

Received September 7, 2020, accepted October 21, 2020, date of publication October 27, 2020, date of current version November 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3034141

Deep Reinforcement Learning for Traffic Signal Control: A Review

FAIZAN RASHEED¹, KOK-LIM ALVIN YAU¹, (Senior Member, IEEE),
RAFIDAH MD. NOOR², (Member, IEEE), CELIMUGE WU³, (Senior Member, IEEE),
AND YEH-CHING LOW¹, (Member, IEEE)

¹Department of Computing and Information Systems, Sunway University, Subang Jaya 47500, Malaysia

²Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur 50603, Malaysia

³Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo 182-8585, Japan

Corresponding author: Kok-Lim Alvin Yau (koklimy@sunway.edu.my)


This work was supported in part by A Novel Clustering algorithm based on Reinforcement Learning for the Optimization of Global and Local Network Performances in Mobile Networks funded by the Malaysian Ministry of Education through Fundamental Research Grant Scheme under Grant FRGS/1/2019/ICT03/SYUC/01/1, and in part by the Partnership between Sunway University and the University of Malaya under Grant CR-UM-SST-DCIS-2018-01 and Grant RK004-2017.

ABSTRACT Traffic congestion is a complex, vexing, and growing issue day by day in most urban areas worldwide. The integration of the newly emerging deep learning approach and the traditional reinforcement learning approach has created an advanced approach called deep reinforcement learning (DRL) that has shown promising results in solving high-dimensional and complex problems, including traffic congestion. This article presents a review of the attributes of traffic signal control (TSC), as well as DRL architectures and methods applied to TSC, which helps to understand how DRL has been applied to address traffic congestion and achieve performance enhancement. The review also covers simulation platforms, a complexity analysis, as well as guidelines and design considerations for the application of DRL to TSC. Finally, this article presents open issues and new research areas with the objective to spark new interest in this research field. To the best of our knowledge, this is the first review article that focuses on the application of DRL to TSC.

INDEX TERMS Artificial intelligence, deep learning, deep reinforcement learning, traffic signal control.

I. INTRODUCTION

With rapid population growth and urbanization, traffic demand is steadily rising in metropolises worldwide. Traffic signal controls (TSCs) are installed to monitor traffic flows and alleviate traffic congestion at intersections [1]–[3]. During traffic congestion, vehicles move slowly or stop at lanes, and the queue length of the vehicles increases [4]. Congestion that occurs in a single lane has a single-point-of-failure effect as it can also affect the traffic conditions of the other lanes at the same and neighboring intersections. There are three main reasons that exacerbate congestion. Firstly, traffic entering an intersection is greater than the traffic leaving it. Secondly, *cross-blocking* occurs when vehicles cannot cross an intersection despite a green signal being activated as the respective lane of the downstream intersection has become fully occupied. Thirdly, *green idling* occurs when no vehicle is present at an intersection when a green signal is activated.

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Wang .

TSCs have three traditional signal colors: *red* indicates a *stop*, *yellow* indicates a *slow down*, and *green* indicates a *go*. A *cycle* consists of a predetermined sequence of traffic phases, and the *cycle length* is the time interval of a cycle. A *traffic phase* consists of a combination of green signals allocated to a set of lanes simultaneously for non-conflicting movements at an intersection. A short moment of all red signals is included in between traffic phases to provide safe transition, causing some time loss. A *traffic phase split* (or the green time) is the time interval, which is part of a cycle length, allocated for a traffic phase.

Traditionally, TSCs determine the traffic phase splits to manage traffic flows and alleviate traffic congestion using three main approaches. Firstly, a *deterministic TSC* applies a pre-timed control system that uses the Webster formula based on historical traffic data collected at different times [5]. Secondly, a *semi-dynamic TSC* applies an actuated control system that uses current (or instantaneous) traffic conditions, rather than longer-term traffic conditions. For example, green signals are activated at lanes with vehicles [6]. Thirdly, a

TABLE 1. List of abbreviations.

Abbreviations	
RL	Reinforcement Learning
MARL	Multi-agent Reinforcement Learning
DL	Deep Learning
DRL	Deep Reinforcement Learning
DQN	Deep Q-network
DNN	Deep Neural Network
CNN	Convolutional Neural Network
SAE	Stacked Auto Encoder
3DQN	Double Dueling Deep Q-network
LSTM	Long Short-term Memory
FC	Fully-Connected
FCLN	Fully-Connected Layer Network
PG	Policy-gradient
A2C	Advantage Actor Critic
TSC	Traffic Signal Control

fully-dynamic TSC applies an actuated control system that uses longer-term traffic conditions. For example, a traffic phase split is increased with the average waiting time and queue length of vehicles at a lane. While the semi-dynamic TSC approach uses a single inductive loop detector installed at a lane to detect the presence of vehicles, the fully-dynamic TSC uses at least two inductive detectors to measure the queue length [7]–[11].

Research has been undertaken to investigate TSCs that can optimize traffic signal scheduling and timing, such as adjusting traffic phase splits, in order to ameliorate traffic congestions at moderately and heavily trafficked single or multiple intersections. Reinforcement learning (RL) has been the preferred unsupervised artificial intelligence technique for accomplishing a fully-dynamic TSC [12]. RL possesses the capability to learn the relationships between actions and their effects on the operating environment (or states). Specifically, RL adjusts traffic phase splits, or even skips traffic phases, according to traffic conditions which are dynamic and can be unpredictable. Nevertheless, RL is marred by the curse of dimensionality, an issue whereby the number of states (or the state space) becomes too large, leading to two main shortcomings [13]. *Firstly*, a higher computational cost must be incurred to explore all state-action pairs in order to identify optimal actions, causing a longer learning time. *Secondly*, a larger storage capacity is required to store knowledge (or Q -values).

Recently, deep learning (DL), which is an advanced artificial intelligence technique, has been successfully combined with RL to provide deep reinforcement learning (DRL), and it has shown to address the shortcomings of RL [14]. DRL has three main advantages. *Firstly*, DRL enables a continuous state space representation, so there can be a large number of states. *Secondly*, DRL reduces the learning time required to explore all state-action pairs and identify optimal actions. *Thirdly*, DRL uses several layers of neurons

to store the weights (or network parameters) of the links connecting the neurons, which are used to approximate the Q -values efficiently in order to address the storage capacity issue in RL [15].

A. SIGNIFICANCE OF DEEP REINFORCEMENT LEARNING FOR TRAFFIC SIGNAL CONTROL

Several success stories of the use of DRL over the years have brought new and refreshed enthusiasm to the world of artificial intelligence. In 2013, DeepMind introduces DRL applied to playing a range of Atari 2600 games with super-human performance [14]. Later in 2016, DRL is trained by DeepMind to play the Alpha Go board game, which defeated a host of world champions [16]. Subsequently, DRL has been applied in many real-world applications, such as robotics [17], natural language processing [18], health care [19], business management [20], Industry 4.0 [21], smart grid [22], computer vision [23], transportation, particularly TSC and driver-less vehicles [24]. In view of these developments, this article presents a comprehensive review of the limited work on DRL applied to TSC, motivated by the goal of achieving better-than-human intelligence solutions. In general, DRL offers main advantages that are appealing to TSCs as follows:

- DRL enables an agent to adapt to the real-time traffic condition that evolves in a complex and unpredictable manner due to unexpected disturbances, such as bad weather conditions and road accidents.
- DRL can be a model-based or model-free approach, and it enables an agent to perform self-learning on the fly without having prior knowledge about the operating environment, including traffic condition and network. The model-based approach creates a model of the operating environment, and then selects an action and observes feedback from the model [25]. On the other hand, the model-free approach does not create a model of the operating environment. The model-free approach has been chosen for TSC because it has lower complexity and computational requirement compared to the model-based approach.
- DRL represents reward with system goal(s) and performance measure(s), which take account of multiple factors that affect system performance from the operating environment. For instance, the reward changes with the average waiting time, queue length, or throughput of vehicles at an intersection (see Section IV).
- DRL addresses the curse of dimensionality, which adversely affect the traditional RL approach, when applied to TSC. This is because TSC has a large state space as there are multiple factors affecting a traffic network.

Most importantly, DRL provides added advantages compared to other approaches applied to TSC as shown in Table 2.

Meanwhile, the trend of the number of papers published from January 2016 to March 2020 on the use of DRL to TSC

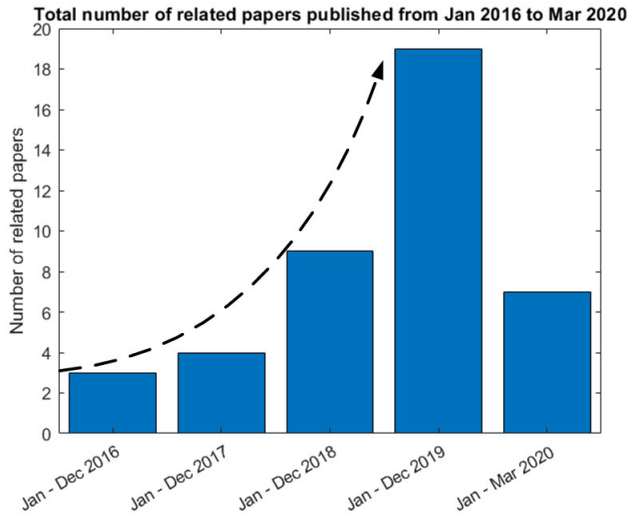


FIGURE 1. The trend of related papers published in recent years on the use of DRL to TSC.

is shown in Fig. 1. This study was conducted on three well-known literature databases with scientific scope, namely Web of Science, ScienceDirect, and IEEEXplore Digital Library.

B. OUR CONTRIBUTIONS

While general reviews of designing TSCs using RL [13], [30]–[32], multi-agent systems [29], big data [34], DL [35], and other artificial intelligence approaches, such as fuzzy systems [33], have been presented, this article complements their works by focusing on the DRL approach, particularly on how DRL models can be applied to formulate the TSC problem, and how the strengths of various DRL approaches can provide added advantages in addressing the challenges brought about by traffic management and control. To the best of our knowledge, this is the first comprehensive article that contributes to the body of knowledge by providing systematic and extensive synthesis, analysis and summary of limited DRL schemes applied to TSC, which helps to identify research gaps in existing schemes and explore future research directions. Various technical aspects of DRL-based TSCs, including DRL models, DRL methods, DL architectures, simulation platforms, complexity analysis and performance measures, are covered to enhance the technicality of article.

C. ORGANIZATION OF THIS ARTICLE

The rest of this article is organized as shown in Fig. 2. Section II presents an overview of DL, RL, and DRL, as well as various DL architectures with RL methods. The simulation platforms are also presented. Section III presents the attributes of TSC systems. Section IV presents the representations of DRL models and complexity analysis for TSC. Section V presents the application of DRL to TSC. Section VI presents the guidelines and design considerations for the application of DRL to TSC. Section VII presents open issues. Finally, Section VIII concludes this article.

II. BACKGROUND

This section presents an overview of DL, RL, and DRL, as well as various DL architectures with RL methods. In addition, simulation platforms are presented.

A. DEEP LEARNING

Deep learning (DL) is an advanced artificial intelligence approach that consists of a *deep neural network* (DNN), such as a fully-connected layer network (FCLN) [36], [37]. The term “deep” indicates that the neural network consists of a large number of hidden layers (e.g., up to 150 layers [38]), which may be fully-connected (FC) with each other, while a traditional neural network generally consists of a much lower number of hidden layers (e.g., two or three layers [39]). Fig. 3 shows a FCLN architecture that consists of three main types of layers, namely the *input*, *hidden*, and *output* layers, and it is an interconnected assembly of *neurons* (i.e., processing elements) that are capable of learning unstructured and complex data [40]. During training, data flows from the input layer to the output layer. The output y_k of a neuron k in the hidden and output layers is as follows [41]:

$$y_k = \varphi \left(\sum_{j=0} w_{kj} \cdot x_j \right) \quad (1)$$

where w_{kj} represents the weight (or network parameter), which is assigned on the basis of the relative importance of input x_j compared to other inputs, and $\varphi(\cdot)$ represents the activation function at neuron k .

There are various kinds of DL architectures applied to TSC, including the traditional FCLN, convolutional neural network (CNN), stacked auto encoder (SAE), dueling network, and long short-term memory (LSTM) (see Section II-D for more details).

B. REINFORCEMENT LEARNING

Reinforcement learning (RL) is the third paradigm of artificial intelligence, which is different from the supervised learning and unsupervised learning approaches. It enables an agent to explore and exploit different state-action pairs so that it achieves the best possible positive reward (or negative cost) for system performance enhancement as time goes by $t = 1, 2, 3, \dots$ [42]–[45]. Algorithm 1 presents the traditional RL algorithm [42]. At time instant $t \in T$, an agent observes its Markovian (or memoryless) decision-making factors (or *state* $s_t \in S$) in the dynamic and stochastic operating environment, and selects and performs an *action* $a_t \in A$ [46]–[49]. Subsequently, the agent observes the next state s_{t+1} and receives an *immediate* reward (or cost) $r_{t+1}(s_{t+1})$, which depends on the next state s_{t+1} for the state-action pair (s_t, a_t) . Then, it updates Q -value $Q_t(s_t, a_t)$, which represents knowledge, for the state-action pair. The Q -value $Q_t(s_t, a_t)$ represents the appropriateness of taking action a_t under state s_t , and it is updated using Q -function as follows [50]:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha \delta_t(s_t, a_t) \quad (2)$$

TABLE 2. Comparison of DRL with other approaches applied to TSC.

Approach	Description	Limitation	Solution
Fuzzy Logic [26]	Defines membership functions that map inputs to outputs, taking analog (or continuous) values between 0 and 1. It provides partially true and false values, instead of absolute values.	Larger values may fall outside the range of 0 and 1 due to the effects of disturbances, such as bad weather conditions and road accidents, to TSC.	DRL has the ability to approximate the Q -value of a given state due to the spatial correlation nature between neurons [15], which helps DRL to better deal with the effects of disturbances.
Genetic Algorithm [27]	Defines a set of parameters or variables, namely genes, which are joined to form a string of genes called chromosome. The algorithm creates an initial population of multiple genes and chromosomes randomly, and then runs itself for a predefined time period.	Larger number of chromosomes and complicated representations are needed as the problem expands and becomes more complex, such as a large traffic network with multiple intersections.	DRL has the ability to approximate the Q -values for a larger number of state-action pairs for solving complex and large problems.
Dynamic Programming [28]	Breaks a complex problem into a collection of simpler sub-problems. Subsequently, the sub-problems are solved separately. The solutions are stored in a memory-based data structure.	Longer time is needed to compute multiple solutions (e.g., change traffic phase and adjust traffic phase split) for multiple sub-problems as a result of multiple simultaneous changes (e.g., queue length and waiting time) to a traffic network.	DRL uses experience replay and target network to reduce the time required to train the main network while improving the stability of training. This helps DRL to provide multiple solutions (e.g., change of traffic phase and adjustment of traffic phase split) for addressing multiple simultaneous changes (e.g., queue length and waiting time) to a traffic network with reduced training time.
RL [12]	Explores and exploits different state-action pairs so that the highest possible rewards are achieved and accumulated over time for system performance enhancement.	Large number of states (or a large state space) is needed to provide abstract representations of high-dimensional and complex inputs as a result of the curse of dimensionality.	DRL has the ability to deal with high dimensional state space, and so it addresses the curse of dimensionality.

where $0 \leq \alpha \leq 1$ is the learning rate, and $\delta_t(s_t, a_t)$ is the temporal difference, which is based on the Bellman equation that represents the difference between immediate and discounted rewards for two successive estimations as follows [51]:

$$\delta_t(s_t, a_t) = r_{t+1}(s_{t+1}) + \gamma \max_{a \in A} Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \quad (3)$$

where $\gamma \max_{a \in A} Q_t(s_{t+1}, a)$ represents the discounted reward, which is the expected maximum Q -value at time $t + 1$ and so on, and $0 \leq \gamma \leq 1$ represents a discount factor that shows the preference for the discounted reward. In other words, the immediate reward $r_{t+1}(s_{t+1})$ represents a short-term reward, while the discounted reward $\gamma \max_{a \in A} Q_t(s_{t+1}, a)$ represents a long-term reward. As time goes by $t = 1, 2, 3, \dots$, the agent explores, updates, and stores the Q -values $Q_t(s_{t+1}, a)$ of all the state-action pairs (s_t, a_t) in a two-dimensional Q -table.

