Deep Q-Learning in Traffic Signal Control: A Literature Review

Chantal Schneider [a], 4930835

[a] Delft University of Technology – Faculty of Technology, Policy and Management – Jaffalaan 5, 2528 BX Delft

ARTICLE INFO

Keywords:
traffic signal control,
reinforcement learning,
deep Q-learning,
mixed traffic,
connected vehicles,
intelligent intersection management

Abbreviations:

RL	Reinforcement learning
TSC	Traffic signal control
DQN	Deep Q-learning/Q-network

ABSTRACT

Traffic congestion is a complex problem with far-reaching economic, environmental and societal impacts. Of particular interest are urban intersections, which have become traffic bottlenecks due to badly timed traffic signal plans. In recent years, intelligent traffic signal control based on connected vehicle technology is on the rise. Particularly reinforcement learning (RL) approaches are seen as a promising method to adaptively control intersections in real-time. RL for traffic signal control is a fast-developing field, with many new traffic controllers being proposed rapidly. This paper conducts a systematic literature review on the state-of-the-art in traffic signal control (TSC) for the most commonly applied RL method: deep Q-learning. The review discusses different design choices researchers have to make when designing a new reinforcement learning-based traffic signal controller. Specifically, the paper discusses environment specifications (network topologies, traffic movements and phases), agent specifications (states, actions, rewards, model extensions), and other specifications (traffic generation, performance indicators, base cases). Furthermore, the review identifies open research gaps and makes suggestions for future research. The paper concludes that much of the design process of current RL-based traffic signal controllers is trial-and-error based, without clear guidelines or best practices regarding agent design. To facilitate better systematic comparisons between proposed agents, it is highly suggested to develop standardized environments with standardized scenarios in order to benchmark controllers. This will allow for faster development and evaluation.

1. Introduction

Traffic congestion has negative economic, environmental and social impacts. In 2017, US commuters aggregated a national travel delay of 8.8 billion hours, which translated to a congestion cost of 179 billion dollars (Schrank, Eisele, & Lomax, 2019). 3.3 billion gallons of fuel were wasted in traffic jams, which increased air pollution and CO2 emissions. Furthermore, higher congestion levels were linked to higher accident frequencies (Chang & Xiang, 2003; Marchesini & Weijermars, 2010), leading to ten-thousands traffic-related fatalities (Florin & Olariu, 2015). Since traffic volumes are ever-increasing, congestion has increased over time and is expected to worsen by another 60% until 2030 (Rafter, Anvari, & Box, 2018).

One of the causes of urban congestion are badly timed traffic signal controllers (Du, Shangguan, Rong, & Chai, 2019). In the US alone, traffic signals were estimated to cause 5-10% of all traffic delays or 295 million vehicle-hours (Denney, Curtis, & Olson, 2012). To improve traffic signal plans, research

efforts have been made to create self-adaptive traffic controllers, i.e. controllers which adapt their signal plans in real-time to the current traffic demand (Jing, Huang, & Chen, 2017).

To observe the current traffic situation, connected vehicle technology has been proposed to be used (L. Chen & Englund, 2016). Vehicle connectivity enables vehicles to communicate vehicle trajectory information (e.g. speed, position, planned route), as well as information on road and traffic conditions in real-time with the traffic signal controller or other vehicles. The controller can gather data from all vehicles around the intersection and use this to create optimal signal plans.

An approach in traffic signal control (TSC) which has gained popularity quickly in recent years is reinforcement learning (RL). RL has been shown to be an effective method to increase traffic performance compared to traditional TSC. Wang, Yang, Liang and Liu (2018) reviewed different methods to control self-adaptive TSC systems and concluded that TSC based on RL will likely become the future of TSC. Compared to other approaches used in intelligent TSC (e.g. optimization-based, rule-based, hybrid, other machine learning-based approaches), RL has several advantages: (1) RL algorithms are model-free approaches, i.e. they do not require a human to create a precise mathematical model of the system environment, making them much less complex to build (2) RL algorithms can be used in real-time, which is a requirement for self-adaptive TSC (3) RL models are adaptive, i.e. they can adapt to dynamically changing traffic situations.

The first RL traffic controller was already proposed in 1994, however at that time it could not yet be implemented due to computational limitations (Mikami & Kakazu, 1994). The first adaptive controller using RL was proposed in 2003 (Abdulhai, Pringle, & Karakoulas, 2003). Yet, early RL controllers were limited by the fact that RL models could not be applied to very complex problems due to the need for discretized states. Only with the introduction of deep RL by Mnih et al. in 2015 were models able to learn increasingly complex relations. Since then, RL for TSC became increasingly popular, resulting in many newly proposed RL controllers in literature (P. Chen, Zhu, & Lu, 2019).

Since the field is developing so quickly, existing literature reviews are outdated very quickly. Existing reviews either focus only on deep RL but not on TSC, or they review RL for TSC, but have been published before the rise of deep RL. Regarding non-TSC specific reviews, papers by e.g. Arulkumaran et al. (2017), Y. Li (2018) and Mousavi, Schukat and Howley (2016) were found. Reviews that focus explicitly on TSC have been published by El-Tantawy, Abdulhai and Abdelgawad (2014), Mannion et al. (2016) and Yau et al. (2017). The later three reviews survey the state-of-the-art in RL for TSC until their respective publishing dates, yet none of them include deep RL controllers. However, since these reviews have been released, many new and more promising traffic controllers have been published.

This paper aims to fill this gap by reviewing the state-of-the-art in TSC based on deep RL up to date. Since this field is very vast, this review will focus only on the most popular RL approach in TSC: Deep Q-learning (DQN / deep Q-networks). The goal of this article is to provide a comprehensive overview of the field by detailing the different design decisions researchers have to make when designing a new DQN-based traffic signal controller. Furthermore, this paper aims to identify shortcomings of current literature and to suggest directions for future research. Overall, the following two research question will be answered:

1) **What is the current state-of-the-art in intelligent traffic signal control using deep Q-Learning?**
2) **What limitations exist in current literature and how could future research alleviate them?**

The remainder of this paper is structured as follows: Section 2 briefly introduces the background on deep Q-learning. Section 3 outlines the literature search methodology. Section 4 presents the state-of-the-art in TSC using DQN, sorted by the different types of design decisions. Section 5 discusses the identified literature gaps and suggestions for future research, as well as the limitations of the review. Section 5 concludes the paper.

2. Deep Q-Learning Learning background

This section provides a brief introduction to reinforcement learning, Q-learning and deep Q-learning. For a more extensive overview, please refer to Sutton and Barto (2018).

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a subset of Machine Learning in which models learn to make sequences of decisions within complex and/or uncertain environments.

In RL, decision makers (called *agents*) interact with their surroundings (*environment*): agents select *actions*, and the environment responds to these actions by changing by changing *state* and by returning a *reward*. Agents aim to find the best solution for a problem, without any external supervision or hints from the modeler. Instead, agents initially apply a trial-and-error process to select actions, and solely learn which actions are optimal via the obtained reward. The goal of agents is to maximize their gained rewards. RL algorithms can correlate immediate actions with both immediate and delayed returns. Over time, the algorithm gains experience and can exploit this knowledge to gain higher rewards.

Formally, RL problems are generally modeled as Markov Decision Processes (MDP). An MDP can be defined as a four-tuple $< S, A, R, T >$, where $S = \{s_1, \dots, s_n\}$ is a finite set of states and $A = \{a_1, \dots, a_m\}$ is a finite set of actions. The function $T: S\ x\ A\ x\ S \rightarrow [0,1]$ defines the transition function specifying the probability of taking action a in state s and ending up in state s'. The reward for taking action a in state s and ending up in state s' is represented by reward function $R: S\ x\ A\ x\ S \rightarrow\ \mathbb{R}$.

The system fulfills the Markov property if the outcome of an action only depends on the previous state and action, such that

$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots) = (s_{t+1}|s_t, a_t, r_t) \quad (1)$$

To determine which actions agents will choose when in a certain state, they use policies. Policy π maps from states to actions: $\pi: S \rightarrow A$. This means that if the agent is in state s_0 and acts under policy π, it will choose action $a_0 = \pi(s_0)$ and will transition to state s_1. For this transition, the agent will receive reward $r_0 = R(s_0, a_0, s_1)$.

Rewards that are received are a combination of immediate returns and delayed returns. Delayed returns are discounted by a discount factor $\gamma\ \in [0,1]$. The agent's goal is to maximize its expected reward over time. Usually, it should give preference to short-term (immediate) rewards over long-term (delayed) rewards. The goal of the agent is to maximize its total return. The total return, the expected discounted cumulative reward over time is calculated as:

$$R_t\ = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots = E[\sum_{k=0}^{\infty} \gamma^k r_{t+k}] \quad (2)$$

Two different value functions can be defined: a state value function and a state-action value function. The value of state s under policy π is $V^{\pi}(s)$. It represents the expected return of following policy π when starting in state s. It is defined as

$$V^{\pi}(s) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k} | s_t = s] = E_{\pi}[r_t + \gamma V^{\pi}(s_{t+1}) | s_t = s]$$
$$= \sum_{s' \in S} T(s, \pi(s), s') \left(R(s, a, s') + \gamma V^{\pi}(s')\right) \quad (3)$$

where s' denotes the next state. This equation is also known as the Bellman equation (Sutton & Barto, 2018). It describes the relationship between the value of state s and its successor states.

Furthermore, a state-action value function $Q^{\pi}(s, a)$ can be defined. It shows the expected value of taking action a while the agent is in state s while following policy π. In other words, it is a measure of how good the state-action pair is. It is defined as

$$Q^{\pi}(s, a) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k} | s_t = s, a_t = a]$$
$$= \sum_{s' \in S} T(s, \pi(s), s') \left(R(s, a, s') + \gamma V^{\pi}(s')\right) \quad (4)$$

An optimal policy π^* (i.e. a policy that results in expected values equal or greater than any other policies for all states) satisfies

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma V^*(s')\right) \quad (5)$$

$$Q^*(s, a) = \sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma \max_{a' \in A} Q^*(s', a')\right) \quad (6)$$

If T and R are known, the optimal policy can be found e.g. via dynamic programming methods that use the recursion in equation (5). These approaches are known as model-based approaches. These approaches do not require the agent to interact with the environment directly (Y. Li, 2018). However, in many cases the system is too complex to be able to determine T and R upfront. In these cases, RL algorithms can be applied. These model-free approaches rely on the earlier described trial-and-error processes to sample the underlying MDP to learn the optimal policies.

