learning and approximation capabilities. By harnessing the power of neural networks, the model is able to continuously learn and adapt to new information, enabling it to predict traffic patterns. Through continuous training, the model is able to reason about changes in traffic states that are distributed in the same way as the training data. With a traffic inference model, reinforcement learning algorithms can interact with the model, which in turn significantly reduces the number of actual interactions with the environment and improves the efficiency of data utilization.

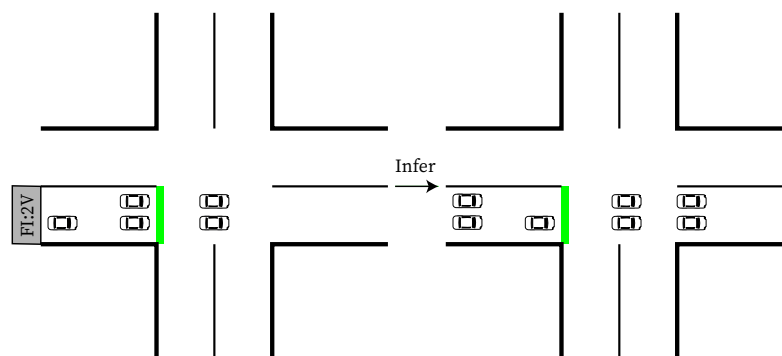


Figure 1. Traffic inference with future information. The green line means the light is green for the moment.

In the Formula (7), the data of the agent’s interaction with the environment is organized in the form $\langle s, a, r, s' \rangle$, where s is the current state, a is the selected action, r is the reward obtained, and s' is the state at the next time step after the action is executed. According

to the Formula (7), the q-table converges to the expectation under a given state transfer distribution after continuous iteration to convergence, r, s' , for a given s, a . Specifically,

$$s' \rightarrow \mathbb{E}[S_{t+1}|S_t = s, A_t = a], \tag{8}$$

$$r \rightarrow \mathbb{E}[R_t|S_t = s, A_t = a], \tag{9}$$

it can be deduced that:

$$Q(s, a) \rightarrow \mathbb{E}[R_t|S_t = s, A_t = a] + \max_{a'} \gamma Q(\mathbb{E}[S_{t+1}|S_t = s, A_t = a], a'). \tag{10}$$

For the traffic signal control environment, the traffic flow is constantly changing and the change of traffic flow also changes the dynamics of the environment. Therefore, the traffic inference model trained based only on s, a is insensitive to changes in the traffic flow and can only correspond to the given traffic flow. In order to make the trained traffic inference model, i.e., the combined model of the state inference model and the reward inference model, able to cope with different traffic flows and complete inference under different traffic flows, we added the future traffic flow information based on upstream intersections into the model. Specifically,

$$s' \rightarrow \mathbb{E}[S_{t+1}|S_t = s, A_t = a, F_t = f], \tag{11}$$

$$r \rightarrow \mathbb{E}[R_t|S_t = s, A_t = a, F_t = f], \tag{12}$$

it can be deduced that:

$$Q(s, a) \rightarrow \mathbb{E}[R_t|S_t = s, A_t = a, F_t = f] + \max_{a'} \gamma Q(\mathbb{E}[S_{t+1}|S_t = s, A_t = a, F_t = f], a'), \tag{13}$$

where F_t represents future information, which can influence the environment dynamics. As shown in Figure 1, the inference of traffic state evolution will become easier when future traffic flow information is provided, which can be obtained from upstream intersections.

In addition, inspired by FRAP, our model will also use inference based on same-direction-phase lane groups to further improve the efficiency of data utilization by making the inference model invariant to data with symmetry. Overall, we design a novel neural network model to reason about the traffic state at the next time step based on the current traffic state, the actions taken, and the future information as well as the rewards. Specifically,

$$s^{(i)}, a^{(i)}, f^{(i)} \rightarrow s'^{(i)}, r^{(i)}, \tag{14}$$

where i is the index of same-direction-phase lane group.

4.2. Model Design for Traffic Inference

Inference of traffic states is very challenging due to the large space of traffic states. For example, for the intersection shown in Figure 2, assuming that the number of lanes is k and the capacity of each lane is n , the number of states of this traffic is $k \times n^k$, and the number of traffic states has already reached 800 million when $k = 8$ and $n = 10$ only.

On this issue, we are inspired by the principle of invariance. Signal control should be invariant to symmetries such as rotation and flip, as proposed by [21]. As shown in Figure 2, let the traffic state be defined as

$$[N_{north}, N_{east}, N_{south}, N_{west}], \tag{15}$$

where N is the number of vehicles in the lane and the subscript is the direction. Then, the state in diagram (a) is $[1, 2, 3, 5]$ and the state in diagram (b) is $[3, 5, 1, 2]$, and both states should be considered as the same for the agent controlling the traffic signal.

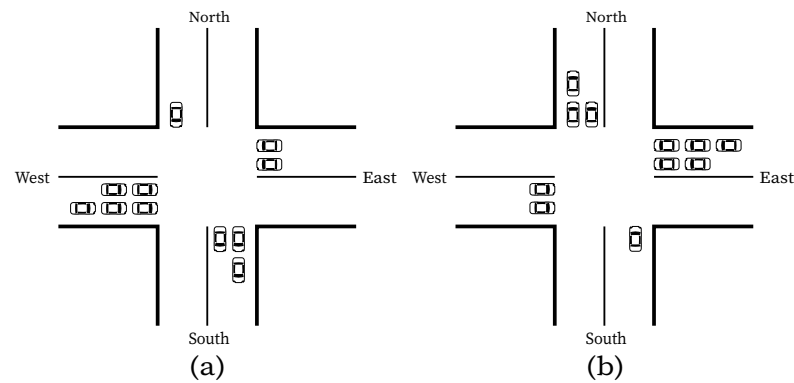


Figure 2. Traffic state with symmetry. (a,b) are similar traffic scenarios.

