

Fault-Tolerant MARL for CAVs under Observation Perturbations for Highway On-Ramp Merging

Yuchen Shi^{ID}, Huaxin Pei^{ID}, Yi Zhang^{ID}, Senior Member, IEEE, Danya Yao^{ID}, Member, IEEE

Abstract—Multi-Agent Reinforcement Learning (MARL) holds significant promise for enabling cooperative driving among Connected and Automated Vehicles (CAVs). However, its practical application is hindered by a critical limitation, i.e., insufficient fault tolerance against observational faults. Such faults, which appear as perturbations in the vehicles' perceived data, can substantially compromise the performance of MARL-based driving systems. Addressing this problem presents two primary challenges. One is to generate adversarial perturbations that effectively stress the policy during training, and the other is to equip vehicles with the capability to mitigate the impact of corrupted observations. To overcome the challenges, we propose a fault-tolerant MARL method for cooperative on-ramp vehicles incorporating two key agents. First, an adversarial fault injection agent is co-trained to generate perturbations that actively challenge and harden the vehicle policies. Second, we design a novel fault-tolerant vehicle agent equipped with a self-diagnosis capability, which leverages the inherent spatio-temporal correlations in vehicle state sequences to detect faults and reconstruct credible observations, thereby shielding the policy from misleading inputs. Experiments in a simulated highway merging scenario demonstrate that our method significantly outperforms baseline MARL approaches, achieving near-fault-free levels of safety and efficiency under various observation fault patterns.

Index Terms—Cooperative Driving, Multi-agent Reinforcement Learning, Fault Tolerance, Perturbed Observations, On-ramp Merging

I. INTRODUCTION

MULTI-VEHICLE cooperative autonomous driving is anticipated to significantly enhance traffic efficiency and safety [1], [2], [3], [4]. Among the driving scenarios, highway on-ramp merging represents a critical bottleneck where coordination faults can lead to severe congestion and safety hazards. Through real-time information exchange, CAVs can acquire a more comprehensive understanding of traffic conditions, enabling them to make more efficient and safer decisions. However, unavoidable communication and perception faults [5], [6], [7] cause the observed data to deviate from the ground truth. Such observation perturbations directly mislead the decision-making of vehicles receiving faulty data, subsequently propagating the disruptive impact throughout the collaborative system. This propagation can cause the vehicle

This work was supported in part by the National Natural Science Foundation of China under Grant 62503259. (*Corresponding author: Huaxin Pei*.)

Yuchen Shi and Huaxin Pei are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: shiyuche21@mails.tsinghua.edu.cn; phx17@tsinghua.org.cn).

Yi Zhang and Danya Yao are with the Department of Automation, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China. (e-mail: zhyi@tsinghua.edu.cn; yaody@tsinghua.edu.cn).

fleet to lose its cooperative decision-making capability and potentially trigger safety incidents [8].

With the rapid advancement of Reinforcement Learning (RL) technology, its application in the autonomous driving domain has become increasingly prevalent [9]. For cooperative decision-making problems, MARL demonstrates significant advantages, exhibiting powerful exploration capabilities and the potential to handle complex environmental scenarios [10], [11], [12]. However, the high performance of vehicles' policies relies on idealized environmental assumptions. If potential communication or perception faults are overlooked during training, vehicles will likely struggle to respond effectively when encountering unforeseen faults during execution due to insufficient experience and unprepared policies [13]. In on-ramp merging scenarios, if a vehicle misobserves the positions and velocities of the surrounding vehicles, it is highly likely to compromise traffic safety or efficiency, especially at the merging point, as illustrated in Fig. 1. Regrettably, to the best of our knowledge, current MARL studies for cooperative driving rarely consider the impact mechanisms of potential perception or communication faults. The lack of fault tolerance against such potential disturbances has become a critical bottleneck hindering the practical deployment of MARL-based cooperative driving algorithms.

Considering faults during the training process is necessary for improving fault-tolerant ability. Furthermore, even intentionally incorporating fault simulations presents significant challenges. The first challenge is to generate perturbations that effectively stress the policy during training. Compared to rule-based systems, the impact of observation perturbations on MARL's neural networks is harder to model intuitively [14], owing to their black-box and data-driven characteristics. Consequently, random or manual fault injection mostly introduces perturbations with inappropriate magnitudes which have negligible impact, leading the training process to inadequately cover challenging key faults. The second challenge is to equip vehicles with the capability to mitigate the impact of corrupted observations. This stems from the inherent uncertainty under faults: a vehicle cannot determine if its own observations are perturbed, while its perceptions of other vehicles' critical states such as positions and velocities may also be erroneous. This dual uncertainty creates a dilemma: blindly trusting faulty data may lead to unsafe decisions, while excessive distrust forces overly conservative behaviors, significantly compromising traffic efficiency.

To address these challenges, this paper proposes *Observational Fault-Tolerant MARL* (OFT-MARL) for CAVs in on-ramp merging scenarios. OFT-MARL incorporates

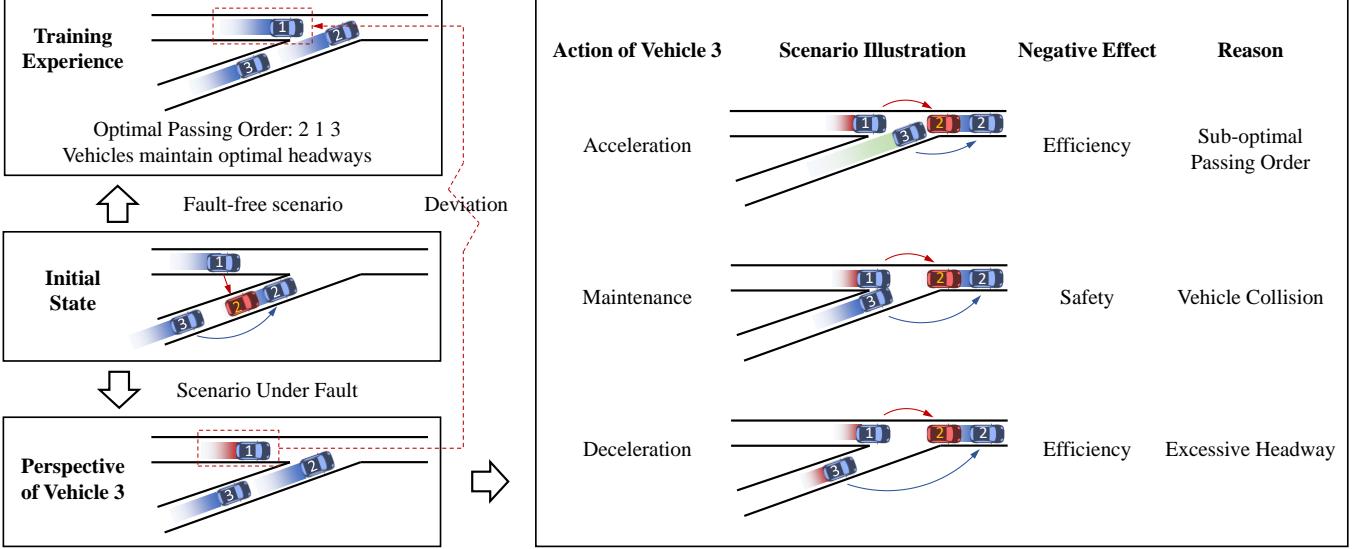


Fig. 1: Vehicle 1 suddenly misobserves the position of vehicle 2, perceiving it at the location marked in red (Left Middle). In the fault-free scenario, all vehicles would maintain velocities and proceed according to the original cooperative strategy (Left Top). However, influenced by the fault, vehicle 1 chooses to decelerate and yield, which is contrasted with the training experience from the perspective of vehicle 3 (Left Bottom). Further, vehicle 3’s action becomes unpredictable due to deviations in vehicle 1’s behavior. Regardless of vehicle 3’s action, traffic efficiency or safety may be affected for different reasons (Right). The blue, green, and red colors behind vehicles indicate their maintenance, acceleration and deceleration behaviors respectively.

two key modules, forming a defensive-offensive synergy: the fault injection agent hardens the policy during training while the fault-tolerant vehicle agent shields it, resulting in a fault-tolerant system. OFT-MARL is designed as follows:

To tackle the first challenge of effective perturbation, we introduce a global fault injection agent. This agent aggregates observations and fault information from all vehicles, and calculates the most disruptive observation perturbation strategy based on global situation analysis. The fault injection agent is co-trained concurrently with the vehicles’ policies. By continuously exposing the vehicle agents to these optimally adversarial perturbations, their policy’s fault-tolerant capabilities are substantially enhanced through adversarial learning.

Simultaneously, to address the second challenge brought about by corrupted observations, we design the vehicle agent with an intrinsic fault diagnosis capability. This capability is grounded in the spatio-temporal correlation inherent in vehicle operational data. For instance, vehicle trajectories should exhibit continuity and smoothness, without frequent occurrences of sharp acceleration or deceleration during normal driving. By analyzing the temporal characteristics of state sequences, the vehicle can discriminate data that significantly deviates from these typical dynamics and subsequently reconstruct credible estimates of the true observations. This provides vehicles with credibility assessments and correction references, thereby assisting them in forming more reasonable and efficient policies.