During action selection, an agent selects either *exploration* or *exploitation*. Exploration selects a random action with a small probability ε to update its Q -value so that better actions can be identified in a dynamic and stochastic operating environment as time progresses. On the other hand, exploitation selects the best-known (or greedy) action with probability $1 - \varepsilon$ to maximize the state value using the *value function* as follows [52]:

$$v_t^\pi(s_t) = \max_{a \in A} Q_t(s_t, a) \quad (4)$$

where π is the policy, which is applied by the agent to decide the next action a_{t+1} based on the current state s_t , and it is defined as follows [53]:

$$\pi(s_t) = \arg \max_{a \in A} Q_t(s_t, a) \quad (5)$$

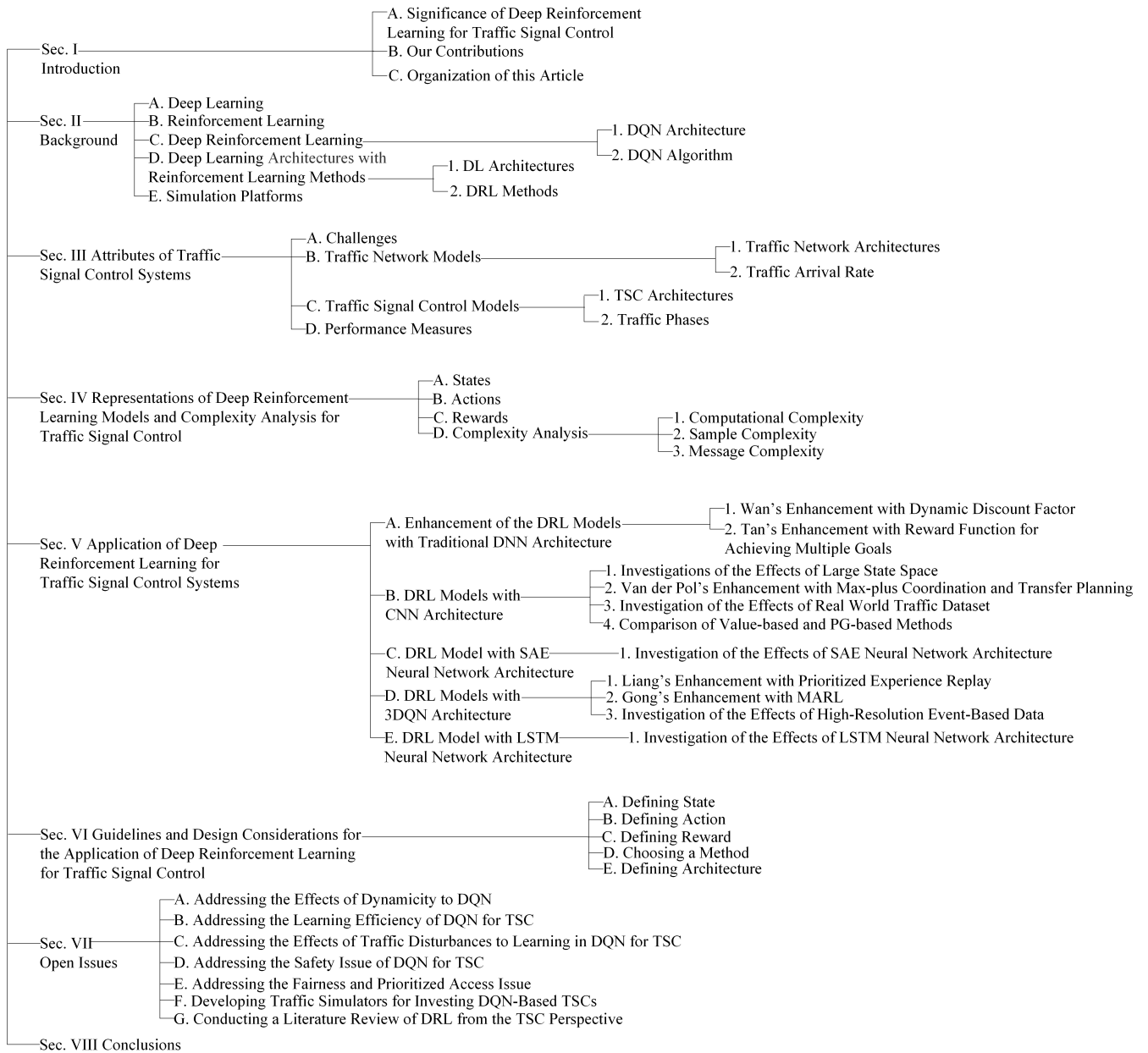


FIGURE 2. Organization of this article.

Hence, an agent selects an action with the maximum Q -value. For simplicity, only exploitation is shown in algorithms presented in this article.

C. DEEP REINFORCEMENT LEARNING

Deep reinforcement learning (DRL) is the combination of two artificial intelligence approaches (i.e., DL and RL). *Deep Q-network* (DQN) is the first DRL method proposed by DeepMind [14], and it has been widely used in TSC. DQN has two main features, namely *experience replay* and *target network* [54]–[57]. Using experience replay, an agent stores an experience in a replay memory, and subsequently trains itself using experiences randomly selected from the replay

memory [58]. Using target network, an agent utilizes a duplicate of the main network, and uses its weights to calculate target Q -values subsequently used to calculate a loss function minimized using gradient descent [59]. The weights of the target networks are fixed (or updated after a certain number of iterations) to improve training stability. During training, the target Q -value is used to compute the loss of a selected action in order to stabilize training, and it is updated every certain number of iterations [60]–[63]. The main network enables an agent to select an action after observing its state from the environment, and subsequently updates its main Q -values. The rest of this section presents the DQN architecture and algorithm, respectively.

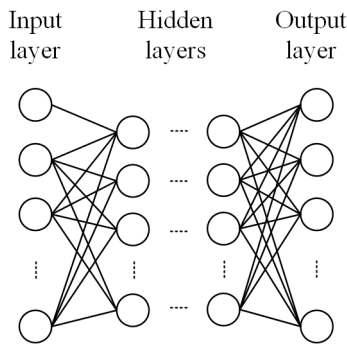


FIGURE 3. FCLN architecture.

Algorithm 1 RL Algorithm Embedded in an Agent

- 1: **Procedure**
- 2: observe current state $s_t \in \mathcal{S}$
- 3: **for** $t = 1 : T$ **do**
- 4: select action $a_t \in A$ using Equation (5)
- 5: perform action $a_t \in A$
- 6: receive reward $r_{t+1}(s_{t+1})$ and next state s_{t+1}
- 7: update Q -value $Q_{t+1}(s_t, a_t)$ using Equation (2)
- 8: **end for**
- 9: **End Procedure**

1) DQN ARCHITECTURE

DQN possesses one of the different kinds of DL architectures, such as FCLN, CNN, SAE, 3DQN, and LSTM. FCLN has been widely used with DQN. Fig. 4 presents the architecture of DQN. An agent has three main components, namely the replay memory, the main network, and the target network. The replay memory is a dataset of an agent’s experiences $D_t = (e_1, e_2, \dots, e_t, \dots)$, which are gathered when the agent interact with the environment as time goes by $t = 1, 2, 3, \dots$. Subsequently, the experiences D_t are used during the training process. The main network consists of a FCLN, and the weight θ_k of the FCLN is used to approximate its Q -values $Q(s, a; \theta_k)$ at iteration k . The main network is used to select an action a_t for a particular state s_t observed from the environment in order to achieve the best possible reward $r_{t+1}(s_{t+1})$ and next state s_{t+1} at the next time instant $t + 1$. The target network is a duplicate of the main network, and the weight θ^- of the FCLN is used to approximate its Q -values $Q(s, a; \theta_k^-)$ after the k th iteration. There are two main differences between the Q -values of the main and target networks. Firstly, the main network is used during action selection and training, while the target network is used during training only. The target network improves the training stability, without which the policy may oscillate between the main and target Q -values in a single network. Secondly, the Q -values $Q(s, a; \theta_k)$ of the main network are updated in every iteration k , while the weights θ_j^- of the target network are updated by copying the weights θ_j of the main network at every C steps, which is equivalent to k iterations.

2) DQN ALGORITHM

Algorithm 2 shows the algorithm for DQN. At episode $m \in M$, an agent observes the current state $s_m \in \mathcal{S}$. At time instant $t \in T$, the agent selects an action $a_t \in A$ using Equation (5), which is given by the Q -value of the main network; subsequently it receives the reward $r_{t+1}(s_{t+1})$ and observes the next state s_{t+1} , and stores its experience $e_t = (s_t, a_t, r_t, s_{t+1}, a_{t+1})$ in the replay memory $D_t = (e_1, e_2, \dots, e_t, \dots)$. Subsequently, the agent samples a mini-batch of experiences from the replay memory D_t in a random manner to learn the weights θ_j . At iteration $j \in J$, the agent updates the target Q -values of the target network, specifically $Q_j(s_j, a_j^*; \theta_j^-) \approx Q^*(s_j, a_j)$. The weights θ_j^- of the target network is replaced with the weights θ_j of the main network in order to provide updated Q -values $Q(s, a; \theta_k^-)$ of the target network as time goes by. The weights θ_j^- of the target network is fixed to minimize the loss between the Q -values of the main and target networks, which helps to stabilize Q -values. The loss function at iteration j is minimized to train the main network as follows [64]:

$$L_j(\theta_j) = \mathbb{E}_{s_j, a_j \sim p(\cdot)} \left[\left(y_j - Q_j(s_j, a_j; \theta_j) \right)^2 \right] \quad (7)$$

where $p(s, a)$ represents the probability distribution of a state-action pair (s, a) , and y_j represents the target given by θ_{j-1}^- in the previous iteration $j - 1$. The gradient of the loss function $\nabla_{\theta} L_j(\theta_j)$ is given as follows [65]:

$$\nabla_{\theta} L_j(\theta_j) = \mathbb{E}_{s_j, a_j \sim p(\cdot)} \left[\left(y_j - Q_j(s_j, a_j; \theta_j) \right) \nabla_{\theta_j} Q_j(s_j, a_j; \theta_j) \right] \quad (8)$$

During backpropagation, a backward pass uses gradient descent, whereby the weights θ_j of the main network are updated in the opposite direction, to achieve the minimum value of $\nabla_{\theta} L_j(\theta_j)$.

D. DEEP LEARNING ARCHITECTURES WITH REINFORCEMENT LEARNING METHODS

This section presents DL architectures used with RL methods applied to different types of traffic network models, including single intersection [66]–[72], [99], multi intersections [74], real world [77]–[79], and grid [76]. Fig. 12 presents the DRL attributes for TSC.

1) DL ARCHITECTURES

The DL architectures used with RL methods for TSCs are as follows:

- N.1 The traditional *FCLN* has been adopted in [70], [72] to approximate the Q -values of TSCs (see Section II-A).
- N.2 *Convolutional neural network (CNN)* has been widely adopted in [66], [69], [74], [76], [78] to approximate the Q -values of TSCs. While the traditional DL approach consists of fully connected (FC) layers, CNN has two main types of layers, namely the *convolutional layer*, and as well as the traditional *FC layer* (see Fig. 5 for

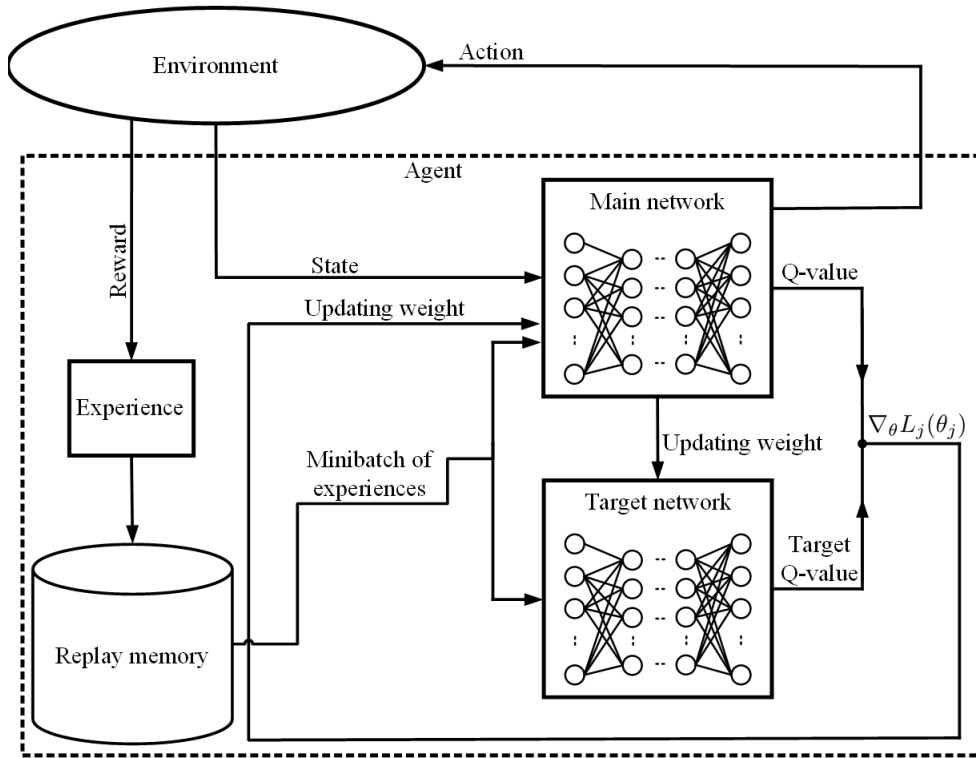


FIGURE 4. DQN architecture.

Algorithm 2 Traditional DQN Algorithm

- 1: **Procedure**
- 2: **for** episode = 1 : M **do**
- 3: { Observation process }
- 4: observe current state $s_t \in \mathcal{S}$
- 5: **for** $t = 1 : T$ **do**
- 6: { Action selection process }
- 7: select action $a_t \in A$ using Equation (5)
- 8: receive reward $r_{t+1}(s_{t+1})$ and next state s_{t+1}
- 9: store experience $(s_t, a_t, r_{t+1}(s_{t+1}), s_{t+1})$ in replay memory D_t
- 10: { Training process }
- 11: sample a random minibatch of experiences $(s_t, a_t, r_{t+1}(s_{t+1}), s_{t+1})$ from replay memory D_t
- 12: **for** $j = 1 : N$ **do**
- 13: set target

$$y_j = \begin{cases} r_{j+1}(s_{j+1}), & \text{if episodes terminate at } s_{j+1} \\ r_{j+1}(s_{j+1}) + \gamma \max_a Q(s_{j+1}, a; \theta_j), & \text{otherwise} \end{cases} \quad (6)$$

- 14: perform a gradient descent optimization on $(y_j - Q(s_j, a_j; \theta_j))^2$ with respect to θ_j using Equation (8)
- 15: reset $\theta^- = \theta$ in every C steps
- 16: **end for**
- 17: **end for**
- 18: **end for**
- 19: **End Procedure**

an example of a CNN architecture [69]). In Fig. 5, CNN has one input layer, two convolutional layers, two FC layers, and one output layer. Each convolutional layer

consists of three parts, namely *convolution*, *pooling*, and *activation*. The data flows from the input layer to the output layer. The layers are as follows:

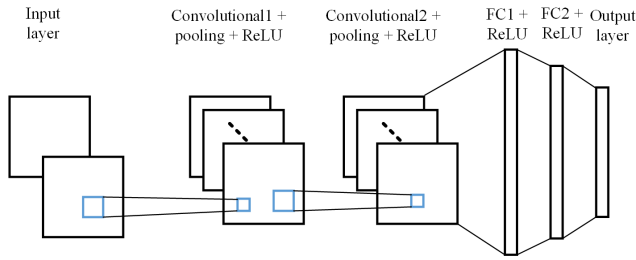


FIGURE 5. An example of a CNN architecture [40].

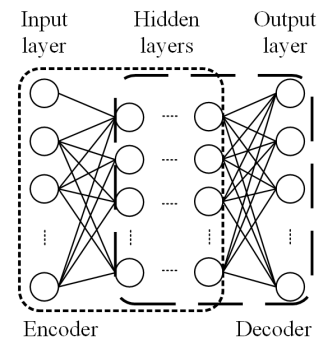


FIGURE 6. An example of a SAE architecture [38]. The encoder is enclosed with a dotted line, and the decoder is enclosed with a dashed line.

- The input layer represents the state s_t^i , such as the position (S.5) and speed (S.6) of a vehicle. For instance, at an intersection, with a grid size of 60×60 , the input state s_t^i represents the position and speed of a vehicle, and it has a size of $60 \times 60 \times 2$.
- Two convolutional layers consist of k filters (or kernels), in which each filter consists of a set of weights. Each weight aggregates local patches (e.g., the pixels of an image) from the previous layer and shifts the aggregated local patches for a fixed number of steps defined by the stride each time. By pooling, the salient values from the local patches replace the whole patch in order to remove the less important information and reduce the dimensionality of the input state s_t^i . Next, the activation function (i.e., ReLU) activates the units of patches.
- Two FC layers.
- The output layer provides the Q -values $Q_t^i(s_t^i, a_t^i)$ of all possible actions a_t^i .

N.3 *Stacked auto encoder (SAE) neural network*, which performs encoding and decoding functions, has been adopted in [67] to approximate the Q -values of TSCs. Fig. 6 shows an example of a SAE architecture. The data flows from the input layer to the output layer. The layers are as follows:

- The input layer represents the state s_t^i .
- The encoder maps the input data into hidden representations (i.e., feature extraction). The encoding process is given by:

$$E(x) = f(W_1x + b) \quad (9)$$

where $f(\cdot)$ is the encoding function, W_1 is a weight matrix used to reduce the number of parameters to learn, and b is the bias vector, which stores the value of 1 in order to produce an output for the next layer that differs from 0 whenever the a feature value is 0.

- The decoder reconstructs the input data from the hidden representations. The decoding process is given by:

$$D(x) = g(W_2E(x) + b) \quad (10)$$

where $g(\cdot)$ is the decoding function, and W_2 is a transpose matrix of the weight matrix W_1 . Subsequently,

the reconstruction error, which is a measure of the discrepancy between input and its reconstruction by decoding, of the obtained parameters θ_{ae} is minimized as follows:

$$\theta_{ae} = \arg \min_{\theta_{ae}} L(x, D) = \arg \min_{\theta_{ae}} \frac{1}{2} \sum ||x - D(x)||^2 \quad (11)$$

- The output layer provides the Q -values $Q_t^i(s_t^i, a_t^i)$ of all possible actions $a_t^i \in A^i$.
- N.4 *Double dueling deep Q-network (3DQN)*, with its FC layer split into two separate streams, has been adopted in [71], [73], [77] to approximate the Q -values of TSCs. The 3DQN architecture consists of double Q -learning [85] and a dueling network [86] as shown in an example of a 3DQN architecture in Fig. 7. In double Q -learning, a max operator decouples the selection of an action from the evaluation of an action; while in traditional Q -learning, a max operator uses the same value for both selection and evaluation of an action. The dueling network has two separate streams to estimate the state value and the advantage of each action separately. The data flows from the input layer to the output layer. The layers are as follows:

- The input layer represents the state s_t^i .
- Three convolutional layers consist of k filters (or kernels), in which each filter consists of a set of weights.
- Two FC layers, in which the second FC layer is split into two separate streams: a) the state value $V(s_t^i)$ provides an estimate of the value function that measures the absolute value of a state s_t^i ; and b) the advantage $A(s_t^i, a_t^i)$ of performing an action a_t^i under a state s_t^i that represents the contribution of the action to the value function compared to all possible actions. The 3DQN architecture uses a FC layer that splits into two separate streams, while the CNN architecture uses the traditional FC layers.

