Deep Reinforcement Learning in Transportation Research: A Review

Nahid Parvez Farazi, Bo Zou*, Tanvir Ahamed, Limon Barua
Department of Civil, Material, and Environmental Engineering
University of Illinois at Chicago

Abstract: Applying and adapting deep reinforcement learning (DRL) to tackle transportation problems is an emerging interdisciplinary field. While rapidly growing, a comprehensive and synthetic review of existing DRL applications and adaptations in transportation research remains missing. The objective of this paper is to fill this gap. We expose the broad transportation research community to the methodological fundamentals of DRL, and present what have been accomplished in the literature by reviewing a total of 155 relevant papers that have appeared between 2016 and 2020. Based on the review, we further synthesize the applicability, strengths, shortcomings, issues, and directions for future DRL research in transportation, along with a discussion on the available DRL research resources. We hope that this review will serve as a useful reference for the transportation community to better understand DRL and its many potentials to advance research, and to stimulate further explorations in this exciting area.

Keywords: deep reinforcement learning, transportation research, application domain, literature review, synthetic discussion.

* Corresponding Author. Email: bzou@uic.edu.

1 Introduction

Moving people and freight in a safe, efficient, and sustainable manner involves a wide range of decision-making tasks. Deep reinforcement learning (DRL), by integrating the power of deep learning and reinforcement learning (RL), provides a generic and flexible framework for sequential decision-making that is amenable to many transportation operation and planning problems. The potential of applying DRL to tackling transportation problems has been broadened even further with the increasing availability of computation power. As a result, DRL has been attracting soaring interests from the transportation research community. Just within a few years, a large number of DRL based transportation studies have emerged, with reported results outperforming existing benchmarks.

One salient feature of DRL is that high quality results can be generated at an extremely fast pace once the DRL agent (which makes decisions) is trained, which is very crucial in highly dynamic environments that demand real-time decisions, as are relevant to many transportation operations such as driving and traffic control. In addition, the ability of DRL to solve large, complex problems makes it promising to tackle planning aspects of transportation, for example vehicle routing, timetabling, and path following. It is therefore not surprising that over 150 papers have appeared in the literature just between 2016 and 2020 (July) (Fig. 1). While the literature continues to grow, what we find missing is a comprehensive, synthetic review of existing DRL applications and adaptations in transportation. This paper intends to fill the gap.

The objective of this review is to expose the broad transportation research community to the methodological fundamentals of DRL, present what have been accomplished in applying/adapting DRL to tackling various transportation problems in different domains, and synthesize the applicability, strengths, shortcomings, common and application-specific issues, and directions for future DRL research in transportation. Our compilation of papers comes primarily from search on Google Scholar as well as a few existing surveys which have a narrower focus (Haydari and Yilmaz (2020) on DRL for intelligent transportation systems, and Talpaert et al. (2019) and Kiran et al. (2021) on DRL for autonomous driving). We begin the search with generic keywords including “deep reinforcement learning” and “transportation”, to collect a laundry list of potentially relevant papers, from which an initial idea of the transportation application domains are formed. Then, for each domain, we perform a more directed search with domain-specific keywords. As an example, for the domain of autonomous driving, keywords “lane changing behavior”, “autonomous control”, and “end-to-end autonomous driving” are used in addition to “deep reinforcement learning” in the search.

The literature search leads to 155 papers collected from 2016 to July 2020. No relevant papers can be found prior to 2016, which indicates the relative short history of DRL for tackling tranpsortaiton problems. The 155 papers fall into seven application domains: 1) autonomous driving; 2) emergy efficient driving; 3)

adaptive traffic signal control; 4) other types of traffic control; 5) vehicle routing optimization; 6) rail transportation; and 7) maritime transportation. Among these papers, 75 come from conference and symposium proceedings, 55 are published in peer-reviewed journals, and the remaning 25 are unpublished papers available in online archives. Fig. 1 shows the distribution of the 155 papers over the years and across the seven domains. Note that for some papers which have an updated version after July 2020, our review is based on the latest version at the time of our paper writing.

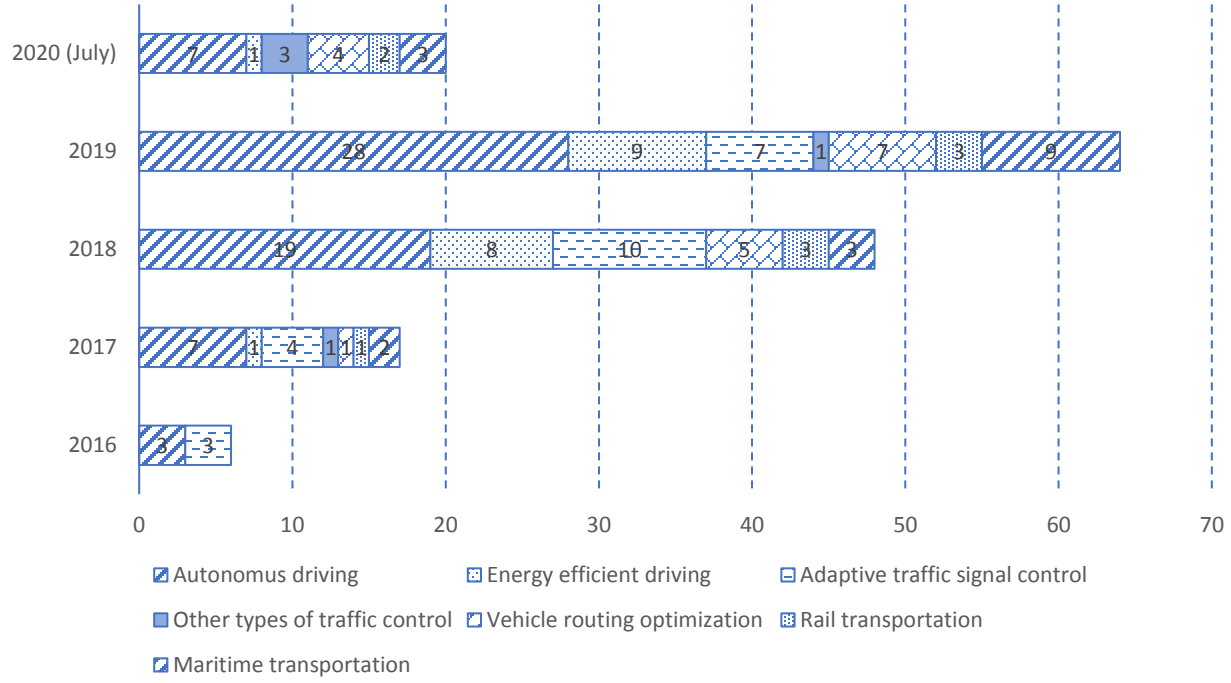

Figure 1. Distribution of DRL papers in seven transportation domains.

The remainder of the paper is organized as follows. In section 2, we first offer a methodological overview of DRL, starting from the fundamentals of RL and then moving on to DRL with focus on different algorithms and extensions. Section 3 presents a comprehensive review of the existing literature on using DRL to address a variety of transportation problems in the seven application domains. Based on the review, a synthetic discussion of the applicability, strengths, shortcomings, issues, and directions for future DRL research in transportation is conducted in section 4. We also provide information on avaialble resources, particularly existing built-in platforms in section 5. Concluding remarks are given in section 6.

2 Methodological background

This section presents the methodological background of DRL. We first offer a brief discussion of RL, based on which we describe how RL is enhanced by integrating deep learning, which gives birth to "Deep

Reinforcement Learning". Both popular DRL algorithms that have been considered in transportation research and extensions of these algorithms are covered.

2.1 Reinforcement learning

RL represents one of the three categories of machine learning (the other two are supervised learning and unsupervised learning). The focus of RL is to train an agent such that the agent can optimize its behavior by learning from its experiences of interacting with the environment. More specifically, RL is a sequential decision process with the agent being the decision maker. At each decision point, the agent has information about the current state of the environment and selects an action that deems the most appropriate based on his experiences at that point. The action taken transitions the environment to a new state. Meanwhile, the agent gets some reward, i.e., reinforcement, as a signal of how good or bad the action taken is.

To formulate the sequential decision process, RL employs Markov Decision Processes (MDP) as the mathematical foundation to keep track of the progression of the decision process. To do so, the following notations are introduced. Set $\boldsymbol{S}$ includes the possible states of the environment. Set $\boldsymbol{A}$ contains the possible actions that the agent can take. Set $\boldsymbol{R}$ includes the possible rewards as a result of the agent taking an action at a given state. At time step t, the environment is in state s_t and from this state, the agent takes an action a_t. The action taken results in a transition of the environment to a new state s_{t+1} at the next time step $t+1$. Meanwhile, the agent receives a reward r_t as a result of the action taken. The reward is a function of state-action pair: $\mathcal{R}(s_t, a_t) \rightarrow r_t$. This agent-environment interaction is further shown in Fig. 2.

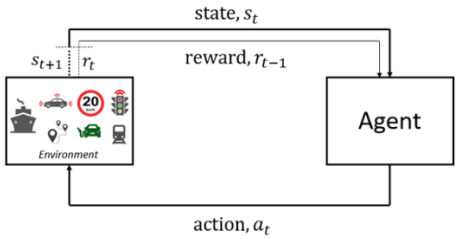

Figure 2. Illustration of agent-environment interaction.

Since actions are taken sequentially, the objective of the agent is to maximize the cumulative reward, which is the expected return over the entire time period. At a time step t, the expected return $\bar{R}_t$ is the sum of rewards from the current time step onward till the last time step T:

$$\bar{R}_t = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T \quad (1)$$

If we consider that the reward is received over a long period, then a discount factor γ may be incorporated to reflect discounting. The expected return is:

$$\bar{R}_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+\mathrm{k}} \quad (2)$$

Exploration vs. exploitation: When taking an action, the agent needs to keep in mind the tradeoff between taking the best action based on the experiences accumulated so far (exploit) and gathering new experiences in order to make better actions in the future (explore). The agent must do both and try a variety of actions to progressively favor those that appear to be best (Sutton and Barton, 2018). A common approach to account for the tradeoff is the ε-greedy strategy, under which the agent takes a random action with a probability ε. At the beginning of the training, ε is set to 1 to ensure that the agent starts by purely exploring the environment. Over time, ε is gradually reduced with a decay rate, to allow for more exploitation as more experiences are accumulated.

Model-based vs. model-free approach: Depending on the environment transition behavior, RL algorithms can be classified into two classes: model-based and model-free. In model-based algorithms, given a state-action pair (s_t, a_t), a transition function $\mathcal{T}(s_{t+1}|s_t, a_t)$, which indicates the probability of state transition given the current state s_t and the action taken a_t, is used to predict the next state s_{t+1}. As such, a model-based algorithm decides on a course of actions by anticipating future situations before they actually occur (Sutton and Barto, 2018). In contrast, for model-free algorithms the agent does not need any model or transition function, but relies on a trial-and-error process. Model-free algorithms are relatively simple, inexpensive, and widely applied in transportation research. Our review in this section mainly focuses on model-free algorithms. Model-free algorithms can be further classified into three classes: 1) value-based algorithms; 2) policy-based algorithms; and 3) actor-critic algorithms, the latter combining value-based and policy-based algorithms. Below we provide a brief overview of these three classes.

2.1.1 Value-based algorithms

Value-based RL algorithms involve estimating the value (i.e., expected return) of a given state at a given time. This is referred to as estimating the value function. A value function tells the agent how good it is for the agent to: 1) be in a state at a given time, or 2) take an action from a state at a given time. Accordingly, two types of value function exist: state-value function and action-value function. The state-value function $V_\pi(s)$, defined by Eq. (3), gives the expected return when the environment starts in state s and follows policy $\pi = \pi(a|s)$ which is a mapping from states to probabilities of selecting each possible action. On the other hand, the action-value function $Q_\pi(s, a)$, expressed in Eq. (4), is the expected return starting from state s, taking action a, and thereafter following a policy π.

$$V_\pi(s) = E_\pi(\bar{R}_t|s_t = s) = E_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\,|s_t = s\right) \quad (3)$$

$$Q_\pi(s,\ a) = E_\pi(\bar{R}_t\ |s_t = s,\ a_t = a) = \ E_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\ |s_t = s, a_t = a\right) \tag{4}$$

One of the most popular and widely used value-based RL algorithms is "*Q-learning*" (Watkins and Dayan, 1992), which is an off-policy approach. Q-learning algorithm enables the agent to choose an action $a \in \boldsymbol{A}$ with the highest Q-value available from state $s \in \boldsymbol{S}$ based on a Q matrix which is a mapping for a discrete state-action space. The Q matrix is updated every time step following the Bellman optimality equation as shown in Eq. (5), where r is the reward obtained and α is the learning rate which takes values between 0 and 1. Here, *off-policy* means that regardless of the policy (as reflected by the Q matrix) being used to direct the agent to take action a at the current state s, the agent will update Q-value of state-action pair (s, a) using the transitioned state's optimal Q-value, i.e., $\max_{a' \in \boldsymbol{A}} Q(s', a')$. Thus, the selection of action a at the current state s and selection of action a' at the next state s' are not from the same policy. This is different from the *on-policy* approach, where action selection at the next state s' follows the same policy as in selecting action a at the current state s.

$$Q(s,a) \longleftarrow (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a' \in \boldsymbol{A}} Q(s', a')\right] \tag{5}$$

2.1.2 Policy-based algorithms

Unlike value-based algorithms, policy-based algorithms do not require estimating the value of a certain state or state-action pair, but search for an optimal policy π^* directly. Typically, a parameterized policy π_θ is chosen, with parameter θ constantly updated towards maximizing the expected return:

$$V_{\pi_\theta}(s) = E_{\pi_\theta}\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\ |s_t = s\right) \tag{6}$$

Policy-based algorithms are particularly suitable for very large or infinite action space. To illustrate, consider $J(\theta)$ as some scalar performance measure. Policy-based RL algorithms seek to maximize performance, by updating policy parameter θ through either a gradient-free or a gradient-based approach (Deisenroth et al., 2013). Under the gradient-based approach, the update is done by:

$$\theta' \leftarrow \theta + \alpha\widehat{\nabla J(\theta)} \tag{7}$$

where $\widehat{\nabla J(\theta)}$ is a stochastic estimate of the gradient of $J(\theta)$ with respect to θ (Sutton and Barto, 2018). A popular policy-based algorithm is REINFORCE (Williams, 1992), in which update of θ at time step t involves only the action taken (a) from the current state (s):

$$\theta' \leftarrow \theta + \alpha\gamma^t \bar{R}_t \nabla \ln \pi_\theta(a|s,\theta) \tag{8}$$

where $\bar{R}_t$ is the expected return at time step t. Note that although the update only requires action taken, REINFORCE uses the expected return from the current time step. Therefore, REINFORCE is well defined only for a task that has a terminal state and only after all updates in an episode are complete.

2.1.3 Actor-critic algorithms

Value-based and policy-based algorithms both have limitations. For value-based algorithms, they cannot handle problems that involve continuous (real-valued) and high-dimensional action space. For policy-based algorithms, gradient estimators may have large variances (Konda and Tsitsiklis, 2000). Moreover, with changes to policy, the new gradient is estimated irrespective of the previous policies. Therefore, the agent is not learning with respect to the accumulation of previous information. To overcome the limitations, an actor-critic approach has been suggested that combines the two classes of algorithms (Konda and Tsitsiklis, 2000; Grondman et al., 2012). In the actor-critic approach, the agent is trained using two estimators (Fig. 3). One is a critic function which approximates and updates the value function. The other one is an actor function which controls the agent's behavior based on policy. Based on the value function derived from the critic function, the actor function's policy parameter is updated in the direction of performance improvement. While the actor function controls the agent's behavior based on policy, the critic function evaluates the selected action based on the value function.

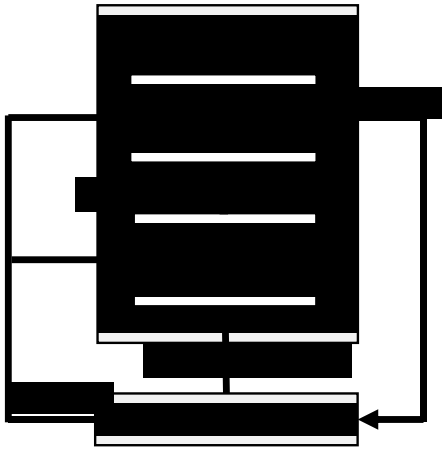

Figure 3. Conceptual framework for actor-critic algorithms.

2.2 Deep reinforcement learning

In this section, we move our discussions from RL to DRL. In principle, if the agent knows the optimal Q-value $Q_*(s, a)$ for every state-action pair, then the objective of RL is achieved. For every state, the agent will find the highest Q-value from the Q matrix and choose the corresponding action. However, such a table of Q-value can only be developed by recursively solving Eq. (5) with a small state and action space. For many real-world problems that are associated with a large state and action space, a tabular format becomes computationally inefficient and even infeasible. To mitigate the "curse of dimensionality", deep learning is

used as a function approximator and integrated with RL. A parameterized deep neural network (DNN) approximates optimal Q-values instead of computing Q-values directly using Eq. (5). Different types of artificial neural network, such as convolutional neural network (CNN) and recurrent neural network (RNN), are used to deal with very large state and action space (LeCun et al., 2015).

2.2.1 DRL algorithms

This section presents two of the most popular DRL algorithms in transportation research: deep Q-network (DQN) and deep deterministic policy gradient (DDPG). For DQN, two variants, namely double DQN and dueling DQN, are also reviewed. Readers interested in further algorithmic development can refer to methodological reviews (Li, 2018; François-Lavet et al., 2018; Arulkumaran et al., 2017) and online resources (Weng, 2020).