We introduce the principle that traffic inference should be invariant to symmetries such as rotations and flips so that the model can learn more efficiently from the data and also easily adapt to different intersection structures. We divide traffic inference into four steps: traffic state decomposition, traffic information representation, traffic inference, and traffic state composition. Next, the meaning, role, and specific design of each step are elaborated.

4.2.1. Traffic Information Decomposition

As shown in Figure 3, the meaning of this step is to decompose the collected traffic state, action and future information of the intersection into the state, action, and future information of the lane group. After decomposing the traffic states into lane groups, the principle of invariance is guaranteed. The formula is as follows:

$$(s^{(1)}, s^{(2)}, \dots, s^{(n)}) = Decomp(s), \tag{16}$$

$$(a^{(1)}, a^{(2)}, \dots, a^{(n)}) = Decomp(a), \tag{17}$$

$$(f^{(1)}, f^{(2)}, \dots, f^{(n)}) = Decomp(f), \tag{18}$$

where n is the number of same-direction-phase lane group.

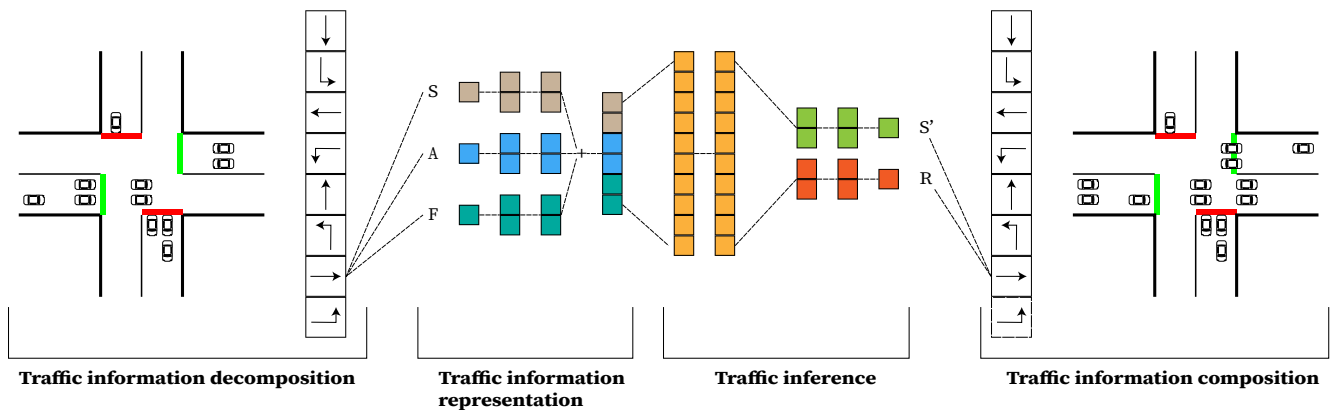


Figure 3. Neural network structure.

4.2.2. Traffic Information Representation

As shown in Figure 3, the implication of this step is that the collected lane group state, action, and future information are encoded separately, similar to word embedding, to obtain their representation vectors. Its role is to convert the raw data into a representation vector that can be recognized by the neural network. Specifically, we denote the collected lane group status, action, and future information as $s^{(i)}, a^{(i)}, f^{(i)}$, respectively. $s^{(i)}, a^{(i)}, f^{(i)}$ are each passed into three independent K_1 hidden layer fully connected neural networks as in-

puts to generate the corresponding representation vectors, denoted as e^s, e^a, e^f , respectively. The formula is as follows:

$$h_1^s = \text{ReLU}(W_1^s e^s + b_1^s), \quad (19)$$

$$h_k^s = \text{ReLU}(W_k^s h_{k-1}^s + b_k^s), k \in [2, K_1], \quad (20)$$

$$e^s = h_{K_1}^s. \quad (21)$$

The calculation of e^a, e^f is the same as for Formulas (19)–(21).

4.2.3. Traffic Inference

As shown in Figure 3, this step converts the extracted feature vector into a reward with the state of the next time step, and traffic inference is completed. Specifically, v is passed into two independent K_2, K_3 layers fully connected neural networks to compute the reward, r , with the state of the next time step, s' . The formula is as follows:

$$h_1^r = \text{ReLU}(W_1^r v + b_1^r), h_1^{s'} = \text{ReLU}(W_1^{s'} v + b_1^{s'}), \quad (22)$$

$$h_k^r = \text{ReLU}(W_k^r h_{k-1}^r + b_k^r), k \in [2, K_2 - 1], \quad (23)$$

$$h_k^{s'} = \text{ReLU}(W_k^{s'} h_{k-1}^{s'} + b_k^{s'}), k \in [2, K_3 - 1], \quad (24)$$

$$r = \text{Sigmoid}(W_{K_2}^r h_{K_2-1}^r + b_{K_2}^r), \quad (25)$$

$$s' = \text{Sigmoid}(W_{K_3}^{s'} h_{K_3-1}^{s'} + b_{K_3}^{s'}). \quad (26)$$

4.2.4. Traffic Information Composition

As shown in Figure 3, this step composes the reward $r^{(i)}$ obtained by inference for each lane group with the state $s'^{(i)}$ of the next time step to obtain the intersection reward and the state of the intersection at the next time step. The formula is as follows:

$$s' = \text{Comp}(s'^{(1)}, s'^{(2)}, \dots, s'^{(n)}), \quad (27)$$

$$r = \text{Comp}(r^{(1)}, r^{(2)}, \dots, r^{(n)}), \quad (28)$$

where n is the number of same-direction-phase lane group.

