Journal of Big Data Analytics in Transportation manuscript No.
(will be inserted by the editor)

Robust Deep Reinforcement Learning for Traffic Signal Control

Kai Liang Tan · Anuj Sharma · Soumik Sarkar

Received: date / Accepted: date

Abstract A traffic signal is a fundamental part of the traffic control system to reduce congestion and enhance safety. Since the inception of motorized vehicles, traffic signal controllers are put in place to coordinate and maintain traffic flow. With the number of vehicles on the road increasing exponentially, it is imperative to innovate new traffic control frameworks to cope with the high-density traffic demand. In this regard, recent advances in machine/deep learning have enabled significant progress towards reducing congestion using reinforcement learning for traffic signal control. However, most of these works are still not ready for deployment due to assumptions of perfect knowledge of the traffic environment. In reality, congestion detection or prediction systems are at best able to approximate the traffic state with significant noise. In this work, we propose a robust training framework for reinforcement learning agents that can handle such noisy approximation of the traffic states. Specifically, we show that by carefully adding synthetic perturbations to the state space such as the queue length during training, the reinforcement learning agents can be robustified. Conceptually, our approach is similar to adversar-

Kai Liang Tan
Department of Mechanical Engineering
Iowa State University
Ames, Iowa 50014
E-mail: kailiang@iastate.edu

Anuj Sharma
Department or Civil Engineering
Iowa State University
Ames, Iowa 50014
E-mail: anujs@iastate.edu

Soumik Sarkar
Department of Mechanical Engineering
Iowa State University
Ames, Iowa 50014
E-mail: soumiks@iastate.edu

ial training schemes and can lead to successful deployment of reinforcement learning agent-based traffic signal controllers.

Keywords Deep Learning · Deep Reinforcement Learning · Robust · Adaptive Signal Control Technologies

1 Introduction

Transportation is an integral part of our lives to commute from one place to another. As vehicles became more affordable to the public, the number of congestion also increased linearly with the number of cars sold. A study by Texas A&M Transportation Institute on the effects of congestion since 1982 showed overwhelmingly negative effects on our society [16]. For 35 years, the total traffic delay increased by 8.8 billion hours, fuel waste increased by 3.35 billion gallons, which equates to a total congestion cost of 179 billion dollars. Hence, it is imperative to introduce newer Adaptive Signal Control Technologies (ASCT) methods to reduce the number of congestion drastically.

Traditionally, the traffic control problem is formulated as an optimization problem, where certain unrealistic assumptions are necessary to make the problem tractable [34]. With these assumptions, a traffic engineer crafts a timing plan based on historical traffic volumes for a specific intersection [27, 40]. This procedure is costly and time-consuming, yet it does not provide the flexibility to adapt to changing traffic demands if there was a special event. To provide more flexibility, ASCT can reallocate green time from a minor road to the major road when there is no demand on the side streets [12]. However, ASCT is still based on some pre-defined rules which will not cover some special cases in daily traffic such as ending a green phase right before a large platoon of vehicles are arriving [37].

To overcome the difficulty of pre-defining rules to cover all possible traffic scenarios, recently researchers in the machine learning community tackled the traffic control problem as a deep reinforcement learning (RL) problem [19, 31, 37]. By formulating the problem as a deep RL problem, the RL agent can interact with a simulated traffic environment for many episodes until it converges to an optimal policy, such that it covers all possible traffic scenarios. Generally, researchers utilize information from the simulation environment as either state information or a reward feedback signal for the deep RL agent. In reality, this information is obtained from sensors or communication technologies such as induction loop detectors, traffic cameras, RFID sensors, or Bluetooth information from vehicles [20]. These traffic sensors are known to have reliability issues with sensor performance and possible failures due to several different factors, such as wear and tear from lack of maintenance for traffic sensors, unreliable number of Bluetooth drivers, varying weather conditions, occlusions, or under-performing computer vision detection algorithm for traffic cameras, etc [4, 23, 32, 44]. With all the uncertainty from issues mentioned above, it is imperative that deep RL agent learns a more robust policy in the event these sensors provide information that is slightly different from the actual traffic states [9, 11, 29, 30, 33].

In this paper, we utilize various sensor information to define a set of state representation for the deep RL agent. Utilizing a variety of state representations broadens the selection of states used to represent the current state of the environment as close to the real world as possible. To train a more robust agent, we employ a similar approach as randomized smoothing [6] by incorporating noise into the state dimension. More specifically, the noise is introduced into the queue length dimension only to mimic real-world queue length approximation methods. With perturbations introduced, we show that the deep RL agent learns a more robust policy towards several degrees of noise. This broadens the selection of states used to represent the current environment state, mimicking real-world situations.

2 Related Works

2.1 Traditional Traffic Control

Traffic signals operated in the United States are separated into three types of control methods: 1) Pre-timed 2) Semi-actuated 3) Fully-actuated [12]. Pre-timed control uses a fixed timing schedule employed by experienced traffic engineers based on monitoring historical traffic patterns and data for any given intersection. Pre-timed control is generally used when the distance between two or more traffic intersections is relatively small and the traffic pattern throughout the day is consistent (i.e., Downtown Manhattan or Chicago). This traffic controller is the cheapest among the three types of control methods because it does not require any traffic sensor to function. A semi-actuated controller is generally used in an intersection with a major and minor road. The controller would always allocate the green phase to the major road but switches to the minor road to clear small volume of vehicles. This control method needs an induction loop sensor near the stop bar on the minor road to detect incoming vehicles required for actuation [13]. A fully-actuated controller is used when both conflicting directions are major roads, where induction loop sensors are required in all directions to detect traffic flow conditions.

2.2 Deep Reinforcement Learning in Traffic Control

Deep RL as an ASCT approach for minimizing congestion has been shown to perform better than conventional methods of traffic controllers [19, 31, 37]. Typically, the deep RL agent is trained inside a traffic simulator where the simulator is capable of simulating a good approximate model of complex traffic conditions. Examples of such simulators include SUMO [14], VISSIM [8] and AIMSUM [2]. In the simulation, the deep RL agent takes a state and reward input that describes the current traffic state and decides on the best

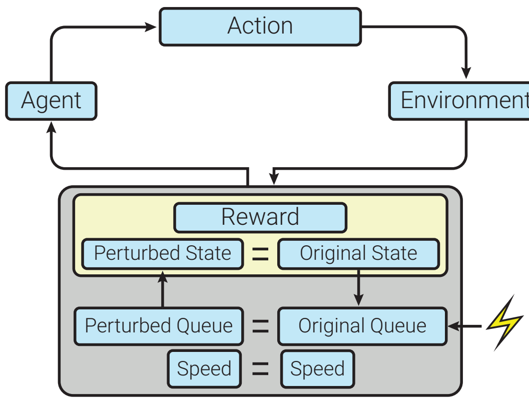

Fig. 1 Deep Reinforcement Learning Loop: This figure illustrates the interaction loop between agent and environment. The agent interacts with the environment with an action, where the environment returns a state and reward. In this paper, only the queue length dimension can be perturbed with synthetic noise, while other dimensions are unaffected. A perfect queue length is hard to obtain with approximation methods. Hence this perturbation introduces a better representation of uncertainty instead of complete knowledge in the simulation.

