Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey

Ammar Haydari, *Student Member, IEEE*, Yasin Yilmaz, *Member, IEEE*

***Abstract*—Latest technological improvements increased the quality of transportation. New data-driven approaches bring out a new research direction for all control-based systems, e.g., in transportation, robotics, IoT and power systems. Combining data-driven applications with transportation systems plays a key role in recent transportation applications. In this paper, the latest deep reinforcement learning (RL) based traffic control applications are surveyed. Specifically, traffic signal control (TSC) applications based on (deep) RL, which have been studied extensively in the literature, are discussed in detail. Different problem formulations, RL parameters, and simulation environments for TSC are discussed comprehensively. In the literature, there are also several autonomous driving applications studied with deep RL models. Our survey extensively summarizes existing works in this field by categorizing them with respect to application types, control models and studied algorithms. In the end, we discuss the challenges and open questions regarding deep RL-based transportation applications.**

***Index Terms*—Deep reinforcement learning, Intelligent transportation systems, Traffic signal control, Autonomous driving, Multi-agent systems.**

I. Introduction

With increasing urbanization and latest advances in autonomous technologies, transportation studies evolved to more intelligent systems, called intelligent transportation systems (ITS). Artificial intelligence (AI) tries to control systems with minimal human intervention. Combination of ITS and AI provides effective solutions for the 21st century transportation studies. The main goal of ITS is providing safe, effective and reliable transportation systems to participants. For this purpose, optimal traffic signal control (TSC), autonomous vehicle control, traffic flow control are some of the key research areas.

The future transportation systems are expected to include full autonomy such as autonomous traffic management and autonomous driving. Even now, semi-autonomous vehicles occupy the roads and the level of autonomy is likely to increase in near future. There are several reasons why authorities want autonomy in ITS such as time saving for drivers, energy saving for environment, and safety for all participants. Travel time savings can be provided by coordinated and connected traffic systems that can be controlled more efficiently using self-autonomous systems. When vehicles spend more times on traffic, fuel consumption increases, which has environmental and economic impacts. Another reason why human intervention is tried to be minimized is the unpredictable nature of human behavior. It is expected that autonomous driving will decrease traffic accidents and increase the quality of transportation. For all the reasons stated above, there is a high demand on various aspects of autonomous controls in ITS. One popular approach is to use experience-based learning models, similar to human learning.

Growing population in urban areas causes a high volume of traffic, supported by the fact that the annual congestion cost for a driver in the US was 97 hours and $1,348 in 2018 [1]. Hence, controlling traffic lights with adaptive modules is a recent research focus in ITS. Designing an adaptive traffic management system through traffic signals is an effective solution for reducing the traffic congestion. The best approach for optimizing traffic lights is still an open question for researchers, but one promising approach for optimum TSC is to use learning-based AI techniques.

There are three main machine learning paradigms. Supervised learning makes decision based on the output labels provided in training. Unsupervised learning works based on pattern discovery without having the pre-knowledge of output labels. The third machine learning paradigm is reinforcement learning (RL), which takes sequential actions rooted in Markov Decision Process (MDP) with a rewarding or penalizing criterion. RL combined with deep learning, named deep RL, is currently accepted as the state-of-the art learning framework in control systems. While RL can solve complex control problems, deep learning helps to approximate highly nonlinear functions from complex dataset.

Recently, many deep RL based solution methods are presented for different ITS applications. There is an increasing interest on RL based control mechanisms in ITS applications such as traffic management systems and autonomous driving applications. Gathering all the data-driven ITS studies related to deep RL and discussing such applications together in a paper is needed for informing ITS researchers on deep RL, as well as deep RL researchers on ITS.

In this paper, we review the deep RL applications proposed for ITS problems, predominantly for TSC. Different RL approaches from the literature are discussed. TSC solutions based on standard RL techniques have already been studied before the invention of deep RL. Hence, we believe standard RL techniques also have high importance in reviewing the deep RL solutions for ITS, in particular TSC. Since traffic intersection models are mainly connected and distributed, multi-agent dynamic control techniques, which are also extensively covered in this survey, play a key role in RL-based ITS applications.

A. Contributions

This paper presents a comprehensive survey on deep RL applications for ITS by discussing a theoretical overview of deep RL, different problem formulations for TSC, various deep RL applications for TSC and other ITS topics, and finally challenges with future research directions. The targeted audience are the ITS researchers who want to have a jump start in learning deep RL techniques, and also deep RL researchers who are interested in ITS applications. We also believe that this survey will serve as "a compact handbook of deep RL in ITS" for more experienced researchers to review the existing methods and open challenges. Our contributions can be summarized as follows.

- The first comprehensive survey of RL and deep RL based applications in ITS is presented.
- From a broad concept, theoretical background of RL and deep RL models, especially those which are used in the ITS literature, are explained.
- Existing works in TSC that use RL and deep RL are discussed and clearly summarized in tables for appropriate comparisons.
- Similarly, different deep RL applications in other ITS areas, such as autonomous driving, are presented and summarized in a table for comparison.

B. Organization

The paper organized as follows.

- Section II: Related Work
- Section III: Deep RL: An Overview
  - Section III-A: Reinforcement Learning
  - Section III-B: Deep Reinforcement Learning
  - Section III-C: Summary of Deep RL
- Section IV: Deep RL Settings for TSC
  - Section IV-A: State
  - Section IV-B: Action
  - Section IV-C: Reward
  - Section IV-D: Neural Network Structure
  - Section IV-E: Simulation Environments
- Section V: Deep RL Applications for TSC
  - Section V-A: Standard RL Applications
  - Section V-B: Deep RL Applications
- Section VI: Deep RL for Other ITS Applications
  - Section VI-A: Autonomous Driving
  - Section VI-B: Energy Management
  - Section VI-C: Road Control
  - Section VI-D: Various ITS Applications
- Section VII: Challenges and Open Research Questions

II. Related Work

The earliest work summarizing AI models for TSC including RL and other approaches dates back to 2007 [2]. At that time, fuzzy logic, artificial neural networks and RL was three main popular AI methods researchers applied on TSC. Due to the connectedness of ITS components, such as intersections, multi-agent models provide a more complete and realistic solution than single-agent models. Hence, formulating the TSC problem as a multi-agent system has a high research potential. The opportunities and research directions of multi-agent RL for TSC is studied in [3]. [4] discusses the popular RL methods in the literature from an experimental perspective. Another comprehensive TSC survey for RL methods is presented in [5]. A recent survey presented in [6] studies AI methods in ITS from a broad perspective. It considers applications of supervised learning, unsupervised learning and RL for vehicle to everything communications.

Abduljabbar et al. summarizes the literature of AI based transportation applications in [7] with three main topics: transportation management applications, public transportation, and autonomous vehicles. In [8], authors discuss the TSC methods in general, including classical control methods, actuated control, green-wave, max-band systems, and RL based control methods. *Veres et al.* highlights the trends and challenges of deep learning applications in ITS [9]. Deep learning models play a significant role in deep RL. Nonlinear neural networks overcome traditional challenges such as scalability in the data-driven ITS applications. Lately, a survey of deep RL applications for autonomous vehicles is presented in [10], where authors discuss recent works with the challenges of real-world deployment of such RL-based autonomous driving methods. In addition to autonomous driving, in this survey we discuss a broad class of ITS applications where deep RL is gaining popularity, together with a comprehensive overview of the deep RL concept.

There is no survey in the literature dedicated to the deep RL applications for ITS, which we believe is a very timely topic in the ITS research. Thus, this paper will fill an important gap for ITS researchers and deep RL researchers interested in ITS.

III. Deep RL: An Overview

Deep RL is one of the most successful AI models and the closest machine learning paradigm to human learning. It combines deep neural networks and RL for more efficient and stabilized function approximations especially for high-dimensional and infinite-state problems. This section describes the theoretical background of traditional RL and major deep RL algorithms implemented in ITS applications.

A. Reinforcement Learning

RL is a general learning tool where an agent interacts with the environment to learn how to behave in an environment without having any prior knowledge by learning to maximize a numerically defined reward (or to minimize a penalty). After taking an action, RL agent receives a feedback from the environment at each time step t about the performance of its action. Using this feedback (reward or penalty) it iteratively updates its action policy to reach to an optimum control policy. RL learns from experiences with the environment, exhibiting a trial-and-error kind of learning, similar to human learning [11]. The fundamental trade-off between exploration and exploitation in RL strikes a balance between new actions and learned actions. From a computational perspective, RL is a data-driven approach which iteratively computes an approximate solution to the optimum control policy. Hence, it is also known as approximate dynamic programming [11] which is one type of sequential optimization problem for dynamic programming (DP).

In a general RL model, an agent controlled with an algorithm, observes the system state s_t at each time step t and receives a reward r_t from its environment/system after taking the action a_t. After taking an action based on the current policy π, the system transitions to the next state s_{t+1}. After every interaction, RL agent updates its knowledge about the environment. Fig 1 depicts the schematic of the RL process.

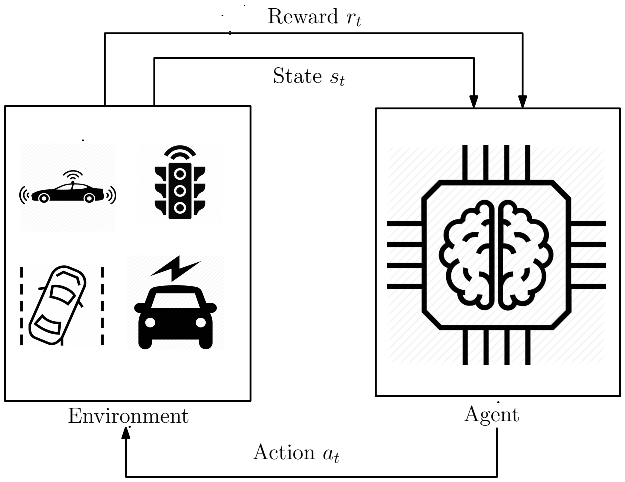


Fig. 1: Reinforcement learning control loop.

1) **Markov Decision Process:** RL methodology formally comes from a Markov Decision Process (MDP), which is a general mathematical framework sequential decision making algorithms. MDP consist of 5 elements in a tuple:

- A set of states $\mathcal{S}$,
- A set of actions $\mathcal{A}$,
- Transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$ which maps a state-action pair for each time t to a distribution of next state s_{t+1},
- Reward function $\mathcal{R}(s_t, a_t, s_{t+1})$ which gives the reward for taking action a_t from state s_t when transitioning to the next state s_{t+1},
- Discount factor γ between 0 and 1 for future rewards.

The essential Markov property is that given the current state s_t, the next state s_{t+1} of system is independent from the previous states $(s_0, s_1, ..., s_{t-1})$. In control systems including transportation systems, MDP models are mostly *episodic* in which the system has a terminal point for each episode based on the end time T or the end state s_T. The goal of an MDP agent is to find the best policy π^* that maximizes the expected cumulative reward $\mathbb{E}[R_t|s, \pi]$ for each state s and cumulative discounted reward (i.e., return)

$$R_t = \sum_{i=0}^{T-1} \gamma^i r_{t+i}, \tag{1}$$

with the discount parameter γ which reflects the importance of future rewards. Choosing a larger γ value between 0 and 1 means that agent's actions have higher dependency on future reward. Whereas, a smaller γ value results in actions that mostly care about the instantaneous reward r_t.

In general, RL agent can act in two ways: (i) by knowing/learning the transition probability $\mathcal{T}$ from state s_t to s_{t+1}, which is called model-based RL, (ii) and by exploring the environment without learning a transition model, which is called model-free RL. Model-free RL algorithms are also divided into two main groups as value-based and policy-based methods. While in value-based RL, the agent at each iteration updates a value function that maps each state-action pair to a value, in policy-based methods, policy is updated at each iteration using policy gradient [11]. We next explain the value-based and policy-based RL methods in detail.

2) **Value-based RL:** Value function determines how good a state is for the agent by estimating the value (i.e., expected return) of being in a given state s under a policy π as

$$V^\pi(s) = \mathbb{E}[R_t|s, \pi]. \tag{2}$$

The optimum value function $V^*(s)$ describes the maximized state value function over the policy for all states:

$$V^*(s) = \max_\pi V^\pi(s), \forall s \in \mathcal{S}. \tag{3}$$

Adding the effect of action, state-action value function named as quality function (Q-function) is commonly used to reflect the expected return in a state-action pair:

$$Q^\pi(s, a) = \mathbb{E}[R_t|s, a, \pi]. \tag{4}$$

Optimum action value function (Q-function) is calculated similarly to the optimum state value function by maximizing its expected return over all states. Relation between the optimum state and action value functions is given by

$$V^*(s) = \max_a Q^*(s, a), \forall s \in \mathcal{S}. \tag{5}$$

Q-function $Q^*(s, a)$ provides the optimum policy π^* by selecting the action a that maximizes the Q-value for the state s:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q^*(s, a), \forall s \in \mathcal{S}. \tag{6}$$

Based on the definitions above, there are two main value-based RL algorithms: Q-learning [12] and SARSA [13], which are classified as off-policy RL algorithm, and on-policy RL algorithm, respectively. In both algorithms, the values of state-action pairs (Q-value) are stored in a Q-table, and are learned via the recursive nature of Bellman equations utilizing the Markov property:

$$Q^\pi(s_t, a_t) \ = \ \mathbb{E}_\pi[r_t + \gamma Q^\pi(s_{t+1}, \pi(s_{t+1}))]. \tag{7}$$

In practice, Q^π estimates are updated with a learning rate α to improve the estimation as follows

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha(y_t - Q^\pi(s_t, a_t)) \tag{8}$$

where y_t is the temporal difference (TD) target for $Q^\pi(s_t, a_t)$. The TD step size is a user-defined parameter and determines how many experience steps (i.e., actions) to consider in computing y_t, the new instantaneous estimate for $Q^\pi(s_t, a_t)$. The rewards $R_t^{(n)} = \sum_{i=0}^{n-1} \gamma^i r_{t+i}$ in the predefined number of

n TD steps, together with the Q-value $Q^{\pi}(s_{t+n}, a_{t+n})$ after n steps give y_t. The difference between Q-learning and SARSA becomes clear in this stage. Q-learning is an off-policy model, in which actions of the agent are updated by maximizing Q-values over the action, whereas SARSA is an on-policy model, in which actions of the agent are updated according to the policy π derived from the Q-function:

$$y_t^{Q-learning} = R_t^{(n)} + \gamma^n \max_{a_{t+n}} Q^{\pi}(s_{t+n}, a_{t+n}), \tag{9}$$

$$y_t^{SARSA} = R_t^{(n)} + \gamma^n Q^{\pi}(s_{t+n}, a_{t+n}). \tag{10}$$

While Q-learning follows a greedy approach to update its Q-value estimates, SARSA follows the same policy for both updating Q-values and taking actions. To encourage exploring new states usually an ϵ-greedy policy is used for taking actions in both Q-learning and SARSA. In the ϵ-greedy policy, a random action is taken with probability ϵ, and the best action with respect to the current policy defined by $Q(s, a)$ is taken with probability $1 - \epsilon$.