4.2.5. Summary of Traffic Inference Model

In summary, we have provided a detailed description of the traffic inference model. The network mainly consists of an encoding layer, an inference layer, and a decoding layer. State, action, and future information are first decomposed based on lane groups. The decomposed data are then passed into the encoding layer, where they are encoded and then concatenated into a representation vector. The representation vector is then passed into the inference layer for traffic inference, and the obtained results are passed into two separate decoding layers to be decoded into rewards and states for the next time step. Finally, the states and rewards of all lane groups are combined into the states and rewards of the intersection.

4.3. Discussion

4.3.1. Invariance Principle

The fundamental implication of the invariance principle is that there is a commonality between the state transitions of multiple lane groups at an intersection. Therefore, the task of the inference model can be changed from inference of the traffic state transition of the whole intersection to inference of the traffic state transition of one lane group by decomposing the whole intersection into multiple lane groups. Since the state transition between lane groups has commonality, the inference model can learn generic rules that can cope with the state transition of lane groups. Moreover, in the intersection shown in Figure 2, by decomposing the intersection according to the same direction phase, it can be decomposed into eight groups, and the diversity and quantity of data are enhanced. In

addition, this increment is achieved without degrading the data quality, thus accelerating the convergence of the model while significantly improving the data efficiency.

4.3.2. Future Information

The ability of the traffic inference model to learn the traffic evolution patterns, in addition to coming from the neural network, also stems from the future information provided by the upstream intersections. Without future information, the inference model will no longer learn the pattern of traffic state evolution. Imagine that when traffic states are identical, but traffic volumes are different, the evolution of traffic states is completely different. In the case where no future information is provided, the model will produce the same results. The results are based on past experience and are a prediction of the evolution of the traffic state. With the addition of future information, the model will learn to reason about the next change in traffic state from the current traffic state, the corresponding traffic signal, and the next traffic flow information, and the certainty of this change is substantially improved compared to not using future information.

5. Reinforcement Learning Based on Traffic Inference Model

5.1. Method Overview

As current overall computer performance continues to rise and computing power continues to increase, a broad trend in deep reinforcement learning is that combining more computation with more powerful models and larger datasets produces more impressive results [17]. A growing number of examples prove this opinion: the key factor that allows algorithms such as Gorila [40], A3C [13], GPU A2C [41], distributed PPO [42], Alpha Zero [43], and Ape-X [17] to succeed is the ability to efficiently use more computational resources.

Ape-X is a high-throughput reinforcement learning algorithm that utilizes a heavily parallelized environment. It distributes the generation and selection of experience data and separates it from the training of agents, and it has been demonstrated that this improvement is effective in improving the DRL algorithm. Ape-X divides the standard DRL algorithm into two parts. The first part explores the environment and evaluates the optimality of the current agent, and the collected data are then stored in a shared experience replay, which is called acting. The exploration policies for each environment are individually set, and all data are aggregated into a pool of prioritized experience. The data diversity obtained based on different exploration policies is richer and provides support for improving the performance of the agent. The second part is to continuously sample the batch data from the experience replay in a prioritized manner and continuously update the policy, which is called learning. Moreover, the two parts run asynchronously without using any synchronization strategy, which has the advantage that the two parts, action and learning, will each proceed as fast as possible without having to wait for each other. Inspired by this, our algorithm also adopts an asynchronous approach, including following its diverse exploration strategy.

However, under the traffic signal control task, it is very costly for agents to perform trial and error. The high throughput model-free reinforcement learning algorithm, which is less focused on data efficiency and needs to constantly and frequently interact with the environment to obtain diverse data to improve performance, is less applicable under this task. We therefore propose a reinforcement learning algorithm based on a traffic inference model in which data is used more efficiently, thus improving data efficiency. This stems from the fact that the traffic inference model can continuously infer new data based on existing data instead of the data generated by the large amount of interaction with the environment that is needed to train the agent. Moreover, the traffic inference model continuously learns the patterns of traffic state evolution from the data generated at intersections.

5.2. Method Design

For ease of reading, the functions and formulas provided by the inference model are restated here. The inference model infers the state of the next time step, s' , and the reward r obtained based on the evolution of the state after taking action a under the state s , future information f . The formulas are listed again in the following:

$$s, a, f \rightarrow s', r. \tag{29}$$

The proposed algorithm has a total of five components, which are learner, actor, replay, model, and model replay Figure 4.

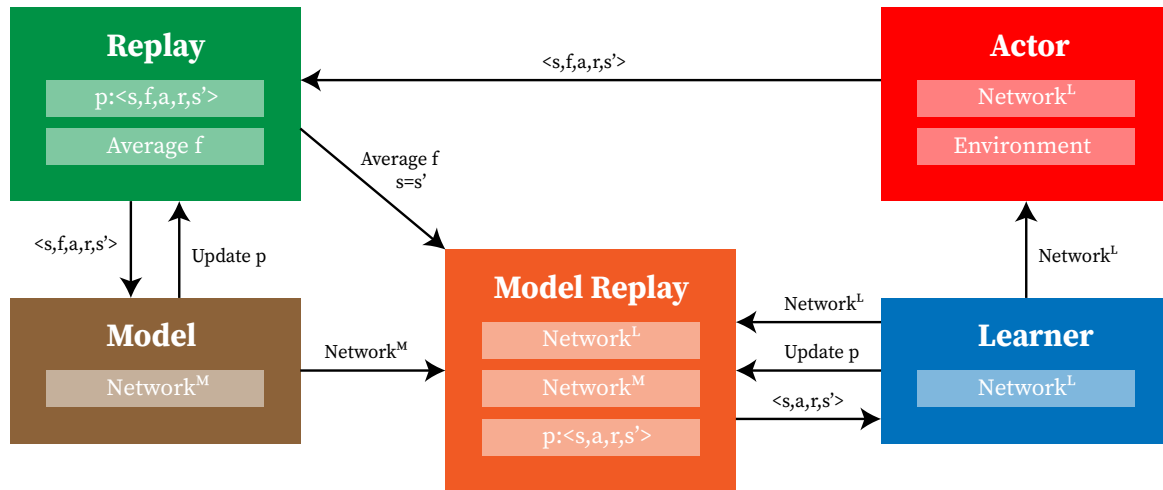


Figure 4. Method architecture.