action. After numerous training iterations under various traffic conditions, the agent would learn a generalized policy that minimizes congestion. Research in the deep learning community for traffic control generally revolves around five major aspects:

1. Novel architectures: Researchers in this section strive to develop novel Deep RL architectures that specifically benefit the traffic control problem. Wei *et al.* formulated their network with a phase gate to explicitly activate different layers depending on the current traffic phase [42]. Lin *et al.* represented the state space as a 2-D grid containing sensor information for a grid network, which acts as input into a Residual Network [10] to learn the relationship between states and actions [22].
2. State space design: Over the past few years, researchers have tried various state-space representations for the traffic control problem. Most approaches revolve around capturing and representing the current state of traffic so the deep RL agent can decide on the best action to take. Notably, researchers tried discretizing roads [21, 31, 43], segmenting lanes into bins [41], top-down view of the traffic intersection [42] and utilizing traffic sensors [19, 37, 42].
3. Action definition: Several researchers approached the same optimization problem with different action definition. To highlight several common approaches: Van *et. al.* directly switches between two green phases without safety measures (no yellow, red, and minimum green phases). Liang *et. al.* maintains a set of green time for four phases. At each simulation step, the agent can add or subtract 5 seconds from the green time set. Li *et. al.* is similar to Van *et. al.*, but will not switch phases if a minimum green time has not been reached to avoid flickering phase switches. Wei *et. al.*

includes a 3-second yellow before turning red if the agent decides to switch green phases. Tan *et. al.* includes yellow, red clearance, and minimum green time if the agent decides to switch green phases. Timings of each yellow, red clearance and minimum green was obtained from the Federal Highway Authority.

4. Reward crafting: Reward crafting is an essential part of Deep RL research in ASCT to ensure smooth and efficient convergence of the agent. The community experimented with various feedback signals such as average delay [1], average travel time [25], queue lengths [19] and weighted combination of multiple variables [31, 37, 42].
5. Application type: This section is split into two subsections: single intersection and corridor/grid level intersections. The former focuses solely on solving a single intersection optimization problem via innovating in the four major sections mentioned above [19, 37, 42]. The latter specifically focuses on either multi-agent algorithms [5], or some form of transfer knowledge from a single agent deployed at multiple intersections with communication [31, 41].

Current research gaps are practical applications and novel architectures. More specifically, there is a lack of traffic control utilizing deep learning methods that directly translates to field-testing. This paper aims to contribute to that research direction by introducing noise into state space design to model a more realistic noisy nature of sensors in the real world.

2.3 Noise-robust Deep Reinforcement Learning Agents

While adversarial machine learning has been thoroughly studied in machine learning [7, 15, 26], the robustification of deep RL agents against an adversarial attack or sensor anomaly have been relatively less studied. Lee *et al.* [17] proposed a gradient-based white-box attack in action space. This represents an adversary subtly attacking actuators with minimal energy without alerting the authorities. Tan *et. al.* [36] expanded on the work done by Lee *et. al.* by introducing the gradient-based white box adversary into the training scheme to train a robust agent. Rodrigues *et al.* [33] developed a callback-based RL framework to test out different scenarios that the authors are using to gain informative insights towards building a more robust RL agent.

3 Methodology

3.1 Deep Reinforcement Learning

The recently developed variant of RL-leveraging deep neural networks representing the value or policy function is known as Deep RL. In general, RL can be categorized into model-based RL and model-free RL. The key difference between these methods is that model-based RL has to learn the environment

Table 1 Defined Sets of State Space

State Spaces	Description
SS:Basic	Contains traffic signal phases and queue lengths.
SS:Actuation	Phase and induction loop actuation.
SS:Pressure	Traffic phase and residual formulation as max pressure [39].
SS:Speed	Phase and induction loop averaged speeds.
SS:Speed Actuation	Phase and induction loop actuation and average speeds.

model (e.g., learning a transition function F that maps input t to output $t{+}1$.). In contrast, model-free RL has direct access to the environment for training. In this work, we focus exclusively on model-free RL algorithms since we have access to the environment model. There are two main algorithmic approaches for solving the RL problem: Policy Optimization and Value-based. The former directly optimizes the parameter θ of the agent, while the latter optimizes a Q-function (based on the Bellman equation [3]) that approximates the optimal action-value function.

Formally, let s_t, a_t, and r_t be defined as the state, action, and reward for time step t respectively. At time step t, the deep RL agent π, observes the state s_t and decide upon action a_t to interact with the environment model. Based on the environment model's transition function $E(s_t, a_t)$, it will return a transitioned state observation s_{t+1} and the reward for the transitioned state r_t. This interaction between the deep RL agent and the environment model is repeated until a termination condition is met (e.g., if the agent accomplished the environment task), or until a predefined total time step T.

By refining the deep RL agent's policy $\pi(a|s)$ over many simulation iterations, the goal is to obtain an optimal policy $\pi^*(a|s)$ such that every action the agent takes maximizes the cumulative future rewards G_t.

$$G_t = \sum_{t=0}^{T} R = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{T-1} r_t \tag{1}$$

The constant γ ranges from $[0, 1]$ depending on the configuration of the agent. Values closer towards 0 make the agent favor short term rewards, whereas values closer towards 1 make the agent maximize possible higher future rewards.

3.2 State Space Formulation

Various sensor observations can be obtained to serve as the state information for the deep RL agents. For this study, we utilized information from the induction loop sensor and traffic signals. Induction loop sensors provide vehicle speed and occupancy information, while traffic signals provide phase information for respective lanes. For this experiment, we are interested in studying how well the deep RL agent performs under imperfect state information in the queue length dimension, assuming there are no outliers in both speed and

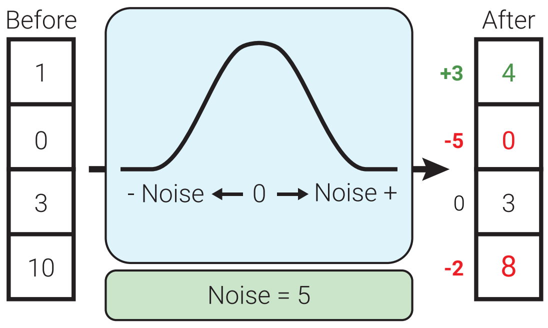

Fig. 2 Synthetic Noise Addition in the queue length dimension: This figure illustrates the noise injection method to the queue length dimension using a discrete Gaussian distribution with $\mu = 0$ and the tails of the Gaussian approximately defined with the desired noise level. An example queue length vector with a length of 4 is shown on the left. After passing through the noise injection method, the resulting perturbed queue length vector is shown at the right. At the bottom left element, the sampled integer from the discrete Gaussian is -2, hence the true queue length is reduced by 2. The second element happens to sample 0, which retains the true queue from before. The third element illustrates if the sampled value is negative, and the true queue value is already at zero, the maximum it can be reduced is up to 0. The fourth element simply shows that a positive integer is sampled hence increasing the true queue value by 3.

