Uncertainty-aware Grounded Action Transformation towards Sim-to-Real Transfer for Traffic Signal Control

Longchao Da
New Jersey Institute of Technology
Newark, New Jersey, USA
ld49@njit.edu

Hao Mei
New Jersey Institute of Technology
Newark, New Jersey, USA
hm467@njit.edu

Romir Sharma
The West Windsor-Plainsboro High School South
West Windsor, New Jersey, USA
sharmaromir@gmail.com

Hua Wei*
New Jersey Institute of Technology
Newark, New Jersey, USA
hua.wei@njit.edu

ABSTRACT

Traffic signal control (TSC) is a complex and important task that affects the daily lives of millions of people. Reinforcement Learning (RL) has shown promising results in optimizing traffic signal control, but current RL-based TSC methods are mainly trained in simulation and suffer from the performance gap between simulation and the real world. In this paper, we propose a simulation-to-real-world (sim-to-real) transfer approach called UGAT, which transfers a learned policy trained from a simulated environment to a real-world environment by dynamically transforming actions in the simulation with uncertainty to mitigate the domain gap of transition dynamics. We evaluate our method on a simulated traffic environment and show that it significantly improves the performance of the transferred RL policy in the real world.

CCS CONCEPTS

• **Information systems → Decision support systems**.

KEYWORDS

traffic signal control, uncertainty quantification, simulation to reality transfer

1 INTRODUCTION

Traffic Signal Control (TSC) is crucial for improving traffic flow, reducing congestion in modern transportation systems, and benefiting individuals and society as a whole. Traffic signal control remains an active research topic because of the high complexity of the problem. The traffic situations are highly dynamic and require traffic signal plans to adapt to different situations, making it necessary to develop effective algorithms that can adjust to changing traffic conditions.

Recent advances in reinforcement learning (RL) techniques have shown superiority over traditional approaches in TSC [9]. In RL, an agent aims to learn a policy through trial and error by interacting with an environment to maximize the cumulative expected reward over time. The biggest advantage of RL is that it can directly learn how to generate adaptive signal plans by observing the feedback from the environment.

One major issue of applying current RL-based traffic signal control approaches in the real world is that these methods are mostly trained in simulation and suffer from the performance gap between simulation and the real world. Training in the simulation provides an efficient way to avoid the cost of learning RL-based policies in the real world. However, due to the high complexity of real-world dynamics, simulations are not always representative of real-world scenarios [4], which can limit the performance of RL-based TSC models in practice. For example, a traffic simulator might have default acceleration or deceleration settings for vehicles, while in the real world, the vehicle settings can vary widely depending on weather conditions, vehicle types, and many other factors. The inherent mismatches between simulation and real-world hinder RL-based models trained in simulation from achieving similar performance in the real world, as is shown in Figure 1.

To close this gap, existing literature in TSC has been focusing on modifying the traffic simulator to better match the real world with real-world data [14] so that the policy or model can be transferred from simulation-to-real-world (sim-to-real) without much performance gap. However, in practice, the internal parameters of the simulator cannot be easily modified. To address this challenge, Grounded Action Transformation is a popular technique that seeks to induce simulator transitions to more closely match the real world. However, the current GAT technique is mainly applied to robotics, and few studies have been conducted on the traffic signal control problem.

In this paper, we present Uncertainty-aware Grounded Action Transformation (UGAT), an approach that bridges the domain gap of transition dynamics by dynamically transforming actions in the simulation with uncertainty. UGAT learns to mitigate the discrepancy between the simulated and real-world dynamics under the framework of grounded action transformation (GAT), which learns an inverse model that can generate an action to ground the next state in the real world with a desired next state predicted by the forward model learned in simulation. Specifically, to avoid enlarging the transition dynamics gap induced by the grounding actions with high uncertainty, UGAT dynamically decides when to transform the actions by quantifying the uncertainty in the forward model. Our experiments demonstrate the existence of the performance gap in traffic signal control problems and further show that UGAT has a good performance in mitigating the gap with higher efficiency and stability.