2.2.1.1 Deep Q-network (DQN)

Proposed by Minh et al. (2015), DQN uses a DNN (e.g., CNN) as the function approximator to approximate Q-value associated with a state-action pair. During training, DQN follows the ε-greedy strategy to choose an action between exploration and exploitation at each time step. A salient feature of DQN training is experience replay, which involves a replay memory $\boldsymbol{M}$ that stores the agent's experiences during training. An experience is associated with the agent taking an action at a given state and time step, observing the state transition, and getting a reward. Thus the experience is denoted as $e_t = (s_t, a_{t,} r_t, s_{t+1})$. Once $\boldsymbol{M}$ is full, the oldest experience is removed from $\boldsymbol{M}$ to create space for the next new experience.

In DQN, DNN is trained using a minibatch $U(M)$ of a randomly selected sample (experiences) from $\boldsymbol{M}$. The employment of experience replay with minibatch sampling brings several advantages. First, learning from random samples results in less correlation compared to learning directly from consecutive samples, which increases the learning efficiency. Second, experience replay gives greater data efficiency by allowing each experience to be used in many weight updates. Third, by averaging the behavior distribution over many previous states, experience replay contributes to smoothing out learning and avoiding oscillation or divergence in the parameters (Mnih et al., 2015).

After assigning a random weight θ to the DNN, for each experience the input (state s) is allowed to propagate through the DNN (first forward pass). The output $Q(s, a; \theta)$ is compared with the target optimal Q-value $Q_*(s_t, a_t)$ to estimate the loss. Ideally, based on the Bellman optimality equation, the expected value of this target Q-value should equal $r + \gamma \max_{a' \in \boldsymbol{A}} Q_*(s', a')$. Thus, the loss function can be written as:

$$\mathcal{L}(\theta) = E_{s,a,r,s' \sim U(M)} \left[\left(r + \gamma \max_{a' \in A} Q_*(s', a') - Q(s, a: \theta) \right)^2 \right] \quad (9)$$

In Eq. (9), the right-hand side has an unknown part $Q_*(s', a')$ which denotes the optimal Q-value of the next iteration. One way to approximate this unknown quantity is to do a second forward pass in the DNN before performing any gradient descent step, i.e., both first and second forward passes will be based on the same DNN weight parameter. In the second forward pass, the transitioned state s' of the corresponding experience is taken as input in the same DNN (i.e., same weight parameters θ) to predict state-action values $Q(s', a': \theta), \forall a' \in A$. After performing these two forward passes and assuming $Q(s', a'; \theta) \approx Q_*(s', a')$ in Eq. (9), the loss value can be calculated. However, there is a major drawback in this two-forward pass procedure. Since the second forward pass is done in the same network with the same network parameter θ, both Q-values and target Q-values will update in the same direction. As a result, the correlation between the Q-values and target Q-values can be high, which may cause oscillation or divergence of the policy during training.

To tackle this issue, Minh et al. (2015) propose a novel technique by creating a parallel network, called target network, which is structurally cloned of the original DNN. At the beginning of training, the target network parameter θ' is set to be the same as the original DNN's, i.e., θ. Unlike θ, θ' is kept frozen for a certain number of time steps before an update. Then, θ' is updated to the current value of the DNN parameter θ. The loss function is shown in Eq. (10), where $r + \gamma \max_{a' \in A} Q_*(s', a': \theta')$ denotes the target Q-value. Then, θ is updated through backpropagation and gradient descent.

$$\mathcal{L}(\theta) = E_{s,a,r,s' \sim U(M)} \left[\left(r + \gamma \max_{a' \in A} Q_*(s', a': \theta') - Q(s, a: \theta) \right)^2 \right] \tag{10}$$

Double DQN

Double DQN is a modified version of DQN and tabular double Q-learning algorithms (van Hasselt et al., 2010, 2016). To facilitate exposition, let y^{DQN} denote the target Q-value. In Eq. (10), y^{DQN} is:

$$y^{DQN} = r + \gamma \max_{a' \in \boldsymbol{A}} Q(s', a';\ \theta') \tag{11}$$

With this target Q-value, (greedy) action selection and action evaluation are performed using the same network with parameter θ', which may end up selecting overestimated policies that in turn leads to overoptimistic value estimates. The idea of double DQN is to decouple action evaluation from action selection, by using the target network as a second value function approximator. Consequently, the target Q-value is estimated as:

$$y^{DoubleDQN} = r + \gamma Q \left(s', \operatorname*{argmax}_{a' \in \boldsymbol{A}} Q\ (s', a'; \theta); \theta' \right) \tag{12}$$

In Eq. (12), the action selection uses greedy policy based on the Q-network with parameter θ. The evaluation of the action is based on the target network with parameter θ'. Numerical experimentation shows that double DQN can provide more stable learning and find better policies than DQN (van Hasselt et al., 2016).

Dueling DQN

Dueling DQN is another modified DQN that follows a dueling network architecture (Wang et al., 2016), which involve two sequences of fully connected convolution layers. The two sequences estimate the state value function and the advantage function separately. The advantage function $A^{\pi}(s,a)$, which is for an action $a{\sim}\pi(s)$ from a state s based on policy π, is the difference between the Q-value associated with this state-action pair $Q^{\pi}(s,a)$ and the state value function $V^{\pi}(s)$. Combining the estimates of the state value function and then advantage function from two separate sequences, a Q function is estimated as follows:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\boldsymbol{A}|}\sum_{a'} A(s,a';\theta,\alpha) \right) \tag{13}$$

where α and β are parameters of the two fully connected layers. The key idea behind the dueling network architecture is to avoid unnecessary estimation of the value of every action choice from a state. Sometimes, just knowing the value of the state suffices for the agent to identify the best action to take without knowing the individual value for every action choice. Dueling DQN can be combined with double DQN. Double dueling DQN with prioritized experience replay is the most updated variant of DQN.

2.2.1.2 Deep deterministic policy gradient (DDPG)

Deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015) is an actor-critic based algorithm which can operate on continuous action space. Based on the deterministic policy gradient algorithm (Silver et al., 2014), DDPG employs a parameterized actor function (which stores the current policy) with a parameterized critic function that approximates (using the Bellman optimality equation) and updates the value function using samples. In this way, DDPG can tackle large variance in policy gradients of actor-only methods. DDPG enables the agent to interact with the environment and employs gradient descent to improve the policy using a minibatch collected from replay memory. Using sampled policy gradient, the actor policy is updated using Eq. (14).

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_a Q(s,a;\theta^{Q})|_{s=s_i,a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s;\theta^{\mu})|_{s_i} \tag{14}$$

where N is the size of the minibatch and $\mu(s;\theta^{\mu})$ denotes the selected action based on the current policy. At the beginning of training, parameters of the actor and the critic networks are initialized as θ^{μ} and θ^{Q}. Parameters of the corresponding target networks (denoted as μ' and Q') are $\theta^{\mu'}$ and $\theta^{Q'}$. The critic network is updated using gradient descent on the loss function $\mathcal{L}(\theta^{Q})$:

$$\mathcal{L}(\theta^{Q}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - Q(s_i, a_i; \theta^{Q})\right)^2 \tag{15}$$

where

$$y_i = r_i + \gamma Q'\left(s_i', \mu\left(s_i'; \theta^{\mu'}\right); \theta^{Q'}\right) \tag{16}$$

DDPG suggests employing two target networks cloning the actor and critic networks. However, instead of directly copying policies (as done in DQN), DDPG uses a soft target update using Eqs. (17)-(18) where $\tau \ll 1$. The soft target update strategy enables more stable learning. The proposed soft target update is performed at regular intervals, using duplicate actor and critic networks by slowly maintaining the learned networks (target values).

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'} \tag{17}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'} \tag{18}$$

Apart from DDPG, a few other algorithms using the actor-critic architecture such as Proximal Policy Optimization (PPO) (Schulman et al., 2017), Asynchronous Advantage Actor-Critic (A3C; Mnih et al., 2016), and Advantage Actor-Critic (A2C; OpenAI, 2017) have also seen increasing use in transportation research.

2.2.2 Some extensions in DRL

2.2.2.1 Multi-agent systems

Sometimes, a system can have multiple agents each making its own decisions. It is possible for DRL to incorporate multiple agents to interact with the environment and learn simultaneously. However, given the interaction dynamics of the agents, the environment in a multi-agent system is not stationary and no longer retains Markov property (Tan, 1993; Laurent et al., 2011; Nowé et al., 2012). Instead, multi-agent RL/DRL problems are usually formulated as a stochastic game (or Markov game) (Littman, 1994; Busoniu et al., 2010). A stochastic game is characterized by $(S, \boldsymbol{A_1}, .., \boldsymbol{A_N}, P, R_1, .., R_N)$ where N is the number of agents; S denotes the state space of the environment; $\boldsymbol{A_i}$ $(i = 1, .., N)$ is the set of actions available to agent i; P is the state transition probability function; and R_i $(i = 1, .., N)$ is the reward function for agent i. Further introducing joint action set $\boldsymbol{A} = \boldsymbol{A}_1 \times \boldsymbol{A}_2 \times \cdots \boldsymbol{A}_N$, the state transition probability function can be defined as $P: S \times \boldsymbol{A} \rightarrow S$. At a time step t, state transition and reward depend on the joint actions $\boldsymbol{a}_t =$

$(a_{1,t}, a_{2,t}, \ldots, a_{N,t})$. Depending on specification of the agents' reward functions, a multi-agent system can be cooperative, competitive (adversarial), or mixed (Buşoniu et al. 2010; Hernandez-Leal et al., 2019).

2.2.2.2 Hierarchical DRL

As mentioned earlier, a drawback of RL is the curse of dimensionality (Barto and Mahadevan, 2003). Although DRL can deal with large state and action space, challenges remain as to performing DRL in a computationally efficient manner. By creating a hierarchy of policies, hierarchical RL (HRL) increases both learning efficiency and solution quality. By dividing policies into several sub-policies, the action space for a subpolicy becomes smaller which aids in better exploration of the environment. One of the earliest HRL uses is Feudal RL with a hierarchy of managers (Dayan and Hinton, 1993), where one level of managers can control sub-managers and assign a goal to each sub-manager. Meanwhile, these managers are controlled by super-managers. Another earlier use of HRL is the options framework (Sutton et al., 1999), which allows higher level policies to focus on goals and lower level sub-policies to focus on learning of controls. With the advent of DRL, several hierarchical DRL algorithms have been proposed. Kulkarni et al. (2016) integrate DQN with HRL and propose the h-DQN framework. A deep RNN based approach is proposed by Vezhnevets et al. (2016) to learn macro (high) level policies. Vezhnevets et al. (2017) propose FeUdal Networks by taking a long short-term memory (LSTM) network on top of a representation learned by a CNN as the baseline.

2.2.2.3 Asynchronous DRL

The robustness and efficiency of DRL training using DNN can be compromised due to the correlation between updates in the sequential process of learning. While Minh et al. (2015) propose experience replay in DQN to mitigate this, replay memory takes a toll on memory and computation power. As an alternative, asynchronous methods are proposed and can perform parallel and independent computing processes. Minh et al. (2016) propose asynchronous advantage actor-critic (A3C) which accommodates several actor-critic agents to learn parallelly and independently. Every agent acts on a different part of the environment with a different set of parameters. The updates from every agent are received by a global network and combined asynchronously to achieve a global policy. Note that although multiple agents work parallelly in A3C, A3C does not belong to multi-agent RL as agents in A3C are independent and do not interact. A synchronous and deterministic variant of A3C is known as advantage actor-critic (A2C). Minh et al. (2016)'s experiments show that A3C can achieve faster training than DQN.

2.2.2.4 Imitation learning and inverse RL

Imitation learning is a process of learning from demonstrations, also known as "apprenticeship learning". It is motivated by the following question: If the agent has no idea about the reward, how can the agent learn about the environment to find the best policy? Using a set of expert demonstrations (typically

defined by humans), the agent tries to learn the best policy imitating the experts' decisions. The expert demonstrations are provided in the form of trajectories $\tau = (s_0, a_0, s_1, a_1, \dots)$. One way to learn from expert demonstrations is to extract reward signals, known as inverse RL (Ng and Russell, 2000). In inverse RL, the agent first learns a reward signal from the expert demonstrations, and then uses this reward signal to find the optimal policy (Sutton and Barto, 2018). Readers may refer to Stadie et al. (2017), Hester et al. (2018), and Wulfmeier et al. (2015) for further details about imitation learning and inverse RL.

3 Deep reinforcement learning in transportation research

With a methodological background of DRL, in this section we conduct a comprehensive review of the literature on using DRL algorithms to address a variety of transportation problems. The existing work is grouped in seven categories by application domain: 1) autonomous driving; 2) energy efficient driving; 3) adaptive traffic signal control; 4) other types of traffic control; 5) vehicle routing optimization; 6) rail transportation; and 7) maritime transportation. For each application domain, our review focuses on the DRL tasks involved, how state and action space and reward are characterized, and the DRL algorithms used. To the extent possible, performance comparison of DRL with existing methods is also mentioned.

3.1 Autonomous driving

Autonomous driving consists of multiple tasks, including: sensing of surroundings, situation perception, action selection based on perception, strategic planning for execution of the selected action, and execution of the selected action (Talpaert et al., 2019; Kiran et al., 2020). The tasks are often divided into those at the higher level and those at the lower level. Higher-level tasks pertain to decision-making based on reasoning of the surrounding environment. Lower-level tasks relate to system control to execute the decision (Mirchevska et al., 2018; Chen et al., 2018). DRL has produced promising results at both levels.

3.1.1 Lane changing

Lane changing is a higher-level decision in autonomous driving. It pertains to the autonomous agent deciding whether to stay on the same lane or switch to a different lane, based on sensory inputs from the surrounding environment. Various DRL algorithms have been employed to design safe and efficient lane changing strategies, as shown in Table 1. Most existing work focuses on a single agent for an ego vehicle, while a few consider multi-agent systems (Shalev-Shwartz et al., 2016; Yi, 2018; Chen et al., 2018). Wang et al. (2019a) consider the environment as cooperative; Jiang et al. (2019) and Wang et al. (2020) characterize a system as both adversarial and cooperative with respect to surrounding vehicles. To deal with variable-size inputs, Huegle et al. (2019, 2020) incorporate a new DRL architecture called Deep Sets, which outperforms CNN and RNN based DRL.

The design of state space in existing lane changing studies is similar, including information on the absolute position and speed of the ego vehicle, relative positions and relative speeds of ego and surrounding vehicles, and distance and gap between vehicles. Depending on the number of vehicles considered, Chen et al. (2018) argue that the dimension of the state space can be up to 20 to be comprehensive enough for better decisions. For higher-level decision-making, the action space needs to have at least three actions: turn to the left lane, turn to the right lane, and keep the current lane. While most studies consider only these three lateral actions, a few further considers longitudinal actions, such as speed change by acceleration, deceleration, and keeping the current speed in DRL rather than relying on rule-based models.

Given that safety, comfort, and efficiency are the three most important criteria for autonomous driving, weighted attributing factors related to these criteria are used in the reward signal. Common attributing factors are: velocity maximization, collision avoidance, lane change completion, and safe distance keeping. Some additional factors, such as cooperation among surrounding vehicles (Wang et al., 2019a), discouragement of near-crash actions (Wang et al., 2020; Bai et al., 2019), and avoidance of unnecessary lane change (Alizadeh et al., 2019; Chen et al., 2018; Min et al., 2018; Makantasis et al 2019; Hoel et al., 2019), are also considered.

Higher-level decision-making cannot guarantee a collision-free trajectory without a transfer mechanism for the higher-level policy to be executed by lower-level motion control (Makantasis et al., 2019). For example, the higher level can decide to turn to the left lane. But in order to turn, the extent to which the steering angle needs to be changed needs to be determined by the lower level. To this end, several studies have combined higher-level decision-making and lower-level motion control in a hierarchical DRL architecture (Shi et al., 2019; Chen et al., 2019; Duan et al., 2020).

3.1.2 Motion control

Lower-level motion control concerns execution of the planned trajectory or decision taken at the higher level. Traditionally, vehicle motion control is achieved by model predictive control (Paden et al., 2016). Learning-based motion control is developed only recently. The actions involve longitudinal and lateral adjustments by changing vehicle acceleration and steering angle. Since actions are continuous, policy-based DRL algorithms in an actor-critic architecture are mostly used (Table 1). In designing the state space, most studies consider relative positions of the front and rear axles, and the current steering angle. Vinitsky et al. (2018a, 2018b) further include information of surrounding human-driven vehicle. Lin et al. (2019) design the state space differently with a continuous function featuring gap-keeping error and delayed acceleration.

Reward signal design has a considerable variation in motion control. Although it is common to give reward to a successful task and penalty to a failed one, the reward signal should reflect some overarching goal as well. For instance, focusing on longitudinal motions, Buechel et al. (2018) design the reward signal

with the longitudinal velocity difference. Vinitsky et al. (2018a) consider maximizing total throughput of a bottleneck in reward. In Folkers et al. (2019), the goal is to explore a parking lot without facing obstacles. As compared to the consideration of an overarching goal, a more recent approach is end-to-end autonomous driving, whose policies refer to derivation of control signal from the raw image pixel as input feature recorded by onboard cameras. A few related studies are shown in the last column of Table 1. The majority of the studies in both lane changing and motion control consider the state to be fully observable, with a few exception as shown in Table 1.

3.1.3 Miscellaneous tasks

Besides lane changing and motion control, other task-specific DRL applications also appear in the literature, such as car-following, intersection navigation, ramp merging, and even consideration of pedestrian safety. Relevant papers tackling these tasks are summarized in Table 2.

Table 1. Exsting DRL applications to lane changing and motion control in autonomous driving.