occupancy actuations. Figure 1 illustrates the generic RL loop with agent and environment interaction. In the general RL scenario, the environment returns a state and reward that will be given directly to the RL agent. In this experiment, the queue length dimension is injected with synthetic noise before passing to the RL agent. Additionally, we investigate if the deep RL agent will leverage other state information such as occupancy and average speed when the queue length dimension does not accurately describe the current congestion in the environment. In this experiment, we train deep RL agents with the set of defined state spaces listed in Table 1. *SS:Basic* is defined with traffic signals and queue length information for each traffic direction. The other four definitions of state spaces include the same input as *SS:Basic* except for *SS:Pressure*, where the queue length information is described with residual queue [39]. *SS:Actuation* uses total number of actuation in the last time step as additional state input. *SS:Speed* uses speed as the additional state input, averaged across the number of vehicles that pass through the induction loop in the last time step. The final state definition, *SS:Speed Actuation*, represents the state space with traffic phase, queue length, average speed, and the number of actuation together.

Several studies have shown that it is possible to closely approximate the true total number of queue length if the appropriate sensor setup and condition is present [18, 24, 35]. To introduce uncertainties in the queue length dimension for training the deep RL agent, we add a perturbation δ_t into the state space S_t using a discrete Gaussian distribution. Figure 2 shows an example of the noise insertion to the queue length dimension. The discrete Gaussian distribution has a mean μ at 0 and the tails of the discrete Gaussian approxi-

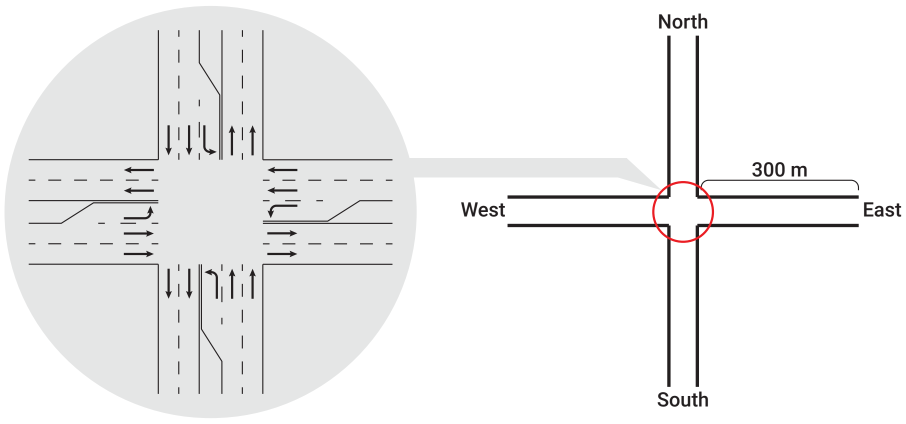

Fig. 3 Traffic Intersection Setup: We performed an experiment using a single intersection with a through and protected left turn. All four road segments are 300 meters long, with each road containing two lanes.

mately defined with the desired injection noise. For this example in Figure 2, the queue length dimension is 4, hence 4 separate samples from the discrete Gaussian distribution is obtained to apply to each element in the queue length vector. The sampled noise can be positive, zero, or negative. The true queue length can be reduced to zero at max if the sampled noise is negative and larger than the current queue element.

3.3 Action Space Formulation

This experimental setup consists of through and protected left-turn lanes (as seen in Figure 3). Since the intersection includes through and protected left turns, the agent will change through a set of pre-defined phase list configuration. The action space is a discrete action space, where $a_t = 0$ keeps the current phase configuration, $a_t = 1$ triggers to the next phase in the phase list, and $a_t = 2$ skips to the next two phases in the phase list. Suppose the agent decides to keep the current phase. In that case, the simulation will only step through one second of simulation time, allowing the agent to immediately make another decision in the next simulation step. To change the current phase configurations, the environment will simulate through the required yellow, red clearance, and minimum green time before allowing the agent to make another decision again.

3.4 Reward Formulation

Defining the correct reward function for the deep RL agent given the environment is fundamental to the convergence of the agent. The reward function acts as a feedback loop to inform how good the previous action decided by

Table 2 Model Hyperparameters

Parameter	Value
Explorer	Linear Decay
Optimizer	Adam
Minibatch	16
Replay start	50,000
Decay end	200,000
Gamma	0.90
Target update	2000
Epsilon	1e-5

the agent was conditioned upon the state information provided from the environment. The deep RL agent's goal is to eventually maximize the long-term rewards obtained for the whole trajectory. Hence, it is imperative that the correct reward function is crafted for the ASCT environment. For this experiment, we used a simplified version of the reward function used in Tan *et. al.* [37] as follows,

$$R = -\sum_{i \in lanes} Q_i - w_1 * \sum_{i \in lanes} O_i \tag{2}$$

The reward function in Equation 2 is defined by the summation of queue lengths Q_i, where i represents each direction-bound lanes in the intersection. The negative term associated together with it encourages the deep RL agent to maximize this variable such that congestion is minimized. The second variable is defined as the overflow variable O_i for all direction-bound lanes in the intersection. If a given lane reaches its maximum detectable vehicle queue lengths (i.e., if a sensor can detect up to 20 vehicles, but the true queue might be longer than that), this term will start a timer as long the given lane is still at the maximum detectable capacity. The associated weight parameter w_1 decides how heavy to penalize the agent for each second the lane is at maximum capacity. This parameter aims to prevent the agent from allowing other lanes to wait too long to avoid activating the overflow penalizing variable. The value of w_1 varies depending on each experimental setting via a trial-and-error process. For this experimental setting, the w_1 value is 5. Both variables' summation will drive the deep RL agent to minimize congestion at the intersection level while not keeping other lanes from waiting too long.

4 Experimental Setup

We conducted our experiment with Simulation of Urban MObility (SUMO) [14], an open-sourced traffic simulator capable of simulating microscopic traffic scenarios. We chose SUMO because it is a cross-platform compatible and a lightweight simulator. This allows the user to easily deploy and train multiple experimental runs anywhere. We also used another open-source library, ChainerRL, for deep RL agent model implementations and modifications.