In both Q-learning and SARSA, the case with maximum TD steps, typically denoted with $n = \infty$ to express the end of episode, corresponds to a fully experience-based technique called Monte-Carlo RL, in which the Q-values are updated only once at the end of each episode. This means the same policy is used without any updates to take actions throughout an episode. The TD(λ) technique generalizes TD learning by averaging all TD targets with steps from 1 to ∞ with exponentially decaying weights, where λ is the decay rate [11].

3) **Policy-based RL:** Policy-based RL algorithms treat the policy π_θ as a probability distribution over state-action pairs parameterized by θ. Policy parameters θ are updated in order to maximize an objective function $J(\theta)$, such as the expected return $\mathbb{E}_{\pi_\theta}[R_t|\theta] = \mathbb{E}_{\pi_\theta}[Q^{\pi_\theta}(s_t, a_t)|\theta]$. The performance of policy-based methods are typically better than that of value-based methods on continuous control problems with infinite-dimensional action space or high-dimensional problems since policy does not require to explore all the states in a large and continuous space and store them in a table. Although there are some effective gradient-free approaches in the literature for optimizing policies in non-RL methods [14], gradient-based methods are known to be more useful for policy optimization in all types of RL algorithms.

Here, we briefly discuss the policy gradient-based RL algorithms, which select actions using the gradient of objective function $J(\theta)$ with respect to θ, called the policy gradient. In the well-known policy gradient algorithm REINFORCE [15], the objective function is the expected return, and using the log-derivative trick $\nabla \log \pi_\theta = \frac{\nabla \pi_\theta}{\pi_\theta}$ the policy gradient is written as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[Q^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta]. \tag{11}$$

Since computing the entire gradient is not efficient, REINFORCE uses the popular stochastic gradient descent technique to approximate the gradient in updating the parameters θ. Using the return R_t at time t as an estimator of $Q^{\pi_\theta}(s_t, a_t)$ in each Monte-Carlo iteration it performs the update

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta R_t, \tag{12}$$

where α is the learning rate. Specifically, θ is updated in the $\nabla_\theta \log \pi_\theta$ direction with weight R_t. That is, if the approximate policy gradient corresponds to a high reward R_t, this gradient direction is *reinforced* by the algorithm while updating the parameters.

One problem with the Monte-Carlo policy gradient is its high variance. To reduce the variance in policy gradient estimates Actor-Critic algorithms use the state value function $V^{\pi_\theta(s)}$ as a baseline. Instead of $Q^{\pi_\theta}(s, a)$, the advantage function [16] $A^{\pi_\theta}(s, a) = Q^{\pi_\theta}(s, a) - V^{\pi_\theta}(s)$ is used in the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[A^{\pi_\theta}(s, a) \nabla_\theta \log \pi_\theta]. \tag{13}$$

The advantage function, being positive or negative, determines the update direction: go in the same/opposite direction of actions yielding higher/lower reward than average. Actor-Critic method is further discussed in Section III-B3 within the Deep RL discussion.

4) **Multi-Agent RL:** Many real world problems require interacting multiple agents to maximize the learning performance. Learning with multiple agents is a challenging task since each agent should consider other agents' actions to reach a globally optimum solution. Increasing the number of agents also increases the state-action dimensions, thus decomposing the tasks between agents is a scalable approach for large control systems. There are two main issues with high-dimensional systems in multi-agent RL in terms of state and actions: stability and adaptation of agents to the environment [17]. When each agent optimizes its action without considering close agents, the optimal learning for overall system would become non-stationary. There are several approaches to address this problem in multi-agent RL systems such as distributed learning, cooperative learning and competitive learning [17].

B. Deep Reinforcement Learning

In high-dimensional state spaces, standard RL algorithms cannot efficiently compute the value functions or policy functions for all states. Although some linear function approximation methods are proposed for solving the large state space problem in RL, their capabilities are still up to a certain point. In high-dimensional and complex systems, standard RL approaches cannot learn informative features of the environment for effective function approximation. However, this problem can be easily handled by deep learning based function approximators, in which deep neural networks are trained to learn the optimal policy or value functions. Different neural network structures such as convolutional neural network (CNN) and recurrent neural network (RNN) are used for training RL algorithms in large state spaces [18].

The main concept of deep learning is to extract useful patterns from data. Deep learning models are roughly inspired by the multi-layered structure of human neural system. Today, deep learning has applications in a wide spectrum of areas, including computer vision, speech recognition, natural language processing, and the deep RL applications.

*1) **Deep Q-Network:*** Since value-based RL algorithms learn the Q-function by populating a Q-table, it is not feasible to visit all the states and actions in a large state space and continuous action problems. The leading approach to this problem, called Deep Q-Network (DQN) [19], is to approximate the Q-function with deep neural networks. Original DQN receives raw input image as state, and estimates Q-values from them using CNNs. Denoting the neural network parameters with θ the Q-function approximation is written as $Q(s, a; \theta)$. The output of neural network is the best action selected according to (6) using a discrete set of approximate action values.

The major contribution of *Mnih et al.* [19] was two novel techniques to stabilize learning with deep neural networks: target network and experience replay. The original DQN algorithm is shown to significantly outperform the expert human performance on several classic Atari video games. The complete DQN algorithm with experience replay and target network is given by Algorithm 1.

Target Network: One of the main parts of DQN that stabilize learning is the target network. DQN has two separate networks denoted as the main network that approximates the Q-function, and the target network that gives the TD target for updating the main network. In the training phase, while the main network parameters θ are updated after every action, target network parameters θ^{-} are updated after a certain period of time. The reason why target network is not updated after every iteration is that it adjusts the main network updates to keep the value estimations in control. If both networks were updated at the same time, the change in the main network would be exaggerated due to the feedback loop by the target network, resulting in an unstable network. Similar to (9), 1-step TD target y_t is written as

$$y_t^{DQN} = r_t + \gamma \max_{a_{t+1}} Q^{\pi}(s_{t+1}, a_{t+1}; \theta_t^{-}), \qquad (14)$$

where $Q^{\pi}(s_{t+1}, a_{t+1}; \theta_t^{-})$ denotes the target network.

Experience Replay: DQN introduces another distinct feature called experience replay which stores recent experiences (s_t, a_t, r_t, s_{t+1}) in replay memory, and samples batches uniformly from the replay memory for training neural network. There are two main reasons why experience replay is used in DQN. Firstly, it prevents the agent from getting stuck into the recent trajectories by doing random sampling since RL agents are prone to temporal correlations in the consecutive samples. Furthermore, instead of learning over full observations, DQN agent learns over mini-batches that increases the efficiency of the training. In a fixed-size memory defined for experience replay, the memory stores only recent M samples by removing the oldest experience for allocating a space to the latest sample. The same technique is applied in other deep RL algorithms [20], [21].

Prioritized Experience Replay: Experience replay technique samples experiences uniformly from the memory, however, some experiences has more impact on learning than the others. A new approach prioritizing significant actions over other actions is proposed in [22] by changing the sampling distribution of DQN algorithm. The overall idea for prioritized experience replay is that the samples with higher TD error, $y_t^{DQN} - Q^{\pi}(s_t, a_t; \theta_t^{-})$, receives higher ranking in terms of probability than the other samples by applying a stochastic sampling with proportional prioritization or rank-based prioritization. The experiences are sampled based on the assigned probabilities.

Algorithm 1 DQN algorithm

```
Input Replay memory size $M$, batch size $d$, number of episodes $E$, and number of time steps $T$
Inititalize Main network weights $\theta$
Inititalize Target network weights $\theta^{-}$
Inititalize Replay memory
for $e = 1, \dots, E$ do
    Inititalize state $s_1$, and action $a_1$
    for $t = 1, \dots, T$ do
        Take action $a_t = \text{argmax}_a Q^{\pi}(s_t, a; \theta)$ with probability $1 - \epsilon$ or a random action with probability $\epsilon$

        Get reward $r_t$ and observe next state $s_{t+1}$
        if Replay capacity $M$ is full then
            Delete the oldest tuple in memory
        end if
        Store the tuple $(s_t, a_t, r_t, s_{t+1})$ to replay memory
        Sample random $d$ tuples from replay memory
        $y_t = \begin{cases} r_t, & \text{if } t = T. \\ r_t + \gamma \max_a Q^{\pi}(s_{t+1}, a_{t+1}; \theta_t^{-}), & \text{otherwise.} \end{cases}$
        Perform policy gradient using $y_t$ for updating $\theta$
        Update target network every $N$ step, $\theta^{-} = \theta$
    end for
end for
```

*2) **Double Dueling DQN:*** DQN is the improved version of the standard Q-Learning algorithm with a single estimator. Both DQN and Q-Learning overestimates some actions due to having single Q function estimations. Authors in [23] proposes doubling the estimators for action selection with main network and action evaluation with target network separately in loss minimization similar to the tabular double Q-learning technique [24]. Instead of selecting the Q value that maximizes future reward using the target network (see Eq. (14)), double DQN network selects the action using the main network and evaluates it using the target network. Action selection is decoupled with target network for better Q-value estimation:

$$y_t^{DDQN} = r_t + \gamma Q^{\pi}(s_{t+1}, \underset{a_{t+1}}{\text{argmax}}\, Q^{\pi}(s_{t+1}, a_{t+1}; \theta); \theta_t^{-}). \qquad (15)$$

Another improved version of DQN is a dueling network architecture which estimates state value function $V^{\pi}(s)$ and advantage function $A^{\pi}(s, a)$ separately for each action [25]. Output of the combination of these two networks is a Q-value for a discrete set of actions through an aggregation layer. This way dueling DQN learns the important state values without their corresponding effects on the actions since state value function $V^{\pi}(s)$ is an action-free estimation.

These two doubling and dueling models on DQN algorithm with prioritized experience replay are accepted as the state-of-the-art for discrete action-based deep RL.

3) **Actor Critic Methods:** Actor-critic RL models are in between policy-based algorithms and value-based algorithms due to having two estimators: actor using Q-value estimation and critic using state value function estimation (see Fig. 2). While actor controls the agent's behaviors based on policy, critic evaluates the taken action based on value function. There are recent papers that deal with the variations of actor-critic models using the deep RL approach [20], [21], [26], in which function approximators for both actor and critic are based on deep neural networks.

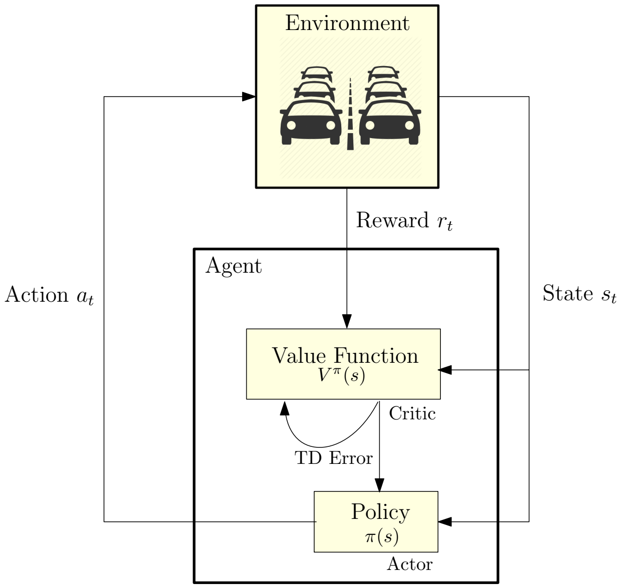


Fig. 2: Actor Critic control loop.

Standard DQN techniques with single network estimator are suitable for low-dimensional discrete action spaces. A recent actor-critic algorithm called deep deterministic policy gradient (DDPG) is introduced for solving high-dimensional continuous control problems with deterministic policy gradient approach estimating over state space instead of stochastic policy gradient estimating over state and action spaces together [20]. One of the differences of DDPG from standard DQN is that it uses a new soft target update model doing frequent soft updates.

4) **Asynchronous Methods:** Improvements in hardware systems allowed RL researchers to perform parallel computing with multiple CPUs or GPUs, which increases the learning pace. First parallel models tested on DQNs advanced the agent performance in terms of lower training time and higher convergence results. For instance, the asynchronous multiple actor-learner model proposed in [27] achieve very high performance in both continuous and discrete action spaces. Multiple actor-learners enable the RL agent to explore the environment with different exploration rates. Furthermore, asynchronous updates do not require replay memory, and learners use accumulated multiple gradients of all experiments done in a predefined update period T. Asynchronous advantage actor-critic (A3C), a state-of-the-art deep RL algorithm, updates policy and value networks asynchronously over parallel processors. Each network is separately updated within the update period T, and the shared main network is updated with respect to the parameters θ^{π} and θ^{V}. The synchronous and simpler version of A3C is known as advantage actor-critic (A2C).