Specifically, the actor is the part that interacts directly with the environment. Actor periodically obtains the latest network parameters from the learner, selects actions using an ϵ -greedy strategy, and continuously interacts with the environment. The ϵ value is related to the loss of the model, specifically, $\epsilon = c l_t(\theta)$, where c is a constant. In the early stage of training, the actor has a large exploration probability to perform sufficient initial exploration of the environment to prevent overfitting. Subsequently, after the loss of the model is gradually reduced, ϵ will decrease, meaning that the exploration demand of the environment by the training model is no longer huge, and the policy performance of the network parameters obtained from the learner can be fully utilized. The experience obtained from the interaction of the actor with the environment is stored in the replay. Specifically, the experience is defined as $\langle s, a, r, s' \rangle$.

In addition to saving the experience and recording the priority of the experience, replay also counts the mean value of f in the deposited experience. Moreover, as experience is deposited, the value is continuously updated in certain steps, specifically,

$$\bar{f}_k = \bar{f}_{k-1} + \frac{1}{k}(\bar{f}_k - \bar{f}_{k-1}). \tag{30}$$

The model, i.e., the traffic inference model, starts working when a certain number of experiences in the replay is reached. The model first calculates the priority for the experiences in the replay. Specifically, the priority is calculated using the formula

$$p = \frac{1}{2}(r_t - r(S_t, A_t, \theta))^2 + \frac{1}{2}(s_{t+1} - s'(S_t, A_t, \theta))^2. \tag{31}$$

Subsequently, the experience data are randomly selected with the priority as the probability for training and the loss formula

$$l_t(\theta) = \frac{1}{2}(r_t - r(S_t, A_t, \theta))^2 + \frac{1}{2}(s_{t+1} - s'(S_t, A_t, \theta))^2 + c\|\theta\|^2. \quad (32)$$

Random sampling with probability rather than directly with priority prevents overfitting. In addition, the model periodically passes the network parameters to the model replay.

Common to model replay and replay is their same function of storing experience as well as prioritization. The model replay is much richer, and its other major function is to continuously generate new experiences using the network parameters obtained from the model and learner. First, model replay periodically obtains the latest parameters from the model and learner to obtain the most accurate current traffic inference capability and the strongest current signal control policy. Subsequently, in model replay, a certain number of s' are randomly sampled from replay and the policy network parameters obtained from learner used to calculate a with a random ϵ value using an ϵ -greedy strategy. Then, it takes the latest \bar{f} recorded by the replay and randomly generates f for each experience based on a normal distribution. The experiences are then combined into $\langle s, f, a \rangle$ and input to the traffic model to finally get the experience $\langle s, f, a, r, s' \rangle$, which is initialized with the priority using the policy network parameters and deposited into the internal replay. Until the internal experience reaches a certain number, s' continues to be obtained from the internal replay.

The learner is responsible for training the network parameters of the agent. It waits for the experience in the model replay to reach a certain number and then starts training. Similar to the model, the learner first calculates the priority for the experiences in the model replay and then randomly selects the experiences for training using priority as the probability. The formula for calculating the priority and loss is based on the algorithm used. The learner periodically passes the network parameters to the actor and to the model replay.

It is worth mentioning that in the proposed algorithm, the actor, model, model replay, and learner are used in an asynchronous method. The advantage of this is that each part only needs to wait for the initial conditions to be met to start working, and it therefore no longer waits for synchronization of the other parts to work in each round of the loop. Specifically, the algorithm is divided into five parts: learner, actor, replay, model, and model replay. Except for replay, the other four parts run asynchronously, and the flow is shown in the pseudo-code Algorithms 1–4.

Algorithm 1 Actor

```

1: procedure ACTOR( $B, T$ )
2:    $\theta_0 \leftarrow$  LEARNER.PARAMETERS()
3:    $s_0 \leftarrow$  ENVIRONMENT.INITIALIZE()
4:   for  $t = 1$  to  $T$  do
5:      $a_{t-1} \leftarrow$  COMPUTEACTION( $s_{t-1}, \theta_{t-1}$ )
6:      $\langle r_{t-1}, s_t, f_{t-1} \rangle \leftarrow$  ENVIRONMENT.STEP( $a_{t-1}$ )
7:     LOCALBUFFER.ADD( $\langle s_{t-1}, f_{t-1}, a_{t-1}, r_{t-1}, s_t \rangle$ )
8:     if LOCALBUFFER.SIZE()  $\geq B$  then
9:        $\tau \leftarrow$  LOCALBUFFER.GET( $B$ )
10:      REPLAY.ADD( $\tau$ )
11:      REPLAY.UPDATEAVERAGE( $\tau_f$ )
12:     end if
13:     PERIODICALLY( $\theta_t \leftarrow$  LEARNER.PARAMETERS())
14:   end for
15: end procedure