		Lane changing		**Motion control**	
		Lateral decision only	Lateral and longitudinal decisions	Conversion of higher-level decision to lower-level control	End-to-end driving
DQN and variants	DQN	Mirchevska et al. (2018) Feng et al. (2019) Jiang et al. (2019) Wang et al. (2019a, c) Chen et al. (2018) Alizadeh et al. (2019) Li and Czarnecki (2019)	Wolf et al. (2018) Min et al. (2018) Fayjie et al. (2018) Li and Czarnecki (2019) Ye et al. (2019)	Lee et al. (2019) (higher level), Chen et al. (2019b)	
	Double DQN		Zhang et al. (2019) Hoel et al. (2018) Nageshrao et. (2019) Makantasis et al. (2019)		
	Dueling DQN		Bai et al. (2019)		
Actor-critic architecture	DDPG	An and Jung (2019) Yi (2018)		Paxton et al. (2017) (lower level) Buechel et al. (2018) Lin et al. (2019) Bejar and Morán (2019) Lee et al. (2019) (lower level) Wang et al. (2018b)	Yu et al. (2018) Sallab et al. (2016, 2017)
	PPO		Ye et al. (2020)	Folkers et al. (2019)	
	TRPO			Vinitsky et al. (2018a)	
Other types	IRL	Sharifzadeh et al. (2016)	Wang et al. (2020) You et al. (2019)		
	HRL	Shi et al. (2019a) Chen et al. (2019a) Duan et al. (2020)	Shalev-Shwartz et al. (2016) Nosrati et al. (2018)	Xu et al. (2018)	
	Deep QL		Mukadam et al. (2017)	Paxton et al. (2017) (higher level), Wang et al. (2018a, 2019b)	
	AlphaGo		Hoel et al. (2019)		
	Policy-based DRL				Aradi et al. (2018)
State observability	Full	Mirchevska et al. (2018), Feng et al. (2019), Chen et al. (2018) An and Jung (2019) Wang et al. (2019a, c) Alizadeh et al. (2019) Shi et al. (2019a) Duan et al. (2020)	Ye et al. (2020) Zhang et al. (2019) Mukadam et al. (2017) Hoel et al. (2019) Nageshrao et. (2019) Bai et al. (2019) Fayjie et al. (2018) Makantasis et al. (2019) Wolf et al. (2018)	Paxton et al. (2017) Buechel et al. (2018) Lin et al. (2019) Xu et al. (2018) Vinitsky et al. (2018a) Folkers et al. (2019) Bejar and Morán (2019)	

Partial	Jiang et al. (2019) Chen et al. (2019a),	Hoel et al. (2019)	Lee et al. (2019)	Sallab et al. (2016, 2017)

Table 2. Summary of DRL applications to miscellaneous tasks in autonomous driving.

Task	Reference	Method	S.O.*	State space	Action	Reward
Car following	Zhu et al. (2018)	DDPG	Full	Following vehicle's speed, spacing, and velocity difference	Acceleration	Disparity between simulated and observed speed and spacing
	Zhu et al. (2019)	DDPG	Full	Following vehicle's speed, spacing, and velocity difference	Acceleration	Function of time to collision, headway, and acceleration change
	Wu et al. (2019a)	TRPO	Full	State of charge, distance, and speed of leader and follower (EV)	Acceleration	Function of distance and electricity consumption
	Qu et al. (2020)	DDPG	Full	Speed, gap, and relative speed with leader	Acceleration	Function of speed and time gap
	Bacchiani et al. (2019)	Multi-agent A3C	Full	Visual (space, obstacle, path) and numerical (speed, target speed, elapsed time ratio, distance to goal)	Acceleration, brake or maintaining same speed	Numerical reward for success and penalty for collision
Intersection navigation	Isele et al. (2018)	DQN	Full	Image snapshot indicating heading angle and velocity (unsignalized intersection)	Wait, move forward slowly, and go	Numerical reward for success and penalty for collision
	Zhou et al. (2019a)	DDPG	Full	Vehicle and signal specific information	Acceleration	Function of speed, gap and predicted arrival time at intersection
	Kashihara (2017)	Deep QL	Full	Image of roadway intersection (highway junction)	Moving up, down, left, and right	Numerical reward for success and penalty for collision
Ramp merging	Wang and Chan (2017)	DQN with LSTM	Parial	Speeds and positions of ego, gap front, and gap back vehicles	Acceleration and steering	Function of ego vehicle's acceleration, steering angle, speed, and distance to surrounding vehicles
	Wang and Chan (2018)	Deep QL	Full	Speeds and positions of ego, gap front, and gap back vehicles	Acceleration	Function of ego vehicle's acceleration, steering angle, speed, and distance to surrounding vehicles
	Nishi et al. (2019)	Passive actor-critic	Full	Speeds and positions of ego and gap back vehicles	Acceleration	Derived from value function and control dynamics due to action
	Nassef et al. (2020)	Dueling DQN	Full	Coordinates, speed, heading, acceleration, and size of ego, gap front, and gap back vehicles	Accelerate, decelerate, turn right, turn left, and do nothing	Inverse distance to merging point, and inverse of speed and acceleration
	Nishitani et al. (2020)	Double dueling DQN	Full	Images containing information on road shape, and ego and surrounding vehicles	Acceleration	Average speed of all vehicles from merging to reaching terminal area
Safety specific	Chae et al. (2017)	DQN	Full	Relative position of obstacle (pedestrian) and vehicle's speed	No brake, weak brake, mid brake, and strong brake	Combination of two penalties for early brake and actual collision

* S.O. means "state observability".

3.2 Energy efficient driving

Electric vehicles (EVs) have significant promise to reduce transportation fossil fuel use and emissions. Among different types of EVs, hybrid EVs (HEVs) combine the benefits of internal combustion engines and electric motors to reduce emission, at the same time addressing the low driving range issue. Designing an energy management system for HEVs, which controls the combined use of electricity and fossil fuel to achieve the best energy efficiency, is thus important. DRL has shown promising results in supporting energy management for various types of HEVs, including series-parallel plug-in hybrid electric bus (Wu et al., 2019b), power-split hybrid electric bus (Wu et al., 2018), hybrid electric tracked vehicle (Han et al., 2019), multiple battery based EV (Chaoui et al., 2018), hybrid electric bus (Tan et al., 2019, Li et al., 2019b), plug-in HEV (Hu et al., 2018), and series HEV (Li et al., 2018).

DRL research in HEV energy management aims to overcome some limitations of traditional rule- and optimization-based approaches, which require inputs of expert knowledge, comprehensive information on driving cycles and roads, and driving cycle prediction (Wu et al., 2019b). DRL trains the agent to optimally determine electricity-fuel split based on vehicle dynamics and vehicle-road interactions. By imposing rules during learning (e.g., always operating in the lower region of the brake-specific fuel consumption curve), DRL learning and energy performance (fuel economy and energy management stability) can be further enhanced (Lian et al., 2020). In terms of reward design, existing research considers energy use and savings, by keeping track of fuel consumption rate and battery state-of-charge (SoC) (Qi et al., 2017, 2019; Liessner et al., 2018a, 2018b, 2019; Li et al., 2018, 2019a, 2019b: Lian et al., 2020; Tan et al., 2019; Zhao et al., 2018; Chaoui et al., 2018; Han et al., 2019; Hu et al., 2018; Wu et al., 2018, 2019b). Change in trip distance has also been considered in reward design, in the context of HEV last-mile delivery (Wang et al., 2019d, 2019e).

In specifying the state space, the most important information for energy efficient driving includes vehicle dynamics (velocity and acceleration) and energy state (power demand and SoC), which are considered in all reviewed studies. Additional information has been incorporated as well. Tan et al. (2019) and Wu et al. (2019b) keep track of vehicle velocities from previous states. A more granular level of vehicle state representation, including wheel rotation and torque, gear configuration, battery temperature, and derating effect, is presented in Liessner et al. (2018, 2019). Apart from the information that is internal to the vehicle, the state space may include external information such as traveled distance (Wang et al., 2019d, 2019e; Wu et al., 2019b), distance to destination (Qi et al., 2019), expected future distance (in the context of vehicle touring) (Wang et al., 2019d, 2019e), and road conditions related to terrain and slope (Li et al., 2019b). All the reviewed DRL applications in energy efficient driving consider states to be fully observable.

The design of the action space focuses on optimizing energy use between different power sources. Some researchers consider changing power supply from internal combustion engines (Qi et al., 2019; Hu et al., 2018; Han et al., 2019; Wu et al., 2018, 2019b; Lian et al., 2020), while a few others look into changing energy output from the electric motor (Liessner et al., 2018a, 2019; Zhao et al., 2018). A third approach aims to maintain a balance among multiple inputs (e.g., multiple batteries) of the same power source (Li et al., 2019a, 2019b). To deal with continuous action space (changing power from different sources), actor-critic based DDPG algorithms are employed (Liessner et al., 2018a, 2018b; Li et al., 2019a, 2019b; Lian et al., 2020; Wang et al., 2019d; Tan et al., 2019; Wu et al., 2019b). Successes are also reported using DQN (Qi et al., 2017; Hu et al., 2018; Wu et al., 2018), dueling DQN (Qi et al., 2019; Li et al., 2018), and double Q-learning (Wang et al., 2019e; Han et al., 2019). DRL training is mainly done in simulated platforms, with some using real-world trip data (Qi et al., 2017, 2019; Wang et al., 2019d, 2019e). A summary of the DRL applications in energy efficient driving is provided in Table 3.

Table 3. Summary of DRL applications in energy efficient driving.

Vehicle type		Reference	Method	State	Action	Reward
Hybrid	HEV	Liessner et al. (2018a), (2018b)	DDPG	Vehicle dynamics, SoC, battery temperature	Power output for the electric motor	Negative of total energy used
		Lian et al. (2020)	DDPG with rule interposition	SoC, velocity, and acceleration	Continuous engine power	Fuel consumption of engine and the cost of battery charge
		Zhao et al. (2018)	Deep Q-learning	Power demand, predicted power demand, velocity, SoC	Discharge of battery pack, gear configuration	Fuel consumption
	Series HEV	Li et al. (2018), (2019a)	Dueling DQN (2018), DDPG (2019a)	Engine power, SoC, velocity, acceleration	Change in engine power	Function of fuel consumption rate and SoC
	HE Bus	Li et al. (2019b)	DDPG	Velocity (current and previous three second), acceleration, SoC, state of clutch, and road terrain and slope	Speed and torque (continuous), and four discrete powertrain modes	Fuel consumption rate
		Tan et al. (2019)	DRL with actor-critic architecture	SoC, velocities (current and previous three seconds), acceleration	Change in engine torque and rotational speed, traction motor torque and clutch state	Cost of fuel and electricity consumption
		Wu et al. (2018)	DQN	SoC, engine current power, velocity, and acceleration	Change in engine power	Function of fuel usage and SoC
	HE tracked	Han et al. (2019)	Double deep Q-learning	SoC, power demand, longitudinal and angular speed, and acceleration	Change in rotating speed	Function of fuel usage and SoC
Plug-in Hybrid	PHEV	Qi et al. (2017), (2019)	DQN (2017) DDQN (2019)	Power demand, SoC, distance to destination	Change in engine power	Function of power supply from internal combustion engine
		Hu et al. (2018)	Double Q-learning	SoC, required torque	Change in torque output	Function of fuel consumption
	PHE bus	Wu et al. (2019b)	DDPG	SoC, current past and future speed, acceleration, number of passenger, and traveled distance	Change in engine torque, engine rotational speed, and traction motor torque	Cost of fuel and electricity consumption
Electric	Extended range	Wang et al. (2019d), (2019e)	Double Q-learning (2019d), DDPG (2019e) with rule interposing	SoC, Fuel usage, location, travelled time and distance, and expected trip distance	Changing the expected trip distance	Function of fuel usage, SoC, change in trip distance and fuel usage compensating factor
	EV	Chaoui et al. (2018)	Double Q-learning	SoC of every batteries	Power splitting among batteries	Function of deviation of SoCs of all batteries

3.3 Adaptive traffic signal control

Adaptive traffic signal control makes signal timing decisions considering real-time traffic conditions at one or multiple intersections. The constantly changing dynamics of traffic means that adaptive traffic signal control is a challenging sequential decision-making problem with large search space. Applications of DRL in this domain have produced promising results, in the context of both a single intersection and a network of coordinated intersections. DQN is employed along with CNN which allows the state to be represented as a stack of images or an image-like grid. Besides DQN, actor-critic based architectures, such as DDPG (Canas, 2017), A2C (Coşkun et al., 2018; Chu et al., 2019), and PPO (Lin et al., 2018a), are used as well. Resaerch also integrates LSTM with policy networks to deal with partially observable environments (Shi and Chen, 2018; Chu et al., 2019) and continuous motion of vehicles (Choe et al., 2018). Table 4 summarizes existing DRL research on adaptive traffic signal control, categorized by attributes including problem setup, state and action space design, reward specification, and the algorithm used.

In general, a DRL agent of adaptive traffic signal control is trained so that it can optimally adjust traffic signal timing in real time. The reward function is designed considering vehicle waiting or delay time, queue length, and queue discharge. The most common characterization of state space is by including present vehicle positions and speeds, and signal phase in an image-like grid representation. Research also covers how to retrieve state information from raw pixels of intersection snapshots. Design of the action space depends on intersection configuration and complexity of the simulations. With no provisions for left-, right, and U-turns, the action space for a two-phase traffic signal system consists of only two actions: 1) green on east-west traffic; 2) green on north-south traffic. The action space becomes more elaborate for four-phase intersections. Continuous action space is investigated as well, where instead of changing the phase at every time step, the focus is on updating phase duration.

When adaptive traffic signal control is considered in a traffic network, intersection coordination is needed to reduce overall waiting time. In this case, multi-agent DRL is appropriate where each intersection is treated as an agent. Given that decisions are made at individual intersections, the design of state and action space for an intersection is similar to the case of a single intersection. The difference is how to combine rewards across intersections for coordination and cooperation, which is usually achieved by a centralized global optimization. Intersection coordination has also been approached by using just one single DRL agent. Interested readers may find additional information on both RL and DRL applications to traffic signal control in Haydari and Yilmaz (2020).

3.4 Other types of traffic control

Besides adaptive traffic signal control, DRL is used for other types of traffic control including: variable speed limit control, ramp metering, and lane pricing. For variable speed limit control, the action space consists of different speed limits that may be imposed. The state space captures the current traffic situation of the roadway, represented by multiple continuous or discrete variables such as traffic density and flow at various road sections (Nezafat, 2019; Ke et al., 2020a). The numbers of lanes in the upstream, mainline, and on-ramp merging sections as well as occupancy of these sections are also suggested to be part of the state space (Wu et al., 2020). The common objective of variable speed limit control is minimizing total travel time or traffic delay (Wu et al., 2020; Nezafat, 2019, Ke et al., 2020a). Other considered objectives include minimizing vehicle emission (Wu et al., 2020) and crash probability (Wu et al., 2020). Due to the continuous nature of the action space (speed limit), actor-critic based policy gradient algorithms such as DDPG (Wu et al., 2020) and A3C (Nezafat, 2019) are employed. Alternatively, when speed limits are treated as discrete, double DQN is used (Ke et al., 2020a). Results show that DRL outperforms state-of-the-art solutions using feedback control and Q-learning (Nezafat, 2019; Wu et al., 2020).

For ramp metering, DRL is reported to outperform state-of-the-practice ramp metering policy ALINEA and achieve precise adaptive highway ramp metering without model calibration (Belletti et al., 2017). Using a partial differential equation to simulate highway vehicle density, Belletti et al. (2017) adopt DNN integrated REINFORCE algorithm to approximate optimal control policies for adaptive highway ramp metering. For lane pricing, Pandey et al. (2020) apply DRL to dynamic tolling lanes. Considering the environment as a partially observable MDP, a policy gradient approach is employed that enables changing tolls with real-time observations. Tolls are modeled as continuous and stochastic variables, and are determined using a feedforward neural network. DRL is found to outperform feedback control heuristics by generating up to 9.5% higher revenues and reducing system travel time by up to 10.4%. A brief summary of DRL applications in other types of traffic control is included in Table 5.

Table 4. Summary of DRL applications in adaptive traffic signal control.

Attributes		Single intersection	Coordinated intersections
Problem setup	Single-agent setting	Gao et al. (2017), Genders and Razavi (2016), Li et al. (2016), Ha-li and Ke (2017), Liang et al. (2019), Muresan et al. (2019), Wan and Hwang (2018), Shabestary and Abdulhai (2018), Choe et al. (2018), Coşkun et al. (2018)	Canas (2017), Liu et al. (2018), Lin et al. (2018a)
	Multi-agent setting		Van der Pol and Oliehoek (2016), Liu et al. (2017), Calvo and Dusparic (2018), Shi and Chen (2018), Gong et al. (2019), Chu et al. (2019), Zhang et al. (2019), Ge et al. (2019)
State space	Raw pixels from the intersection snapshot	Mousavi et al. (2017), Garg et al. (2018)	
	Vehicular information	Gao et al. (2017), Genders and Razavi (2016), Li et al. (2016), Ha-li and Ke (2017), Liang et al. (2019), Muresan et al. (2019), Wan and Hwang (2018), Choe et al. (2018), Coşkun et al. (2018), Shabestary and Abdulhai (2018), Calvo and Dusparic (2018)	Van der Pol and Oliehoek (2016), Gong et al. (2019), Tan et al. (2019), Zhang et al. (2019), Lin et al. (2018a), Liu et al. (2017), Shi and Chen (2018), Liu et al. (2018), Calvo and Dusparic (2018), Ge et al. (2019),
	Others	Canas (2017)	Chu et al. (2019), Canas (2017)
Action space	Two-phase intersection	Li et al. (2016), Gao et al. (2017), Mousavi et al. (2017)	Van der Pol and Oliehoek (2016), Gong et al. (2019), Chu et al. (2019), Zhang et al. (2019), Lin et al. (2018a), Liu et al. (2017), Shi and Chen (2018), Ge et al. (2019), Calvo and Dusparic (2018), Liu et al. (2018)
	More than two phases	Genders and Razavi (2016), Wan and Hwang (2018), Choe et al. (2018), Coşkun et al. (2018), Shabestary and Abdulhai (2018), Calvo and Dusparic (2018)	Canas (2017)
	Phase update	Liang et al. (2019), Canas (2017)	Liu et al. (2017), Gong et al. (2019), Shi and Chen (2018), Calvo and Dusparic (2018), Ge et al. (2019)
Reward	Waiting time/delay	Genders and Razavi (2016), Gao et al. (2017), Choe et al. (2018), Mousavi et al. (2017), Shabestary and Abdulhai (2018), Wan and Hwang (2018), Liang et al. (2019)	Lin et al. (2018a), Ge et al. (2019)
	Queue length/discharge	Muresan et al. (2019)	Van der Pol and Oliehoek (2016), Tan et al. (2019),
	Combination of both	Li et al. (2016)	Chu et al. (2019), Zhang et al. (2019)
	Others	Canas (2017)	Canas (2017)
Algorithm	DQN	Gao et al. (2017), Li et al. (2016), Wan and Hwang (2018), Shabestary and Abdulhai (2018), Choe et al. (2018), Coşkun et al. (2018), Genders and Razavi (2016), Liang et al. (2019)	Van der Pol and Oliehoek (2016), Gong et al. (2019), Ge et al. (2019)
	DRL with CNN	Ha-li and Ke (2017), Muresan et al. (2019)	
	Actor-critic based	Coşkun et al. (2018), Canas (2017)	Canas (2017), Lin et al. (2018a), Chu et al. (2019)
	Others	Mousavi et al. (2017), Shi and Chen (2018), Garg et al. (2018), Calvo and Dusparic (2018)	Zhang et al. (2019), Shi and Chen (2018), Liu et al. (2017), Calvo and Dusparic (2018)
State observability	Full	Gao et al. (2017), Genders and Razavi (2016), Li et al. (2016), Mousavi et al. (2017), Garg et al. (2018), Ha-li and Ke (2017), Liang et al. (2019), Muresan et al. (2019), Wan and Hwang (2018), Shabestary	Van der Pol and Oliehoek (2016), Gong et al. (2019), Tan et al. (2019), Lin et al. (2018a), Liu et al. (2017), Ge et al. (2019) Liu et al. (2018)

		and Abdulhai (2018), Coşkun et al. (2018), Casas (2017), Calvo and Dusparic (2018)	
	Partial	Choe et al. (2018)	Zhang et al. (2019), Chu et al. (2019), Shi and Chen (2018),

Table 5: Summary of DRL applications in other types of traffic control.