*Corresponding Author

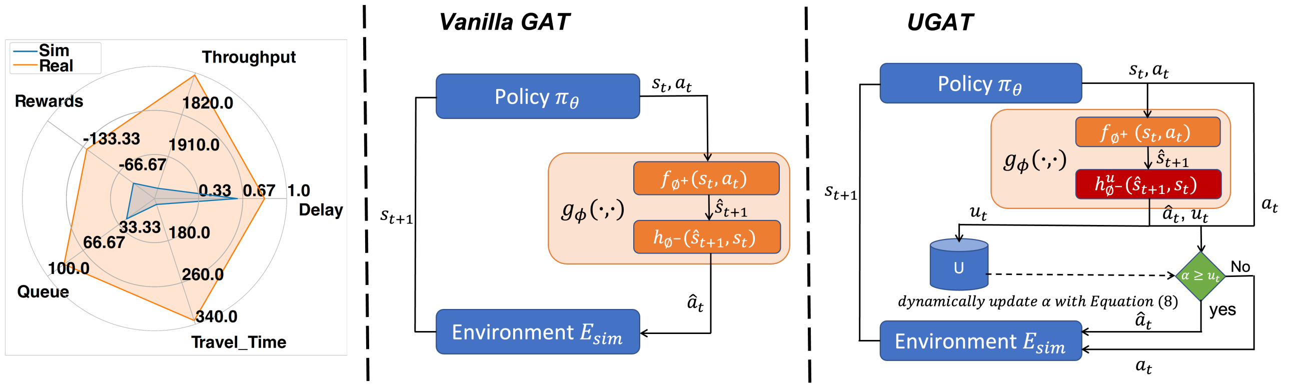


Figure 1: The performance gap in sim-to-real transfer and the schematic of GAT and UGAT. Left: The method [13] trained in simulation has a performance drop when transferred to the real world in all five evaluation metrics in TSC. Middle: GAT method takes grounded action $\hat{a}_t$ when a policy returns an action a_t from the E_{sim}. Grounded actions taken with high model uncertainty on $g_{\phi(\cdot,\cdot)}$ will enlarge the transition between P_θ and P^*, making the gap between E_{sim} and E_{real} large and policy learning step not stable. Right: UGAT quantifies the model's uncertainty and decide to take or reject the grounded action $\hat{a}_t$ based on the current output of model uncertainty u_t given the current s_t state and action a_t. This behavior will mitigate the gap between P_ϕ and P^* and make the policy learning step stable.

2 PRELIMINARIES

This section formalizes the TSC problem, its RL solutions and introduces the grounded action transformation (GAT) framework for sim-to-real transfer.

2.1 Concepts of TSC and RL Solutions

In the TSC problem, following existing work [1, 11, 13, 16], each traffic signal controller decides the phase of an intersection, which is a set of pre-defined combinations of traffic movements that do not conflict while passing through the intersection. Given the current condition of this intersection, the traffic signal controller will choose a phase for the next time interval Δt to minimize the average queue length on lanes around this intersection. The TSC problem is defined as an MDP which could be characterized by $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$ where each $\mathcal{S}$ and $\mathcal{A}$ are the system state space, and action space, transition dynamics P describes the probability distribution of next state $s_{t+1} \in \mathcal{S}$, Reward r is a scalar return from the environment, the π_θ represents policy and γ is the discount factor.

An RL approach solves this problem by maximizing the long-term expectation of discounted accumulation reward adjusted by discount factor γ. The discounted accumulated reward is

$$\mathbb{E}_{(s_t,a_t)\sim(\pi_\theta,\mathcal{M})}\left[\sum_{t=0}^{T} \gamma^{T-t} r_t(s_t, a_t)\right] \tag{1}$$

Since the action space $\mathcal{A}$ is discrete, we follow the past work using Deep Q-network (DQN) [13] to optimize the RL policy. In the past RL-based TSC works, the above-described procedure is conducted in the simulation environment E_{sim}.

2.2 Grounded Action Transformation

Grounded action transformation (GAT) is a framework originally proposed in robotics to improve robotic learning by using trajectories from the physical world E_{real} to modify E_{sim}. Under the GAT framework, MDP in E_{sim} is imperfect and modifiable, and it can be parameterized as a transition dynamic $P_\phi(\cdot|s, a)$. Given real-world dataset $\mathcal{D}_{real} = \{\tau^1, \tau^2, \ldots, \tau^I\}$, where $\tau^i = (s^i_0, a^i_0, s^i_1, \ldots, a^i_{T-1}, s^i_T)$ is a trajectory collected by running a policy π_θ in E_{real}, GAT aims to minimize differences between transition dynamics by finding ϕ^*:

$$\phi^* = \arg\min_\phi \sum_{\tau^i \in \mathcal{D}_{real}} \sum_{t=0}^{T-1} d(P^*(s^i_{t+1}|s^i_t, a^i_t), P_\phi(s^i_{t+1}|s^i_t, a^i_t)) \tag{2}$$

where $d(\cdot)$ is the distance between two dynamics, P^* is the real world transition dynamics, and P_ϕ is the simulation transition dynamics.