```

Algorithm 2 Model

```

1: procedure MODEL( $T$ )
2:    $\theta_0 \leftarrow$  INITIALIZENETWORK()
3:   for  $t = 1$  to  $T$  do
4:      $\tau \leftarrow$  REPLAY.SAMPLE()
5:      $l_t \leftarrow$  COMPUTELOSS( $\tau$ )
6:      $\theta_{t+1} \leftarrow$  UPDATEPARAMETERS( $l_t, \theta_t$ )
7:      $p \leftarrow$  COMPUTEPRIORITIES( $\tau$ )
8:     REPLAY.SETPRIORITY( $\tau_{id}, p$ )
9:     PERIODICALLY(REPLAY.REMOVEOUTDATED())
10:  end for
11: end procedure

```

Algorithm 3 Model Replay

```

1: procedure MODEL REPLAY( $T$ )
2:    $\theta_0^L \leftarrow$  LEARNER.PARAMETERS()
3:    $\theta_0^M \leftarrow$  MODEL.PARAMETERS()
4:   for  $t = 1$  to  $T$  do
5:      $\tau_{s=s'} \leftarrow$  REPLAY.SAMPLE()
6:      $\bar{f} \leftarrow$  REPLAY.GETAVERAGE()
7:      $\tau_f \leftarrow$  SAMPLE( $\bar{f}$ )
8:      $\tau_a \leftarrow$  COMPUTEACTION( $\tau_s, \theta_{t-1}^L$ )
9:      $\tau_{\langle r, s' \rangle} \leftarrow$  TRAFFICINFERENCE( $\tau_{\langle s, f, a \rangle}, \theta_{t-1}^M$ )
10:     $p \leftarrow$  COMPUTEPRIORITIES( $\tau$ )
11:    LOCALBUFFER.ADD( $\tau_{\langle s, f, a, r, s' \rangle}, p$ )
12:    PERIODICALLY( $\theta_t^L \leftarrow$  LEARNER.PARAMETERS())
13:    PERIODICALLY( $\theta_t^M \leftarrow$  MODEL.PARAMETERS())
14:  end for
15: end procedure

```

Algorithm 4 Learner

```

1: procedure LEARNER( $T$ )
2:    $\theta_0 \leftarrow$  INITIALIZENETWORK()
3:   for  $t = 1$  to  $T$  do
4:      $\tau \leftarrow$  MODELREPLAY.SAMPLE()
5:      $l_t \leftarrow$  COMPUTELOSS( $\tau$ )
6:      $\theta_{t+1} \leftarrow$  UPDATEPARAMETERS( $l_t, \theta_t$ )
7:      $p \leftarrow$  COMPUTEPRIORITIES( $\tau$ )
8:     MODELREPLAY.SETPRIORITY( $\tau_{id}, p$ )
9:     PERIODICALLY(MODELREPLAY.REMOVEOUTDATED())
10:  end for
11: end procedure

```

5.3. DQN with Traffic Inference Model

Reinforcement learning based on traffic inference models needs to be combined with specific agent models, and our choice is to use a variant of DQN with some components of Rainbow [12]. Specifically, we used double [35] and multi-objective bootstrap [5] as training methods. The function approximation is performed using dueling [34] in the model part. The loss used is calculated as $l_t(\theta) = \frac{1}{2}(G_t - Q(S_t, A_t, \theta))^2$. The specific formula for G_t is

$$G_t = R_t + \gamma R_{t+1} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n}, \underset{a}{\operatorname{argmax}} Q(S_{t+n}, a, \theta), \theta^-), \quad (33)$$

where t is the time step index, θ^- is the fixed target network, and θ is the updated network.

6. Experiment

6.1. Environment

6.1.1. RL Environment

In previous work, the definitions of state, action, and reward under the traffic signal control task have varied. Ref. [44] proposed, and demonstrated with reasoning and experiments, that the queue length in the lane is proportional to the delay time with vehicles under the traffic signal control task. Moreover, the number of queued vehicles on the lane, as well as the traffic signal, is sufficient to describe the dynamics of the environment. The experimental results of [21] demonstrate that the performance of the agent under traffic signal control is insensitive to the time interval of signal control. Considering that the longer the time interval, the more drastic the environmental transitions, we chose 10 s in order to reduce the complexity of the traffic inference model to learn the environmental state transitions. Therefore, the reinforcement learning environment we used is defined as follows:

- State: the number of queued vehicles S_v in each lane and the corresponding traffic signal S_i .
- Reward: Negative value of the average number of vehicles in queue on the lane. Specifically,

$$R_t = -\left(\sum_{i=1..N_l} n_i\right) / N_l, \quad (34)$$

where N_l is the number of lanes and n is the number of vehicles in the lane.

- Action: Signal phase is selected every 10 s.

6.1.2. Experiment Settings

Following the previous work, the experiments will be carried out based on simulation. We chose SUMO (Simulation of Urban Mobility), an open source traffic system simulation software that enables microscopic simulation of traffic flow, i.e., specific to the route of each vehicle on the road [45]. The experiments will be conducted based on the collected real traffic flow data. Specifically, each vehicle is defined as $\langle o, d, t \rangle$, where o is the departure point, d is the destination, and t is the departure time.

6.1.3. Datasets

We use a realistic traffic dataset from Hangzhou, China, provided by the collaborator, and a publicly available traffic dataset from Atlanta, USA.

- Suzhou: Our collaborator records live traffic data through the electronic police at intersections in downtown Suzhou. As vehicles pass through the intersection, the vehicle information recorded by all cameras is recorded and aggregated. The processed dataset contains the camera locations as well as desensitized information and time stamps of the vehicles recorded by each camera at the intersection for both approaching and exiting the intersection. We selected the five busiest intersections among all the intersections and intercepted the data from 12:00 p.m. to 2:00 p.m. on a certain day, and each intersection contains about 6000 points of vehicle flow data on average.
- Atlanta: The dataset contains vehicle track data, collected on 8 November 2006, on Peachtree Street in Atlanta, Georgia. It includes data from 12:45 p.m. to 1:00 p.m., and from 4:00 p.m. to 4:15 p.m. The data were collected from eight cameras mounted on a 30-story building located on Peachtree Street in Atlanta, Georgia. The vehicle track data was transcribed from the video data with a high degree of accuracy. In total, there are five intersections covered by the dataset, and each intersection contains about 2000 points of vehicle flow data on average.

