Resilient Path Tracking of Autonomous Driving under Few-shot Action Space Attacks

YUTING WU, Nanyang Technological University, Singapore
XIN LOU, Singapore Institute of Technology, Singapore
PENGFEI ZHOU, University of Pittsburgh, USA
RUI TAN, Nanyang Technological University, Singapore
ZBIGNIEW T. KALBARCZYK, University of Illinois at Urbana-Champaign, USA
RAVISHANKAR K. IYER, University of Illinois at Urbana-Champaign, USA

Modern autonomous vehicles face growing cybersecurity risks, especially from action space attacks that directly target vehicle actuators. This paper systematically evaluates the resilience of three representative autonomous driving (AD) architectures, including modular, end-to-end, and feature-fused agents, against few-shot action space attacks crafted via deep reinforcement learning under a black-box setting. The adversary perturbs the vehicle's lateral control only during safety-critical moments, using either a camera or an inertial measurement unit. Our results reveal distinct vulnerabilities and behavioral patterns across AD architectures, which underscore the necessity for adaptive and robust defense strategies. However, existing adversarial training defense methods show limitations of overfitting and reliance on attack knowledge. To address these limitations, we propose a learning-based Path Correction System (PCS) that integrates traditional feedback control with an adversarially trained correction loop. The correction loop is selectively activated by a kinematic-model-based attack detector to counteract abnormal control deviations. Evaluation experiments show that PCS reduces path-tracking deviation by 78% when the system is under attack.

CCS Concepts: • **Computer systems organization → Sensors and actuators**; *Reliability*.

Additional Key Words and Phrases: Action Space Attack, Autonomous Driving, Deep Reinforcement Learning, Cybersecurity

ACM Reference Format:
Yuting Wu, Xin Lou, Pengfei Zhou, Rui Tan, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. 2025. Resilient Path Tracking of Autonomous Driving under Few-shot Action Space Attacks. In *Proceedings of TCPS (Conference acronym 'XX)*. ACM, New York, NY, USA, 28 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The rapid advancement of autonomous driving (AD) technology in recent years has drawn significant research focus to its cybersecurity concerns. While prior work has extensively studied sensor-level attacks [2, 3, 6, 25], actuator compromise has emerged as a critical threat vector. By gaining control over actuator units, attackers may circumvent upstream defenses from perception to

Authors' Contact Information: Yuting Wu, Nanyang Technological University, Singapore, yuting.wu@ntu.edu.sg; Xin Lou, Singapore Institute of Technology, Singapore; Pengfei Zhou, University of Pittsburgh, Pittsburgh, Pennsylvania, USA; Rui Tan, Nanyang Technological University, Singapore; Zbigniew T. Kalbarczyk, University of Illinois at Urbana-Champaign, Urbana-Champaign, Illinois, USA; Ravishankar K. Iyer, University of Illinois at Urbana-Champaign, Urbana-Champaign, Illinois, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

control and have a direct impact on the vehicle's physical state. These actuator-level threats, known as *action space attacks*, pose serious risks to system safety. Although recent studies have revealed the vulnerability of individual AD agents to such attacks [19, 24, 47], systematic comparisons across architectural designs remain limited. This gap raises a fundamental question: *how does system architecture influence the resilience of AD agents to action space attacks?*

To answer this question, we evaluate three representative AD agents: 1) *modular driving agents*, which employ a hierarchical pipeline with dedicated modules for specific driving tasks, 2) *end-to-end driving agents* built with deep reinforcement learning (DRL), which directly map sensor inputs to actuator outputs through a single policy network, and 3) advanced *feature-fused driving agents*, which incorporate detailed vehicle state from modular components into DRL training to improve driving performance. We conduct evaluation experiments in the CARLA simulator [15] using freeway scenarios that capture key aspects of real-world driving complexity, and measure the performance of each agent in tracking planned trajectories. Thus, the path tracking deviation is used as the primary metric to quantify how well each agent maintains stable and accurate control in the presence of action space attacks.

In this study, we investigate action space attacks that perturb the ego vehicle's lateral control to induce side collisions with nearby vehicles. We assume a strategic adversary that performs *few-shot* interventions during high-risk maneuvers such as lane changes and overtaking. We refer to these intervals as *safety-critical moments*, characterized by narrowed safety margins caused by rapid changes in relative speed and dynamic interactions between vehicles. For realism of the threat model, we impose three practical constraints on the adversary. First, the attacker is limited to *additive* perturbations on the ego vehicle's lateral control. Perturbation magnitudes are capped by an *attack budget* that is related to either the attacker's physical capability limit or a stealth-preserving maximum choice selected to evade potential physics-based attack detectors [11, 17]. Second, the adversary operates under a black-box setting and has no access to the driving agent's internal model, parameters, or decision rules. Third, the adversary may only have partial state observation because of practical sensing and deployability limits. Under these constraints, we train the adversary's policy using DRL and develop two attack variants: a vision-based approach using camera sensors and a motion-based approach relying exclusively on inertial measurement units (IMUs). These variants demonstrate a trade-off between sensing modalities and deployability: cameras offer rich visual information but are difficult for the adversary to mount and conceal, whereas IMUs provide coarse motion cues but allow covert in-vehicle deployment.

We begin by evaluating the behavioral vulnerabilities of different AD architectures under action space attacks. Modular agents exhibit strong robustness to minor disturbances but become highly susceptible under high-budget attacks. In contrast, the end-to-end DRL agent, which processes raw images, is more resilient to strong perturbations yet loses precision under nominal driving or under low-budget attacks. The feature-fused agent maintains stability under low-budget attacks but degrades sharply once the attack budget exceeds a threshold. The results suggest the necessity of a defense mechanism capable of maintaining robustness across nominal driving as well as under low- and high-budget attacks. To this end, we first revisit adversarial training, a common approach to improving policy robustness against action space attacks [26, 41]. In our preliminary study [45], we investigated fine-tuning-based adversarial training and extended it with progressive neural networks (PNNs) [37] to enhance adaptation across attack scenarios. Results showed that, while the adversarial training and its extended variant mitigate some vulnerabilities, they introduce new limitations such as overfitting and reliance on known attack patterns.