The rest of the paper is organized as follows. Section II reviews related work, followed by background knowledge in Section III. Section IV details the proposed fault-tolerant MARL approach for CAVs. Section V showcases and analyzes the experimental results. Finally, Section VI concludes the

paper and suggests future research directions.

II. RELATED WORKS

A. Cooperative Driving

Traditional cooperative driving methods are typically achieved by establishing explicit optimization objectives, designing right-of-way allocation rules, and implementing trajectory planning [15], [16], [17], [18]. However, traditional methods can be limited in scalability and adaptability in highly dynamic and complex scenarios. This has motivated the adoption of MARL, which excels at learning cooperative strategies directly from interaction data.

To tackle various practical challenges in traffic, researchers have developed solutions built upon foundational frameworks. Ramp merging scenario is one of the typical scenario for algorithm verification. For instance, Chen *et al.* [19] incorporated a priority-based safety supervisor into MA2C to enhance safety considerations. Pan *et al.* [20] proposed Trust-MARL based on MASAC to control mixed traffic flows on ramps. Chen *et al.* [21] integrated graph neural networks with MADQN to handle dynamic vehicle numbers and rapidly expanding joint action spaces for cooperative lane change control during merging. In this work, to achieve precise velocity control, we enhance the fault tolerance of the MADDPG [22] algorithm, which is suitable for continuous action control, and validate it in ramp scenarios.

B. Fault Tolerance for Cooperative Driving

Classical fault-tolerant methods for cooperative driving are typically implemented through scheduling rule design

or game-theoretic optimization. Pei *et al.* [23] constructs a rule-based fault-tolerant cooperative driving strategy. He *et al.* [24] proposed a global planning and local gaming framework. Liu *et al.* [25] introduced a method for real-time decision adjustments, where faulty and normal vehicles can execute distinct decision models in a distributed manner. Ma *et al.* [26] transformed robust autonomous intersection control into a weighted maximal clique problem with restrictions and employed a heuristic algorithm to explore the solution space. Such traditional methods possess certain advantages in fault handling, for their relatively fixed and predictable decision rules. To address potential input biases in RL-based vehicle policy networks, He *et al.* utilized Bayesian optimization to approximate input perturbations for black-box vehicle policies [27], while applied gradient descent for white-box scenarios with known policy gradients [28]. However, most studies about robust RL-based vehicle policies focus on single-vehicle settings, with limited research on multi-vehicle interactions.

III. BACKGROUND

A. Markov Games

A multi-agent system with N agents can be modeled as decentralized partially observable Markov decision processes (Dec-POMDPs) [29], [30], defined as $\langle S, A, T, R, Y, O, \gamma \rangle$, where S is the set of states, $A = \{A_1, \dots, A_N\}$ is the set of joint actions, T is the transition function, $O = \{O_1, \dots, O_N\}$ is the set of observations, and $Y = \{Y_1, \dots, Y_N\}$ is the set of observation functions. At each time step t , agent i receives a partial observation $o_i = Y_i(s) : S \rightarrow O_i$, takes action a_i according to policy $\pi_{\theta_i} : a_i = \pi_{\theta_i}(o_i) : O_i \rightarrow A_i$, and receives a reward $r_i = R(s, a_1, \dots, a_N) : S \times A_1 \times \dots \times A_N \rightarrow \mathbb{R}$. Therefore, the environment at the next time step can be described as $s' = T(s, a_1, \dots, a_N) : S \times A_1 \times \dots \times A_N \rightarrow S$. The objective for each agent is to maximize its expected discounted reward $\mathbb{E}[R_i] = \mathbb{E}[\sum_{t'=t}^{t_0} \gamma^{t'-t} r_{i,t'}]$, where $\gamma \in [0, 1]$ is the discount factor, $r_{i,t'}$ is the reward at time step t' and t_0 is the end time step of an episode.

B. Deep Deterministic Policy Gradient (DDPG)

DDPG [31] helps an agent to learn a deterministic continuous action $a = \mu_{\theta}(s)$, where μ is the actor network parameterized by θ . The policy gradient is:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{s \sim U(\mathcal{D})} [\nabla_{\theta} \mu_{\theta}(s) \nabla_a Q^{\phi}(s, a)|_{a=\mu_{\theta}(s)}]. \quad (1)$$

Here, $U(\mathcal{D})$ denotes experiences sampled from the replay buffer \mathcal{D} , which stores tuples (s, s', a, r) . $Q^{\phi}(s, a)$ is the Q-value function, also called the critic network. Its parameters ϕ is updated via:

$$\begin{aligned} \mathcal{L}(\phi) &= \mathbb{E}_{(s, a, r, s') \sim U(\mathcal{D})} [(Q^{\phi}(x, a) - y)^2], \\ y &= r + \gamma Q^{\phi'}(s', a')|_{a'=\mu_{\theta'}(s')}, \end{aligned} \quad (2)$$

where $\mu_{\theta'}$ is the target policy with delayed parameters θ' .

MADDPG [22] is a variant of DDPG for multi-agent systems, with centralized Q-value function and decentralized policies.

Considering N agents with continuous deterministic policies $\mu = \{\mu_{\theta_1}, \dots, \mu_{\theta_N}\}$, the policy gradient for agent i is:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{(x, a) \sim U(\mathcal{D})} [\nabla_{\theta_i} \mu_{\theta_i}(o_i) \nabla_a Q_i^{\phi_i}(x, a)|_{a_i=\mu_{\theta_i}(o_i)}]. \quad (3)$$

Here, $Q_i^{\phi_i}(x, a)$ is agent i 's Q-value function, which accepts all agents' actions $a = [a_1, \dots, a_N]$ and state information x . In this paper, x is defined as the concatenation of all agents' observations $[o_1, \dots, o_N]$. \mathcal{D} stores tuples $(x, x', a, r_1, \dots, r_N)$, encapsulating all agents' experiences. ϕ_i is updated via:

$$\begin{aligned} \mathcal{L}(\phi_i) &= \mathbb{E}_{(x, a, r, x') \sim U(\mathcal{D})} [(Q_i^{\phi_i}(x, a) - y_i)^2], \\ y_i &= r_i + \gamma Q_i^{\phi'}(x', a'_1, \dots, a'_N)|_{a'_j=\mu_{\theta'_j}(o'_j)}. \end{aligned} \quad (4)$$

The DDPG and MADDPG frameworks described above form the foundation of our method. We introduce novel fault-tolerant mechanisms considering vehicle characteristics within the established frameworks to address the vulnerability to observational faults.

IV. METHODS

This section details OFT-MARL designed for the highway on-ramp merging scenario. As illustrated in Fig. 2, the framework is composed of the driving scenario along with two interacting agents. The core of our approach lies in the offensive-defensive synergy established between the fault injection agent and the fault-tolerant vehicle agent.

Initially, a problem formulation is established (Section IV-A), followed by the formalization of the observation model and fault model to define the mechanism of observation perturbations (Section IV-B). Subsequently, a fault injection agent is introduced, which adaptively generates strategically disruptive perturbations within bounded constraints from the true observations and reapplies them to the scenario (Section IV-C). The fault-tolerant vehicle agent is equipped with the capability to leverage sequential dependencies for simultaneous fault detection and observation reconstruction, thereby generating reliable actions (Section IV-D). Finally, all components are integrated via a joint training framework (Section IV-E), with details of each component depicted in Fig. 3.

A. Problem Formulation

The scenario addressed in this study is a highway on-ramp merging configuration. Denote the fleet size by N , where each vehicle $i \in \{1, \dots, N\}$ has a state vector $s_i \in \mathbb{R}^d$. Normally, vehicle i gets true observation $o_i = Y_i(s)$. However, communication or perception faults impair observation function Y_i , affecting the accuracy of the observation and causing o_i to diverge from the ground-truth state s . To facilitate a clear representation, we denote o_i and Y_i as the pristine observation and observation function, respectively, while \hat{o}_i and \hat{Y}_i represent the potentially perturbed observation and observation function. The space of all possible perturbed observations is denoted by \hat{O} .