Problem	Reference	Method	S.O.*	State	Action	Reward
Variable speed limit control	Nezafat (2019)	A3C	Full	Traffic density at the bottlekneck and network entrance	Selection of speed limit from a discritized predefined range	Delay reduction rate
	Ke et al. (2020a)	Double DQN	Full	Demand flow of upstream mainline, demand flow of on-ramp, density at the downstream bottleneck; density at the upstream area, and density on the on-ramp	Selection of speed limit from a discritized predefined range	Function of critical density and downstream density
	Wu et al. (2020)	DDPG	Full	Number of lane in upstream mainline and on-ramp, and occupancy rate	Selection of speed limit from a discritized predefined range	Funciton traffic flow, velocity at the bottleneck, and emission
Ramp metering	Belletti et al. (2017)	REINFORCE	Full	Traffic density simulated by a partial differntial equaiton	Selection of incoming flow from discretized predefined range	Negative of deviation from intended density
Lane pricing	Pandey et al. (2020)	Vanilla policy gradient, PPO	Partial	Current toll update and number of vehicles in all cells of a cell transmission model	Toll charged (continous but rounded)	Function of total travel time and total revenue

* S.O. means "state observability".

3.5 Vehicle routing optimization

Vehicle routing optimization is another area where the use and adaptation of DRL has been actively pursued, including both travelling salesman problems (Bello et al., 2016; Khalil et al., 2017; Kool et al., 2018) and vehicle routing problems (VRP) (Nazari et al., 2018; Kullman et al., 2019; Balaji et al., 2019; Zhao et al., 2020; Zhang et al., 2020; Peng et al., 2020; Yu et al., 2019; Chen et al., 2019c). Several new techniques are proposed to solve routing problems, including: 1) attention models based on an encoder-decoder architecture (Zhang et al., 2020; Peng et al., 2020; Nazari et al., 2018; Zhao et al., 2020); 2) a graph embedding network to represent the policy that captures the property of a node in the context of its graph neighborhood (Khalil et al., 2017); and 3) an Atari-fied representation of the environment (Kullman et al., 2019). Research has also been connected to practical contexts including urban freight deliveries and on-demand ridesharing for passenger transportation. In the following three paragraphs, existing DRL papers for urban freight delivery and on-demand ridesharing are reviewed. We also discuss a couple of papers applying DRL to vehicle holding control, which is important in bus operations and can be viewed a modified route optimization problem. A brief summary of the papers reviewed in this section is also included in Table 6.

Practical urban freight delivery problems often involve capacity constraints of delivery vehicles, distinction between pickup and delivery locations, and limited time windows for delivery. As a result, the complexity of routing problems is augmented compared to classic VRP. Nazari et al. (2018) consider a parameterized stochastic policy to solve VRP with limited vehicle capacity. A policy gradient DRL algorithm is applied to optimize parameters of the stochastic policy. Yu et al. (2019) opt for a distributed neural optimization strategy to solve a pickup and delivery problem with vehicle capacity and time window constraints. The authors adopt a graph embedded pointer network to progressively develop a complete tour for each vehicle. In a similar vein, Chen et al. (2019c) investigate a heterogeneous fleet of vehicles and drones for scheduling same-day delivery service. Balaji et al. (2019) solve an on-demand delivery driver model, which is essentially a VRP variant (stochastic dynamic VRP) using APE-X DQN algorithm. The results show superiority of DQN over traditional approaches. Most recently, Ahamed et al. (2020) propose DQN with proble-specific state representation, embedded heuristics, and rule-interposing to optimize crowdsourced urban parcel deliveries.

DRL based on-demand ridesharing problems are examined from three perspectives: 1) order dispatching, which aims to match rider requests with available vehicles (Wang et al., 2018c; Zhou et al., 2019b; Qin et al., 2020; Ke et al., 2020b; Tang et al., 2019); 2) vehicle repositioning, to proactively reposition idle vehicles from one zone to another zone to balance vehicle supply and rider demand (Al-Abbasi et al., 2019; Oda and Joe-Wong, 2018; Oda and Tachibana, 2018; Shi et al., 2019b; Liu et al., 2020;

Wen et al., 2017; Mao et al., 2020; Lin et al., 2018b); and 3) joint decisions on order dispatching and vehicle repositioning (Singh et al., 2019; Kullman et al., 2020; Holler et al., 2020; Jin et al., 2019; Liang et al., 2021). The majority of the existing works adopt a decentralized approach where each vehicle is considered a DRL agent with limited or no coordination with other vehicles, although exceptions exist in which a DRL agent plays the role of a central agency making routing decisions with a higher degree of coordination (Mao et al., 2020; Liu et al., 2020).

Among the on-demand ridesharing studies, the common elements in state representation are current location and destination of vehicles, time of the day, and rider demand and vehicle supply at the zonal level. Anticipated future demand is also considered in some studies (Wen et al., 2017; Al-Abbasi et al., 2019; Ke et al., 2020b; Kullman et al., 2020). More differences are present in the design of action space. For order dispatching, actions pertain to assigning requests to vehicles. Vehicle repositioning is concerned about moving empty vehicles between zones. Both types of actions are considered for joint decisions. The specification of reward varies, including minimizing waiting time (Wen et al., 2017; Singh et al., 2019), maximizing revenue (Wang et al., 2018c, Holler et al., 2019; Kullman et al., 2020; Qin et al., 2020), and minimizing idle/en-route time (Oda and Joe-Wong, 2018; Al-Abbasi et al., 2019).

Vehicle holding control is an important issue in bus operations that can mitigate bunching (multiple buses along the same route arriving at a stop at the same time). DRL has exhibited potential to address this issue to determine holding time for buses at different locations. Considering each bus as a single agent, multi-agent DRL is applied using prioritized double DQN (Alesiani et al., 2018) and PPO algorithms (Wang and Sun, 2020). The objective is to optimize forward and backward bus headway while minimizing holding time for every bus. The action pertains to determining holding time of each bus at every stop. In Wang and Sun (2020), a comprehensive state space is presented where passenger and vehicle headway related information is considered. The results show that DRL outperforms existing rule-based methods.

Table 6: Summary of DRL applications in vehicle routing optimization.

Problem type		Reference	Method	S.O.*	State characterization	Action	Reward
Urban freight delivery		Balaji et al. (2019)	DQN	Full	Pickup location, drivers and requests information	Wait, accept or pickup a request	Value of all delivered requests minus the cost
		Nazari et al. (2018)	Actor-critic	Full	Customer location and demand	Which node to visit next	Negative tour length
		Kullman et al. (2019)	D3QN	Full	Visual representation of vehicles and customers location	Wait at current location or instruct vehicle to visit to a customer node	Negative tour length
		Zhang et al. (2020)	Multi-agent DRL with attention	Full	Current vehicle location and remaining vehicle capacity	Decoder outputs the next customer to visit	Negative tour length
		Peng et al. (2020)	REINFORCE	Full	Partial solution instance and the features of each node using dynamic attention model	Which node to visit next	Negative tour length
		Yu et al. (2019)	A3C	Full	Available requests, charging stations, next stops of other vehicles in the system, battery charging demand of each vehicle	Which node to visit next	Function of number of delivery completion, travelled distance and penalties for constraint violation
		Zhao et al. (2020)	Actor-critic	Full	Vehicle location and capacity, customer demand and time windows	Which node to visit next	Negative of tour length
		Chen et al. (2019c)	DQN	Full	Time, location of customers, and set of planned routes	Whether a request is accepted, and if so, which vehicle or drone will provide service	Measured based on order assignment decision
On-demand ridesharing	Order dispatching	Wang et al. (2018c)	DQN	Full	Vehicle location and time	Assignment	Revenue from a trip
		Zhou et al. (2019b)	Deep Q-learning	Partial	Grid index, number of idle vehicles, number of requests, and distribution of requests' destinations	Source grid index, target grid index, order duration, and price	Function of accumulated driver income and request response rate
		Qin et al. (2020)	Double DQN	Full	Vehicle location and time of the day	Assignment of a request, or stay idle	Revenue
		Ke et al. (2020b)	Multi-agent DQN, PPO, A2C, actor-critic	Full	Number of unserved requests and idle vehicle, expected arrival rate of new requests and vehicle, request location, request waiting time and distance from matched vehicle	Assignment of a request to a vehicle, or delay the assignment	Function of delayed time
		Tang et al. (2019)	DRL	Full	Vehicle location, time of the day, supply-demand status, day of the week, driver service statics, holiday indicator	Request assignment or stay idle	Revenue from a trip
	Vehicle repositioning	Al-Abbasi et al. (2019)	DQN	Full	Vehicles' current status, number of vehicles and predicted future demand for a time slot	Whether the vehicle under study should pick up new riders; If yes, which zone to go to	Idle/en-route time and fuel use
		Oda and Joe-Wong (2018)	DQN	Full	Number of predicted requests and vehicles available in a region, vehicle location, time of the day, day of the week	Dispatch vehicles to different regions	Number of rejects and the vehicles' idle cruising time

Problem type		Reference	Method	S.O.*	State characterization	Action	Reward
		Oda and Tachibana (2018)	DQN	Partial	Individual vehicle's state by current location, current time, global supply and demand of a time interval	Dispatch agent vehicle to a target grid cell	Collected fare during a given time slot minus the working costs and cruising costs
		Shi et al. (2019b)	DQN	Full	Concealed location of vehicles with differential privacy, time and day, competition measure from nearby vehicles	Guide a cruising vehicle to one of the four adjacent cells	Profit made for a passenger
		Liu et al. (2020)	Double DQN		Vehicle location, supply and demand of each zone and neighboring zones	Repositioning of vehicle	Discrete value depending on supply-demand ratio in each zone
		Wen et al. (2017)	DQN	Full	Distribution of idle and in-service vehicles and predicted demand	Decision on rebalancing	Waiting time
		Mao et al. (2020)	Actor-critic	Full	Current time interval, waiting passenger demand, and available vehicle count	Repositioning of all vehicles	Negative of the total operational costs
		Lin et al. (2018b)	DQN, A2C	Full	Spatial distributions of available vehicles and requests	Allocating the agent to one of its six neighboring grids or staying in the current grid	Averaged revenue of all agents arriving at the same grid
	Joint decision	Singh et al. (2019)	DQN	Full	Vehicle status, predicted future demand and number of vehicles at each zone during a time interval	Which region to dispatch a vehicle	Waiting time, gap between supply demand, fleet size, and fuel consumption
		Kullman et al. (2020)	DQN, DDQN, D3QN	Full	Current time, requests' location, and vehicles information	Assignment, repositioning, or recharging decision	Base fare plus a charge proportional to the distance of the request
		Holler et al. (2020)	DQN, PPO	Full	Driver location, time to order/reposition completion	Single assignment or repositioning	Revenue
		Jin et al. (2019)	DDPG with RNN	Partial	Number of vehicles, number of relocatable vehicle, number of requests, entropy (function of vehicle availability), request features (price, duration)	Ranking a weight vector to rank and select the specific request or relocatable vehicle's destination	Sum of accumulated driver income and request response rate
		Liang et al. (2021)	DQN, A2C; Mean-field DQN and A2C	Full	Empty vehicle: current location and time Occupied vehicle: destination and time of arrival at the destination	Single assignment and repositioning	Profit in terms of trip fare and driving cost
Vehicle holding control		Alesiani et al. (2018)	Prioritized Double DQN	Partial	Departure time, arrival time and headway	Holding time (discretized predefined time)	Function of headway difference
		Wang and Sun (2020)	PPO	Full	Passenger and vehicle headway related information	Holding time of each bus at every stop	Forward and backward bus headway

* S.O. means "state observability".

3.6 Rail transportation

Using DRL, promising results have been reported in multiple rail transportation areas, including train timetable rescheduling (Ning et al., 2019; Obara et al., 2018; Wang et al., 2019f; Yang et al., 2019), automatic train operations (Zhou and Song, 2018; Zhou et al., 2020; Zhu et al., 2017), and train shunting operations (Peer et al., 2018). For each of the problems, we review relevant publications and provide a summary in Table 7. Train timetable rescheduling problems involve finding a feasible timetable of a train either by re-routing, re-ordering, re-timing, or canceling in case of uncertain disturbances associated with equipment/system failure along the railway line. In automatic train operations problems, the velocity and trajectory of a high-speed train need to be determined based on uncertain situations (e.g., changing trip time and waiting time due to passenger boarding). Train shunting operation problems pertain to matching incoming and outgoing train timetables as well as scheduling maintenance and cleaning activities in a shunting yard. The rail transportation problems are mainly approached by DQN and DDPG algorithms, with a fully observable environment. Researchers often resort to simulated environments. Very limited efforts try to incorporate real train track information in training (Peer et al., 2018; Yang et al., 2019; Zhou et al., 2020).

In the case of a disruptive event, the goal of train timetable rescheduling is to recover a train's operational order by readjusting its timetable while minimizing delay (Ning et al., 2019; Wang et al., 2019f), passenger dissatisfaction (Obara et al., 2018), or energy consumption (Yang et al., 2019). Ning et al. (2019) and Wang et al. (2019f) consider actual arrival and departure times as state, reordering of departure sequences as action, and negative average total delay as reward. Obara et al. (2018) describe train delay in a graph environment and consider graph deformation in action space specification. The authors consider delay, stoppage, driving time, frequency, and connection in the reward function. Focusing on reducing train energy consumption, Yang et al. (2019) design their reward function considering both recovered energy and traction energy. Ying et al. (2020) consider multiple trains in a train scheduling problem.

One of the objectives for automatic train operations is minimizing energy consumption (Zhou and Song, 2018; Zhu et al., 2017), subject to delay (Zhu et al., 2017), punctuality, and riding comfort (Zhou et al., 2020) constraints. In doing so, Zhou and Song (2018) and Zhou et al. (2020) define speed and train position as state. The magnitude of acceleration and deceleration is considered as action. In terms of reward specification, besides energy consumption, train delay is also included (Zhou et al., 2020). To minimize profile tracking error and energy consumption, Zhu et al. (2017) specify reward considering speed deviation from the target speed, based on which to make acceleration/deceleration decisions. For state characterization, the authors include speed, position of the train, relative position from the front train, rail wireless network strength, and a binary indicator of whether the train starts using the wireless network. To

tackle train shunting operation problems, Peer et al. (2018) use arrival and departure train conditions in the shunting yard to describe the state, and parking and departure decisions of each track as actions, to minimize the error in train parking and departure from the yard.

Table 7: Summary of DRL applications in rail transportation.

Application Area	Reference	Method	State	Action	Reward
Train timetable rescheduling	Obara et al. (2018)	DQN	Delay condition	Rescheduling timetable	Delay, stoppage, driving time, frequency of train
	Ning et al. (2019)	DQN	Actual arrival and departure times	Reordering of the departure sequences	Negative average of total delay
	Wang et al. (2019f)	Monte Carlo tree search	Actual arrival and departure times	Reordering of the departure sequences	Negative average of total delay
	Yang et al. (2019)	DDPG	Number of the departing train and its last dwelling time, current speed and position of the other running trains	Speed and the dwelling time of the departing train	Recover energy and the negative of traction energy
Automatic train operations	Zhu et al. (2017)	DQN	Speed, position of the train, relative position from the front train, rail wireless network strength, and a binary indicator of whether the train starts using the wireless network	Accelerate or decelerate and their magnatude	Speed deviation from the target speed
	Zhou and Song (2018)	DDPG	Speed and train position	Magnitude of acceleration and deceleration	Energy consumption
	Zhou et al. (2020)	DDPG	Speed and train position	Magnitude of acceleration and deceleration	Energy consumption and dealy
Train shunting operations	Peer et al. (2018)	DQN	Arrival and departure time	Parking and departure decisions	Correct parking and correct departure

3.7 Maritime transportation

Most DRL applications in maritime transportation are in the context of Autonomous Ship (AS) driving, more specifically in AS path following and collision avoidance. Given the dynamic and complex nature of AS driving, existing analytical methods such as model predictive control are often not suitable for practical applications (Zhao et al., 2019). DRL presents a promising alternative and has shown some success in solving AS path following and simultaneous path following and collision avoidance problems (Table 4).