To find ϕ efficiently, GAT takes the agent's state s_t and action a_t predicted by policy π_θ as input and outputs a grounded action $\hat{a}_t$. Specifically, it uses an action transformation function parameterized with ϕ:

$$\hat{a}_t = g_\phi(s_t, a_t) = h_{\phi^-}(s_t, f_{\phi^+}(s_t, a_t)) \tag{3}$$

which includes two specific functions: a forward model f_{ϕ^+}, and an inverse model h_{ϕ^-}, as is shown in Fig. 1.

- *The forward model* f_{ϕ^+} is trained with the data from E_{real}, aiming to predict the next possible state $\hat{s}_{t+1}$ given current state s_t and action a_t:

$$\hat{s}_{t+1} = f_{\phi^+}(s_t, a_t) \tag{4}$$

- *The inverse model* h_{ϕ^-} is trained with the data from E_{sim}, aiming to predict the possible action $\hat{a}_t$ that could lead the current state s_t to the given next state. Specifically, the inverse model in GAT takes $\hat{s}_{t+1}$, the output from the forward model, as its input for the next state:

$$\hat{a}_t = h_{\phi^-}(\hat{s}_{t+1}, s_t) \tag{5}$$

Given current state s_t and the action a_t predicted by the policy π_θ, the grounded action $\hat{a}_t$ takes place in E_{sim} will make the resulted s_{t+1} in E_{sim} close to the predicted next state $\hat{s}_{t+1}$ in E_{real}, which makes the dynamics $P_\phi(s_{t+1}|s_t, \hat{a}_t)$ in simulation close to the real-world dynamics $P^*(\hat{s}_{t+1}|s_t, a_t)$. Therefore, the policy π_θ is learned in E_{sim} with P_ϕ close to P^* will have a smaller performance gap when transferred to E_{real} with P^*.

3 METHODS

To mitigate the gap in the transition dynamics between traffic simulations and real-world traffic systems, we use the vanilla GAT and analyze its limitations. To overcome the problem in vanilla GAT, we propose UGAT to further leverage uncertainty quantification to take grounded action dynamically.

3.1 Vanilla GAT for TSC

We use the vanilla GAT for the traffic signal control problem by specifying the learning of f_{ϕ^+} and h_{ϕ^-}:

• *The forward model* $f_{\phi^+}(s_t, a_t)$ in traffic signal control problem predicts the next traffic state $\hat{s}_{t+1}$ in the real world given taken action a_t and the current traffic state s_t. We approximate f_{ϕ^+} with a deep neural network and optimize ϕ^+ by minimizing the Mean Squared Error (MSE) loss:

$$\mathcal{L}(\phi^+) = MSE(\hat{s}^i_{t+1}, s^i_{t+1}) = MSE(f_{\phi^+}(s^i_t, a^i_t), s^i_{t+1}) \tag{6}$$

where s^i_t, a^i_t, s^i_{t+1} are sampled from the trajectories collected from E_{real}.

• *The inverse model* $h_{\phi^-}(\hat{s}_{t+1}, s_t)$ in traffic signal control predicts the grounded action $\hat{a}^i_t$ in simulation E_{sim} to reproduce the same traffic states $\hat{s}_{t+1}$. We approximate h_{ϕ^-} with a deep neural network and optimize ϕ^- by minimizing the Categorical Cross-Entropy (CCE) loss since the target a^i_t is a discrete value:

$$\mathcal{L}(\phi^-) = CCE(\hat{a}^i_t, a^i_t) = CCE(h_{\phi^-}(s^i_{t+1}, s^i_t), a^i_t) \tag{7}$$

where s^i_t, a^i_t, s^i_{t+1} are sampled from the trajectories collected from E_{sim}.

3.2 Uncertainty-aware GAT

In this section, we will introduce the limitations of the vanilla GAT and propose an uncertainty-aware method on GAT that can benefit from quantifying model uncertainty.

3.2.1 *Model Uncertainty on* g_ϕ.

The vanilla GAT takes supervised learning to train the action transformation function g_ϕ, and grounded action transformation $\hat{a}$ is taken at each step while improving in the E_{sim}. However, the action transformation function g_ϕ could have high model uncertainty on unseen state and action inputs, which is likely to happen during the exploration of RL. With high model uncertainty on g_ϕ, the grounded action $\hat{a}$ in Equation (3) is likely to enlarge the dynamics gap instead of mitigating it. The enlarged gap will make policy learning unstable and hard to converge.

