# Joint-Local Grounded Action Transformation for Sim-to-Real Transfer in Multi-Agent Traffic Control

**Justin Turnau, Longchao Da, Khoa Vo, Ferdous Al Rafi, Shreyas Bachiraju, Tiejin Chen, Hua Wei**

**Keywords:** Traffic Signal Control, Multi-Agent Reinforcement Learning, Sim-to-Real Transfer

## Summary

Traffic Signal Control (TSC) is essential for managing urban traffic flow and reducing congestion. Reinforcement Learning (RL) offers an adaptive method for TSC by responding to dynamic traffic patterns, with multi-agent RL (MARL) gaining traction as intersections naturally function as coordinated agents. However, due to shifts in environmental dynamics, implementing MARL-based TSC policies in the real world often leads to a significant performance drop, known as the sim-to-real gap. Grounded Action Transformation (GAT) has successfully mitigated this gap in single-agent RL for TSC, but real-world traffic networks, which involve numerous interacting intersections, are better suited to a MARL framework. In this work, we introduce JL-GAT, an application of GAT to MARL-based TSC that balances scalability with enhanced grounding capability by incorporating information from neighboring agents. JL-GAT adopts a decentralized approach to GAT, allowing for the scalability often required in real-world traffic networks while still capturing key interactions between agents. Comprehensive experiments on various road networks and ablation studies demonstrate the effectiveness of JL-GAT.

## Contribution(s)

1. We introduce Joint-Local Grounded Action Transformation (JL-GAT), a scalable framework for bridging the sim-to-real gap in MARL-based traffic signal control that incorporates state and action information from neighboring agents into Grounded Action Transformation (GAT) models using a sensing radius.
   **Context:** None
2. To the best of our knowledge, we are the first to apply Grounded Action Transformation (GAT) to the multi-agent setting, introducing two natural applications of GAT alongside our proposed method, JL-GAT.
   **Context:** None
3. We introduce the cascading invalidation effect, a novel challenge in JL-GAT that arises when integrating state and action information from nearby agents, and propose both a direct solution and an alternative approach that effectively mitigates the issue.
   **Context:** None
4. We conduct thorough empirical evaluations of JL-GAT in the domain of multi-agent traffic signal control, demonstrating its effectiveness in reducing the sim-to-real gap.
   **Context:** None

# Joint-Local Grounded Action Transformation for Sim-to-Real Transfer in Multi-Agent Traffic Control

**Justin Turnau, Longchao Da, Khoa Vo, Ferdous Al Rafi, Shreyas Bachiraju, Tiejin Chen, Hua Wei**

{jturnau,longchao,ngocbach,sbachira,tchen169,hua.wei}@asu.edu, rafirafi155@gmail.com

**Arizona State University**

## Abstract

Traffic Signal Control (TSC) is essential for managing urban traffic flow and reducing congestion. Reinforcement Learning (RL) offers an adaptive method for TSC by responding to dynamic traffic patterns, with multi-agent RL (MARL) gaining traction as intersections naturally function as coordinated agents. However, due to shifts in environmental dynamics, implementing MARL-based TSC policies in the real world often leads to a significant performance drop, known as the sim-to-real gap. Grounded Action Transformation (GAT) has successfully mitigated this gap in single-agent RL for TSC, but real-world traffic networks, which involve numerous interacting intersections, are better suited to a MARL framework. In this work, we introduce JL-GAT, an application of GAT to MARL-based TSC that balances scalability with enhanced grounding capability by incorporating information from neighboring agents. JL-GAT adopts a decentralized approach to GAT, allowing for the scalability often required in real-world traffic networks while still capturing key interactions between agents. Comprehensive experiments on various road networks and ablation studies demonstrate the effectiveness of JL-GAT. The code is publicly available at https://github.com/DaRL-LibSignal/JL-GAT/.

## 1 Introduction

Reinforcement Learning (RL) is well-suited for sequential decision-making, enabling agents to learn effective policies through interaction with the environment (Roijers et al., 2013b). This data-driven design, together with the ability to adaptively refine policies, makes RL a powerful approach to complex real-world problems. Traffic Signal Control (TSC) is an effective way to reduce congestion, minimize travel times, and improve urban mobility (Wei et al., 2018). By modeling TSC as a sequential decision-making problem, where each traffic signal chooses timing and phases based on evolving traffic conditions, RL can deliver flexible, efficient control strategies. Thus, RL-driven TSC appears as a dynamic and robust alternative to static or rule-based methods in transportation research (Wei et al., 2019b).

In addition to treating an intersection-coupled traffic signal as a single agent, multi-agent reinforcement learning (MARL) is essential for scaling up traffic signal control to complex urban networks (Jiang et al., 2024). By deploying a network of agents, each controlling an individual intersection, MARL facilitates decentralized decision-making while maintaining coordinated across the entire system (Chen et al., 2020). It allows each agent to learn local policies that are responsive to immediate traffic conditions yet also adapt through communication and cooperation with neighboring agents to optimize overall traffic flow, which is more suitable for managing large-scale, dynamic transportation environments such as those found in real-world applications (Balmer et al., 2004).

In order to learn the traffic signal control policies, a direct way is to leverage the existing traffic simulators (e.g., SUMO (Behrisch et al., 2011), CityFlow (Zhang et al., 2019; Da et al., 2024a)) as an interactive environment and explore control policies. While simulators offer a controlled environment to train and evaluate RL-based TSC policies, transitioning these models from simulation to the real world introduces a challenging gap known as the sim-to-real issue (Da et al., 2023a). Discrepancies between the simulated and real environments, such as unmodeled traffic dynamics (Da et al., 2023b), sensor noise (Qadri et al., 2020), and unpredictable driver behaviors (Lee & Moura, 2015), can lead to significant deviations in performance. Therefore, robust sim-to-real techniques are essential to bridge this gap and ensure the performance observed in simulation translates to real-world urban settings.

The preliminary research from (Da et al., 2023a) has identified the severity of the sim-to-real issue in RL-based TSC. There are several proposed solutions to mitigate the sim-to-real gap, either by calibrating the simulator's realism (Müller et al., 2021) or by using transfer learning in the RL training paradigm, such as grounded action transformation (GAT) (Da et al., 2024b). JL-GAT enhances GAT by integrating neighboring agents' information to capture local interactions, improving transition dynamics modeling. This strengthens policy training, boosts real-world performance, and minimizes the sim-to-real gap, ultimately enhancing urban mobility and reducing congestion.

## 2 Related Work

### 2.1 Reinforcement Learning for Multi-Agent Traffic Signal Control

Reinforcement Learning for multi-agent traffic signal control has emerged as a promising approach to alleviate urban traffic congestion by enabling intersections to operate as cooperative agents (Choy et al., 2003). Under this framework, each traffic signal controller is treated as an agent that learns optimal control policies through local interactions with the environment and limited communication with neighboring intersections (Balaji & Srinivasan, 2010). Unlike traditional rule-based methods that rely on pre-defined heuristics (Dion & Hellinga, 2002), RL-based approaches dynamically adapt to real-time traffic conditions, yielding significant improvements in vehicle travel time and delay reduction (Zheng et al., 2019). Multi-agent reinforcement learning (MARL) introduces both additional complexities and opportunities compared to single-agent settings (Roijers et al., 2013a). Coordination among multiple agents can enhance overall network performance by balancing local decisions with global objectives, yet challenges such as environmental non-stationarity and the need for scalable communication strategies persist (Chen et al., 2020). Recent advances in MARL have explored solutions like centralized training with decentralized execution and cooperative learning schemes to overcome these challenges (Huang et al., 2021). Moreover, while many existing RL-based TSC methods focus on optimizing performance within simulated environments (Mei et al., 2024), the sim-to-real gap remains a critical hurdle (Da et al., 2023a). Some recent studies have attempted to narrow this gap but only focus on the single-agent settings (Da et al., 2023b; 2024b), whereas our approach applies the work to more complex multi-agent settings, which hold great potential for more scalable traffic signal control systems capable of effectively responding to dynamic traffic patterns.

### 2.2 Sim-to-Real Methods for RL

Sim-to-real transfer methods in RL broadly fall into three main categories (Zhao et al., 2020; Da et al., 2025). The first category, ***domain randomization*** (Tobin, 2019; Andrychowicz et al., 2020; Wei et al., 2022), focuses on training policies that are robust to environmental variations by relying heavily on simulated data, which is particularly advantageous when facing uncertain or evolving target domains. The second category, ***domain adaptation*** (Tzeng et al., 2019; Han et al., 2019), addresses the challenge of distribution shifts between the source and target environments by aligning feature representations. Although many techniques in this category are aimed at bridging gaps in robotic perception (Tzeng et al., 2015; Fang et al., 2018; Bousmalis et al., 2018; James et al.,

2019), in the traffic signal control domain the discrepancy is mainly due to differences in dynamics, since most methods use vectorized observations such as lane-level vehicle counts or delays. The third category involves ***grounding methods***, which aim to reduce simulator bias and improve alignment with real-world dynamics. In contrast to system identification approaches (Cutler et al., 2014; Cully et al., 2015) that seek to learn exact physical parameters, Grounded Action Transformation (GAT) (Hanna & Stone, 2017) modifies the simulator dynamics via grounded actions, showing promising results for sim-to-real transfer in robotics (Zhang et al., 2025). Recent work (Desai et al., 2020b; Karnan et al., 2020; Desai et al., 2020a) has further advanced grounding methods by incorporating stochastic modeling, reinforcement learning, and imitation-from-observation techniques. Our approach, JL-GAT, builds on the GAT framework, introducing novel multi-agent designs and proposing local-joint solutions.