For AS path following problems, actions considered include rudder angle (Martinsen and Lekkas, 2018), course angle (Woo et al., 2019), and rudder angle with propeller rotation (Rejaili and Figueiredo, 2018). Reward is specified to reflect the extent to which a ship deviates from a predefined path. Martinsen and Lekkas (2018) and Woo et al. (2019) define the reward function as cross-track error. Rejaili and Figueiredo (2018) define reward as the negative of penalty for deviation. Path guideline, distance from the ship mass center to the guideline, and the angle between the longitudinal axis of the ship and the guideline are considered in representing the state. Ship speed and angular velocity are further considered in Martinsen and Lekkas (2018) and Rejaili and Figueiredo (2018).

For simultaneous modeling of path following and collision avoidance, a number of control methods – including model-based and model-free – have been investigated. DRL is mostly applied in the context of model-free methods. One stream of research concerns static obstacle avoidance. Safety-related measures are specified in reward. For example, Sawada (2019) uses safe passing distance from obstacles, while some other researchers consider the number of collision instances (Amendola et al., 2019; Layek et al., 2017; Shen et al., 2019; Zhang et al., 2019). In terms of the state space, information on ship position, orientation, turning rate, and distance from the obstacle is always included. Distances from other ships and ship width and length are further considered in Shen et al. (2019) and Sawada (2019). Rudder action, heading angle, rudder angle, and turning rate are included in the action space (Layek et al., 2017; Amendola et al., 2019; Shen et al., 2019; Zhang et al., 2019; Sawada, 2019).

Another stream of research deals with both static and dynamic obstacles (such as environmental disturbances). Navigation decisions including rudder actions or course actions are made at every time step, which are informed by state of the AS including positions of the ship and obstacles. Also considered in the state space are speed (Cheng-bo et al., 2019; Wang et al., 2019), angular velocity (Cheng and Zhang, 2018; Zhao et al., 2019; Zhao and Roh, 2019), surge, and sway (Cheng and Zhang, 2018). Except for Zhao and Roh (2019), all studies consider navigation of a single ship using single-agent DQN.

Table 8. Summary of DRL applications in maritime transportation.

Application area		Reference	Method	State	Action	Reward
Path following		Martinsen and Lekkas (2018)	Actor-critic	Cross-track error, course and heading to the path, surge, sway, yaw rate, and derivatives of the path	Rudder angle	Negative cross-track error
		Rejaili and Figueiredo (2018)	DQN and DDPG	Distance and longitudinal axis of AS to guideline, horizontal and vertical speed, and angular velocity	Rudder angle and propeller rotation	Deviation from guideline and speed setpoint
		Woo et al. (2019)	DDPG	Position	Course angle	Deviation from route
Path following and collision avoidance	Static obstacles	Layek et al. (2017)	DDPG	Position, orientation, and actual turning rate	Turning rate	Angle between ship and obstacles
		Amendola et al. (2019)	DQN	Distance between ship and channel centerline, course over ground, turn rate, and last rudder level	Rudder angle	Proximity of ship to destination, collision avoidance, and deviation from center
		Sawada (2019)	PPO	Ship position, breadth, length, and speed	Heading and rudder angle	Safe passing distance and deviation from route
		Shen et al. (2019)	DQN	Distance from obstacles and other ships	Rudder angle	Obstacle avoidance
		Zhang et al. (2019)	DQN	AS speed and course, position of obstacle and target point, distance between ship and obstacle, and distance between AS and target point	Heading angle	Approach to target point and obstacle avoidance
	Static and dynamic obstacles	Cheng and Zhang (2018)	DQN	Ship position, heading, surge, sway, and angular rate, and obstacle position	Rudder angle	Obstacle avoidance and target approach
		Cheng-bo et al. (2019)	DQN	ship position, speed, and course, distance from target and obstacles, and obstacle position	Heading angle	Proximity of ship to destination, collision avoidance, and deviation from center
		Etemad et al. (2020)	DQN	Position	Heading angle	Approach to target point and obstacle avoidance
		Zhao and Roh (2019)	Multi-agent policy-based DRL	Ship position, velocity, and length, distance between current position and destination, and relative angle between ship and destination	Rudder angle	Proximity of ship to its destination, avoiding collision, and drift
		Wang et al. (2019g)	DQN	Speed, position of obstacles and destination, and distance of vessel from obstacles and destination	Rudder angle	Approach to target point, obstacle avoidance, and deviation from route
		Zhao et al. (2019)	Policy-based DRL	Ship position, velocity, and length, distance between current position and destination, and relative angle between ship and the destination	Rudder angle	Proximity of ship to its destination, avoiding collision, and drift
		Guo et al. (2020)	Actor critic	Longitude and latitude	Heading angle and speed	Approach to target point and obstacle avoidance
		Chen et al. (2020)	DQN	Image of the current position	Rudder angle	Approach to target point and obstacle avoidance
Port management		Shen et al. (2017)	DQN	Container loading condition at every time step	Scheduling container	Availability, reshuffling, and yard crane shifting

Besides AS path planning and collision avoidance, DRL has been tried for port management. Shen et al. (2017) employ DQN for quay and yard crane scheduling for Ningbo Port in China. Table 8 summarizes the reviewed studies for AS path following, path following and collision avoidance, and port management. The reviewed studies are categorized based on the application area, method used, and state, action, and reward specifications. Regardless of the application area and methodology used, the exiting literature of DRL applications in martime transportation always considers the environment to be fully observable.

4 Synthetic discussions

4.1 Applicability

While many DRL applications and adaptations have been reported as shown in section 3, there does not exist a single, universal rule for DRL system design to tackle all transportation problems. To ensure successful DRL use, one needs to have an in-depth understanding of the nature of the specific transportation problem investigated as well as DRL. In principle, a DRL algorithm improves policies by having the agent interact with the environment. As such, any problems whose solution can be improved by trial-and-error, i.e., incorporating feedback from the environment from one trial to the next, can be suitable for DRL. Also, problems that place importance to completing a full task rather than periodical success at intermediate steps and entail delayed reward are suitable for DRL. An example of this, as we have seen, is end-to-end autonomous driving where a collision before the end of the journey would overshadow earlier success in lane changing. To model sequential decision-making, MDP offers an adequate framework for a fully observable environment. Note that DRL can also be applied to environments with partially observable MDP, by incorporating RNN as a function approximator (Heess et al., 2015; Hausknecht et al., 2015).

It is difficult to conclude which DRL algorithm is best for a specific transportation problem, although candidate DRL algorithms can be identified. For example, problems where states can be incorporated as a stack of images, DRL algorithms embedding CNN as a function approximator (e.g., DQN) are appropriate. Problems like adaptive traffic signal control and end-to-end autonomous driving can adopt double dueling DQN with prioritized experience replay. Both policy based and actor-critic based algorithms can deal with continuous action space. The actor-critic architecture is preferred when optimal policy and its value are both expected. In this regard, algorithms like DDPG and PPO can be used for energy management of hybrid or electric vehicles and autonomous driving motion control. Asynchronous algorithms like A3C, which are also based on the actor-critic architecture, can accelerate learning by parallel processing. Although better solutions are generated using PPO in Atari gaming platform than using A3C, A3C may still be considered given its fast speed in learning. DRL algorithms with attention and pointer networks are popular for sequence-to-sequence modeling and adopted in vehicle routing optimization.

Based on our review of the existing literature, a mapping of DRL algorithm and extension use to the seven application domains is plotted in Fig. 4. The left column of the figure lists specific DRL algorithms. All the full names corresponding to the acronyms can be found in section 2. The right column lists the extensions. The middle column lists the application areas. The arrow widths are proportional to the number of applications found in the literature.

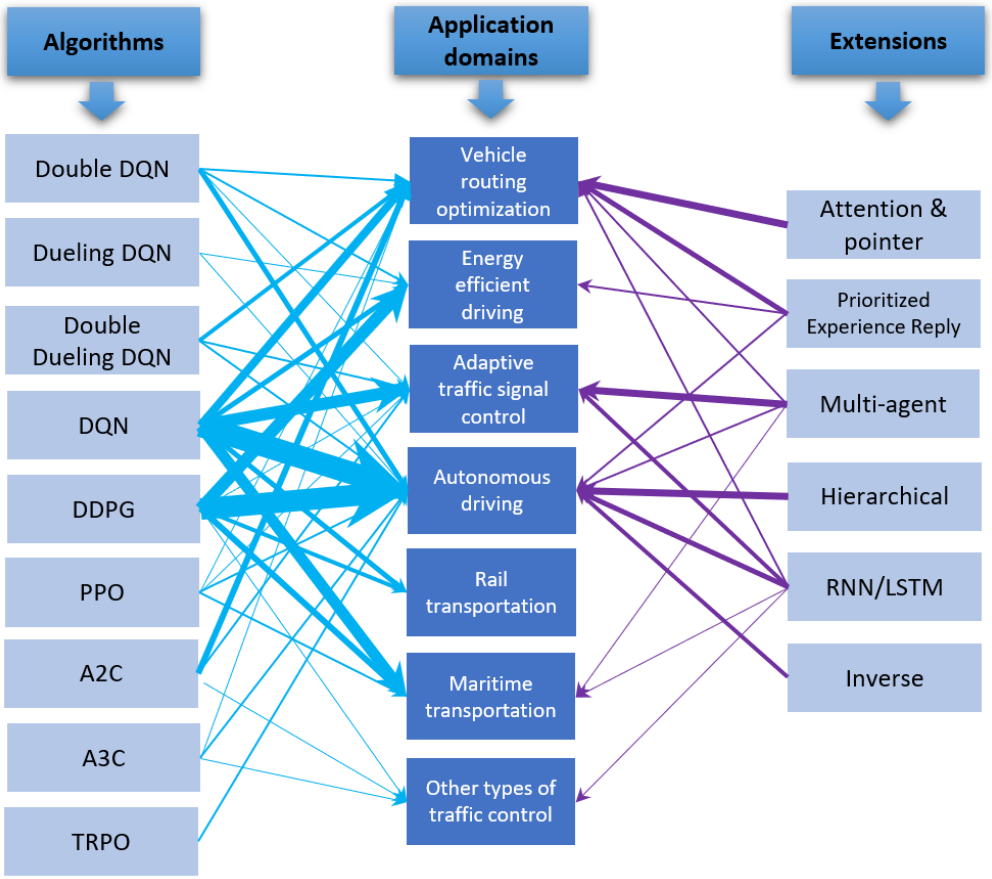

Figure 4. Mapping of DRL algorithm and extension use to transportation application domains.

Among the extensions used in DRL, three of them are worth highlighting: multi-agent, hierarchical, and inverse DRL. First, a multi-agent DRL framework is especially amenable to tackling problems in a distributed environment with agent interactions. As an example, a multi-agent framework can characterize cooperative and adversarial intentions of neighboring vehicles in autonomous driving. Another case of agent interactions which multi-agent DRL suits is traffic control among coordinated intersections.

Second, hierarchical DRL is useful when decisions can be decomposed into multiple layers. For instance, if the action space can be divided into two levels: "what to do" and "how to do", then a hierarchical framework can make the overall learning and implementation very efficient. In vehicle routing optimization problems, the type of neighborhood moves to select can be considered as a higher-level decision and how

to actually implement the selected neighborhood move as a lower-level decision. Similarly, in autonomous driving, the decision on what action to take can be viewed as a higher-level problem, while how to execute the action is a lower-level control problem.

Third, inverse DRL can be used for problems that require optimization of multiple factors through a common reward signal, which helps improving learning efficiency. In real-world problems, successful completion of a task may depend on several factors with a need for parameterization for the reward function. Weights of these parameters should be specified, but would involve intense tuning if done manually. For example, the decision on lane changing depends on whether the agent successfully completes the task, but also on marginal safety risk, smoothness, and comfortability of the lane change. Instead of explicitly defining a reward function at the beginning of training, inverse DRL offers a way to extract a proper reward signal to be used for seeking the optimal policy.

4.2 Strengths and shortcomings of DRL

This section discusses the strengths and shortcomings of DRL applications in tackling transportation problems. Among the most remarkable strengths of DRL are: 1) ability to generate high-quality solutions in a very short amount of time; and 2) generality and scalability in solving varying problem instances. On the other hand, DRL is not without drawbacks. Shortcomings include: stability in training, computation power requirement, and hyperparameter tuning. Below we provide further discussions.

Strengths: As its first strength, DRL can generate high-quality solutions fast. Through extensive training, promising performance of DRL has been reported in guiding autonomous decision tasks (Minh et al., 2015; Silver et al., 2016, 2017). The review in section 3 has shown that in many applications DRL produces results with superior quality to existing benchmarks while taking only a matter of seconds to yield good solutions, as compared to traditional methods such as heuristics which require considerably longer computation time. It should be noted that training of DRL algorithms can be done offline, and thus has little negative effect on problem solving time.

The second strength of DRL is its generality. While interacting with the environment, a DRL algorithm accumulates and learns from experiences by encountering and trying to solve different problems. Thus, once trained, DRL can produce sound solutions for varying problem instances including those not exactly encountered. The ability of DRL to remain useful in "an uncharted territory" presents an advantage over traditional optimization techniques. With deep learning further embedded, the scalability of DRL to solve problems of different sizes is also enhanced. This has been testified by the considerably better results obtained by DRL algorithms than existing benchmarks for problems with large state and action space (Minh et al., 2015) and even with continuous action space (Lillicrap et al., 2016; Schulman et al., 2017).

Shortcomings: A few shortcomings have been recognized in DRL training. The first one is training stability, particularly for DQN, which involves a target network updated in regular intervals. At each update, the values of different parameters could experience unstable jumps (Van der Pol and Oliehoek, 2016). Thus it may be difficult to see a clear trend of learning improvement. The second shortcoming is interpretability of DRL (Chakraborty et al., 2017; Zhang and Zhu, 2018). A lack of interpretability can make it difficult to understand contributions of each component in the state to the final solution. Furthermore, DRL algorithms are criticized for lack of reproducibility due to the use of simulated data (Henderson et al., 2018, Hoffman et al., 2020). Also, inadequate training (e.g., overtrained in a particular environment) can give rise to the issue of generality, which in turn compromises transferability of the DRL agent to other environments.

Another shortcoming of DRL – perhaps more of a challenge – is the significant computation power required in training. DRL training involves updating weights of an artificial neural network by hundreds of thousands of iterations over a large amount of data. The extensive training calls for very high computation power. With state-of-the-art GPU, parallel processing can be leveraged as an efficient way of updating weights in a neural network. But GPUs are expensive. Although deep learning platforms like Tensorflow and PyTorch have built-in functionality to optimize the use between CPU and GPU, a tradeoff between computational efficiency and financial capability needs to be made.

Apart from the above, an additional shortcoming of DRL pertains to hyperparameter tuning. Every DRL algorithm involves a certain number of hyperparameters to be tuned. This tuning process is generally very time consuming (Henderson et al., 2018), as it involves grid search of all hyperparameters. Given the large number of combinations of possible hyperparameter values, hyperparameter tuning and consequently neural network training are usually slow. For some problem instances, it can take days (Mnih et al., 2015; Yu et al., 2019).

4.3 Issues and future research directions

With the understanding of DRL's applicability, strengths, and shortcomings, this section discusses issues present in DRL application/adaptation for tackling transportation problems, and suggest potential future research directions. These discussions are organized in two parts. The first part focuses on issues and research directions of DRL application/adaptation that are common in transportation research. In the second part, we look into issues and research directions specific to different transportation domains.

4.3.1 Common issues and research directions

We highlight five common issues in DRL applications/adaptations to the transportation domain. The first one pertains to the use of simulated platforms and synthetic data in DRL training and testing. As real-world situations can be more complicated than simulated data, DRL algorithms trained based on simulated platforms and synthetic data need to be validated before applying to real scenarios with confidence. The

need for validation is particularly critical to applications for which safety is of paramount importance (e.g., driving related). It is possible that, no matter how many scenarios a DRL agent is exposed to in a simulated environment, the agent cannot fully learn the complexity of real-world situations. Thus, a crucial research direction is transferring training of a DRL agent from a simulated environment to the real world in a safe, secured, and efficient manner. For exmaple, the trained DRL agent may need to be first tested in a physical test bed and then in a controlled environment with rigorous fail safe, and satisfy existing benchmarks for safety protocol in every tested scenario, before moving to real-world implementation. So far, only limited efforts are made to transfer the trained DRL agent from a simulator to a physical test bed (Chalaki et al., 2019).

The second common issue is the need for common platforms for evaluating different DRL algorithms in solving specific types of transportation problems. With a common platform, the inputs and problem instances for DRL training and testing are standardized, making it possible to perform "apple-to-apple" comparison. For example, for shared taxi routing problems, comparing performance of different DRL algorithms based on the same service area, fleet of shared taxis, and rider request density and distribution. Having such common platforms is critical to understanding the advantages, limitations, and appropriateness of different DRL algorithms and helping inform algorithm choice for particular problems. However, as such common platforms remain lacking, research efforts are clearly needed for common platform development for DRL algorithm evaluation.

As the third common issue, the existing studies focus on decision-making of either vehicles or infrastructure (traffic signals, ramp metering, etc.). To our knowledge, no study exists in the literature that simultaneously considers decision-making of both vehicles and infrastructure. Given that vehicles and infrastructure are interdependent in transportation systems, fixing one part while optimizing the other would likely yield suboptimal outcomes compared to explicit decision-making on both. This is especially relevant to transportation, which is rapidly evolving towards ubiquitous connectivity between vehicles and infrastructure. Yet vehicles and infrastructure may still possess some degree of operational autonomy and independence. Thus, additional research is needed to better encompass the vehicle-infrastructure nexus in full scale. A multi-agent DRL setting embedding cooperation, competition, and feedback among agents will be of particular interest in this regard and warrant further investigation.