We can express the optimal value function and policy solely in terms of the Q-function:

$$V^*(s) = \max_{a \in A} Q^*(s, a) \quad (7)$$

$$\pi^*(s) = \arg\max_{a \in A} Q^*(s, a) \quad (8)$$

Now, we do not need to know the transition function T or the reward R explicitly. Instead, the Q-function is sufficient to determine the next action. This Q-function is iteratively estimated and updated by the RL algorithm.

2.2 Tabular Q-learning

Traditional Q-learning uses a lookup table of all possible Q-values of state-action pairs and iteratively updates the Q-values. The updates at each time step t are performed using the following equation:

$$\underbrace{Q_{t+1}(s_t,a_t)}_{\text{new Q-value}} = \underbrace{Q_t(s_t,a_t)}_{\text{old Q-value}} + \underbrace{\alpha}_{\text{learning rate}} \overbrace{\left[\underbrace{r_t + \gamma \underbrace{\max_{a' \in A} Q_t(s_{t+1},a)}_{\text{estimate of optimal future Q-value}}}_{\text{new value (temporal difference target)}} - \underbrace{Q_t(s_t,a_t)}_{\text{old Q-value}}\right]}^{\text{temporal difference}} \quad (9)$$

It can be seen that the algorithm uses the Bellman equation for the value function update, by weighting the old Q-value and the temporal difference. The learning rate (or step size) $\alpha \in [0,1]$ determines the speed that new information is being learned. A learning rate of 0 means the algorithm learns nothing, while a value of 1 means that only the most recent information is considered to choose new actions (and thus forgetting old learned knowledge).

2.3 Function approximation

A problem with tabular Q-learning is the so-called "curse of dimensionality": if the number of state-action pairs becomes too large or even unlimited (for continuous states) problems arise. Too much storage space would be needed to store every state-action pair and the computational costs and learning times would increase exponentially (Arulkumaran et al., 2017; Yau et al., 2017). Tabular Q-learning works well for small problems, however in the real-world state-action spaces are rarely small or finite.

To solve this, function approximation can be applied. It is used to approximates the value of a function and allows us to generalize experiences to other similar (potentially unencountered) state-action pairs. In the case of Q-learning, function approximation can be used to generalize learned Q-values of state-action pairs to predict similar state-action pairs. As such, the algorithm is likely to learn Q-values faster.

Instead of having to store every state-action pair Q-value in a table, the learned function $Q(s,a)$ will be parameterized by a learned weight θ. One method to approximate the weights is gradient descent. In gradient descent, θ is updated by minimizing the mean squared error (MSE) between the current $Q(s,a)$ estimate and the true $Q^\pi(s,a)$ estimate under policy π. The update is done as follows:

$$MSE(\theta) = \sum_{s \in S} P(s)[\underbrace{Q^\pi(s,a,\theta^*)}_{\text{target Q}} - \underbrace{Q_t(s,a,\theta_t)}_{\text{current Q estimate}}]^2 \quad (10)$$

$$\frac{\partial}{\partial \theta_t} MSE(\theta) = 2[Q^\pi(s,a,\theta^*) - Q_t(s,a,\theta_t)] \frac{\partial}{\partial \theta_t} Q_t(s,a,\theta_t) \quad (11)$$

Where $P(s)$ is the sampling distribution (i.e. the probability to visit state s under policy π).

Since the target Q^π is not known, we estimate it by using the reward in the current time step and a discounted estimate of the next state's best Q-value using the current Q_t estimate.

$$Q^\pi(s,a,\theta^*) \approx r_t + \gamma \left[\max_{a' \in A} Q_t(s_{t+1},a',\theta_t)\right] \quad (12)$$

Since it uses the max-operator, equation (11) is an optimistic estimate of the Q-value at time step t.

2.4 Deep Q-learning

In deep Q-learning, deep neural networks (NN) are used to approximate the Q-values (also called DQN: deep Q-networks). NNs are universal approximators that can approximate complex and highly non-linear functions. Deep NNs are NNs which have multiple hidden layers. For an introduction to neural networks, refer to Bishop (1996).

The idea is that NNs are used to update the Q-values, similar as in equation (9). To update the NN's weights needed to approximate the Q-functions, commonly gradient descent and backpropagation are used. To use this, we need a cost function that measures the difference between the NN's estimated Q-value and the actual Q-value. The goal is to minimize this error function. Since the actual Q-value is unknown, we can again estimate it by using the temporal difference target of equation (9). It represents the total expected reward of all future time steps, including discounted future rewards. Then we can iteratively update the target value.

Thus, we can set up our loss function as follows, using the squared error loss:

$$L = \frac{1}{2}\left[Q(s_t, a_t) - \left(r_t + \gamma V(s_{t+1})\right)\right] \tag{13}$$

Now we can perform gradient descent using the derivative of the loss function:

$$\frac{\partial L}{\partial Q(s_t, a_t)} = Q(s_t, a_t) - \left(r_t + \gamma V(s_{t+1})\right) \tag{14}$$

Using this, an update rule for the Q-value can be created:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) - \alpha\left[Q(s_t, a_t) - \left(r_t + \gamma V(s_{t+1})\right)\right] \tag{15}$$

Where the best estimate for the value of the next state is the value of the expected best action in that state:

$$V(s) = \max_{a \in A} Q_{target}(s, a) \tag{16}$$

In deep Q-learning, many extensions have been proposed that improve the stability and performance of the model. The most commonly applied ones are target network freezing and experience replay (Mnih et al., 2015). Recent research is experimenting with more complex extensions, such as the Rainbow extensions (Hessel et al., 2018) or recurrent networks (Hochreiter & Schmidhuber, 1997).

3. Literature search method

A systematic review of existing literature was conducted through the databases Scopus, IEEE and Google Scholar. The goal was to exhaustively find papers that discuss applying DQN in TSC. The search terms used for each search are shown in Table 1.

Table 1 Search terms used in the literature review on RL approaches in TSC

Key concept	Search terms used
Traffic signal control	Intersection, "traffic light", "traffic signal", "traffic signal control", signal*
Road traffic	Vehicle, car
Reinforcement learning	reinforcement learning

Deep Q-learning	DQN, “deep reinforcement learning”, deep Q-learning

All searches were performed on the title, abstract and keywords. To limit results from other transport domains, the keywords “UAV”, “unmanned aerial vehicle”, “aerial”, “underwater” and “air” were excluded. In each case, articles were filtered first by their titles and by screening the abstracts. For the relevant results, the full-body text was used. Lastly, for the selected literature the references inside the paper were reviewed to find additional papers (‘backward snowballing’).

Only papers that focused on isolated signalized intersection control were included. This excluded studies that focused on non-signalized intersections, on-ramp merging or AV trajectory control. It also excluded studies that focused on multi-agent control in networks of intersections. Network control was excluded to narrow the scope of the review.

Furthermore, studies that included pedestrians, lane changes or traffic rule violations were considered out of the scope of this review, since their focus is on human behavior rather than traffic efficiency. Other articles that were out of the scope were papers on pedestrian or human driving prediction algorithms, cross-modal learning and human-vehicle interactions.

To guarantee the quality of included articles, all included papers must describe the implemented RL algorithm and methodology (experimental setup, list of parameters) and must provide quantitative experimental results.

A full list of all included papers, as well as the results can be found in the attached excel file. Due to page size limitations, it could not be included in this paper.

4. State-of-the-art of Deep Q-learning in traffic signal control

Section 2 provided a brief introduction to deep Q-learning. To implement a new DQN agent for TSC, the modeler will have to design the environment (i.e. the traffic system) and the agent itself. Furthermore, several decisions regarding the evaluation of the agents have to be made. This section will discuss each of these components separately. Note that all discussions assume that traffic drives on the right side of the road, but conclusions can easily be adjusted to left-side driving.

4.1 Environment specification

4.1.1 Network topologies

Many different types of network topologies can be chosen for the traffic model. In general, it is possible to study either isolated intersections, multiple intersections in an arterial or a network of intersections. Furthermore, it is possible to study either synthetic topologies or real-world topologies (Yau et al., 2017). Synthetic layouts allow for better control of the environment, but real-world intersections allow to better adapt the TSC to the specific requirements for that intersection.

In the reviewed literature, many papers focused on isolated intersections. All of these papers used synthetic intersection layouts of bidirectional 4-way intersections. Researchers chose different numbers of lanes for the legs: between 1 and 4. Recently, more interest is being shown in studying coordinated signal control in arterial or networks.

4.1.2 Possible movements and traffic phases

When looking at the single intersection level, the modeler must decide which movements are permitted for vehicles and how traffic phases should be assigned (Yau et al., 2017). Since nearly all papers assumed 4-way intersections, the discussion will only be based on these.

Concerning the allowed movements, the following options were found:

- only through traffic (i.e. driving straight without turns)
- through traffic and turns
- movements as in the real-world (usually includes through traffic and turns, but could also include U-turns or one-way streets).

The possible traffic (green) phases are related to this. The options found here are either two traffic phases, 4 traffic phases or more than 4 traffic phases.

Some researchers decided to not allow turns. (e.g. P. Chen et al., 2019; L. Li, Lv, & Wang, 2016; Mousavi, Schukat, & Howley, 2017; Nawar, Fares, & Al-Sammak, 2019; Pol & Oliehoek, 2016; Wei, Zheng, Yao, & Li, 2018; Wu, Kong, & Fan, 2019; R. Zhang et al., 2020). In these models, only two phases were used: North-South green and East-West green. This was done to simplify the model; however, it reduced the models' real-world applicability. Often these models were meant as proofs-of-concept for newly proposed ideas.

The other popular choice was to use 4 green phases. While other green phase options would be possible (see (Yau et al., 2017)), the common choice was to use the option shown in Figure 1. Two traffic phases allow through and right-turning traffic and two traffic phases allow left-turning traffic.