To overcome the enlarged gap induced by $\hat{a}$ with high model uncertainty in g_ϕ, we need uncertainty quantification methods [5] to keep track of the uncertainty of g_ϕ. Specifically, we would like the action transformation function to output an uncertainty value u_t in addition to $\hat{a}_t$:

$$\hat{a}_t, u_t = g_\phi(s_t, a_t) = h_{\phi^-}(f_{\phi^+}(s_t, a_t), s_t) \tag{8}$$

In general, any methods capable of quantifying the uncertainty of a predicted class from a deep neural network (since h_{ϕ^-} is implemented with deep neural networks) could be utilized, like evidential deep learning (EDL), Concrete Dropout [2], Deep Ensembles [6], etc. In this paper, we explored different state-of-the-art uncertainty quantification methods and found out that they all perform well with our method (their experimental results can be found in Section 4.2.4). We adopted EDL as the default in our method as it performs the best with our method.

Intuitively, during action grounding, whenever model $g_\phi(s_t, a_t)$ returns a grounded action $\hat{a}_t$, if the uncertainty u_t is less than the threshold α, the grounded action $\hat{a}_t$ will be taken in the simulation environment E_{sim} for policy improvement; otherwise, we will reject $\hat{a}_t$ and take the original a_t. This uncertainty quantification allows us to evaluate the reliability of the transformation model and take grounded actions $\hat{a}$ when the model is certain that the resulting transition $P_\phi(s_t, \hat{a}_t)$ would mirror that of the real-world environment E_{real} transition $P^*(s_t, a_t)$. This process enables us to minimize the gap in Equation (3) between the policy training environment E_{sim} and the policy testing environment E_{real}, thereby mitigating the performance gap.

3.2.2 *Dynamic Grounding Rate* α.

The threshold α, which we referred to as the grounding rate, helps us to decide when to filter out $\hat{a}_t$ with uncertainty u_t. One naive approach of deciding the grounding rate α is to treat it as a hyperparameter for training and keep it fixed during the training process. However, since $g_\phi(s_t, a_t)$ keeps being updated during the training process, the model uncertainty of g_ϕ is dynamically changing. Even with the same s_t and a_t, the output u_t and $\hat{a}_t$ from $g_\phi(s_t, a_t)$ could be different in different training iterations.

An alternative yet feasible approach is to set grounding rate α dynamically changing with the model uncertainty during different training iterations. To dynamically adjust the grounding rate with the changing of model uncertainty, we keep track of the model uncertainty u_t of $g_\phi(s_t, a_t)$ during each training iteration. At the end of each iteration i, we update the grounding rate α for the next iteration based on the past record of model uncertainty by calculating the mean:

$$\alpha = \frac{\sum_{e=1}^{E} \sum_{t=0}^{T-1} u^e_t}{T \times E} \tag{9}$$

from the logged uncertainties in the last E epochs. This dynamic grounding rate α can synchronously adjust α with the update of g_ϕ and relief efforts on hyper-parameter tuning.

3.3 Training Algorithm

The overall algorithm for UGAT is shown in Algorithm 1. We firstly pre-train the RL policy π_θ for M epochs in the simulation environment E_{sim}. Then each training iteration of UGAT starts with collecting datasets for E_{sim} and E_{real} following the data collection process in [3]. With the collected data, g_ϕ will be updated by training the forward model f_{ϕ^+} and inverse model h_{ϕ^-}. With the updated g_ϕ, we start to use the policy π_θ to interact with E_{sim} for policy training. Before the action a_t outputted by $\pi_\theta(s_t)$ is taken into the environment E_{sim}, UGAT grounds the actions through $\hat{a}_t$ and u_t from $g_\phi(s_t, a_t)$. If the model uncertainty u_t is greater than the grounding rate α, the grounded action $\hat{a}_t$ is rejected and we execute origin action a_t in the simulation E_{sim}. Then u_t is added into logged uncertainty U. The RL policy π_θ updates during the interaction with E_{sim}. After E rounds of intersections, we update α with Equation (9) for the next round of policy training.

4 EXPERIMENT AND RESULTS

In this section, we conduct experiments to answer the following questions:

•**RQ1:** Does performance gap exist in TSC?

•**RQ2:** Can UGAT effectively mitigate the performance gap?

•**RQ3:** How do dynamic grounding rate α, uncertainty quantification, and action grounding influence the performance of UGAT?

•**RQ4:** Does UGAT method perform stably with various uncertainty quantification methods?