Table 3 Performance results for both Nominal and Robust agent across different noise levels

Nominally Trained (Noise 0)											
State Space	Average Reward										
	0	1	2	3	4	5	6	7	8	9	10
SS:Basic	**-2.400**	-3.511	-4.955	-6.747	-9.317	-8.287	-10.079	-9.329	-8.738	-9.369	-9.223
SS:Actuation	-3.108	**-2.290**	-2.514	-3.400	-3.842	-4.294	-4.967	-5.220	-5.150	-4.875	-5.646
SS:Pressure	**-2.597**	-2.738	-3.253	-3.786	-5.365	-6.598	-6.950	-8.529	-7.076	-6.219	-7.246
SS:Speed	-3.945	**-3.247**	-3.479	-4.268	-4.137	-5.136	-5.721	-6.163	-6.425	-5.889	-6.258
SS:Speed Actuation	**-4.279**	-7.959	-11.495	-13.288	-15.503	-12.551	-12.742	-11.147	-12.483	-10.889	-13.431
Robustly Trained (Noise 5)											
State Space	Average Reward										
	0	1	2	3	4	5	6	7	8	9	10
SS:Basic	-6.858	-5.671	-4.660	-4.086	-3.606	-3.705	-3.312	-3.282	**-3.101**	-3.148	-3.470
SS:Actuation	-5.001	-4.784	-3.610	-3.211	-2.951	-2.710	-3.577	**-2.637**	-2.913	-3.675	-2.757
SS:Pressure	-5.515	-4.611	-3.691	-3.411	-3.242	-3.297	-3.139	**-2.964**	-3.121	-3.131	-3.238
SS:Speed	-6.126	-3.814	-3.198	-2.788	-3.029	-2.779	-2.785	-2.809	**-2.684**	-2.807	-2.801
SS:Speed Actuation	-3.627	-3.118	-3.032	-2.961	**-2.932**	-3.123	-3.156	-3.222	-3.056	-3.080	-3.134

We modeled our problem as a 4-way intersection problem with protected left-turns, as seen in Figure 3. Each direction bound (North, South, East, and West) has two through lanes and one protected left turn. We generated various traffic volumes for each training simulation, where one simulation episode simulates 4 hours of real-world traffic. Traffic input volume in Northbound and Southbound for this experiment is 500 ± 100 vehicles for each outbound direction. Similarly, traffic input volume in Eastbound and Westbound is 100 ± 50 vehicles for each outbound direction. We maintained heavier traffic to be N-S bound instead of generalizing it to W-S bound traffic since this experiment's scope is to train a robust deep RL agent instead of attaining state-of-the-art performance results.

For this experiment, we chose Double DQN [38] as the deep RL architecture because it is an improved DQN [28] in terms of stability of learning by reducing maximization bias using two Q-networks. The architecture consists of two fully connected layers to craft the agent's policy. The first layer consists of 64 neurons, followed by 32 neurons in the second layer. Each fully connected layer is coupled with a rectified linear unit (ReLU) function. The final output of the second layer is passed through a discrete action-value function with the shape of 3, which produces the Q-values of the action spaces available for this environment setup. The other model's hyperparameters are listed in Table 2.

We trained two deep RL agents: one with no noise induced into the queue length dimension (Nominal), another with noise $\delta = 5$ (Robust). We trained each agent across 5 different seeds to ensure the results produced are not seed-specific.

5 Results and Discussion

5.1 Robustness Improvements

In this section, we want to study the effectiveness of the training with noisy queue length approximation. We tested each Nominal and Robust agent with noise ranging from 0 to 10. Since each agent is trained with 5 different seeds, each trained seed agent is tested across 5 different seeds as well. This produces

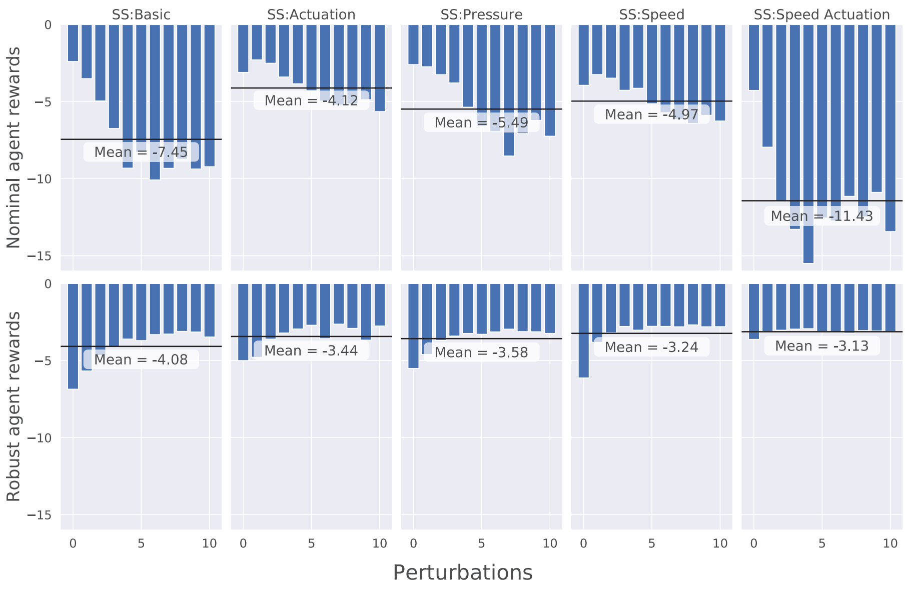

Fig. 4 Nominal vs Robust Performance: This plot illustrates overall inference performance for deep RL agents trained nominally and robustly. The top row represents Nominal agents, while the bottom row represents Robust agents. Each column represents the state space set used to train deep RL agents. The y-axis for each row shows the rewards obtained during test time to distinguish good performers against poor performers. In an ideal situation with no congestion, the reward will be 0. Hence a good performing agent will obtain less negative rewards (closer to 0) while a bad performing agent will receive more negative rewards (further from 0). The x-axis shows the noise injection level added to the queue length vector. 0 represents no perturbation added, while 10 represents ±10 if possible. The horizontal bar across each column shows the average performance across perturbations.

25 testing results in which the average of all 25 runs is taken as the result for each noise level. This similar process is repeated for each different state space listed in Tab. 1. All results across state space and noise level are shown in Tab. 3 and Fig. 4.

5.1.1 Nominal Performance

Generally, the Nominal agent would perform the best in the clean environment $\delta = 0$. The summary results in Tab. 3 shows that three of out five Nominal agents performed the best in the clean environment (indicated by bold). Both remaining Nominal agents trained with *SS:Actuation* and *SS:Speed* performed the best in the environment with very minimal perturbation ($\delta = 1$). As the synthetic noise perturbation increases, the performance of each agent trained with corresponding state-space decreases. *SS:Speed Actuation* is heavily affected by the increase in synthetic noise perturbation, followed by *SS:Basic*. *SS:Actuation*, *SS:Pressure*, and *SS:Speed* performance did decrease as the noise level increased, but not as significantly compared to *SS:Basic* and *SS:Speed Actuation*. For *SS:Basic*, a decrease in agent's performance along with the increase in noise level is understandable since it relies on queue length to describe the current state representation of the environment. However, *SS:Speed Actuation* contains more state information to understand the current state conditions. A possible explanation would be that the agent trained with *SS:Speed Actuation* did not converge to the optimal policy yet given the same training parameters as other state spaces sets. From Tab. 3, the best performance for *SS:Speed Actuation* is much worse than other state-space sets.