6.1.4. Methods for Comparison

In order to evaluate the performance of the proposed algorithms, as a comparison, we have selected several classical signal control algorithms as well as high performance reinforcement learning algorithms. The selected methods are as follows:

- Fixedtime [46]: A fixed signal timing scheme determined in advance.
- SOTL [36]: Self-organizing traffic light control is a classical control method for the task of traffic signal control. The phase is switched when the cumulative summation of conflicting phases breaks a preset value and the number of vehicles in the released phase is not less than the preset value. The purpose is to sense the control signal while preventing the complete traffic flow from being cut off.
- DQN [35]: A deep Q network with extensions such as double, multi-target guidance, and dueling is trained in the environment with control of traffic signals using a synchronized environment according to the previous setting.
- A3C [41]: The classic high-performance algorithm in the field of actor-critic RL. The performance of the actor-critic RL algorithm is significantly improved using asynchronous as well as extensions such as advantage. Train and control traffic signals in the environment according to the previous setting, where multiple asynchronous parallel environments are used.
- Ape-X [17]: An algorithm with superior performance in model-free RL. The performance of model-free RL is improved to a new level based on distributed prioritized empirical replay. Trains and controls traffic signals in an environment according to the previous setting, where multiple asynchronous parallel environments are used.

6.1.5. Evaluation Metrics

We used the most representative metric under the traffic signal control task, average travel time, to evaluate the performance of signal control. Specifically, the average travel time is the average of the time taken by all vehicles from entering the intersection to exiting the intersection.

6.2. Results

6.2.1. Overall Performance

As shown in the Tables 1 and 2, the performance of the proposed method is ahead of the other methods at all junctions. Consistent with the expectation, the reinforcement learning method has stronger performance compared to the traditional methods. Moreover, among the reinforcement learning methods, the proposed approach outperforms Ape-X and A3C in terms of data efficiency and performance. In the proposed method, agents are trained by the traffic inference model to obtain larger and more diverse training data and derive better strategies based on these data.

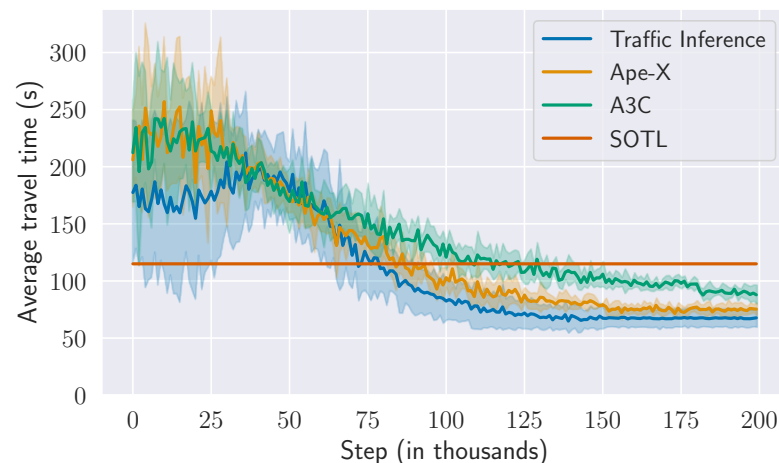
Table 1. Average travel time for the five intersections in Suzhou, in seconds. The bolded result is the optimal result.

Method	1	2	3	4	5
Fixedtime	234.08	274.47	264.68	222.28	249.98
SOTL	112.05	114.94	125.56	115.93	106.39
DQN	93.43	92.12	104.66	107.30	92.62
A3C	92.47	90.60	105.50	102.01	89.38
Ape-X	76.30	78.50	86.87	84.33	81.90
Traffic Inference	71.46	61.57	70.70	67.41	65.36

Table 2. Average travel time for the five intersections in Atlanta, in seconds. The bolded result is the optimal result.

Method	1	2	3	4	5
Fixedtime	335.28	269.15	334.36	142.47	346.51
SOTL	129.27	75.29	137.20	93.05	138.15
DQN	102.06	58.54	98.11	143.98	75.07
A3C	78.79	61.20	78.22	85.45	76.23
Ape-X	67.22	47.24	66.20	74.60	62.14
Traffic Inference	63.68	41.46	62.36	68.24	60.27

Specifically, as shown in Figure 5, in the early training period, the proposed method is less efficient than Ape-X and A3C in terms of policy improvement. This is due to the fact that in the early period, the loss of the traffic inference model is still not converged, and its inference ability is still under training. Moreover, as the number of interactions with the environment increases, the traffic inference model starts to gradually converge. At this point, the efficiency of the proposed method for policy improvement starts to improve and gradually overtakes A3C and Ape-X, finally surpassing all other methods in terms of performance. Comparing the number of interactions between the agent and the environment, it is observed that a large number of interactions is spared using the proposed method compared with Ape-X and A3C while achieving the same performance.

**Figure 5.** Convergence speed of RL methods.

6.2.2. Generalization Capabilities of Traffic Inference Models

One of the major advantages of the traffic inference model is its ability to generalize across different traffic scenarios. Since the model is based on future information, traffic state, and current signals, its inference capability can be generalized to different traffic scenarios. Therefore, the model trained in different traffic scenarios can still be utilized as a pre-trained model to shorten the time for the agent to reach the maximum policy change efficiency.