Algorithm 1: UGAT with uncertainty quantification

```
Input: Initial policy $\pi_\theta$, forward model $f_{\phi^+}$, inverse model
       $h_{\phi^-}$, real-world dataset $\mathcal{D}_{real}$, simulation dataset
       $\mathcal{D}_{sim}$, grounding rate $\alpha$ = inf
Output: Policy $\pi_\theta$, $f_{\phi^+}$, $h_{\phi^-}$
1  Pre-train policy $\pi_\theta$ for M iterations in $E_{sim}$
2  for $i$ = 1,2, ..., $I$ do
3      Rollout policy $\pi_\theta$ in $E_{sim}$ and add data to $\mathcal{D}_{sim}$
4      Rollout policy $\pi_\theta$ in $E_{real}$ and add data to $\mathcal{D}_{real}$
5      # Transformation function update step
6      Update $f_{\phi^+}$ with Equation (6)
7      Update $h_{\phi^-}$ with Equation (7)
8      Reset logged uncertainty $U^i = List()$
9      # Policy training
10     for $e$ = 1, 2, ..., $E$ do
11         # Action grounding step
12         for $t$ = 0, 1 ,..., T-1 do
13             $a_t = \pi(s_t)$
14             Calculate $\hat{a}_t$ and $u_t$ with Equation (8)
15             if $u_t^e \geq \alpha$ then
16                 $\hat{a}_t = a_t$ # Reject grounded action
17             end
18             $U.append(u_t^e)$
19         end
20         # Policy update step
21         Improve policy $\pi_\theta$ with reinforcement learning
22     end
23     Update $\alpha$ with Equation (9)
24 end
```

4.1 Experiment Settings

We introduce the environment setup, commonly used metrics, important hyperparameters, and model structures.

In this paper, we implement UGAT upon LibSignal [8], an open-sourced traffic signal control library that integrates multiple simulation environments. We treat Cityflow [14] as the simulation environment E_{sim} and SUMO [7] as the real-world environment E_{real}. In later sections, we use E_{sim} and E_{real} by default unless specified. To mimic real-world settings, we consider four configurations in SUMO under two types of real-world scenarios: heavy industry roads and special weather-conditioned roads, with their specific parameters defined in Table 1.

Table 1: Real-world Configurations for E_{real}

Setting	accel (m/s^2)	decel (m/s^2)	eDecel (m/s^2)	sDelay (s)	Description
Default	2.60	4.50	9.00	0.00	–
V1	1.00	2.50	6.00	0.50	Lighter loaded vehicles
V2	1.00	2.50	6.00	0.75	Heavier loaded vehicles
V3	0.75	3.50	6.00	0.25	Rainy weather
V4	0.50	1.50	2.00	0.50	Snowy weather

4.1.1 *Environment Setup.* • *Default setting* [1]. The default parameters for SUMO and CityFlow describe the normal settings of the vehicle's movement in E_{sim}.
• *Heavy industry roads.* We model the places where the majority of vehicles could be heavy trucks. In Table 1, for the vehicles in $V1$ and $V2$, their accelerating, decelerating, and emergency decelerating rates are more likely to be slower than the default settings.
• *Special weather-conditioned roads.* We consider $V3$ and $V4$ means rainy and snowy conditions respectively. In Table 1, the vehicles' accelerating, decelerating, and emergency decelerating rates are smaller than the default, and the startup delays are larger. For snowy weather, the first three rates are smaller than rainy ones, and the startup delay discrepancy in snowy conditions is extended to mimic the tire slip.

4.1.2 *Evaluation Metrics.* Following the literatures in TSC [12, 15], we adopt commonly used traffic signal control metrics: Average Travel Time (ATT), Throughput (TP), Reward, Queue, Delay. For ATT, Queue, and Delay, the smaller, the better. for others, the larger, the better.

In this work, our goal is to mitigate the performance gap of trained policy π_θ between E_{sim} and E_{real}, so we calculate the gap Δ for each referred metric as ATT_Δ, TP_Δ, $Reward_\Delta$, $Queue_\Delta$, and $Delay_\Delta$. For certain metric ψ, $\psi_\Delta = \psi_{real} - \psi_{sim}$. Because in real-world settings, policy π_θ tends to perform worse than in simulation, so the ATT, $Queue$, and $Delay$ in E_{real} are normally larger than those in E_{sim}. Based on the goal of mitigating the gap, improving π_θ performance in E_{sim}, we expect: for ATT_Δ, $Queue_\Delta$, and $Delay_\Delta$, the smaller the better. Because TP_Δ, $Reward_\Delta$ will be negative values, the larger, the better. The forward and inverse models are implemented with the feed-forward neural network with three hidden layers, which have 64, 128, and 20 hidden neurons and are optimized with Adam optimizer.

4.2 Experiment Results

4.2.1 *Gap between real-world and simulator (RQ1).* To testify whether the performance gap exists in traffic signal control tasks, we use the Direct-Transfer method: train a policy model π_{test} in E_{sim} by the DQN method for 300 epochs, which guarantees its training convergence and directly transfer π_{test} to 4 different E_{real} settings as shown in Table 1.