5.1.2 Robust Performance

By introducing perturbation noise $\delta = 5$ during training, the deep RL agent learns a more robust policy to maximize reward instead of allocating heavy attention on the queue length dimension. From the Robust model performance (bottom row) in Figure 4, it is evident that the average performance improved across increasing noise levels for each state space as compared to their nominal agent counterpart. The average performance improvement comes from the fact that the Robust agent is trained with $\delta = 5$, a moderate noise level compared to the range of noise we use for testing. Trivially, since the noise generation is based on a discretized Gaussian distribution, as seen in Figure 2, the deep RL agent's policy should perform well within the presence of noise level ± 5. Environments with little to no noise introduced did not perform as well because those environments have minimal perturbation in the queue length dimension. As the noise level increases, the deep RL agent's rewards increase gradually until it plateaus as the noise approaches 5. However, some state sets performed better than others despite the same noise setting being imposed on the environment. This might indicate that some additional states might have helpful information for the deep RL to describe the current environment state other than just queue length. In *SS:Basic*, we observe that it is one of the worst-performing state definitions. Even after increasing noise, the deep RL agent did not manage to attain reward of more than -3 in Table 3. *SS:Actuation*, *SS:Pressure*, and *SS:Speed* had similar performances on average. This is interesting because this shows other state spaces sets such as actuation, average speed, and residual queue helps to infer the environment state well enough to perform better than directly utilizing queue length alone. Finally, *SS:Speed Actuation* performance is almost the same throughout all noise levels. Since the queue length dimension is no longer as reliable as before, *SS:Speed Actuation* has more state information to utilize. The deep RL agent learns to diversify and use average speed, actuation, and the noisy queue length information to maximize its rewards. Although the agent trained with *SS:Speed Actuation* did not receive the best rewards compared to other state spaces, it produced a more robust policy against various noise levels.

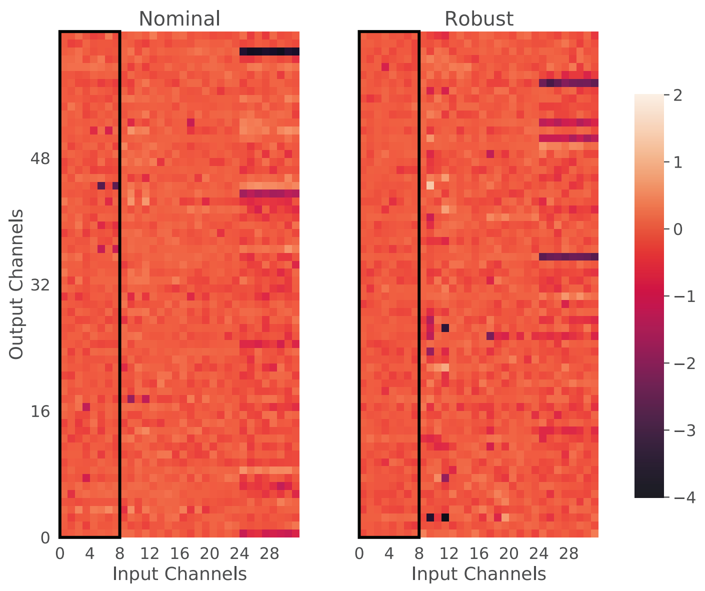

Fig. 5 Nominal and Robust agent weights trained with SS:Speed Actuation. The bottom x-axis represents the input dimension of 32. The inputs are grouped by the type of state information: queue length, average speed, actuation, and phase signals. Each group is 8 elements long. The Nominal agent weight distribution is nontrivial (boxed region) to fully utilize the queue length input. This generates better latent vector representation in the first layer. However, the Robust agent weights show a smoother weight distribution compared to the Nominal agent, indicating the agent learned not to overly rely on the queue length inputs to represent the latent vectors.

5.2 Visual Improvements

To further understand which aspect of the deep RL agent is improved upon from its nominal counterpart, we chose to compare the differences between the Nominal and Robust agents only with *SS:Speed Actuation*. This section will investigate the visual changes in the agent's policy and change in performance.

5.2.1 Weights Visualization

To understand how the policy of the deep RL agent adapted to the noisy queue approximation, we decided to inspect and compare the weights of both the Nominal agent and the Robust agent. Figure 5 visualizes the first layer weights of the deep RL agent trained using *SS:Speed Actuation* state space. The left column shows the Nominal agent, while the right column shows the Robust agent. The length of the state space for *SS:Speed Actuation* is 32, which reflects the input channels in the x-axis. The output channel is 64, which is consistent in Sec. 4. The input vector is grouped by the type of state information: queue length, average speed, actuation, and phase signal. One empirical observation is the difference between both agent's weights in the queue length dimension.

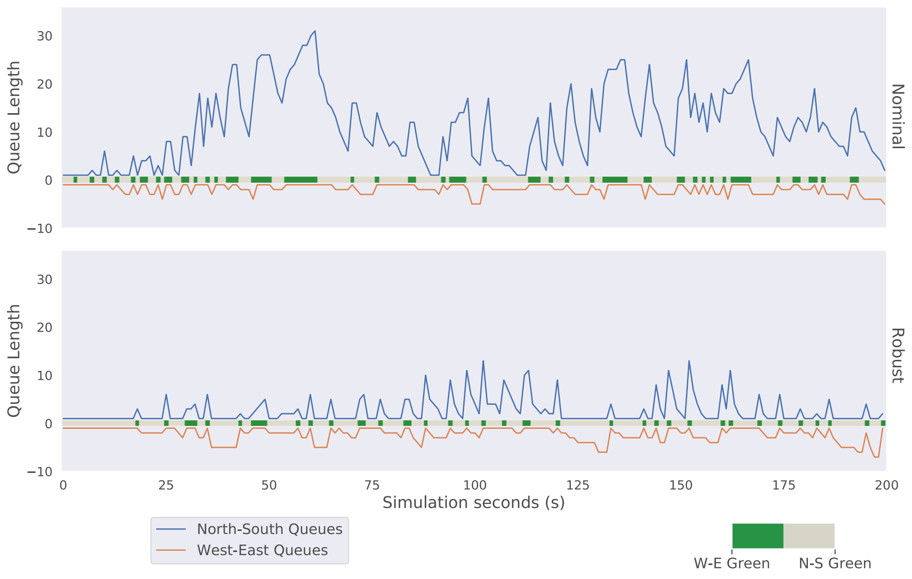

Fig. 6 Simulation Time Performance for SS:Speed Actuation: This figure shows the performance of Nominal agent (top row) versus Robust agent (bottom row) across the simulation from time interval $t = (0, 200)$ in a noise injection environment ($\delta = 10$).The y-axis represents the queue length values, while the x-axis represents simulation seconds. Spanning along the x-axis direction in each subplot, there is a summation in queue lengths in the North-South direction (blue line) and West-East direction (orange line). In between those two lines are the current simplified phase signal, either green in North-South or West-East. A quick observation shows the Nominal agent struggles to keep the North-South congestion low showing a severe noise is highly likely injected during certain periods of time. However, the Robust agent manages to prevent an accumulation of congestion despite the same traffic and noise being introduced.