## 3 Preliminaries

This section introduces the necessary background for understanding our proposed method, including the formulation of the multi-agent reinforcement learning (MARL) traffic signal control (TSC) problem and an overview of Grounded Action Transformation (GAT) [1].

### 3.1 Multi-Agent Traffic Signal Control

We formulate TSC as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), where each intersection acts as an agent observing partial traffic states and optimizing local control policies to maximize cumulative reward. See Section 8 in the Supplementary Materials for full notation.

### 3.2 Agent Design

For our agent design, we align with the most prevalent works in the TSC domain, such as PressLight (Wei et al., 2019a), with slight modifications, and use it consistently across all experiments. We summarize the state representation, action space, reward function, and learning method for our agents in Section 9 of the Supplementary Materials.

### 3.3 Grounded Action Transformation

Grounded Action Transformation (GAT) is a framework designed to align simulated environments with real-world dynamics using real trajectories $\mathcal{D}_{\text{real}} = \{\tau^1, \ldots, \tau^I\}$ collected by executing a policy $\pi_\theta$ in the real environment $E_{\text{real}}$. Let $P^*$ denote the real-world transition dynamics and $P_\phi$ denote the parameterized transition function of the simulator $E_{\text{sim}}$. GAT optimizes $\phi$ to minimize the discrepancy between $P^*$ and $P_\phi$:

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

where $d(\cdot)$ is a distance measure (e.g., Kullback-Leibler divergence).

Given a policy $\pi_\theta$ that outputs an action $a_t$ to take in a given state $s_t$, GAT employs an action transformation function $g_\phi(s_t, a_t)$ parameterized by $\phi$ to compute a grounded action $\hat{a}_t$:

$$\hat{a}_t = g_\phi(s_t, a_t) = h_{\phi^-}\left(s_t, f_{\phi^+}(s_t, a_t)\right). \tag{2}$$

The vanilla GAT framework consists of two models: a forward model $f_{\phi^+}$ and an inverse model $h_{\phi^-}$. The **forward model** takes as input the current state $s_t$ and the action $a_t$ from $E_{\text{sim}}$ and predicts

[1] The detailed notation summary is shown in Table 6.

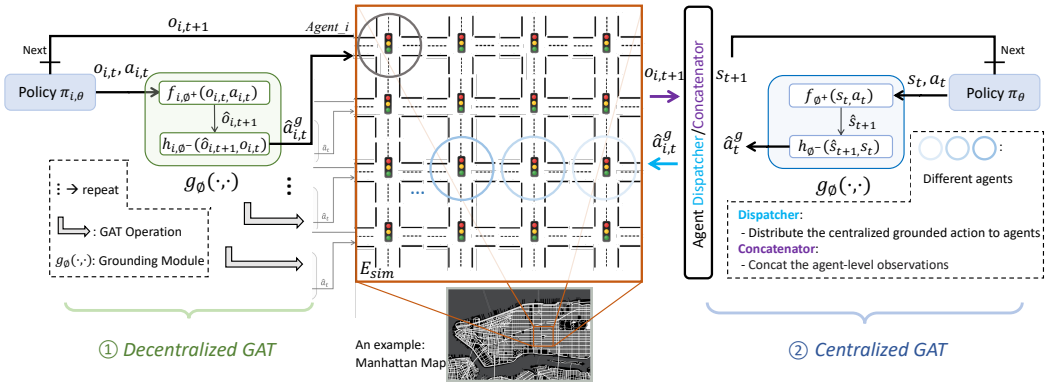


Figure 1: Overview of centralized and decentralized GAT in a 4×4 traffic network. **Left**: In decentralized GAT, each agent $i$ observes its state $o_{i,t}$ and selects an action $a_{i,t}$ via policy $\pi_{i,\theta}$. This action is passed through a forward model $f_{i,\phi^+}$ to predict the next observation $\hat{o}_{i,t+1}$, which is then input to the inverse model $h_{i,\phi^-}$ to produce the grounded action $\hat{a}^g_{i,t}$. This process runs independently for all agents. **Right**: Centralized GAT follows the same logic, but observations $o_{i,t}$ are concatenated into a global state $s_t$, and input to a shared forward model $f_{\phi^+}$ to produce $\hat{s}_{t+1}$ (a composition of all $\hat{o}_{i,t+1}$). This is then passed to a centralized inverse model $h_{\phi^-}$ to yield the global grounded actions $\hat{a}^g_t$, which are dispatched to agents to replace their original actions.

the next state $\hat{s}_{t+1}$ in $E_{\text{real}}$: $\hat{s}_{t+1} = f_{\phi^+}(s_t, a_t)$. The **inverse model**, in turn, receives the current state $s_t$ from $E_{\text{sim}}$ and the predicted next state $\hat{s}_{t+1}$ from the forward model, generating a grounded action $\hat{a}_t$ that attempts to transition $s_t$ to $\hat{s}_{t+1}$ in $E_{\text{sim}}$: $\hat{a}_t = h_{\phi^-}(s_t, \hat{s}_{t+1})$. With effective grounding, the simulator's transition dynamics, $P_\phi$, better approximate those of the real environment, $P^*$. This alignment facilitates more effective policy training in $E_{\text{sim}}$, as GAT reduces the discrepancy in transition dynamics, leading to more realistic state transitions and ultimately reducing the sim-to-real gap. Note that the forward model $f_{\phi^+}$ is trained using data collected in $E_{\text{real}}$ and the inverse model $h_{\phi^-}$ is trained using data collected in $E_{\text{sim}}$.

## 4 Grounded Action Transformation in Multi-Agent Settings

Grounded Action Transformation (GAT) bridges the sim-to-real gap using forward and inverse models to align simulator and real-world dynamics. In multi-agent settings, this alignment is challenged by inter-agent interactions. As shown in Figure 1, GAT can be centralized, capturing global dynamics at the cost of scalability, or decentralized, scaling well but ignoring multi-agent interactions. This section presents both approaches as a foundation for our proposed hybrid method, Joint-Local GAT (JL-GAT) in Section 5.

### 4.1 Centralized Grounded Action Transformation

A natural approach to multi-agent GAT is to treat the environment as a single-agent system by using global state and action inputs to a shared forward and inverse model. We provide an overview of centralized GAT in Figure 1. This enables the modeling of inter-agent dynamics but increases learning complexity as the number of agents grows. We retain the GAT objective from Equation (1), modifying it to use global states and actions. Following (Da et al., 2024b), we model $f_{\phi^+}$ and $h_{\phi^-}$ as neural networks trained via MSE and CCE losses. Unlike vanilla GAT, our inputs and outputs are global state-action tuples $s_t, a_t$, composed of all agent observations $o_{i,t}$ and actions $a_{i,t}$.

- The *centralized forward model*, applied to traffic signal control, aims to predict the next global traffic state $\hat{s}_{t+1}$ in the real environment $E_{\text{real}}$ after agents take global actions $a_t$ in the global traffic state $s_t$.

- The *centralized inverse model*, applied to traffic signal control, considers the global traffic state $s_t$ in $E_{\text{sim}}$ and predicted global next traffic state $\hat{s}_{t+1}$ in $E_{\text{real}}$ from the forward model to predict global grounded actions $\hat{a}^{\text{g}}_t$. Note the inputs to the inverse model $h_{\phi^-}$ are global states and actions, but we compute CCE Loss to optimize $\phi^-$ by extracting the individual grounded actions $\hat{a}^{\text{g}}_{i,t}$ from the global grounded actions $\hat{a}^{\text{g}}_t$ and averaging across all agents for each sample.

### 4.2 Decentralized Grounded Action Transformation

A second intuitive approach to applying GAT to multi-agent settings is to assign each agent its own forward and inverse model. In this decentralized framework, each agent's GAT models operate independently, utilizing only their own information as if they were in a single-agent setting. We provide an overview of decentralized GAT in Figure 1. This improves scalability, allowing models to focus on local dynamics per agent. However, they ignore the influence of other agents, limiting their ability to model global dynamics. We follow (Da et al., 2024b), modifying inputs to use local observations in line with the Dec-POMDP formulation described in Section 3.1. Each agent $i$ learns its own $f_{i,\phi^+}$ and $h_{i,\phi^-}$ to model a local grounded transition function $P_\phi$, still optimizing Equation (1) to minimize the discrepancy between $P^*$ and $P_\phi$.