Table 2: Direct-Transfer and UGAT performance in E_{sim}

Env	*ATT*	*TP*	*Reward*	*Queue*	*Delay*
E_{sim}	**111.23**$_{\pm 0.05}$	**1978**$_{\pm 1}$	**-39.44**$_{\pm 0.03}$	**26.11**$_{\pm 0.05}$	**0.62**$_{\pm 0.01}$

The results in Fig. 2 exist 5 metrics showing the performance on two environments: E_{sim} and E_{real}. The blue line is connected with the 5 metric results in E_{sim} and the orange one represents the results in E_{real}. Compared with the performance in the E_{sim}, a clear metrics gap appears when π_{test} is applied to 4 E_{real} settings. Our experiment justifies the existence of performance gaps, which directs us to further study the method's generalizability and mitigate such problem.

[1]https://sumo.dlr.de/docs/Definition_of_Vehicles,_Vehicle_Types,_and_Routes.html

Table 3: The performance using Direct-Transfer method compared with using UGAT method. The (·) shows the metric gap ψ_Δ from E_{real} to E_{sim} and the ± shows the standard deviation with 3 runs. The ↑ means that the higher value for the metric indicates a better performance and ↓ means that the lower value indicates a better performance

Setting	Direct Transfer					UGAT				
	$ATT(\Delta\downarrow)$	$TP(\Delta\uparrow)$	$Reward(\Delta\uparrow)$	$Queue(\Delta\downarrow)$	$Delay(\Delta\downarrow)$	$ATT(\Delta\downarrow)$	$TP(\Delta\uparrow)$	$Reward(\Delta\uparrow)$	$Queue(\Delta\downarrow)$	$Delay(\Delta\downarrow)$
V1	158.93(47.69)	1901(-77)	-71.55(-32.11)	47.71(21.59)	0.73(0.11)	**144.72(33.49)**$_{\pm 3.61}$	**1925(-52)**$_{\pm 4.58}$	**-59.38(-19.94)**$_{\pm 3.08}$	**39.58(13.47)**$_{\pm 2.04}$	**0.67(0.05)**$_{\pm 0.01}$
V2	177.27(66.03)	1898(-80)	-87.71(-48.27)	58.59(32.47)	0.76(0.14)	**164.65(53.52)**$_{\pm 12.94}$	**1907(-71)**$_{\pm 13.06}$	**-75.18(-35.74**$_{\pm 8.37}$	**50.25(24.14)**$_{\pm 5.56}$	**0.72(0.10)**$_{\pm 0.01}$
V3	205.86(94.63)	1877(-101)	-101.26(-61.82)	67.62(41.51)	0.76(0.14)	**183.22(71.99)**$_{\pm 13.22}$	**1900(-78)**$_{\pm 13.08}$	**-82.38(-42.94)**$_{\pm 9.11}$	**55.05(28.94)**$_{\pm 6.08}$	**0.72(0.10)**$_{\pm 0.01}$
V4	332.48(221.25)	1735(-252)	-126.71(-87.23)	84.53(58.42)	0.83(0.21)	**284.26(173.03)**$_{\pm 6.67}$	**1794(-184)**$_{\pm 12.05}$	**-111.68(-72.24)**$_{\pm 7.25}$	**74.54(48.43)**$_{\pm 4.82}$	**0.8(0.18)**$_{\pm 0.01}$

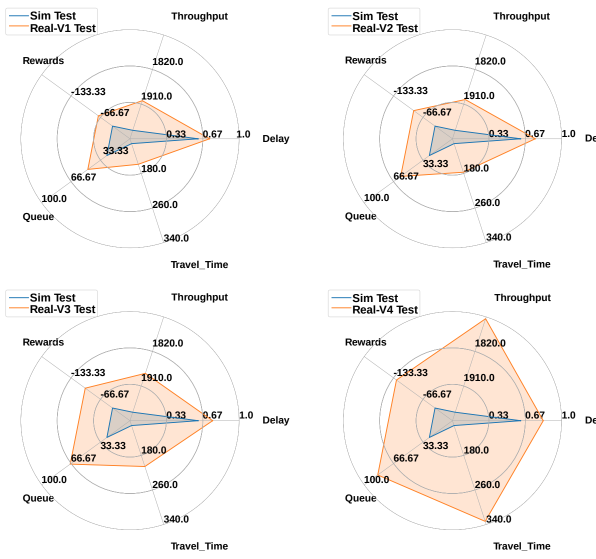


Figure 2: The performance gap using Direct-Transfer to train in E_{sim} and tested in 4 E_{real} settings.