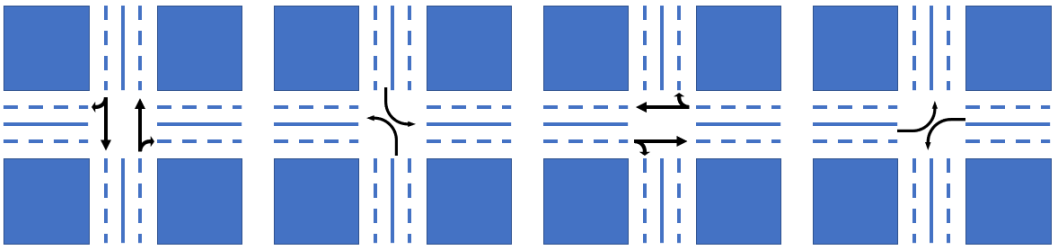

Figure 1 Commonly chosen combination of green phases

A few papers even added as many as 8 phases of compatible movements, see e.g. (Guo, Wang, Chan, & Askary, 2019; Kim, Jung, Kim, & Lee, 2019).

Ultimately, the choice of allowed traffic movements and green signals will influence what kind of intersections the model will be applicable for. In practice, the choice of intersection will depend on the local traffic situation. Nevertheless, adding more possible green phases to the model can make the controller more flexible, but it will also become harder to train the agent. A trade-off will have to be made.

4.1.2.1 Additional phase constraints

Many modelers add transitions between phases. Some researchers add a few seconds of yellow time after switching phases (e.g. S. Wang et al., 2019; Wei et al., 2018). Additionally, some models also include a few seconds of all-red phases during transitions (e.g. Coşkun et al., 2019). Both yellow and

red phases allow vehicles to clear the intersection, and thus increase safety. It also makes the model more similar to real-life.

Some researchers added extra logic to the allowed phases, a so-called "sanity check" (Zhang et al., 2020). This involves adding a minimum and/or maximum duration for green phases. This means the controller may not switch phase until the minimum green duration has passed and must switch phase if the maximum duration was reached. Minimum phase lengths are implemented to ensure that at least one vehicle can pass during a green phase. Maximum phase lengths are implemented to ensure fairness between lanes since some reward functions could lead to shorter queues becoming less prioritized and having to wait infinitely long compared to longer queues. Other authors however argue that imposing minimum and maximum durations a priori is not needed since the algorithm would develop such policies by itself if necessary (Genders & Razavi, 2018).

4.2 Agent specifications

4.2.1 States

The state representation is an important input for RL models. The state represents the agent's decision-making factors, i.e. it represents everything the agent knows about the current situation at time t and the agent must decide which action to take based only on this knowledge. Within the literature, many different state representations were found. States can either be singular or consist of several different sub-states (i.e. measures for different factors combined into one overall state) (Yau et al., 2017). Additionally, the state of one intersection may include information about other surrounding intersections. The following (sub)state representations were found in DQN literature:

- **Queue size:** Number of vehicles waiting/halting within a leg or lane. To determine which vehicles are waiting, a maximum speed has to be defined (e.g. 0.1 m/s). All vehicles below this speed are counted as waiting in the queue, all others are not. Used e.g. by M. Guo et al. (2019) and L. Li et al. (2016).
- **Vehicle position:** Physical position of vehicles within the intersection. Represented via discrete traffic state encoding (DTSE). In DTSE, lane segments of length l beginning at the stop line are split into discrete cells of length c (Genders & Razavi, 2016). All cells are combined into a state vector or matrix. For DTSE, the size of the cell is critical: if it is too long, the individual vehicle dynamics are lost, but if it is too short, the computational cost will increase drastically. Generally, the cell length is chosen slightly larger than the average car. In the case of vehicle positions, binary DTSE is used: if a vehicle is inside the cell, the cell will be set to 1, else it will be 0.
- **Distance to the nearest vehicle at each approach:** Distance [m] from the stop line to the closest vehicle that is driving towards the intersection for each leg. If no vehicle is approaching on a leg, the maximum distance is used. Representation used by R. Zhang et al. (2020).
- **Vehicle density:** Can be measured per leg (e.g. R. Zhang et al., 2020), per lane (e.g. Genders & Razavi, 2016) or via DTSE (Zeng, Hu, & Zhang, 2018). Is encoded either as an absolute number of vehicles on the leg/lane, as a number of vehicles per area or as $\frac{\#vehicles}{\max \#vehicles}$.
- **Speed.** Speed of vehicles at time t [m/s]. Can be either measured as average speed for the whole leg or lane (e.g. Genders & Razavi, 2018) or represented via DTSE per vehicle (e.g. T. Zhao & Wang, 2019). Speeds can be absolute or normalized with respect to the maximum allowed speed.

- **Delay.** Delay [s] for vehicles from entering the intersection to leaving the intersection (i.e. crossing the stop line). Measured as the difference between the time needed to cross the intersection at the maximum allowed speed and the time needed to actually cross the intersection. Delay can be measured per vehicle or as average per leg over time. Used for example in (Kim et al., 2019).
- **Vehicle waiting time.** Similar to vehicle delay. Measures the time [s] that vehicles have spent under a certain speed threshold (e.g. under 0.1 m/s). The waiting time per vehicle can either be reset to 0 every time the vehicle starts moving or kept as a cumulative value. Nawar et al. (2019) for example used the vehicle waiting time normalized to the maximum waiting time in the network.
- **Vehicle emissions.** CO2, NOx or other emissions created by vehicles. Used e.g. by Kim et al. (2019).
- **Red/green/yellow timing:** elapsed time [s] since the beginning of the red/green/yellow phase for a specific leg or lane. See for example in (Shabestary & Abdulhai, 2019).
- **Yellow phase indicator:** binary value which is set to 1 if there currently is a yellow phase, 0 if not. See (Zhang et al., 2020).
- **Current traffic phase:** Combination of green signals currently activated. Within the state representation, phases can be encoded differently. Option 1 is to simply assign numbers to traffic phases (phase 1, phase 2, phase 3, …) and to simply use the phase number for the state. Option 2 is to one-hot encode phases. Here a vector with the same length as the number of traffic phases is created. Every entry in this vector is set to 0, except the current phase which is set to 1 (see e.g. (Zeng et al., 2018)). Option 3 is to use DTSE and to encode per lane (e.g. Nawar et al., 2019) or for every cell if they currently have a green phase (1) or not (0) (e.g. Genders & Razavi, 2016). Option 4 is to use +/- signs in another indicator. R. Zhang et al. (2020) for example encoded the current phase by making the detected car count a positive number if that leg had a green phase, and a negative number if it had a red phase. Apart from encoding the current phase, in some cases, researchers also encode the next phase (e.g. Wei et al., 2018). Note that this is only possible if the traffic phases are in a fixed order.
- **Current time.** Elapsed time [s; min; h; d; m] since a specified time. Different representations are possible, such as encoding the current hour of the day, day in the week, or month of the year. R. Zhang et al. (2020) for example encoded the current time in hours since midnight, discretized into 24 steps of one hour.
- **Raw pixel snapshot.** Camera snapshot of the full traffic situation at the current time. Uses all pixels as input. Used for example by Mousavi et al. (2017).

As can be seen, there are many different possibilities for creating a state representation. Earlier RL methods such as tabular Q-learning only allowed for lower dimensional state representations, and usually required a certain amount of discretization. The most common choices in state representations for tabular Q-learning were queue length, delay or flow rates (Genders & Razavi, 2016; Mannion et al., 2016; Yau et al., 2017). No data at the individual vehicle level could be included. The developments of DQN now allow researchers to include rich high-dimensional data (Liang, Du, Wang, & Han, 2018). Deep Q-learning however removed these restrictions, leading to authors proposing very detailed state representations, often at the individual vehicle level, with many authors proposing to use DTSE.

Some authors argue that the controller can make better decisions the more data it has available (Zhang et al., 2020). They argue that if not all information is included in the state, relevant decision factors may be missing, thus the model would not be able to create optimal policies.

Three studies find that using higher-dimensional state representations outperforms RL-algorithms which only use queue size as state. Mousavi et al. (2017) find that using raw pixel inputs from a SUMO snapshot as state leads to a 67-73% reduction in queue length and cumulative delay compared to using a shallow network RL-algorithm that uses queue length as state. Genders & Razavi (2016) also compare a high-dimensional state DQN to a shallow NN RL-algorithm. As state representation they use a combination of a one-hot encoded traffic phase vector with a 2-layered DTSE that includes the vehicle position and speed. They found that the higher-state representation led to an 88% reduction in average cumulative delay, a 66% reduction in queue length, a 20% reduction in average travel time and a similar throughput. Shabestary and Abdulhai (2019) compared DQN using 2-layer DTSE (representing vehicle position and vehicle speed) to tabular Q-learning and found that the higher-dimensional state representation led to a 23.4% reduction in intersection travel time, 39.3% reduction for in queue time, and 36.0% shorter queue lengths with 42.3% fewer variations. As can be seen, higher-dimensional state representations led to significant improvements in various performance indicators compared to only using queue length. Genders & Razavi (2018) and Gao, Shen, Liu, Ito, & Shiratori (2017) speculate that this is because if a model only uses the queue length as state representation, this would ignore all vehicles which are still driving (so they would not be counted as a member of the queue). Shabestary & Abdulhai (2019) provide similar reasoning: they explain that if a controller only uses queue length as input, it cannot distinguish between 100 slow-moving vehicles or an empty lane. Once a lane with a long queue gets a green phase, all these vehicles suddenly start moving slowly. However, since they are now above the cutoff speed to be counted as "in a queue", they now suddenly become "invisible" to the controller. As such, different environmental situations could be represented by the same state representation but gain completely different rewards even if the same actions are picked. This can lead to instability.

Similar reasoning can be used against using average traffic flows. Average traffic flows use historical traffic data to calculate e.g. average speeds or delays over a certain time interval. The problem is that these are just approximations of the current traffic states, and useful information could be left out (Genders & Razavi, 2016). In case of using average travel delays, the problem is that these kinds of metrics can only be gathered once vehicles have left the intersection, thus it is only a delayed representation of the environment. The controller would thus always be slightly behind (Gao et al., 2017).

Additionally, many metrics such as queue length and average vehicle flows cannot be measured directly. As such, they have to be preprocessed and abstracted by experts using prior knowledge (Shabestary & Abdulhai, 2019). To use queue lengths for example, experts must specify a speed threshold. For average flows, assumptions about e.g. vehicle lengths must be made. Such abstractions and discretization leads to a loss of information and can lead to problems.