Experiments were conducted to compare the increase in agent policy improvement efficiency between the two pre-trained models: specifically, the model pre-trained under traffic scenarios in different cities, and the model pre-trained under the same traffic scenarios. As shown in Figure 6, both pre-trained models allow the agent to achieve the highest policy improvement efficiency at the early stage of training, and the increase is approximately the same.



Figure 6. Convergence speed of traffic inference methods.

7. Discussion

Compared with model-free DRL methods, model-based DRL methods tend to take advantage of interacting with the model while reducing the number of interactions with the actual environment. If the advantages offered by models are fully exploited, their performance and convergence speed will exceed that of the more general model-free approaches. It is worth noting that the models that offer advantages are often task-dependent and specifically designed and therefore not usually generalizable across tasks.

Following this concept, we designed a model for traffic signal control tasks, namely a traffic inference model. The model is able to learn the transition patterns of traffic states that are multiplexed in different traffic flow scenarios. To take full advantage of the benefits that the traffic inference model can bring, we propose a DRL approach based on the traffic inference model, in which the traffic inference model is used to reduce the number of interactions between the agent and the environment.

It is worth mentioning that although the generalization of the proposed method is greatly improved compared with the model-free DRL method and can adapt to different traffic flow scenarios, it still has drawbacks. If the state transition pattern in a traffic flow scenario changes drastically, the policy performance of the agent will degrade until the traffic inference model converges again for the new state transition pattern. For example, when a sudden traffic accident occurs, the traffic state transition pattern will change dramatically, and the traffic inference model needs to re-adapt to these changes before these changes will react to the agent's policy, which is lagging compared with model-free DRL.

8. Conclusions

In this paper, we first proposed a traffic inference model for the traffic signal control task and designed the structure of the neural network based on the invariance principle with future information. As discussed in the previous section, the convergence speed of the model is improved based on the invariance principle. Based on future information, the model learns the transition patterns of traffic states that can be reused in different traffic scenarios.

Then, in order to take full advantage of the benefits that the traffic inference model can bring, inspired by Ape-X, we proposed a DRL method based on the traffic inference model. Drawing on the experience of Ape-X, we used an asynchronous architecture to accelerate the convergence speed of the agent and a multi-exploration strategy to enhance the diversity of the obtained data in order to improve the policy performance of the agent. Critically, the number of agent–environment interactions is reduced using the traffic inference model.

We designed and conducted comprehensive experiments based on two datasets collected from reality. The experimental results show that our method is ahead of other methods in data efficiency and outperforms them in terms of performance. We also verified

in our experiments that the inference patterns learned by the traffic inference model can be adapted to different traffic scenarios.

The application of the method to multiple intersections may be an important research direction in future work. The traffic signal control task is more complex under multiple intersections than under single intersections, and the proposed method may still need to be extended to improve the performance. The extension of the traffic inference model is also a major direction. The neural network structure design of the traffic inference model may be improved. In addition, the traffic inference model may benefit from other information besides future information. Finally, our proposed method is inspired by Ape-X, and it may be possible to take fuller advantage of the model using other methods, such as Monte Carlo tree search based on the traffic inference model, and propose new methods.

Author Contributions: Conceptualization, H.W. and J.Z.; methodology, J.Z.; software, B.G.; validation, H.W., J.Z. and B.G.; formal analysis, H.W.; investigation, J.Z.; resources, H.W.; data curation, B.G.; writing—original draft preparation, J.Z.; writing—review and editing, H.W.; visualization, B.G.; supervision, H.W.; project administration, H.W.; funding acquisition, H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Hunt, P.; Robertson, D.; Bretherton, R.; Royle, M.C. The SCOOT on-line traffic signal optimisation technique. *Traffic Eng. Control* **1982**, *23*, 190–192.
- Luk, J. Two traffic-responsive area traffic control methods: SCAT and SCOOT. *Traffic Eng. Control* **1984**, *25*, 14.
- Gokulan, B.P.; Srinivasan, D. Distributed geometric fuzzy multiagent urban traffic signal control. *IEEE Trans. Intell. Transp. Syst.* **2010**, *11*, 714–727. [[CrossRef](#)]
- Srinivasan, D.; Choy, M.C.; Cheu, R.L. Neural networks for real-time traffic signal control. *IEEE Trans. Intell. Transp. Syst.* **2006**, *7*, 261–272. [[CrossRef](#)]
- Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
- Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. *Adv. Neural Inf. Process. Syst.* **1999**, *12*, 1057–1063.
- Tesauro, G. Temporal difference learning and TD-Gammon. *Commun. ACM* **1995**, *38*, 58–68. [[CrossRef](#)]
- Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [[CrossRef](#)]
- LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [[CrossRef](#)]
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [[CrossRef](#)]
- Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [[CrossRef](#)]
- Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the Thirty-second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
- Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 1928–1937.
- Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1587–1596.
- Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
- Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
- Horgan, D.; Quan, J.; Budden, D.; Barth-Maron, G.; Hessel, M.; Van Hasselt, H.; Silver, D. Distributed prioritized experience replay. *arXiv* **2018**, arXiv:1803.00933.