C. Summary of Deep RL

In this section, we discussed the background of deep RL, including policy-based and value-based RL models. Before discussing the details of deep RL applications in ITS, it is worth mentioning that certain deep RL algorithms are preferred in different applications depending on the specifications of application domain. While developing new deep RL techniques is an active research area, Q-learning based DQN and actor-critic based DDPG algorithms continue to dominate the RL-based ITS controllers. For high-dimensional state spaces, deep RL methods are preferred over standard RL methods. With regard to action space, policy-based deep RL methods are more suitable for continuous action spaces than value-based deep RL methods. For discrete action spaces, ITS controllers typically use DQN and its variants due to their simpler structures compared to policy-based methods. In general, we can say that Q-learning based DQN models are typically used for less complicated systems which have limited state and action spaces, whereas policy-based or actor-critic algorithms are preferred mainly for large complicated systems, including multi-agent control systems. We should also note here that in many cases the designer can choose between discrete and continuous state and action spaces while setting up the problem. For instance, in TSC, as discussed in the following section, some authors define a continuous action as how much time to extend green light while some other authors define a discrete action space as choosing the green light direction.

IV. Deep RL Settings for TSC

Up to this point, we discussed the importance of AI in traffic systems and theoretical background of RL, in particular deep RL. One of the main application areas of deep RL in ITS is controlling signalized intersections. Since most of the existing works are application-oriented, proposed methods differ from each other in various aspects – e.g., applying deep RL to different intersection models with different technology to monitor traffic, characterizing the RL model with different state-action-reward representations, and using different neural network structures. Hence, a direct performance comparison between them is usually not possible.

In these applications, a learning algorithm (deep RL in our case) is implemented in the TSC center to control traffic signals adaptive to the traffic flow. First, the control unit collects the state information, which can be in different formats such as queue length, position of vehicles, speed of vehicles etc., and then control unit takes an action based on the current policy of proposed deep RL method. Finally, agent (control unit) gets a reward with respect to the taken action. By following these steps agent tries to find an optimal policy in order to minimize the congestion on intersection.

Dealing with the TSC problem on simulators by using RL algorithms requires a good problem formulation in several parts: state, action, reward definitions and neural network structure. In this section, we will discuss these main deep RL configurations together with the traffic simulators used in the literature.

A. State

The learning performance is highly dependent on an accurate and concrete state definition. Therefore, there are many different state representations used for RL applications on traffic lights. Authors in [28] and [29] considered raw RGB images as a state representation following the same approach as the original DQN [19]. Another similar image-like state representation takes the snapshot of the controlled intersection for forming position and speed of the vehicles [30]. Image-like representation format, called discrete traffic state encoding (DTSE), is one of the most popular state definitions in the TSC applications [29], [31]–[39]. The reason why researchers prefer to use DTSE is that it acquires the highest available resolution and a realistic set of information from the intersection. Considering n lanes in an intersection, each intersection is divided into cells whose size is on average one vehicle starting from the stopping point of intersection to m meters back. Speed and position of vehicles, signal phases, and accelerations are shown in separate arrays in DTSE. Different variations of those four input types are selected by different researcher. For example, while some researchers select speed and position together [31], [33], some others select only one of the four input types for the state representation, such as the position of vehicles [23], [29]. While DTSE considers the lane characteristics only, [30] considers full camera view that also includes road side information in the state definition. Today, many intersections have high quality cameras monitoring the traffic in intersection. To enable DTSE-type state representation, these equipment can be easily extended for monitoring the roads connecting to intersections.

Another common approach for state representations is forming a feature-based value vector. Instead of vehicle-based state representation, in feature-based state form, average or total value of specific information for each lane is represented on a vector. Queue length, cumulative waiting time in a phase cycle, average speed on a lane, phase duration (green, red, yellow), and number of vehicles in each lane are some of the common features used for state representation. Typically, a combination of such information is collected from intersection [40]–[43]. One advantage of such information is that they can be easily collected by road sensors or loop detectors. There are also some other unique traffic features that are not commonly used by researchers such as scoring based on the max speed on lane detectors [41], signal control threshold metrics [44], and left turn occupations [45]. Two common forms of state representations, DTSE and feature vector, are shown in Fig 3.

For TSC models with multiple intersections, state definitions also include neighboring traffic light information such as signal phase, number of vehicles, and average speed [34], [44], [46].

B. Action

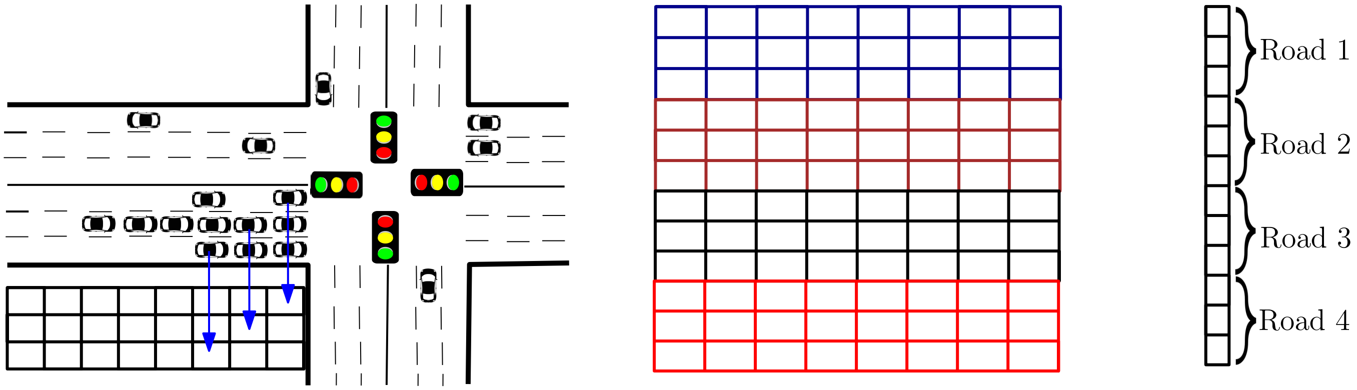
The action taken by the RL algorithm from a set of possible actions after receiving the state has a critical impact on learning. In a single four-way intersection, each direction is controlled with green, red and yellow phases. There are several common action selections for a single intersection. The most common one is choosing one of the possible green phases. Another one is the binary action selection keeping the same phase or changing the direction. Third and relatively less common action model is updating the phase duration with a predefined length.

For a single intersection, mostly there are 4 possible green phases; North-South Green (NSG), East-West Green (EWG), North-South Advance Left Green (NSLG), East-West Advance Left Green (EWLG). During the green phase for a direction, vehicles proceed through the intersection to the allowed direction. When the action selection setting is to select one of the possible green phases, deep RL agent selects an action from these four green phases at each time t. After following the yellow and red transitions, the chosen action is performed on traffic lights. Successful agent learning and safety traffic also depend on right red and yellow phase definitions. The early applications simplify the phase definitions to two green phases only, North-South Green (NSG) and East-West Green (EWG) [32], [40] ignoring the left turns. Another action selection model is binary action, in which green phase interval length is defined beforehand, and at each time t, agent decides to either maintain the same phase or proceed to the next phase in a predefined cycle, e.g., NSG → EWG → NSLG → EWLG. When agent selects the action to change the phase, before executing the next green phase, yellow and red transition phases are executed first to have a safe traffic flow [33], [37], [38], [42], [47].

Most of the applications consider discrete action selection from a set of actions, however there are also a few applications that consider continuous outputs [20], that only controls the duration of the next phase. This type of action definition mostly suitable for multiple intersections. Based on the predefined min and max phase duration, algorithm predicts a time length for the current phase [41], [48].

C. Reward

States in RL can be a feature vector or a high-dimensional matrix, and similarly actions can be a continuous value or a vector of discrete choices. However, reward is always a scalar value which is a function of the traffic data. The role of reward in RL is analyzing the quality of taken action with respect to the state, i.e., penalizing or awarding the agent for the corresponding action. Waiting time, cumulative delay, and queue length are the most common reward definitions in TSC. Waiting time is given by the sum of the times that vehicles are stopped. Delay is the difference between the waiting times of continuous green phases. Queue length is calculated for each lane in an intersection. A special congestion function in transportation planning defined by U.S. Bureau of Public Roads (BPR) is used in some works for the reward definition [34], [47]. While in some works absolute value of the traffic



Vehicle-based state array for one road in a 4-way intersection

Image-like representation of one intersection with 4 incoming roads

Feature-based state vector for one intersection

Fig. 3: Two popular types of state representation: DTSE matrix (middle) and feature-based vector (right). Left figure shows the traffic model with the corresponding vehicle-based state array. In each cell, one vehicle is represented. The matrix in the middle shows a full matrix for one intersection with each road in different colors. Right figure is a feature-based state vector, where each cell represents a lane.

data is used as a reward, in some others negative value or average value are also used.

D. Neural Network Structure

The structure of deep neural networks has a high impact on learning in deep RL. Thus, different neural network structures are proposed for TSC in the literature. Multi-layer perceptron (MP), i.e., the standard fully connected neural network model, is a useful tool for classic data classification. An extension of multi-layer perceptron with kernel filters is convolutional neural network (CNN), which provide high performance on mapping image to an output. Standard DQN considers CNN that uses consecutive raw pixel frames for state definition. There are many TSC papers that use CNNs for DTSE state definitions (see Fig. 3), e.g., [31], [33], [49]. Residual networks (ResNet) are used to deal with the overfitting problem in CNN-based deep network structures [34]. Another convolution-based network structure for operations in graphs is graph convolutional networks (GCN). Recurrent neural networks (RNN), e.g., Long Short-Term Memory (LSTM), are designed to work with sequential data. Since in TSC controlling is done sequentially, RNN is also used in deep RL settings [35], [37]. Another type of neural network model is autoencoder that learns an encoding for high-dimensional input data in a lower-dimensional subspace. The encoded input can be decoded to reconstruct the input, which is commonly used for clearing the noise on input data [40].

E. Simulation environments

RL and deep RL applications for TSC are mostly performed on traffic simulators due to life-threatening conditions in real-world experiments. Some authors also use real datasets for experimental study, but still they create a simulation environment based on the real data [50]. Microscopic individual vehicle-based simulators have been used throughout the years for ITS applications. The earliest available traffic simulator is the Java-based Green Light District (GLD) traffic simulator [51], that was initially proposed for an RL-based TSC problem. Many RL papers perform their experiments on the GLD simulator (see Table II), however the most popular open source traffic simulator is Simulation Urban Mobility (SUMO) [52]. Open source platforms allow users to modify the simulator for their purposes freely. SUMO enables users to interact with the environment using Python through the traffic control interface (TraCI) library. Different traffic models can be dynamically simulated, including personal vehicles, public vehicles and pedestrians. AIMSUN is a commercial traffic simulator designed and marketed by Transport Simulation Systems (Spain) [53]. Paramics is one of the well-known traffic simulators distributed by Quadstone Paramics (UK) [54]. VISSIM [55] is a simulator preferred by researchers due to its interaction with MATLAB, similar to AIMSUN.

V. Deep RL Applications for TSC

This section focuses on (deep) RL studies for adaptive TSC. Summary of the works are shown in separate tables for both RL and deep RL models. We can classify learning-based models into two groups in terms of the number of agents: single agent RL which learns the optimal policy with one agent for the entire TSC network, and multi-agent RL which uses multiple agents in the network for acquiring optimal policy. For both standard RL and deep RL-based TSC works, we will discuss the proposed models based on their characteristic features such as state, action, reward definitions, and neural network structure.

A. Standard RL applications

1) **Single agent RL:** Optimizing intersections with a learning agent receives high attention from researchers since the second half of 1990s. The agent interacts with a simulation

TABLE I: Outline of Single Agent RL approaches for TSC

Work	RL method	State	Action	Reward	Result comparison
Thorpe et al. [56]	SARSA	Vehicle count Fixed vehicle distance Variable vehicle distance	Binary phase	Fixed penalty (-1)	Fixed-time Different states
Abudlhai et al. [57]	Q-learning	Queue length	Binary phase	Total delay	Fixed-time
Camponogara et al. [58]	Q-learning	Position & # vehicles	Green & Red phases	# waiting vehicles	Random policy Longest queue first
Wen et al. [59]	SARSA	# vehicles	Binary phase	Coefficients of state	Fixed-time Actuated control
El-Tantawy et al. [60]	Q-learning	# vehicles / Queue length Queue length Cumulative delay	Green phases	Change in cum. delay	Fixed-time
El-Tantawy et al. [61]	Q-learning SARSA TD error	# vehicles / Queue length Queue length Cumulative delay	Binary phase Green phases	Immediate delay Cumulative delay Queue length # stops	Fixed-time Actuated control
Shoufeng et al. [62]	Q-learning	Total delay	Time change in green phase	Total delay	Fixed-time
Toubhi et al. [63]	Q-learning	Max. residual queue	Green phase duration	Queue length Cumulative delay Throughput	Vehicle demand

environment to learn an optimum control policy for traffic intersection using an RL algorithm. The ultimate goal is mostly controlling a network of coordinated intersections, but the initial step of this research is targeting how to control a single intersection with RL. Now we present some RL-based single intersection studies with their distinct features.