For these reasons, many authors argue against using discretization or abstraction by experts. Instead, they suggest using high-dimensional states such as raw pixel images or DTSE, so that the deep RL-algorithm can extract and learn the relevant features by itself, without prior knowledge (Gao et al., 2017; Genders & Razavi, 2016; Mousavi et al., 2016).

Other authors however argue that certain state representations would not work in practice since this data would either be impossible to obtain in real-time or it would require too expensive sensors to gather this data (Choe, Baek, Woon, Kong, & Member, 2018; Coşkun et al., 2019; Genders & Razavi, 2018; Horsuwan & Aswakul, 2019; Mousavi et al., 2017; Rodrigues & Azevedo, 2019; S. Wang et al., 2019; T. Wu et al., 2019). Many of these authors then work under the assumption that the traffic environment is only observable via infrastructure sensors, such as induction loops, cameras or radar. However, as stated in chapter 2, the rise in connected vehicles could solve this problem. Using vehicle communication could enable the traffic controller to gather individual vehicle-based real-time traffic information, without the need for expensive infrastructure sensors (Liang et al., 2018; Zhang et al., 2020).

Nevertheless, the fact that higher-dimensional state representations always gain significantly better results is not undisputed in literature. Genders & Razavi (2018) compared three different levels of details in state representations using the same asynchronous advantage actor-critic RL-algorithm with the same rewards, traffic situation, training parameters and model parameters for all three cases. For each of the three state representations, the authors included the one-hot encoded current traffic phase and the time spent in that phase. For the low-resolution state, they used leg occupancy and average leg speed. For the middle-resolution state, lane queue lengths and lane density were used. In the high-resolution state, a 1-layer DTSE of vehicle positions was used. After running the experiments, they found that there was no difference in traffic throughput between the different controllers and only a 9% difference in queue length between the high and the low/medium-state representations. Only the level of delay was significantly affected: the middle- and high-resolution states led to a reduction of the average vehicle delay of 21 and 25% respectively. The authors conclude that in many cases, lower- or medium-resolution state representations may be sufficient. Yet, they mention that the lack of significantly better performance of the high-resolution controller could have been caused by the fact that the experiments only used a shallow NN, rather than a deep network.

We can conclude that it is not completely clear which state representations would be most suitable. In general, researchers have gained very good results when using individual vehicle-based representations (such as position and speed DTSE), especially when in combination with current traffic phase information and elapsed time (Fang, Chen, & Liu, 2019; Gao et al., 2017; Pol & Oliehoek, 2016; Shabestary & Abdulhai, 2019; Wei et al., 2018; Zeng et al., 2018; Zeng, Hu, & Zhang, 2019). Ultimately, adding more information will likely improve the agent's performance (as long as only raw unabstracted data is used), but it will also increase the agent's training time. A trade-off will have to be made.

4.2.2 Actions

The action space represents what changes the traffic controller can make. Within the literature, two types of actions were specified:

- **Pick next traffic phase.** In this action representation, the controller chooses the next traffic phase for a certain number of time steps (usually for the next 1 to 10s). The choice of traffic phases is determined by the modeler (see section 4.1.2). For this action representations, traffic phases are acyclic, meaning they do not appear in any fixed order. For an example, see (Du et al., 2019).
- **Pick traffic phase split.** In this representation, traffic phases happen in a fixed order. The controller can then choose the duration of the current traffic phase. In some models, the

controller can extend the current phase for a certain number of time steps or decide to switch to the next phase (e.g. Zeng et al., 2018). In other models, the duration of a phase is determined at the beginning of the phase and cannot be extended (e.g. Liang et al., 2018). In some cases, the controller may even decide to spend 0 s in a certain traffic phase, thus effectively skipping the phase and making the phases acyclic (e.g. Zeng et al., 2018).

Both types of action representations were found numerous times in literature. Researchers which use the first type argue that these controllers can more dynamically adapt to traffic, since the controller has free choice of all phases, without being constrained by predetermined orders. Proponents of the second type however argue that having a fixed order is more predictable for humans, and thus increases safety (Choe et al., 2018).

Note that in case of models that only have two phases, effectively both types of action representations become the same.

4.2.3 Rewards

Rewards are the measure by which agents determine how good or bad a certain action in a certain state was. As such, rewards can either be positive (rewards) or negative (punishments). The agent chooses actions such that expected rewards are optimized. Like for the state representation, there are many different metrics for rewards in literature. Some authors use only one metric, while others use a weighted reward function consisting of two or more metrics. The following metrics were found in DQN literature:

- **Queue length.** See section 4.2.1 on states. For an example see (L. Li et al., 2016).
- **Vehicle delay.** See section 4.2.1. Used e.g. by R. Zhang et al. (2020).
- **Vehicle travel time.** Similar to delay, but instead of using the difference with the optimal travel time, the actual time to pass the intersection is used. Used for example by Wei et al. (2018).
- **Vehicle waiting time.** See section 4.2.1. Used e.g. by Nawar et al. (2019).
- **Number of waiting vehicles.** Number of vehicles under a certain speed threshold. Can be measured for the full intersection (Zeng et al., 2019) or as the absolute or relative difference between legs (Coşkun et al., 2019).
- **Number of vehicles.** Total number of vehicles in the intersection region, regardless of the vehicles' speeds. See e.g. (Choe et al., 2018).
- **Intersection throughput.** Number of vehicles that pass over the stop line. Used e.g. by Zeng et al. (2019).
- **Comparative performance with fixed time control.** Proposed by Du et al. (2019). See discussion below.
- **Fuel/energy consumption.** Amount of fuel used. Can be used as reward to create a controller that reduces environmental impact (Islam, Aziz, Wang, & Young, 2019).
- **Vehicle emissions.** See section 4.2.1. Used by Fang et al. (2019) to reduce environmental impacts of intersections.
- **Penalty for (emergency) stops.** Negative reward for every vehicle that did an (emergency) stop. See e.g. (Y. Wu, Chen, & Zhu, 2019)
- **Phase change**. Penalizes agent if the phase switches. Used to avoid constant flickering of traffic signals. See e.g. (Zeng et al., 2019).

- **Number of teleports.** Teleports are specific to SUMO and only happen in case of would-be collisions or traffic jams. Used by Pol & Oliehoek (2016).

Other rewards that were mentioned in reviews (Mannion et al., 2016; Yau et al., 2017), but which were used in non-DQN algorithms were: delay incurred during phase transitions, appropriateness of green times, achieving green waves, accident avoidance and speed restrictions.

Note that all rewards mentioned are metrics which the agent wants to reduce. As such, the rewards will have negative values, thus the term punishment may be more suitable.

Different authors have implemented vehicle-based rewards (e.g. queue length, waiting time, delay) in different ways. Some authors use cumulative values, others use averaged value per vehicle, and others use relative values between different legs or lanes. Using averaged values is more intuitive for humans to understand and to evaluate. However, average values provide the controller with no information about the total number of vehicles in the intersection. For example, having 1 or 100 vehicles with an average queue length of 10 vehicles would give the same reward, even if the former is a much better situation than the latter. The argument for using differences between legs is to promote fairness between legs.

Additionally, many rewards can either be implemented as an absolute real value (e.g. the current absolute queue length), a discretized or binary value (e.g. 1 if there was a phase change in the last time step, 0 if not) or as a measure of change between time steps (e.g. change in queue length between time step t and t+1). Using values of change immediately shows the controller whether the action improves (positive reward value) or worsens (negative reward value) the current situation. Yet if agents use the absolute value, they have a sense of the order of magnitude of a reward.

Furthermore, some models use squared rewards. Brys, Pham, & Taylor (2014) for example used the cumulative squared vehicle delay so that fewer large delays are prioritized over many short delays. This not only encourages fairness between road users but also leads to faster learning rates.

Different arguments can be used for and against certain rewards. Like for the state representation, some authors argue that certain metrics are only obtainable in simulators, but not in the real world (Mannion et al., 2016). Yet, as mentioned, connected vehicles will be able to overcome this shortcoming.

The most commonly used reward is delay. This metric is vehicle-based and intuitively represents how much time loss was caused by the traffic signal for a specific vehicle. Since the goal of the controller is to reduce the time that vehicles spend in traffic, this seems a suitable metric. The issue however is that delays can only be measured after a vehicle leaves the intersection. This will cause delayed rewards, so the controller may not be able to assign rewards accurately to actions, thus it will cause slower learning or unstable control. Furthermore, delays may not properly penalize traffic jams. For a two-lane approach for example, a controller would not be able to distinguish between one blocked and one full speed lane vs two lanes at half speed, since they would lead to similar rewards, even though the traffic flows are very different (Pol, 2016). Similar reasoning can be used for using travel time as a reward.

The second most commonly used reward is waiting time. Intuitively longer waiting time implies that there is more congestion, thus it is a suitable metric. The problem with using waiting time is that the controller converges to a policy in which the traffic phase changes every time a new action is chosen.

This is because as mentioned in section 4.2.1, only vehicles under a certain speed threshold are counted as "waiting". If the controller switches very often between phases, the vehicles in the new phase's queue start moving slowly and are no longer counted as "waiting". The controller can thus create policies in which vehicles drive slowly, but hardly truly halt (Pol, 2016). The same problem exists when using the number of waiting vehicles.

A policy that leads to constantly flickering lights will cause vehicles to start and stop frequently. For humans this style of driving is uncomfortable and not desirable (Coşkun et al., 2019). To avoid this, some authors add a penalty for the number of (emergency) stops. Other authors decide to instead use a penalty for phase changes. Phase changes are also undesirable by itself since they require yellow and red transition phases to allow vehicles from one road to stop, before giving a green signal to the new lane. During these transition phases however few or no vehicles can pass the intersection, thus reducing its throughput.

Yet another reward that is frequently used is the intersection throughput. The drawback of throughput is that it does not take into account how long queues are or how long vehicles have been waiting. In over-saturated traffic conditions this could lead to the agent choosing one green only and leaving the other directions red forever. This would lead to unfair results.