To address the problem, our methodology proceeds as follows: first, we formally define Y_i and \hat{Y}_i while constraining the domain of \hat{O} ; second, we introduce the fault injection method $f : O \rightarrow \hat{O}$; and finally, we articulate the policy

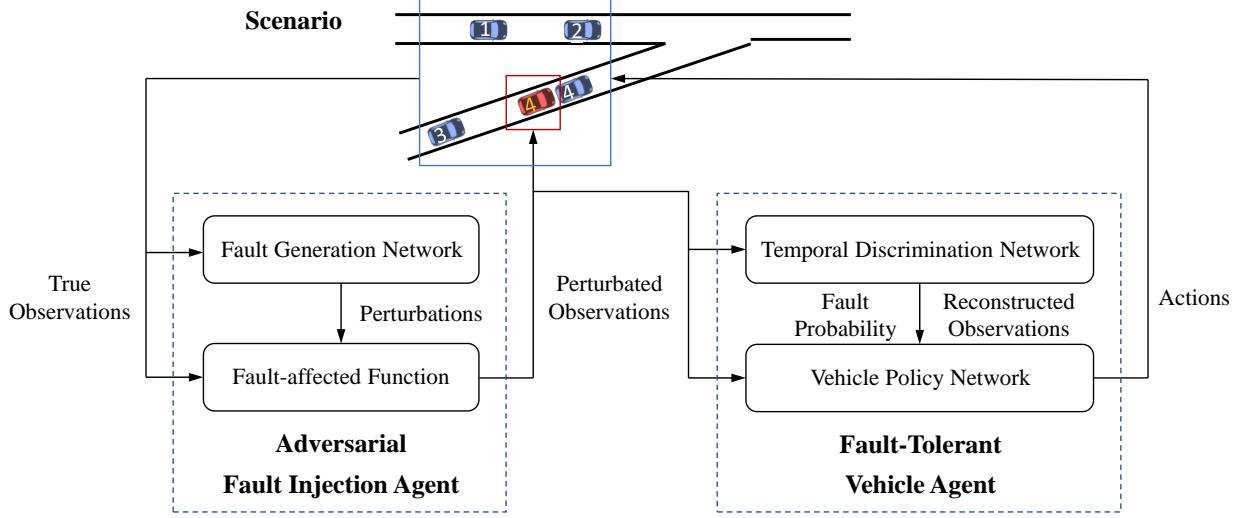


Fig. 2: Overall framework of OFT-MARL.

$\mu : \hat{O}_i \rightarrow A_i$ that maps perturbed observations to actions. These components collectively form the foundational issues examined in the subsequent sections.

B. Observation Modeling under Faults

We formally define the observation model under normal and fault conditions, and establish perturbation constraints.

Each vehicle's policy depends primarily on neighboring vehicles. To reduce computational and communication overhead, vehicle i observes only its own state and those of m surrounding vehicles [19], [32]. Let $\mathcal{N}_i = [\mathcal{N}_i^1, \dots, \mathcal{N}_i^m]$ be the neighbor index array of vehicle i , where $\mathcal{N}_i^j \in \{0, 1, \dots, N\}$, $j = 1, \dots, m$. The neighbor may not exist due to spatial distribution or insufficient vehicle density, in which case we set $\mathcal{N}_i^j = 0$. As depicted in Fig. 3 Top, $\mathcal{N}_1 = [2, 0, 4, 3]$ shows the distribution of vehicles at 4 relative positions around vehicle 1. The true observation vector $o_i \in \mathbb{R}^{(m+1)d}$ is constructed as:

$$o_i = Y_i(s) = [s_i, o_{i,1}, \dots, o_{i,m}], \quad (5)$$

where s_i is the state of vehicle i and $o_{i,j} \in \mathbb{R}^d$ is the observation vector from vehicle i to its j -th neighbor:

$$o_{i,j} = \begin{cases} \mathbf{0}_d, & \mathcal{N}_i^j = 0, \\ o_{i\mathcal{N}_i^j}, & \mathcal{N}_i^j > 0. \end{cases} \quad (6)$$

Here, $\mathbf{0}_d$ denotes a d -dimensional zero vector when neighbor j is absent and o_{ij} is the true observation from vehicle i to vehicle \mathcal{N}_i^j .

The observation vector considering potential faults $\hat{o}_i \in \mathbb{R}^{(m+1)d}$ is constructed as:

$$\hat{o}_i = \hat{Y}_i(s) = [s_i, \hat{o}_{i,1}, \dots, \hat{o}_{i,m}], \quad (7)$$

where $\hat{o}_{i,j}$ is the observation vector from vehicle i to its j -th neighbor, which may contain potential perturbations:

$$\hat{o}_{i,j} = \begin{cases} \mathbf{0}_d, & \mathcal{N}_i^j = 0, \\ f(o_{i\mathcal{N}_i^j}, b), & \mathcal{N}_i^j > 0 \text{ under fault condition,} \\ o_{i\mathcal{N}_i^j}, & \mathcal{N}_i^j > 0 \text{ under normal condition.} \end{cases} \quad (8)$$

Here, $f(o_{i\mathcal{N}_i^j}, b)$ presents the fault-affected observation. b is the deviation magnitude, applied to the true observation $o_{i\mathcal{N}_i^j}$ through fault-affected function f . The function will be detailedly described in Section IV-C.

To bound fault magnitudes, we define an l -norm ball $B(o)$ [33], [34]. The perturbed observation $f(o, b)$ should lie within the ball centered at o :

$$B(o) := \{f(o, b) \in \mathbb{R}^d : \|f(o, b) - o\|_l \leq \epsilon\}, \quad (9)$$

where $\epsilon > 0$ is the perturbation budget. This constraint ensures the problem remains well-posed, as excessive perturbations provide limited practical utility for policy decisions and are readily detectable in real-world scenarios.

C. Adversarial Fault Injection Agent

Although fault mechanisms are defined, determining optimal perturbation amplitudes across state dimensions remains challenging. Indiscriminate random fault injection may cause vehicles to frequently learn responses to minor faults while remaining vulnerable to critical faults. Meanwhile, it is difficult to manually set up valid faults, for the impact of observation deviations on RL-based policies is challenging to analyze theoretically. To address the problem, we introduce a global fault injection agent that adaptively introduces disruptive faults during training.

We impose a constraint on the fault injection agent: it cannot arbitrarily choose which vehicle receives a faulty observation or which neighbor's state is perturbed. Instead, these are randomly selected. As exemplified in Fig. 3 (Top), it is randomly determined that the fault recipient vehicle 1 misobserves the fault target vehicle 4. After selecting fault recipient and target vehicles, indicators presenting the fault information and observations of all vehicles together contain the requisite information for finding the effective perturbations from a system-wide global perspective.

As illustrated in Fig. 3 (Middle), the workflow of the fault injection agent involves three key steps: *i*) aggregating global

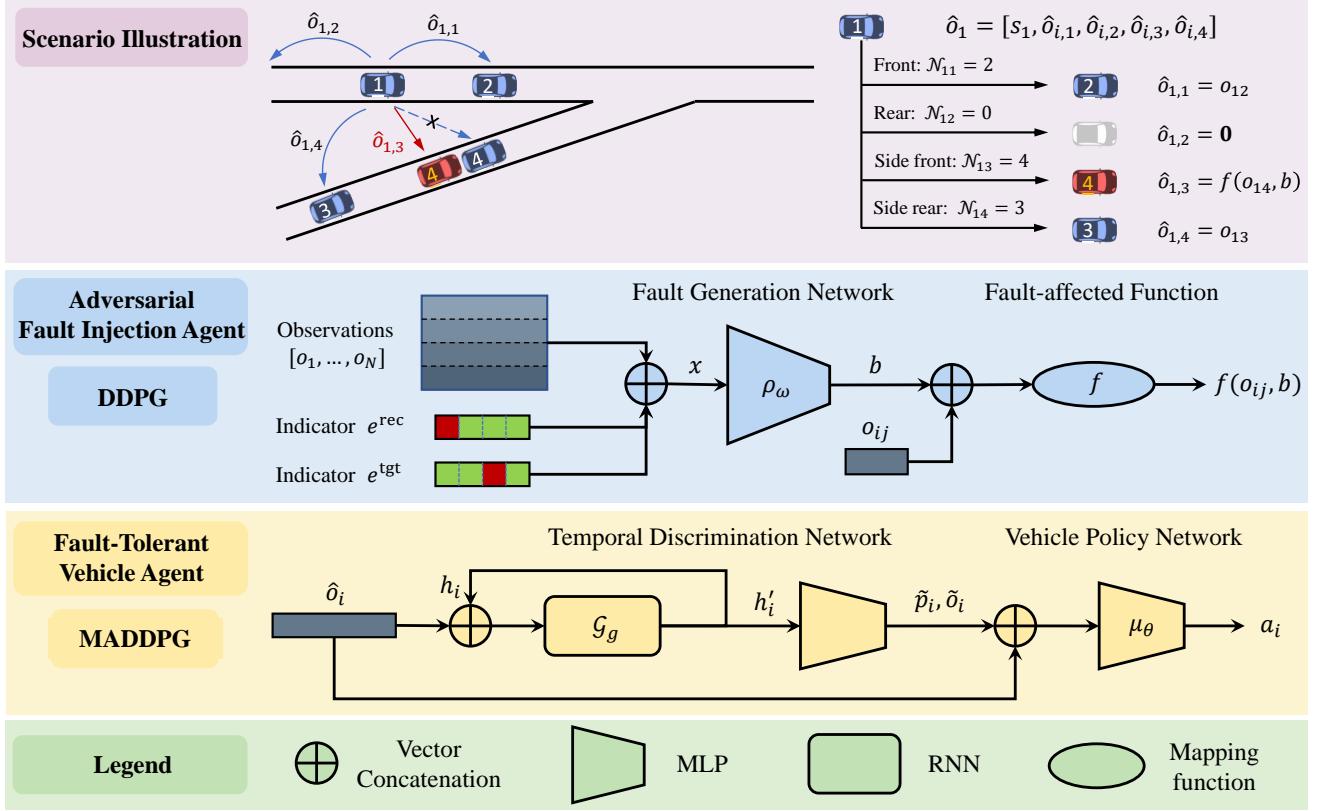


Fig. 3: Schematic diagram of OFT-MARL. Top: Illustration of a on-ramp merging scenario case (left) and the composition of the observation for vehicle 1 within the scenario (right). Middle: Workflow of the adversarial fault injection agent, which takes the observations of all vehicles and two fault indicators as input, processes them through the fault generation network to output perturbation magnitude b , and then transforms the true observation o_{ij} into fault-affected observation $f(o_{ij}, b)$ via the fault-affected function. Bottom: Workflow of the fault-tolerant vehicle agent, which takes potentially perturbed observation \hat{o}_i along with the hidden state h_i from the previous timestep (via the GRU), outputs the hidden state h'_i for the next timestep, further processes it through an MLP network to generate fault predictions \tilde{p}_i and observation estimates \tilde{o}_i , and finally combines these with the observation \hat{o}_i to produce acceleration action a_i through the action output network.

observations and fault indicators to form the input vector x ; *ii*) processing x through the policy network to compute perturbation magnitude b ; and *iii*) using a fault-affected function to apply b to the true observation.