The fourth common issue is accommodation of constraints, which are common in transportation optimization and control problems. To our knowledge, most existing works deal with constraints by including penalty in the reward specification, which is appropriate if the constraints are "soft", meaning that they could be violated but incur a cost. However, for constraints that are "hard", i.e., constraints must be strictly met, introducing penalty to reward cannot eliminate constraint violation and thus is not a perfect

solution. To our knowledge, how to appropriately account for hard constraints has not attracted enough attention in the existing DRL research in the transportation domain. We note that an alternative way to deal with hard constraints is by masking the constraints while designing the training environment, to keep the exploration space away from constraint violation, as considered in autonomous driving (Mukadam et al., 2018) and VRP (Nazari et al., 2018). Nonetheless, more research is still needed to further explore masking and other techniques to cope with hard constraints in various transportation domains.

Finally, and especially from the practitioners' standpoint, it is important that solutions generated by DRL are robust to changes in data and the environment. This relates to the generality and transferability issues discussed earlier in section 4.2. To ensure robustness, comprehensive sensitivity analysis of a trained DRL agent to different data and/or environments is warranted. In this regard, a few attempts have been made, including in traffic signal control (Calvo et al., 2018; Li et al., 2016; Ge, 2019), autonomous driving (Duan et al., 2019; Feng et al., 2019; Hoel et al., 2018), rail transportation (Yang et al., 2019) and maritime transportation (Etemad et al., 2020). Sensitivity analysis should be extended to other domains. For example, DRL based energy efficient driving may consider a variety of vehicle driving cycles. For parcel delivery and vehicle dispatching, a trained agent is expected be insensitive to changes in customer demand within a reasonable range. If a trained agent turns out to be sensitive, then one needs to reexamine the DRL model specification and training, including whether the design of the environment is appropriate and whether the training data are adequate and sufficient (so that the DRL agent is not undertrained).

4.3.2 Application-specific issues and research directions

For autonomous driving, three issues are identified. First, some actions (such as acceleration and speed) are treated as discrete, which is less realistic. Using policy based or actor-critic based algorithms, these actions can be considered as continuous, but further explorations are needed. Second, sensing and perceiving surrounding vehicles is critical. The existing literature has not paid much attention to distinguish human-driven and autonomous vehicles, which have different behavior and are likely to coexist in transportation systems for the foreseeable future (Noruzoliaee et al., 2018; Noruzoliaee and Zou, 2021; Zou et al., 2021). Research to refine the environment design needs to take this into account. Third, while existing research mostly deals with one or two specific aspects of autonomous driving such as lane changing, motion control, and collision avoidance, future modeling needs to be more comprehensive to ensure that DRL-driven decision-making leads to safe, reliable, and efficient autonomous driving.

For energy efficient driving, an important issue is that the reward signal is usually designed based on engine power supply. The relationship between fuel economy and engine power is complex. Most existing work lacks proper characterization of the relationship (Hu et al., 2018). Moreover, the success of energy management for an HEV depends not only on the state and dynamics of the vehicle and the surrounding

environment, but on other factors that are paid less attention to, such as trip distance, trip time, average achievable speed during the trip, and driver's behavior that may vary during the trip. Precise information on such factors may not be readily available at the time of decision-making, so some estimation or expectations need to be made. Also, given that plug-in HEVs are gaining increasing popularity, more efforts should be directed to energy management for plug-in HEVs.

For adaptive traffic signal control, three issues are identified. The first issue pertains to design of the state space. Existing research does not adequately reflect vehicle acceleration and deceleration for incoming traffic during yellow and red phases. This issue indeed is highlighted in an early adaptive traffic signal control study (Genders and Razavi, 2016), but so far has not been addressed. The second issue relates to research using a grid representation of traffic (a grid taking value 1 indicates vehicle presence and 0 otherwise), which usually assumes a uniform intersection environment with the same number of lanes at every entry and exits, and the same occupied space and inter-vehicle spacing on each lane. This may not be realistic in practice. Third, while most studies focus on optimizing system efficiency (e.g., minimizing overall wait time at intersections), equity consideration is largely absent but equally important to ensure all vehicles are fairly treated. More research is needed to address this issue.

In the case of other types of traffic control, we suggest two directions for further exploration. First, future DRL research for variable speed limit control could consider separate speed limit thresholds for different lanes of a roadway. For dynamic lane pricing, the existing literature assumes a macroscopic traffic flow simulation that cannot capture the impact of lane changes on traffic. Instead, microscopic traffic flow simulations have the potential to overcome this issue and thus could be considered in problem design.

Apart from autonomous driving, energy efficient driving, and traffic control, DRL applications in vehicle routing optimization, rail, and maritime transportation also face some issues worth further investigations. For vehicle routing optimization, as multiple objectives, e.g., minimizing total travel time, minimizing unserved requests, or minimizing customer wait time, are involved, adequately specifying reward is important. in addition, more methodological explorations are needed to enable and improve coordinated routing of a fleet of vehicles. For rail transportation, scalability and lack of well-defined benchmarks are two unresolved critical issues for assessing quality of DRL solutions. For maritime transportation, environmental disturbances such as ocean current and wind speed can significantly affect ship operations and should be included in state space design. Another promising direction is energy efficient ship navigation, which can be informed by the body of similar work for ground vehicles.

5 Available resources for DRL research

Finally, as DRL implementation is not a straightforward process, it is worth mentioning the available resources that can be leveraged in DRL research. Existing built-in platforms that facilitate development of new DRL algorithms or enable use and adaptation of existing DRL algorithms are of particular interest. Among them, OpenAI Baselines is a set of high-quality implementations of DRL algorithms which allow researchers to replicate different existing DRL algorithms, and refine and identify improvement ideas (Dhariwal et al., 2017). Stable Baselines presents a significant improvement upon OpenAI Baselines, featuring a platform for almost all DRL algorithms (Hill et al, 2017). Apart from OpenAI Baselines and Stable Baselines, TensorForce provides a framework for DRL which is built on deep learning library Tensorflow with several algorithm implementations (Kuhnle, 2017). In TensorFlow library, Tensorflow Agents is a versatile RL platform where an agent takes charge of two main responsibilities: 1) defining a policy to interact with the environment; and 2) learning the policy from collected experience (Guadarrama et al., 2018). KerasRL implements some state-of-the-art RL algorithms in Python and seamlessly integrates the algorithms with deep learning library Keras (Plappert, 2016). Also, deep Q-learning, policy gradients, and Q-value policy gradients algorithm for small to medium scale research have been implemented in rlpyt (Stooke and Abbeel, 2019). Table 5 provides a summary of the aforementioned platforms, the available DRL algorithms, and the library used by each platform.

Table 9. Summary of platforms for DRL development, available DRL algorithms, and libraries used.

Platform	Available DRL algorithms	Library used
OpenAI Baselines	A2C, Actor-Critic with Experience Replay, Actor-Critic using Kronecker-Factored Trust Region, DDPG, DQN, Generative Adversarial Imitation Learning (GAIL), Hindsight Experience Replay, TRPO, PPO,	Tensorflow
Stable Baselines	A2C, ACER, ACKTR, DDPG, GAIL, HER, PPO, TRPO, Soft Actor-Critic (SAC), Double Dueling DQN with prioritized experience replay	Tensorflow
TensorForce	Dueling and double DQN, Vanilla Policy Gradient (PG), Continuous DQN, A2C, A3C, TRPO, PPO	Tensorflow
Tensorflow Agents	DQN, double DQN, DDPG, Twin Delayed DDPG, Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning, PPO, SAC	Tensorflow
KerasRL	Dueling and double DQN, DDPG, Continuous DQN, Cross-Entropy Method, Deep SARSA	Keras
rlpyt	A2C, PPO, Dueling and double DQN, DDPG, Twin Delayed DDPG, SAC	PyTorch

6 Conclusion

Although the introduction of DRL to transportation research is still nascent, the rapidly growing body of literature has clearly demonstrated an interdisciplinary prospective of DRL applications and adaptation

in tackling a variety of transportation problems, especially those with a sequential decision nature. In this paper, a comprehensive and synthetic review of recent DRL research related to solving transportation problems is conducted. As the review mainly targets the general audience of the transportation community, we start by providing a methodological background of DRL, including the basic algorithms and extensions. Then we delve into reviewing specific applications and adaptations of DRL in transportation in seven identified domains, namely autonomous driving, energy efficient driving, adaptive traffic signal control, other types of traffic control, vehicle routing optimization, rail transportation, and maritime transportation. Built on the detailed review, a synthetic discussion on the applicability, strengths, and shortcomings of DRL as it pertains to transportation research is provided. The discussion also identifies common and application-specific issues, based on which future research directions are suggested. Finally, we provide information on the available platforms and their features for actual DRL implementation. We hope that this review will serve as a useful reference for the transportation research community to better understand what have been accomplished to date and what are the issues, prospects, and potentials of DRL for transportation, to stimulate further research in this exciting area.

Acknowledgement

This work presented in this paper was funded in part by the National Science Foundation CMMI-1663411. The support is gratefully acknowledged.

Reference

1. Ahamed, T., Zou, B., Farazi, N., and Tulabandhula, T. (2021). Deep reinforcement learning for crowdsourced urban delivery: System states characterization, heuristics-guided action choice, and rule-interposing integration. arXiv preprint arXiv:2011.14430.
2. Al-Abbasi, A. O., Ghosh, A., and Aggarwal, V. (2019). Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, 20(12), 4714-4727.
3. Alesiani, F., & Gkiotsalitis, K. (2018). Reinforcement learning-based bus holding for high-frequency services. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 3162-3168). IEEE.
4. Alizadeh, A., Moghadam, M., Bicer, Y., Ure, N. K., Yavas, U., and Kurtulus, C. (2019). Automated Lane Change Decision Making using Deep Reinforcement Learning in Dynamic and Uncertain Highway Environment. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC) (pp. 1399-1404). IEEE.
5. Amendola, J., Tannuri, E. A., Cozman, F. G., and Reali Costa, A. H. (2019). Port Channel Navigation Subjected to Environmental Conditions Using Reinforcement Learning. In International Conference on Offshore Mechanics and Arctic Engineering (Vol. 58844, p. V07AT06A042). American Society of Mechanical Engineers.
6. An, H., and Jung, J. I. (2019). Decision-making system for lane change using deep reinforcement learning in connected and automated driving. Electronics, 8(5), 543.
7. Aradi, S., Becsi, T., and Gaspar, P. (2018). Policy gradient based reinforcement learning approach for autonomous highway driving. In 2018 IEEE Conference on Control Technology and Applications (CCTA) (pp. 670-675). IEEE.
8. Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). A brief survey of deep reinforcement learning. arXiv preprint arXiv:1708.05866.
9. Bacchiani, G., Molinari, D., and Patander, M. (2019). Microscopic traffic simulation by cooperative multi-agent deep reinforcement learning. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (pp. 1547-1555). International Foundation for Autonomous Agents and Multiagent Systems.
10. Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
11. Bai, Z., Shangguan, W., Cai, B., and Chai, L. (2019). Deep Reinforcement Learning Based High-level Driving Behavior Decision-making Model in Heterogeneous Traffic. In 2019 Chinese Control Conference (CCC) (pp. 8600-8605). IEEE.
12. Balaji, B., Bell-Masterson, J., Bilgin, E., Damianou, A., Garcia, P.M., Jain, A., Luo, R., Maggiar, A., Narayanaswamy, B. and Ye, C. (2019). ORL: Reinforcement Learning Benchmarks for Online Stochastic Optimization Problems. arXiv preprint arXiv:1911.10641.
13. Barto, A. G., and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. Discrete event dynamic systems, 13(1-2), 41-77.
14. Bejar, E., and Morán, A. (2019). Reverse parking a car-like mobile robot with deep reinforcement learning and preview control. In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 0377-0383). IEEE.
15. Belletti, F., Haziza, D., Gomes, G., and Bayen, A. M. (2017). Expert level control of ramp metering based on multi-task deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, 19(4), 1198-1207.
16. Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. (2016). Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.
17. Brackstone, M., and McDonald, M. (1999). Car-following: a historical review. Transportation Research Part F: Traffic Psychology and Behaviour, 2(4), 181-196.
18. Buechel, M., and Knoll, A. (2018). Deep reinforcement learning for predictive longitudinal control of automated vehicles. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 2391-2397). IEEE.
19. Buşoniu, L., Babuška, R., and De Schutter, B. (2010). Multi-agent reinforcement learning: An overview. In Innovations in multi-agent systems and applications-1 (pp. 183-221). Springer, Berlin, Heidelberg.
20. Calvo, J. A., & Dusparic, I. (2018). Heterogeneous Multi-Agent Deep Reinforcement Learning for Traffic Lights Control. In AICS (pp. 2-13).
21. Casas, N. (2017). Deep deterministic policy gradient for urban traffic light control. arXiv preprint arXiv:1703.09035.
22. Chae, H., Kang, C. M., Kim, B., Kim, J., Chung, C. C., and Choi, J. W. (2017). Autonomous braking system via deep reinforcement learning. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (pp. 1-6). IEEE.
23. Chakraborty, S., Tomsett, R., Raghavendra, R., Harborne, D., Alzantot, M., Cerutti, F., Srivastava, M., Preece, A., Julier, S., Rao, R.M. and Kelley, T.D. (2017). Interpretability of deep learning models: a survey of results. In 2017 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computed, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI) (pp. 1-6). IEEE.

24. Chalaki, B., Beaver, L., Remer, B., Jang, K., Vinitsky, E., Bayen, A., and Malikopoulos, A. A. (2019). Zero-shot autonomous vehicle policy transfer: From simulation to real-world via adversarial learning. arXiv preprint arXiv:1903.05252.
25. Chaoui, H., Gualous, H., Boulon, L., and Kelouwani, S. (2018). Deep reinforcement learning energy management system for multiple battery based electric vehicles. In 2018 IEEE Vehicle Power and Propulsion Conference (VPPC) (pp. 1-6). IEEE.
26. Chen, C., Ma, F., Liub, J., Negenborn, R. R., Liu, Y., & Yan, X. (2020). Controlling a cargo ship without human experience based on deep Q-network. Journal of Intelligent & Fuzzy Systems,vol. 39, no. 5, pp. 7363-7379.
27. Chen, C., Qian, J., Yao, H., Luo, J., Zhang, H., and Liu, W. (2018). Towards comprehensive maneuver decisions for lane change using reinforcement learning. NIPS Workshop on Machine Learning for Intelligent Transportation Systems (MLITS).
28. Chen, I. M., Zhao, C., and Chan, C. Y. (2019b). A Deep Reinforcement Learning-Based Approach to Intelligent Powertrain Control for Automated Vehicles. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC) (pp. 2620-2625). IEEE.
29. Chen, X., Ulmer, M. W., and Thomas, B. W. (2019c). Deep Q-Learning for Same-Day Delivery with a Heterogeneous Fleet of Vehicles and Drones. arXiv preprint arXiv:1910.11901.
30. Chen, Y., Dong, C., Palanisamy, P., Mudalige, P., Muelling, K., and Dolan, J. M. (2019a). Attention-based Hierarchical Deep Reinforcement Learning for Lane Change Behaviors in Autonomous Driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 0-0).
31. Cheng, Y., and Zhang, W. (2018). Concise deep reinforcement learning obstacle avoidance for underactuated unmanned marine vessels. Neurocomputing, 272, 63-73.
32. Cheng-bo, W. A. N. G., Xin-yu, Z. H. A. N. G., Jia-wei, Z. H. A. N. G., Zhi-guo, D. I. N. G., and Lan-xuan, A. N. (2019). Navigation behavioural decision-making of MASS based on deep reinforcement learning and artificial potential field. In Journal of Physics: Conference Series (Vol. 1357, No. 1, p. 012026). IOP Publishing.
33. Choe, C. J., Baek, S., Woon, B., & Kong, S. H. (2018). Deep Q Learning with LSTM for Traffic Light Control. In 2018 24th Asia-Pacific Conference on Communications (APCC) (pp. 331-336). IEEE.
34. Chu, T., Wang, J., Codecà, L., and Li, Z. (2019). Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Transactions on Intelligent Transportation Systems.
35. Coşkun, M., Baggag, A., & Chawla, S. (2018). Deep reinforcement learning for traffic light optimization. In 2018 IEEE International Conference on Data Mining Workshops (ICDMW) (pp. 564-571). IEEE.
36. Dayan, P., and Hinton, G. E. (1993). Feudal reinforcement learning. In Advances in neural information processing systems (pp. 271-278).
37. Deisenroth, M. P., Neumann, G., and Peters, J. (2013). A survey on policy search for robotics. now publishers.
38. Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y. and Zhokhov, P., (2017). Openai baselines.
39. Duan, J., Li, S. E., Guan, Y., Sun, Q., and Cheng, B. (2020). Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data. IET Intelligent Transport Systems.
40. Etemad, M., Zare, N., Sarvmaili, M., Soares, A., Machado, B. B., & Matwin, S. (2020, May). Using Deep Reinforcement Learning Methods for Autonomous Vessels in 2D Environments. In Canadian Conference on Artificial Intelligence (pp. 220-231). Springer, Cham.
41. Fayjie, A. R., Hossain, S., Oualid, D., and Lee, D. J. (2018). Driverless car: Autonomous driving using deep reinforcement learning in urban environment. In 2018 15th International Conference on Ubiquitous Robots (UR) (pp. 896-901). IEEE.
42. Feng, X., Hu, J., Huo, Y., and Zhang, Y. (2019). Autonomous Lane Change Decision Making Using Different Deep Reinforcement Learning Methods. In CICTP 2019 (pp. 5563-5575).
43. Folkers, A., Rick, M., and Büskens, C. (2019). Controlling an Autonomous Vehicle with Deep Reinforcement Learning. In 2019 IEEE Intelligent Vehicles Symposium (IV) (pp. 2025-2031). IEEE.
44. François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018). An introduction to deep reinforcement learning. arXiv preprint arXiv:1811.12560.
45. Gao, J., Shen, Y., Liu, J., Ito, M., and Shiratori, N. (2017). Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network. arXiv preprint arXiv:1705.02755.
46. Garg, D., Chli, M., and Vogiatzis, G. (2018). Deep reinforcement learning for autonomous traffic light control. In 2018 3rd IEEE International Conference on Intelligent Transportation Engineering (ICITE) (pp. 214-218). IEEE.
47. Ge, H., Song, Y., Wu, C., Ren, J., & Tan, G. (2019). Cooperative deep Q-learning with Q-value transfer for multi-intersection signal control. IEEE Access, 7, 40797-40809.
48. Genders, W., and Razavi, S. (2016). Using a deep reinforcement learning agent for traffic signal control. arXiv preprint arXiv:1611.01142.
49. Gong, Y., Abdel-Aty, M., Cai, Q., and Rahman, M. S. (2019). Decentralized network level adaptive signal control by multi-agent deep reinforcement learning. Transportation Research Interdisciplinary Perspectives, 1, 100020.
50. Grondman, I., Busoniu, L., Lopes, G. A., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(6), 1291-1307.

51. Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Harris, C., Vanhoucke, V. and Brevdo, E., 2018. TF-Agents: A library for reinforcement learning in tensorflow.
52. Guo, S., Zhang, X., Zheng, Y., and Du, Y. (2020). An Autonomous Path Planning Model for Unmanned Ships Based on Deep Reinforcement Learning. Sensors, 20(2), 426.
53. Ha-li, P., and Ke, D. (2017). An intersection signal control method based on deep reinforcement learning. In 2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA) (pp. 344-348). IEEE.
54. Han, X., He, H., Wu, J., Peng, J., and Li, Y. (2019). Energy management based on reinforcement learning with double deep Q-learning for a hybrid electric tracked vehicle. Applied Energy, 254, 113708.
55. Hausknecht, M., and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. In 2015 AAAI Fall Symposium Series.
56. Haydari, A., & Yilmaz, Y. (2020). Deep reinforcement learning for intelligent transportation systems: A survey. IEEE Transactions on Intelligent Transportation Systems. DOI: 10.1109/TITS.2020.3008612.
57. Heess, N., Hunt, J. J., Lillicrap, T. P., and Silver, D. (2015). Memory-based control with recurrent neural networks. arXiv preprint arXiv:1512.04455.
58. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).
59. Hernandez-Leal, P., Kartal, B., and Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. Autonomous Agents and Multi-Agent Systems, 33(6), 750-797.
60. Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Dan, H., Quan, J., Sendonaris, A., Osband, I., Dulac-Arnold, G., Agapiou, J., Leibo, J. Z., Gruslys, A. (2018). Deep q-learning from demonstrations. In Thirty-Second AAAI Conference on Artificial Intelligence.
61. Hill, A., Raffin, A., Ernestus, M., Gleave, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M. and Radford, A., 2018. Stable baselines. GitHub repository.
62. Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
63. Hoel, C. J., Driggs-Campbell, K., Wolff, K., Laine, L., and Kochenderfer, M. (2019). Combining planning and deep reinforcement learning in tactical decision making for autonomous driving. IEEE Transactions on Intelligent Vehicles.
64. Hoel, C. J., Wolff, K., and Laine, L. (2018). Automated speed and lane change decision making using deep reinforcement learning. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 2148-2155). IEEE.
65. Hoffman, M., Shahriari, B., Aslanides, J., Barth-Maron, G., Behbahani, F., Norman, T., Abdolmaleki, A., Cassirer, A., Yang, F., Baumli, K. and Henderson, S. (2020). Acme: A Research Framework for Distributed Reinforcement Learning. arXiv preprint arXiv:2006.00979.
66. Holler, J., Vuorio, R., Qin, Z., Tang, X., Jiao, Y., Jin, T., Singh, S., Wang, C. and Ye, J. (2019). Deep Reinforcement Learning for Multi-Driver Vehicle Dispatching and Repositioning Problem. arXiv preprint arXiv:1911.11260.
67. Hu, Y., Li, W., Xu, K., Zahid, T., Qin, F., and Li, C. (2018). Energy management strategy for a hybrid electric vehicle based on deep reinforcement learning. Applied Sciences, 8(2), 187.
68. Huegle, M., Kalweit, G., Mirchevska, B., Werling, M., and Boedecker, J. (2019). Dynamic Input for Deep Reinforcement Learning in Autonomous Driving. arXiv preprint arXiv:1907.10994.
69. Huegle, M., Kalweit, G., Werling, M., & Boedecker, J. (2020). Dynamic interaction-aware scene understanding for reinforcement learning in autonomous driving. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4329-4335). IEEE.
70. Ioffe, S., and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.
71. Isele, D., Rahimi, R., Cosgun, A., Subramanian, K., and Fujimura, K. (2018). Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 2034-2039). IEEE.
72. Jiang, S., Chen, J., and Shen, M. (2019). An Interactive Lane Change Decision Making Model With Deep Reinforcement Learning. In 2019 7th International Conference on Control, Mechatronics and Automation (ICCMA) (pp. 370-376). IEEE.
73. Jin, J., Zhou, M., Zhang, W., Li, M., Guo, Z., Qin, Z., Jiao, Y., Tang, X., Wang, C., Wang, J. and Wu, G. (2019). Coride: joint order dispatching and fleet management for multi-scale ride-hailing platforms. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (pp. 1983-1992).
74. Kashihara, K. (2017). Deep Q learning for traffic simulation in autonomous driving at a highway junction. In 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 984-988). IEEE.
75. Ke, Z., Li, Z., Cao, Z., & Liu, P. (2020a). Enhancing Transferability of Deep Reinforcement Learning-Based Variable Speed Limit\endgraf Control Using Transfer Learning. IEEE Transactions on Intelligent Transportation Systems.
76. Ke, J., Xiao, F., Yang, H., & Ye, J. (2020b). Learning to delay in ride-sourcing systems: a multi-agent deep reinforcement learning framework. IEEE Transactions on Knowledge and Data Engineering.
77. Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. (2017). Learning combinatorial optimization algorithms over graphs. In Advances in Neural Information Processing Systems (pp. 6348-6358).

78. Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A.A., Yogamani, S. and Pérez, P., (2021). Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems. DOI: 10.1109/TITS.2021.3054625.
79. Konda, V. R., and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In Advances in neural information processing systems (pp. 1008-1014).
80. Kool, W., Hoof, H. V., and Welling, M. (2018). Attention solves your TSP, approximately. Statistics, 1050, 22.
81. Koutník, J., Cuccu, G., Schmidhuber, J., and Gomez, F. (2013). Evolving large-scale neural networks for vision-based reinforcement learning. In Proceedings of the 15th annual conference on Genetic and evolutionary computation (pp. 1061-1068).
82. Kuhnle, A., Schaarschmidt, M., and Fricke, K. (2017). Tensorforce: a tensorflow library for applied reinforcement learning. Web page.
83. Kulkarni, T. D., Narasimhan, K., Saeedi, A., and Tenenbaum, J. (2016). Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In Advances in neural information processing systems (pp. 3675-3683).
84. Kullman, N. D., Mendoza, J. E., Cousineau, M., and Goodson, J. C. (2019). Atari-fying the Vehicle Routing Problem with Stochastic Service Requests. arXiv preprint arXiv:1911.05922.
85. Kullman, N., Cousineau, M., Goodson, J., and Mendoza, J. (2020). Dynamic Ridehailing with Electric Vehicles. INFORMS.
86. Laurent, G. J., Matignon, L., and Fort-Piat, L. (2011). The world of independent learners is not Markovian. International Journal of Knowledge-based and Intelligent Engineering Systems, 15(1), 55-64.
87. Layek, A., Vien, N. A., and Chung, T. (2017). Deep reinforcement learning algorithms for steering an underactuated ship. In 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI) (pp. 602-607). IEEE
88. LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.
89. Lee, J., Balakrishnan, A., Gaurav, A., Czarnecki, K., and Sedwards, S. (2019). Wisemove: A framework for safe deep reinforcement learning for autonomous driving. arXiv preprint arXiv:1902.04118.
90. Li, C., and Czarnecki, K. (2019). Urban driving with multi-objective deep reinforcement learning. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (pp. 359-367). International Foundation for Autonomous Agents and Multiagent Systems.
91. Li, L., Lv, Y., and Wang, F. Y. (2016). Traffic signal timing via deep reinforcement learning. IEEE/CAA Journal of Automatica Sinica, 3(3), 247-254.
92. Li, Y. (2018). Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274.
93. Li, Y., He, H., Khajepour, A., Wang, H., and Peng, J. (2019b). Energy management for a power-split hybrid electric bus via deep reinforcement learning with terrain information. Applied Energy, 255, 113762.
94. Li, Y., He, H., Peng, J., and Wang, H. (2019a). Deep Reinforcement Learning-Based Energy Management for a Series Hybrid Electric Vehicle Enabled by History Cumulative Trip Information. IEEE Transactions on Vehicular Technology, 68(8), 7416-7430.
95. Li, Y., He, H., Peng, J., and Wu, J. (2018). Energy Management Strategy for a Series Hybrid Electric Vehicle Using Improved Deep Q-network Learning Algorithm with Prioritized Replay. DEStech Transactions on Environment, Energy and Earth Sciences, (iceee).
96. Lian, R., Peng, J., Wu, Y., Tan, H., and Zhang, H. (2020). Rule-interposing deep reinforcement learning based energy management strategy for power-split hybrid electric vehicle. Energy, 117297.
97. Liang, X., Du, X., Wang, G., and Han, Z. (2019). A deep reinforcement learning network for traffic light cycle control. IEEE Transactions on Vehicular Technology, 68(2), 1243-1253.
98. Liang, E., Wen, K., Lam, W. H., Sumalee, A., & Zhong, R. (2021). An Integrated Reinforcement Learning and Centralized Programming Approach for Online Taxi Dispatching. IEEE Transactions on Neural Networks and Learning Systems.
99. Liessner, R., Dietermann, A. M., and Bäker, B. (2018b). Safe deep reinforcement learning hybrid electric vehicle energy management. In International Conference on Agents and Artificial Intelligence (pp. 161-181). Springer, Cham.
100. Liessner, R., Schmitt, J., Dietermann, A., and Bäker, B. (2019). Hyperparameter Optimization for Deep Reinforcement Learning in Vehicle Energy Management. In ICAART 2019.
101. Liessner, R., Schroer, C., Dietermann, A. M., and Bäker, B. (2018a). Deep Reinforcement Learning for Advanced Energy Management of Hybrid Electric Vehicles. In ICAART (2) (pp. 61-72).
102. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.
103. Lin, K., Zhao, R., Xu, Z., & Zhou, J. (2018b). Efficient large-scale fleet management via multi-agent deep reinforcement learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1774-1783).
104. Lin, Y., Dai, X., Li, L., & Wang, F. Y. (2018a). An efficient deep reinforcement learning model for urban traffic control. arXiv preprint arXiv:1808.01876.

105. Lin, Y., McPhee, J., and Azad, N. L. (2019). Longitudinal dynamic versus kinematic models for car-following control using deep reinforcement learning. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC) (pp. 1504-1510). IEEE.
106. Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In Proceedings of the 11th international conference on machine learning (pp. 157–163). New Brunswick, NJ, USA
107. Littman, M. L. (2001). Value-function reinforcement learning in Markov games. Cognitive Systems Research, 2(1), 55–66
108. Liu, M., Deng, J., Xu, M., Zhang, X., & Wang, W. (2017). Cooperative deep reinforcement learning for tra ic signal control. In The 7th International Workshop on Urban Computing (UrbComp 2018).
109. Liu, X. Y., Ding, Z., Borst, S., & Walid, A. (2018). Deep reinforcement learning for intelligent transportation systems. arXiv preprint arXiv:1812.00979.
110. Liu, Z., Li, J., & Wu, K. (2020). Context-Aware Taxi Dispatching at City-Scale Using Deep Reinforcement Learning. IEEE Transactions on Intelligent Transportation Systems.
111. Makantasis, K., Kontorinaki, M., and Nikolos, I. (2019). A deep reinforcement learning driving policy for autonomous road vehicles. arXiv preprint arXiv:1905.09046.Martinsen, A. B., and Lekkas, A. M. (2018). Curved path following with deep reinforcement learning: Results from three vessel models. In OCEANS 2018 MTS/IEEE Charleston (pp. 1-8). IEEE.
112. Mao, C., Liu, Y., & Shen, Z. J. M. (2020). Dispatch of autonomous vehicles for taxi services: A deep reinforcement learning approach. Transportation Research Part C: Emerging Technologies, 115, 102626.
113. Martinsen, A. B., & Lekkas, A. M. (2018, October). Curved path following with deep reinforcement learning: Results from three vessel models. In OCEANS 2018 MTS/IEEE Charleston (pp. 1-8). IEEE.
114. Min, K., and Kim, H. (2018). Deep q learning based high level driving policy determination. In 2018 IEEE Intelligent Vehicles Symposium (IV) (pp. 226-231). IEEE.
115. Mirchevska, B., Pek, C., Werling, M., Althoff, M., and Boedecker, J. (2018). High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 2156-2162). IEEE.
116. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In International conference on machine learning (pp. 1928-1937).
117. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.
118. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.
119. Mousavi, S. S., Schukat, M., and Howley, E. (2017). Traffic light control using deep policy-gradient and value-function-based reinforcement learning. IET Intelligent Transport Systems, 11(7), 417-423.
120. Mukadam, M., Cosgun, A., Nakhaei, A., and Fujimura, K. (2017). Tactical decision making for lane changing with deep reinforcement learning. Proceedings of the Conference on Neural Information Processing Systems (NIPS).
121. Muresan, M., Fu, L., and Pan, G. (2019). Adaptive traffic signal control with deep reinforcement learning an exploratory investigation. arXiv preprint arXiv:1901.00960.
122. Nageshrao, S., Tseng, H. E., and Filev, D. (2019). Autonomous highway driving using deep reinforcement learning. In 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC) (pp. 2326-2331). IEEE.
123. Nassef, O., Sequeira, L., Salam, E., & Mahmoodi, T. (2020, October). Deep Reinforcement Learning in Lane Merge Coordination for Connected Vehicles. In 2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications (pp. 1-7). IEEE.
124. Nazari, M., Oroojlooy, A., Snyder, L., and Takác, M. (2018). Reinforcement learning for solving the vehicle routing problem. In Advances in Neural Information Processing Systems (pp. 9839-9849).
125. Nezafat, R. V. (2019). Deep Reinforcement Learning Approach for Lagrangian Control: Improving Freeway Bottleneck Throughput Via Variable Speed Limit.
126. Ng, A. Y., and Russell, S. J. (2000). Algorithms for inverse reinforcement learning. In Icml (Vol. 1, p. 2).
127. Ning, L., Li, Y., Zhou, M., Song, H., and Dong, H. (2019). A Deep Reinforcement Learning Approach to High-speed Train Timetable Rescheduling under Disturbances. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC) (pp. 3469-3474). IEEE.
128. Nishi, T., Doshi, P., and Prokhorov, D. (2019). Merging in congested freeway traffic using multipolicy decision making and passive actor-critic learning. IEEE Transactions on Intelligent Vehicles, 4(2), 287-297.
129. Nishitani, I., Yang, H., Guo, R., Keshavamurthy, S., & Oguchi, K. (2020). Deep Merging: Vehicle Merging Controller Based on Deep Reinforcement Learning with Embedding Network. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 216-221). IEEE.
130. Noruzoliaee, M., and Zou, B. (2021). One-to-many matching and section-based formulation of autonomous ridesharing equilibrium. Under review.
131. Noruzoliaee, M., Zou, B., and Liu, Y. (2018). Roads in transition: integrated modeling of a manufacturer-traveler-road infrastructure system in a mixed autonomous/human driving environment. Transportation Research Part C: Emerging Technologies, 90, 307-333.