4.2.2 *Gap mitigating by uncertainty-aware UGAT (RQ2).*

To verify whether the UGAT can effectively mitigate the performance gap, we compare the performance of directly transferring policies trained in E_{sim} to E_{real} with the policies learned under UGAT in four E_{real} settings. Because they are using the same E_{sim}, so performance in E_{sim} eventually converges to stable results with tiny variance as shown in Table 2. The (·) in Table 3 describes the performance gap from E_{real} to E_{sim} using the results above and calculated as $\psi_\Delta = \psi_{real} - \psi_{sim}$. The gap directly reflects the methods' ability to generalize from E_{sim} to E_{real}. We have the following observations:

• Compared with the Direct-Transfer method, applying UGAT mitigates the performance gap Δ: ATT_Δ, $Queue_\Delta$, and $Delay_\Delta$ in UGAT are smaller than in Direct-Transfer column, and TP_Δ, $Reward_\Delta$ are the larger.

• On the original metrics of traffic signal control, UGAT could improve the performance of policy π_ϕ performance as well. When comparing the two methods, UGAT could reach much lower ATT and higher TP than Direct-Transfer.

• In Table 3, the experiments include 5 different metrics across 4 various real-world settings. Based on the reported results, we can conclude that UGAT is both robust and effective.

Table 4: Ablation Study of UGAT on $V1$

Structure	ATT_Δ $(\Delta\downarrow)$	TP_Δ $(\Delta\uparrow)$	$Reward_\Delta$ $(\Delta\uparrow)$	$Queue_\Delta$ $(\Delta\downarrow)$	$Delay_\Delta$ $(\Delta\downarrow)$
UGAT	**33.49**$_{\pm 3.61}$	**-52**$_{\pm 4.58}$	**-19.94**$_{\pm 3.08}$	**13.47**$_{\pm 2.04}$	**0.05**$_{\pm 0.01}$
w/o dynamic α	39.12$_{\pm 4.21}$	-72$_{\pm 7.61}$	-25.07$_{\pm 5.71}$	16.88$_{\pm 5.11}$	0.08$_{\pm 0.01}$
w/o α, uncertainty	44.87$_{\pm 4.81}$	-73$_{\pm 12.99}$	-30.59$_{\pm 3.80}$	20.50$_{\pm 1.97}$	0.09$_{\pm 0.01}$
w/o Grounding	47.71$_{\pm 6.73}$	-77$_{\pm 10.64}$	-32.11$_{\pm 4.24}$	21.60$_{\pm 3.12}$	0.11$_{\pm 0.02}$

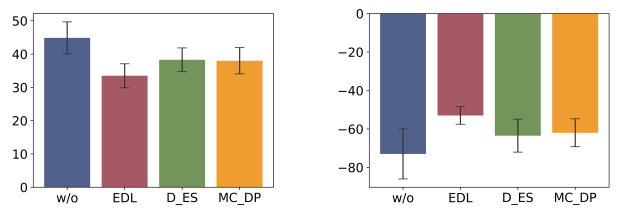
4.2.3 *Ablation Study (RQ3).*

To understand how the dynamic grounding module, uncertainty quantification module, and action grounding module influence the method's performance, we conduct an ablation study. Note that when the dynamic grounding module is removed, the grounding rate α is set as a static value of 0.5. The results in Table 4 show that each module contributes to the UGAT performance.

Table 5: Static vs dynamic α on $V1$

α	ATT_Δ $(\Delta\downarrow)$	TP_Δ $(\Delta\uparrow)$	$Reward_\Delta$ $(\Delta\uparrow)$	$Queue_\Delta$ $(\Delta\downarrow)$	$Delay_\Delta$ $(\Delta\downarrow)$
dynamic	**33.49**$_{\pm 3.61}$	**-52**$_{\pm 4.58}$	**-19.94**$_{\pm 3.08}$	**13.47**$_{\pm 2.04}$	**0.05**$_{\pm 0.01}$
0.2	68.59$_{\pm 7.14}$	-117$_{\pm 12.53}$	-40.42$_{\pm 3.92}$	27.11$_{\pm 4.29}$	0.12$_{\pm 0.05}$
0.4	55.87$_{\pm 7.83}$	-73$_{\pm 13.01}$	-30.69$_{\pm 4.54}$	20.30$_{\pm 3.28}$	0.12$_{\pm 0.01}$
0.5	39.12$_{\pm 4.21}$	-72$_{\pm 7.61}$	-25.07$_{\pm 5.71}$	16.88$_{\pm 5.11}$	0.08$_{\pm 0.01}$
0.6	47.09$_{\pm 2.79}$	-77$_{\pm 4.68}$	-34.11$_{\pm 3.99}$	21.31$_{\pm 2.38}$	0.10$_{\pm 0.03}$
0.8	48.53$_{\pm 6.70}$	-85$_{\pm 9.17}$	-37.85$_{\pm 6.23}$	25.60$_{\pm 2.91}$	0.11$_{\pm 0.01}$