- The *decentralized forward model*, applied to traffic signal control, aims to predict the next state (observation) $\hat{o}_{t+1}$ of traffic in the real environment $E_{\text{real}}$ for each agent $i$ after the action $a_{i,t}$ is taken in the current traffic observation $o_{i,t}$.
- The *decentralized inverse model*, applied to traffic signal control, considers the traffic observation $o_{i,t}$ in $E_{\text{sim}}$ and the predicted next observation $\hat{o}_{i,t+1}$ in $E_{\text{real}}$ from the forward model to predict the grounded action $\hat{a}^{\text{g}}_{i,t}$ for each agent $i$.

## 5 JL-GAT: Joint-Local Grounded Action Transformation

By modifying our decentralized GAT formulation in Section 4.2 to incorporate local joint state and action information for each agent, we arrive at JL-GAT as shown in Figure 2. JL-GAT strikes a balance between the two multi-agent applications of GAT, centralized and decentralized, introduced in Section 4. With this hybrid approach, JL-GAT reaps unique benefits from both approaches, allowing GAT to be applied in large-scale multi-agent settings while still capturing essential agent interactions that influence the transition dynamics of the environment.

### 5.1 Overview of JL-GAT

We introduced two natural ways to apply GAT to multi-agent environments in Section 4: a centralized approach, which uses a single forward and inverse model to capture global information, and a decentralized approach, where each agent has its own GAT model, considering only its own state and actions. Although centralized GAT captures global interactions, it struggles to scale as the agent count grows. In contrast, decentralized GAT simplifies learning but ignores inter-agent dynamics that are critical to transition modeling. To overcome these limitations, we propose JL-GAT, visualized in Figure 2. The core idea behind JL-GAT is simple yet powerful: combine the strengths of both approaches by considering multi-agent interactions, such as in centralized GAT, while retaining the scalability of the decentralized approach. JL-GAT achieves this by incorporating state and action information from neighboring agents into decentralized GAT models, preserving local agent interactions while maintaining the scalability of a decentralized setup. This results in more realistic simulated transitions, narrowing the sim-to-real gap.

### 5.2 Formulation of JL-GAT

In this section, we formally define our proposed method, JL-GAT. We first continue with the decentralized GAT approach described in Section 4.2, which includes a single forward and inverse model

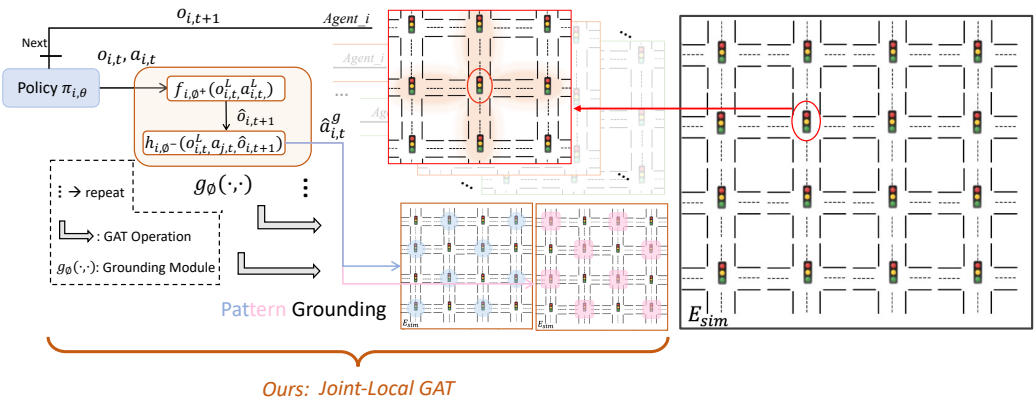


Figure 2: Overview of JL-GAT. The pipeline follows these steps: Each agent $i$ first observes its state $o_{i,t}$ and selects an action $a_{i,t}$ using its policy $\pi_{i,\theta}$. The agent then incorporates neighboring agent observations and actions $o_{j,t}$, $a_{j,t}$ within a predefined sensing radius $r$, considering those within a Manhattan distance of $r$ or less. The 3×3 grid in the top center illustrates the neighboring information used for grounding when $r = 1$. The forward model $f_{i,\phi^+}$ takes in $o_{i,t}$, $a_{i,t}$ and neighboring $o_{j,t}$, $a_{j,t}$, forming the local joint observation $o^L_{i,t}$ and local joint action $a^L_{i,t}$. The forward model $f_{i,\phi^+}$ then predicts the next observation $\hat{o}_{i,t+1}$ for agent $i$. This predicted observation, along with the local joint observation $o^L_{i,t}$ and assumed neighboring actions $a_{j,t}$, is fed into the inverse model $h_{i,\phi^-}$. The inverse model $h_{i,\phi^-}$ outputs a grounded action $\hat{a}^g_{i,t}$ for agent $i$ to take instead of $a_{i,t}$. Finally, we address the cascading invalidation effect, a novel challenge arising with JL-GAT, by introducing pattern grounding, illustrated in the bottom center.

for each agent, extending it to reach the formulation of JL-GAT. Then, we introduce the new objective for JL-GAT. Lastly, we outline the forward and inverse model setup used in JL-GAT, discussing the intuition behind the modifications and their benefits.

### 5.2.1 JL-GAT from Decentralized GAT

We build on the decentralized GAT formulation introduced in Section 4.2, where for each agent $i$, we incorporate neighboring state and action information. We define the local joint state $o^L_{i,t}$ and action $a^L_{i,t}$ of agent $i$ as its own observation $o_{i,t}$ and action $a_{i,t}$ at time $t$ combined with the observation and action information $o_{j,t}$, $a_{j,t}$ of agents $j$ within a predefined sensing radius $r$:

$$o^L_{i,t} = \{o_{i,t}\} \cup \{o_{j,t} \mid d(i,j) \leq r\}, \quad a^L_{i,t} = \{a_{i,t}\} \cup \{a_{j,t} \mid d(i,j) \leq r\}$$

where the Manhattan distance between agents $i$ and $j$ is defined as: $d(i,j) = |x_i - x_j| + |y_i - y_j|$, with $x_i, y_i$ and $x_j, y_j$ representing the positions of agents $i$ and $j$ in a 2D coordinate space.

### 5.2.2 Objective Function for JL-GAT

The formulation of JL-GAT requires modifications to the objective in decentralized GAT shown in Equation (3). Given real-world trajectories $\mathcal{D}_{\text{real}} = \{\tau^1, \ldots, \tau^I\}$, where each trajectory $\tau^k = (s^k_t, a^k_t, s^k_{t+1})_{t=0}^{T-1}$ is collected by executing policies in the real environment $E_{\text{real}}$, our new objective is to learn a grounded simulator transition function $P_{i,\phi}$ for each agent $i$ that minimizes:

$$\phi^* = \arg\min_{\phi} \sum_{\tau^k \in \mathcal{D}_{\text{real}}} \sum_{t=0}^{T-1} d\left(P^*_i(o^k_{i,t+1} \mid o^{L,k}_{i,t}, a^{L,k}_{i,t}), P_{i,\phi}(o^k_{i,t+1} \mid o^{L,k}_{i,t}, a^{k,L}_{i,t})\right), \tag{3}$$

where $P_i^*$ represents real-world transition dynamics for an agent $i$ and $d(\cdot)$ is a divergence measure (e.g., Kullback-Leibler divergence). We arrive at this objective by replacing the single-agent observations and actions from the vanilla GAT objective shown in Equation (1) with local joint states (observations) and actions. Note that JL-GAT attempts to model the transition to the next individual observation $o_{i,t+1}^k$ for a trajectory $k$ as opposed to a local joint observation.

#### 5.2.3 Forward and Inverse Models in JL-GAT

In this section, we present the forward and inverse models employed in JL-GAT. We then highlight the advantages of our modifications to both vanilla and decentralized GAT. Finally, we explain how we strike a balance between centralized and decentralized GAT, effectively combining the strengths of both approaches.

• *The forward model of JL-GAT* predicts the next individual state $\hat{o}_{i,t+1}$ (observation) that would occur in the real environment $E_{\text{real}}$ for agent $i$ if the local joint action $a_{i,t}^L$ was taken in local joint state $o_{i,t}^L$ at time $t$. Applied to traffic signal control, the forward model predicts the next real environment traffic state that would occur if the local joint action is taken in the current local joint traffic state:

$$\hat{o}_{i,t+1} = f_{i,\phi^+}(o_{i,t}^L, a_{i,t}^L) \tag{4}$$

Our setup of the forward model builds on the forward model of the decentralized setup in Section 4.2, where we also approximate the forward model $f_{i,\phi^+}$ with a deep neural network for each agent $i$, now considering local joint information instead of only individual information, and optimize $\phi^+$ by minimizing the Mean Squared Error (MSE) loss:

$$\mathcal{L}(\phi^+) = \text{MSE}(o_{i,t+1}, \hat{o}_{i,t+1}) = \text{MSE}(o_{i,t+1}, f_{i,\phi^+}(o_{i,t}^L, a_{i,t}^L)) \tag{5}$$

where $o_{i,t}^L$, $a_{i,t}^L$, and $o_{i,t+1}$ are sampled from trajectories collected in $E_{\text{real}}$. Note that the forward model in JL-GAT predicts a single next state (observation) $\hat{o}_{i,t+1}$ for each agent $i$ as in the decentralized GAT setup. In this way, JL-GAT avoids the pitfall of attempting to predict neighboring agent observations, as those neighbors may be influenced by other agents at distance $d$ beyond the predefined radius $r$. Furthermore, by including the actions $a_{j,t}$ of neighboring agents $j$ within $r$, the forward model assumes that the neighboring agent actions will remain fixed. This assumption has significant implications for the setup of the inverse model in JL-GAT, and if violated, gives way to the *cascading invalidation effect* described in Section 5.3.