18. Wang, H.; Yuan, Y.; Yang, X.T.; Zhao, T.; Liu, Y. Deep Q learning-based traffic signal control algorithms: Model development and evaluation with field data. *J. Intell. Transp. Syst.* **2021**, 1–21. [[CrossRef](#)]
19. Li, L.; Lv, Y.; Wang, F.Y. Traffic signal timing via deep reinforcement learning. *IEEE/CAA J. Autom. Sin.* **2016**, 3, 247–254.
20. Liang, X.; Du, X.; Wang, G.; Han, Z. A deep q learning network for traffic lights' cycle control in vehicular networks. *IEEE Trans. Veh. Technol.* **2019**, 68, 1243–1253. [[CrossRef](#)]
21. Zheng, G.; Xiong, Y.; Zang, X.; Feng, J.; Wei, H.; Zhang, H.; Li, Y.; Xu, K.; Li, Z. Learning phase competition for traffic signal control. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1963–1972.
22. Gregurić, M.; Vujić, M.; Alexopoulos, C.; Miletić, M. Application of deep reinforcement learning in traffic signal control: An overview and impact of open traffic data. *Appl. Sci.* **2020**, 10, 4011. [[CrossRef](#)]
23. Miletić, M.; Ivanjko, E.; Gregurić, M.; Kušić, K. A review of reinforcement learning applications in adaptive traffic signal control. *IET Intell. Transp. Syst.* **2022**, 16, 1269–1285. [[CrossRef](#)]
24. Noaen, M.; Naik, A.; Goodman, L.; Crebo, J.; Abrar, T.; Abad, Z.S.H.; Bazzan, A.L.; Far, B. Reinforcement learning in urban network traffic signal control: A systematic literature review. *Expert Syst. Appl.* **2022**, 199, 116830. [[CrossRef](#)]
25. Faqir, N.; Loqman, C.; Boumhidi, J. Deep Q-learning Approach based on CNN and XGBoost for Traffic Signal Control. *Int. J. Adv. Comput. Sci. Appl.* **2022**, 13, 9. [[CrossRef](#)]
26. Aslani, M.; Mesgari, M.S.; Wiering, M. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. *Transp. Res. Part C Emerg. Technol.* **2017**, 85, 732–752. [[CrossRef](#)]
27. Chu, T.; Wang, J.; Codecà, L.; Li, Z. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Trans. Intell. Transp. Syst.* **2019**, 21, 1086–1095. [[CrossRef](#)]
28. Wei, H.; Xu, N.; Zhang, H.; Zheng, G.; Zang, X.; Chen, C.; Zhang, W.; Zhu, Y.; Xu, K.; Li, Z. Colight: Learning network-level cooperation for traffic signal control. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1913–1922.
29. Chen, C.; Wei, H.; Xu, N.; Zheng, G.; Yang, M.; Xiong, Y.; Xu, K.; Li, Z. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 3414–3421.
30. Wu, T.; Zhou, P.; Liu, K.; Yuan, Y.; Wang, X.; Huang, H.; Wu, D.O. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. *IEEE Trans. Veh. Technol.* **2020**, 69, 8243–8256. [[CrossRef](#)]
31. Zhang, C.; Jin, S.; Xue, W.; Xie, X.; Chen, S.; Chen, R. Independent reinforcement learning for weakly cooperative multiagent traffic control problem. *IEEE Trans. Veh. Technol.* **2021**, 70, 7426–7436. [[CrossRef](#)]
32. Xiong, Y.; Zheng, G.; Xu, K.; Li, Z. Learning traffic signal control from demonstrations. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 2289–2292.
33. Zang, X.; Yao, H.; Zheng, G.; Xu, N.; Xu, K.; Li, Z. Metalight: Value-based meta-reinforcement learning for traffic signal control. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 1153–1160.
34. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 1995–2003.
35. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
36. Cools, S.B.; Gershenson, C.; D'Hooghe, B. Self-organizing traffic lights: A realistic simulation. In *Advances in Applied Self-Organizing Systems*; Springer: London, UK, 2013; pp. 45–55.
37. Zhang, L.; Wu, Q.; Shen, J.; Lü, L.; Du, B.; Wu, J. Expression might be enough: Representing pressure and demand for reinforcement learning based traffic signal control. In Proceedings of the International Conference on Machine Learning, PMLR, Baltimore, MA, USA, 17–23 July 2022; pp. 26645–26654.
38. Kim, D.; Jeong, O. Cooperative traffic signal control with traffic flow prediction in multi-intersection. *Sensors* **2019**, 20, 137. [[CrossRef](#)]
39. Kim, Y.; Huh, J.H.; Chung, M. Traffic Inference System Using Correlation Analysis with Various Predicted Big Data. *Electronics* **2021**, 10, 354. [[CrossRef](#)]
40. Nair, A.; Srinivasan, P.; Blackwell, S.; Alcicek, C.; Fearon, R.; De Maria, A.; Panneershelvam, V.; Suleyman, M.; Beattie, C.; Petersen, S.; et al. Massively parallel methods for deep reinforcement learning. *arXiv* **2015**, arXiv:1507.04296.
41. Babaeizadeh, M.; Frosio, I.; Tyree, S.; Clemons, J.; Kautz, J. Reinforcement learning through asynchronous advantage actor-critic on a gpu. *arXiv* **2016**, arXiv:1611.06256.
42. Heess, N.; TB, D.; Sriram, S.; Lemmon, J.; Merel, J.; Wayne, G.; Tassa, Y.; Erez, T.; Wang, Z.; Eslami, S.; et al. Emergence of locomotion behaviours in rich environments. *arXiv* **2017**, arXiv:1707.02286.
43. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv* **2017**, arXiv:1712.01815.
44. Zheng, G.; Zang, X.; Xu, N.; Wei, H.; Yu, Z.; Gayah, V.; Xu, K.; Li, Z. Diagnosing reinforcement learning for traffic signal control. *arXiv* **2019**, arXiv:1905.04716.

45. Lopez, P.A.; Behrisch, M.; Bieker-Walz, L.; Erdmann, J.; Flötteröd, Y.P.; Hilbrich, R.; Lücken, L.; Rummel, J.; Wagner, P.; Wießner, E. Microscopic traffic simulation using sumo. In Proceedings of the IEEE 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 2575–2582.
46. Miller, A.J. Settings for fixed-cycle traffic signals. *J. Oper. Res. Soc.* **1963**, *14*, 373–386. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.