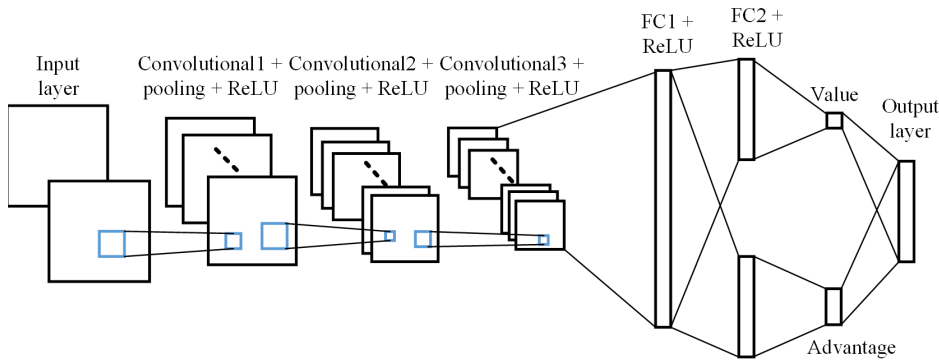


FIGURE 7. An example of a 3DQN architecture [42].

- The output layer provides the Q -values $Q_t^i(s_t^i, a_t^i)$ of all possible actions $a_t^i \in A$ as follows:

$$Q_t^i(s_t^i, a_t^i) = V(s_t^i) + \left(A(s_t^i, a_t^i) - \frac{1}{|A|} \sum_{a_{t+1}^i} A(s_t^i, a_{t+1}^i) \right) \quad (12)$$

where a positive value of $A(s_t^i, a_t^i)$ indicates that the action a_t^i has a better reward (or performance) compared to the average performance of all possible actions, and vice-versa.

The 3DQN architecture uses the double DQN (DDQN) algorithm. While the traditional DQN algorithm uses the same max operator and values for both selection and evaluation of an action, the DDQN algorithm uses the target as follows:

$$y_j^{DDQN} = r_{j+1}(s_{j+1}) + \gamma Q(s_{j+1}, \arg \max_a Q(s_{j+1}, a; \theta_j); \theta_j^-) \quad (13)$$

N.5 The traditional *long short-term memory (LSTM) neural network*, which is based on a recurrent neural network, consists of a memory cell; and it has been adopted in [79] to approximate the Q -values of TSCs. Fig. 8 shows an example of a LSTM architecture. The data flows from the input layer to the output layer. The layers are as follows:

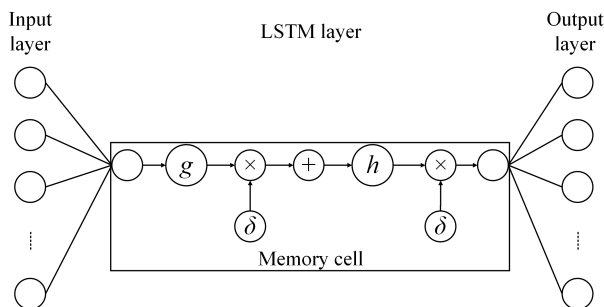


FIGURE 8. Traditional LSTM neural network architecture.

- The input layer represents the state s_t^i .
- The LSTM layer consists of a memory cell that maintains a time window of states [87]. In Fig. 8, the memory cell has an input gate g , and an output gate h . In addition, there are two nodes for multiplication \times , one node for summation $+$, and two gate activation functions δ that transform data into a value between 0 and 1.
- The output layer provides the Q -values $Q_t^i(s_t^i, a_t^i)$ of all possible actions $a_t^i \in A^i$.

2) DRL METHODS

There are three DRL methods applied to DL architectures for TSCs as follows:

- E.1 *Value-based method.* The value-based method is the traditional DQN method (see Sections II-B and II-C), which has been adopted in [66]–[74], [76]–[78], [99]. The value-based DQN maps each state-action pair (s_t, a_t) to a state value $V_t(s_t)$ learned using value function (see Equation (4)) in order to identify the best possible action for each state.
- E.2 *Policy-gradient (PG)-based method.* DQN with the PG-based method selects an action $a_t \in A$ based on a policy with probability distribution $\pi(a_t | s_t; \theta)$ given a state $s_t \in S$, where the probability distribution is learned by performing gradient descent on the policy parameter (i.e., the weight θ of DQN). Equation (8) is revised as follows:

$$\nabla_{\theta} L_j(\theta_j) = \mathbb{E}_{s_j, a_j \sim p(\cdot)} \left[\sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R_t \right] \quad (14)$$

where $R_t = \sum_t \gamma r_t$ represents the reward function. PG-based DQN has been adopted in [68].

- E.3 *Advantage actor critic (A2C)-based method.* DQN with the A2C-based method is a hybrid approach that combines both value-based and PG-based DQN methods. Each agent has an actor that controls how it behaves (i.e., PG-based), and a critic that measures the suitability of the selected action (i.e., value-based). Equation (8) is

revised as follows:

$$\nabla_{\theta} L_j(\theta_j) = \mathbb{E}_{s_t, a_t \sim p(\cdot)} \left[\sum_t \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) A_t(s_t, a_t) \right] \quad (15)$$

where $A_t(s_t, a_t) = Q_t(s_t, a_t) - V_t(s_t)$ is the advantage function. A2C-based DQN has been adopted in [79].

E. SIMULATION PLATFORMS

Traffic simulators are simulation platforms that evaluate, compare, and optimize DRL-based TSCs. In general, a traffic simulator provides a graphical user interface and essential features to simulate TSCs, vehicles, and roads in order to gather: a) local statistics, such as the queue length of the vehicles at an intersection; and b) global statistics, such as the queue length of the vehicles at all intersections in a traffic network. There are two approaches for investigation: a) the *macroscopic* approach that focuses on traffic flows, such as traffic density, the speed limit of the lanes, and the vehicle distributions [101]; and b) the *microscopic* approach that focuses on the mobility characteristics of an individual vehicle, such as the driving speed and direction [102]. In most investigations in the application of DRL to TSCs, traffic simulators are based on the microscopic approach [66], [67], [70], [77], including SUMO [103], [104], Paramics [105], [106], VISSIM [107], [108], and Aimsun Next [109], [110]. Some schemes have considered real world traffic network in simulations, including real world traffic network based on Florida designed in Aimsun Next [77], as well as Jinan [78] and Monaco [79] designed in SUMO.

III. ATTRIBUTES OF TRAFFIC SIGNAL CONTROL SYSTEMS

The attributes of the intersections, traffic, and TSCs have brought about challenges to traffic management. This section presents these attributes to provide a better understanding about the TSC problem, which is solved using DRL as presented in Section IV. Figure 9 presents various aspects of the TSC attributes. In addition, performance measures are presented.

A. CHALLENGES

DRL addresses the following two main challenges of TSC that causes congestion, green idling, and cross-blocking:

- C.1 *Inappropriate traffic phase sequence*: A traffic phase consists of a combination of green signals allocated to a set of lanes simultaneously for non-conflicting and safe traffic flows at an intersection. TSC has different kinds of traffic phases (see Section III-C2 for more details) characterized by: a) *with opposing through traffic* (T.2.1); b) *without opposing through traffic* (T.2.2); and c) *with group-based individual traffic* (T.2.3). These traffic phases can be activated in an in-order (i.e., round-robin) or out-of-order manner.
- C.2 *Inappropriate traffic phase split*: A traffic phase split represents the time interval allocated for a traffic phase.

For simplicity, we can focus on the green time of the traffic phase with green signals, in which the rest of the traffic phases receive red signals. Too long of a green time can cause cross-blocking and green idling when the traffic volume is high and low, respectively. Too short of a green time can increase the queue length of a lane, resulting in congestion. The maximum and minimum durations of a traffic phase split can be imposed. The maximum duration prevents a long waiting time for vehicles at other lanes, while the minimum duration ensures that at least a single waiting vehicle can cross an intersection.

B. TRAFFIC NETWORK MODELS

The traffic network models reflect the traffic conditions, which can be characterized by their architectures and traffic arrival rates.

1) TRAFFIC NETWORK ARCHITECTURES

A traffic network consists of a single or multiple intersections and edge nodes. Each intersection has multiple legs in different directions, and each leg has a single or multiple lanes so that a vehicle can either turn right, turn left, or go straight. A vehicle enters a traffic network through an edge node, traverses from one intersection to another, and leaves through another edge node in a closed traffic environment. A left-hand traffic network is considered throughout the article, even though a similar description can be applied to a right-hand traffic network.

- M.1.1 *Single intersection traffic network* represents a traffic network with a single intersection [66]–[72].
- M.1.2 *Multi intersection traffic network* represents a traffic network with multiple intersections, such as two intersections with a single closed link in between the intersections (see Fig. 10a) and three intersections with a central intersection and two out-bound intersections (see Fig. 10b) [74]. Each vehicle can cross one intersection (e.g., west), two intersections (e.g., west-central) or three intersections (e.g., west-central-east).
- M.1.3 *Real world traffic network* represents a traffic network based on the layout of a city, and so a larger number of intersections are considered. For instance, investigations are conducted based on 8 intersections in Florida (United States) [77], 24 intersections in Jinan (China) [78], and 30 intersections in Monaco [79].
- M.1.4 *Grid traffic network* represents a traffic network based on a grid topology, such as 2×2 (see Fig. 10c) and 3×3 traffic networks [76].

An intersection i has l legs. Each leg $l^i \in L^i$ has d lanes, whereby each lane is represented by $d^i \in D^i$. The number of lanes is ignored when a leg has a single lane $d = 1$.

A traffic network also consists of hardware devices (e.g., video-based traffic detectors, inductive loop detectors,

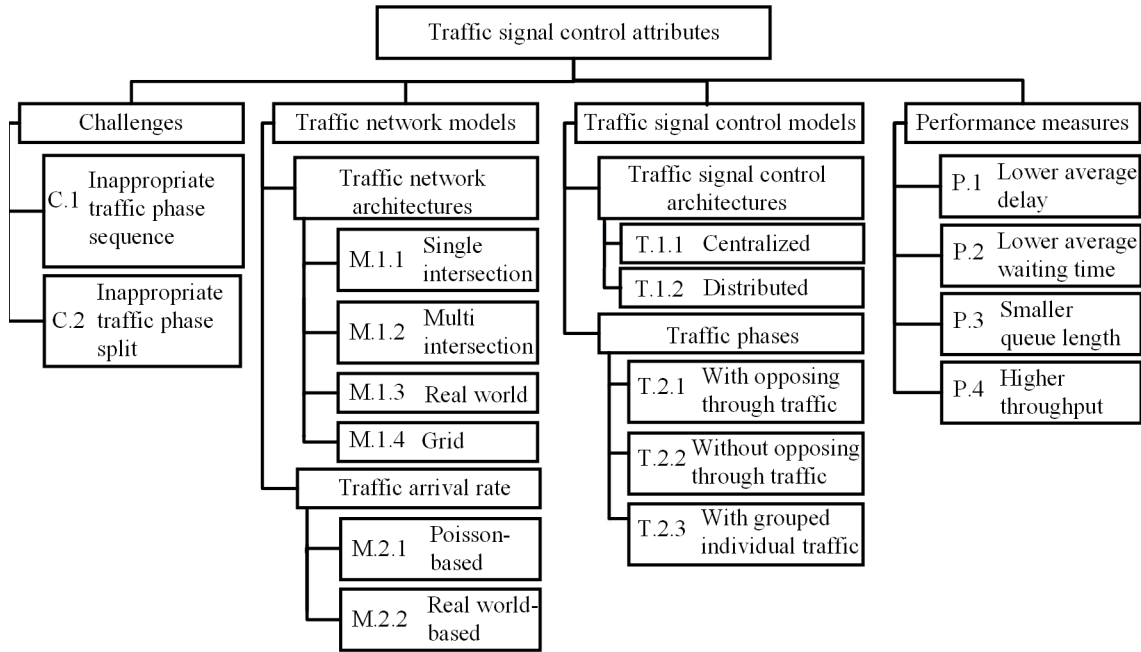


FIGURE 9. TSC attributes.

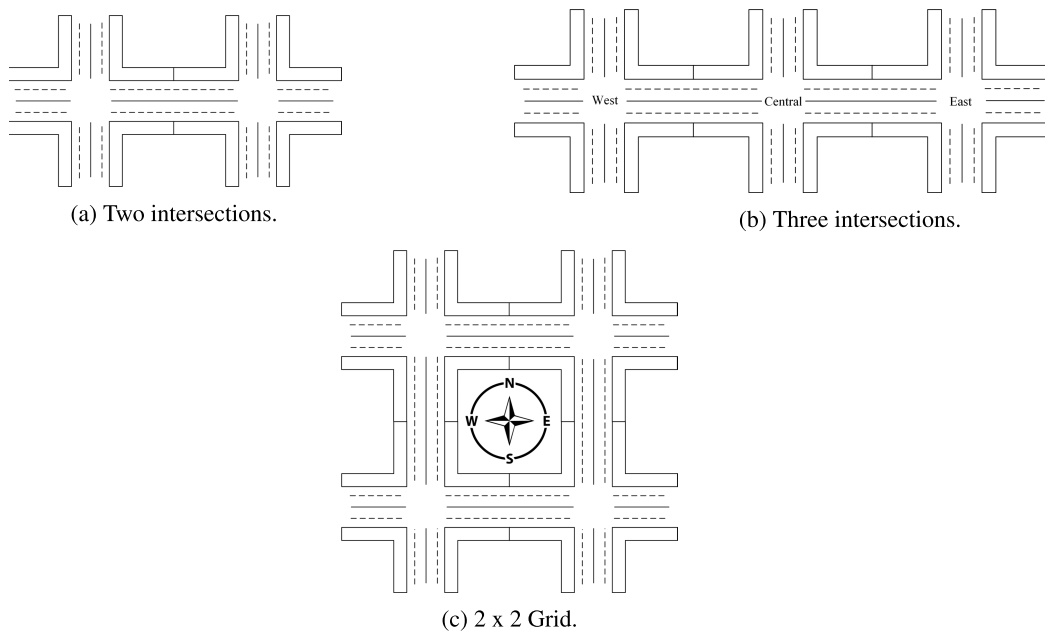


FIGURE 10. Traffic network architectures.

and camera sensors) installed at intersections to gather local statistics (e.g., traffic arrival and departure rates, the occupancy of a lane, as well as the queue length and waiting time of vehicles) over time. The hardware devices process (e.g., aggregate) and send the statistics to agents (or TSCs) in order to estimate longer-term information (e.g., the average queue length). Alternatively, the hardware devices can gather shorter-term information (e.g., the instantaneous queue length) at intersections at any time instant. The TSCs can communicate among themselves using wired

connections, and can communicate with vehicles using wireless communication.

2) TRAFFIC ARRIVAL RATE

Traffic arrival rate characterizes: a) the number of vehicles entering a traffic network through edge nodes; or b) arriving at an intersection within a time duration (e.g., 2,000 vehicles per hour) [67]. The traffic arrival rate affects the traffic volume leading to a crowded or sparse traffic network. When a vehicle arrives, it is placed at the end of a queue at one

of the lanes of the legs at an intersection. Each lane can accommodate a certain number of vehicles. Traffic models and statistical distributions can be used to characterize the steady-state dynamics of the traffic arrival rate as follows:

M.2.1 *Poisson-based traffic arrival rate* determines the probability of the number of vehicles n arriving at an intersection i within a time period t_p based on [80]:

$$P_{t_p}^{i,n} = \frac{(\mu^i t_p)^n}{n!} e^{-\mu^i t_p} \quad (16)$$

where μ is the arrival rate of vehicles. Using the Poisson process, there are three main properties that attribute to a realistic traffic model [69]: a) the inter-arrival time is exponentially distributed; b) the inter-arrival time is memoryless; and c) the number of incoming vehicles at different lanes are independent of each other.

M.2.2 *Real world-based traffic arrival rate* determines the probability of the number of vehicles arriving at an intersection within a time period based on the traversing properties of vehicles (e.g., lane switching, vehicle overtaking, driving direction, driving speed, and the physical position of the destination). Some real world-based traffic models are the car-following model [71] and the Nagel-Schreckenberg model [81].

C. TRAFFIC SIGNAL CONTROL MODELS

TSCs can be characterized by their architectures (e.g., TSCs and their relationship) and traffic phases. While the architecture characterizes the operation at the global level (or multiple intersections), the traffic phase characterizes operation at the local level (or a single intersection).