• *The inverse model of JL-GAT* predicts a grounded action $\hat{a}_{i,t}^{\text{g}}$ for agent $i$ at time $t$ that would attempt to transition the current local joint observation $o_{i,t}^L$ to the predicted individual next observation $\hat{o}_{i,t+1}$ in the simulated environment $E_{\text{sim}}$. We further deviate from previous grounded action transformation works by including additional action information into the inverse model to predict a grounded action $\hat{a}_{i,t}^{\text{g}}$ for agent $i$. We use $a_{i,t}^L$, which incorporates the actions $a_{j,t}$ of neighboring agents $j$ within a predefined radius $r$ (as described in Section 5.2.1), as input to the inverse model for JL-GAT. This implicitly assumes that their actions in $E_{\text{sim}}$ remain unchanged at time $t$:

$$\hat{a}_{i,t}^{\text{g}} = h_{i,\phi^-}(o_{i,t}^L, a_{i,t}^L, \hat{o}_{i,t+1}) \tag{6}$$

Including neighboring agent actions $a_{j,t}$ into the inverse model is invaluable for multi-agent settings, as it allows us to capture local agent interactions that affect the transition dynamics of a single agent $i$. Furthermore, we previously assumed neighboring agent actions would remain unchanged with our input to the forward model, thus it is a natural extension of the inverse model to also include this information. A key insight is that these assumptions lead to the *cascading invalidation effect* described in Section 5.3. We conduct an ablation study in Section 6.4, on this additional information, further reinforcing its necessity in JL-GAT. As in the forward model, we build on the inverse model

from decentralized GAT in Section 4.2 and approximate $h_{i,\phi^-}$ with a deep neural network for each agent $i$ and optimize $\phi^-$ by minimizing the Categorical Cross-Entropy (CCE) Loss:

$$\mathcal{L}(\phi^-) = CCE(a^{\text{g}}_{i,t}, \hat{a}^{\text{g}}_{i,t}) = CCE(a^{\text{g}}_{i,t}, h_{i,\phi^-}(o^L_{i,t}, a^L_{i,t}, \hat{o}_{i,t+1})) \tag{7}$$

where $a^{\text{g}}_{i,t}$, $o^L_{i,t}$, and $\hat{o}_{i,t+1}$ are sampled from trajectories collected in $E_{\text{sim}}$.

### 5.3 Cascading Invalidation Effect

While adapting JL-GAT to include local joint information, we observe a unique challenge, namely the *cascading invalidation effect*. This problem arises from the use of state and action information from neighboring agents to predict the next state that would occur in $E_{\text{real}}$, as shown in Equation (4). When using neighboring state and action information to attempt to bring the transition dynamics of $E_{\text{sim}}$ closer to $E_{\text{real}}$, the underlying assumption is that the actions of neighbor agents will remain unchanged in $E_{\text{sim}}$. If the actions of an agent and one of its neighbors within the predefined radius $r$ are grounded simultaneously, both grounded actions become invalid and may no longer aid in reducing the sim-to-real gap. This is due to the fact that while grounding actions, we assume neighbor actions will not change. We also observe this effect cascade through a network of agents if grounding sequentially, as each agent grounds their action, assuming neighbor actions will remain unchanged. To overcome the cascading invalidation effect, we propose two different approaches:

• *Pattern Grounding*. This approach is simple yet effective: we set a pattern to ground specific agents during a training epoch to avoid any grounding assumption conflicts. We visualize pattern grounding in Figure 2. For example, in our experiments for traffic signal control, we utilize a 1x3 traffic network and apply pattern grounding by grounding only the first and last agent for an epoch. Then, we ground only the agent in between them for the next epoch, alternating between the two set grounding patterns. This directly overcomes the cascading invalidation effect by avoiding grounding agents whose actions have been assumed fixed, but a rigid grounding pattern reduces flexibility during training. This approach can also be paired with *probabilistic grounding*, but for our evaluations, we focused solely on applying each technique separately.

• *Probabilistic Grounding*. In this approach, we let $P^i_{\text{ground}}(t)$ represent the probability of grounding an action $a_{i,t}$ for each agent $i$ at time step $t$: $P^i_{\text{ground}}(t) = p_{\text{ground}}$. Using probability to determine when grounding occurs introduces flexibility by allowing different grounding patterns to emerge naturally across epochs, as opposed to a fixed or rigid scheme. As demonstrated in Tables 1 and 2, this approach led to strong performance for JL-GAT. Although probabilistic grounding does not directly overcome the cascading invalidation effect as pattern grounding does, it often circumvents this challenge by using a fixed probability to ground, which introduces some trade-offs. In particular, this can lead to training scenarios in the simulated environment $E_{\text{sim}}$ that do not accurately reflect the transition dynamics of the real environment $E_{\text{real}}$. This is due to the less restrictive grounding requirements in probabilistic grounding compared to pattern grounding, which enables agents to ground their actions independently without requiring consideration of whether neighboring agents are simultaneously utilizing their actions for grounding. Furthermore, decreasing the grounding probability $P^i_{\text{ground}}(t)$ for each agent $i$ inherently mitigates the likelihood of cascading invalidation. However, this comes at the cost of reducing the amount of grounding during training, which may result in a larger sim-to-real gap. We experiment with various probabilities in Section 6.5, where we recommend $1/N$ as a starting point for probabilistic grounding based on empirical evaluation.

We acknowledge that there are several alternative solutions to the cascading invalidation effect that remain to be explored, such as clustering groups for grounding, learned grounding patterns, and algorithmic approaches to grounding. These avenues are left for future work.

### 5.4 Training Algorithm

We present the training procedure for JL-GAT in Algorithm 1. The algorithm takes as input initial policies $\pi_{i,\theta}$, forward models $f_{i,\phi^+}$, and inverse models $h_{i,\phi^-}$ for each agent $i$, as well as simulation and real-world datasets $\mathcal{D}_{\text{sim}}$ and $\mathcal{D}_{\text{real}}$ (collected offline or from rollouts (Da et al., 2023b)). A sensing radius $r$ is required to determine neighboring agent interactions for grounding, and an optional grounding pattern or probability may also be specified. The output includes updated policies and models for each agent. Training begins with $M$ iterations of policy pre-training in $E_{\text{sim}}$, followed by multiple epochs consisting of: (1) optional policy rollouts in $E_{\text{sim}}$ and $E_{\text{real}}$ to populate $\mathcal{D}_{\text{sim}}$ and $\mathcal{D}_{\text{real}}$; (2) updates to $f_{i,\phi^+}$ and $h_{i,\phi^-}$ using the collected data; (3) policy training episodes using grounded actions to align simulated dynamics with the real world; and (4) reinforcement learning-based policy updates in $E_{\text{sim}}$ with improved dynamics to reduce the sim-to-real gap.

## 6 Experiments and Results

In this section, we introduce our experiment setup and evaluation metrics, which closely follow that of (Da et al., 2024b), demonstrating both the existence of a performance gap between simulation and real environments and the effectiveness of JL-GAT in reducing this gap. We also perform an ablation study to demonstrate the necessity of all additional information to the forward and inverse models in JL-GAT. Lastly, we perform evaluations with different probabilistic grounding settings and explore the pairing of JL-GAT with uncertainty quantification from (Da et al., 2023b).

### 6.1 Environments

We built our implementation of JL-GAT on top of LibSignal (Mei et al., 2024), an open-source environment for traffic signal control with multiple simulation environments. For our experiments, we consider CityFlow (Zhang et al., 2019) as the simulation environment $E_{\text{sim}}$, and SUMO (Behrisch et al., 2011) as the real environment $E_{\text{real}}$. We use a sim-to-sim setup to mimic a sim-to-real deployment process with the main benefit of reproducibility and avoiding the negative impact of unexpected behaviors in the real world, as described in (Da et al., 2024b;c). Our experiments consider two environmental conditions to showcase the sim-to-real gap: rainy and snowy, and we detail their parameter settings in Table 5 (Supplementary Materials).

- *Default settings*. This represents the default settings for CityFlow and SUMO, which we consider $E_{\text{sim}}$ and $E_{\text{real}}$, respectively.
- *Adverse Weather conditions*. We model the effect of adverse weather conditions that are unaccounted for when training a TSC policy in $E_{\text{sim}}$ by varying parameters in $E_{\text{real}}$, such as acceleration, deceleration, emergency deceleration, and startup delay shown in Table 5. We attempt to mimic real-world adverse weather effects, such as wet and icy roads, by reducing the acceleration and deceleration rates of vehicles and increasing their startup delay.