As can be seen, different rewards have different advantages and disadvantages, and may be suitable for different objectives or traffic situations. Islam et al. (2019) for example tested three different reward functions: total detected control delay, total detected energy consumption and total detected energy consumption with penalty for stops. They found that reward function 2 provided undesirable results, reward 1 provided better performance in trip delay and reward 3 better energy consumption.

Due to the different strengths and weaknesses of reward functions, authors started to experiment with weighted reward functions. Mannion et al. (2016) for example tested three different reward functions for a tabular Q-learning algorithm: 1. Change in average queue length between time steps, 2. Change in cumulative waiting time between time steps, 3. Weighted reward of 1 and 2. They concluded that reward function 2 performs best under steady traffic flows (i.e. for balanced flow between legs) and reward 1 for highly variable flows. Reward function 3's performance was in between and is thus suggested as the best allrounder to handle dynamically changing traffic. Like Mannion et al. (2016), many authors adopted weighted reward functions to adapt to different situations (Horsuwan & Aswakul, 2019; Nawar et al., 2019; Pol & Oliehoek, 2016; S. Wang et al., 2019; Wei et al., 2018; Y. Wu et al., 2019; Zeng et al., 2019; Zhang et al., 2020). However, finding suitable weights for the sub-rewards was found to be a complex and time-consuming task (Mannion et al., 2016; Pol, 2016).

In general, a problem with many rewards is that traffic is a dynamic system. The arrival rate of vehicles is constantly changing, but most controllers do not take this into account for their reward. As such, if the arrival rate increases, agents may receive a punishment even if they make the right decision (and vice versa). The rewards in these cases do not represent how good the controller's action was (Du et al., 2019). An attempt to solve this was made by Du et al. (2019). In their model, they use a weighted reward of delay and waiting time. But rather than using absolute values or the change compared to the previous phase, they compare the current delay and waiting time to the delay and waiting time caused by a fixed-time controller operating under the same traffic scenario (i.e. with the same arrival rates, vehicle positions and speeds). In this case, agents receive a reward if they perform better than fixed time controllers and a punishment otherwise.

Overall rewards are a trade-off between optimizing traffic flows, driving comfort and fairness. Some authors even add environmental considerations into the reward (Fang et al., 2019; Islam, Aziz, Wang, & Young, 2018; Kim et al., 2019), thus adding another trade-off dimension.

4.2.4 Deep Q-learning extensions and robustness

Just using a neural network as a function approximator can lead to instability and convergence problems. To solve this, nearly all DQN-models use some type of experience replay and many use target networks. As such, DQN with experience replay and target networks can be considered the base or vanilla DQN algorithm.

Additionally, many authors have attempted to improve the stability and performance of TSC algorithms by using rainbow agent extensions, i.e. double Q-learning (Van Hasselt, Guez, & Silver, 2015), prioritized experience replay (Schaul, Quan, Antonoglou, & Silver, 2016), dueling networks (Z. Wang et al., 2016), multi-step bootstrap targets (Sutton, 1988), distributional Q-learning (Bellemare, Dabney, & Munos, 2017) and noisy nets (Fortunato et al., 2017). To evaluate the performance improvements, authors conducted ablation studies or compared the performance against regular DQN. Pol and Oliehoek (2016) evaluated multiple DQN algorithms: base DQN, double Q-learning DQN, prioritized replay DQN and DQN with batch normalization. They found that prioritized experience replay led to an increase of average rewards, that double DQN tends to get stuck in local optima and thus does not improve performance and that batch normalization leads to less stability. Nawar et al. (2019) evaluated the performance of a DQN algorithm with compact rainbow extensions (i.e. the 3 rainbow extensions: prioritized experience replay, multi-step learning, distributional RL) to regular DQN. In the experiments the compact rainbow agent was more stable and led to lower vehicle waiting times, lower trip times and lower fuel consumption compared to regular DQN. Fang et al. (2019) conducted extensive experiments and compared DQN with prioritized experience replay, double Q-learning and dueling networks to (1) regular DQN (2) double DQN with random experience replay (3) dueling DQN with random experience replay (4) double dueling DQN with random experience replay. The results showed that the double dueling controller with prioritized experience replay outperforms all other controllers. Prioritized experience replay was most crucial in performance improvement, leading to around 15% performance improvement compared to regular DQN. Double Q-learning and dueling networks each added around 5% additional performance improvements. Liang et al. (2019) conducted ablation studies on a DQN algorithm with double Q-learning, dueling networks and prioritized experience replay and also compared it to regular DQN. They found that all three extensions contributed to faster learning times, higher rewards and improved performance metrics. From the experiments we can conclude that using rainbow extensions[1] leads to performance improvements. Especially prioritized experience replay and dueling networks gave positive results. Double Q-learning led to improved performance in experiments by Fang et al. (2019) and Liang et al. (2019), but not for Pol and Oliehoek (2016). Overall, the results are in line with the original rainbow experiments by Hessel et al. (2018).

Wei et al. (2018) recently proposed two new extensions: **memory palace** and **phase gates**. In ablation studies, the memory palace method improves results only for unbalanced traffic scenarios, while the phase gate improves the performance in all scenarios. Zeng et al. (2019) used the idea of memory palaces and also tested the extension of **mixed Q-networks** (i.e. a network in which the softmax outputs are replaced with fuzzy classification results). They found no performance

[1] This conclusion has not yet been proven for noisy nets.

differences between regular DQN, DQN with memory palaces, DQN with fuzzy classification or DQN with both extensions. As such, the effects on performance remain unclear.

Lastly, some authors have experimented with using **recurrent Q-networks**, specifically using long short term memory (LSTM). Other studies assume that states are 100% observable and that sensors provide 100% accurate information at all times. However, sensors are not ideal, and mistakes can happen, thus leading to false or missing state inputs. Recurrent networks are used to combat this since they can process sequential information (i.e. they can learn to understand vehicle trajectories and integrate this historical data with current sensor inputs) (Choe et al., 2018; Zhao & Wang, 2019). Three authors compared recurrent DQN with base DQN. T. Zhao and Wang (2019) found that recurrent DQN slightly outperforms DQN for 100% observable states, but significantly outperforms regular DQN in terms of rewards, waiting times, robustness and stability for less than 100% observable states. Choe et al. (2018) found that recurrent DQN reduced average travel times by 23% and overall vehicle waiting times by 10% compared to regular DQN. Lastly, Zeng et al. (2019) found that recurrent DQN led to more efficient exploration. In 100% observable states, regular and recurrent DQN led to similar performance, but under less than 100% observability recurrent DQN significantly outperforms regular DQN. From the experiments, it can be concluded that recurrent DQN can significantly improve the controller's performance, especially if states are not 100% observable.

4.3 Other specifications

4.3.1 Traffic generation

To train and evaluate the controller, it needs to be exposed to traffic scenarios. Scenarios can be constructed in different ways. To create a scenario, researchers must specify both arrival rates and turning ratios for each leg and lane. Some relevant differences are described below:

- **Under-saturated traffic vs saturated traffic vs over-saturated traffic.** Different types of traffic saturation exist. In under-saturated conditions, the vehicle arrival rate is lower than the potential intersection throughput; in saturated traffic the arrival rate is equal to the potential throughput; and in over-saturated traffic the arrival rate is higher than the potential throughput. Under-saturated traffic can e.g. be found during the night, saturated traffic during early afternoons, and over-saturated traffic during rush-hour.
- **Constant vs dynamically changing traffic.** Researchers can either model traffic such that the arrival rates remain constant over time, or as changing over time. Poisson distributions are by far the most popular traffic generation method, but other distributions are suitable as well (Genders & Razavi, 2016). In DQN literature, examples of sinusoidal functions (Du et al., 2019), Weibull distributions (Genders & Razavi, 2016; Vidali, 2018) and Burr distributions (Genders & Razavi, 2016) were found.
- **Balanced vs unbalanced traffic.** Traffic can either have similar arrival rates between lanes (i.e. balanced) or unequal (i.e. unbalanced).
- **Synthetic data vs real-world.** Arrival rates and turning probabilities can either be created synthetically, or data from real-world traffic can be used. For proof-of-concept models, synthetic data may be better suited, since all factors are controllable. For real-world implementations, historical traffic data from the intersection is more useful, since this allows the controller to optimally adapt to the specific intersection.

The exact choice of traffic scenario(s) will depend on the purpose of the model. Nevertheless, some general aspects can be discussed.

To train a model, traffic scenarios should be stochastic in nature. If training scenarios are deterministic, this would mean that the agent is trained over and over on the same scenario, which can lead to severe overfitting. To avoid this, statistical distributions can be used.

Furthermore, agents will only be able to optimally control traffic in scenarios which they have been trained on (Rodrigues & Azevedo, 2019). If the scenario during testing is too different than during training, favorable performance is not guaranteed. Due to this, it is advised to train agents on many different scenarios, as long as they are relevant for the intersection in question. A major limitation in a lot of the reviewed studies was that they only consider one specific traffic scenario, rather than a mix of different ones. This only allows evaluation of a controller on that specific scenario, but not on others. This is problematic, since it has been shown that some controllers perform well in certain scenarios, but not in others (Mannion et al., 2016).

Additionally, some studies only train their controllers on constant traffic (e.g. L. Li et al., 2016; Mousavi et al., 2017; Nawar et al., 2019; Pol & Oliehoek, 2016). However, fixed-time controllers perform especially well for constant traffic flows (Abdulhai et al., 2003), and methods such as the Webster method (Webster, 1958) allow researchers to find optimal cycle times. Fixed-time controllers have even been shown to outperform RL-algorithms for constant, over-saturated traffic (Vidali, 2018). Using RL-algorithms for constant traffic only would not be worth the computational cost. Instead, the actual benefit of adaptive DQN controllers is to control dynamic traffic situations (Abdulhai et al., 2003; Yang, Tan, & Menendez, 2017).

Some good examples of models that have been trained and tested on dynamic traffic demands can be seen in M. Guo et al. (2019), Vidali (2018) and Wei et al. (2018).

4.3.2 Performance indicators

To assess the performance of a proposed controller, key performance indicators (KPI) must be gathered. Various performance metrics were found in literature:

- Gained reward
- Queue length
- Throughput / Number of passed vehicles / Number of completed trips
- Waiting time
- Travel time
- Delay
- Vehicle speed
- Green time per leg/lane
- Number of stops
- Fuel consumption
- Emissions

The found performance metrics mostly overlap with the previously discussed state and reward representations. For the specific descriptions and discussion of the metrics, please refer to sections 4.2.1 and 4.2.3.