This suggests that by training the agent with discrete Gaussian noise, the agent suggests a better generalization compared to exploiting the queue length dimension alone. In the Nominal agent, the highlighted column contains more differences in values between each other. In contrast, the Robust agent has averaged values across the columns. This observation is consistent across all seeds and all state definitions in this experiment.

5.2.2 Simulation Visualization

Figure 6 illustrates the simulation performance during the evaluation environment with noise level $\delta = 5$, where the heavy traffic flow is on the North-South bound traffic flow. The y-axis represents the queue length accumulation at the given time t, while the x-axis represents the time simulation time. The simplified phase is shown along the x-axis at the given time t. The darker green color represents the green phase given for West-East bound vehicles, while the lighter green would correspond to the green phase for North-South bound vehicles. The top and bottom row corresponds to the Nominal and Robust agents' performance, respectively. The Nominal agent has shown to struggle with correctly discerning the high traffic flow in the West-East direction due to the poisoned state space from $t = 25$ to $t = 60$. This resulted in incorrectly giving green to West-East direction even though the proportion of congestion was clearly in the North-South direction. From $t = 75$, the nominal agent managed to clear off the congestion built up in the W-E direction, possibly because the state space at that time was not heavily perturbed. Not long after, when $t = 115$, there was heavy noise which caused the agent to incorrectly cause another major congestion in the North-South direction. This shows how fragile an agent trained with the assumption of perfect congestion knowledge can easily cause a build-up in congestion, which is not desirable to drivers. Given the same traffic volume, the Robust agent managed to reduce the congestion in the North-South direction. This caused a slight increase in volume in the West-East direction. However, this is still highly desirable because the robust agent managed to curb the congestion build-up from the rush of volume in the North-South direction.

5.3 Realistic Noise Simulation as a Truck Event

In this section, we are interested in testing the robustness of the Robust agent under realistic noise settings in the real world. These realistic noise characteristics are often random and unknown, making it hard to train realistic noise-aware deep RL agents. We attempt to emulate realistic scenarios where we can formulate the noise conditions based on known a known Truck Event problem.

Queue prediction systems generally perform the best for standard-sized passenger vehicles with total vehicle body lengths of 5 meters. These system's performance deteriorates when trucks and semi-trailers are present. On average, trucks and semi-trailers are longer and larger than passenger vehicles. This can potentially cause the queue prediction systems to underestimate the true queue (Vision-based system), overestimate the true queue(Induction Loop-based system), or both (Vision-Induction Hybrid). A vision-based system relies on computer vision algorithms to count the total number of idle vehicles in the intersection. A truck could potentially block passenger vehicles that are either directly behind or adjacent to the truck, ultimately undercounting the total number of idle vehicles in the intersection. Induction Loop-based system relies on a combination of advance detectors, stop bar detectors, and phase change data to determine the queue lengths [35]. A truck might potentially trigger double or triple counts due to the length of the truck or multiple actuation triggers. This would ultimately force the queue prediction system to overestimate the true queue in the intersection. The Hybrid system relies on both traffic cameras and induction loops to determine the queues, which could cause both underestimating and overestimating the true queues in the intersection.

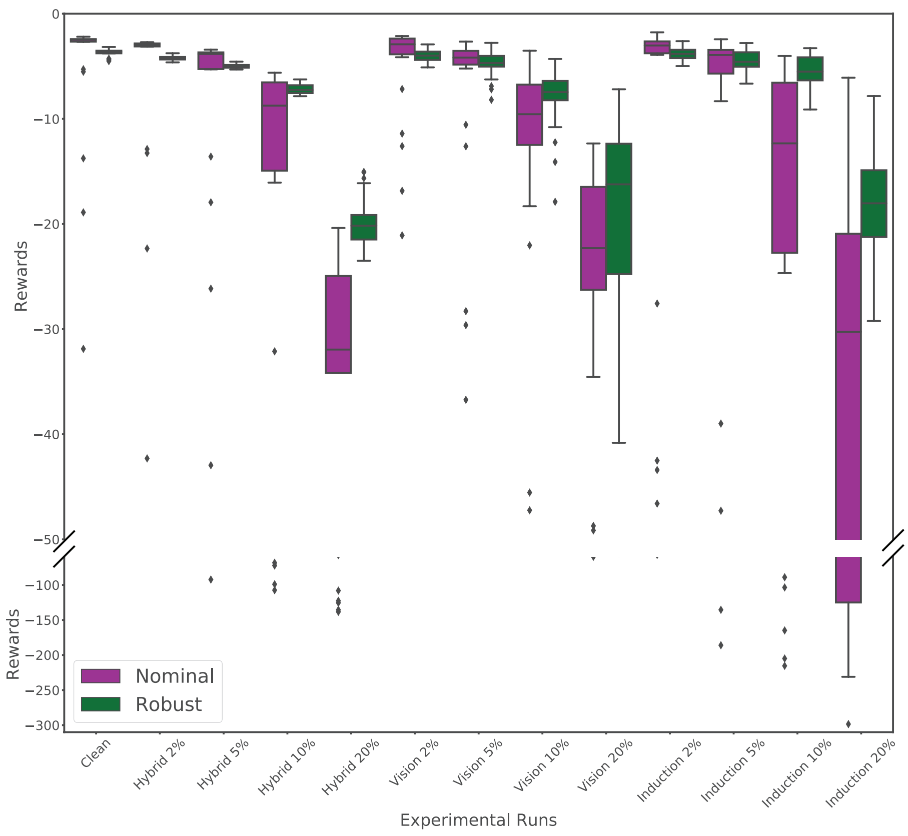

Fig. 7 Agent's Performance on Truck Event: This illustration shows Nominal and Robust agent performance across different experimental runs with pre-defined environment conditions. The x-axis represents different experimental runs; the y-axis shows the rewards. A good performing agent would obtain a higher reward. Notice the y-axis on the top has a different tick interval compared to the bottom. A quick glance shows how the Nominal agent is more vulnerable to noisy perturbations (especially with higher levels of noise) compared to the Robust agent.

To emulate the scenarios above, we introduced trucks into the simulation, where each truck has a probability of disrupting the queue prediction system. If a truck is approaching the intersection on the eastbound through lane (East-West), the simulation will specifically insert noise perturbation into the eastbound through queue value only. This rule is applied to all trucks that are approaching the intersection. Since each truck approaching the intersection can disrupt the queue prediction system, we vary the total number of trucks with a density parameter 2%, 5%, 10%, and 20% of the total passenger vehicles in the simulation. Three different queue prediction systems - Vision, Induction, and Hybrid, can be approximately replicated by using a discrete Gaussian distribution. To replicate the Vision-based system, we sample from the discrete Gaussian distribution and clip all positive values sampled. This