### 6.2 Evaluation Metrics

Building on common practices in traffic signal control (TSC), as described in recent literature (Wei et al., 2021), we adopt the following standard metrics to assess policy performance. Average Travel Time (ATT) represents the average travel time $t$ for vehicles in a given road network, where lower ATT values indicate better control policy performance. Queue measures the number of vehicles waiting at a particular intersection, and we report the average queue over all intersections in a given road network, with smaller values being preferable. Delay captures the average time $t$ that vehicles wait in the traffic network, where lower delay is desirable. Throughput (TP) quantifies the number of vehicles that have completed their trip in a given road network, with higher TP values being better. Lastly, reward represents the return associated with taking an action $a_t$ in a state $s_t$ in RL. We use the same reward metric as (Wei et al., 2019a), defining the reward as negative pressure, and we report the sum of rewards for all intersections in our experiments.

In this work, we adopt the calculation metric for the performance gap between $E_{\text{sim}}$ and $E_{\text{real}}$ from (Da et al., 2024b) and (Da et al., 2023b). Specifically, for a metric $\psi$, we use the following equation to calculate the gap $\Delta$: $\psi_{\Delta} = \psi_{\text{real}} - \psi_{\text{sim}}$. Our goal is to reduce this sim-to-real gap by bringing the transition dynamics of $E_{\text{sim}}$ closer to $E_{\text{real}}$ while training through GAT. We report the $\Delta$ values for each metric, where smaller values are better for $ATT_{\Delta}$, $Queue_{\Delta}$, and $Delay_{\Delta}$, and larger values are better for $TP_{\Delta}$, and $Reward_{\Delta}$ because they are negative values.

### 6.3 Main Results

To highlight the sim-to-real gap in multi-agent traffic signal control (TSC), we conduct experiments in the rainy and snowy settings introduced in Section 6.1, with parameters detailed in Table 5. We begin by evaluating the Direct Transfer approach: agents (Section 3.2) are trained from scratch in $E_{\text{sim}}$ through six independent trials of 300 epochs each. After six independent trials for each network size, the best-performing policies based on lowest average travel time (ATT) are tested in $E_{\text{real}}$. These policies then serve as initialization for various GAT-based multi-agent training configurations, including JL-GAT. The resulting performance metrics are visualized in Figures 4 and 5 in the Supplementary Materials. Full numerical results, including standard deviations and sim-to-real gap calculations, are provided in Tables 1 and 2. A clear performance drop is observed when directly transferring policies from $E_{\text{sim}}$ to $E_{\text{real}}$, illustrating the sim-to-real gap.

Table 1: Rainy environment performance using Direct Transfer as compared to centralized GAT, decentralized GAT, and two versions of our proposed method JL-GAT. For each GAT configuration and network size pair, we run six independent trials and identify the best epoch in each trial based on the lowest average travel time (ATT) in $E_{\text{real}}$. Reported metrics are averaged across these six best epochs (one per trial). The value in () shows the metric gap $\psi$ between $E_{\text{sim}}$ and $E_{\text{real}}$ and $\pm$ shows the sample standard deviation after six trials. The $\uparrow$ indicates that a higher value represents a better performance for a metric and the $\downarrow$ indicates that a lower value represents a better performance for a metric. Note that Direct Transfer is reported as the policies from the best performing epoch (by lowest ATT) in $E_{\text{sim}}$ being tested in $E_{\text{real}}$ after six trials of 300 epochs.

| Network | Method | ATT ($\Delta \downarrow$) | Queue ($\Delta \downarrow$) | Delay ($\Delta \downarrow$) | TP ($\Delta \uparrow$) | Reward ($\Delta \uparrow$) |
|---|---|---|---|---|---|---|
| 1x3 | Direct Transfer | 309.90 (188.64) | 67.66 (43.60) | 0.64 (0.23) | 4784 (-776) | -202.85 (-141.21) |
| | Centralized GAT | 297.57(176.31)±16.12 | 65.59(41.53)±5.83 | 0.63(0.22)±0.01 | 4857(-702)±96.37 | -189.87(-128.23)±15.64 |
| | Decentralized GAT | 276.92(155.66)±16.53 | 59.07(35.01)±6.00 | 0.63(0.22)±0.01 | 5004(-555)±119.40 | -175.30(-113.66)±16.12 |
| | JL-GAT (Pattern) | 265.64(144.38)±6.72 | 51.96(27.90)±3.52 | 0.62(0.21)±0.005 | **5073(-487)±39.36** | -156.27(-94.63)±10.10 |
| | JL-GAT (Probabilistic $1/N$ = 33%) | **263.01(141.75)±2.59** | **50.63(26.57)±1.76** | **0.61(0.20)±0.005** | 5065(-494)±39.77 | **-152.12(-90.48)±5.08** |
| 4x4 | Direct Transfer | 485.63(158.38) | 6.89(5.39) | 0.19(0.11) | 2608(-320) | -90.77(-71.48) |
| | Centralized GAT | 485.63(158.38)±0.00 | 6.89(5.39)±0.00 | 0.19(0.11)±0.00 | 2608(-320)±0.00 | -90.77(-71.48)±0.00 |
| | Decentralized GAT | 476.69(149.44)±4.53 | 6.39(4.88)±0.37 | 0.18(0.10)±0.004 | 2620(-307)±10.30 | -84.31(-65.03)±2.38 |
| | JL-GAT (Pattern) | 468.81(141.56)±2.42 | 5.99(4.48)±0.12 | **0.18(0.10)±0.002** | **2627(-300)±5.05** | **-83.47(-64.18)±2.08** |
| | JL-GAT (Probabilistic $1/N$ = 6.25%) | **467.11(139.86)±1.77** | **5.85(4.34)±0.17** | 0.18(0.10)±0.004 | 2625(-302)±7.06 | -85.33(-66.04)±1.60 |

Table 2: Snowy environment performance using Direct Transfer as compared to centralized GAT, decentralized GAT, and two versions of our proposed method JL-GAT. Refer to Table 1 for details on the reporting methodology.

| Network | Method | ATT ($\Delta \downarrow$) | Queue ($\Delta \downarrow$) | Delay ($\Delta \downarrow$) | TP ($\Delta \uparrow$) | Reward ($\Delta \uparrow$) |
|---|---|---|---|---|---|---|
| 1x3 | Direct Transfer | 473.29 (352.02) | 49.11 (25.05) | 0.66 (0.24) | 4297 (-1263) | -160.69 (-99.05) |
| | Centralized GAT | 472.67(351.41)±1.51 | 49.20(25.14)±0.21 | **0.65(0.24)±0.004** | 4316(-1243)±47.77 | -160.46(-98.82)±0.57 |
| | Decentralized GAT | 463.37(342.11)±11.84 | 54.27(30.21)±7.52 | 0.66(0.25)±0.01 | 4402(-1157)±96.01 | -166.85(-105.21)±17.78 |
| | JL-GAT (Pattern) | 459.28(338.02)±2.79 | 50.59(26.53)±5.12 | 0.65(0.24)±0.01 | 4414(-1145)±40.40 | -157.20(-95.56)±11.17 |
| | JL-GAT (Probabilistic $1/N$ = 33%) | **456.14(334.88)±6.09** | **46.39(22.33)±3.26** | 0.65(0.24)±0.005 | **4436(-1123)±27.18** | **-147.97(-86.33)±9.20** |
| 4x4 | Direct Transfer | 593.06 (265.81) | 6.83 (5.33) | 0.20 (0.12) | 2423 (-505) | -96.28 (-76.99) |
| | Centralized GAT | 593.06(265.81)±0.00 | 6.83(5.33)±0.00 | 0.20(0.12)±0.00 | 2423(-505)±0.00 | -96.28(-76.99)±0.00 |
| | Decentralized GAT | 573.07(245.82)±4.09 | 5.70(4.19)±0.27 | 0.19(0.11)±0.004 | 2467(-460)±4.97 | -83.90(-64.61)±3.17 |
| | JL-GAT (Pattern) | 567.75(240.50)±1.96 | 5.50(3.99)±0.08 | 0.19(0.11)±0.005 | **2471(-457)±7.85** | -83.83(-64.54)±1.51 |
| | JL-GAT (Probabilistic $1/N$ = 6.25%) | **566.22(238.97)±2.64** | **5.28(3.77)±0.18** | **0.18(0.10)±0.004** | 2470(-457)±3.97 | **-82.32(-63.03)±1.24** |

### 6.4 Ablation Study

To show how different parts in JL-GAT help sim-to-real transfer, we conduct an ablation study on the addition of neighboring information in the forward and inverse models of JL-GAT. For this study,

we focus on the rainy 1x3 environment while systematically varying the removal of neighboring states and action information used in JL-GAT. We present the average performance of each metric for the best episode of each method. These results are based on two trials over 300 epochs, as shown in Figure 3, with full details including sim-to-real gap computation and sample standard deviation shown in Table 7. The last two methods failed to improve the Direct Transfer models used for initialization, indicating the necessity of all required modules for JL-GAT.