In general, some authors report the results in terms of cumulative values, while others report values averaged over vehicles. In some papers also differences between legs/lanes are reported. Most authors also include figures showing the change of the KPIs over time within a traffic scenario. Furthermore, since the traffic scenarios are stochastic, performance measures must be evaluated over several runs. Ideally, both the averages and standard deviations of the runs should be reported and discussed.

4.3.3 Base case(s)

To evaluate the performance of a proposed model, researchers compare their model against other models, the so-called base cases. Different base cases were found in literature:

- Fixed-time control
- **Other traditional TSC:** actuated control, longest queue first, time-loss based control, traditional adaptive control
- Earlier proposed state-of-the-art RL-methods (including non-DQN methods)
- **Other, less sophisticated RL-algorithms:** tabular Q-learning, shallow NN RL-algorithms
- **Model variations:** Ablation studies on their model (i.e. their model minus certain extensions)

The most common base case used was fixed-time control. However, fixed-time control may not be the best base case, since RL-algorithms were shown to outperform or perform equally well as fixed-time control in nearly all cases, and thus do not provide much new information. Ideally, both ablation studies and comparisons with state-of-the-art controllers would be conducted to assess the performance in different scenarios and using different KPI. However due to time and space limitations, this is oftentimes not done.

5. Discussion

5.1 Identified literature gaps

Within the reviewed literature on using DQN in TSC and using RL in mixed TSC, many literature gaps were identified. This section will outline them briefly.

Unclarity about best state representation. In literature, many different state representations were found, ranging from simple to very complex. Authors argue for and against different representations, but only a few authors attempted to do cross-comparisons between different state representations. However, oftentimes in these cross-comparisons authored compared high-dimensional state representations (e.g. DTSE) against low-dimensional state representations (e.g. queue length) (e.g. Genders & Razavi, 2018; Mousavi et al., 2017). Yet, a cross-comparison between different state representations of similar or same information density (e.g. one-hot encoded current green phases vs +/- encoded green phases) is lacking. As such, for future research it is suggested to conduct a systematic study that compares different state representations. In these studies, all model parameters should be kept constant, apart from the state representation. It should be evaluated how different state representations impact model performance and stability, and under which circumstances which state representation is most suitable.

Unclarity about the best reward representation. Like for the state representation, many different reward representations have been suggested in literature. Picking a suitable reward representation is arguably even more difficult since the full reward needs to be aggregated into a single value. Yet, it was seen that different rewards have different advantages and disadvantages, with no reward being

objectively the best. To combat this, some researchers have attempted to use weighted reward functions, yet this causes the issue of how different sub-rewards should be weighted. Overall, like for the state representation, a systematic study comparing the impact of different reward representations on model performance and stability is lacking. Research needs to investigate which rewards work under which circumstances.

Lack of systematic fine-tuning of agents. When designing agents, many hyperparameters have to be set (e.g. the number of episodes to train the model, the target network freeze interval, the memory size). These hyperparameters influence model performance and stability. Yet, not all papers report these values, and nearly none of the studies present how they determined these values. For practice this means that research from previous studies are not reproduceable, but also that it is more difficult to design new agents.

Balancing algorithm goals. When designing a traffic signal controller, several goals have to be balanced: traffic efficiency, safety, fairness to all traffic participants, driver comfort, environmental impact, etc. In current literature on RL in TSC however, most of the reviewed papers focused only on traffic efficiency, and only a few focused on energy or fuel-efficiency or minimizing emissions. Yet, the other goals are oftentimes neglected. Thus, research is needed on how to include and balance the different goals.

> **Fairness.** When designing a controller, only a few researchers explicitly considered fairness. Yet, fairness is an important aspect of TSC. If for example, a majority of the vehicles arrive from one direction, while only very few vehicles from another, a pure traffic efficiency conscious controller would likely "sacrifice" the few vehicles for the many. However, having some vehicles wait forever is undesirable. Some authors attempted to solve this by imposing maximum phase constraints, yet this does not solve the problem that the agent is not actively aware of fairness. Others have experimented with using squared delays to prioritize vehicles that have been waiting for a long time over vehicles that have just arrived. Yet, overall fairness does not gain enough attention. Future research could attempt to explicitly include fairness constraints within a weighted reward.
>
> **Impact on safety.** No studies were found which explicitly investigated the impact of their controller on traffic safety. Some authors included boundary conditions to ensure safety (e.g. yellow and red clearance phases), but none of the studies explicitly included it in the agent's goal.

Lack of variety in traffic scenarios and network topologies. Most literature uses uniformly distributed and balanced traffic scenarios. Yet, as discussed this is unrealistic in the real world, and it does not allow the RL agent to show its true advantages compared to fixed-time controllers. Furthermore, most controllers are only tested on artificial 4-way intersections. However, in the real world many different types of intersections exist. Overall, there is a need to train and test RL agents on a wider range of traffic scenarios (e.g. dynamic, unbalanced, different saturation levels) and network topologies (e.g. 3-way intersections, non-geometric layouts), in order to be able to evaluate for what types of intersections RL would be useful and for which intersections other controllers would be more suitable.

Lack of other traffic participants. All investigated RL agents focus only on vehicles (i.e. regular cars). Other traffic participants, e.g. buses, trucks, cyclists, pedestrians and public transport have been excluded. Excluding them makes models much simpler, but also less realistic. Future research

could investigate the ability of RL agents to accommodate these other traffic participants, particularly regarding safety.

Robustness. Only few papers have investigated the robustness of algorithms (Rodrigues & Azevedo, 2019). Trained RL agents could however break in many ways: signal failures could happen, communication between traffic participants and the infrastructure could be delayed or broken down, road accidents or other unexpected traffic situations could happen, or traffic demand could radically change from the trained scenarios. Good TSC should show a certain level of robustness to the kinds of problems. First suggestions on how to achieve this have been made, but further research will be needed.

Online learning. Related to the problem of robustness is that of online learning. Generally, once an agent is trained and implemented in an intersection, it will only perform greedy actions. This however means that the agent is unable to learn while it is operating, and thus unable to adapt to slow changes in traffic scenarios. A possible solution for this would be to retrain the agent at regular intervals, but due this is time-consuming and expensive. Another solution would be to train the agent while it is online, i.e. while it is controlling the intersection. However, it must be assured that when the agent does explorative actions, that traffic is not unnecessarily hindered. Strategies on achieving this are left for future research.

Impact of mixed traffic scenarios. In the nearby future, traffic will consist of a mixture of conventional and connected and/or automated vehicles. RL TSC may exploit data from connected vehicles to gain better results. Yet, most researchers assume all vehicles are connected, and only few researchers have investigated CV-penetration rates of less than 100%. Several studies (Zeng et al., 2018; Zhang et al., 2020; Zhao & Wang, 2019) have used recurrent DQN agents and found that these compared to 100% CV-penetration lead to nearly equally good performance under CV-penetration rates as low as 10%. However, other agents may lead to even better results.

Furthermore, the problem with current mixed traffic agents is that they must be trained under a fixed penetration rate. This is problematic since it is likely that traffic will consist of different mixtures of CVs and RVs (e.g. different penetration rates at different times of the day or week). Ideally, an agent would be able to accommodate different mixtures of vehicle types. How this can be achieved is currently unknown.

Problematic model validation. Due to time, cost, safety restrictions and possible public inconvenience caused by sub-optimal strategies it is not possible to test newly proposed TSC on real-world intersections. Because of this, controllers must be tested in simulators. However, to get valid results, the simulator must be able to accurately simulate real-world traffic and driving behaviors. While simulators can represent many types of traffic situations, simulators are never able to capture real-life behaviors 100% accurately. Thus, to completely validate whether proposed strategies would work in the real-world, field studies will be needed.

Lack of benchmarks and systematic comparison between controllers. Since deep RL is such a fast-developing field, new agents are being proposed very quickly. Researchers are not only experimenting with the state, reward and action representations but also with the types of RL models and extensions. In recent years, many new DQN extensions and other RL algorithms have been successfully applied in other domains. Many researchers have started to implement these controllers also for TSC. Yet, up to date there is a lack of systematic comparisons between controllers. Many

researchers compare their agents to traditional TSC, but only few compare them to state-of-the-art RL controllers (e.g. Fang et al., 2019).

Part of this problem may be caused by the fact that unlike in some other domains, there are no standardized test scenarios TSC. For the development of current deep RL agents for example, many researchers test their algorithms on standardized environments such as the Cartpole problem, the Doom game (Lample & Chaplot, 2017) or the Atari 2600 benchmark (Hessel et al., 2018; Mnih et al., 2015). Many of these environments have been combined into an easily accessible toolkit called OpenAI Gym (Brockman et al., 2016). Having such standardized environments with standardized scenarios enables a better comparison between model types and allows benchmarking, and thus speeds up algorithm evaluation (OpenAI, 2016). Future research may look into developing such an environment.

5.2 Recommendations for future research

Recommendations in this section relate to the identified research gaps in the previous section.

A major limitation in reinforcement learning-based traffic signal control research is the lack of standardized testing scenarios and environments. Currently, each research team is implemented its agents in slightly different ways (e.g. different network topologies, different traffic demand generation), which reduces the ability to do cross-comparisons between different studies. If a set of standardized environments and scenarios would exist, researchers could test their proposed algorithms on these. Having unified testing scenarios would facilitate benchmarking and objective comparison of agents, and thus more systematic evaluations. When developing a set of environments, care must be taken that traffic scenarios cover the relevant different types of situations (e.g. under-saturated, saturated, over-saturated traffic; constant and dynamic traffic). The set of scenarios should neither be too small (which would limit researchers), nor be too large (which would hamper cross-comparisons). Regarding the implementation, similar approaches can be taken as for the implementation of the Atari or Doom environments in the OpenAI Gym (OpenAI, 2016).

Additionally, more research is needed on how agents can be fine-tuned. At the moment, no best practices regarding choices in states, rewards, network topologies and hyperparameters exist, making it time-consuming to design new agents. Having clearer guidelines would significantly reduce the time to develop agents since researchers would not have to conduct as many fine-tuning experiments by themselves. Thus, it is suggested to systematically conduct and document fine-tuning experiments.