Formally, the input vector $x = [o_1, \dots, o_N, e^{\text{rec}}, e^{\text{tgt}}]$ concatenates: *i*) All vehicle observations o_i ; *ii*) Fault recipient indicator e^{rec} ; *iii*) Fault target indicator e^{tgt} . Here, e^{rec} indicates vehicle i which gets perturbed observations and e^{tgt} indicates vehicle \mathcal{N}_i^j which vehicle i misobserves. Both are one-hot vectors:

$$\begin{aligned} e^{\text{rec}} &= [\delta_{ki}]_{k=1}^N = \begin{cases} 1, & k = i \\ 0, & \text{otherwise} \end{cases}, \\ e^{\text{tgt}} &= [\delta_{kj}]_{k=1}^m = \begin{cases} 1, & k = j \\ 0, & \text{otherwise} \end{cases}, \end{aligned} \quad (10)$$

where δ_{ki} , δ_{kj} are Kronecker delta functions.

When vehicle i misobserves neighbor \mathcal{N}_i^j , the input-output relationship is:

$$b = \rho_\omega(x), \quad (11)$$

where $b \in \mathbb{R}^{d_0}$ with $d_0 \leq d$ denotes perturbations applied to d_0 state dimensions of $o_{i\mathcal{N}_i^j}$. Here ρ represents the fault injection agent's policy network with parameters ω .

In this paper, we adapt a basic add function as the fault-affected function. The $B(o)$ ball is a ∞ -norm ball. Thus,

$$f(o_{ij}, b) = o_{ij} + \text{clip}(b, -\epsilon, \epsilon). \quad (12)$$

Here, $\text{clip}(b, -\epsilon, \epsilon)$ means each dimension of b is restricted in $[-\epsilon, \epsilon]$.

D. Fault-Tolerant Vehicle Agent

Observation perturbations pose a significant threat to vehicle policies, particularly those based on conventional Multi-Layer Perceptron (MLP) architectures. This vulnerability stems from the fundamental memoryless nature of MLPs as feedforward networks. While MLPs are well-suited for idealized fully observable Markovian environments where decisions can be made optimally based on accurate current observations, they become critically inadequate in the presence of faults. Specifically, they lack the capability to detect observational faults or extract diagnostic information from corrupted observations.

This inherent limitation severely compromises the safety and operational efficiency of vehicle policies when observation perturbations occur. Fortunately, vehicular operational data exhibit strong spatio-temporal correlations. For instance, vehicle kinematics are bound by physical constraints, trajectories demonstrate continuity and smoothness, and driving behaviors generally adhere to established traffic norms. These consistent temporal patterns provide a viable means to mitigate perturbations by enabling the discrimination of potential faults and the reconstruction of credible observation estimates.

We therefore introduce a temporal discrimination network to equip the fault-tolerant agent with these capabilities. This network is implemented using a Gated Recurrent Unit (GRU), an advanced version of RNNs that utilizes sophisticated gating mechanisms to regulate information flow over extended sequences. The GRU's hidden state, which contains historical information, precisely suits the temporal characteristics of vehicle data. When abrupt deviations manifest in observation streams, this network identifies compromised observation dimensions and generates relatively reliable estimates. Augmenting observation vectors with these outputs significantly enhances fault tolerance.

As illustrated in Fig. 3 (Bottom), the workflow of the proposed fault-tolerant vehicle agent involves three steps: *i*) updating the hidden state h'_i encoding historical information; *ii*) generating fault probability \tilde{p}_i and the reconstructed observation estimate \tilde{o}_i from h'_i ; *iii*) outputting the vehicle action a_i through the policy network.

For vehicle i , the temporal discrimination network \mathcal{G} with parameters g operates as:

$$\begin{cases} \tilde{p}_i, \tilde{o}_i = \mathcal{G}_g(\hat{o}_i, h_i), \\ h'_i = \sigma(\hat{o}_i, h_i), \\ \tilde{p}_i = \text{sigmoid}(W_p h'_i + b_p), \\ \tilde{o}_i = W_o h'_i + b_o. \end{cases} \quad (13)$$

where h'_i and h_i denotes the hidden states for the next and the current timestep, with $\tilde{p}_i \in [0, 1]^m$ indicating fault probability of each neighbor, and $\tilde{o}_i \in \mathbb{R}^{d_0}$ representing the reconstructed observation. σ denotes the GRU and sigmoid denotes the activation function. W_p, W_o and b_p, b_o represents the weights and biases of the linear layer respectively.

With the help of fault identification and observation prediction, the policy network processes the augmented inputs to output the acceleration action a_i :

$$a_i = \mu_\theta(\hat{o}_i, \tilde{p}_i, \tilde{o}_i), \quad (14)$$

where μ is the policy network with parameters θ shared across all vehicles.

E. Joint Training Framework

The joint training framework is characterized by a co-evolutionary dynamic between the two agents. As illustrated in Fig. 2, the fault injection agent's perturbations directly affect the vehicle observations, which in turn influence both the temporal network's estimations and the policy network's decisions. Conversely, the vehicle policy's resilience against perturbations

shapes the fault injection agent's learning objective. This co-evolutionary process creates an offensive-defensive synergy where each agent continuously adapts to the other's strategies, ultimately leading to fault-tolerant policy learning.

This synergy of the two agents is implemented through three interconnected network components that undergo joint optimization using distinct learning paradigms, concluded in Table I. Within the vehicle agent, the temporal discrimination network \mathcal{G}_g is trained via supervised learning, while the actor-critic architecture, comprising the policy network μ_θ (actor) and critic networks $\{Q_i^{\psi_i}\}_{i=1}^N$, is optimized using the MADDPG framework. The fault injection agent employs its own actor-critic scheme, where the actor network ρ_ω generates adversarial perturbations and the critic network Q^ϕ evaluates their disruptive impact, both trained with DDPG.

TABLE I: Learning paradigms

| Networks | Paradigm | Reason |
|----------------------------|---------------------|---------------------------------|
| \mathcal{G}_g | Supervised learning | explicit supervision signals |
| $Q_i^{\psi_i}, \mu_\theta$ | MADDPG | multi-agent cooperative policy |
| Q^ϕ, ρ_ω | DDPG | single-agent adversarial policy |

The temporal network \mathcal{G}_g is co-trained with reinforcement learning using supervised signals. We minimize the composite mean squared error (MSE) between its outputs and ground-truth references:

$$\mathcal{L}(g) = \mathbb{E}_{(o, \hat{o}, h) \sim \mathcal{D}_v} \left[\|\tilde{p}_i - p_i\|_2^2 + \sum_{j=1}^m \|\tilde{o}_{ij} - o_{ij}\|_2^2 \right], \quad (15)$$

where $p_i \in \{0, 1\}^m$ denotes the ground-truth fault indicator vector.