1) TSC ARCHITECTURES

In the context of DRL, the TSC architecture is as follows:

T.1.1 *Centralized model* enables a centralized agent to gather local statistics (e.g., the queue length (S.1) of the lanes) from all or neighboring agents, and selects an action (e.g., the type of traffic phase (A.1)), which optimizes the system-wide performance. Subsequently, the centralized agent either executes the action or sends the action or knowledge to distributed agents (e.g., all or neighboring agents). The distributed agents may either execute the action or use the knowledge to select their respective actions. A centralized model has three main issues with regard to efficiency, scalability, and robustness. *Firstly*, the centralized model has a single point of failure whereby the malfunctioning of the centralized agent can affect the traffic condition of the entire traffic network. *Secondly*, the centralized agent experiences the curse of dimensionality. *Thirdly*, the centralized agent incurs significant communication overhead for information exchange. This model has been widely adopted by traditional TSCs, including GLIDE [82], SCOOT [83], and SCAT [84].

T.1.2 *Distributed model* enables multiple distributed agents to gather local statistics and select their respective actions. Hence, a complex problem is segregated into sub-problems solved by the distributed agents, contributing to higher efficiency and robustness. In the context of DRL, the distributed model enables the agents to optimize a global Q -value in a traffic network in order to achieve the global objective of a traffic network. To provide a global view of the operating environment, the distributed agents observe their respective local operating environment and learn about their neighboring agents' information (e.g., rewards and Q -values). The agents select their respective actions, and the global Q -value converges to an optimal equilibrium in order to achieve an optimal joint action as time goes by.

2) TRAFFIC PHASES

The traffic phases can be characterized as follows:

T.2.1 *With opposing through traffic* is a traffic phase, which incorporates a four-phase traffic sequence, in which traffic travels through two opposing lanes simultaneously as shown in Fig. 11a. It is preferred at intersections where: a) either through or turning traffic volume is significantly higher; and b) the through and turning traffic use separate lanes [67].

T.2.2 *Without opposing through traffic* is a traffic phase, which incorporates a four-phase traffic sequence, in which traffic travels through two lanes without opposing each other simultaneously as shown in Fig. 11b. It is preferred at intersections where: a) the through and turning traffic volumes are equal, and b) the through and turning traffic share a single lane [70].

T.2.3 *With grouped individual traffic* is a traffic phase in which green signals are individually allocated to lanes for a particular time period as long as the selected combination of traffic movements are non-conflicting at an intersection [66].

D. PERFORMANCE MEASURES

There are four performance measures achieved by DRL models as follows:

P.1 *Lower average delay* reduces the average time required by vehicles to cross an intersection or to traverse from a source to a destination. The average time also includes the average waiting and travelling times during cross-blocking and congestion.

P.2 *Lower average waiting time* reduces the average waiting time of the vehicles (see (R.1) in Section IV-C).

P.3 *Smaller queue length* reduces the queue length of the vehicles (see (S.1) in Section IV-A and (R.2) in Section IV-C).

P.4 *Higher throughput* increases the number of vehicles crossing an intersection, or reaching their destinations, within a certain time period (e.g., a single cycle).

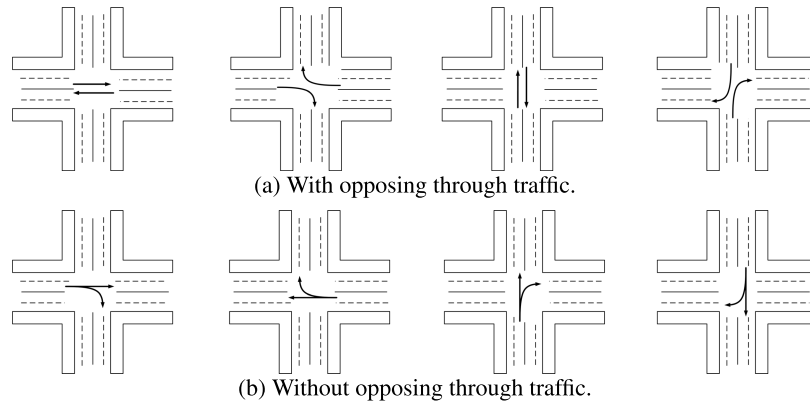


FIGURE 11. Traffic phases.

IV. REPRESENTATIONS OF DEEP REINFORCEMENT LEARNING MODELS AND COMPLEXITY ANALYSIS FOR TRAFFIC SIGNAL CONTROL

The traditional DRL approach for TSC has been widely used in the literature [66], [69] [74]. Extension to the traditional DRL approach with enhanced features has also been investigated as presented in Section II-D. The DRL agent can be embedded in TSC to coordinate vehicles [75], [76]. The rest of this section presents the attributes of DRL for TSC systems. Fig. 12 presents the DRL attributes for TSC. In addition, complexity analysis is presented.

A. STATES

The state $s_t^i \in S^i$ of an agent i represents its decision-making factors. Each state can consist of j sub-states $s_t^{i,j} = (s_t^{i,1}, s_t^{i,2}, s_t^{i,3}, \dots, s_t^{i,j})$, in which the sub-states have different representations at intersection i . In the context of DRL, there are six main representations for a state s_t^i :

- S.1 *Queue length* represents the number of waiting vehicles at a lane or a leg, and so it changes with the traffic arrival and departure rates. A waiting vehicle has a speed of 0 km/h. The state s_t^i can represent the maximum queue length among the lanes, in which the number of states is given by the maximum queue length. As an example, suppose there are three waiting vehicles at the leg l of intersection i , so the states are $s_t^{i,l} = 1$ for the three waiting vehicles and $s_t^{i,l} = 0$ for the moving vehicles [67].
- S.2 *Red timing* represents the time elapsed since the traffic signal of a lane turned into red at an intersection. The state is reset to a zero value $s_t^{i,l} = 0$ whenever green and yellow signals are activated, and $s_t^{i,l} = 1$ whenever a red signal is activated, at the lane d^{li} of a leg l^i at an intersection i [70]. The state $s_t^{i,li}$ can also represent the red timing $t_{r,t}^{i,li}$ of the lanes $d^{li} \in D^{li}$ of a leg l^i at an intersection i .
- S.3 *Green timing* represents the time elapsed since the traffic signal of a lane turned into green at an intersection. The state is reset to a zero value $s_t^{i,l} = 0$ whenever red

and yellow signals are activated, and $s_t^{i,l} = 1$ whenever green signal is activated, at the lane d^{li} of a leg l^i at an intersection i . The state $s_t^{i,li}$ can also represent the green timing $t_{g,t}^{i,li}$ of the lanes $d^{li} \in D^{li}$ of a leg l^i at an intersection i [70].

- S.4 *Current traffic phase* represents the traffic phase being activated at the time of decision making. At time t , the state s_t^i can represent the traffic phase at an intersection i , in which the number of substates is given by the number of candidate traffic phases. As an example, the state $s_t^i = (s_t^{i,1}, s_t^{i,2}, s_t^{i,3}, \dots, s_t^{i,8}) = (1, 0, 0, 0, 0, 0, 0, 0)$ represents that only traffic phase 1 is activated at time t [69].
- S.5 *Vehicle position* represents the physical position of a waiting vehicle at the lane d^{li} of a leg l^i at an intersection i . Consider a lane segmented into small cells from the intersection i , in which each cell can accommodate a single vehicle. The state $s_t^i = (s_t^{i,1}, s_t^{i,2}, s_t^{i,3}, \dots, s_t^{i,j})$ represents the position of a cell, with the cell $s_t^{i,1}$ being the nearest to the intersection i , and the cell $s_t^{i,j}$ being the maximum queue length [74].
- S.6 *Vehicle speed* represents the speed of a moving vehicle at the lane d^{li} of a leg l^i at an intersection i . Consider a lane segmented into small cells, in which each cell can measure the speed of a single vehicle. At time t , the state $s_t^i = (s_t^{i,1}, s_t^{i,2}, s_t^{i,3}, \dots, s_t^{i,j})$ represents the speed of a vehicle from the intersection i , whereby $s_t^{i,*} = \{0, 0.1, 0.2, \dots, 0.9, 1\}$, $s_t^i = 1$ represents the maximum legal speed of a vehicle (e.g., 90 km/h), and $s_t^i = 0$ represents the minimum speed (i.e., 0 km/h) [66].

B. ACTIONS

The action $a_t^i \in A^i$ of an agent i represents its selected action. In the context of DRL, there are two main representations for an action a_t^i :

- A.1 *Traffic phase type* represents the selection of a combination of green signals allocated simultaneously for non-conflicting traffic flows at an intersection. The traffic phases can be activated in one of

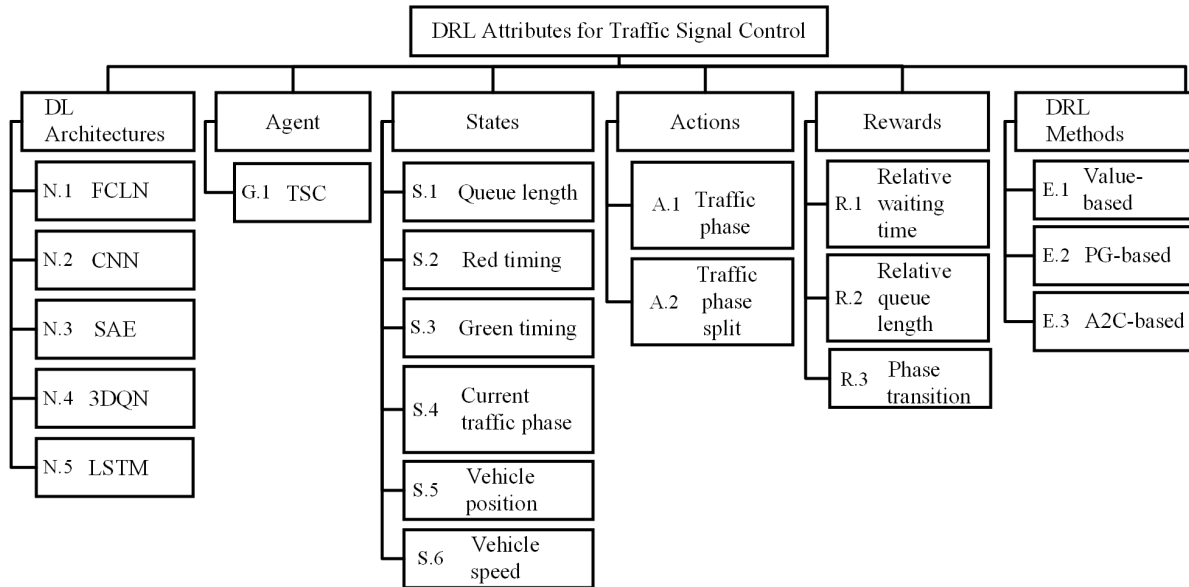


FIGURE 12. DRL attributes for TSC.

these manners: a) in-order (i.e., round-robin with certain periods of traffic phase splits); and b) out-of-order. At time t , an action $a_t^i = \{a_1^i, a_2^i, a_3^i, \dots, a_n^i\}$ at an intersection i represents one of the activated traffic phases. The number of candidate actions is equal to the number of traffic phases [66], [67].

A.2 *Traffic phase split* represents the selection of a time interval for a traffic phase at an intersection i . The action $a_t^i = \{a_1^i, a_2^i\}$ represents whether agent i keeps the current traffic phase (a_1^i), or switches to another traffic phase (a_2^i) which normally happens when the current traffic phase does not receive the best possible reward [71].

C. REWARDS

The reward $r_{t+1}^i(s_{t+1}^i) \in R^i$ of an agent i represents its feedback from the operating environment, where R^i is a set of potential rewards at agent i . The reward value can be fixed, such as $r_{t+1}^i(s_{t+1}^i) = 1$ that represents a reward and $r_{t+1}^i(s_{t+1}^i) = 0$ that represents a cost (or penalty). In the context of DRL, there are three main representations for a reward $r_{t+1}^i(s_{t+1}^i)$ as follows:

R.1 *Relative waiting time*. In this representation, an agent receives rewards (or costs) that change with the average waiting time of the vehicles at an intersection. The average waiting time of the vehicles at an intersection can increase due to cross-blocking, congestion, or red signal. The reward $r_{t+1}^i(s_{t+1}^i)$ is a relative value. As an example, $r_{t+1}^i(s_{t+1}^i) = W_t^i - W_{t+1}^i$ represents the difference of the average total waiting time of all vehicles at intersection i at time t and time $t + 1$ (or between traffic phases) [69], [71] [66], [68].

R.2 *Relative queue length*. In this representation, an agent receives rewards (or costs) that change with the

increment/decrement of the queue length of the vehicles at an intersection. The reward $r_{t+1}^i(s_{t+1}^i)$ is a relative value. As an example, $r_{t+1}^i(s_{t+1}^i) = n_{c,t+1}^i - n_{q,t+1}^i$ represents the difference between the number of vehicles crossing an intersection $n_{c,t+1}^i$ and the queue length $n_{q,t+1}^i$, and it indicates whether the green time is sufficient or not at an intersection i [79].

R.3 *Phase transition* represents the cost of a traffic phase transition, such as the time delay incurred during the transition of a traffic phase [74].

D. COMPLEXITY ANALYSIS

In this section, the computational, sample, and message complexities of DRL models for TSCs are estimated. The complexity analysis conducted in this section is inspired by similar investigation performed in [111], [112]. The complexity analysis has two levels: a) *agent-wise* that considers all the state-action pairs (s_t, a_t) of an agent, and b) *network-wide* that considers all agents in a network. Note that, we focus on exploitation actions while analyzing the DRL algorithm. The parameters for complexity analysis are shown in Table 3. The agent-wise and network-wide complexities of DRL for TSCs are presented in Table 4. In the table, it should be noted that: a) an agent-wise complexity is shown without $|I|$, and a network-wide complexity is shown with $|I|$; and b) DRL models with a single agent show agent-wise complexities, and DRL models with multiple agents show network-wide complexities. The three types of complexities of DRL models for TSC is presented in the rest of this section.

1) COMPUTATIONAL COMPLEXITY

Computational complexity estimates the number of times the DRL algorithm is being executed in order to calculate the Q -values for all actions of the agents, and it also refers to

TABLE 3. Parameters for complexity analysis.

Parameter	Description
$ S $	Number of sub-states
$ A $	Number of sub-actions for each sub-state.
$ R $	Number of rewards for each state-action pair (s_t, a_t) .
$ I $	Number of agents in a network.
$ J $	Number of neighboring agents of an agent in a network.

the complexity of action selection. For the agent-wise computational complexity, there are $|S|$ states, and $|A|$ actions, and so the complexity is $O(|S||A|)$ and the network-wide complexity is $O(|I||S||A|)$. As an example, in Gong's DRL model, where the traditional MARL algorithm (see Algorithm 3) is used, an agent i calculates the reward and updates its Q -value (Step 9), so the agent-wise complexity is $O(|S|) + O(|S||A|)$, which can be simplified as $O(|S||A|)$. Therefore, the network-wide complexity is $O(|I||S|) + O(|I||S||A|)$, which can be simplified as $O(|I||S||A|)$.

For estimating the agent-wise and network-wide computational complexities of the A2C-based method, which consists of actor and critic networks, the number of neurons in each layer as well as the number of layers in the network are considered. For the actor network, the number of neurons in the m th layer is U_m , and the number of layers in the actor network is M . So, the computational complexity of the m th layer is $O(U_{m-1}U_m + U_mU_{m+1})$, and the computational complexity of the actor network is $O(\sum_{m=2}^{M-1}(U_{m-1}U_m + U_mU_{m+1}))$. For the critic network, the number of neurons in the n th layer of the critic network is C_n , and the number of layers in the critic network is N . So, the computational complexity of the n th layer is $O(C_{n-1}C_n + C_nC_{n+1})$, and the computational complexity of critic network is $O(\sum_{n=2}^{N-1}(C_{n-1}C_n + C_nC_{n+1}))$. During execution, both actor and critic networks are used, so the agent-wise computational complexity of A2C-based method is $O(\sum_{m=2}^{M-1}(U_{m-1}U_m + U_mU_{m+1}) + \sum_{n=2}^{N-1}(C_{n-1}C_n + C_nC_{n+1}))$, and it is $O(|U_m, C_n|)$ for simplicity. So, the network-wide computational complexity is $O(|I||U_m, C_n|)$.

2) SAMPLE COMPLEXITY

Sample complexity estimates the number of experiences an agent takes to learn in order to behave well during and after training, and it also refers to the complexity of the training process. As an example, in the traditional DRL algorithm (see Algorithm 2), an agent gains an experience after interacting with the environment, which is stored in its replay memory (see Step 7 of Algorithm 2), so the agent-wise complexity is $O(|S||A||R|)$ given that there are $|R|$ rewards. Therefore, the network-wide complexity is $O(|I||S||A||R|)$.

3) MESSAGE COMPLEXITY

Message complexity is the number of messages exchanged among the agents in order to update a Q -value. As an

example, in Gong's MARL algorithm for DRL (see Algorithm 3), each agent i exchanges its traffic network condition (i.e., queue length) with its neighboring agents J (see Step 4 and 5 of Algorithm 3), so the agent-wise complexity is $\leq |J|$, and the network-wide complexity is $\leq |I||J|$.