Traffic signal control with RL-based machine learning is pioneered by the work [56], which applies the model-free SARSA algorithm on a single intersection. In this work, *Thorpe and Anderson* considered two scenarios: a four-lane intersection without yellow transition phase, and a 4×4 grid style connected intersections where each intersection learns its own Q values separately. After this initial research, several solutions are proposed for single-intersection and multi-intersection traffic networks, where coordinated multi-agent and multi-objective RL dominate the RL research for adaptive TSC. Another SARSA-based TSC method for a single intersection is proposed by *Wen et al.* [59] with a stochastic control mechanism considering more realistic traffic situations. A specific state space is introduced in this work by partitioning the number of vehicles into sparse discrete values. Authors showed that their proposed model outperforms the fixed-time controller and actuated controller with respect to number of vehicles in queue.

In [57], authors proposed a model-free Q-learning algorithm for a single intersection with queue length as the state representation and total delay between two action cycles as the reward function. This is the first paper that proposes a simple binary action model that switches the phase direction only. The results of this work are compared with the fixed-time signal controller in different traffic flow patterns in terms of average vehicle delay. A similar Q-learning based RL model is proposed by *Camponogara and Kraus Jr.* [58] based on a distributed Q-learning technique on two intersections by assigning separate Q values for each individual agent.

Abdulhai et al. [60], proposed the first RL-based real intersection scenario in Toronto, Canada by using Q-learning with three different state definitions. First state definition is a two-valued function: number of arriving vehicles to the current green direction and number of queued vehicles in the red direction. Other states are defined as queue length and cumulative delay regardless of traffic light. The variable-phase action model in this work selects a green phase among four possible phases defined for a single intersection (NSG, EWG, NSLG, EWLG) instead of a binary action model in a fixed cycle. The same work is extended to a more general concept discussing several on-policy, off-policy RL algorithms on various state, action, reward definitions in an experimental view [61]. Along with the three state representations and the variable-phase action model in [60], authors tested their models with also the binary action model in a fixed green-phase cycle, and four reward functions, which are immediate delay, cumulative delay, queue length and the number of stops. Different RL algorithms, namely Q-learning, SARSA, and TD error are tested in different state, action, and reward settings on a single intersection. Further, two different multi-intersection configurations, 5 intersections in Toronto downtown and a large-scale network of Toronto downtown, are considered for comparison with fixed-time signal control, and actuated signal control models on Paramics simulator. *Toubhi et al.* [63], assessed three reward definitions, queue length, cumulative delay, and throughput, with Q-learning on a single intersection. The performance of each reward definition is explored on high demand and low demand traffic patterns. There are some other works that also deal with the single intersection control problem using the Q-learning approach [62], [64]. The summary of the presented works are given in Table I.

2) **Multi-agent RL:** Applying single agent RL algorithms individually on different intersections can be a good solution up to a certain point, however large intersection networks suffer from this approach. A cooperative learning approach is needed to reach an optimum policy over all network. Several multi-agent learning models are proposed for controlling multiple intersections cooperatively.

While most of the RL applications for single intersection consider model-free algorithms, such as Q-learning and SARSA, early multi-agent papers proposed model-based RL algorithms that form a transition probability distribution. A

TABLE II: Overview of Multi-agent RL approaches for TSC

Work	RL method	Solution approach	Scenario	Simulator	Result comparison
Wiering [65]	Model-based RL	Waiting time sharing	3 by 2 grid	Not specified	Fixed-time controller Random controller Largest queue first
Steingrover et al. [66]	Model-based RL	Congestion value sharing	12 mixed intersections	GDL	TC-1 [65]
Iša et al. [67]	Model-based RL	Congestion & accident value sharing	12 mixed intersections	GDL	TC-1 [65] TC-most Accident car removing
Kuyer et al. [68]	Model-based RL	Coordination graph based max plus	3 intersections 4 intersections 15 mixed intersections	GDL	TC-1 [65] TC-SBC [66] Max-plus
Bakker et al. [69]	Model-based RL	Partially observed MDP	15 mixed intersections	GDL	TC-1 [65] Diff. partial observation techniques
Houli et al. [70]	Model-based RL	Multi-objective learning	Real road map in Beijing	Paramics	Fixed controller Actuated control Single agent RL
Brys et al. [71]	SARSA	Multi-objective learning Tile coding	2 by 2 grid	AIM	Actuated control Distributed learning [72]
Khamis et al. [73]	Model-based RL with Bayesian trans. func.	Multi-objective learning	12 mixed intersections	GLD	TC-1 [65]
Khamis et al. [74]	Model-based RL with Bayesian trans. func.	Multi-objective learning Agent Cooperation	12 mixed intersections	GLD	TC-1 [65]
Khamis et al. [75]	Model-based RL with Bayesian trans. func. Hybrid exploration	Multi-objective learning Agent Cooperation	22 mixed intersections	GLD	TC-1 [65] SOTL [76]
Jin et al. [77]	SARSA with function approximators	Multi-objective learning Threshold lexicographic ordering	3 intersections in Stockholm	SUMO	Comparison between multiple function approximators
Prashanth et al. [78]	Q-learning Actor-critic	Function approximation	2 by 2 grid 5 intersectins	GLD	Fixed-time control No function approx.
Prashanth et al. [79]	Q-learning	Function approximation	2 by 2 grid 3 by 3 grid 5 intersections 9 intersections	GLD	Fixed-time control No function approx. SOTL [76]
Pham et al. [80]	SARSA	Tile coding	2 by 2 grid	AIM	Random RL Distributed learning [72]
Abdoos et al. [81]	Q-learning	2-level hierarchical control	3 by 3 grid	AIMSUN	1-level Q-learning
Arel et al. [82]	Q-learning	Neural networks Hierarchical control	5 intersections	MATLAB	Longest queue first
El-Tantawy et al. [83]	Q-learning	Indirect coordination Direct coordination	5 intersections	Paramics	Comp. between proposed models
El-Tantawy et al. [84]	Q-learning	Indirect coordination Direct coordination	Real road map in downtown Toronto	Paramics	Fixed-time control Semi-actuated control Full actuated control
Salkham et al. [85]	Q-learning	Adaptive round robin based collaboration	Real road map in Dublin City	UTC	Independent RL Fixed-time SAT-like [86]
Aziz et al. [87]	Av. expected reward	Multi-reward structure	8 intersections 11 intersections	VISSIM	Q-learning SARSA Fixed-time control Adaptive control
Aslani et al. [88]	Actor-critic	Tile coding Radial basis functions	Real road map in downtown Tehran	AIMSUN	Q-learning Fixed-time control Actuated control
Xu et al. [89]	Q-learning	Non-zero sum based Markov game	3 by 3 grid	MATLAB	Ind. Q-learning Fixed-time control Longest queue first
Abdoos et al. [90]	Q-learning	State discretization	50 intersections	AIMSUN	Fixed-time control
Balaji et al. [91]	Q-learning	Neighboor cooperation	Real road map in Singapore	Paramics	Hierarchical MS [92] Cooperative ensemble Actuated control
Cahill et al. [93]	Q-learning	CUSUM-based pattern change detection	Real road map in Dublin City	UTC	SAT-like [86]
Araghi et al. [94]	Q-learning	Distributed learning	3 by 3 grid	Paramics	Fixed-time control

prominent multi-agent RL work for large traffic networks is [65] by *Wiering*, in which three algorithms were proposed, namely TC-1, TC-2, TC-3, based on the coordination between vehicles and intersections considering local and global information for the state function. States are formed based on the traffic light configuration of intersections, position of the vehicles and the destination of the vehicles at each intersection. The approach for creating a state representation in this early work is not realistic due to unknown destination for each vehicle. The proposed models iteratively update value functions to minimize the waiting times of vehicles. The results are compared with four standard TSC models: fixed-time control, random control, longest queue first, and most-car model. Several works extended the *Wiering*'s approach in different perspectives. For example, *Steingrover et al.* proposed an extension to the TC-1 method by including congestion information on other intersections [66]. Two different extensions, called TC-SBC and TC-GAC, are proposed by the authors. The former increases the state size by adding congestion values to the state space, whereas the latter uses a congestion factor while computing the value function instead of increasing the state space. *Isa et al.* [67] proposed a further improvement to the TC-1 method by including congestion and accident information in the state representation, which further increases the state representation. While the works presented until now does not consider coordination between agents for joint action selection, *Kuyer et al.* introduces a new approach that enables coordination between agents by using the max-plus algorithm [68]. In this model, agents coordinate with each other to reach optimum joint actions in finite iterations. Another multi-agent RL model is proposed by *Bakker et al.* [69] with partial observations for the state spaces of connected intersections. This case is of interest when the system cannot access the full state information due to some reasons such as faulty sensors. All these works [66]–[69] use *Wiering*'s approach [65] as a benchmark.

Multi-objectivity is gaining popularity in RL [95] due to its capabilities in complex environments. When a single objective is selected for the overall traffic system, such as the *Wiering*'s objective which aims to decrease the waiting time of all vehicles, it may not serve well the needs of different traffic conditions. Authors in [70] consider a multi-objective approach in their multi-agent RL work for TSC. In particular, vehicle stops, average waiting time, and maximum queue length are targeted as objectives for low, medium, and high traffic volume, respectively. Different Q functions are updated with appropriate reward functions in these three traffic conditions. *Taylor et al.* proposes a non-RL based basic learning algorithm called Distributed Coordination of Exploration and Exploitation (DCEE) [72] to tackle the TSC problem. Authors in [71] and [80] consider a multi-agent RL-based SARSA algorithm with tile coding and compare it with DCEE under different traffic conditions.

Khamis et al. studied multi-objective RL control for traffic signals in three papers [73]–[75]. In the first paper [73], authors considered Bayesian transition probability for model-based RL using several objectives for forming the reward function. The second paper followed the same approach [74] with more specific objectives. The third paper [75] extends the previous works to a total of seven objectives with a novel cooperative exploration function and experiments in several road conditions and vehicle demands. The paper also improves the practicality of GLD traffic simulator from different perspectives, e.g., continuous control, probabilistic travel demand. The results of these three papers are compared with TC-1 proposed by *Wiering* [65] and the adaptive SOTL method [76]. The latest and most compact RL-based multi-objective multi-agent TSC study is presented in [77]. In this work, travel delay and fuel consumption are defined as learning objectives for the RL agents, and a specific technique called threshold lexicographic ordering is used for online multi-objective adaptation. SARSA is experimented in this work with several function approximators, one of which is based on neural networks. It is worth noting that SARSA with Q-value estimation is not considered a deep RL approach since it does not include experience replay and target network tricks discussed in III-B1.

Before DQN was introduced, function approximators were popular for Q-functions with large state space. For instance, authors in [78], [79] proposed two RL models for TSC using function approximation-based Q-learning and actor-critic policy iteration. The proposed Q-learning method outperforms standard Q-learning with full state representation. A novel neural network-based multi-agent RL for TSC is proposed in [82], which uses local agents and global agents. While local agent controls the traffic lights via the longest queue first algorithm, global agent controls the traffic lights with a neural network-based Q-learning approach, which is very similar to DQN discussed in III-B.

Actor-critic-based multi-agent RL is an emerging field that uses continuous state representation. Discretizing the state space is prone to missing information about the state. *Aslani et al.* proposed a continuous space actor-critic control model for multiple intersections [88], in which tile coding and radial basis-based function approximators are presented. Although state space is continuous, action space to determine the duration of the next green phase is discrete. In experiments, discrete and continuous state space based actor-critic models are tested in the city of Tehran. In another work, two-layer hierarchical multi-agent RL method is studied [81], which implements a single agent for each intersection using Q-learning, and controls a wide area network with function approximator based on tile coding on second layer.

There are several studies offering coordination between the neighbor agents for reaching a joint optimum performance. To this end, *Tantawy et al.* proposed a Q-learning based multi-agent RL approach for road network coordination [83], [84]. RL agents learn the coordination directly or indirectly, called MARLIN-DC and MARLIN-IC. While a small-scale road network is presented in [83], in the extended paper [84] authors investigate a large network of 59 intersections in downtown Toronto. [91] presents another coordination-based TSC model implementing distributed Q-learning agents for a large network where neighbour agents share congestion values with each other. Experiments are performed on the Paramics simulation environment using a real traffic network in Singapore with different travel demand configurations. *Xu et al.* [89] proposed a coordination module based on nonzero-sum Markov game for the multi-agent RL environment. Q-learning is used on each intersection as a single agent, and their coordination is controlled with a Markov game-based mathematical model.

A new technique for multiple intersection environments is proposed in [87] using the R-Markov Average Reward technique and a multi-objective reward definition for RL. The result of this work is compared with fixed-time controller, actuated controller, Q-learning and SARSA on the Paramics simulation environment by simulating an 18-intersection network. *Chu et al.* [96], proposed a regional to central multi-agent RL model for large-scale traffic networks. In low traffic density, authors claim that for large-scale networks collaboration is not needed between regions, i.e., learning the traffic model in local region is enough to attain a globally appropriate learning. *Araghi et al.*, [94] presents a distributed Q-learning based multi-agent RL controller that predicts the green phase duration on the next phase cycle. Other multi-agent RL applications are studied in [85], [90], [93]. Table II gives an overview of the multi-agent RL works.

B. Deep RL applications

Here we discuss deep RL-based TSC applications considering. A summary of the discussed works is provided in Table III considering the used deep RL algorithms, network structures, simulation environments, and comparison with benchmarks.

1) **Single agent deep RL:** In recent years, deep RL based learning tools for adaptive intersection controls gained a great attention from transportation researchers. After researchers proposed several architectures for different traffic scenarios using standard RL in the last two decades, invention of deep RL made a huge impact on the ITS research, in particular TSC. Due to its capability of dealing with large state space, a number of deep RL models have been proposed for controlling traffic lights. The deep RL paradigm is basically based on approximating Q-functions with deep neural networks. The earliest work using this approach is [82]. Although a neural network-based RL model is proposed in this paper, it is not a full DQN algorithm due to lack of experience replay and target network, which are essential components of DQN [19].