Each vehicle possesses a dedicated critic $Q_i^{\psi_i}$ that receives ground-truth global states $o = (o_1, \dots, o_N)$ to ensure stability:

$$\begin{aligned} \mathcal{L}(\psi_i) &= \mathbb{E}_{(o, a, r, o', h') \sim \mathcal{D}_v} \left[\left(Q_i^{\psi_i}(o, a) - y_i \right)^2 \right], \\ y_i &= r_i + \gamma Q_i^{\psi'_i}(o', a'_1, \dots, a'_N), \\ a'_j &= \mu_{\theta'}(\hat{o}'_j, \tilde{p}'_i, \tilde{o}'_i), \end{aligned} \quad (16)$$

where ψ'_i and θ' denote target network parameters. It is worth noting that the fault injection agent ρ_ω is incorporated in order to compute perturbed next observation \hat{o}'_j . The policy gradient update follows:

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{(o, \hat{o}, a, h) \sim \mathcal{D}_v} \left[\nabla_\theta \mu_\theta(\hat{o}_i, \tilde{p}_i, \tilde{o}_i) \nabla_{a_i} Q_i^{\psi_i}(o, a) \right], \\ a_i &= \mu_\theta(\hat{o}_i, \tilde{p}_i, \tilde{o}_i). \end{aligned} \quad (17)$$

The fault injection agent's objective is to maximize policy disruption by minimizing collective rewards:

$$r' = - \sum_{i=1}^N r_i. \quad (18)$$

The fault injection agent possesses a critic network Q with parameters ϕ , updated to minimize the temporal difference loss:

$$\begin{aligned} \mathcal{L}(\phi) &= \mathbb{E}_{(x, b, r', x') \sim \mathcal{D}_f} \left[\left(Q^\phi(x, b) - y \right)^2 \right], \\ y &= r' + \gamma Q^{\phi'}(x', b')|_{b'=\rho_\omega'(x')}. \end{aligned} \quad (19)$$

Algorithm 1 OFT-MARL

Require: Batch size k , replay buffers \mathcal{D}_v (vehicle agent) and \mathcal{D}_f (fault injection agent), number of vehicles N , maximum training episodes E , maximum steps per episode S , network update frequency K

- 1: Initialize step counter $\text{Total_steps} \leftarrow 0$. Randomly initialize the vehicle agent critic network $Q_i^{\psi_i}$, the vehicle agent actor network μ_θ , the fault injection critic network Q^ϕ , the fault injection actor network ρ_ω , and the temporal discrimination network \mathcal{G}_g .
- 2: **for** episode $e = 1$ to E **do**
- 3: Initialize environment state, fault configuration, and GRU hidden states for all vehicles
- 4: **for** step $t = 1$ to S **do**
- 5: **if** fault condition is triggered **then**
- 6: Compute perturbation: $b \leftarrow \rho_\omega(x)$
- 7: **else**
- 8: $b \leftarrow 0$ (no perturbation)
- 9: **end if**
- 10: Generate perturbed observations \hat{o}_i using Equ. (7), (8) and (12)
- 11: Compute temporal features: $(\tilde{p}_i, \tilde{o}_i, h'_i) \leftarrow \mathcal{G}_g(\hat{o}_i, h_i)$ via Equ. (13)
- 12: Select actions: $a_i \leftarrow \mu_\theta(\hat{o}_i, \tilde{p}_i, \tilde{o}_i)$ for each vehicle i
- 13: Execute actions, observe next state s' and rewards r_i
- 14: Store transition (x, b, r', x') in \mathcal{D}_f if fault has occurred
- 15: Store vehicle experiences $(o, \hat{o}, a, r, o', h, h')$ in \mathcal{D}_v
- 16: $\text{Total_steps} \leftarrow \text{Total_steps} + 1$
- 17: **if** $\text{Total_steps} \bmod K = 0$ **then**
- 18: **for** vehicle $i = 1$ to N **do**
- 19: Sample mini-batch of k transitions from \mathcal{D}_v
- 20: Update temporal network \mathcal{G}_g via Equ. (15)
- 21: Update critic $Q_i^{\psi_i}$ via Equ. (16)
- 22: Update actor μ_θ via Equ. (17)
- 23: **end for**
- 24: Sample mini-batch of k transitions from \mathcal{D}_f
- 25: Update fault injection critic Q^ϕ via Equ. (19)
- 26: Update fault injection actor ρ_ω via Equ. (20)
- 27: Update target networks parameters $\psi'_i, \theta', \phi', \omega'$
- 28: **end if**
- 29: Advance to next state: $o \leftarrow o', h \leftarrow h'$
- 30: **end for**
- 31: **end for**

where ϕ' and ω' denote target network parameters. The policy gradient update is:

$$\nabla_\omega J(\omega) = \mathbb{E}_{x \sim \mathcal{D}_f} \left[\nabla_\omega \rho_\omega(x) \nabla_b Q^\phi(x, b) \Big|_{b=\rho_\omega(x)} \right]. \quad (20)$$

The complete algorithm is described in Algorithm 1.

V. EXPERIMENTS

A. Setup

1) *Experiments Overview:* This section conducts experiments to validate the effectiveness of the proposed methodology. We evaluate the contribution of each module by comparing the performance across different experimental configurations, as summarized in Table II.

In Subsection V-B, we justify the necessity of accounting for potential faults. Policies are trained using vanilla MADDPG in fault-free scenarios, and are then evaluated under both fault-free and random fault conditions, demonstrating the importance of enhancing fault tolerance.

TABLE II: Experimental configurations in two dimensions

| Dimension | Configurations | Description |
|-----------------|-------------------|----------------------------------|
| Fault Injection | Fault-free | $b = 0$ |
| | Random fault | $b \sim U(-\epsilon, \epsilon)$ |
| | Adversarial fault | $b = \rho_\omega(x)$ |
| Policy Training | Vanilla MADDPG | Baseline without fault tolerance |
| | OFT-MARL w/o GRU | Proposed method without GRU |
| | OFT-MARL | Full proposed method |

In Subsection V-C, we introduce random faults during training for all three policy methods. Subsequently, a fault injection agent is trained against each fixed pre-trained policy. By comparing performance under random faults versus faults from the fault injection agent during testing, we validate the specific role of the proposed fault injection agent.

In Subsection V-D, all three policy methods are trained with the proposed fault injection agent generating adversarial faults. Policies are evaluated to verify the effectiveness of the overall training architecture and the fault-tolerant vehicle agent.

Furthermore, policies are tested under diverse fault patterns to assess the generalization capability.

In Subsection V-E, we perform an isolated analysis of the outputs from the temporal network to further confirm its contribution.

2) *Scenario*: We design a scenario modified from the Highway Environment [35]. As depicted in Fig. 3 Top, the highway merges consist of single-lane main road and on-ramp with $N = 4$ vehicles. Each vehicle i observes $m = 4$ neighboring vehicles:

$$\mathcal{N}_i^1 : \text{front vehicle}$$

$$\mathcal{N}_i^2 : \text{rear vehicle}$$

$$\mathcal{N}_i^3 : \text{side front vehicle}$$

$$\mathcal{N}_i^4 : \text{side rear vehicle}$$

Thus vehicle i 's observation vector $o_i \in \mathbb{R}^{(4+1) \times 4}$ concatenates its own state and relative observations of these neighbors:

$$o_i = [\underbrace{s_i}_{\text{ego}}, \underbrace{o_i \mathcal{N}_i^1}_{\text{front}}, \underbrace{o_i \mathcal{N}_i^2}_{\text{rear}}, \underbrace{o_i \mathcal{N}_i^3}_{\text{side front}}, \underbrace{o_i \mathcal{N}_i^4}_{\text{side rear}}]. \quad (21)$$

State dimensions $d = 4$ comprise: (a) existence Boolean, (b) longitudinal position, (c) velocity, and (d) lane ID. Neighbor observations use relative values for position, velocity and lane. Since the existence and lane features are considered unperturbable or difficult to perturb, only the relative position and velocity can be perturbed, which means $d_0 = 2$.

Vehicles collaborate to achieve safe and efficient merging, with success defined as all vehicles passing a designated point. The reward function combines:

$$r = -1 + r_{\text{goal}} + r_{\text{vel}} + r_{\text{col}}, \quad (22)$$

where -1 prevents vehicles from learning overly conservative local optimal policies, r_{goal} is a time-discounted task completion bonus, r_{vel} is the reward for maintaining appropriate velocity, and r_{col} is the penalty for vehicle collision.

In order to better present the adverse effects of the faults, we increase the randomness of the initial velocities, set strict vehicle headway, and endow vehicles with heterogeneous acceleration characteristics. This prevents trivialization of decision-making tasks and ensures the necessity of the fault tolerance of the vehicle policy.

B. Necessity of Fault tolerance

We first train vehicle policies using vanilla MADDPG framework without considering potential faults. The training performance is illustrated in Fig. 4 (a). During evaluation, we test policies trained with 6 different random seeds under two conditions: without faults and with random faults. Results in Table 4 (b) demonstrate significant performance degradation when random faults are introduced.

As exemplified in the test scenario of Fig. 5, the true positions of the vehicles are depicted in green. After introducing random faults, at timestep 9 (Fig. 5 (a)), vehicle 1 receives corrupted observation regarding vehicle 4, falsely perceiving its position at the red dashed bounding box. This perception error leads to failure in recognizing the imminent collision risk

and consequently no deceleration is conducted. The collision ultimately occurs at timestep 11 (Fig. 5 (b)), indicated by the blue highlight.

This experiment substantiates that training processes must account for potential observational deviations, as even random perturbations can critically compromise performance.