V. APPLICATION OF DEEP REINFORCEMENT LEARNING FOR TRAFFIC SIGNAL CONTROL SYSTEMS

This section presents the limited application of the traditional and enhanced DRL models to TSCs. There are five main DL architectures, namely the traditional FCLN (N.1), CNN (N.2), SAE (N.3), 3DQN (N.4), and LSTM (N.5), which are applied to TSC in the literature. The DL architectures are essential to cater for the high-dimensional state space in order to address the curse of dimensionality in TSCs. Hence, this section is presented from the DRL perspective, rather than TSCs, and so the categorization is based on the DL architectures. Nevertheless, the TSC attributes are captured by the state representations, such as queue length (S.1), red timing (S.2), green timing (S.3), current traffic phase (S.4), vehicle position (S.5), and vehicle speed (S.6). A summary of the various DRL models and their descriptions applied to TSCs is presented in Table 5. Each DRL model has its strength. For instance, while 3DQN has been widely used to increase the learning speed, CNN has been widely used to analyze visual imagery. Table 7 presents a summary of the DRL attributes of the DRL-based TSCs proposed in the literature. In the literature, all DRL models are embedded in TSCs, and so the agent is TSC (G.1). A summary of the TSC attributes applied in the investigations of DRL-based TSCs is presented in Table 6. Table 8 presents a summary of key contributions, quantitative results/findings, and future directions of DRL-based TSC investigations. Subsequently, Section VI makes use of Tables 5-8 to provide guidelines and design considerations for identifying DRL solutions for different TSC problems. Table 9 presents a summary of the performance measures and simulation platforms applied to the DRL-based TSC investigations.

A. ENHANCEMENT OF THE DRL MODELS WITH TRADITIONAL FCLN ARCHITECTURE

The enhancement of various DRL models based on the traditional FCLN architecture and the value-based approach for TSCs are presented.

1) WAN'S ENHANCEMENT WITH DYNAMIC DISCOUNT FACTOR

Wan *et al.* [70] incorporate a dynamic discount factor, which is an enhancement to the discount factor γ in Equation (6), to the traditional FCLN architecture (N.1) and the value-based approach (E.1). The combination of the traditional FCLN architecture and the value-based approach allows this approach to use FC layers (see Fig. 3) to provide efficient storage while mapping each state-action pair to a state value (see Table 5 for more details). The DRL model optimizes the Q -values to address the challenge of inappropriate traffic

TABLE 4. Agent-wise and network-wide complexities of DRL for TSCs.

DRL Model	Complexities		
	Computational	Sample	Message
Traditional FCLN with value-based method	$O(S A)$	$O(S A R)$	
CNN with value-based method	$O(I S A)$	$O(I S A R)$	$\leq I J $
CNN with PG-based method	$O(S A)$	$O(S A R)$	
SAE with value-based method	$O(S A)$	$O(S A R)$	
3DQN with value-based method	$O(I S A)$	$O(I S A R)$	$\leq I J $
LSTM with A2C-based method	$O(I U_m, C_n)$	$O(I S A R)$	$\leq I J $

phase sequence (C.1) using a centralized model (T.1.1) in a single intersection traffic network (M.1.1) with (T.2.1) and without opposing through traffic (T.2.2). The traffic is characterized by Poisson process (M.2.1). This model is embedded in the TSC of the intersection (G.1). The state s_t represents the queue length (S.1), the red (S.2) and green (S.3) timings, and the current traffic phase (S.4). The action a_t represents the type of traffic phase to be activated in the next time instant (A.1). The reward $r_{t+1}(s_{t+1})$ represents the relative waiting time (R.1) of the vehicles. In the proposed scheme, the dynamic discount factor takes account of the time delay between action selection and action execution. When the next action (i.e., a traffic phase) is selected, it may not be executed immediately since a traffic phase can only change every pre-defined time period (i.e., five seconds). Hence, the discount factor reduces when the time delay increases so that the expected Q -value varies accurately. Equation (6) is revised as follows:

$$y_j = \begin{cases} r_{j+1}(s_{j+1}), & \text{if an episode terminates} \\ & \text{with } s_{j+1} \\ r_{j+1}(s_{j+1}) \\ + \Gamma \max_a Q(s_{j+1}, a; \theta_j), & \text{otherwise} \end{cases} \quad (17)$$

where $\Gamma = 1 - \tau(1 - \gamma)$ represents the dynamic discount factor, and τ represents the time interval between two consecutive actions. Higher τ represents a longer time delay between action selection and action execution, and vice-versa.

The proposed scheme has been shown to increase the throughput (P.4) and reduce the average delay (P.1) of the vehicles.

2) TAN'S ENHANCEMENT WITH REWARD FUNCTION FOR ACHIEVING MULTIPLE GOALS

Tan *et al.* [72] incorporate a novel reward function, which is an enhancement to the reward function $r_{t+1}^i(s_{t+1}^i) = W_t^i - W_{t+1}^i$ (R.1), to the traditional FCLN architecture (N.1) and the value-based approach (E.1). The combination of the traditional FCLN architecture and the value-based approach allows this approach to use FC layers (see Fig. 3) to provide efficient storage while mapping each state-action pair to a state value (see Table 5 for more details). The DRL model optimizes the Q -values to address the challenge of inappropriate traffic phase sequence (C.1) using a centralized model

(T.1.1) in a single intersection traffic network (M.1.1) with opposing through traffic (T.2.1). The traffic is characterized by Poisson process (M.2.1). This model is embedded in the TSC of the intersection (G.1). The state s_t represents the queue length (S.1) of the vehicles. The action a_t represents the type of traffic phase to be activated in the next time instant (A.1). In the proposed scheme, a novel reward function is defined to achieve multiple goals as follows:

$$r_{t+1}(s_{t+1}) = (n_{q,t}^i - n_{q,t+1}^i) + (n_{c,t}^i - n_{c,t+1}^i) + (W_t^i - W_{t+1}^i) \quad (18)$$

where, with reference to an intersection i at time t and $t + 1$, the $n_{q,t}^i - n_{q,t+1}^i$ represents the difference in the total number of waiting vehicles, $n_{c,t}^i - n_{c,t+1}^i$ represents the difference in the number of crossing vehicles, and $W_t^i - W_{t+1}^i$ represents the difference in the total waiting time of all vehicles.

The proposed scheme has been shown to reduce the queue length (P.3) of the vehicles.

B. DRL MODELS WITH CNN ARCHITECTURE

The application of various DRL models with the traditional CNN architecture, as well as value-based and PG-based approaches, for TSCs are presented.

1) INVESTIGATIONS OF THE EFFECTS OF LARGE STATE SPACE

Genders *et al.* [66] investigate the use of a large state space to incorporate more information about the traffic. This is because some popular state representations, such as queue length (S.1) [72], [79], ignore the current traffic phase and moving vehicles, including the position (S.5) and speed (S.6) of vehicles. The DRL model is based on the CNN architecture (N.2) and the value-based approach (E.1). The combination of the CNN architecture and the value-based approach allows this approach to the convolutional architecture (see Fig. 5) to analyze visual imagery while mapping each state-action pair to a state value (see Table 5 for more details). This model optimizes the Q -values to address the challenge of inappropriate traffic phase sequence (C.1) using a centralized model (T.1.1) in a single intersection traffic network (M.1.1) with grouped individual traffic (T.2.3). The traffic is characterized by Poisson process (M.2.1). This model is embedded in the TSC of the intersection (G.1). The state s_t represents the

TABLE 5. Summary of DRL models to address the challenges of TSC.

DRL Model	Description	Strength
Traditional FCLN with value-based method	Enables an agent to map each state-action pair to a value in order to identify the best possible action for each state.	Provides storage efficiently.
CNN with value-based method	Enables an agent with convolutional layers to map each state-action pair to a value in order to identify the best possible action for each state.	Analyzes visual imagery efficiently.
CNN with PG-based method	Enables an agent with convolutional layers to select an action following a policy based on probability distribution learned by gradient descent on the policy parameters.	Analyzes visual imagery efficiently.
SAE with value-based method	Enables an agent to perform encoding and decoding functions, while mapping each state-action pair to a value, in order to identify the best possible action for each state.	Compresses data efficiently.
3DQN with value-based method	Enables an agent to split the FC layer into two separate streams, while mapping each state-action pair to a value, in order to identify the best possible action for each state.	Increases learning speed.
LSTM with A2C-based method	Enables an agent to use an <i>actor</i> to control its behavior and a <i>critic</i> to measure the suitability of the selected action.	Provides memory to memorize previous inputs efficiently.

current traffic phase (S.4), the vehicle position (S.5), and the vehicle speed (S.6), and they are fed to the input layer of the CNN architecture. The action a_t represents the type of traffic phase to be activated in the next time instant (A.1). The reward $r_{t+1}(s_{t+1})$ represents the relative waiting time (R.1) of the vehicles. In this model, the traditional DQN algorithm (see Algorithm 2) is used, which is based on the value-based method. This value-based method identifies the best possible action (i.e., A.1) for the states (i.e., S.4, S.5, and S.6). The use of a large state space allows agents to incorporate more relevant information about the traffic, and it has shown to increase the computational and storage complexities, and reduce the learning rate. Nevertheless, the proposed scheme has shown to increase throughput (P.4) and reduces the average delay (P.1) and queue length (P.3) of the vehicles.

Similar model and approach has been adopted by Gao et al. [69]. There are two main differences. *Firstly*, the action a_t represents the choice to either keep the current traffic phase or switch to the next traffic phase in a predetermined sequence of traffic phases at the next time instant (A.2), which helps to address the challenge of inappropriate traffic phase split (C.2). *Secondly*, it uses the centralized model (T.1.1) with (T.2.1) and without opposing through traffic (T.2.2). The proposed scheme has shown to reduce the average delay (P.1) and waiting time (P.2).

2) VAN DER POL'S ENHANCEMENT WITH MAX-PLUS COORDINATION AND TRANSFER PLANNING

Van der Pol et al. [74], [76] incorporate max-plus coordination [88] and transfer planning [89] into the traditional DQN

algorithm (see Algorithm 2) in order to enable coordination among multiple agents. The DRL model is based on the CNN architecture (N.2) and the value-based approach (E.1). The combination of the CNN architecture and the value-based approach allows this approach to use the convolutional architecture (see Fig. 5) to analyze visual imagery while mapping each state-action pair to a state value (see Table 5 for more details). This model optimizes the Q -values to address the challenge of inappropriate traffic phase sequence (C.1) using a distributed model (T.1.2) in a multi intersection traffic network (M.1.2) and a grid traffic network (M.1.4) with opposing through traffic (T.2.1). The traffic is characterized by a real world traffic model (M.2.2), specifically the Krauß car-following model [90]. This model is embedded in each intersection (G.1), where the state s_t represents the current traffic phase (S.4), the vehicle position (S.5), and the vehicle speed (S.6). The action a_t represents the type of traffic phase to be activated in the next time instant (A.1). The reward $r_{t+1}(s_{t+1})$ represents the relative waiting time (R.1) of the vehicles, and the phase transition (R.3) of the traffic phases.

The max-plus coordination algorithm, which serves as the enhancement for multi-agent reinforcement learning (MRL) [91]–[95], enables an agent to learn about its neighboring agents' information, such as locally optimized payoff values (e.g., reward achieved by an individual agent). The proposed scheme maximizes a global Q -function, which is the linear combination of the local Q -values, as follows:

$$Q_{G_t}^i(s_t^i, a_t^i) = \sum_n Q_{n_t \in N_t}^i(s_{n_t}^i, a_{n_t}^i) \quad (19)$$

TABLE 6. Summary of TSC attributes applied in DRL-based TSCs.

DL Architectures	Reference	Challenges	Traffic Network Architectures				Traffic Characteristics		TSC Architectures	Traffic Phases		
		C.1 Inappropriate traffic phase sequence C.2 Inappropriate traffic phase split	M.1.1 Single intersection M.1.2 Multi intersection	M.1.3 Real world M.1.4 Grid	M.2.1 Poisson-based M.2.2 Real world-based	T.1.1 Centralized model T.1.2 Distributed model	T.2.1 With opposing through traffic T.2.2 Without opposing through traffic T.2.3 With grouped individual traffic					
N.1 FCLN	Wan et al. [70], 2018	✓	✓		✓	✓	✓	✓	✓		✓	
	Tan et al. [72], 2019	✓	✓			✓	✓	✓	✓		✓	
N.2 CNN	Genders et al. [66], 2016	✓	✓			✓	✓	✓				✓
	Mousavi et al. [68], 2017	✓	✓			✓	✓	✓	✓	✓		
	Gao et al. [69], 2017		✓			✓	✓	✓	✓	✓	✓	
	Van et al. [74], [76], 2016	✓		✓	✓		✓	✓	✓	✓	✓	
	Wei et al. [78], 2018	✓	✓		✓		✓	✓	✓	✓	✓	
N.3 SAE	Li et al. [67], 2016		✓	✓		✓	✓	✓	✓	✓	✓	
N.4 3DQN	Liang et al. [71], 2018		✓	✓		✓	✓	✓	✓	✓	✓	✓
	Wang et al. [73], 2019	✓	✓			✓	✓	✓	✓	✓	✓	✓
	Gong et al. [77], 2019	✓		✓	✓		✓	✓	✓	✓	✓	✓
N.5 LSTM	Chu et al. [79], 2019	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓

where N corresponds to a set of all agents in the network. The transfer planning approach enables agents to learn a large problem by decomposing it into smaller source problems. The term ‘transfer’ refers to the transferring of learning among multiple agents. The max-plus coordination algorithm and the transfer planning approach compute the global Q -value in order to achieve the global objective of a traffic network.

The proposed scheme has been shown to reduce the average delay (P.1) of the vehicles.

3) INVESTIGATION OF THE EFFECTS OF REAL WORLD TRAFFIC DATASET

Wei et al. [78] investigate the use of a real world traffic dataset consisting of data of more than 405 million vehicles recorded by using 1,704 surveillance cameras in Jinan, China covering 935 locations, out of which 43 of them are four-way intersections. The data is collected within a time period from 1st to 31st August 2016. The DRL model is based on the CNN architecture (N.2) and the value-based approach (E.1). The combination of the CNN architecture and the value-based approach allows this approach to use the convolutional

architecture (see Fig. 5) to analyze visual imagery while mapping each state-action pair to a state value (see Table 5 for more details). This model optimizes the Q -values to address the challenge of inappropriate traffic phase sequence (C.1) using a centralized model (T.1.1) in a real world traffic network (M.1.3), which is based on an urban traffic network in Jinan, China, with opposing through traffic (T.2.1). The traffic is characterized by a real world traffic model (M.2.2). This model is embedded in each intersection (G.1), where the state s_t represents the queue length (S.1), the current traffic phase (S.4), and the vehicle position (S.5). The proposed scheme is applied to 24 intersections. The action a_t represents the type of traffic phase to be activated in the next time instant (A.1). The reward $r_{t+1}(s_{t+1})$ represents the relative waiting time (R.1), the relative queue length (R.2) of the vehicles, and the phase transition (R.3) of the traffic phases. In the proposed scheme, the recorded data consists of the timing information (i.e., peak hours 7-9 A.M. and 5-7 P.M., and non-peak hours), the ID of each surveillance camera, and vehicular data (i.e., the position (S.5) of each vehicle). The recorded real world traffic data is fed to the input layer of the CNN architecture, and the output layer provides the Q -value of each possible

TABLE 7. Summary of DRL attributes for TSCs.

DL Architectures	Reference	States	Actions	Rewards	DRL Methods
		S.1 Queue length S.2 Red timing S.3 Green timing S.4 Current traffic phase S.5 Vehicle position S.6 Vehicle speed	A.1 Traffic phase A.2 Traffic phase split	R.1 Relative waiting time R.2 Relative queue length R.3 Phase transition	E.1 Value-based E.2 PG-based E.3 A2C-based
N.1 FCLN	Wan et al. [70], 2018	✓ ✓ ✓ ✓	✓	✓	✓
	Tan et al. [72], 2019	✓	✓	✓ ✓	✓
N.2 CNN	Genders et al. [66], 2016		✓ ✓ ✓	✓	✓
	Mousavi et al. [68], 2017	✓	✓	✓	✓ ✓
	Gao et al. [69], 2017		✓ ✓ ✓	✓	✓
	Van et al. [74], [76], 2016		✓ ✓ ✓	✓	✓
	Wei et al. [78], 2018	✓	✓ ✓	✓	✓ ✓ ✓
N.3 SAE	Li et al. [67], 2016	✓	✓	✓ ✓	✓
N.4 3DQN	Liang et al. [71], 2018		✓ ✓	✓	✓
	Gong et al. [77], 2019	✓	✓	✓	✓
	Wang et al. [73], 2019		✓ ✓	✓ ✓	✓
N.5 LSTM	Chu et al. [79], 2019	✓	✓	✓ ✓	✓

action, which is the type of traffic phase to be activated in the next time instant (A.1).

The proposed scheme has been shown to increase throughput (P.4) and reduce the average delay (P.1) and the queue length (P.3) of the vehicles.

4) COMPARISON OF VALUE-BASED AND PG-BASED METHODS

Mousavi et al. [68] compare the two different types of DRL methods, namely the value-based method (E.1) and the PG-based method (E.2), in TSCs. The DRL model is based on the CNN architecture (N.2). The combination of the CNN architecture and both value-based and PG-based methods allows this approach to use the convolutional architecture (see Fig. 5) to analyze visual imagery while mapping each state-action pair to a state value and selecting an action for a particular state (i.e., image) (see Table 5 for more details). This model optimizes the Q-values to address the challenge of inappropriate traffic phase sequence (C.1) using a centralized model (T.1.1) in a single intersection traffic network (M.1.1) with opposing through traffic (T.2.1). The traffic is characterized by Poisson process (M.2.1). This model is embedded in the TSC of the intersection (G.1). The state s_t represents the current traffic phase (S.4), and the queue length (S.1) of the vehicles. The action a_t represents the type of traffic phase to be activated in the next time instant (A.1). The reward $r_{t+1}(s_{t+1})$ represents the relative waiting time (R.1) of the

vehicles. The value-based method maps each state-action pair to a value $V_t(s_t)$ in order to identify the best possible action for each state, and the PG-based method selects an action for a certain state based on a policy. The value-based method achieves a slightly higher value of reward and outperforms the PG-based method.

The proposed scheme has been shown to reduce the average delay (P.1) and the queue length (P.3) of the vehicles, and so both value-based and PG-based methods are suitable for TSC.