To understand how the dynamically adjusted grounding rate α influences the sim-to-real training, an in-depth ablation study on α is further conducted and shown in Table 5. We enable the uncertainty quantification module EDL, and then manually set the α from 0.2 to 0.8. During the training, the model will output uncertainty u, but the threshold α is static, if $u < \alpha$ then conduct action grounding, otherwise, reject the action as described in Algorithm 1. We compare the results with UGAT which leverages uncertainty to dynamically adjust the α. It is explicit that UGAT method guarantees a good improvement in the model's performance.



(a) ATT_Δ of 4 methods   (b) TP_Δ of 4 methods

Figure 3: Uncertainty investigation across 4 methods on $V1$

4.2.4 *Different Uncertainty Methods in UGAT (RQ4).*

In previous experiments, we use the uncertainty quantification method

EDL [10]. To better understand the benefit of model uncertainty, we conduct experiments with other uncertainty quantification methods including EDL, Concrete Dropout [2], Deep Ensembles [6] to compare them with the method removed uncertainty module. In Fig. 3, w/o is the version without uncertainty methods, and the other three are EDL, Deep Ensembles (D_ES), and MC Dropout (MC_DP), the ATT_Δ is calculated as $\psi_\Delta = \psi_{real} - \psi_{sim}$, the smaller, the better, and for TP_Δ, the larger the better. The results show that the explored model uncertainty quantification methods consistently reduces the performance gap, and the EDL performs best.

5 CONCLUSION

In this paper, we present the existence of the performance gap in traffic signal control problems and propose an uncertainty-aware grounding action transformation method (UGAT) that can dynamically transform actions in the simulation with uncertainty to mitigate the discrepancy between simulated and real-world dynamics. The experiments demonstrate that UGAT has an excellent performance in bridging the performance gap with higher efficiency and stability. This research is a step towards improving the real-world applicability of RL-based TSC models.

REFERENCES

[1] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. 2020. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3414–3421.

[2] Yarin Gal, Jiri Hron, and Alex Kendall. 2017. Concrete dropout. *Advances in neural information processing systems* 30 (2017).

[3] Josiah Hanna and Peter Stone. 2017. Grounded action transformation for robot learning in simulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[4] Yifeng Jiang, Tingnan Zhang, Daniel Ho, Yunfei Bai, C Karen Liu, Sergey Levine, and Jie Tan. 2021. Simgan: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2884–2890.

[5] HM Dipu Kabir, Abbas Khosravi, Mohammad Anwar Hosen, and Saeid Nahavandi. 2018. Neural network-based uncertainty quantification: A survey of methodologies and applications. *IEEE access* 6 (2018), 36218–36234.

[6] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems* 30 (2017).

[7] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic traffic simulation using sumo. In *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE, 2575–2582.

[8] Hao Mei, Xiaoliang Lei, Longchao Da, Bin Shi, and Hua Wei. 2022. LibSignal: An Open Library for Traffic Signal Control. *arXiv preprint arXiv:2211.10649* (2022).

[9] Mohammad Noaeen, Atharva Naik, Liana Goodman, Jared Crebo, Taimoor Abrar, Zahra Shakeri Hossein Abad, Ana LC Bazzan, and Behrouz Far. 2022. Reinforcement learning in urban network traffic signal control: A systematic literature review. *Expert Systems with Applications* (2022), 116830.

[10] Murat Sensoy, Lance Kaplan, and Melih Kandemir. 2018. Evidential deep learning to quantify classification uncertainty. *Advances in neural information processing systems* 31 (2018).

[11] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. 2019. Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management.* 1913–1922.

[12] Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. 2021. Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *ACM SIGKDD Explorations Newsletter* 22, 2 (2021), 12–18.

[13] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. 2018. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2496–2505.

[14] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. 2019. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The world wide web conference.* 3620–3624.

[15] Dongbin Zhao, Yujie Dai, and Zhen Zhang. 2011. Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42, 4 (2011), 485–494.

[16] Guanjie Zheng, Yuanhao Xiong, Xinshi Zang, Jie Feng, Hua Wei, Huichu Zhang, Yong Li, Kai Xu, and Zhenhui Li. 2019. Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM international conference on information and knowledge management.* 1963–1972.