Table 4 Result Summary for Truck Event

Experimental Run	Agent	Mean + Standard Deviation	Min	25%	50%	75%	Max
Clean	Nominal	-4.9 ± 6.8	-31.8	-2.6	-2.4	-2.4	-2.1
	Robust	-3.7 ± 0.3	-4.5	-3.7	-3.6	-3.5	-3.1
Hybrid 2%	Nominal	-8.0 ± 12.8	-52.5	-3.1	-2.9	-2.8	-2.7
	Robust	-4.2 ± 0.2	-4.6	-4.3	-4.2	-4.1	-3.7
Hybrid 5%	Nominal	-10.9 ± 19.2	-92.3	-5.2	-3.8	-3.7	-3.4
	Robust	-5.0 ± 0.2	-5.3	-5.1	-5.0	-4.9	-4.5
Hybrid 10%	Nominal	-22.3 ± 30.0	-107.4	-14.9	-8.7	-6.5	-5.6
	Robust	-7.1 ± 0.5	-7.8	-7.5	-7.2	-6.8	-6.2
Hybrid 20%	Nominal	-49.1 ± 40.1	-138.3	-34.1	-31.9	-24.9	-20.3
	Robust	-19.8 ± 2.4	-23.5	-21.5	-20.1	-19.1	-15.0
Vision 2%	Nominal	-5.0 ± 5.0	-21.1	-3.8	-2.9	-2.3	-2.1
	Robust	-4.0 ± 0.6	-5.1	-4.4	-4.0	-3.6	-2.9
Vision 5%	Nominal	-7.8 ± 9.3	-36.7	-4.8	-4.2	-3.5	-2.6
	Robust	-4.7 ± 1.3	-8.2	-5.0	-4.7	-4.0	-2.8
Vision 10%	Nominal	-12.7 ± 11.0	-47.2	-12.5	-9.5	-6.7	-3.5
	Robust	-8.0 ± 3.0	-17.9	-8.2	-7.4	-6.4	-4.3
Vision 20%	Nominal	-25.9 ± 13.3	-59.7	-26.2	-22.3	-16.4	-12.3
	Robust	-20.2 ± 10.7	-54.2	-24.7	-16.2	-12.3	-7.2
Induction 2%	Nominal	-11.0 ± 17.1	-57.1	-3.7	-3.0	-2.6	-1.7
	Robust	-3.8 ± 0.6	-5.0	-4.2	-3.8	-3.4	-2.6
Induction 5%	Nominal	-19.7 ± 44.4	-186.0	-5.7	-3.9	-3.4	-2.4
	Robust	-4.3 ± 0.9	-6.6	-5.0	-4.5	-3.6	-2.8
Induction 10%	Nominal	-40.2 ± 63.6	-215.4	-22.7	-12.3	-6.5	-4.0
	Robust	-5.6 ± 1.6	-9.1	-6.3	-5.5	-4.1	-3.3
Induction 20%	Nominal	-85.3 ± 98.9	-327.2	-125.0	-30.2	-20.9	-6.1
	Robust	-18.3 ± 5.4	-29.2	-21.2	-18.0	-14.9	-7.8

leaves only non-positive values in the sample for possible perturbation. Similarly, to replicate the Induction-based system, we sample from the discrete Gaussian distribution and clip all negative values to obtain only non-negative samples. The Hybrid system will use the samples directly from the discrete Gaussian distribution without any clippings.

Results from Fig. 7 show box plots of nominal and robust agent performance in each experimental run. The x-axis represents rewards obtained by agents, while the y-axis represents different experimental runs. Each experiment contains two box plots representing Nominal and Robust agents. A good performing agent would ideally achieve rewards closer to 0. Note that the top subplot y-axis reward intervals are different from the bottom subplot. The full data points of this box plot are shown in Tab. 4.

From the left of Fig. 7, the performance of both Nominal and Robust agent in the Clean experiment are expected. The Clean experiment returns the exact queue in the simulation without any noise introduced. The Nominal agent performance is better than the Robust agent, with the first quartile, median, and third quartile values of -2.6, -2.4, and -2.4 respectively, as seen in Tab. 4. However, some of the Nominal agents run attained a very weak performance seen as outliers in Fig. 7. By visual comparison, the number of outliers for the Nominal agent versus the Robust agent is drastically different. This might be possible since the agent trained with *SS:Speed Actuation* might not yet converge to an optimal policy as mentioned in Sec. 5.1.1. The next experimental runs are Hybrid with increasing truck density of 2%, 5%, 10%, and 20%. Across four of these runs, both agents' performance reduced when comparing to the Clean experiment. As the truck density increases, the Nominal agent performance deteriorates severely compared to the Robust agent counterpart. The Robust agent might have seen many of these inaccurate queue lengths during training time, hence it learned a more robust policy against those scenarios. In Tab. 4, the standard deviation of the Nominal agent across all four Hybrid runs is very large when compared to the Robust agent, thus attaining a more consistent performance. The next four runs are Vision runs with similar increments in truck densities. The immediate observation from these runs reveals that the inter-quartile range is higher than both Hybrid and Induction runs in their respective densities. This reveals that underestimating queues does not affect the performance of both Nominal and Robust agents as severely compared to Hybrid and Induction runs. Finally, the Induction runs illustrate that the Nominal agent's performance is greatly affected as the truck density increases. For Induction 20% run, the inter-quartile range spans from -20.93 (upper quartile) to -125.02 (lower quartile). This reveals that a nominally trained agent does not perform well if the true queue length is perturbed by overestimating them. Inversely, the Robust agent performs better in the Induction runs compared to Vision runs. This suggests the Robust agent learns to maximize its rewards in the overestimating scenario more than its underestimating counterpart since the Nominal agent's performance suffered drastically in the Induction runs.

6 Conclusion & Future Works

In this paper, we proposed robustly trained deep RL agents with a noisy queue length dimension mimicking situations in the real world, where queue length information acquisition might not be accurate. By leveraging reliable sensors such as induction loop sensor, we can directly obtain crucial information such as average speed and actuation. The deep RL agent can utilize those sensor information along with the noisy queue length dimension to perform in a gracefully degraded condition and not fail catastrophically across a range of noise levels. We discussed the performances of the deep RL agent trained with different state space definitions and how it performed in various noise levels. We also empirically show the difference between a nominally trained agent and a robustly trained agent in the policy and simulation performance. Finally, we emulated a realistic noise scenario to test the robustness of the train deep RL agent. A potential future direction for this work could be investigating how this robust training affects a corridor-level ASTC, since a corridor traffic control requires more coordination between traffic signals at each intersection.

Acknowledgements This work was supported in part by NSF grant CNS-1845969. We would like to thank Subhadipto Poddar for his insights in general traffic intersection knowledge.

Conflict of interest

Anuj Sharma works for ETALYC Inc. that works in the area of adaptive signal control.