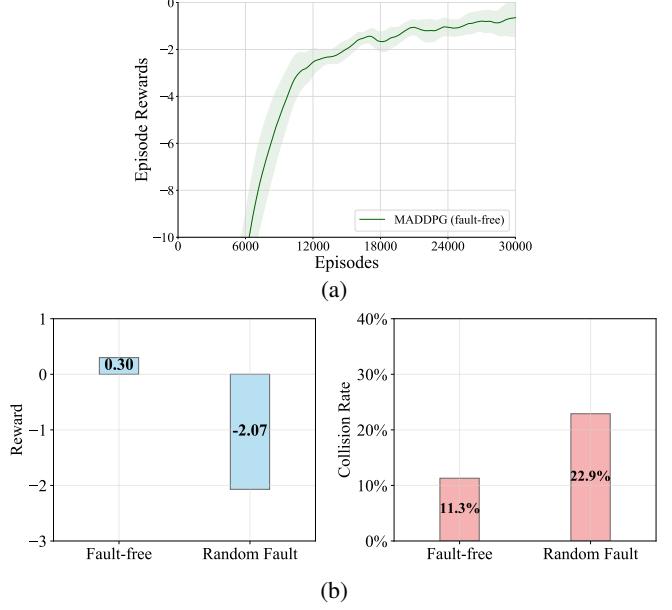


Fig. 4: (a) Average reward curves for MADDPG in fault-free scenarios. The error bar is a 95% confidence interval across 6 runs. (b) Performance decline of methods trained in fault-free scenarios and tested under fault-free and random fault conditions.

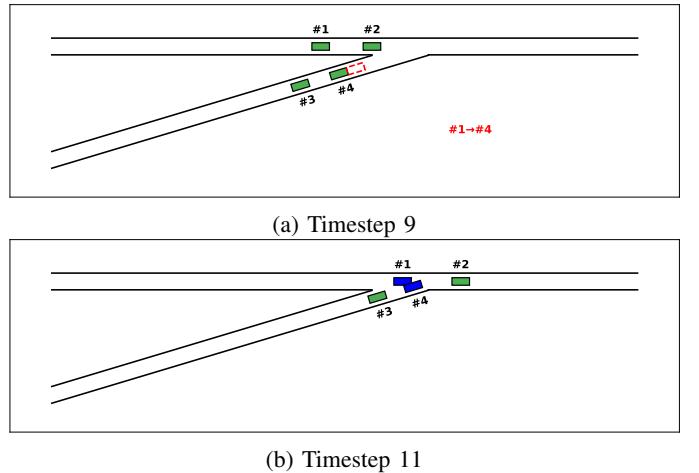


Fig. 5: Influence of random faults.

C. Effect of the Fault Injection Agent

To validate the efficacy of the proposed fault injection agent, it is necessary to demonstrate that introducing merely random faults during training is insufficient. Hence, we first train the policy with random faults and keep it fixed. Then, we introduce

the fault injection agent and proceed to train this agent. As illustrated in Fig. 6, the progressively decreasing reward curve during training indicates that the negative impact of faults on the vehicle policy is gradually increasing. The testing results, summarized in Table III, show that while the collision rate changes marginally, the task completion timesteps increase significantly. This phenomenon occurs because the reduction in rewards can be attributed to two primary factors: causing vehicle collisions or impairing traffic efficiency. When the fault injection agent targets traffic efficiency delays, the safety metric may conversely improve. Consequently, the overall collision rate remains relatively unchanged.

A specific scenario is analyzed, in which vehicle 4 has perturbed observation when perceiving vehicle 2. In the first case, shown in Fig. 7 (a) and (b), at timestep 9, the fault injection agent biases vehicle 4’s observation of vehicle 2’s position forward and its speed higher. This leads vehicle 4 to perceive a much larger time headway than reality, ultimately resulting in a collision between vehicle 2 and vehicle 4 at timestep 11. In the second case, depicted in Fig. 7 (c) and (d), also at timestep 9, the fault injection agent acts conversely—biasing the observed position backward and the speed lower. Consequently, vehicle 4 decelerates to avoid a potential collision, triggering a chain reaction where vehicles 1 and 3 also brake successively. By timestep 15, the last vehicle, vehicle 3, finally passes the merging point, but is already significantly behind vehicle 2, indicating markedly reduced traffic efficiency.

The experiments in this section demonstrate that policies trained solely with random faults possess inadequate fault tolerance. In contrast, the fault injection agent can strategically select the most disruptive observation perturbation based on the specific scenario context, thereby effectively compromising either safety or efficiency.

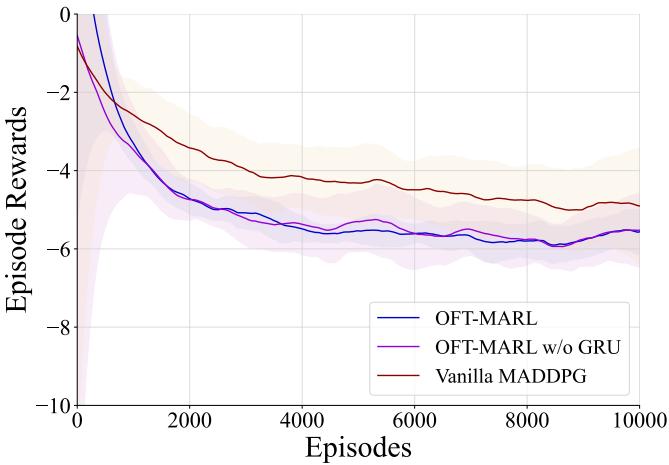


Fig. 6: Average reward curves for training the fault injection agent with different fixed vehicle policies. The decline in vehicle rewards demonstrates the increasing effectiveness of the adversarial agent. Error bars indicate 95% confidence interval across 6 runs.

TABLE III: Performance decline of methods trained with random faults: testing under random vs. adversarial faults

| Method | Fault Condition | Reward | Collision Rate | Timesteps |
|---------------------|-----------------|--------|----------------|-----------|
| Vanilla MADDPG | Random | -1.26 | 21.2% | 17.39 |
| | Adversarial | -4.91 | 18.4% | 18.45 |
| | △ | 3.65↓ | 2.8%↓ | 1.06↑ |
| OFT-MARL w/o GRU | Random | -0.49 | 14.4% | 17.55 |
| | Adversarial | -5.56 | 10.6% | 18.90 |
| | △ | 5.07↓ | 3.8%↓ | 1.35↑ |
| OFT-MARL | Random | -0.14 | 12.9% | 17.54 |
| | Adversarial | -5.72 | 13.3% | 18.86 |
| | △ | 5.58↓ | 0.4%↑ | 1.32↑ |

△: Performance variation between two fault conditions

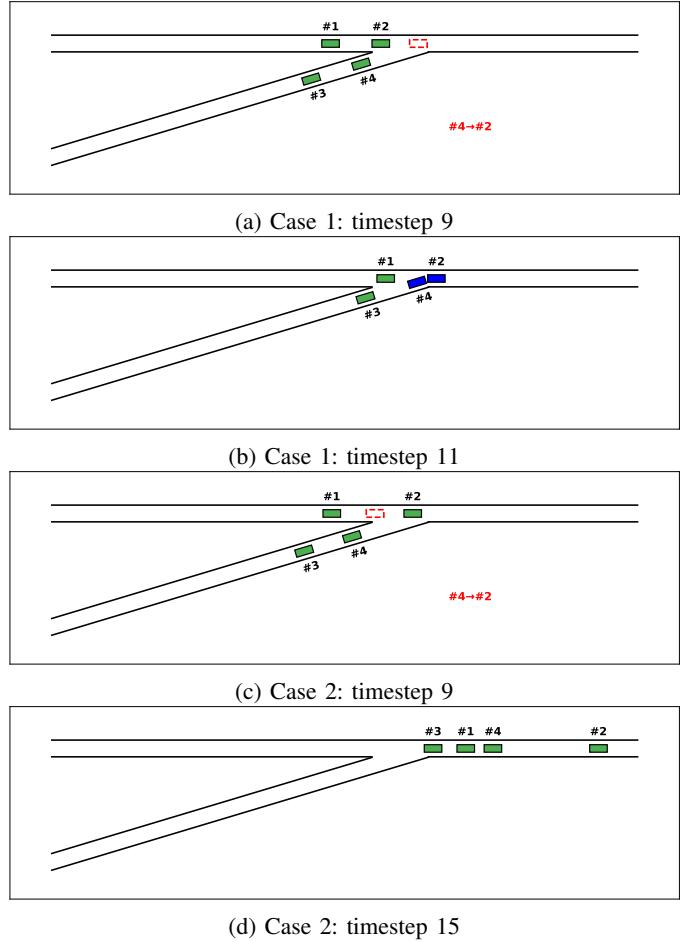


Fig. 7: Influence of adversarial faults introduced by the fault injection agent on fleet behavior. (a)-(b) Case 1: Safety compromise. (c)-(d) Case 2: Efficiency degradation.

D. Training of Fault-tolerant Policies

Having verified the efficacy of the fault injection agent, we employ it to generate adversarial faults during training, concurrently training both the agent and the vehicle policies. Under identical fault injection conditions, we compare the performance of different methods. As shown in Fig. 8, the standard MADDPG achieves the highest reward in the absence of faults. In contrast, vanilla MADDPG yields the poorest reward, as it fails to account for the impact of observation

deviations on the accuracy of both critics and actors during training. OFT-MARL w/o GRU builds upon vanilla MADDPG by adopting advanced training strategy of critics and actors according to Alg. 1. This modification leads to a substantial improvement in the performance of the trained vehicle policies. Furthermore, the proposed OFT-MARL introduces a temporal network that assists the policy network in detecting faults and estimating true observations from corrupted observations by leveraging historical information. This architecture further enhances the performance of OFT-MARL, enabling it to approach the performance of standard MADDPG under fault-free conditions.