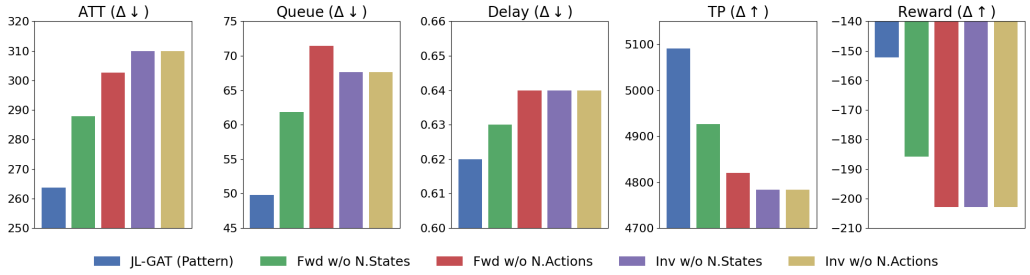


Figure 3: The ablation study on JL-GAT. We systematically vary the information used in the GAT models of JL-GAT to demonstrate the necessity of including neighboring agent information in all parts of GAT. The bars show the average performance of each metric over the best episodes of each method after two trials in the 1x3 rainy environment. Each plot displays the methods in the order they appear from left to right, as indicated in the legend. Full details including sim-to-real gap computation and sample standard deviation are shown in Table 7.

### 6.5 Probabilistic Grounding Settings

We experiment with various probability grounding settings for JL-GAT to test the robustness of JL-GAT for different probability settings. We focus on four different variations of probability grounding, including $1/N$, which sets the grounding probability proportional to the number of agents in the environment. We report the best performance for each setting over 300 epochs in Table 3. The results show that using a probability of 0.2 produces the best performance across all metrics in the 1x3 rainy environment. However, we recommend $1/N$ as a starting place for probabilistic grounding, as our results from Tables 1, 2, and 3 demonstrate a strong performance from the $1/N$ setting.

Table 3: Probability grounding settings for JL-GAT in 1x3 rainy environment.

| Probability | ATT (Δ ↓) | Queue (Δ ↓) | Delay (Δ ↓) | TP (Δ ↑) | Reward (Δ ↑) |
|---|---|---|---|---|---|
| 0.2 | **260.77(139.51)±4.73** | **50.23(26.17)±2.24** | 0.62(0.21)±0.005 | **5115(-445)±36.06** | **-151.34(-89.69)±5.09** |
| 0.5 | 281.73(160.47)±29.87 | 56.19(32.14)±16.36 | **0.61(0.20)±0.01** | 4909(-651)±209.30 | -170.52(-108.87)±39.52 |
| 0.8 | 297.75(176.49)±6.70 | 66.78(42.73)±5.97 | 0.63(0.22)±0.0001 | 4828(-732)±276.48 | -187.69(-126.05)±7.32 |
| 1/N (0.3) | 261.56(140.30)±1.30 | 50.28(26.22)±2.59 | **0.61(0.20)±0.01** | 5062(-498)±25.38 | -155.33(-93.68)±4.24 |

### 6.6 JL-GAT with Uncertainty Quantification

Sim-to-real transfer can introduce uncertainty in action effectiveness due to discrepancies between simulated and real-world dynamics. Prior work has investigated uncertainty quantification (UQ) techniques to improve the reliability of decision-making (Abdar et al., 2021; Liu et al., 2025) such as MC dropout (Gal & Ghahramani, 2016), Deep Ensembles (Rahaman et al., 2021), Evidential Deep Learning (EDL) (Deng et al., 2023), and methods based on eigenvalues (Thompson et al., 2019), etc. To explore whether UQ can enhance JL-GAT, we incorporate the dynamic grounding rate method from (Da et al., 2023b), which adjusts the application of grounding based on model uncertainty. Specifically, for each agent, we compute the average model uncertainty over the previous two epochs and use it to determine whether to ground the current action. If the agent's predicted uncertainty exceeds a dynamic threshold, the original (non-grounded) action is used instead. We evaluate this

uncertainty-aware version of JL-GAT in both rainy and snowy environments over three trials of 300 epochs each, and display the results in Table 4. The results indicate that integrating UQ with JL-GAT further reduces the sim-to-real gap in the 1×3 setting.

Table 4: Uncertainty quantification in JL-GAT for 1x3 traffic network.

| Environment | Method | ATT (Δ ↓) | Queue (Δ ↓) | Delay (Δ ↓) | TP (Δ ↑) | Reward (Δ ↑) |
|---|---|---|---|---|---|---|
| **Rainy** | JL-GAT (Pattern) | 263.61(142.35)±4.66 | 49.82(25.76)±1.46 | **0.62(0.21)±0.004** | 5091(-469)±20.26 | -152.20(-90.55)±5.96 |
| | JL-GAT w/ Uncertainty | **261.53(140.26)±4.56** | **49.65(25.59)±4.19** | **0.62(0.21)±0.01** | **5092(-468)±16.07** | **-148.15(-86.51)±11.73** |
| **Snowy** | JL-GAT (Pattern) | 459.46(338.20)±3.89 | 47.13(23.07)±4.56 | 0.65(0.24)±0.01 | 4417(-1143)±20.26 | -150.40(-88.76)±12.10 |
| | JL-GAT w/ Uncertainty | **456.92(335.66)±4.87** | **44.51(20.45)±8.23** | **0.64(0.23)±0.02** | **4444(-1116)±48.87** | **-141.41(-79.76)±15.80** |

## 7 Conclusion

We have identified a significant performance gap that arises when directly transferring MARL-based TSC policies from simulation to the real world, primarily due to shifts in environment dynamics. To address this, we proposed JL-GAT, a scalable framework that extends Grounded Action Transformation (GAT) to the MARL-based TSC setting. JL-GAT enhances the performance of a decentralized approach to GAT, where each agent has its own GAT models, by incorporating neighboring agent information. This allows JL-GAT to model inter-agent dynamics as in a centralized approach, without sacrificing the scalability of a decentralized approach. Extensive experiments across diverse traffic networks and simulated adverse weather conditions confirm that the hybrid design of JL-GAT consistently reduces the sim-to-real performance gap. A key challenge we identified in the multi-agent GAT setting is the *cascading invalidation effect*, which arises when multiple agents simultaneously ground their actions under the incorrect assumption that neighboring agents' actions remain fixed. Although we introduced two methods to mitigate this issue, a promising direction for future work lies in dynamically selecting which agents should engage in GAT and when.

## Acknowledgments

The work was partially supported by NSF awards #2421839, Amazon Research Awards, NAIRR #240120 and used AWS through the CloudBank project, which is supported by NSF grant #1925001. The views and conclusions in this paper are those of the authors and should not be interpreted as representing any funding agencies.

## References

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

PG Balaji and Dipti Srinivasan. Multi-agent system in urban traffic signal control. *IEEE Computational Intelligence Magazine*, 5(4):43–51, 2010.

Michael Balmer, Kai Nagel, and Bryan Raney. Large-scale multi-agent simulations for transportation applications. In *Intelligent Transportation Systems*, volume 8, pp. 205–221. Taylor & Francis, 2004.

Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 4243–4250. IEEE, 2018.

Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 3414–3421, 2020.

Min Chee Choy, Dipti Srinivasan, and Ruey Long Cheu. Cooperative, hybrid agent architecture for real-time traffic signal control. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: systems and humans*, 33(5):597–607, 2003.

Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, may 2015. DOI: 10.1038/nature14422. URL https://doi.org/10.1038%2Fnature14422.

Mark Cutler, Thomas J. Walsh, and Jonathan P. How. Reinforcement learning with multi-fidelity simulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3888–3895, 2014. DOI: 10.1109/ICRA.2014.6907423.

Longchao Da, Hao Mei, Romir Sharma, and Hua Wei. Sim2real transfer for traffic signal control. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pp. 1–2. IEEE, 2023a.

Longchao Da, Hao Mei, Romir Sharma, and Hua Wei. Uncertainty-aware grounded action transformation towards sim-to-real transfer for traffic signal control. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 1124–1129. IEEE, 2023b.

Longchao Da, Chen Chu, Weinan Zhang, and Hua Wei. Cityflower: An efficient and realistic traffic simulator with embedded machine learning models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 368–373. Springer, 2024a.

Longchao Da, Minquan Gao, Hao Mei, and Hua Wei. Prompt to transfer: Sim-to-real transfer for traffic signal control with prompt learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 82–90, 2024b.

Longchao Da, Porter Jenkins, Trevor Schwantes, Jeffrey Dotson, and Hua Wei. Probabilistic offline policy ranking with approximate bayesian computation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 20370–20378, 2024c.

Longchao Da, Justin Turnau, Thirulogasankar Pranav Kutralingam, Alvaro Velasquez, Paulo Shakarian, and Hua Wei. A survey of sim-to-real methods in rl: Progress, prospects and challenges with foundation models. *arXiv preprint arXiv:2502.13187*, 2025.

Danruo Deng, Guangyong Chen, Yang Yu, Furui Liu, and Pheng-Ann Heng. Uncertainty estimation by fisher information-based evidential deep learning. In *International conference on machine learning*, pp. 7596–7616. PMLR, 2023.

Siddarth Desai, Ishan Durugkar, Haresh Karnan, Garrett Warnell, Josiah Hanna, and Peter Stone. An imitation from observation approach to transfer learning with dynamics mismatch. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS 2020)*, December 2020a.