Another future research suggestion is to further develop existing traffic signal controllers' models. Deep reinforcement learning is a fast-developing field in which authors continuously develop new model types or extensions to existing models. Combining several extensions could lead to significantly increased performance compared to models without extensions or only single extensions (as e.g. shown in the highly-cited Rainbow paper by Hessel et al. (2018)). But it may also be possible that extensions do not combine well. Additionally, it may be possible that other model types than the most commonly used deep Q-learning model perform better, such as the hybrid actor-critic model. For traffic signal control specifically, only a limited amount of papers systematically compared model types and/or extensions in ablation studies (Fang et al., 2019; Wei et al., 2018; Zeng et al., 2019). Yet, many other newly developed methods exist which may improve performance further, but which have not yet been investigated for traffic signal control. For example, Kapturowski, Ostrovski, Quan, Munos, & Dabney (2019) have created an agent that combined recurrent deep Q-

learning with prioritized experience replay and distributed Q-learning, which outperforms existing deep RL agents on the Atari-57 benchmark. Similar studies could be conducted in TSC.

Furthermore, more research is needed which looks into multi-goal agents. Currently, many authors focus solely on traffic efficiency, at the expense of other goals. Yet, safety, fairness, environmental impacts, driver comfort and social acceptance are all goals which also have to be taken into account when designing a new traffic signal controller. For example, if a controller finds a strategy in which green phases continuously let only a single car pass, this would lead to uncomfortable driving. Another example would be if an agent sacrifices a single queuing vehicle for the better performance of the vehicles in a long queue and thus letting the single vehicle wait forever. In existing controllers, it is difficult to include multiple goals. Partly this is caused by the fact that the algorithms are model-free which makes it not possible to change the inner workings. And partly it is caused due to it being difficult to weigh several goals and combine them into a balanced reward function. This leads to the next research suggestion, namely, to investigate how multiple rewards can be balanced to gain good performance.

Lastly, future research should investigate whether agents will be able to perform well in the real-world under less ideal circumstances than the simulator provides. Most studies currently assume ideal conditions, yet the real-world is more chaotic. For example, unexpected or previously unencountered traffic situations may happen (e.g. accidents, radical demand changes), lane switches and traffic law violations could take place or signal failures, or communication delays or errors could happen. These types of issues may compromise the robustness of the agent, and thus adversely affect its ability to reduce traffic congestion.

5.3 Literature review limitations

This literature review is subject to several limitations. Despite care being taken to exhaustively select literature on DQN in TSC, some articles may have been missed due to a use of different terminology, due to not being included in the search engines which were used or due to being published in a language other than English.

Furthermore, for this review only literature on deep Q-learning was included. Yet, in the past many researchers have used tabular Q-learning, and recently researchers have started to use a variety of other RL methods (e.g. actor-critic methods, policy-gradient methods). Since these topics have not been investigated, some of the identified literature gaps may have been already discussed in those papers.

The same limitation applies to the fact that multi-intersection DQN models have been excluded from the review. Recently many efforts have been made to create RL controllers which do not only consider a single intersection, but which optimize a network of intersections. Research on network performance is needed, since connecting several RL agents which are trained on isolated intersections may not lead to optimal results. This is because each intersection would make its own locally optimal decisions but would not cooperate with its neighbors. These “selfish”, local decisions could negatively impact other agents (e.g. block another vehicle, oversaturate another controller’s capacity) and lead to negative emergent patterns for the system as a whole (e.g. decreased intersection/network throughput) (Martínez-Díaz, Soriguera, & Pérez, 2019). Some researchers have started to investigate this problem (e.g. Chu, Wang, Codeca, & Li, 2020; Gong, Abdel-Aty, Cai, & Rahman, 2019; Hussain, Wang, & Jiahua, 2020; Klöckner & Klose, 2020; Lee, Chung, & Sohn, 2019; Xu

et al., 2020; Yin, Wang, & Li, 2020). The field of network control is getting increasingly larger and warrants a separate review of its own. As such, it was not included in this review.

Lastly, while this paper has attempted to discuss as many design choices as possible, the mentioned topics are not exhaustive. Topics that have not been discussed in this review are e.g. the types of exploration strategies, the types of optimizers or the types of simulators that have been used.

5. Conclusion

This paper aimed to review the current state-of-the-art in intelligent traffic signal control using deep Q-Learning, to identify gaps in existing literature and to provide suggestions for future research directions. These research questions have been answered by conducting a systematic literature review.

By doing this, this paper has contributed a comprehensive review of the current state-of-the-art which can guide new researchers to get a good overview of the field of deep Q-learning in TSC and what types of decisions have to be made. Furthermore, the made suggestions can hopefully guide future researchers what aspects should be investigated more in-depth.

Overall it can be concluded that deep RL in TSC is a promising new field that is developing very quickly. New controllers are continuously being proposed, and many researchers are working on increasingly better-performing algorithms. When designing an RL agent for TSC, many design choices have to be made. In literature, many different choices regarding the state, action and reward representation, the RL agent, network topologies, traffic generation, KPIs and base cases have to be made. Yet, systematic comparisons and best practices for these choices are lacking. Much of the design process currently seems to be trial-and-error based. To facilitate better systematic comparisons between proposed agents, it is highly suggested to develop standardized environments with standardized scenarios in order to benchmark controllers. This will allow for faster development and evaluation.

References

Abdulhai, B., Pringle, R., & Karakoulas, G. J. (2003). Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*. https://doi.org/10.1061/(ASCE)0733-947X(2003)129:3(278)

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep Reinforcement Learning: A brief survey. *IEEE Signal Processing Magazine*, *34*(6), 26–38. https://doi.org/10.1007/978-981-13-8285-7

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A distributional perspective on reinforcement learning. *34th International Conference on Machine Learning, ICML 2017*.

Bishop, C. M. (1996). Neural networks: a pattern recognition perspective. *Neural Networks*. https://doi.org/10.1.1.46.8742

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai Gym. *ArXiv Preprint ArXiv:1606.01540*.

Brys, T., Pham, T. T., & Taylor, M. E. (2014). Distributed learning and multi-objectivity in traffic light control. *Connection Science*. https://doi.org/10.1080/09540091.2014.885282

Chang, G. L., & Xiang, H. (2003). The relationship between congestion levels and accidents. In *State Highway Adiministration*.

Chen, L., & Englund, C. (2016). Cooperative Intersection Management: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, *17*(2), 570–586. https://doi.org/10.1109/TITS.2015.2471812

Chen, P., Zhu, Z., & Lu, G. (2019). An Adaptive Control Method for Arterial Signal Coordination Based on Deep Reinforcement Learning. *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, *100191*(37), 3553–3558. https://doi.org/10.1109/ITSC.2019.8917051

Choe, C., Baek, S., Woon, B., Kong, S., & Member, S. (2018). Deep Q Learning with LSTM for Traffic Light Control. *2018 24th Asia-Pacific Conference on Communications (APCC)*, 331–336.

Chu, T., Wang, J., Codeca, L., & Li, Z. (2020). Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, *21*(3), 1086–1095. https://doi.org/10.1109/TITS.2019.2901791

Coşkun, M., Baggag, A., & Chawla, S. (2019). Deep reinforcement learning for traffic light optimization. *IEEE International Conference on Data Mining Workshops, ICDMW*, 564–571. https://doi.org/10.1109/ICDMW.2018.00088

Denney, R. W., Curtis, E., & Olson, P. (2012). The national traffic signal report card. *ITE Journal (Institute of Transportation Engineers)*.

Du, Y., Shangguan, W., Rong, D., & Chai, L. (2019). RA-TSC: Learning Adaptive Traffic Signal Control Strategy via Deep Reinforcement Learning. *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, 3275–3280. https://doi.org/10.1109/ITSC.2019.8916967

El-Tantawy, S., Abdulhai, B., & Abdelgawad, H. (2014). Design of reinforcement learning parameters for seamless application of adaptive traffic signal control. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*. https://doi.org/10.1080/15472450.2013.810991

Fang, S., Chen, F., & Liu, H. (2019). Dueling Double Deep Q-Network for Adaptive Traffic Signal Control with Low Exhaust Emissions in A Single Intersection. *IOP Conference Series: Materials Science and Engineering*, *612*(5). https://doi.org/10.1088/1757-899X/612/5/052039

Florin, R., & Olariu, S. (2015). A survey of vehicular communications for traffic signal optimization. *Vehicular Communications*, *2*(2), 70–79. https://doi.org/10.1016/j.vehcom.2015.03.002

Fortunato, M., Azar, M., Piot, B., Menick, J., Hessel, M., Osband, I., … Legg, S. (2017). Noisy networks for exploration. *ArXiv Preprint ArXiv:1706.10295.*

Gao, J., Shen, Y., Liu, J., Ito, M., & Shiratori, N. (2017). *Adaptive Traffic Signal Control : Deep Reinforcement Learning Algorithm with Experience Replay and Target Network* (pp. 1–10). pp. 1–10. arXiv preprint arXiv:1705.02755.

Genders, W., & Razavi, S. (2016). Using a Deep Reinforcement Learning Agent for Traffic Signal Control. *ArXiv Preprint ArXiv:1611.01142*, pp. 1–9.

Genders, W., & Razavi, S. (2018). Evaluating reinforcement learning state representations for adaptive traffic signal control. *Procedia Computer Science*, *130*, 26–33. https://doi.org/10.1016/j.procs.2018.04.008

Gong, Y., Abdel-Aty, M., Cai, Q., & Rahman, M. S. (2019). Decentralized network level adaptive signal control by multi-agent deep reinforcement learning. *Transportation Research Interdisciplinary Perspectives*, *1*(100020). https://doi.org/10.1016/j.trip.2019.100020

Guo, M., Wang, P., Chan, C. Y., & Askary, S. (2019). A Reinforcement Learning Approach for Intelligent Traffic Signal Control at Urban Intersections. *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, 4242–4247. https://doi.org/10.1109/ITSC.2019.8917268

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., … Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 3215–3222.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*. https://doi.org/10.1162/neco.1997.9.8.1735

Horsuwan, T., & Aswakul, C. (2019). Reinforcement learning agent under partial observability for traffic light control in presence of gridlocks. *EPiC Series in Computing*, *62*, 29–47. https://doi.org/10.29007/bdgn

Hussain, A., Wang, T., & Jiahua, C. (2020). *Optimizing Traffic Lights with Multi-agent Deep Reinforcement Learning and V2X communication* (pp. 1–6). pp. 1–6. Retrieved from http://arxiv.org/abs/2002.09853

Islam, S. M. A. B. Al, Aziz, H. M. A., Wang, H., & Young, S. (2019). *Investigating the Impact of Connected Vehicle Market Share on the Performance of Reinforcement-Learning Based Traffic Signal Control (No. ORNL/TM-2019/1233)*. Oak Ridge, TN (United States).