As indicated by the test results in Table IV, OFT-MARL outperforms other methods across all metrics, and its performance is close to that achieved in the fault-free scenario. These results demonstrate that the proposed method effectively mitigates the harmful effects of observation perturbations. Even when the fault injection agent deliberately introduces adversarial faults, OFT-MARL maintains performance levels comparable to those under fault-free conditions.

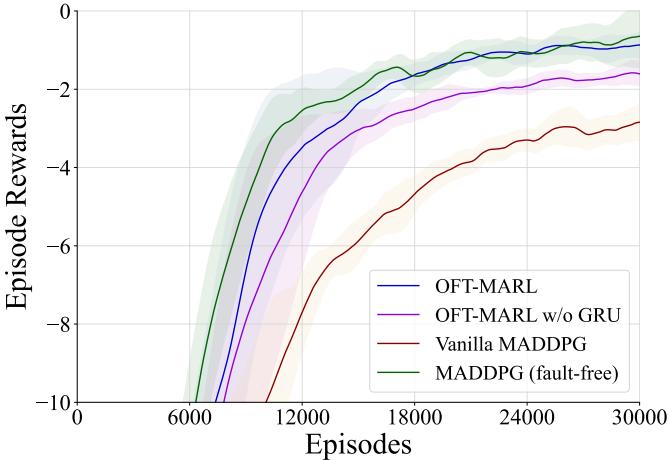


Fig. 8: Average reward curves for 3 methods trained with the fault injection agent and for MADDPG without considering faults. Error bars indicate 95% confidence interval across 6 runs.

TABLE IV: Performance comparison of methods trained with adversarial faults

| Method | Reward | Collision Rate | Timesteps |
|---------------------|--------|----------------|-----------|
| MADDPG (fault-free) | 0.30 | 11.3% | 17.52 |
| Vanilla MADDPG | -2.36 | 22.9% | 17.61 |
| OFT-MARL w/o GRU | -1.07 | 15.1% | 17.65 |
| OFT-MARL | -0.38 | 12.5% | 17.60 |

To evaluate the generalization capability of each method under diverse fault conditions, we select the best-performing policy from 6 random seeds for each approach and test it against multiple fault injection strategies. The fault injection methodologies encompassed: (1) fault-free conditions, (2) random faults, (3) faults generated by fault injection agents trained with the three vehicle policy training methods, and (4)

faults generated by retrained fault injection agents with fixed vehicle policies.

As demonstrated in Table V, the proposed OFT-MARL maintains fault-tolerant performance across various fault patterns, demonstrating generalization capability beyond its training-specific fault pattern. Notably, even when confronted with adversarially retrained fault injection policies targeting the fixed vehicle policy, the performance degradation remains marginal. The observed performance reduction is substantially smaller compared to the results presented in Table III. These findings collectively indicate that OFT-MARL exhibits considerable resilience against diverse fault patterns.

E. Analysis of Vehicle Agent's Fault-Diagnosis Capability

To gain deeper insights into how the fault-tolerant mechanism within our vehicle agent contributes to enhancing fault tolerance, we conduct an independent analysis of the temporal network's fault-diagnosis outputs. First, we examine the accuracy of the temporal network in fault detection. Since the network outputs probability values and the dataset exhibits class imbalance, we binarize the predictions using a threshold of 0.5 to construct the confusion matrix. As shown in Table VI, based on 69456 judgments collected over 1000 episodes, the network successfully identifies fault states 9296 times and correctly classifies normal states 59711 times. There are 412 instances where fault states are misclassified as normal, and 37 instances where normal states are falsely identified as faulty. This results in an accuracy of 99.3%, a precision of 99.6%, and a recall of 95.8%.

Subsequently, we evaluate the accuracy of the temporal network in predicting true observations from corrupted observations. The results presented in Table VII demonstrate that the network can effectively recover significant original observation errors to an acceptable level of prediction loss. For instance, considering the mean absolute error (MAE) of position estimation, the fault injection agent introduces an average position deviation of 9.10 m between observed and true values. The temporal network reduces this error to 3.36 m, corresponding to a 63.1% error correction. Recovery rates also exceed 50% among other metrics. The high accuracy in fault detection and the significant error reduction in observation estimation directly explain why the vehicle agent maintains high performance under faults. By providing reliable fault alerts and corrected state references, the temporal network empowers the policy network to make safer and more efficient decisions.

To exhibit the performance of fault-tolerant mechanism under different fault patterns, we examine two representative cases as shown in Fig. 9.

Case 1 investigates the network's response to abrupt state changes. At timestep 4, vehicle 2 observes a sudden forward position shift of 6.0 m in vehicle 3. The network immediately detects this anomaly with high probability and generates a predicted position 0.8 m ahead of the actual position, indicated by the purple rectangle.

Case 2 examines the network's capability in handling gradual state deviations. Starting from timestep 4, vehicle 2's observations of vehicle 3 exhibit progressively increasing

TABLE V: Generalization performance evaluation under various fault injection conditions.

| Test Condition | OFT-MARL | | | OFT-MARL w/o GRU | | | Vanilla MADDPG | | |
|------------------------|--------------|----------------|-----------|------------------|----------------|-----------|----------------|----------------|-----------|
| | Reward | Collision Rate | Timesteps | Reward | Collision Rate | Timesteps | Reward | Collision Rate | Timesteps |
| Fault-free | 1.12 | 2.4% | 17.67 | 0.29 | 10.1% | 17.51 | -0.12 | 13.1% | 17.47 |
| Random fault | -0.63 | 10.4% | 17.73 | -1.34 | 19.0% | 17.52 | -1.44 | 19.6% | 17.49 |
| OFT-MARL fault | 0.17 | 6.8% | 17.70 | -0.71 | 14.6% | 17.57 | -1.38 | 18.5% | 17.53 |
| OFT-MARL w/o GRU fault | -1.14 | 8.7% | 17.90 | -0.88 | 13.2% | 17.67 | -2.12 | 15.9% | 17.83 |
| Vanilla MADDPG fault | -0.32 | 8.2% | 17.74 | -1.33 | 17.8% | 17.58 | -1.07 | 17.6% | 17.51 |
| Retrained fault | -1.63 | 10.1% | 17.97 | -2.32 | 15.2% | 17.93 | -5.21 | 21.6% | 18.36 |

TABLE VI: Confusion matrix of fault detection results

| | | Predicted | | Total |
|--------|--------|-----------|--------|-------|
| | | Fault | Normal | |
| Actual | Fault | 9296 | 412 | 9708 |
| | Normal | 37 | 59711 | 59748 |
| Total | | 9333 | 60123 | 69456 |

TABLE VII: Observation Prediction Recovery Percentage

| | Position (m) | | Velocity (m/s) | |
|------------|--------------|-------|----------------|-------|
| | MAE | MSE | MAE | MSE |
| Original | 9.10 | 86.68 | 3.46 | 12.83 |
| Prediction | 3.36 | 19.22 | 1.45 | 3.22 |
| Recovery | 63.1% | 77.8% | 58.1% | 74.9% |

errors: position deviation increases by 1.0 m per timestep and velocity deviation by 0.40 m/s per timestep. Although the observation prediction remains accurate at timestep 4, the network fails to initially identify the fault due to minor deviations. However, as vehicle 3 continues to accelerate in vehicle 2's view, which contradicts the normal driving pattern, the temporal network gradually increases its fault probability assessment. At timestep 7, vehicle 2 successfully identifies the fault. The observed position and velocity contain a 4.0 m and 1.60 m/s forward error respectively. The network's predictions substantially correct these deviations, reducing the position error to 1.4 m and the velocity error to 0.36 m/s.

In both scenarios shown in panels (b) and (d), without the diagnostic capability, vehicle 2 would misinterpret vehicle 3's behavior as aggressive positioning and potentially initiate unnecessary deceleration. This could either cause collisions with actually positioned vehicles or significantly increase traffic delay. Equipped with our fault-tolerant design, vehicle 2 correctly maintains speed and safely precedes vehicle 3 through the conflict point.