The initial work on controlling traffic signals with a deep RL approach is [31] by *Genders et al.*. In this work, authors use discrete traffic state encoding model, called DTSE, to form an image-like state representation based on detailed information from the traffic environment. The proposed state model is an input to CNN for approximating the Q-values of discrete actions. The experiments are performed on the SUMO simulation environment with a single intersection where 4 green phases are selected as actions. In order to show the power of CNN on the DTSE state form, the results are compared with Q-learning using a single layer neural network. In [98], the same authors studied the effects of different state representations for intersection optimization using the A3C algorithm. Three separate state definitions are experimented on a single intersection using a dynamic traffic environment. The first form of state definition considered in the paper is given by the occupancy and average speed of each lane. The second state definition is the queue length and vehicle density for each lane. The third state form is the image like representation, DTSE, with Boolean position information in which the existence of vehicle is represented with 1. The results show that the resolution of state representation does not effect the performance of RL agent in terms of delay and queue length. The same authors, in a recent paper [103], studied asynchronous deep RL model for TSC. In asynchronous n-step Q-learning [27], the main job is divided to multiple processors, and each processor learns its local optimal parameters individually. Global parameters for the general network is updated after every n-step. The proposed architecture in [103] improves the performance almost 40% compared to the fixed-time and actuated traffic controllers.

Authors in [40], proposed an autoencoder-based deep RL algorithm for a single intersection with dynamic traffic flow. Autoencoders are considered for action selection by mapping input queue length to a low-dimensional action set. Bottleneck layer, which is the output of decoding part, is used for Q-function approximation. The results are compared with standard Q-learning using the Paramics simulator. Currently, this is the only work in the literature that uses autoencoders to approximate action values. In [33], *Gao et al.* proposed a new neural network architecture in which state is a combination of the speed and position of vehicles based on DTSE. The output of neural network is binary action whether to keep the same action or change the action in a predefined phase cycle. The proposed model is compared with the fixed-time controller and the longest queue first controller.

The authors in [28], presented two deep RL algorithms for controlling isolated intersections: value-based DQN, and policy-based actor critic. The state for both agents is raw consecutive image frames following exactly the same approach with original DQN. As stated in the original paper [19], DQN algorithm suffers from instability issues. [28] shows that the policy-based deep RL technique solves this issue by having a smooth convergence and a stable trend after convergence. *Shabestary et al.* [36] proposed a DQN-based solution for adaptive traffic signal control on an isolated intersection using a new reward definition. The reward and action defined in this paper are change in the cumulative delay and 8 different green phases, as opposed to the commonly used binary action set or 4 green phases for a single intersection.

Choe et al. proposed a RNN-based DQN model in a single intersection TSC scenario [37]. It is shown that the performance of RNN-based DQN decreased the travel time compared to the popular CNN structure. A policy gradient-based deep RL method is proposed for adaptive traffic intersection control in [29], which presents experiments on a novel realistic traffic environment called Unity3D by using raw pixels as an input state to policy-based DQN. The proposed model has similar results with the fixed-time intersection control model. An action value-based DQN with a novel discount factor is proposed by *C. Wan et al.* [45]. The proposed dynamic discount factor takes execution time into account with the help of infinite geometric series. The proposed model is tested on a single intersection using the SUMO simulator by comparing

TABLE III: Outline of Deep RL approaches for TSC.

Work	Deep RL neural network structure	Multi-agent	State - DTSE	Scenario	Simulator	Result comparison
Genders et al. [31]	DQN - CNN	No	Yes	Single int.	SUMO	MP(64) DQN
Van der Pool et al. [49]	DQN - CNN	Max-plus Transfer planning	Yes	Single int. 2 intersections 3 intersections 2 by 2 grid	SUMO	Model based RL
Van der Pool et al. [32]	DQN - CNN	Max-plus Transfer planning	Yes	2 intersections 3 intersections 2 by 2 grid	SUMO	Model based RL
Li et al. [40]	DQN - Autoencoder	No	No	Single int.	Paramics	Q-learning
Gao et al. [33]	DQN - CNN	No	Yes	Single int.	SUMO	Fixed-time control Longest queue first
Liu et al. [34]	DQN-ResNeT [97]	Policy sharing	Yes	2 by 2 grid	SUMO	SOTL DQN without CNN Q-learning
Casas [41]	DDPG	Multiple actor- critic learner	No	Single int., 6 intersections, Real map from Barcelona	Aumsim	Q-learning Random
Shi et al. [35]	DQN-RNN	Max-plus Transfer planning	Yes	2 by 2 grid	USTCMTS2.1	Fixed-time control Q-learning
Mousavi et al. [28]	DQN-CNN A2C-CNN	No	Real image	Single int.	SUMO	Fixed-time control
Lin et al. [42]	A2C-CNN	Multiple actor- critic learners	No	3 by 3 grid	SUMO	Fixed-time control Actuated control
Genders et al. [98]	A3C-MP	No	Yes	Single int.	SUMO	Actuated control
Shabestary et al. [36]	DQN-CNN	No	Yes	Single int.	Paramics	Different rewards Q-learning
Choe et al. [37]	DQN-RNN	No	Yes	Single int.	SUMO	CNN-DQN
Garg et al. [29]	DQN-CNN (Policy based)	No	Yes	Single int.	Unity3d	Fixed-time control No traffic light
Coskun et al. [99]	DQN-CNN Actor-Critic	Joint learning	No	4 intersections	SUMO	DQN(Policy based)
Wei et al. [38]	DQN-CNN	No	Yes	Single int.	SUMO Real dataset	Fixed-time SOTL
Natafgi et al. [50]	DQN-CNN	No	No	Single int.	SUMO	Fixed tim
Nishi et al. [100]	NFQI-Graph CNN [101]	No	No	6 intersections	SUMO	Fixed-time CNN-DQN
Wan et al. [45]	Modified DQN	No	No	Single int.	VISSIM	DQN Fixed-time control
Calvo et al. [39]	DQN-CNN	Independent DQN fingerprints	Yes	3 intersections	SUMO	Fixed-time control
Genders [48]	DDPG	Multiple learners	No	Real map from Luxemburg	SUMO	Fixed-time control
Chu et al. [102]	A2C-RNN	Policy sharing	No	5 by 5 grid Monaco city map	SUMO	Ind-Q-learning Ind-DQN Ind-A2C
Liang et al. [30]	Double Dueling DQN-CNN	No	Yes	Single int.	SUMO	Fixed-time control Actuated control DQN
Genders et al. [103]	Asynchronous n-step Q-learning	No	No	Single int.	SUMO	Linear learning Actuated control Random control
Zhou et al. [44]	DQN-MP	Threshold based	No	Real map from New york city	SUMO	Diff veh. demands
Xu et al. [47]	DQN-RNN	Critical node discovery	No	20 intersections 50 intersections 100 intersections	SUMO	Fixed-time SOTL Q-learning DQN
Tan et al. [104]	DQN (Value based) DDPG (Wolpertinger)	Hierarchical cooperation	No	6 intersections 12 intersections 24 intersections	SUMO	Fixed-time control Q-learning DQN
Ge et al. [46]	DQN-CNN	Q value transfer	Yes	Heterogeneous 4 int. 2 by 3 grid	SUMO	Dist. Q-learning DQN
Liu et al. [105]	DQN-MP	No	No	Single int. 4 intersections	Python	No comparison
Zhang et al. [106]	DQN	No	No	Arterial topology 4 by 4 grid	SUMO	Partially Observable states

it with the fixed-time controller and the standard DQN-based controller.

A new DQN-based controller, called IntelliLight, with a new network architecture is described in [38]. The reward function consist of multiple components: sum of the queue length over all lanes, sum of delay, sum of waiting time, traffic light state indicator, number of vehicles that passed the intersection since the last action, and sum of travel times since the last action. The proposed method is experimented on SUMO using a single intersection. A real dataset collected from real cameras in China is used as an input to SUMO. IntelliLight is selected as a benchmark in [107], which introduces a new transfer learning model with a batch learning framework. The same real-world data and a synthetic simulation data which generates traffic with uniform distribution is used on an isolated intersection for experiments. Another DQN-based study for traffic light control with a real dataset is presented in [50]. Data from a three-way non-homogeneous real intersection in Lebanon is used. The experimental results are compared with the real-world fixed-time controller that is in use at the intersection in terms of queue length and delay.

A different deep RL model in terms of action set and the deep RL algorithm is studied by *Liang et al.*, [30]. This work updates the next phase duration in the phase cycle instead of choosing an action from a green phase set. Considering a 4-phase single intersection, phase change duration is defined. The selected phase duration can be added or subtracted from the duration of the next cycle phase. In this model, for a four-way intersection the action set includes 9 discrete actions. The proposed algorithm in this paper considers new DQN techniques, namely double dueling DQN and prioritized experience replay, to improve the performance. In another paper, *Jang et al.* [43], discusses how to integrate a DQN agent with a traffic simulator through the Java-based AnyLogic multipurpose simulator. A different approach for state definitions is proposed by *Liu et al.* [105] for examining the impacts of DQN on green-wave patterns in a linear road topology. The experiments are performed only on a Python environment that creates traffic data from a probability distribution without using any traffic simulator. Moreover, considering the Dedicated Short-Range Communication (DSRC) technology for vehicle-to-infrastructure (V2I) communication, *Zhang et al.* [106] addresses TSC under partial detection of vehicles at an intersection. Their motivation to study TSC with undetected vehicles comes from the case where not all vehicles use DSRC.

2) **Multi-agent deep RL:** The first deep RL-based multiple intersection control mechanism is presented in [32], which defines a new reward function and proposes a coordination tool for multiple traffic lights. The reward definition in this paper considers a combination of specific traffic conditions, namely accidents or jam, emergency stops, and traffic light changes, and the waiting time of all vehicles. The reward function properly penalizes each specific traffic situation. For coordination of multiple intersections to have a high traffic flow rate, this paper uses a transfer planning technique for a smaller set of intersections and links the learning results to a larger set of intersections with the max-plus coordination algorithm. In this work, the benchmark is one of the early coordination-based RL methods proposed in [65]. As expected, the DQN-based coordination method outperforms the earlier standard RL-based method. This paper is expanded to a master thesis [23] by presenting the results for single agent scenario and different multi-agent scenarios. In [35], similar to [32], a multi-agent deep RL approach on a 2-by-2 intersection grid model is proposed, in which max-plus and transfer learning are used for reaching the global optimal learning with coordination. This paper differs from [32] mainly by using RNN, in particular LSTM, layers instead of fully connected layers for Q-function approximation. Deep RL approach with RNN structure is shown to result in lower average delay compared to Q-learning and fixed-time control in both low and high traffic demand scenarios.

Liu et al. [34] introduced a cooperative deep RL model for controlling multiple intersections with multiple agents. The presented algorithm is DQN with a ResNet structure used to form the state space. The reward function penalizes the system based on the driver behavior and waiting time with a BPR function (see Section IV-C). Cooperation between agents is assured by sharing the policy with other agents every n-step. Experiments for this study are done using a 2-by-2 intersection model on SUMO. SOTL, Q-learning and DQN are selected as reference points for validating the proposed model.

Multiple traffic intersections can be represented as a network graph in which lane connections between roads form a directed graph. *Nish et al.* [100] presented a GCN-based neural network structure for the RL agent. GCN is combined with a specific RL algorithm called k-step neural fitted Q-iteration [101] that updates the agent in a distributed manner by assigning one agent for each intersection considering the whole network to form the state space. The experiment results show that the GCN-based algorithm decreases the waiting time on all 6 intersections compared to the fixed-time controller and standard CNN-based RL controller. A hierarchical control structure is presented in [44] for TSC. The lower layer optimizes the local area traffic via intersection control while the top layer optimizes the city-level traffic by tuning the degree of optimization of local areas in the lower layer. In this research, multi-intersection learning is built on threshold values collected from individual intersections. The action set of higher level controller is increasing or decreasing the threshold values that change the sensitivity of each intersection to the neighbor intersections. The learning model in this paper is different from the other deep RL-based intersection controllers such that the model decreases the algorithm complexity in higher level control through a threshold-based mechanism instead of setting the phase cycles or phase duration.

Cooperative multi-agent deep RL model is investigated in [39]. Here, an agent with an independent double dueling DQN model supported with prioritized experience replay is assigned to each intersection. In order to improve the coordination performance, a special sampling technique, fingerprint, is used in experience replay. Fingerprint technique estimates Q-functions with neighbor agent's policy via Bayesian inference [108]. The proposed model is tested on SUMO with heterogeneous multiple intersections. The results show that the proposed algorithm outperforms the fixed-time controller and the DQN controller without experience replay on several travel demand scenarios.

One of the approaches in multi-agent systems is updating only the critical edges to increase the efficiency. [47] first identifies important nodes based on multiple criteria with a specific ranking algorithm, CRRank, that creates a trip network using a bidirectional tripartite graph. Based on data and tripartite graph, system ranks the edges based on assigned scores. Once critical intersections are identified, RNN structured DQN agent learns the optimal policy. The model is tested with 20, 50 and 100 intersections on SUMO comparing its results with fixed-time, SOTL, Q-learning and DQN controllers. Recently, a cooperative deep RL method with Q-value transfer is proposed in [46]. At each intersection, a DQN agent controls the traffic light by receiving Q-values from other agents for learning the optimal policy. The proposed algorithm is supported with extensive experiments on homogeneous and heterogeneous intersections. It is important to have a heterogeneous traffic scenario because all the intersections do not have the same characteristics such as the number of roads and number of lanes. The authors compared their results with two benchmark papers: coordinated Q-learning [32] and distributed Q-learning [94] approaches.