Islam, S. M. A. B. Al, Aziz, H. M. A., Wang, H., & Young, S. E. (2018). Minimizing energy consumption from connected signalized intersections by reinforcement learning. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, *2018-Novem*, 1870–1875. https://doi.org/10.1109/ITSC.2018.8569891

Jing, P., Huang, H., & Chen, L. (2017). An adaptive traffic signal control in a connected vehicle environment: A systematic review. *Information (Switzerland)*, *8*(3). https://doi.org/10.3390/info8030101

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., & Dabney, W. (2019). Recurrent experience replay in distributed reinforcement learning. *7th International Conference on Learning Representations, ICLR 2019*, 1–19.

Kim, J., Jung, S., Kim, K., & Lee, S. (2019). The Real-Time Traffic Signal Control System for the Minimum Emission using Reinforcement Learning in Vehicle-to-Everything (V2X) Environment. *Chemical Engineering Transactions*, *72*(March 2018), 91–96. https://doi.org/10.3303/CET1972016

Klöckner, R., & Klose, P. (2020). Deep-MARLIN: Using Deep Multi-Agent Reinforcement Learning for Adaptive Traffic Light Control. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3378184.3378194

Lample, G., & Chaplot, D. S. (2017). Playing FPS Games with Deep Reinforcement Learning. *Thirty-First AAAI Conference on Artificial Intelligence*.

Lee, J., Chung, J., & Sohn, K. (2019). Reinforcement learning for joint control of traffic signals in a transportation network. *IEEE Transactions on Vehicular Technology*, 1–12.

Li, L., Lv, Y., & Wang, F.-Y. (2016). Traffic Signal Timing via Deep Reinforcement Learning. *IEEE/CAA*

Journal of Automatica Sinica, *3*(3), 247–254. https://doi.org/10.1007/978-981-13-8683-1_12

Li, Y. (2018). Deep Reinforcement Learning: An Overview. *ArXiv Preprint ArXiv:1701.07274*. https://doi.org/10.1007/978-3-319-56991-8_32

Liang, X., Du, X., Member, S., & Wang, G. (2019). A Deep Reinforcement Learning Network for Traffic Light Cycle Control. *IEEE Transactions on Vehicular Technology*, *68*(2), 1243–1253. https://doi.org/10.1109/TVT.2018.2890726

Liang, X., Du, X., Wang, G., & Han, Z. (2018). Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks. *IEEE Transactions on Vehicular Technology*, *68*(2), 1–11. https://doi.org/10.1109/TVT.2018.2890726

Mannion, P., Duggan, J., & Howley, E. (2016). An Experimental Review of Reinforcement Learning Algorithms for Adaptive Traffic Signal Control. *Autonomic Road Transport Support Systems*, 47–66. https://doi.org/10.1007/978-3-319-25808-9

Marchesini, P., & Weijermars, W. (2010). The relationship between road safety and congestion on motorways. *SWOV Institute for Road Safety Research*.

Martínez-Díaz, M., Soriguera, F., & Pérez, I. (2019). Autonomous driving: A bird's eye view. *IET Intelligent Transport Systems*, *13*(4), 563–579. https://doi.org/10.1049/iet-its.2018.5061

Mikami, S., & Kakazu, Y. (1994). Genetic reinforcement learning for cooperative traffic signal control. *IEEE Conference on Evolutionary Computation - Proceedings*. https://doi.org/10.1109/icec.1994.350012

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Mousavi, S. S., Schukat, M., & Howley, E. (2016). Deep Reinforcement Learning: An Overview. *Proceedings of SAI Intelligent Systems Conference*, 426–440. https://doi.org/10.1007/978-3-319-56991-8_32

Mousavi, S. S., Schukat, M., & Howley, E. (2017). Traffic light control using deep policy- gradient and value-function-based reinforcement learning. *IET Intelligent Transport Systems*, *11*, 417–423. https://doi.org/10.1049/iet-its.2017.0153

Nawar, M., Fares, A., & Al-Sammak, A. (2019). Rainbow Deep Reinforcement Learning Agent for Improved Solution of the Traffic Congestion. *Proceedings of the International Japan-Africa Conference on Electronics, Communications and Computations, JAC-ECC 2019*, 80–83. https://doi.org/10.1109/JAC-ECC48896.2019.9051262

OpenAI. (2016). Getting Started with Gym. Retrieved July 1, 2020, from https://gym.openai.com/docs/

Pol, E. van der. (2016). *Deep Reinforcement Learning for Coordination in Traffic Light Control*. University of Amsterdam.

Pol, E. van der, & Oliehoek, F. A. (2016). Coordinated Deep Reinforcement Learners for Traffic Light Control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016).*

Rafter, C. B., Anvari, B., & Box, S. (2018). Traffic responsive intersection control algorithm using GPS data. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. https://doi.org/10.1109/ITSC.2017.8317795

Rodrigues, F., & Azevedo, C. L. (2019). Towards Robust Deep Reinforcement Learning for Traffic Signal Control : Demand Surges , Incidents and Sensor Failures. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 3559–3566. IEEE.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*.

Schrank, D., Eisele, B., & Lomax, T. (2019). *Urban Mobility Report*. Retrieved from https://static.tti.tamu.edu/tti.tamu.edu/documents/mobility-report-2019.pdf

Shabestary, S. M. A., & Abdulhai, B. (2019). Deep reinforcement learning for adaptive traffic signal control. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. https://doi.org/10.1115/DSCC2019-9076

Sutton, R. S. (1988). Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, *3*(1), 9–44. https://doi.org/10.1023/A:1022633531479

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd Editio). Cambridge, MA; London, England: MIT Press.

Van Hasselt, H., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double Q-Learning. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2094–2100.

Vidali, A. (2018). *Simulation of a traffic light scenario controlled by a Deep Reinforcement Learning agent*. Università degli Studi di Milano Bicocca.

Wang, S., Xie, X., Huang, K., Zeng, J., & Cai, Z. (2019). Deep reinforcement learning-based traffic signal control using high-resolution event-based data. *Entropy*, *21*(8), 1–16. https://doi.org/10.3390/e21080744

Wang, Y., Yang, X., Liang, H., & Liu, Y. (2018). A review of the self-adaptive traffic signal control system based on future traffic environment. *Journal of Advanced Transportation*, *2018*. https://doi.org/10.1155/2018/1096123

Wang, Z., Schaul, T., Hessel, M., Lanctot, M., Parisotto, E., Ba, J. L., … Botvinick, M. (2016). Dueling Network Architectures for Deep Reinforcement Learning Hado van Hasselt. *Advances in Neural Information Processing Systems*. https://doi.org/10.1039/c004615a

Webster, F. V. (1958). Traffic signal settings. *Road Research Technical Paper, No.39, Road Research Laboratory, London*.

Wei, H., Zheng, G., Yao, H., & Li, Z. (2018). IntelliLight : A Reinforcement Learning Approach for Intelligent Traffic Light Control. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2496–2505.

Wu, T., Kong, F., & Fan, Z. (2019). Road Model Design Based on Reward Function in Traffic Light Control. *2019 5th International Conference on Control, Automation and Robotics, ICCAR 2019*, 407–412. https://doi.org/10.1109/ICCAR.2019.8813381

Wu, Y., Chen, H., & Zhu, F. (2019). DCL-AIM: Decentralized coordination learning of autonomous intersection management for connected and automated vehicles. *Transportation Research Part C: Emerging Technologies*, *103*(November 2018), 246–260. https://doi.org/10.1016/j.trc.2019.04.012

Xu, M., Wu, J., Huang, L., Zhou, R., Wang, T., & Hu, D. (2020). Network-wide traffic signal control based

on the discovery of critical nodes and deep reinforcement learning. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, *24*(1), 1–10. https://doi.org/10.1080/15472450.2018.1527694

Yang, K., Tan, I., & Menendez, M. (2017). A reinforcement learning based traffic signal control algorithm in a connected vehicle environment. *17th Swiss Transport Research Conference (STRC 2017)*. https://doi.org/10.3929/ethz-a-010782581

Yau, K.-L. A., Qadir, J., Khoo, H. L., Ling, M. H., & Komisarczuk, P. (2017). A survey on reinforcement learning models and algorithms for traffic signal control. *ACM Computing Surveys*, *50*(3), 38. https://doi.org/10.1145/3068287

Yin, M., Wang, Y., & Li, Z. (2020). Optimization of Multi-Intersection Traffic Signal Timing Model Based on Improved Q-Learning Optimization of Multi-Intersection Traffic Signal Timing Model Based on Improved Q-Learning. *IOP Conf. Ser.: Mater. Sci. Eng.* https://doi.org/10.1088/1757-899X/768/7/072100

Zeng, J., Hu, J., & Zhang, Y. (2018). Adaptive Traffic Signal Control with Deep Recurrent Q-learning. *2018 IEEE Intelligent Vehicles Symposium (IV)*, 1215–1220. https://doi.org/10.1109/IVS.2018.8500414

Zeng, J., Hu, J., & Zhang, Y. (2019). Training Reinforcement Learning Agent for Traffic Signal Control under Different Traffic Conditions. *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, 4248–4254. https://doi.org/10.1109/ITSC.2019.8917342

Zhang, R., Ishikawa, A., Wang, W., Striner, B., & Tonguz, O. (2020). Using Reinforcement Learning with Partial Vehicle Detection for Intelligent Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems*, 1–12. https://doi.org/10.1109/TITS.2019.2958859

Zhao, T., & Wang, P. (2019). Traffic Signal Control with Deep Reinforcement learning. *2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS)*, *2*(c), 763–767. https://doi.org/10.1109/ICICAS48597.2019.00164