C. DRL MODEL WITH SAE NEURAL NETWORK ARCHITECTURE

The application of DRL model based on the traditional SAE neural network architecture and the value-based approach for TSC is presented.

1) INVESTIGATION OF THE EFFECTS OF SAE NEURAL NETWORK ARCHITECTURE

Li et al. [67] investigate the use of the SAE neural network architecture that performs encoding and decoding functions to TSC. The DRL model is based on the SAE neural network architecture (N.3) and the value-based approach (E.1). The combination of the SAE architecture and the value-based approach allows this approach to use the encoding and decoding functions (see Fig. 6) to compress data while mapping each state-action pair to a state value (see Table 5 for

TABLE 8. Summary of key contributions, quantitative results/findings, and future directions.

DL Architectures	Reference	Key Contributions	Results/Findings	Future Directions
N.1 FCLN	Wan <i>et al.</i> [70], (2018)	Investigates the use of dynamic discount factors.	Reduces average delay (P.1) by up to 20% compared to baseline (i.e., a deterministic TSC).	To extend the proposed scheme to be applied in multiple intersections. Each intersection may use different DRL methods, such as actor-critic [113], deep deterministic policy gradient [114], and proximal policy optimization [115].
	Tan <i>et al.</i> [72], (2019)	Incorporates a novel reward function.	Reduces average queue length (P.3) by up to 40% compared to baseline (i.e., deterministic and fully-dynamic TSCs).	To extend the proposed scheme to be applied in multiple intersections.
N.2 CNN	Genders <i>et al.</i> [66], (2016)	Investigates the use of a large state space.	Reduces average delay (P.1) by up to 82% and average queue length (P.3) by up to 66% compared to baseline (i.e., a fully-dynamic TSC based on a neural network with one hidden layer).	To extend the proposed scheme to control the red and yellow phases of TSC.
	Gao <i>et al.</i> [69], (2017)	Investigates the use of a large state space	Reduces average delay (P.1) by up to 86% compared to baseline (i.e., a deterministic TSC).	To extend the proposed scheme to be applied in multiple intersections.
	Mousavi <i>et al.</i> [68], (2017)	Compares the value-based (E.1) and PG-based (E.2) methods with a baseline (i.e., a fully-dynamic TSC based on a neural network with one hidden layer).	The PG-based (E.2) method reduces the average delay (P.1) by up to 67% and queue length by up to 72%. The value-based (E.1) method reduces the average delay (P.1) by up to 68% and queue length (P.3) by up to 73%.	To extend the proposed scheme to be applied in multiple intersections.
	Van <i>et al.</i> [74], [76], (2016)	Incorporates max-plus coordination and transfer planning algorithms.	Reduces average delay (P.1) by up to 20% compared to MARL.	To adopt different kinds of DRL approaches because the proposed approach uses the traditional DQN approach, which has shown to be unstable.
	Wei <i>et al.</i> [78], (2018)	Investigates the use of real-world traffic dataset.	Reduces average delay (P.1) by up to 19% and queue length (P.3) by up to 38% compared to baseline (i.e., deterministic and semi-dynamic TSCs).	To extend the proposed scheme to control the yellow phase of TSC.
N.3 SAE	Li <i>et al.</i> [67], (2016)	Investigates the use of the SAE neural network architecture.	Reduces average delay (P.1) by up to 14% compared to baseline (i.e., a fully-dynamic TSC).	
N.4 3DQN	Liang <i>et al.</i> [71], (2018)	Incorporates a prioritized experience replay technique.	Reduces average waiting time (P.2) by up to 20% compared to baseline (i.e., a deterministic TSC).	
	Wang <i>et al.</i> [73], (2019)	Investigates the use of high-resolution event-based data.	Reduces average delay (P.1) by up to 21% and queue length (P.3) by up to 30% compared to baseline (i.e., a deterministic and fully-dynamic TSC).	To extend the proposed scheme to take account of traffic disturbances, including detector noise, traffic accidents, and bad weather conditions.
	Gong <i>et al.</i> [77], (2019)	Incorporates MARL with the 3DQN (N.4) architecture.	Reduces average delay (P.1) by up to 46% compared to baseline (i.e., deterministic and fully-dynamic TSCs).	To extend the proposed scheme to ensure fairness among traffic flows.
N.5 LSTM	Chu <i>et al.</i> [79], (2019)	Investigates the use of the LSTM neural network architecture.	Reduces average delay (P.1) by up to 10% and queue length (P.3) by up to 17% compared to baseline (i.e., a fully-dynamic TSC).	To extend the proposed scheme to improve communication among multiple intersections.

TABLE 9. Summary of traffic simulators and performance measures.

DL Architectures	Reference	Traffic Simulators			Performance Measures			
		SUMO	Paranics	VISSIM	Aimsun Next	P.1 Lower average delay	P.2 Lower average waiting time	P.3 Smaller queue length
N.1 FCLN	Wan et al. [70], 2018			✓		✓		✓
	Tan et al. [72], 2019			✓			✓	
N.2 CNN	Genders et al. [66], 2016	✓				✓	✓	✓
	Mousavi et al. [68], 2017	✓				✓	✓	
	Gao et al. [69], 2017	✓				✓	✓	
	Van et al. [74], [76], 2016	✓				✓		
	Wei et al. [78], 2018	✓				✓	✓	✓
N.3 SAE	Li et al. [67], 2016		✓			✓	✓	
N.4 3DQN	Liang et al. [71], 2018	✓				✓		
	Wang et al. [73], 2019	✓					✓	✓
	Gong et al. [77], 2019				✓	✓		✓
N.5 LSTM	Chu et al. [79], 2019	✓				✓	✓	✓

more details). This model optimizes the Q -values to address the challenge of inappropriate traffic phase split (C.2) using a centralized model (T.1.1) in a single intersection traffic network (M.1.1) with opposing through traffic (T.2.1). This model is embedded in each intersection (G.1), where the state s_t represents the queue length (S.1) of the vehicles. The action a_t represents the choice to either keep the current traffic phase or switch to the next traffic phase in a predetermined sequence of traffic phases at the next time instant (A.2). The reward $r_{t+1}(s_{t+1})$ represents the relative waiting time (R.1) and the relative queue length (R.2) of the vehicles. In the proposed scheme, the SAE neural network architecture consists of one input, two hidden, and one output layers. The input layer encodes the input data, such as the queue length (S.1) of the vehicles, using an encoding function (see Equation (10)), to provide compressed data. The second hidden layer reconstructs the data using a decoding function (see Equation (10)). Finally, the output layer provides the Q -value of each possible action.

The proposed scheme has been shown to reduce the average delay (P.1) and the queue length (P.3) of the vehicles.

D. DRL MODELS WITH 3DQN ARCHITECTURE

The application of various DRL models with the traditional 3DQN architecture and the value-based approach for TSCs is presented.

1) LIANG’S ENHANCEMENT WITH PRIORITIZED EXPERIENCE REPLAY

Liang et al. [71] incorporate a prioritized experience replay approach [96] to the traditional 3DQN architecture (N.4), which consists of double Q -learning and a dueling network, and the value-based approach (E.1), running the DQN algorithm (see Algorithm 2). The combination of the 3DQN architecture and the value-based approach allows this approach to use double Q -learning and a dueling network to increase the learning speed (see Fig. 7) while mapping each state-action pair to a state value (see Table 5 for more details). This model optimizes the Q -values to address the challenge of inappropriate traffic phase split (C.2) using a centralized model (T.1.1) in a single intersection traffic network (M.1.1) with (T.2.1) and without opposing through traffic (T.2.2). The traffic is characterized by a real world traffic model (M.2.2). This model is embedded in the TSC of the intersection (G.1). The state s_t represents the position (S.5) and speed (S.6) of the vehicles. The action a_t represents the choice to either keep the current traffic phase or switch to the next traffic phase in a predetermined sequence of traffic phases at the next time instant (A.2). The reward $r_{t+1}(s_{t+1})$ represents the relative waiting time (R.1) of the vehicles. In the proposed scheme, the prioritized experience replay chooses experiences from the replay memory on the priority basis in order to increase the learning rate. The prioritized experience replay ranks an

experience i , which increases its replay probability, based on the temporal difference error δ calculated as follows:

$$\delta_i = |Q(s, a; \theta)_i - Q(s, a; \theta^-)_i| \quad (20)$$

where an experience with a lower error is being ranked higher (or prioritized). The replay probability of experience i is calculated as follows:

$$P_i = \frac{p_i^\varphi}{\sum_k p_k^\varphi} \quad (21)$$

where p_i is the priority of an experience i , and φ represents the priority level. Higher φ represents a higher priority, and vice-versa, while $\varphi = 0$ represents a random sampling.

The proposed scheme has been shown to reduce the average waiting time (P.2) of the vehicles.

2) GONG'S ENHANCEMENT WITH MARL

Gong *et al.* [77] incorporate MARL to the traditional 3DQN architecture (N.4), which consists of double Q -learning and a dueling network, and the value-based approach (E.1), running the DQN algorithm (see Algorithm 2). The combination of the 3DQN architecture and the value-based approach allows this approach to use double Q -learning and a dueling network to increase the learning speed (see Fig. 7) while mapping each state-action pair to a state value (see Table 5 for more details). MARL enables coordination among multiple agents. This model optimizes the Q -values to address the challenge of inappropriate traffic phase sequence (C.1) in a multi intersection traffic network (M.1.2) and a real world traffic network (M.1.3), which is based on an urban traffic network in Florida, United States, using a distributed model (T.1.2). The traffic is characterized by a real world traffic model (M.2.2). This model is embedded in the TSC of the intersection (G.1). The state s_t represents the queue length (S.1), and the position (S.5), of the vehicles. The action a_t represents the type of traffic phase to be activated in the next time instant (A.1). The reward $r_{t+1}(s_{t+1})$ represents the relative waiting time (R.1) of the vehicles. In the proposed scheme, the MARL algorithm enables agents to exchange information (i.e., rewards and Q -values) with each other in order to coordinate their actions.

Algorithm 3 shows the MARL algorithm for DRL. At time instant t , an agent i observes the current state $s_t^i \in S$ from the operating environment, and sends its own Q -value $Q_t^i(s_t^i, a_t^i)$ to the neighboring agents J^i . Subsequently, following steps 5 to 13 of Algorithm 2, agent i receives the optimal Q -value $\max_{a^j \in A} Q_t^j(s_t^j, a^j)$ from each neighboring agent $j \in J^i$, selects an action $a_t^i \in A$ based on the Q -value at time t , and then receives a reward $r_{t+1}^i(s_{t+1}^i)$ under the next state $s_{t+1}^i \in S$ at time $t + 1$. Finally, the agent i updates Q -value $Q_t^i(s_t^i, a_t^i)$. Based on Equation (2), the Q -value $Q_t^i(s_t^i, a_t^i)$ is updated using Q -function as follows [97]:

$$Q_{t+1}^i(s_t^i, a_t^i) \leftarrow Q_t^i(s_t^i, a_t^i) + \alpha \delta_t^i(s_t^i, a_t^i) \quad (22)$$

Meanwhile, the Q -value $Q_t^j(s_t^j, a_t^j)$ of a neighboring agent $j \in J^i$ is updated using Q -function as follows:

$$Q_{t+1}^j(s_t^j, a_t^j) \leftarrow Q_t^j(s_t^j, a_t^j) + \alpha \delta_t^j(s_t^j, a_t^j) \quad (23)$$

The proposed scheme has been shown to increase the throughput (P.4) and reduce the average delay (P.1) of the vehicles.

Algorithm 3 MARL Algorithm for DRL

```

1: Procedure
2:   for episode = 1: M do
3:     observe current state  $s_t^i$ 
4:     send  $Q$ -value  $Q_t^i(s_t^i, a_t^i)$  to neighboring agents  $J^i$ 
5:     receive  $\max_{a^j \in A} Q_t^j(s_t^j, a^j)$  from agent  $j \in J^i$ 
6:     for  $t = 1 : T$  do
7:       perform steps 5 to 13 of Algorithm 2
8:     end for
9:     update  $Q$ -value  $Q_{t+1}^i(s_t^i, a_t^i)$  using Equation (22)
10:  end for
11: End Procedure

```

3) INVESTIGATION OF THE EFFECTS OF HIGH-RESOLUTION EVENT-BASED DATA

Wang *et al.* [86] investigate the use of high-resolution event-based data that includes a large amount of useful information about vehicles, including their movements and positions. The DRL model is based on the 3DQN architecture (N.4) and the value-based approach (E.1). The combination of the 3DQN architecture and the value-based approach allows this approach to use double Q -learning and a dueling network to increase the learning speed (see Fig. 7) while mapping each state-action pair to a state value (see Table 5 for more details). This model optimizes the Q -values to address the challenge of inappropriate traffic phase sequence (C.1) using a centralized model (T.1.1) in a single intersection traffic network (M.1.1) with (T.2.1) and without opposing through traffic (T.2.2). The traffic is characterized by a real world traffic model (M.2.2). This model is embedded in the TSC of the intersection (G.1). The state s_t represents the green timing (S.3), and the vehicle position (S.5). The action a_t represents the type of traffic phase to be activated (A.1) in the next time instant. The reward $r_{t+1}(s_{t+1})$ represents the relative waiting time (R.1), and the relative queue length (R.2) of the vehicles. The high-resolution event-based data provides a large amount of useful information about the vehicle, such as vehicular movement and position. The high-resolution event-based data keeps track of: a) the time of each vehicle arriving at and departing from an inductive loop detector (or vehicle detector); and b) the time gap between two consecutive vehicles, which is the time gap between the two vehicles arriving at and departing from the detector. The 3DQN architecture consists of one input layer, three convolutional layers, three FC layers (in which the third FC layer is split into two separate streams as explained in N.4), and one output layer. The input layer receives the accurate traffic information, and the output layer provides an accurate Q -value for each possible action based on the accurate information [98].

The proposed scheme has been shown to increase throughput (P.4) and reduce the queue length of vehicles (P.3).

E. DRL MODEL WITH LSTM NEURAL NETWORK ARCHITECTURE

The application of DRL model based on the traditional LSTM neural network architecture and the A2C-based approach for TSC is presented.

1) INVESTIGATION OF THE EFFECTS OF LSTM NEURAL NETWORK ARCHITECTURE

Chu *et al.* [79] investigate the use of LSTM neural network architecture that provides memory to memorize previous inputs of TSC. The DRL model is based on the LSTM neural network architecture (N.5) and the A2C-based approach (E.3). The combination of LSTM and the A2C-based approach allows this approach to use the LSTM neural network (see Fig. 8) to provide memorization of previous inputs while combining both value-based and PG-based methods to control its behavior and to measure the suitability of the selected action (see Table 5 for more details). This model optimizes the Q -values to address the challenge of inappropriate traffic phase sequence (C.1) using a distributed model (T.1.2) in a multi intersection traffic network (M.1.2), an urban traffic network based on Monaco (M.1.3), and a grid traffic network (M.1.4) with opposing through traffic (T.2.1). The traffic is characterized by a real world traffic model (M.2.2). This model is embedded in the TSC of the intersection (G.1). The state s_t represents the queue length (S.1) of the vehicles. The action a_t represents the type of traffic phase to be activated in the next time instant (A.1). The reward $r_{t+1}(s_{t+1})$ represents the relative waiting time (R.1) and the relative queue length (R.2) of the vehicles. In the proposed scheme, the A2C-based method has been used with the LSTM neural network architecture, which consists of one input, one FC, one LSTM (i.e., memory cell), and one output layer. The output layer is separated into two streams: a) *actor*, which controls the behavior of an agent (i.e., policy-based); and b) *critic*, which measures the suitability of the selected action (i.e., value-based). The gradient of the loss function for A2C is calculated using Equation (15).

The proposed scheme has been shown to increase throughput (P.4) and reduce the average delay (P.1) and queue length (P.3) of vehicles.

VI. GUIDELINES AND DESIGN CONSIDERATIONS FOR THE APPLICATION OF DEEP REINFORCEMENT LEARNING FOR TRAFFIC SIGNAL CONTROL SYSTEMS

The guidelines and design considerations for the application of DRL to TSC is presented in this section, which helps in the identification of suitable DRL solutions for different TSC problems. Table 5 provides the description of various DL architectures and DRL methods with their strengths. Table 6 provides a summary of various TSC attributes, including challenges, traffic network architectures, traffic characteristics, TSC architectures, and traffic phases,

which are applied with DRL solutions. Table 7 provides a summary of various DRL attributes, including agent (i.e., TSC), states, actions, rewards, and DRL methods for TSCs. Table 8 provides a summary of various key contributions, quantitative results/findings, and future directions that have been presented in the literature. These tables can be used to identify the suitable DRL solutions for different TSC problems. Two main aspects must be considered when applying DRL to TSCs. *Firstly*, an open issue or a problem needs to be identified and well understood. This includes the objectives, the problem statement, as well as the research questions of the problem. *Secondly*, the research questions are answered. The guidelines and considerations for applying DRL to TSCs are presented based on a sample case study [71] being referred to throughout this subsection. In [71], a DRL model with the 3DQN architecture and the value-based approach is applied to TSC in order to reduce the average travel time of the vehicles. Next, we define the state, action and reward representations, and discuss the selection of the method for DRL. Lastly, we define the DL architecture. In general, the state captures the TSC attributes, such as queue length (S.1), red timing (S.2), green timing (S.3), current traffic phase (S.4), vehicle position (S.5), and vehicle speed (S.6), and so it has a direct relevance to the problem. This explains that the state, action, and reward representations are defined prior to method and network architecture.