132. Nosrati, M.S., Abolfathi, E.A., Elmahgiubi, M., Yadmellat, P., Luo, J., Zhang, Y., Yao, H., Zhang, H. and Jamil, A. (2018). Towards practical hierarchical reinforcement learning for multi-lane autonomous driving. 2018 NIPS MLITS Workshop, 2018.
133. Nowé, A., Vrancx, P., and De Hauwere, Y. M. (2012). Game theory and multi-agent reinforcement learning. In Reinforcement Learning (pp. 441-470). Springer, Berlin, Heidelberg.
134. Obara, M., Kashiyama, T., and Sekimoto, Y. (2018). Deep Reinforcement Learning Approach for Train Rescheduling Utilizing Graph Theory. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 4525-4533). IEEE
135. Oda, T., and Joe-Wong, C. (2018). MOVI: A model-free approach to dynamic fleet management. In IEEE INFOCOM 2018-IEEE Conference on Computer Communications (pp. 2708-2716). IEEE.
136. Oda, T., and Tachibana, Y. (2018). Distributed fleet control with maximum entropy deep reinforcement learning. 2018 NIPS MLITS Workshop, 2018.
137. OpenAI. 2017. OpenAI Baselines: ACKTR & A2C. https://openai.com/blog/baselines-acktr-a2c/
138. Paden, B., Čáp, M., Yong, S. Z., Yershov, D., & Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. IEEE Transactions on intelligent vehicles, 1(1), 33-55.
139. Pandey, V., Wang, E., & Boyles, S. D. (2020). Deep reinforcement learning algorithm for dynamic pricing of express lanes with multiple access locations. Transportation Research Part C: Emerging Technologies, 119, 102715.
140. Papageorgiou, M., Hadj-Salem, H., and Middelham, F. (1997). ALINEA local ramp metering: Summary of field results. Transportation research record, 1603(1), 90-98.
141. Paxton, C., Raman, V., Hager, G. D., and Kobilarov, M. (2017). Combining neural networks and tree search for task and motion planning in challenging environments. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 6059-6066). IEEE.
142. Peer, E., Menkovski, V., Zhang, Y., and Lee, W. J. (2018). Shunting trains with deep reinforcement learning. In 2018 ieee international conference on systems, man, and cybernetics (smc) (pp. 3063-3068). IEEE.
143. Peng, B., Wang, J., and Zhang, Z. (2019). A Deep Reinforcement Learning Algorithm Using Dynamic Attention Model for Vehicle Routing Problems. In International Symposium on Intelligence Computation and Applications (pp. 636-650). Springer, Singapore.
144. Plappert, M. (2016). Keras-rl. GitHub Repository.
145. Qi, X., Luo, Y., Wu, G., Boriboonsomsin, K., and Barth, M. (2019). Deep reinforcement learning enabled self-learning control for energy efficient driving. Transportation Research Part C: Emerging Technologies, 99, 67-81.
146. Qi, X., Luo, Y., Wu, G., Boriboonsomsin, K., and Barth, M. J. (2017). Deep reinforcement learning-based vehicle energy efficiency autonomous learning system. In 2017 IEEE Intelligent Vehicles Symposium (IV) (pp. 1228-1233). IEEE.
147. Qin, Z., Tang, X., Jiao, Y., Zhang, F., Xu, Z., Zhu, H., & Ye, J. (2020). Ride-hailing order dispatching at DiDi via reinforcement learning. INFORMS Journal on Applied Analytics, 50(5), 272-286.
148. Qu, X., Yu, Y., Zhou, M., Lin, C. T., and Wang, X. (2020). Jointly dampening traffic oscillations and improving energy consumption with electric, connected and automated vehicles: A reinforcement learning based approach. Applied Energy, 257, 114030.
149. Rejaili, R. P. A., and Figueiredo, J. M. P. (2018). Deep reinforcement learning algorithms for ship navigation in restricted waters. Mecatrone, 3(1).
150. Sabri, M. F. M., Danapalasingam, K. A., and Rahmat, M. F. (2016). A review on hybrid electric vehicles architecture and energy management strategies. Renewable and Sustainable Energy Reviews, 53, 1433-1442.
151. Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2016). End-to-end deep reinforcement learning for lane keeping assist. arXiv preprint arXiv:1612.04340.
152. Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. Electronic Imaging, 2017(19), 70-76.
153. Sawada, R. (2019). Automatic Collision Avoidance Using Deep Reinforcement Learning with Grid Sensor. In Symposium on Intelligent and Evolutionary Systems (pp. 17-32). Springer, Cham.
154. Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. arXiv preprint arXiv:1511.05952.
155. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
156. Shabestary, S. M. A., & Abdulhai, B. (2018). Deep learning vs. discrete reinforcement learning for adaptive traffic signal control. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 286-293). IEEE.
157. Shalev-Shwartz, S., Shammah, S., and Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295.
158. Sharifzadeh, S., Chiotellis, I., Triebel, R., and Cremers, D. (2016). Learning to drive using inverse reinforcement learning and deep q-networks. arXiv preprint arXiv:1612.03653.
159. Shen, H., Hashimoto, H., Matsuda, A., Taniguchi, Y., Terada, D., and Guo, C. (2019). Automatic collision avoidance of multiple ships based on deep Q-learning. Applied Ocean Research, 86, 268-288.

160. Shen, Y., Zhao, N., Xia, M., and Du, X. (2017). A deep q-learning network for ship stowage planning problem. Polish Maritime Research, 24(s3), 102-109.
161. Shi, D., Ding, J., Errapotu, S. M., Yue, H., Xu, W., Zhou, X., and Pan, M. (2019b). Deep Q -Network-Based Route Scheduling for TNC Vehicles With Passengers' Location Differential Privacy. IEEE Internet of Things Journal, 6(5), 7681-7692.
162. Shi, S., & Chen, F. (2018). Deep Recurrent Q-learning Method for Area Traffic Coordination Control. Journal of Advances in Mathematics and Computer Science, 1-11.
163. Shi, T., Wang, P., Cheng, X., Chan, C. Y., and Huang, D. (2019a). Driving Decision and Control for Automated Lane Change Behavior based on Deep Reinforcement Learning. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC) (pp. 2895-2900). IEEE.
164. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S. (2016). Mastering the game of Go with deep neural networks and tree search. nature, 529(7587), 484.
165. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms.
166. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A. and Chen, Y. (2017). Mastering the game of go without human knowledge. Nature, 550(7676), 354-359.
167. Singh, A., Al-Abbasi, A., and Aggarwal, V. (2019). A reinforcement learning based algorithm for multi-hop ride-sharing: Model-free approach. In Neural Information Processing Systems (Neurips) Workshop.
168. Stadie, B. C., Abbeel, P., and Sutskever, I. (2017). Third-person imitation learning. arXiv preprint arXiv:1703.01703.
169. Stooke, A., and Abbeel, P. (2019). rlpyt: A research code base for deep reinforcement learning in pytorch. arXiv preprint arXiv:1909.01500.
170. Sutton, R. S., Precup, D., and Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial intelligence, 112(1-2), 181-211.
171. Talpaert, V., Sobh, I., Kiran, B. R., Mannion, P., Yogamani, S., El-Sallab, A., and Perez, P. (2019). Exploring applications of deep reinforcement learning for real-world autonomous driving systems. arXiv preprint arXiv:1901.01536.
172. Tan, H., Zhang, H., Peng, J., Jiang, Z., and Wu, Y. (2019). Energy management of hybrid electric bus based on deep reinforcement learning in continuous state and action space. Energy Conversion and Management, 195, 548-560.
173. Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the tenth international conference on machine learning (pp. 330-337).
174. Tan, T., Bao, F., Deng, Y., Jin, A., Dai, Q., and Wang, J. (2019). Cooperative deep reinforcement learning for large-scale traffic grid signal control. IEEE transactions on cybernetics.
175. Tang, X., Qin, Z., Zhang, F., Wang, Z., Xu, Z., Ma, Y., Zhu, H. and Ye, J., (2019). A deep value-network based approach for multi-driver order dispatching. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 1780-1790).
176. Van der Pol, E., and 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).
177. Van Hasselt, H. (2010). Double Q-learning. In Advances in neural information processing systems (pp. 2613-2621).
178. Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In Thirtieth AAAI conference on artificial intelligence.
179. Vezhnevets, A., Mnih, V., Osindero, S., Graves, A., Vinyals, O., and Agapiou, J. (2016). Strategic attentive writer for learning macro-actions. In Advances in neural information processing systems (pp. 3486-3494).
180. Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. (2017). Feudal networks for hierarchical reinforcement learning. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 3540-3549). JMLR. org.
181. Vinitsky, E., Kreidieh, A., Le Flem, L., Kheterpal, N., Jang, K., Wu, C., Wu, F., Liaw, R., Liang, E. and Bayen, A.M. (2018b). Benchmarks for reinforcement learning in mixed-autonomy traffic. In Conference on Robot Learning (pp. 399-409).
182. Vinitsky, E., Parvate, K., Kreidieh, A., Wu, C., and Bayen, A. (2018a). Lagrangian control through deep-rl: Applications to bottleneck decongestion. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 759-765). IEEE.
183. Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In Advances in neural information processing systems (pp. 2692-2700).
184. Wan, C. H., and Hwang, M. C. (2018). Value-based deep reinforcement learning for adaptive isolated intersection signal control. IET Intelligent Transport Systems, 12(9), 1005-1010.
185. Wang, C., Zhang, X., Cong, L., Li, J., and Zhang, J. (2019g). Research on intelligent collision avoidance decision-making of unmanned ship in unknown environments. Evolving Systems, 10(4), 649-658.
186. Wang, G., Hu, J., Li, Z., and Li, L. (2019a). Cooperative Lane Changing via Deep Reinforcement Learning. arXiv preprint arXiv:1906.08662.
187. Wang, J., & Sun, L. (2020). Dynamic holding control to avoid bus bunching: A multi-agent deep reinforcement learning framework. Transportation Research Part C: Emerging Technologies, 116, 102661.

188. Wang, J., Zhang, Q., Zhao, D., and Chen, Y. (2019c). Lane Change Decision-making through Deep Reinforcement Learning with Rule-based Constraints. In 2019 International Joint Conference on Neural Networks (IJCNN) (pp. 1-6). IEEE.
189. Wang, P., and Chan, C. Y. (2017). Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (pp. 1-6). IEEE.
190. Wang, P., and Chan, C. Y. (2018). Autonomous ramp merge maneuver based on reinforcement learning with continuous action space. arXiv preprint arXiv:1803.09203.
191. Wang, P., Chan, C. Y., and de La Fortelle, A. (2018a). A reinforcement learning based approach for automated lane change maneuvers. In 2018 IEEE Intelligent Vehicles Symposium (IV) (pp. 1379-1384). IEEE.
192. Wang, P., Chan, C. Y., and Li, H. (2019b). Automated Driving Maneuvers under Interactive Environment based on Deep Reinforcement Learning. Accepted at the 98th Annual Meeting of the Transportation Research Board (TRB).
193. Wang, P., Li, Y., Shekhar, S., and Northrop, W. F. (2019d). Actor-Critic based Deep Reinforcement Learning Framework for Energy Management of Extended Range Electric Delivery Vehicles. In 2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) (pp. 1379-1384). IEEE.
194. Wang, P., Li, Y., Shekhar, S., and Northrop, W. F. (2019e). A deep reinforcement learning framework for energy management of extended range electric delivery vehicles. In 2019 IEEE Intelligent Vehicles Symposium (IV) (pp. 1837-1842). IEEE.
195. Wang, P., Liu, D., Chen, J., Li, H., and Chan, C. Y. (2020). Human-like Decision Making for Autonomous Driving via Adversarial Inverse Reinforcement Learning. arXiv, arXiv-1911.
196. Wang, R., Zhou, M., Li, Y., Zhang, Q., and Dong, H. (2019f). A Timetable Rescheduling Approach for Railway based on Monte Carlo Tree Search. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC) (pp. 3738-3743). IEEE.
197. Wang, S., Jia, D., and Weng, X. (2018b). Deep reinforcement learning for autonomous driving. arXiv preprint arXiv:1811.11329.
198. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581.
199. Wang, Z., Qin, Z., Tang, X., Ye, J., & Zhu, H. (2018c). Deep reinforcement learning with knowledge transfer for online rides order dispatching. In 2018 IEEE International Conference on Data Mining (ICDM) (pp. 617-626). IEEE.
200. Wen, J., Zhao, J., and Jaillet, P. (2017). Rebalancing shared mobility-on-demand systems: A reinforcement learning approach. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (pp. 220-225). IEEE.
201. Weng, L. (2020). A (Long) Peek into Reinforcement Learning. https://lilianweng.github.io/lil-log/2018/02/19/a-long-peek-into-reinforcement-learning.html
202. Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8(3-4), 229-256.
203. Williams, R. J., and Peng, J. (1991). Function optimization using connectionist reinforcement learning algorithms. Connection Science, 3(3), 241-268.
204. Wolf, P., Kurzer, K., Wingert, T., Kuhnt, F., and Zollner, J. M. (2018). Adaptive behavior generation for autonomous driving using deep reinforcement learning with compact semantic states. In 2018 IEEE Intelligent Vehicles Symposium (IV) (pp. 993-1000). IEEE.
205. Woo, J., Yu, C., and Kim, N. (2019). Deep reinforcement learning-based controller for path following of an unmanned surface vehicle. Ocean Engineering, 183, 155-166.
206. Wu, C., Parvate, K., Kheterpal, N., Dickstein, L., Mehta, A., Vinitsky, E., and Bayen, A. M. (2017). Framework for control and deep reinforcement learning in traffic. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (pp. 1-8). IEEE.
207. Wu, J., He, H., Peng, J., Li, Y., & Li, Z. (2018). Continuous reinforcement learning of energy management with deep Q network for a power split hybrid electric bus. Applied energy, 222, 799-811.
208. Wu, J., He, H., Peng, J., Li, Y., and Li, Z. (2018). Continuous reinforcement learning of energy management with deep Q network for a power split hybrid electric bus. Applied energy, 222, 799-811.
209. Wu, Y., Tan, H., Peng, J., and Ran, B. (2019a). A Deep Reinforcement Learning Based Car Following Model for Electric Vehicle. 智能城市应用, 2(5).
210. Wu, Y., Tan, H., Peng, J., Zhang, H., and He, H. (2019b). Deep reinforcement learning of energy management with continuous control strategy and traffic information for a series-parallel plug-in hybrid electric bus. Applied energy, 247, 454-466.
211. Wu, Y., Tan, H., Qin, L., & Ran, B. (2020). Differential variable speed limits control for freeway recurrent bottlenecks via deep actor-critic algorithm. Transportation research part C: emerging technologies, 117, 102649.
212. Wulfmeier, M., Ondruska, P., and Posner, I. (2015). Maximum entropy deep inverse reinforcement learning. arXiv preprint arXiv:1507.04888.
213. Xu, Z., Tang, C., and Tomizuka, M. (2018). Zero-shot deep reinforcement learning driving policy transfer for autonomous vehicles based on robust control. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (pp. 2865-2871). IEEE.
214. Yang, G., Zhang, F., Gong, C., and Zhang, S. (2019). Application of a Deep Deterministic Policy Gradient Algorithm for Energy-Aimed Timetable Rescheduling Problem. Energies, 12(18), 3461.

215. Ye, F., Cheng, X., Wang, P., and Chan, C. Y. (2020). Automated Lane Change Strategy using Proximal Policy Optimization-based Deep Reinforcement Learning. arXiv preprint arXiv:2002.02667.
216. Ye, Y., Zhang, X., and Sun, J. (2019). Automated vehicle's behavior decision making using deep reinforcement learning and high-fidelity simulation environment. Transportation Research Part C: Emerging Technologies, 107, 155-170.
217. Yi, H. (2018). Deep deterministic policy gradient for autonomous vehicle driving. In Proceedings on the International Conference on Artificial Intelligence (ICAI) (pp. 191-194). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).
218. Ying, C. S., Chow, A. H., & Chin, K. S. (2020). An actor-critic deep reinforcement learning approach for metro train scheduling with rolling stock circulation under stochastic demand. Transportation Research Part B: Methodological, 140, 210-235.
219. You, C., Lu, J., Filev, D., and Tsiotras, P. (2019). Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. Robotics and Autonomous Systems, 114, 1-18.
220. Yu, J. J. Q., Yu, W., and Gu, J. (2019). Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems, 20(10), 3806-3817.
221. Yu, L., Shao, X., Wei, Y., and Zhou, K. (2018). Intelligent land-vehicle model transfer trajectory planning method based on deep reinforcement learning. Sensors, 18(9), 2905.
222. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. In Advances in neural information processing systems (pp. 3391-3401).
223. Zhang, K., Li, M., Zhang, Z., Lin, X., and He, F. (2020). Multi-Vehicle Routing Problems with Soft Time Windows: A Multi-Agent Reinforcement Learning Approach. arXiv preprint arXiv:2002.05513.
224. Zhang, Q. S., and Zhu, S. C. (2018). Visual interpretability for deep learning: a survey. Frontiers of Information Technology and Electronic Engineering, 19(1), 27-39.
225. Zhang, S., Peng, H., Nageshrao, S., and Tseng, E. (2019). Discretionary Lane Change Decision Making using Reinforcement Learning with Model-Based Exploration. In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA) (pp. 844-850). IEEE.
226. Zhang, X., Wang, C., Liu, Y., and Chen, X. (2019). Decision-making for the autonomous navigation of maritime autonomous surface ships based on scene division and deep reinforcement learning. Sensors, 19(18), 4055.
227. Zhang, Z., Yang, J., and Zha, H. (2019). Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization. arXiv preprint arXiv:1909.10651.
228. Zhao, J., Mao, M., Zhao, X., & Zou, J. (2020). A hybrid of deep reinforcement learning and local search for the vehicle routing problems. IEEE Transactions on Intelligent Transportation Systems.
229. Zhao, L., and Roh, M. I. (2019). COLREGs-compliant multiship collision avoidance based on deep reinforcement learning. Ocean Engineering, 191, 106436.
230. Zhao, L., Roh, M. I., and Lee, S. J. (2019). Control method for path following and collision avoidance of autonomous ship based on deep reinforcement learning. Journal of Marine Science and Technology, 27(4), 293-310.
231. Zhao, P., Wang, Y., Chang, N., Zhu, Q., and Lin, X. (2018). A deep reinforcement learning framework for optimizing fuel economy of hybrid electric vehicles. In 2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC) (pp. 196-202). IEEE.
232. Zhou, M., Yu, Y., and Qu, X. (2019a). Development of an efficient driving strategy for connected and automated vehicles at signalized intersections: A reinforcement learning approach. IEEE Transactions on Intelligent Transportation Systems.
233. Zhou, M., Jin, J., Zhang, W., Qin, Z., Jiao, Y., Wang, C., Wu, G., Yu, Y. and Ye, J. (2019b). Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (pp. 2645-2653).
234. Zhou, R., and Song, S. (2018). Optimal automatic train operation via deep reinforcement learning. In 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI) (pp. 103-108). IEEE.
235. Zhou, R., Song, S., Xue, A., You, K., and Wu, H. (2020). Smart Train Operation Algorithms based on Expert Knowledge and Reinforcement Learning. arXiv preprint arXiv:2003.03327.
236. Zhu, L., He, Y., Yu, F. R., Ning, B., Tang, T., and Zhao, N. (2017). Communication-based train control system performance optimization using deep reinforcement learning. IEEE Transactions on Vehicular Technology, 66(12), 10705-10717.
237. Zhu, M., Wang, X., and Wang, Y. (2018). Human-like autonomous car-following model with deep reinforcement learning. Transportation research part C: emerging technologies, 97, 348-368.
238. Zhu, M., Wang, Y., Pu, Z., Hu, J., Wang, X., and Ke, R. (2019). Safe, Efficient, and Comfortable Velocity Control based on Reinforcement Learning for Autonomous Driving. arXiv preprint arXiv:1902.00089.
239. Zou, B., Choobchian, P., and Rozenberg, J. (2021). Cyber resilience of autonomous mobility systems: Cyber-attacks and resilience-enhancing strategies. Journal of Transportation Security, in press. DOI: https://doi.org/10.1007/s12198-021-00230-w.