The work in [41] investigates applying the deep deterministic policy gradient (DDPG) algorithm to a city-scale traffic network. The author formulated the TSC problem with DDPG by controlling the phase duration continuously. The model updates phase duration of all network at once by keeping the total phase cycle constant in order to control the synchronization throughout the network. In this work, a specific information called speed score, calculated using the maximum speed on each detector, is considered for forming the state vector. Three traffic scenarios are tested from small to large networks: isolated intersection, 2-by-3 grid intersections, and a Barcelona city-scale map with 43 intersections. The proposed approach achieves higher reward performance than multi-agent Q-learning controller. It is remarkable that actor critic models can be applied large intersection models without any extra multi-agent control technique. Another DDPG-based deep RL controller for large-scale network is studied by *Genders* in his PhD thesis [48]. The system model consists of a parallel architecture with decentralized actors for each intersection and central learners each of which cover a subset of the intersections. The policy determines the duration of the green phase in each intersection. To test the performance of the model, Luxembourg city map is used on SUMO with 196 intersections, which is the largest test environment for RL-based TSC up to now.

A multiple actor-learner architecture considering the A2C algorithm is presented for multiple intersections by *Lin et al.* in [42]. Multiple actors observe different states, and follow different exploration policies in parallel. Since actor-critic approaches are built on advantage functions, authors consider a technique called general advantage estimation function in the learning process [109]. The presented experiments are performed on a 3-by-3 intersection grid on SUMO, and the results are compared with the fixed-time controller and actuated controller.

Independent Q-learning is one of the popular multi-agent RL approaches in literature. *Chu et al.* [102] recently expanded this approach to independent A2C for multi-agent TSC. The stability problem is addressed with two methods, fingerprints of neighbor intersections and a spatial discount factor. While the former provides each agent with information regarding the local policies and traffic distributions of neighbor agents, the latter enables each agent to focus on improving the local traffic. The network structure in the A2C algorithm is an LSTM-based RNN model. Both synthetic traffic network with a 5-by-5 grid and a real network from Monaco City with 30 intersections are used for performance evaluation.

Systematic learning for large-scale traffic networks is achieved with cooperation in [104]. A large system is divided into subsets in which each local region is controlled with an RL agent. Global learning is achieved by transferring learning policies to the global agent. For local controllers, authors investigated two deep RL algorithms: value based per-action DQN and actor-critic based Wolpertinger-DDPG [110]. Per-action DQN is similar to the standard DQN algorithm, but differs from DQN by considering state-action pair as an input and generating a single Q value. Wolpertinger-DDPG provides a new policy method based on the k-nearest-neighbor approach using DDPG for large-scale discrete action spaces. In experiments, three different traffic networks are used, and the results are compared with a decentralized Q-learning algorithm with linear function approximators, and two rule-based baselines (fixed-time and random-time controllers).

Coskun et al. [99] expands [28], which use value-based DQN and policy-based standard actor-critic, to multiple intersections using value-based DQN and policy-based A2C. The results of both algorithms following deep learning is consistent with the results of standard RL approaches in terms of average reward per episode, where DQN hits higher average reward than A2C.

VI. Deep RL for Other ITS Applications

Several useful deep RL mechanisms have been introduced for various other applications in ITS. One of the major application areas of AI techniques in ITS is autonomous vehicles, where deep RL occupies a great place in this context. Autonomous controlling is studied from various aspects using deep RL approaches. Ramp metering, lane changing, speed acceleration/deceleration, maneuvering on intersections are some of the various examples studied with deep RL (see Table IV).

A. Autonomous Driving

Initial papers presenting deep RL-based control for autonomous vehicles experiment their models on the TORCS game environment [151]. A control framework proposed by *Sallab et al.* [111] uses two types of deep RL methods, DQN approach with RNNs for discrete action set, and actor-critic based DDPG approach for continuous action domain. The authors experimented the algorithms without using replay memory on TORCS, which led to a faster convergence. *Xia et al.* [112] studied a control strategy called deep Q-learning

TABLE IV: Outline of deep RL approaches for other ITS applications

Case Study	Deep RL method	Target application	Solution	Test	Result comparison
Sallab et al. [111]	DQN DDPG	Autonomous driving	Spatial aggregation, Recurrent temporal aggregation	TORCH game	RNN-LSTM Kalman-GRNN
Xia et al. [112]	DQN with filtered experience replay	Autonomous driving	Optimum control	TORCH game	NFQ [101]
Xiong et al. [113]	DDPG	Autonomous driving	Collision avoidance	TORCH game	-
Sharifzadeh et al. [114]	DQN	Autonomous driving	Lane changing	Personal simulator	Expert driver
Hoel et al. [115]	AlphaGo Zero	Autonomous driving	Decision planning with Monte carlo tree search	Personal simulator	MCTS IDM/MOBIL
Hoel et al. [116]	DQN	Autonomous driving	Speed change Lane change	Personal simulator	CNN-FCNN IDM
Chae et al. [117]	DQN	Autonomous breaking	Pedestrian detection	PreScan vehicle simulator	Without Trauma memory
Shi et al. [118]	Hierarchical DQN	Autonomous driving	Safe gap adjustment Lane changing	Personal simulator	-
Wang et al. [119]	Rule-based DQN	Autonomous driving	Lane changing	Udacity simulator	Different policy structures
Ye et al. [120]	DDPG	Autonomous driving	Lane changing Car following	VISSIM	IDM
Makansis et al. [121]	DDQN	Autonomous driving	Optimum highway control	SUMO	DP
Yu et al. [122]	Multi-agent Q-learning	Autonomous driving	Coordination graphs	Personal simulator	Expert driver Independent Q-learning
Qian et al. [123]	Twin delay DDPG	Autonomous driving	Path planning	Personal simulator	Expert driver DQN
Zhou et al. [124]	DDPG	Autonomus driving	Optimum control in TSC intersections	Personal simulator	Human driver Policy gradient DQN
Osinski et al. [125]	PPO2 [126]	Autonomous driving	Optimum control	Real world CARLA	Continuous driving model
Huang et al. [127]	DDPG	Autonomous driving	Human in the loop training	IPG CarMker	Imitation learning
Isele et al. [128]	DQN	Autonomous driving	Navigating in occluded intersections	SUMO	TTC [129]
Kreidieh et al. [130]	TRPO [131]	Stop-and-go wave dissipation	Transfer learning	Flow	Human driver Random policy
Chalaki et al. [132]	TRPO	Ramp metering	Policy transfer to city scale map Adversarial noise injection	Scaled smart city	Human Driver
Jang et al. [133]	TRPO	Ramp metering	Policy transfer to city scale map	Scaled smart city	IDM controller
Belletti et al. [134]	TRPO	Ramp metering	Multi-task conrol	Personal simulator	REINFORCE PPO [135]
Chaoui et al. [136]	DQN	Electric vehicle	Energy management with multiple batteries	Personal simulator	-
Wu et al. [137]	DDPG	Hybrid electric bus	Adaptive energy management to road conditions	Personal simulator	DQN DP
Hu et al. [138]	DQN	Hybrid electric vehicle	Energy management	MATLAB ADVISOR [139]	Different training models
Wu et al. [140]	DDPG	Freeway control	Variable speed limit	SUMO	Q-learning DQN
Wu et al. [141]	ES [142]	Freeway control	Ramp meter Speed limit Lane change	SUMO	No control DQN-RM TRPO-RM DDPG-DVSL
Pandey et al. [143]	Sparce cooperative Q-learning [144]	Toll roads	Dynamic lane management	Personal simulator	Density based Ratio based Random search
Pandey et al. [145]	Vanilla PG Proximal PG [126]	Express lane pricing	Multi-objective opt. Transfer learning	Personal simulator	Feedback Control
Gunarathna et al. [146]	Multi-agent Q-learning	Lane direction change	Dynamic coordination graphs	New york real taxi trips	Different lane changing models
Min et al. [147]	QR-DQN	Driver assistant	Lane keeping Lane change Acceleration control	Unity [148]	DQN DDQN
Schults et al. [149]	DQN	Traffic simulator	Calibrating traffic models	-	-
Bacchiani et al. [150]	A3C	Traffic simulator	Calibrating traffic models	-	-

with filtered experiences (DQFE) for teaching autonomous vehicle how to drive. The learning performance is shown to outperform the neural fitted Q-learning technique on the TORCS game simulator.

A continuous control strategy proposed in [113] combines the DDPG algorithm for continuous actions with a safety control strategy. The combination is needed because only relying on past experiences does not provide a safe autonomous vehicle control. *Hoel et al.* [115] introduced an autonomous driving model including planning and learning with Monte Carlo tree search and deep RL. Driving planning is done with Monte Carlo tree search and learning how to drive is done with deep RL agent using the AlphaGO Zero algorithm [152]. In that work, the proposed method is compared with a baseline called IDM/MOBIL agent for expert driver behaviours [153], [154].

Authors in [120] studied car following and lane changing behaviours of autonomous vehicles using DDDP method on VISSIM. Another RL-based autonomous driving policy is described by *Makantasis et al.* [121] using DDQN with prioritized experience replay in mixed autonomy scenarios. Proposed deep RL-based driving policy is compared with DP-based optimal policy in different traffic densities using SUMO. Deep RL autonomous driving research generally targets individual agents in a mixed autonomy environment or a fully autonomous environment for finding the best driving strategy. However, authors in [122] proposed a multi-agent deep RL approach with dynamic coordination graph. In this study, autonomous vehicles in coordination learn how to behave in a highway scenario. Two distinct coordination graph models, identity-based dynamic coordination and position-based dynamic coordination, are studied in that work. *Qian et al.* [123] described autonomous driving from a different perspective using twin delayed DDPG [155]. They proposed a two-level strategy to fill the gap between decision making and future planning of autonomous vehicle. Autonomous driving in a signalized traffic intersection using DDPG method is proposed by *Zhou et al.* [124]. In a recent autonomous driving study [125], RL methods are analyzed on the traffic simulator CARLA [156] using RGB image inputs collected from a camera. A different training and test strategy is experimented by authors in [127] for DDPG-based autonomous driving using a human-in-the-loop dynamical simulator called IPG CarMaker. While a human driver controls the vehicle on this software, the DDPG agent learns how to drive in two distinct scenarios, forward driving and stopping.

In transportation research, controlling stop-and-go waves with autonomous vehicles is a new approach for which a deep RL-based solution is suggested in [130]. The authors implemented multiple autonomous vehicles controlled by individual deep RL agents to increase the flow of traffic. *Isele et al.* [128] using the DQN approach studied a special case for self-driving vehicles, maneuvering in intersections when driver have partial knowledge about the intersection. In this paper, three action selection modes are tested. First action mode is stop or go, the second mode is having sequential actions, accelerate, decelerate or keep constant velocity, and the last action mode is the combination of first two action modes, wait, move slowly or go. All three action modes are tested on 5 different cases.

The authors in [116] proposed a speed and lane changing framework for autonomous truck-trailer with surrounded vehicles using double DQN. This work considers several traffic situations including a highway traffic and a two-way traffic scenario called overtaking in order to generalize the proposed algorithm. Using an inverse deep RL approach, *Sharifzadeh et al.* [114] presented a driving model for collision-free lane changing on a self-programmed traffic simulator with continuous trajectories. The investigated model includes two separate agents. One agent controls only lane changing without speed adjustment, and the other agent controls lane changing actions with acceleration. Another lane changing application for autonomous vehicles is presented in [118] considering DQN with quadratic Q-function approximator. A hierarchical control technique is implemented as a lane changing module in discrete domain, and a gap adjusting module in continuous domain with separate deep RL agents. Similar to the other papers, authors in [119] proposed a rule-based DQN approach for the lane changing problem for autonomous vehicles.

Most of the learning based control models in ITS test the proposed work on simulators such as autonomous vehicle control, traffic signal control, traffic flow control. The first learned policy transfer from simulator to real world experiments is studied by *Chalaki et al.* [132]. Experiment platform for this research is a scaled city map from University of Delaware, in which behaviors of multiple autonomous vehicles in a roundabout is observed with deep RL control techniques. In order to transfer policies efficiently, adversarial noise is injected into the state and action spaces. The initial results of the same work for single agents with Gaussian noise is studied in [133].

B. Energy Management

Energy management systems are a crucial part of future transportation. There are different resource allocation schemes for electric vehicles. Power consumption varies in different units of vehicle that highly effects the performance of batteries. *Chaoui et al.* proposed a deep RL-based energy management solution to increase the life cycle of parallel batteries [136]. Authors in [138] suggest an optimization model for energy consumption in hybrid vehicles using the DQN formulation. Proposed adaptive learning model provides a better fuel consumption through deep RL-based energy management scheme. *Wu et al.* [137] proposed an energy management solution for hybrid electric buses using an actor-critic based DDPG algorithm. Considering two parameters, number of passengers and traffic information, deep RL agent can optimize the energy consumption with continuous controlling.

C. Road Control

Road controllers are an essential part of traffic control in ITS. There are several works which use deep RL methods for speed limit control, toll road pricing, ramp metering, etc.Dynamic speed limit control among lanes is a challenging task in transportation. *We et al.* [140] studies a dynamic solution method with actor-critic continuous control scheme for variable speed limits control, that increases the flow rate and decreases the emission rate. Deep RL-based lane pricing model for toll roads is proposed in [143] to maximize the total revenue with multiple entrance and exits. Another dynamic lane pricing model for express lanes is proposed in [145], where authors used multi-objective RL and multi-class cell transmission models to enhance the performance of deep RL agent.Highway connections from side roads are controlled with signalized ramp meters. In order to increase the efficiency of the main road flow, a new multi-agent deep RL technique is proposed in [134] for traffic models based on discretized partial differential equations. The control model is tested on a simulated highway scenario with multiple ramp meters. *Wu et al.* [141] proposed a freeway control model using deep RL with various agents for different parts of freeway. Authors' proposal is to use an inflow ramp meter control agent, a dynamic lane speed limit control agent, and a dynamic lane change controller agent in coordination. Traditional roads have fixed number of lanes for incoming and outgoing directions. Lane direction change is studied for improving the traffic flow with multi-agent deep RL and dynamic graph configuration in [146].Autonomous braking system via DQN is proposed in [117], which provides traffic safety in cases where immediate action is required.