A. DEFINING STATE

The decision-making factors that an agent observes from the operating environment should be well defined. Table 7 provides a summary of how states (see Fig. 12) have been represented in the literature. For instance, in [71], the objective is to maximize the reward in order to reduce the average waiting time of the vehicles at an intersection. Therefore, the agent represents a state with the position (S.5) and speed (S.6) of a vehicle. Upon observation of the state, the agent can decide its action, which is based on the state. Similar to other schemes [67], [79], the input layer consists of input neurons. In [71], the input layer represents a grid with a size of 60×60 , where there are $60 \times 60 \times 2$ input states to represent the position (S.5) and speed (S.6) of a vehicle.

B. DEFINING ACTION

The possible actions should be well defined so that an agent can maximize its rewards by taking appropriate actions. Table 7 provides a summary of how actions (see Fig. 12) have been represented in the literature. For instance, in [71], with respect to the objective of reducing the average waiting time of the vehicles, the agent must select an appropriate time interval of a traffic phase. The action represents the choice to either keep the current traffic phase or switch to the next traffic phase in a predetermined sequence of traffic phases at the next time instant (A.2) in order to address the challenge of inappropriate traffic phase split (C.2). In [71], the output layer consists of nine neurons, and each of them represents a possible action.

C. DEFINING REWARD

The reward should be well defined so that it reflects the objectives that an agent aims to achieve after performing an action under the state. Table 7 provides a summary of how rewards (see Fig. 12) have been represented in the literature. For instance, in [71], the reward is the increment/decrement of the average waiting time of the vehicles at an intersection. Therefore, the agent represents the reward with the relative waiting time (R.1). By increasing the reward, an agent improves system performance while achieving its objectives.

D. CHOOSING A METHOD

The objectives of a method (e.g., adjusting the discount factor dynamically, or integrating several mechanisms into a single framework) with respect to the model should be well defined. Table 7 provides a summary of how various methods (see Fig. 12) have been used in the literature. For instance, in [71], several mechanisms, including double Q -learning, dueling network, and prioritized experience replay, are incorporated into a single framework in order to increase learning rate. Higher learning rate reduces the learning time, which is required to explore all state-action pairs in order to identify the optimal action. The optimal action, such as the choice to either keep the current traffic phase or switch to the next traffic phase in a predetermined sequence of traffic phases at the next time instant (A.2), helps a TSC to achieve a smoother traffic flow. The value-based method maps each state-action pair to a value $V_t(s_t)$ in order to identify the best possible action for each state. This helps to achieve the objectives of TSCs, and so it is chosen. The rest of the DRL methods are presented in Section II-D2, which can be selected based on the objectives. For instance, the PG-based method is suitable for the objective of selecting an action for a certain state based on a policy [68].

E. DEFINING ARCHITECTURE

To address the challenges of TSC, a suitable DL architecture for DRL should be well defined. Table 5 provides the description of various DL architectures with different DRL methods as well as their strengths, while Tables 6-8 provide a summary of how various DL architectures (see Fig. 12) have been used in the literature. For instance, in [71], the 3DQN architecture consists of three convolutional and two FC layers, and it is used to capture the position (S.5) and speed (S.6) of a vehicle in order to address the challenge of inappropriate traffic phase split (C.2). Since the position and speed are captured in the form of images and videos, the 3DQN architecture with convolutional layers is selected. The identification of a suitable number of layers is an important aspect. Lower number of layers may struggle to fit the training data, while higher number of layers may cause overfit due to memorizing the properties of training data, which affect the performance negatively.

Similarly, the identification of a suitable number of neurons in each layer is another important aspect. In general,

the number of neurons in the input layer is equivalent to the number of features. For instance, there are eighty neurons in the input layer to represent eighty cells of an intersection, which enable the representation of the queue length (S.1) and the position (S.5) of the vehicles [99]. However, the number of neurons in the hidden layer(s) is not straightforward and are generally determined empirically, although higher number of neurons tend to improve system performance at the expense of increased complexity [100].

VII. OPEN ISSUES

While DRL for TSCs has been investigated in the literature, there are still substantial open issues that have not been well studied for real world deployment. This section presents open issues that can be pursued in this topic in the future.

A. ADDRESSING THE EFFECTS OF DYNAMICITY TO DQN

The state space may be highly dimensional when traffic images are used as part of the state representation [76], [78]. Higher dimension of state representation is essential to represent high-quality images that capture moving vehicles, and this can increase the size of state space. To address this issue, computing techniques, such as discretization and quantization of state space, can be incorporated into DRL applied to TSC in order to encode and decode between the high-dimensional state representation and low-dimensional state representation. The solutions can provide an abstract representation of high-dimensional and complex state representation in order to simplify large, as well as dynamic, states.

B. ADDRESSING THE LEARNING EFFICIENCY OF DQN FOR TSC

Trial-and-error, which is essential to learning in DQN, incurs high learning cost such as a longer learning time that is unacceptable in real-world traffic management. While existing DQN methods generate impressive results in simulated environments, such as the Alpha Go or Atari games [14], they require a large number of trials and errors. Consequently, learning in DQN-based TSCs can cause traffic congestion in real world. Several mechanisms can be applied to increase learning efficiency. Firstly, knowledge exchange among multiple intersections helps to coordinate their actions, whereby the operating environment (e.g., the congestion level) of an intersection affects the congestion level of neighboring intersections since vehicles traverse from one intersection to another. The knowledge (e.g., Q -values) exchanged among multiple intersections takes the traffic condition of individual intersection into consideration to improve the global reward, which reflects the traffic condition of the entire traffic network, and increase the efficiency of learning. Secondly, enhanced exploration approaches, for instance, the model-based exploration approach creates a model of the operating environment, and then selects an action that increases the possibility of exploring unseen states during exploration. The model-based exploration approach has been applied to *Lunar Lander* and *Mountain Car* [116]. Nevertheless,

the model-based approach has higher complexity and computational requirement compared to existing approaches, which are model-free in nature. Future investigations could be pursued to improve the efficiency of learning in DQN for TSC.

C. ADDRESSING THE EFFECTS OF TRAFFIC DISTURBANCES TO LEARNING IN DQN FOR TSC

In the real world deployment of DRL, TSC must be robust and reliable against unexpected traffic disturbances, such as bad weather conditions, road accidents, or construction. However, the available information for such events is usually sparse and incomplete, and data that integrates several factors may be even sparser. Learning under such circumstances can be challenging. LSTM contains memory cells that can store historical information (or data) [79], including predictions and their inaccuracy, that can be explored to reduce the effects of disturbance (e.g., quantifying the effects of disturbance) and improve the accuracy of prediction (e.g., reducing the effects of disturbance) as time goes by. While state captures the traffic conditions that need to be monitored at all times, the disturbance can be captured as event that must be detected whenever it occurs. However, the occurrence of events (e.g., accidents) is likely to be sparse with incomplete information, and so historical information can be useful under such circumstances. Future investigations could be pursued to tackle these factors when collecting the data in order to improve the efficiency of learning from traffic disturbances.

D. ADDRESSING THE SAFETY ISSUE OF DQN FOR TSC

Making DRL agents acceptably safe in real world environment is another pressing area for future research. While DRL models learn from trial-and-error, the learning cost of DRL can be critical, or even fatal in the real world as the malfunction of traffic signals might lead to accidents. Therefore, adopting risk management into DRL helps to prevent unwanted behavior during and after the learning process of DRL agents. Each action is associated with a risk factor, and subsequently rules can be designed to exclude high-risk actions from a set of feasible actions. The risk factors of different actions can be explored and validated in simulation at preliminary stage, and then improved conservatively during operation as time goes by in order to minimize the learning cost. Future investigations could be pursued to address the safety issue of DQN for TSC.

E. ADDRESSING THE FAIRNESS AND PRIORITIZED ACCESS ISSUE

Fairness and prioritized access to intersection using traditional and enhanced DQN approaches have not been investigated in the literature. The reward function can be revised to achieve fairness among traffic flows while traversing from one intersection to another. In addition, there are lack of investigations of prioritized access in the presence of emergency vehicles, such as ambulance and fire engines, that traverse from one intersection to another on a priority basis. In addition to high-resolution data that has been used to

capture the position (S.5) and the speed (S.6) of vehicles [73], detecting certain vehicles (e.g. ambulances and fire engines) accurately using camera (i.e., high-resolution photos), video camera (i.e., high-resolution videos), and sensors, must be integrated to DQN so that it can carry out the right action to prioritize such vehicles. The reward function should also be altered to cater for the prioritized vehicles. Future investigations could be pursued to address these aspects so that fairness among traffic flows can be achieved, and prioritized vehicles can cross an intersection on a priority basis with minimal effects to existing traffic.

F. DEVELOPING TRAFFIC SIMULATORS FOR INVESTIGATING DQN-BASED TSCs

Most traffic simulators adopt the microscopic approach, in which the focus is on the mobility characteristics of an individual vehicle. However, there are lack of investigations using most kinds of TSCs, including the DQN-based TSCs, for controlling the overall traffic flows at the macroscopic level that takes account of the general traffic density, vehicles distributions, and so on. Future investigations could be pursued to explore the use of macroscopic attributes in order to improve the performance achieved by the microscopic approach in providing more accurate results.

G. CONDUCTING A LITERATURE REVIEW OF DRL FROM THE TSC PERSPECTIVE

With the rapid advancement of intelligent transportation systems, a review from the TSC perspective is becoming essential. While DRL has been proposed to implement the fully-dynamic TSC (see Section I), other kinds of TSCs, including the deterministic and semi-dynamic TSCs, may be useful in different kinds of scenarios. As this article addresses this topic from the DRL perspective, another article to understand how has this topic been developed and extended from the TSC perspective can complement this article to provide a complete current research landscape of the application of DRL to TSCs. This is because while DRL has been used to address two main challenges, namely inappropriate traffic phase sequence (C.1), and inappropriate traffic phase split (C.2), in TSCs, other challenges brought about by the enhanced TSCs and intelligent transportation systems over the years may open more investigations into the application of DRL to TSCs. In addition to the current DRL models, namely the centralized model (T.1.1) and the distributed model (T.1.2) used in TSCs, this may require exploring other kinds of models such as a hybrid model with different degrees of centralized and distributed decision-makings. Also, in addition to the current traffic network models, namely the single intersection traffic network (M.1.1), multi intersection traffic network (M.1.2), real world traffic network (M.1.3), and grid traffic network (M.1.4), this may require exploring other kinds of models integrated with recent advancements in the transport ecosystem. While the complex traffic networks may be integrated with other modes of transport, such as walking and cycling, the investigation of DRL

applied to TSCs has been limited to opposing through traffic (T.2.1), without opposing through traffic (T.2.2), and with grouped individual traffic (T.2.3). Hence, further literature review can be conducted from the TSC perspective to provide a new and refreshed look at this topic.

VIII. CONCLUSION

In this article, we present a comprehensive review of the application of deep reinforcement learning (DRL) to traffic signal control (TSC). For smoother traffic flow, the underlying intersection with different architectures and dynamic traffic arrival rates have posed significant challenges to TSCs to select the right choice of traffic phases, as well as their duration. This article discusses how TSC can be formulated as an DRL problem using appropriate representations (i.e., state, action, and reward), and used to solve the problem using a popular DRL approach called deep Q -network (DQN). Subsequently, this article presents various kinds of deep learning (DL) architectures and DRL methods, and highlights their strengths in addressing the challenges brought about by the medium- and heavy-loaded traffic at intersections. After that, the performance measures, simulation platforms, and complexity analysis of the DRL approaches are investigated. This article also provides guidelines and design considerations for the application of DRL to TSC. Finally, we discuss some open issues for future research of DQN-based TSCs.

REFERENCES

- [1] K. Molloy and J. V. Ward Benson, "Traffic signal control system," U.S. Patent 3,754,209, Aug. 21, 1973.
- [2] P. Mirchandani and L. Head, "A real-time traffic signal control system: Architecture, algorithms, and analysis," *Transp. Res. C, Emerg. Technol.*, vol. 9, no. 6, pp. 415–432, Dec. 2001.
- [3] F. Dion and B. Hellinga, "A rule-based real-time traffic responsive signal control system with transit priority: Application to an isolated intersection," *Transp. Res. B, Methodol.*, vol. 36, no. 4, pp. 325–343, May 2002.
- [4] L. Zhao, Y.-C. Lai, K. Park, and N. Ye, "Onset of traffic congestion in complex networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 71, no. 2, Feb. 2005, Art. no. 026125.
- [5] B. Yin, A. El Moudni, and M. Dridi, "Traffic network micro-simulation model and control algorithm based on approximate dynamic programming," *IET Intell. Transp. Syst.*, vol. 10, no. 3, pp. 186–196, Apr. 2016.
- [6] Cools, Seung-Bae, Carlos Gershenson, and Bart D'Hooghe, "Self-organizing traffic lights: A realistic simulation," in *Advances in Applied Self-Organizing Systems*. London, U.K.: Springer, 2013, pp. 45–55.
- [7] 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 Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1140–1150, Sep. 2013.
- [8] L. Chun-Gui, W. Meng, S. Zi-Gaung, L. Fei-Ying, and Z. Zeng-Fang, "Urban traffic signal learning control using fuzzy actor-critic methods," in *Proc. 5th Int. Conf. Natural Comput.*, Aug. 2009, pp. 368–372.
- [9] J. C. Medina and R. F. Benekohal, "Traffic signal control using reinforcement learning and the max-plus algorithm as a coordinating strategy," in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 596–601.
- [10] P. K. J., H. Kumar A. N., and S. Bhatnagar, "Decentralized learning for traffic signal control," in *Proc. 7th Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2015, pp. 1–6.
- [11] L. A. Prashanth and S. Bhatnagar, "Threshold tuning using stochastic optimization for graded signal control," *IEEE Trans. Veh. Technol.*, vol. 61, no. 9, pp. 3865–3880, Nov. 2012.
- [12] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *J. Transp. Eng.*, vol. 129, no. 3, pp. 278–285, May 2003.
- [13] 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 Comput. Surv.*, vol. 50, no. 3, pp. 1–38, Oct. 2017.
- [14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: <http://arxiv.org/abs/1312.5602>
- [15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
- [16] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.
- [17] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3389–3396.
- [18] A. R. Sharma and P. Kaushik, "Literature survey of statistical, deep and reinforcement learning in natural language processing," in *Proc. Int. Conf. Comput., Commun. Autom. (ICCCA)*, May 2017, pp. 350–354.
- [19] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature Med.*, vol. 25, no. 1, pp. 24–29, 2019.
- [20] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," 2017, *arXiv:1706.10059*. [Online]. Available: <http://arxiv.org/abs/1706.10059>
- [21] S. P. K. Spielberg, R. B. Gopaluni, and P. D. Loewen, "Deep reinforcement learning approaches for process control," in *Proc. 6th Int. Symp. Adv. Control Ind. Processes (AdCONIP)*, May 2017, pp. 201–206.
- [22] D. Zhang, X. Han, and C. Deng, "Review on the research and practice of deep learning and reinforcement learning in smart grids," *CSEE J. Power Energy Syst.*, vol. 4, no. 3, pp. 362–370, Sep. 2018.
- [23] Z. Ren, X. Wang, N. Zhang, X. Lv, and L.-J. Li, "Deep reinforcement learning-based image captioning with embedding reward," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 290–298.
- [24] A. E. L. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electron. Imag.*, vol. 19, pp. 70–76, Jan. 2017.
- [25] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7559–7566.
- [26] W. Wei, Y. Zhang, J. B. Mbede, Z. Zhang, and J. Song, "Traffic signal control using fuzzy logic and MOGA," in *Proc. IEEE Int. Conf. Syst., Man Cybern. e-Systems e-Man for Cybern. Cyberspace (Cat.No.01CH6)*, 2001, pp. 1335–1340.
- [27] L. Singh, S. Tripathi, and H. Arora, "Time optimization for traffic signal control using genetic algorithm," *Int. J. Recent Trends Eng.*, vol. 2, no. 2, p. 4, 2009.
- [28] T. Li, D. Zhao, and J. Yi, "Adaptive dynamic programming for multi-intersections traffic signal intelligent control," in *Proc. 11th Int. IEEE Conf. Intell. Transp. Syst.*, Oct. 2008, pp. 286–291.
- [29] A. L. C. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," *Auton. Agents Multi-Agent Syst.*, vol. 18, no. 3, p. 342, 2009.
- [30] H. Wei, G. Zheng, V. Gayah, and Z. Li, "A survey on traffic signal control methods," 2019, *arXiv:1904.08117*. [Online]. Available: <http://arxiv.org/abs/1904.08117>
- [31] 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*. Cham, Switzerland: Birkhauser, 2016, pp. 47–66.
- [32] Z. Liu, "A survey of intelligence methods in urban traffic signal control," *Int. J. Comput. Sci. Netw. Secur.*, vol. 7, no. 7, pp. 105–112, 2007.
- [33] D. Zhao, Y. Dai, and Z. Zhang, "Computational intelligence in urban traffic signal control: A survey," *IEEE Trans. Syst., Man, Cybern., C (Appl. Rev.)*, vol. 42, no. 4, pp. 485–494, Jul. 2012.
- [34] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 1, pp. 383–398, Jan. 2019.