VI. CONCLUSION

This paper highlights the severe impact of observation perturbations on the safety and efficiency of MARL-based cooperative driving systems. To address this, we introduce a fault-tolerant MARL framework integrating an adaptive fault injection agent and a fault-tolerant vehicle agent. The fault injection agent actively generates challenging perturbations during training, exposing agents to critical faults and enhancing

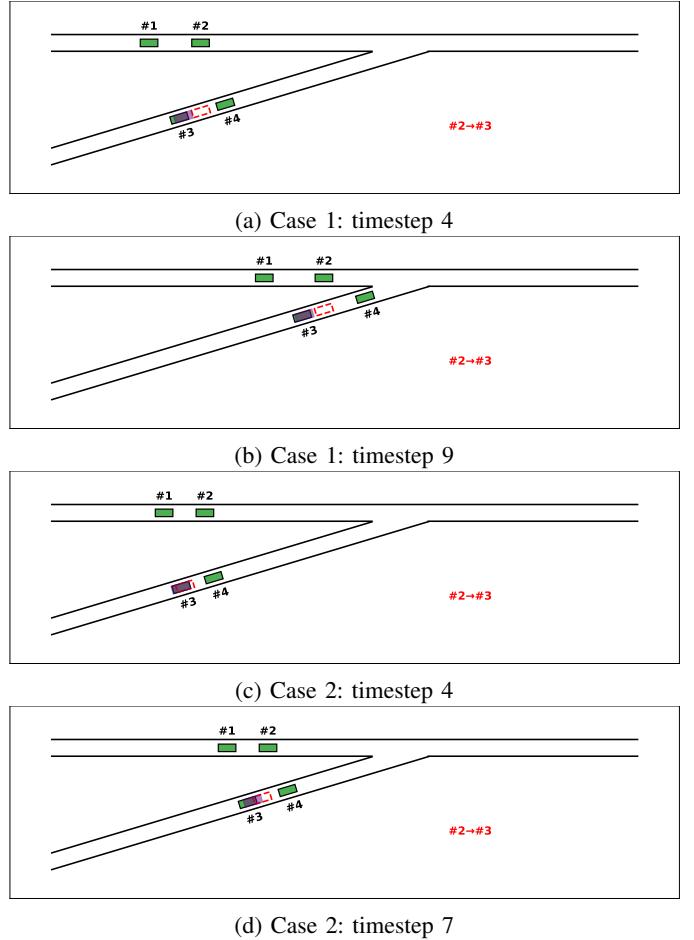


Fig. 9: Exhibition of the vehicle's fault-diagnosis capability in mitigating different fault patterns. (a)-(b) Case 1: Response to an abrupt position shift. (c)-(d) Case 2: Response to progressive deviations.

fault tolerance. The vehicle agent is equipped with the capability to detect faults and reconstruct credible observation estimates by leveraging spatio-temporal vehicle data. Experimental results in highway on-ramp merging scenarios confirm that our approach significantly outperforms vanilla MADDPG and ablated variants under both random and adversarial fault conditions. It nearly matches the performance of fault-free systems while maintaining strong generalization across unseen fault patterns. The temporal network achieves high fault detection accuracy and substantially corrects observation errors.

In future research, the applicability of the method across

diverse scenarios can be investigated. Moreover, it can be integrated with safety verification modules to ensure the security of vehicle policies. Additionally, more data credibility assessment methods, such as incorporating driving habits factors, could be fused into the vehicle agent to enhance fault detection and observation prediction capabilities for heterogeneous vehicles.

REFERENCES

- [1] H. Xu, S. Feng, Y. Zhang, and L. Li, "A grouping-based cooperative driving strategy for cavs merging problems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6125–6136, 2019.
- [2] H. Pei, Y. Zhang, Q. Tao, S. Feng, and L. Li, "Distributed cooperative driving in multi-intersection road networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 6, pp. 5390–5403, 2021.
- [3] H. Pei, Y. Zhang, Y. Zhang, and S. Feng, "Optimal cooperative driving at signal-free intersections with polynomial-time complexity," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12 908–12 920, 2022.
- [4] H. Yu, W. Yang, J. Zhong, Z. Yang, S. Fan, P. Luo, and Z. Nie, "End-to-end autonomous driving through v2x cooperation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 9, pp. 9598–9606, Apr. 2025.
- [5] H. Domajko, "A review of fault types and analytical and hardware redundancy for the sensor functional safety," *Electrotechnical Review/Elekrotehniski Vestnik*, vol. 91, 2024.
- [6] W. Shi, M. B. Alawieh, X. Li, H. Yu, N. Arechiga, and N. Tomatsu, "Efficient statistical validation of machine learning systems for autonomous driving," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–8.
- [7] S. Wang, "An analytical framework for modeling and synthesizing malicious attacks on adaptive cruise control vehicles," *Transportation Research Record*, vol. 2679, no. 5, pp. 142–156, 2023.
- [8] Y. Feng, S. Huang, Q. A. Chen, H. X. Liu, and Z. M. Mao, "Vulnerability of traffic control system under cyberattacks with falsified data," *Transportation research record*, vol. 2672, no. 1, pp. 1–11, 2018.
- [9] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2022.
- [10] E. M. Mianji, M. Fardad, G.-M. Muntean, and I. Tal, "A survey on multi-agent reinforcement learning applications in the internet of vehicles," in *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*, 2024, pp. 1–7.
- [11] Y. Lv, J. Lei, and P. Yi, "A local information aggregation-based multiagent reinforcement learning for robot swarm dynamic task allocation," *IEEE Transactions on Neural Networks and Learning Systems*, 2025.
- [12] L. Canese, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, and S. Spanò, "Multi-agent reinforcement learning: A review of challenges and applications," *Applied Sciences*, vol. 11, no. 11, 2021.
- [13] Y. Shi, H. Pei, L. Feng, Y. Zhang, and D. Yao, "Toward fault tolerance in multi-agent reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 19 007–19 024, 2025.
- [14] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," in *International Conference on Learning Representations*, 2020.
- [15] H. Pei, S. Feng, Y. Zhang, and D. Yao, "A cooperative driving strategy for merging at on-ramps based on dynamic programming," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 12, pp. 11 646–11 656, 2019.
- [16] K. Long, Z. Gao, Z. Jiang, C. Ma, J. Hu, and X. Yang, "Optimization based trajectory planner for multilane roundabouts with connected automation," *Journal of Intelligent Transportation Systems*, vol. 27, no. 3, pp. 411–422, 2023.
- [17] M. Zhang, C. Wang, W. Zhao, J. Liu, and Z. Zhang, "A multi-vehicle self-organized cooperative control strategy for platoon formation in connected environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 3, pp. 4002–4018, 2025.
- [18] Q. Wang, D. Tian, X. Duan, G. Qi, J. Zhou, and D. Zhao, "Efficient and energy-saving cooperative motion planning for multiple connected and autonomous vehicles at unsignalized intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 6, pp. 7921–7938, 2025.
- [19] D. Chen, M. R. Hajidavalloo, Z. Li, K. Chen, Y. Wang, L. Jiang, and Y. Wang, "Deep multi-agent reinforcement learning for highway on-ramp merging in mixed traffic," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, pp. 11 623–11 638, 2023.
- [20] J. Pan, T. Wang, C. Claudel, and J. Shi, "Trust-marl: Trust-based multi-agent reinforcement learning framework for cooperative on-ramp merging control in heterogeneous traffic flow," *arXiv preprint arXiv:2506.12600*, 2025.
- [21] S. Chen, J. Dong, P. Ha, Y. Li, and S. Labi, "Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles," *Computer-Aided Civil and Infrastructure Engineering*, vol. 36, no. 7, pp. 838–857, 2021.
- [22] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6382–6393.
- [23] H. Pei, J. Zhang, Y. Zhang, X. Pei, S. Feng, and L. Li, "Fault-tolerant cooperative driving at signal-free intersections," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 121–134, 2023.
- [24] Z. He, J. Zhang, H. Pei, L. Feng, and D. Yao, "Communication fault-tolerant cooperative driving at on-ramps: A global planning and local gaming strategy," in *2024 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2024, pp. 1165–1170.
- [25] Q. Liu, J. Zhang, W. Zhong, Z. Li, X. J. Ban, S. Li, and L. Li, "Fault-tolerant cooperative driving at highway on-ramps considering communication failure," *Transportation research part C: emerging technologies*, vol. 153, p. 104227, 2023.
- [26] M. Ma and Z. Li, "A time-independent trajectory optimization approach for connected and autonomous vehicles under reservation-based intersection control," *Transportation Research Interdisciplinary Perspectives*, vol. 9, p. 100312, 2021.
- [27] X. He, H. Yang, Z. Hu, and C. Lv, "Robust lane change decision making for autonomous vehicles: An observation adversarial reinforcement learning approach," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 184–193, 2023.
- [28] X. He, B. Lou, H. Yang, and C. Lv, "Robust decision making for autonomous vehicles at highway on-ramps: A constrained adversarial reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 4, pp. 4103–4113, 2023.
- [29] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*, 1994, pp. 157–163.
- [30] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.
- [31] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, D. Silver, Y. Tassa, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations*, 2016.
- [32] S. Han, S. Zhou, J. Wang, L. Pepin, C. Ding, J. Fu, and F. Miao, "A multi-agent reinforcement learning approach for safe and efficient behavior planning of connected autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 5, pp. 3654–3670, 2024.
- [33] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," *Advances in neural information processing systems*, vol. 33, pp. 21 024–21 037, 2020.
- [34] S. He, S. Han, S. Su, S. Han, S. Zou, and F. Miao, "Robust multi-agent reinforcement learning with state uncertainty," *Transactions on Machine Learning Research*, 2023.
- [35] E. Leurent, "An environment for autonomous driving decision-making," <https://github.com/eleurent/highway-env>, 2018.