D. Various ITS Applications

Recently a new tool for optimizing traffic simulators is proposed by *Schultz at al.* [149]. The input, (traffic characteristics) and output (traffic congestion) of traffic simulators are correlated with an adaptive learning technique using DQN. Another computational interface, named Flow, enables easy integration of the deep RL library RLlib [157] with SUMO and Aimsun for various control problems in ITS [158]. *Flow* users can create a custom network via Python to test complex control problems such as ramp meter control, adaptive traffic signalization and flow control with autonomous vehicles. Authors in [150] introduces a traffic simulator which provides a new environment with cooperative multi-agent learning approach for analyzing the behaviours of autonomous vehicles. It is capable of testing various traffic scenarios. *Min et al.* [147] proposed a driver assistant system using quantile regression DQN for various controls such as lane keeping, lane changing, and acceleration control.

VII. Challenges and Open Research questions

Despite the significant interest and effort, and the promising results so far in deep RL-based ITS solutions, there are still many major challenges to address before the proposed research can yield real-world products. In this section, we will discuss the major challenges and open research questions of deep RL for ITS.

All the research outcomes for RL-based ITS controls are experimented on simulators due to life threatening consequences of real-world applications. Recently, authors in [132] presented a policy transfer application from simulation to a city-scale test environment for autonomous driving, but still this line of research is in its infancy. There is a huge gap between real-world deployment and simulator-based applications using learning algorithms. For TSC and other controlling applications in ITS, a real-world deployment is needed in order to prove the applicability of deep RL-based automated control.

Specifically for TSC, simulation-based applications have two approaches in literature, first, simulating an artificial road network with artificial data, second, simulating a road network based on a real dataset. While the second one is close to a realistic test, it only considers the traffic demand in various times of the day without realistic challenges. Another point that researchers need to consider for TSC is increasing the realism of simulation environments, such as including the human intervention scenarios. In order to decrease human intervention in TSC, the control system should be adaptable to unstable traffic situations in the worst case scenarios. To do that, instead of standard traffic models, urban networks with some predictable extreme scenarios should be studied in order to see the consequences of deep RL implementations. We expect that implementing pedestrians and public transportation to the simulation environments will have a high impact on learning performance.

There are so many proposed deep RL models in the literature for controlling traffic lights. While standard RL models have comparisons between each other for validating their proposals, deep RL models on TSC do not have satisfactory comparisons with existing works. For multiple intersections, researchers mostly selected DQN, standard RL, and fixed-time controllers as benchmark. However, they should be especially compared with other multi-agent approaches in the literature, such as distributed control, coordinated control, etc. Another challenge with results is that very few papers compare their performance with actuated controller, which is the most popular real-world TSC implementation.

State definition is a crucial point in deep RL applications. Thus, researchers pay attention to different state forms with different hardware systems such as cameras, loop detectors, and sensors, but still there is no clear agreement on the form of state in deep RL-based TSC applications. State definition highly depends on static devices, hence all of them should always collect data properly. A new research direction could be studying partially observable and noisy state definitions in which some of the devices do not work properly. When RL-based adaptive traffic signals are implemented on intersections, the system must be protected and stable (i.e., robust and resilient) against such kind of failures.

Regarding autonomous vehicles, researchers have been proposing solutions to very specific subsystems without considering the interaction between such subsystems. For more realistic solutions, a unified management and adaptive control strategy is required for several components. For example, an impactful deep RL application should control lane changing, breaking, flow arranging, and energy management components all together. Implementing different learning algorithms for different autonomous vehicle subsystems may cause interoperability issues.

VIII. Conclusion

Considering the increasing world population and urbanization, researchers have been conducting research on ITS applications using learning-based AI techniques. Dynamic nature of traffic systems does not allow a clear easy control mechanism for all ITS applications. Controlling transportation systems through reinforcement learning (RL) approaches is gaining popularity in both industry and academia. There are various research outcomes in recent years for solving automated control problems in ITS, such as traffic lights, autonomous driving, autonomous break, and energy management of vehicles. The most popular deep RL application in ITS is adaptive traffic signal control (TSC) at intersections. We presented a comprehensive review for the deep RL applications in ITS. Key concepts of RL and deep RL, and the settings in which they are applied to TSC were discussed to provide a smooth introduction to the literature. Characteristic details of existing works in several categories were compared in separate tables in order to enable clear comparison. Finally, we also discussed the open research directions and the gap between the existing works and the real-world usage. This survey showed that there are different single agent and multi-agent RL solutions for TSC that outperform the standard control methods in simulation environments. However, existing works have still not been tested in real-world environments except for an autonomous vehicle application for a specific scenario.

References

[1] G. Cookson, "Inrix global traffic scorecard," *INRIX research*, 2018.

[2] Z. Liu, "A survey of intelligence methods in urban traffic signal control," *IJCSNS International Journal of Computer Science and Network Security*, vol. 7, no. 7, pp. 105–112, 2007.

[3] A. L. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," *Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 3, p. 342, 2009.

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

[5] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, "A survey on reinforcement learning models and algorithms for traffic signal control," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, p. 34, 2017.

[6] W. Tong, A. Hussain, W. X. Bo, and S. Maharjan, "Artificial intelligence for vehicle-to-everything: A survey," *IEEE Access*, vol. 7, pp. 10 823–10 843, 2019.

[7] R. Abduljabbar, H. Dia, S. Liyanage, and S. Bagloee, "Applications of artificial intelligence in transport: An overview," *Sustainability*, vol. 11, no. 1, p. 189, 2019.

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

[9] M. Veres and M. Moussa, "Deep learning for intelligent transportation systems: A survey of emerging trends," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[10] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *arXiv preprint arXiv:2002.00444*, 2020.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[12] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[13] G. A. Rummery and M. Niranjan, *On-line Q-learning using connectionist systems*. University of Cambridge, Department of Engineering Cambridge, England, 1994, vol. 37.

[14] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, no. 3, pp. 1247–1293, 2013.

[15] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[16] L. C. Baird, "Reinforcement learning in continuous time: Advantage updating," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 4. IEEE, 1994, pp. 2448–2453.

[17] L. Busoniu, R. Babuska, and B. De Schutter, "Multi-agent reinforcement learning: A survey," in *2006 9th International Conference on Control, Automation, Robotics and Vision*. IEEE, 2006, pp. 1–6.

[18] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, p. 436, 2015.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[21] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[22] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[23] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[24] H. V. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, 2010, pp. 2613–2621.

[25] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[26] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih, "Combining policy gradient and q-learning," *arXiv preprint arXiv:1611.01626*, 2016.

[27] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016, pp. 1928–1937.

[28] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," *IET Intelligent Transport Systems*, vol. 11, no. 7, pp. 417–423, 2017.

[29] D. Garg, M. Chli, and G. Vogiatzis, "Deep reinforcement learning for autonomous traffic light control," in *2018 3rd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*. IEEE, 2018, pp. 214–218.

[30] X. Liang, X. Du, G. Wang, and Z. Han, "A deep reinforcement learning network for traffic light cycle control," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1243–1253, 2019.

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

[32] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, 2016.

[33] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network," *arXiv preprint arXiv:1705.02755*, 2017.

[34] M. LIU, J. DENG, M. XU, X. ZHANG, and W. WANG, "Cooperative deep reinforcement learning for tra ic signal control," 2017.

[35] S. Shi and F. Chen, "Deep recurrent q-learning method for area traffic coordination control," *Journal of Advances in Mathematics and Computer Science*, pp. 1–11, 2018.

[36] S. M. A. Shabestary and B. Abdulhai, "Deep learning vs. discrete reinforcement learning for adaptive traffic signal control," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 286–293.

[37] C.-J. Choe, S. Baek, B. Woon, and S.-H. Kong, "Deep q learning with lstm for traffic light control," in *2018 24th Asia-Pacific Conference on Communications (APCC)*. IEEE, 2018, pp. 331–336.

[38] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2496–2505.

[39] J. J. A. Calvo and I. Dusparic, "Heterogeneous multi-agent deep reinforcement learning for traffic lights control," in *The 26th Irish Conference on Artificial Intelligence and Cognitive Science*, 2018, pp. 1–12.

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

[41] N. Casas, "Deep deterministic policy gradient for urban traffic light control," *arXiv preprint arXiv:1703.09035*, 2017.

[42] Y. Lin, X. Dai, L. Li, and F.-Y. Wang, "An efficient deep reinforcement learning model for urban traffic control," *arXiv preprint arXiv:1808.01876*, 2018.

[43] I. Jang, D. Kim, D. Lee, and Y. Son, "An agent-based simulation modeling with deep reinforcement learning for smart traffic signal control," in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2018, pp. 1028–1030.

[44] P. Zhou, T. Braud, A. Alhilal, P. Hui, and J. Kangasharju, "Erl: Edge based reinforcement learning for optimized urban traffic light control."

[45] C.-H. Wan and M.-C. Hwang, "Value-based deep reinforcement learning for adaptive isolated intersection signal control," *IET Intelligent Transport Systems*, vol. 12, no. 9, pp. 1005–1010, 2018.

[46] H. Ge, Y. Song, C. Wu, J. Ren, and G. Tan, "Cooperative deep q-learning with q-value transfer for multi-intersection signal control," *IEEE Access*, 2019.

[47] M. Xu, J. Wu, L. Huang, R. Zhou, T. Wang, and D. Hu, "Network-wide traffic signal control based on the discovery of critical nodes and deep reinforcement learning," *Journal of Intelligent Transportation Systems*, pp. 1–10, 2018.

[48] W. Genders, "Deep reinforcement learning adaptive traffic signal control," Ph.D. dissertation, 2018.

[49] E. van der Pol, "Deep reinforcement learning for coordination in traffic light control," *Master's thesis, University of Amsterdam*, 2016.

[50] M. B. Natafgi, M. Osman, A. S. Haidar, and L. Hamandi, "Smart traffic light system using machine learning," in *2018 IEEE International Multidisciplinary Conference on Engineering Technology (IMCET)*. IEEE, 2018, pp. 1–6.

[51] M. Wiering, J. Vreeken, J. Van Veenen, and A. Koopman, "Simulation and optimization of traffic in a city," in *IEEE Intelligent Vehicles Symposium, 2004*. IEEE, 2004, pp. 453–458.

[52] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo–simulation of urban mobility: an overview," in *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[53] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday, "Traffic simulation with aimsun," in *Fundamentals of traffic simulation*. Springer, 2010, pp. 173–232.

[54] G. D. Cameron and G. I. Duncan, "Paramicsparallel microscopic simulation of road traffic," *The Journal of Supercomputing*, vol. 10, no. 1, pp. 25–53, 1996.

[55] M. Fellendorf and P. Vortisch, "Microscopic traffic flow simulator vissim," in *Fundamentals of traffic simulation*. Springer, 2010, pp. 63–93.

[56] T. L. Thorpe and C. W. Anderson, "Traffic light control using sarsa with three state representations," Citeseer, Tech. Rep., 1996.

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

[58] E. Camponogara and W. Kraus, "Distributed learning agents in urban traffic control," in *Portuguese Conference on Artificial Intelligence*. Springer, 2003, pp. 324–335.

[59] K. Wen, S. Qu, and Y. Zhang, "A stochastic adaptive control model for isolated intersections," in *2007 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2007, pp. 2256–2260.

[60] S. El-Tantawy and B. Abdulhai, "An agent-based learning towards decentralized and coordinated traffic signal control," in *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2010, pp. 665–670.

[61] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Design of reinforcement learning parameters for seamless application of adaptive traffic signal control," *Journal of Intelligent Transportation Systems*, vol. 18, no. 3, pp. 227–245, 2014.

[62] L. Shoufeng, L. Ximin, and D. Shiqiang, "Q-learning for adaptive traffic signal control based on delay minimization strategy," in *2008 IEEE International Conference on Networking, Sensing and Control*. IEEE, 2008, pp. 687–691.

[63] S. Touhbi, M. A. Babram, T. Nguyen-Huu, N. Marilleau, M. L. Hbid, C. Cambier, and S. Stinckwich, "Adaptive traffic signal control: Exploring reward definition for reinforcement learning," *Procedia Computer Science*, vol. 109, pp. 513–520, 2017.

[64] Y. K. Chin, L. K. Lee, N. Bolong, S. S. Yang, and K. T. K. Teo, "Exploring q-learning optimization in traffic signal timing plan management," in *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks*. IEEE, 2011, pp. 269–274.

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

[66] M. Steingrover, R. Schouten, S. Peelen, E. Nijhuis, B. Bakker *et al.*, "Reinforcement learning of traffic light controllers adapting to traffic congestion." in *BNAIC*. Citeseer, 2005, pp. 216–223.

[67] J. Iša, J. Kooij, R. Koppejan, and L. Kuijer, "Reinforcement learning of traffic light controllers adapting to accidents," *Design and Organisation of Autonomous Systems*, pp. 1–14, 2006.

[68] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis, "Multiagent reinforcement learning for urban traffic control using coordination graphs," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 656–671.

[69] B. Bakker, S. Whiteson, L. Kester, and F. C. Groen, "Traffic light control by multiagent reinforcement learning systems," in *Interactive Collaborative Information Systems*. Springer, 2010, pp. 475–510.

[70] D. Houli, L. Zhiheng, and Z. Yi, "Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network," *EURASIP journal on advances in signal processing*, vol. 2010, no. 1, p. 724035, 2010.