Siddharth Desai, Haresh Karnan, Josiah P. Hanna, Garrett Warnell, and Peter Stone. Stochastic grounded action transformation for robot learning in simulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2020)*, October 2020b.

François Dion and Bruce Hellinga. A rule-based real-time traffic responsive signal control system with transit priority: application to an isolated intersection. *Transportation Research Part B: Methodological*, 36(4):325–343, 2002.

Kuan Fang, Yunfei Bai, Stefan Hinterstoisser, Silvio Savarese, and Mrinal Kalakrishnan. Multi-task domain adaptation for deep learning of instance grasping from simulation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3516–3523. IEEE, 2018.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Te Han, Chao Liu, Wenguang Yang, and Dongxiang Jiang. Learning transferable features in deep convolutional neural networks for diagnosing unseen machine conditions. *ISA transactions*, 93: 341–353, 2019.

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

Hao Huang, Zhiqun Hu, Zhaoming Lu, and Xiangming Wen. Network-scale traffic signal control via multiagent reinforcement learning with deep spatiotemporal attentive network. *IEEE transactions on cybernetics*, 53(1):262–274, 2021.

Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12627–12637, 2019.

Haoyuan Jiang, Ziyue Li, Hua Wei, Xuantang Xiong, Jingqing Ruan, Jiaming Lu, Hangyu Mao, and Rui Zhao. X-light: Cross-city traffic signal control using transformer on transformer as meta multi-agent reinforcement learner. *arXiv preprint arXiv:2404.12090*, 2024.

Haresh Karnan, Siddharth Desai, Josiah P. Hanna, Garrett Warnell, and Peter Stone. Reinforced grounded action transformation for sim-to-real transfer. In *IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS 2020)*, October 2020.

Phyllis C Lee and Antonio C de A Moura. Necessity, unpredictability and opportunity: An exploration of ecological and social drivers of behavioral innovation. In *Animal creativity and innovation*, pp. 317–333. Elsevier, 2015.

Xiaoou Liu, Tiejin Chen, Longchao Da, Chacha Chen, Zhen Lin, and Hua Wei. Uncertainty quantification and confidence calibration in large language models: A survey. *arXiv preprint arXiv:2503.15850*, 2025.

Hao Mei, Xiaoliang Lei, Longchao Da, Bin Shi, and Hua Wei. Libsignal: an open library for traffic signal control. *Machine Learning*, 113(8):5235–5271, 2024.

Arthur Müller, Vishal Rangras, Tobias Ferfers, Florian Hufen, Lukas Schreckenberg, Jürgen Jasperneite, Georg Schnittker, Michael Waldmann, Maxim Friesen, and Marco Wiering. Towards real-world deployment of reinforcement learning for traffic signal control. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 507–514. IEEE, 2021.

Syed Shah Sultan Mohiuddin Qadri, Mahmut Ali Gökçe, and Erdinç Öner. State-of-art review of traffic signal control methods: challenges and opportunities. *European transport research review*, 12:1–23, 2020.

Rahul Rahaman et al. Uncertainty quantification and deep ensembles. *Advances in neural information processing systems*, 34:20063–20075, 2021.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013a.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *Journal of Artificial Intelligence Research*, 48:67–113, 2013b.

Roney L Thompson, Aashwin Ananda Mishra, Gianluca Iaccarino, Wouter Edeling, and Luiz Sampaio. Eigenvector perturbation methodology for uncertainty quantification of turbulence models. *Physical Review Fluids*, 4(4):044603, 2019.

Joshua P Tobin. *Real-World Robotic Perception and Control Using Synthetic Data*. University of California, Berkeley, 2019.

Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Xingchao Peng, Sergey Levine, Kate Saenko, and Trevor Darrell. Towards adapting deep visuomotor representations from simulated to real environments. *arXiv preprint arXiv:1511.07111*, 2(3), 2015.

Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. arxiv 2014. *arXiv preprint arXiv:1412.3474*, 2019.

H. Wei, Guanjie. Zheng, H. Yao, and Z. Li. *Intellilight: A reinforcement learning approach for intelligent traffic light control*. Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018.

Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1290–1298, 2019a.

Hua Wei, Guanjie Zheng, Vikash Gayah, and Zhenhui Li. A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117*, 2019b.

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

Hua Wei, Jingxiao Chen, Xiyang Ji, Hongyang Qin, Minwen Deng, Siqin Li, Liang Wang, Weinan Zhang, Yong Yu, Liu Linc, et al. Honor of kings arena: an environment for generalization in competitive reinforcement learning. *Advances in Neural Information Processing Systems*, 35: 11881–11892, 2022.

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

Yiran Zhang, Khoa Vo, Longchao Da, Tiejin Chen, Xiaoou Liu, and Hua Wei. Reproducible and low-cost sim-to-real environment for traffic signal control. In *Proceedings of the ACM/IEEE 16th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2025)*, pp. 1–2, 2025.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE symposium series on computational intelligence (SSCI)*, pp. 737–744. IEEE, 2020.

Guanjie Zheng, Xinshi Zang, Nan Xu, Hua Wei, Zhengyao Yu, Vikash Gayah, Kai Xu, and Zhenhui Li. Diagnosing reinforcement learning for traffic signal control. *arXiv preprint arXiv:1905.04716*, 2019.

# Supplementary Materials

*The following content was not necessarily subject to peer review.*

Table 5: Environment settings used in all experiments.

| **Environment** | **Accel** ($m/s^2$) | **Decel** ($m/s^2$) | **E. Decel** ($m/s^2$) | **S. Delay** ($s$) |
|---|---|---|---|---|
| Default ($E_{\text{sim}}$) | 2.0 | 4.5 | 9.0 | 0.0 |
| Rainy | 0.75 | 3.5 | 4.0 | 0.25 |
| Snowy | 0.5 | 1.5 | 2.0 | 0.5 |

## 8 Dec-POMDP for MARL-based Traffic Signal Control

The traffic signal control (TSC) problem is modeled as a multi-agent reinforcement learning (MARL) task, where each traffic signal operates as an independent agent in a shared environment. The MARL problem is typically formulated as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), defined by the tuple $\langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, P, R, \Omega_i, O, \gamma \rangle$, where: $\mathcal{N}$ is the set of agents (intersections), $\mathcal{S}$ is the global state space, representing traffic conditions (e.g., vehicle queues, speeds). $\mathcal{A}_i$ is the action space for agent $i$, which includes actions such as switching traffic signal phases. $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition function, where $\mathcal{A} = \prod_{i\in\mathcal{N}} \mathcal{A}_i$ is the joint action space, and $\Delta(\mathcal{S})$ denotes the set of probability distributions over $\mathcal{S}$. $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, which evaluates traffic metrics (e.g., queue length, delay). $\Omega_i$ is the observation space for agent $i$, with $\Omega = \prod_{i\in\mathcal{N}} \Omega_i$ being the joint observation space. $O$ is the observation probability function $O(s', a, o) = P(o \mid s', a)$ and defines the probability of receiving a joint observation $o$ given then next state $s'$ and joint action $a$. $\gamma \in [0, 1)$ is the discount factor.

At each time step $t$, agent $i$ observes its own state $o_{i,t} \in \Omega_i$, selects an action $a_{i,t} \in \mathcal{A}_i$, and receives a reward $r_{i,t}$. Agent actions are taken simultaneously and comprise a global action $a_t$, which transitions the environment from a global state $s_t$ to a global next state $s_{t+1}$, where global states consist of observations $o_{i,t}$ for each agent $i$. Global states and actions are represented as: $s_t = (o_1, o_2, \ldots, o_N)$, and $a_t = (a_1, a_2, \ldots, a_N)$. During training, each agent learns a policy $\pi_i : \Omega_i \to \mathcal{A}_i$ with the goal of maximizing its expected cumulative reward: $J_i = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_{i,t}\right]$.

## 9 Agent Design Details

- **State**. Our state is defined for each agent (intersection) as their own observation $o_{i,t}$ in MARL. For this work, we utilize the state definition from PressLight, simplifying it to include only the number of vehicles in each incoming and outgoing lane without lane segmentation.
- **Action**. Each agent selects an action $a_{i,t} \in \mathcal{A}_i$ at time step $t$ that represents the traffic signal phase $p$. In this work, we utilize the same eight phase TSC action space as in (Da et al., 2023b), and represent all actions as one-hot encoded vectors.
- **Reward**. The reward $r_{i,t}$ for each agent $i$ at time step $t$ is defined as negative pressure in PressLight. The goal of each agent is to minimize pressure, which effectively balances the number of vehicles in the traffic network and keeps traffic flowing efficiently.
- **Learning Method**. Each agent is trained using an independent Deep Q-Network (DQN) with experience replay, enabling efficient sampling of past experiences. This approach follows established methods in traffic signal control (Wei et al., 2018). The objective is to optimize the policy $\pi_{i,t}$ for each agent $i$ by using its individual reward $r_{i,t}$ to improve decision-making over time.

## 10 Code Availability

The code used in our experiments is publicly available at https://github.com/DaRL-LibSignal/JL-GAT/.