References

1. Arel, I., Liu, C., Urbanik, T., Kohls, A.G.: Reinforcement learning-based multi-agent system for network traffic signal control. IET Intelligent Transport Systems **4**(2), 128–135 (2010)
2. Barceló, J., Casas, J.: Dynamic network simulation with aimsun. In: Simulation approaches in transportation analysis, pp. 57–98. Springer (2005)
3. Bellman, R.: Dynamic programming. Science **153**(3731), 34–37 (1966)
4. Chakraborty, P., Adu-Gyamfi, Y.O., Poddar, S., Ahsani, V., Sharma, A., Sarkar, S.: Traffic congestion detection from camera images using deep convolution neural networks. Transportation Research Record **2672**(45), 222–231 (2018)
5. Chu, T., Wang, J., Codecà, L., Li, Z.: Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Transactions on Intelligent Transportation Systems **21**(3), 1086–1095 (2019)
6. Cohen, J.M., Rosenfeld, E., Kolter, J.Z.: Certified adversarial robustness via randomized smoothing. CoRR **abs/1902.02918** (2019). URL http://arxiv.org/abs/1902.02918
7. Esfandiari, Y., Ebrahimi, K., Balu, A., Elia, N., Vaidya, U., Sarkar, S.: A saddle-point dynamical system approach for robust deep learning. arXiv preprint arXiv:1910.08623 (2019)
8. Group, P.: Vissim. http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/ (2019)
9. Havens, A., Jiang, Z., Sarkar, S.: Online robust policy learning in the presence of unknown adversaries. In: Advances in Neural Information Processing Systems, pp. 9916–9926 (2018)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016)
11. Joshi, A., Mukherjee, A., Sarkar, S., Hegde, C.: Semantic adversarial attacks: Parametric transformations that fool deep classifiers. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4773–4783 (2019)
12. Koonce, P., Rodegerdts, L., Lee, K., Quayle, S., Beaird, S., Braud, C., Bonneson, J., Tarnoff, P., Urbanik, T.: Traffic Signal Timing Manual: Chapter 5 - Office of Operations (2010). URL http://ops.fhwa.dot.gov/publications/fhwahop08024/chapter5.htm
13. Koonce, P., Rodegerdts, L., Lee, K., Quayle, S., Beaird, S., Braud, C., Bonneson, J., Tarnoff, P., Urbanik, T.: Traffic Sig-

nal Timing Manual: Chapter 6 - Detectors (2010). URL http://ops.fhwa.dot.gov/publications/fhwahop08024/chapter5.htm
14. Krajzewicz, D., Hertkorn, G., Rössel, C., Wagner, P.: Sumo (simulation of urban mobility)-an open-source traffic simulation. In: Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002), pp. 183–187 (2002)
15. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. arXiv preprint arXiv:1611.01236 (2016)
16. Lasley, P.: 2019 urban mobility report (2019)
17. Lee, X.Y., Ghadai, S., Tan, K.L., Hegde, C., Sarkar, S.: Spatiotemporally constrained action space attacks on deep reinforcement learning agents. In: AAAI, pp. 4577–4584 (2020)
18. Li, B., Cheng, W., Li, L.: Real-time prediction of lane-based queue lengths for signalized intersections. Journal of Advanced Transportation **2018** (2018)
19. Li, L., Lv, Y., Wang, F.Y.: Traffic signal timing via deep reinforcement learning. IEEE/CAA Journal of Automatica Sinica **3**(3), 247–254 (2016)
20. Li, X., Ouyang, Y.: Reliable traffic sensor deployment under probabilistic disruptions and generalized surveillance effectiveness measures. Operations research **60**(5), 1183–1198 (2012)
21. Liang, X., Du, X., Wang, G., Han, Z.: A deep reinforcement learning network for traffic light cycle control. IEEE Transactions on Vehicular Technology **68**(2), 1243–1253 (2019)
22. Lin, Y., Dai, X., Li, L., Wang, F.Y.: An efficient deep reinforcement learning model for urban traffic control. arXiv preprint arXiv:1808.01876 (2018)
23. Liu, C., Zhao, M., Sharma, A., Sarkar, S.: Traffic dynamics exploration and incident detection using spatiotemporal graphical modeling. Journal of Big Data Analytics in Transportation **1**(1), 37–55 (2019)
24. Liu, H.X., Wu, X., Ma, W., Hu, H.: Real-time queue length estimation for congested signalized intersections. Transportation research part C: emerging technologies **17**(4), 412–427 (2009)
25. Liu, M., Deng, J., Xu, M., Zhang, X., Wang, W.: Cooperative deep reinforcement learning for tra ic signal control. In: The 7th International Workshop on Urban Computing (UrbComp 2018) (2017)
26. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083 (2017)
27. Miller, A.J.: Settings for fixed-cycle traffic signals. Journal of the Operational Research Society **14**(4), 373–386 (1963)
28. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
29. Morimoto, J., Doya, K.: Robust reinforcement learning. Neural computation **17**(2), 335–359 (2005)

30. Mukherjee, A., Joshi, A., Hegde, C., Sarkar, S.: Semantic domain adaptation for deep classifiers via gan-based data augmentation
31. Van der Pol, E., Oliehoek, F.A.: Coordinated deep reinforcement learners for traffic light control. Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016) (2016)
32. Rajagopal, R., Varaiya, P.P.: Health of California's loop detector system. California PATH Program, Institute of Transportation Studies, University of (2007)
33. Rodrigues, F., Azevedo, C.L.: Towards robust deep reinforcement learning for traffic signal control: Demand surges, incidents and sensor failures. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 3559–3566. IEEE (2019)
34. Roess, R.P., Prassas, E.S., McShane, W.R.: Traffic engineering. Pearson/Prentice Hall (2004)
35. Sharma, A., Bullock, D.M., Bonneson, J.A.: Input–output and hybrid techniques for real-time prediction of delay and maximum queue length at signalized intersections. Transportation Research Record **2035**(1), 69–80 (2007)
36. Tan, K.L., Esfandiari, Y., Lee, X.Y., Sarkar, S., et al.: Robustifying reinforcement learning agents via action space adversarial training. In: 2020 American Control Conference (ACC), pp. 3959–3964. IEEE (2020)
37. Tan, K.L., Poddar, S., Sarkar, S., Sharma, A.: Deep reinforcement learning for adaptive traffic signal control. In: ASME 2019 Dynamic Systems and Control Conference. American Society of Mechanical Engineers Digital Collection (2019)
38. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Thirtieth AAAI conference on artificial intelligence (2016)
39. Varaiya, P.: Max pressure control of a network of signalized intersections. Transportation Research Part C: Emerging Technologies **36**, 177–195 (2013)
40. Webster, F.: Traffic signal settings, road research technical paper no. 39. Road Research Laboratory (1958)
41. Wei, H., Chen, C., Wu, K., Zheng, G., Yu, Z., Gayah, V., Li, Z.: Deep reinforcement learning for traffic signal control along arterials (2019)
42. Wei, H., Zheng, G., Yao, H., Li, Z.: Intellilight: A reinforcement learning approach for intelligent traffic light control. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2496–2505 (2018)
43. Wiering, M.: Multi-agent reinforcement learning for traffic light control. In: Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000), pp. 1151–1158 (2000)
44. Wu, L., Liu, C., Huang, T., Sharma, A., Sarkar, S.: Traffic sensor health monitoring using spatiotemporal graphical modeling. In: Proceedings of the 2nd ACM SIGKDD Workshop on Machine Learning for Prognostics and Health Management, pp. 13–17 (2017)