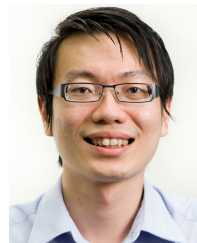
- [35] M. Veres and M. Moussa, "Deep learning for intelligent transportation systems: A survey of emerging trends," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 8, pp. 3152–3168, Aug. 2020.
- [36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [37] Y. Bengio, I. Goodfellow, and A. Courville, *Deep Learning*, vol. 1. Cambridge, MA, USA: MIT Press, 2017.
- [38] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
- [39] J. M. Zurada, *Introduction to Artificial Neural Systems*, vol. 8. St. Paul, Turkey: West, 1992.
- [40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.
- [41] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice-Hall, 1994.
- [42] R. S. Sutton and G. Andrew Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
- [43] C. Szepesvari, "Algorithms for reinforcement learning," *Synth. lectures Artif. Intell. Mach. Learn.*, vol. 4, no. 1, pp. 1–103, 2010.
- [44] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, and D. Hassabis, "Reinforcement learning, fast and slow," *Trends Cognit. Sci.*, vol. 23, no. 5, pp. 408–422, May 2019.
- [45] Nachum, Ofir, Shixiang Shane Gu, Honglak Lee, and Sergey Levine, "Data-efficient hierarchical reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 3303–3313.
- [46] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 6292–6299.
- [47] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, and T. Graepel, "Human-level performance in 3D multiplayer games with population-based reinforcement learning," *Science*, vol. 364, no. 6443, pp. 859–865, May 2019.
- [48] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement learning from imperfect demonstrations," 2018, *arXiv:1802.05313*. [Online]. Available: <http://arxiv.org/abs/1802.05313>
- [49] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proc. 32nd AAAI Conf. Artif. Intell. AAAI*, 2018, pp. 2661–2669.
- [50] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.
- [51] P. R. Montague, "Reinforcement learning: an introduction, by Sutton, RS and Barto, AG," *Trends Cogn. Sci.*, vol. 3, no. 9, p. 360, 1999.
- [52] R. S. Sutton and G. Andrew Barto, *Introduction to Reinforcement Learning*, vol. 135. Cambridge, MA, USA: MIT Press, 1998.
- [53] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.
- [54] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*. [Online]. Available: <http://arxiv.org/abs/1701.07274>
- [55] H. Ye, G. Y. Li, and B.-H.-F. Juang, "Deep reinforcement learning based resource allocation for V2 V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.
- [56] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proc. 32nd AAAI Conf. Artif. Intell. AAAI*, 2018, pp. 3200–3207.
- [57] T. Thi Nguyen and V. Janapa Reddi, "Deep reinforcement learning for cyber security," 2019, *arXiv:1906.05799*. [Online]. Available: <http://arxiv.org/abs/1906.05799>
- [58] Rolnick, David, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne, "Experience replay for continual learning," in *Adv. Neural Inf. Process. Syst.*, pp. 348–358, 2019.
- [59] X.-L. Chen, L. Cao, C.-X. Li, Z.-X. Xu, and J. Lai, "Ensemble network architecture for deep reinforcement learning," *Math. Problems Eng.*, vol. 2018, pp. 1–6, 2018.
- [60] Y. Takano, H. Inoue, R. Thawonmas, and T. Harada, "Self-play for training general fighting game AI," in *Proc. Nicograph Int. (Nicolnt)*, Jul. 2019, p. 120.
- [61] V. Behzadan and W. Hsu, "Analysis and improvement of adversarial training in DQN agents with adversarially-guided exploration (AGE)," 2019, *arXiv:1906.01119*. [Online]. Available: <http://arxiv.org/abs/1906.01119>
- [62] Y. Liu, L. Zhang, Y. Wei, and Z. Wang, "Energy efficient training task assignment scheme for mobile distributed deep learning scenario using DQN," in *Proc. IEEE 7th Int. Conf. Comput. Sci. Netw. Technol. (ICC-SNT)*, Oct. 2019, pp. 442–446.
- [63] G. Liu, R. Wu, H.-T. Cheng, J. Wang, J. Ooi, L. Li, A. Li, W. Lok Sibon Li, C. Boutilier, and E. Chi, "Data efficient training for reinforcement learning with adaptive behavior policy sharing," 2020, *arXiv:2002.05229*. [Online]. Available: <http://arxiv.org/abs/2002.05229>
- [64] Anschel, Oron, Nir Baram, and Nahum Shimkin, "Deep reinforcement learning with averaged target DQN," 2016, *arXiv:1611.01929*. [Online]. Available: <http://arxiv.org/abs/1611.01929>
- [65] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*. [Online]. Available: <http://arxiv.org/abs/1609.04747>
- [66] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," 2016, *arXiv:1611.01142*. [Online]. Available: <http://arxiv.org/abs/1611.01142>
- [67] L. Li, Y. Lv, and F.-Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA J. Automat. Sinica*, vol. 3, no. 3, pp. 247–254, Apr. 2016.
- [68] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," *IET Intell. Transp. Syst.*, vol. 11, no. 7, pp. 417–423, Sep. 2017.
- [69] 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," 2017, *arXiv:1705.02755*. [Online]. Available: <http://arxiv.org/abs/1705.02755>
- [70] C.-H. Wan and M.-C. Hwang, "Value-based deep reinforcement learning for adaptive isolated intersection signal control," *IET Intell. Transp. Syst.*, vol. 12, no. 9, pp. 1005–1010, Nov. 2018.
- [71] X. Liang, X. Du, G. Wang, and Z. Han, "Deep reinforcement learning for traffic light control in vehicular networks," 2018, *arXiv:1803.11115*. [Online]. Available: <http://arxiv.org/abs/1803.11115>
- [72] K. L. Tan, S. Poddar, S. Sarkar, and A. Sharma, "Deep reinforcement learning for adaptive traffic signal control," in *Proc. ASME Dyn. Syst. Control Conf. Amer. Soc. Mech. Eng. Digital Collection*, 2019, Art. no. V003T18A006.
- [73] S. Wang, X. Xie, K. Huang, J. Zeng, and Z. Cai, "Deep reinforcement learning-based traffic signal control using high-resolution event-based data," *Entropy*, vol. 21, no. 8, p. 744, Jul. 2019.
- [74] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," in *Proc. Learn., Inference Control Multi-Agent Syst. (NIPS)*, 2016.
- [75] F. Rasheed, K.-L.-A. Yau, and Y.-C. Low, "Deep reinforcement learning for traffic signal control under disturbances: A case study on sunway city, malaysia," *Future Gener. Comput. Syst.*, vol. 109, pp. 431–445, Aug. 2020.
- [76] E. van der Pol, "Deep reinforcement learning for coordination in traffic light control," M.S. thesis, Dept. Sci., Univ. Amsterdam, Amsterdam, The Netherlands, 2016.
- [77] Y. Gong, M. Abdel-Aty, Q. Cai, and M. S. Rahman, "Decentralized network level adaptive signal control by multi-agent deep reinforcement learning," *Transp. Res. Interdiscipl. Perspect.*, vol. 1, Jun. 2019, Art. no. 100020.
- [78] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 2496–2505.
- [79] T. Chu, J. Wang, L. Codeca, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020.
- [80] Luttinen, R. Tapio, *Statistical Analysis of Vehicle Time Headways*. Espoo, U.K.: Helsinki Univ. Technology, 1996.
- [81] K. Nagel and M. Schreckenberg, "A cellular automaton model for free-way traffic," *J. de Phys. I*, vol. 2, no. 12, pp. 2221–2229, Dec. 1992.
- [82] C. K. Keong, "The GLIDE system—Singapore's urban traffic control system," *Transp. Rev.*, vol. 13, no. 4, no. 1993, pp. 295–305.
- [83] D. I. Robertson and R. D. Bretherton, "Optimizing networks of traffic signals in real time—The SCOOT method," *IEEE Trans. Veh. Technol.*, vol. 40, no. 1, pp. 11–15, Feb. 1991.
- [84] A. G. Sims and K. W. Dobinson, "The sydney coordinated adaptive traffic (SCAT) system philosophy and benefits," *IEEE Trans. Veh. Technol.*, vol. 29, no. 2, pp. 130–137, May 1980.

- [85] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," 2015, *arXiv:1509.06461*. [Online]. Available: <https://arxiv.org/abs/1509.06461>
- [86] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," 2015, *arXiv:1511.06581*. [Online]. Available: <http://arxiv.org/abs/1511.06581>
- [87] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 8, no. 9, pp. 1735–1780, 1997.
- [88] J. R. Kok and N. Vlassis, "Using the max-plus algorithm for multiagent decision making in coordination graphs," in *Robot Soccer World Cup*. Berlin, Germany: Springer, 2005, pp. 1–12.
- [89] F. A. Oliehoek and S. W. M. T. Spaan, "Approximate solutions for factored Dec-POMDPs with many agents," in *Proc. AAMAS*, 2013, pp. 563–570.
- [90] S. Krauß, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics," Ph.D. dissertation, Hauptabteilung Mobilität Systemtechnik, Inst. Transp. Res., Porz, Germany, 1998.
- [91] J. Hu and P. Wellman, "Multiagent reinforcement learning: Theoretical framework and an algorithm," in *Proc. Int. Conf. Mach. Learn.*, vol. 98. Madison, WI, USA, 1998, pp. 242–250.
- [92] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," 2018, *arXiv:1802.08757*. [Online]. Available: <http://arxiv.org/abs/1802.08757>
- [93] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," 2018, *arXiv:1802.05438*. [Online]. Available: <http://arxiv.org/abs/1802.05438>
- [94] J. Yang, A. Nakhaei, D. Isele, K. Fujimura, and H. Zha, "CM3: Cooperative multi-goal multi-stage multi-agent reinforcement learning," 2018, *arXiv:1809.05188*. [Online]. Available: <http://arxiv.org/abs/1809.05188>
- [95] J. Foerster, I. A. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2137–2145.
- [96] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*. [Online]. Available: <http://arxiv.org/abs/1511.05952>
- [97] L. Bu, R. Babu, and B. De Schutter, "A comprehensive survey of multi-agent reinforcement learning," *IEEE Trans. Syst., Man, Cybern., C (Appl. Rev.)*, vol. 38, no. 2, pp. 156–172, Mar. 2008.
- [98] X. Wu and H. X. Liu, "Using high-resolution event-based data for traffic modeling and control: An overview," *Transp. Res. C, Emerg. Technol.*, vol. 42, pp. 28–43, May 2014.
- [99] A. Vidali, L. Crociani, G. Vizzari, and S. Bandini, "A deep reinforcement learning approach to adaptive traffic lights management," in *Proc. Workshop 'From Objects Agents'*, 2019, pp. 42–50.
- [100] D. Stathakis, "How many hidden layers and nodes?" *Int. J. Remote Sens.*, vol. 30, no. 8, pp. 2133–2147, Apr. 2009.
- [101] M. Papageorgiou, "Some remarks on macroscopic traffic flow modelling," *Transp. Res. Part A: Policy Pract.*, vol. 32, no. 5, pp. 323–329, Sep. 1998.
- [102] P. Hidas, "Modelling lane changing and merging in microscopic traffic simulation," *Transp. Res. Part C: Emerg. Technol.*, vol. 10, nos. 5–6, pp. 351–371, Oct. 2002.
- [103] D. Krajzewicz, G. Hertkorn, C. Rossel, and P. Wagner, "SUMO (Simulation of Urban MObility)-an open-source traffic simulation," in *Proc. 4th Middle East Symp. Simulation Modeling (MESM)*, 2002, pp. 183–187.
- [104] Behrisch, Michael, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz, "SUMO-simulation of urban mobility: An overview," in *Proc. SIMUL 3rd Int. Conf. Adv. Syst. Simulation*, ThinkMind, 2011.
- [105] G. D. B. Cameron and G. I. D. Duncan, "PARAMICS—Parallel microscopic simulation of road traffic," *J. Supercomput.*, vol. 10, no. 1, pp. 25–53, 1996.
- [106] Smith, Mark, Gordon Duncan, and Stephen Druitt, "PARAMICS: Microscopic traffic simulation for congestion management," in *Proc. IEE Colloq. Dyn. Control Strategic Inter-Urban Road Netw.*, 1995, p. 8.
- [107] M. Fellendorf and P. Vortisch, "Microscopic traffic flow simulator VIS-SIM," in *Fundamentals of Traffic Simulation*, New York, NY, USA: Springer, 2010, pp. 63–93.
- [108] M. Fellendorf, "VISSIM: A microscopic simulation tool to evaluate actuated signal control including bus priority," in *Proc. 64th Inst. Transp. Eng. Annu. Meeting*, vol. 32. Dallas, TX, USA: Springer, 1994, pp. 1–9.
- [109] J. Barcelo, and J. Casas, "Dynamic network simulation with AIMSUN," in *Simulation Approaches in Transportation Analysis*, Boston, MA, USA: Springer, 2005, pp. 57–98.
- [110] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday, "Traffic simulation with aimsun," in *Fundamentals of Traffic Simulation*. New York, NY, USA: Springer, 2010, pp. 173–232.
- [111] C. Bettstetter and S. König, "On the message and time complexity of a distributed mobility-adaptive clustering algorithm in wireless ad hoc networks," in *Proc. 4th Eur. Wireless Conf.*, 2002, pp. 128–134.
- [112] M. R. Heinen, A. L. C. Bazzan, and P. M. Engel, "Dealing with continuous-state reinforcement learning for intelligent control of traffic signals," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2011, pp. 890–895.
- [113] V. R. Konda and N. John Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.
- [114] X. Chu and H. Ye, "Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning," 2017, *arXiv:1710.00336*. [Online]. Available: <http://arxiv.org/abs/1710.00336>
- [115] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: <http://arxiv.org/abs/1707.06347>
- [116] S. Zhen Gou and Y. Liu, "DQN with model-based exploration: Efficient learning on environments with sparse rewards," 2019, *arXiv:1903.09295*. [Online]. Available: <http://arxiv.org/abs/1903.09295>



M.S. degree from 2018 to 2020.

FAIZAN RASHEED received the B.S. degree in electronics from Isra University, Pakistan. He is currently pursuing the M.S. degree in computer science with Sunway University, Malaysia, under the joint programme of Sunway University and Lancaster University, U.K. His research interests include domain of intelligent transportation systems, machine learning, artificial intelligence, and robotics. He was a recipient of the Jeffery Cheah Foundation Research Scholarship to pursue the



KOK-LIM ALVIN YAU (Senior Member, IEEE) received the B.Eng. degree (Hons.) in electrical and electronics engineering from Universiti Teknologi Petronas, Malaysia, in 2005, the M.Sc. degree in electrical engineering from the National University of Singapore, in 2007, and the Ph.D. degree in network engineering from the Victoria University of Wellington, New Zealand, in 2010. He is currently a Professor with the Department of Computing and Information Systems, Sunway University. He is also a Researcher, a Lecturer, and a Consultant in 5G, cognitive radio, wireless networks, applied artificial intelligence, and reinforcement learning. He also serves as a TPC member and a reviewer for major international conferences, including ICC, VTC, LCN, GLOBECOM, and AINA. He was a recipient of the 2007 Professional Engineer Board of Singapore Gold Medal for being the best graduate of the M.Sc. degree from 2006 to 2007. He has served as the Vice General Co-Chair for the ICOIN'19, the General Co-Chair for the IET ICFNA'14, and the Co-Chair of the Organizing Committee for the IET ICWCA'12. He also serves as an Associate Editor for IEEE Access, an Editor for the *KSII Transactions on Internet and Information Systems*, a Guest Editor for the Special Issues of IEEE Access, *IET Networks*, *IEEE Computational Intelligence Magazine*, and the *Journal of Ambient Intelligence and Humanized Computing* Springer, and a regular reviewer for over 20 journals, including the IEEE journals and magazines, the *Ad Hoc Networks*, the *IET Communications*, and others.



RAFIDAH MD. NOOR (Member, IEEE) received the bachelor's degree in information technology (BIT) from University Utara Malaysia, in 1998, the M.Sc. degree in computer science from University Technology Malaysia, in 2000, and the Ph.D. degree in computing from Lancaster University, U.K., in 2010. She is currently an Associate Professor with the Department of Computer System and Technology, Faculty of Computer Science and Information Technology,

University of Malaya. Her research interests are related to a field of transportation system in computer science research domain. Her research interests include vehicular networks, network mobility, quality of service, and quality of experience. She has several collaborators from China, Taiwan, South Korea, France, Australia, and U.K., who are willing to support in providing excellent research outputs.



CELIMUGE WU (Senior Member, IEEE) received the M.E. degree from the Beijing Institute of Technology, China, in 2006, and the Ph.D. degree from The University of Electro-Communications, Japan, in 2010. He is currently an Associate Professor with the Graduate School of Informatics and Engineering, The University of Electro-Communications. His current research interests include vehicular ad hoc networks, sensor networks, intelligent transport systems, the IoT,

and mobile cloud computing. He is/has been a TPC Co-Chair of Wireless Days 2019 and ICT-DM 2018, and a track Co-Chair of many international conferences, including ICCCN 2019 and IEEE PIMRC 2016. He is/has been serving as an Associate Editor for IEEE ACCESS, *IEICE Transactions on Communications*, *International Journal of Distributed Sensor Networks*, and *MDPI Sensors*, and a Guest Editor for IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE, *IEEE Computational Intelligence Magazine*, and *ACM/Springer Mobile Networks & Applications (MONET)*.



YEH-CHING LOW (Member, IEEE) received the B.Sc. (Hons.), M.Sc., and Ph.D. degrees in statistics from the University of Malaya, in 2004, 2007, and 2016, respectively. She is currently a Senior Lecturer with the Department of Computing and Information Systems, School of Science and Technology, Sunway University, Malaysia. Her research interests include count data analysis, Monte Carlo methods, and applications of statistical inference and probabilistic models, Bayesian

inference, and statistics education. She also serves as a reviewer for international conferences such as the ISI World Statistics Congress 2019 as well as for journal such as *Computers & Industrial Engineering*. She is also a member of Association for Computing Machinery and International Statistical Institute.

...