**Algorithm 1** Algorithm for JL-GAT

**Input:** Initial policies $\pi_{i,\theta}$ for each agent $i$, forward models $f_{i,\phi^+}$ for each agent $i$, inverse models $h_{i,\phi^-}$ for each agent $i$, simulation dataset $\mathcal{D}_{\text{sim}}$, real-world dataset $\mathcal{D}_{\text{sim}}$, sensing radius $r$, grounding pattern or grounding probability $P^i_{\text{ground}}(t)$ for each agent

**Output:** Policies $\pi_{i,\theta}$, forward models $f_{i,\phi^+}$, inverse models $h_{i,\phi^-}$

Pre-train policies $\pi_{i,\theta}$ for each agent $i$ for $M$ iterations in $E_{\text{sim}}$
**for** $e = 1, 2, ..., I$ **do**
    Rollout policy $\pi_{i,\theta}$ for each agent $i$ in $E_{\text{sim}}$ and add data to $\mathcal{D}_{\text{sim}}$ (optional)
    Rollout policy $\pi_{i,\theta}$ for each agent $i$ in $E_{\text{real}}$ and add data to $\mathcal{D}_{\text{real}}$ (optional)
    # Update transformation functions for each agent
    **for** $i = 1, 2, ..., N$ **do**
        Update $f_{i,\phi^+}$ with data from $\mathcal{D}_{\text{real}}$ corresponding to agent $i$ using Equation (5)
        Update $h_{i,\phi^-}$ with data from $\mathcal{D}_{\text{sim}}$ corresponding to agent $i$ using Equation (7)
    **end for**
    # Policy training
    **for** $ep = 1, 2, ..., E$ **do**
        # Action grounding step for each agent $i$ at every time step $t$
        **for** $t = 0, 1, ..., T$-$1$ **do**
            **for** $i = 1, 2, ..., N$ **do**
                $a_{i,t} = \pi_{i,\theta}(o_{i,t})$
                Predict next state $\hat{o}_{i,t+1}$ using Equation (4)
                Calculate grounded action $\hat{a}^{\text{g}}_{i,t}$ using Equation (6)
                # Apply pattern or probabilistic grounding
                **if** grounding is based on a pattern **then**
                    Ground based on a pattern, example shown in Figure 2.
                **else if** grounding is probabilistic **then**
                    Ground with a probability using Equation in Probabilistic Grounding.
                **end if**
            **end for**
        **end for**
        # Policy update step
        Improve policies $\pi_{i,\theta}$ for each agent $i$ with reinforcement learning
    **end for**
**end for**

Table 6: Key Notations and Descriptions in This Paper.

| Symbol | Description |
|---|---|
| $\mathcal{N}$ | Set of agents (traffic signals) |
| $\mathcal{S}$ | Global state space |
| $\mathcal{A}_i$ | Action space for agent $i$ |
| $P$ | Transition function |
| $R$ | Reward function |
| $\gamma$ | Discount factor |
| $o_{i,t}$ | State (observation) of agent $i$ at time $t$ |
| $a_{i,t}$ | Action of agent $i$ at time $t$ |
| $\hat{o}_{i,t+1}$ | Predicted next state (observation) for agent $i$ |
| $\pi_i$ | Policy of agent $i$ |
| $J_i$ | Expected cumulative reward for agent $i$ |
| $\mathcal{D}_{\text{real}}$ | Real-world trajectory dataset |
| $\mathcal{D}_{\text{sim}}$ | Simulation trajectory dataset |
| $P^*$ | Real-world transition dynamics |
| $P_\phi$ | Parameterized simulator dynamics |
| $f_{i,\phi^+}$ | Forward model for agent $i$ |
| $h_{i,\phi^-}$ | Inverse model for agent $i$ |
| $r$ | Sensing radius |
| $d(i,j)$ | Distance between agents $i$ and $j$ |
| $s_t, a_t$ | Global state and action at time $t$ |
| $o^L_{i,t}, a^L_{i,t}$ | Local joint state (observations) and actions for agent $i$ at time $t$ |
| $\hat{a}^{\text{g}}_t$ | Global grounded action at time $t$ |
| $\hat{a}^{\text{g}}_{i,t}$ | Grounded action for agent $i$ at time $t$ |

Table 7: Ablation Study of JL-GAT in 1x3 Rainy Environment.

| Method | ATT (Δ ↓) | Queue (Δ ↓) | Delay (Δ ↓) | TP (Δ ↑) | Reward (Δ ↑) |
|---|---|---|---|---|---|
| JL-GAT (Pattern) | **263.61(142.35)±4.66** | **49.82(25.76)±1.46** | **0.62(0.21)±0.004** | **5091(-469)±20.26** | **-152.20(-90.55)±5.96** |
| Forward Model w/o Neigh. States | 287.96(166.70)±31.03 | 61.82(37.76)±8.26 | 0.63(0.22)±0.01 | 4926(-634)±201.53 | -185.76(-124.11)±24.18 |
| Forward Model w/o Neigh. Actions | 302.65(181.38)±10.26 | 71.41(47.36)±5.30 | 0.64(0.23)±0.01 | 4820(-740)±50.91 | -202.86(-141.22)±0.01 |
| Inverse Model w/o Neigh. States | 309.90(188.64)±0.00 | 67.66(43.60)±0.00 | 0.64(0.23)±0.00 | 4784(-776)±0.00 | -202.85(-141.21)±0.00 |
| Inverse Model w/o Neigh. Actions | 309.90(188.64)±0.00 | 67.66(43.60)±0.00 | 0.64(0.23)±0.00 | 4784(-776)±0.00 | -202.85(-141.21)±0.00 |

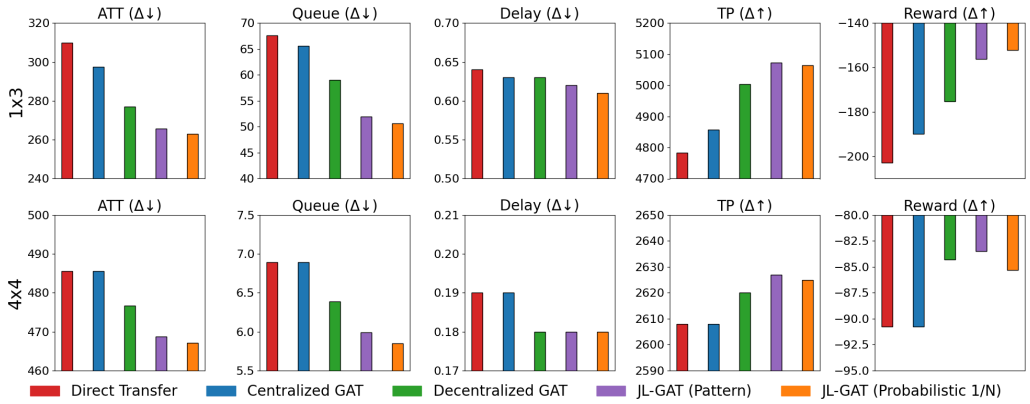


Figure 4: Average performance metrics over the best episode from each trial in the **rainy** environment. Top row: 1×3 traffic network. Bottom row: 4×4 traffic network. The ↑ indicates that a higher value represents a better performance for a metric and the ↓ indicates that a lower value represents a better performance for a metric. Each plot displays the methods in the order they appear from left to right, as indicated in the legend. Full quantitative results, including standard deviations and sim-to-real gap values, are presented in Table 1.

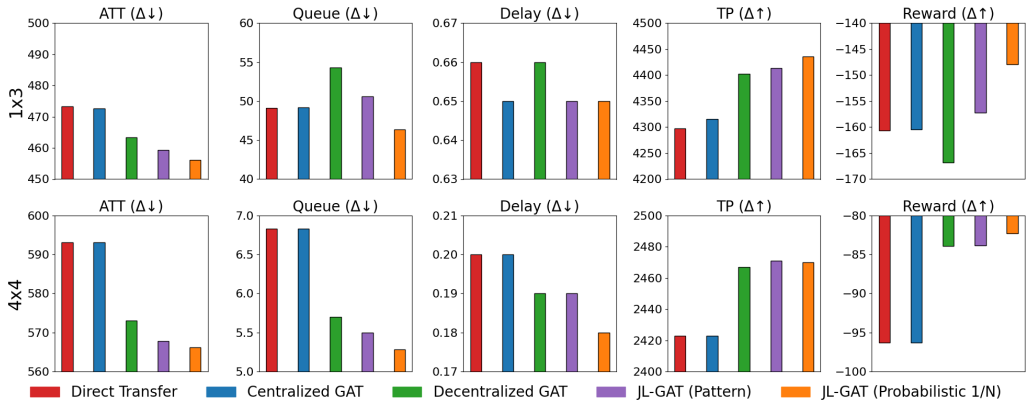


Figure 5: Average performance metrics over the best episode from each trial in the **snowy** environment. Top row: 1×3 traffic network. Bottom row: 4×4 traffic network. The ↑ indicates that a higher value represents a better performance for a metric and the ↓ indicates that a lower value represents a better performance for a metric. Each plot displays the methods in the order they appear from left to right, as indicated in the legend. Full quantitative results, including standard deviations and sim-to-real gap values, are presented in Table 2.