[71] T. Brys, T. T. Pham, and M. E. Taylor, "Distributed learning and multi-objectivity in traffic light control," *Connection Science*, vol. 26, no. 1, pp. 65–83, 2014.

[72] M. E. Taylor, M. Jain, P. Tandon, M. Yokoo, and M. Tambe, "Distributed on-line multi-agent optimization under uncertainty: Balancing exploration and exploitation," *Advances in Complex Systems*, vol. 14, no. 03, pp. 471–528, 2011.

[73] M. A. Khamis, W. Gomaa, and H. El-Shishiny, "Multi-objective traffic light control system based on bayesian probability interpretation," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2012, pp. 995–1000.

[74] M. A. Khamis and W. Gomaa, "Enhanced multiagent multi-objective reinforcement learning for urban traffic light control," in *2012 11th International Conference on Machine Learning and Applications*, vol. 1. IEEE, 2012, pp. 586–591.

[75] G. W. Khamis, Mohamed A, "Adaptive multi-objective reinforcement learning with hybrid exploration for traffic signal control based on cooperative multi-agent framework," *Engineering Applications of Artificial Intelligence*, vol. 29, pp. 134–151, 2014.

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

[77] J. Jin and X. Ma, "A multi-objective agent-based control approach with application in intelligent traffic signal system," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[78] B. S. Prashanth, LA, "Reinforcement learning with average cost for adaptive control of traffic lights at intersections," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2011, pp. 1640–1645.

[79] L. Prashanth and S. Bhatnagar, "Reinforcement learning with function approximation for traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 412–421, 2011.

[80] T. T. Pham, T. Brys, and M. E. Taylor, "Learning coordinated traffic light control," in *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-13)*, vol. 10. IEEE, 2013, pp. 1196–1201.

[81] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Hierarchical control of traffic signals using q-learning with tile coding," *Applied intelligence*, vol. 40, no. 2, pp. 201–213, 2014.

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

[83] S. El-Tantawy and B. Abdulhai, "Multi-agent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc)," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2012, pp. 319–326.

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

[85] A. a. Salkham, R. Cunningham, A. Garg, and V. Cahill, "A collaborative reinforcement learning approach to urban traffic control optimization," in *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*. IEEE Computer Society, 2008, pp. 560–566.

[86] S. Richter *et al.*, "Learning road traffic control: towards practical traffic control using policy gradients," 2006.

[87] H. A. Aziz, F. Zhu, and S. V. Ukkusuri, "Learning-based traffic signal control algorithms with neighborhood information sharing: An application for sustainable mobility," *Journal of Intelligent Transportation Systems*, vol. 22, no. 1, pp. 40–52, 2018.

[88] M. Aslani, M. S. Mesgari, and M. Wiering, "Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 732–752, 2017.

[89] L.-H. Xu, X.-H. Xia, and Q. Luo, "The study of reinforcement learning for traffic self-adaptive control under multiagent markov game environment," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[90] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Traffic light control in non-stationary environments based on multi agent q-learning," in *2011 14th International IEEE conference on intelligent transportation systems (ITSC)*. IEEE, 2011, pp. 1580–1585.

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

[92] C. K. Keong, "The glide systemsingapore's urban traffic control system," *Transport reviews*, vol. 13, no. 4, pp. 295–305, 1993.

[93] V. Cahill *et al.*, "Soilse: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning," in *13th International IEEE Conference on Intelligent Transportation Systems*. IEEE, 2010, pp. 531–538.

[94] S. Araghi, A. Khosravi, and D. Creighton, "Distributed q-learning controller for a multi-intersection traffic network," in *International Conference on Neural Information Processing*. Springer, 2015, pp. 337–344.

[95] C. Liu, X. Xu, and D. Hu, "Multiobjective reinforcement learning: A comprehensive overview," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 385–398, 2015.

[96] T. Chu, S. Qu, and J. Wang, "Large-scale traffic grid signal control with regional reinforcement learning," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 815–820.

[97] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[98] W. Genders and S. Razavi, "Evaluating reinforcement learning state representations for adaptive traffic signal control," *Procedia computer science*, vol. 130, pp. 26–33, 2018.

[99] M. Coşkun, A. Baggag, and S. Chawla, "Deep reinforcement learning for traffic light optimization," in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2018, pp. 564–571.

[100] T. Nishi, K. Otaki, K. Hayakawa, and T. Yoshimura, "Traffic signal control based on reinforcement learning with graph convolutional neural nets," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 877–883.

[101] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *European Conference on Machine Learning*. Springer, 2005, pp. 317–328.

[102] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[103] W. Genders and S. Razavi, "Asynchronous n-step q-learning adaptive traffic signal control," *Journal of Intelligent Transportation Systems*, vol. 23, no. 4, pp. 319–331, 2019.

[104] T. Tan, F. Bao, Y. Deng, A. Jin, Q. Dai, and J. Wang, "Cooperative deep reinforcement learning for large-scale traffic grid signal control," *IEEE transactions on cybernetics*, 2019.

[105] X.-Y. Liu, Z. Ding, S. Borst, and A. Walid, "Deep reinforcement learning for intelligent transportation systems," *32nd Conference on Neural Information Processing Systems (NIPS 2018), Montreal, Canada.*, 2018.

[106] R. Zhang, A. Ishikawa, W. Wang, B. Striner, and O. Tonguz, "Intelligent traffic signal control: Using reinforcement learning with partial detection," *arXiv preprint arXiv:1807.01628*, 2018.

[107] N. Xu, G. Zheng, K. Xu, Y. Zhu, and Z. Li, "Targeted knowledge transfer for learning traffic signal plans," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2019, pp. 175–187.

[108] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1146–1155.

[109] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[110] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.

[111] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[112] W. Xia, H. Li, and B. Li, "A control strategy of autonomous vehicles based on deep reinforcement learning," in *2016 9th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 2. IEEE, 2016, pp. 198–201.

[113] X. Xiong, J. Wang, F. Zhang, and K. Li, "Combining deep reinforcement learning and safety based control for autonomous driving," *arXiv preprint arXiv:1612.00147*, 2016.

[114] S. Sharifzadeh, I. Chiotellis, R. Triebel, and D. Cremers, "Learning to drive using inverse reinforcement learning and deep q-networks," *arXiv preprint arXiv:1612.03653*, 2016.

[115] C.-J. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *arXiv preprint arXiv:1905.02680*, 2019.

[116] C.-J. Hoel, K. Wolff, and L. Laine, "Automated speed and lane change decision making using deep reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2148–2155.

[117] H. Chae, C. M. Kang, B. Kim, J. Kim, C. C. Chung, and J. W. Choi, "Autonomous braking system via deep reinforcement learning," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[118] T. Shi, P. Wang, X. Cheng, and C.-Y. Chan, "Driving decision and control for autonomous lane change based on deep reinforcement learning," *arXiv preprint arXiv:1904.10171*, 2019.

[119] J. Wang, Q. Zhang, D. Zhao, and Y. Chen, "Lane change decision-making through deep reinforcement learning with rule-based constraints," *arXiv preprint arXiv:1904.00231*, 2019.

[120] Y. Ye, X. Zhang, and J. Sun, "Automated vehicles behavior decision making using deep reinforcement learning and high-fidelity simulation environment," *Transportation Research Part C: Emerging Technologies*, vol. 107, pp. 155–170, 2019.

[121] K. Makantasis, M. Kontorinaki, and I. Nikolos, "Deep reinforcement-learning-based driving policy for autonomous road vehicles," *IET Intelligent Transport Systems*, 2019.

[122] C. Yu, X. Wang, X. Xu, M. Zhang, H. Ge, J. Ren, L. Sun, B. Chen, and G. Tan, "Distributed multiagent coordinated learning for autonomous driving in highways based on dynamic coordination graphs," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[123] L. Qian, X. Xu, Y. Zeng, and J. Huang, "Deep, consistent behavioral decision making with planning features for autonomous vehicles," *Electronics*, vol. 8, no. 12, p. 1492, 2019.

[124] M. Zhou, Y. Yu, and X. Qu, "Development of an efficient driving strategy for connected and automated vehicles at signalized intersections: A reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[125] B. Osiński, A. Jakubowski, P. Miłoś, P. Zikecina, C. Galias, and H. Michalewski, "Simulation-based reinforcement learning for real-world autonomous driving," *arXiv preprint arXiv:1911.12905*, 2019.

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

[127] W. Huang, F. Braghin, and S. Arrigoni, "Autonomous vehicle driving via deep deterministic policy gradient," in *ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2019.

[128] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2034–2039.

[129] M. M. Minderhoud and P. H. Bovy, "Extended time-to-collision measures for road traffic safety assessment," *Accident Analysis & Prevention*, vol. 33, no. 1, pp. 89–97, 2001.

[130] A. R. Kreidieh, C. Wu, and A. M. Bayen, "Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 1475–1480.

[131] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

[132] B. Chalaki, L. Beaver, B. Remer, K. Jang, E. Vinitsky, A. Bayen, and A. A. Malikopoulos, "Zero-shot autonomous vehicle policy transfer: From simulation to real-world via adversarial learning," *arXiv preprint arXiv:1903.05252*, 2019.

[133] K. Jang, E. Vinitsky, B. Chalaki, B. Remer, L. Beaver, A. A. Malikopoulos, and A. Bayen, "Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*. ACM, 2019, pp. 291–300.

[134] F. Belletti, D. Haziza, G. Gomes, and A. M. Bayen, "Expert level control of ramp metering based on multi-task deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1198–1207, 2018.

[135] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, 2016, pp. 1329–1338.

[136] H. Chaoui, H. Gualous, L. Boulon, and S. Kelouwani, "Deep reinforcement learning energy management system for multiple battery based electric vehicles," in *2018 IEEE Vehicle Power and Propulsion Conference (VPPC)*. IEEE, 2018, pp. 1–6.

[137] Y. Wu, H. Tan, J. Peng, H. Zhang, and H. He, "Deep reinforcement learning of energy management with continuous control strategy and traffic information for a series-parallel plug-in hybrid electric bus," *Applied Energy*, vol. 247, pp. 454–466, 2019.

[138] Y. Hu, W. Li, K. Xu, T. Zahid, F. Qin, and C. Li, "Energy management strategy for a hybrid electric vehicle based on deep reinforcement learning," *Applied Sciences*, vol. 8, no. 2, p. 187, 2018.

[139] T. Markel, A. Brooker, T. Hendricks, V. Johnson, K. Kelly, B. Kramer, M. OKeefe, S. Sprik, and K. Wipke, "Advisor: a systems analysis tool for advanced vehicle modeling," *Journal of power sources*, vol. 110, no. 2, pp. 255–266, 2002.

[140] Y. Wu, H. Tan, and B. Ran, "Differential variable speed limits control for freeway recurrent bottlenecks via deep reinforcement learning," *arXiv preprint arXiv:1810.10952*, 2018.

[141] Y. Wu, H. Tan, Z. Jiang, and B. Ran, "Es-ctc: A deep neuroevolution model for cooperative intelligent freeway traffic control," *arXiv preprint arXiv:1905.04083*, 2019.

[142] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," *arXiv preprint arXiv:1703.03864*, 2017.

[143] V. Pandey and S. D. Boyles, "Multiagent reinforcement learning algorithm for distributed dynamic pricing of managed lanes," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2346–2351.

[144] J. R. Kok and N. Vlassis, "Collaborative multiagent reinforcement learning by payoff propagation," *Journal of Machine Learning Research*, vol. 7, no. Sep, pp. 1789–1828, 2006.

[145] V. Pandey, E. Wang, and S. D. Boyles, "Deep reinforcement learning algorithm for dynamic pricing of express lanes with multiple access locations," *arXiv preprint arXiv:1909.04760*, 2019.

[146] U. Gunarathna, H. Xie, E. Tanin, S. Karunasekara, and R. Borovica-Gajic, "Dynamic graph configuration with reinforcement learning for connected autonomous vehicle trajectories," *arXiv preprint arXiv:1910.06788*, 2019.

[147] K. Min, H. Kim, and K. Huh, "Deep distributional reinforcement learning based high-level driving policy determination," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 416–424, 2019.

[148] A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, and D. Lange, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, 2018.

[149] L. Schultz and V. Sokolov, "Deep reinforcement learning for dynamic urban transportation problems," *arXiv preprint arXiv:1806.05310*, 2018.

[150] G. Bacchiani, D. Molinari, and M. Patander, "Microscopic traffic simulation by cooperative multi-agent deep reinforcement learning," *arXiv preprint arXiv:1903.01365*, 2019.

[151] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the open racing car simulator," *Software available at http://torcs. sourceforge. net*, vol. 4, no. 6, 2000.

[152] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[153] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[154] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[155] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," *arXiv preprint arXiv:1802.09477*, 2018.

[156] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," *arXiv preprint arXiv:1711.03938*, 2017.

[157] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2018.

[158] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: A modular learning framework for autonomy in traffic," *arXiv preprint arXiv:1710.05465*, 2017.

Ammar Haydari received the B.Sc. degree in Electrical Engineering from Uludag University, Istanbul, Turkey, in 2014. He is currently a M.Sc. student at the Electrical Engineering Department at the University of South Florida, Tampa. His research interests include intelligent transportation systems and machine learning.

Yasin Yılmaz (S'11-M'14) received the Ph.D. degree in Electrical Engineering from Columbia University, New York, NY, in 2014. He is currently an Assistant Professor of Electrical Engineering at the University of South Florida, Tampa. He received the Collaborative Research Award from Columbia University in 2015. His research interests include statistical signal processing, machine learning, and their applications to cybersecurity, cyber-physical systems, IoT networks, transportation systems, energy systems, and communication systems.