A key insight from our preliminary analysis [45] is that different AD architectures exhibit complementary robustness. Modular control provides high precision and stability under nominal driving or mild disturbances, whereas DRL-based policies adapt more effectively to stronger attacks

and can be further enhanced through adversarial training. This observation suggests that integrating their strengths could yield more balanced resilience across attack budgets. Guided by this insight, we design a learning-based Path Correction System (PCS) that unifies their advantages to achieve balanced robustness across attack budgets. PCS augments the modular agent's error-minimizing feedback with a supplementary DRL policy, which effectively reduces path-tracking deviations and alleviates the overfitting issues associated with adversarial training. To preserve nominal performance and runtime efficiency, PCS integrates a kinematic-model-based attack detector that activates the supplementary DRL loop only when an attack is detected.

Our contributions are summarized as follows. **First**, we present a systematic and comprehensive evaluation of action space attacks across multiple AD architectures to reveal key architectural factors that influence robustness. **Second**, we critically analyze two existing adversarial training defense strategies and uncover their inherent limitations, motivating the need for more adaptive and effective solutions. **Third**, we propose PCS, a novel AD architecture that integrates modular feedback control and learning-based correction, where the latter is activated by an attack detector. PCS achieves up to a 78% reduction in path-tracking error when the system is under attack.

Paper organization: Section 2 presents the preliminaries and reviews related work. Section 3 describes the system model. Section 4 details the threat model and the training of the adversary's policy. Section 5 presents attack evaluation results. Section 6 evaluates and analyzes existing defense strategies. Section 7 introduces the proposed PCS architecture and presents its performance. Section 8 presents the attack detector in PCS. Section 9 discusses several related issues. Section 10 concludes this paper.

2 Preliminary and Related Work

2.1 Autonomous Driving Agent

We categorize AD agents as: 1) *modular*, 2) *end-to-end*, and 3) *feature-fused driving agent*.

Modular driving agent: Modular driving agents adopt a hierarchical decision-making architecture, in which individual modules are responsible for distinct subtasks such as route planning, behavior decision-making, motion planning, and path tracking [32]. This design offers clear modular boundaries and interpretability that facilitate maintenance and system upgrades. However, decomposing the driving task into multiple interacting modules increases integration complexity and development overhead [28]. This prompts the search for alternatives.

End-to-end driving agent: End-to-end driving agents have gained growing attention as a cost-effective alternative to modular pipelines [10]. These approaches replace numerous modules with a single policy network that maps raw inputs (e.g., images) directly to action distributions. Typically, end-to-end policies are trained via imitation learning (IL) [33, 42], or DRL [20, 34]. IL leverages expert demonstrations and follows a supervised learning paradigm, resulting in fast convergence but with performance ultimately bounded by the quality of the teacher [8]. In contrast, DRL learns through trial and error, which enables greater adaptability in complex and dynamic environments. Despite recent progress [5, 16, 46], end-to-end agents continue to facing challenges in generalizability [12], reproducibility, and remain vulnerable to cybersecurity threats. In this paper, we implement DRL for end-to-end freeway driving.

Feature-fused driving agent: Feature-fused driving agents employ a hierarchical decision-making structure that uses essential features, such as path deviations and proximity to surrounding vehicles, as inputs to policy networks [4, 23]. This approach yields two main benefits: 1) dimensionality reduction, which simplifies the model and lowers computational overhead, and 2) mitigation of the domain shift problem often encountered in end-to-end architectures that use image inputs [43]. This framework improves the system's adaptability to diverse driving scenarios.

(a) CARLA freeway driving (b) Attack goal: side collision

Fig. 1. (a) Driving scenario in CARLA involving lane changes and overtaking. Green arrows indicate a safe reference path. (b) The attacker aims to cause a side collision with an NPC vehicle; the dashed line indicates the vehicle's trajectory over the past 3 seconds.

2.2 Safety and Security Challenges of DRL in Autonomous Driving

DRL has been successfully applied to a range of AD tasks, including lane keeping [9], lane changes [1], ramp merging [4], and intersection navigation [23]. However, despite ongoing efforts to enhance the safety and reliability of DRL for AD [36], the lack of strict safety guarantees and limited explainability remain significant barriers to its adoption in real-world systems. In addition to these challenges, DRL-based systems are vulnerable to security risks arising from adversarial attacks. Such attacks can be categorized as either *white-box attacks*, which require knowledge of the target model's internal details, or *black-box attacks*, which only require access to the executable agent. Attacks can further be classified as *state space attacks*, which target the agent's inputs by altering input images at the pixel level [27] or manipulating the observed environment [18], and *action space attacks*, which directly manipulate the agent's outputs to drive the system into unsafe states [26]. In this study, we design a black-box action space attacker that induces side collisions, thereby revealing vulnerabilities in safety-critical AD scenarios.

3 System Model

In this section, we describe the analyzed modular, end-to-end DRL, and feature-fused DRL AD agents along with the freeway driving scenario used for evaluation. All agents are implemented or trained in CARLA 0.9.11 [15], an open-source simulator widely used in AD research.

3.1 Freeway Driving Scenario Setup

We construct a freeway driving scenario in CARLA Town 4 (Fig. 1(a)), involving lane changes and overtaking. The ego vehicle, indicated by green vehicle color in Fig. 1(a), is tasked with overtaking six non-player character (NPC) vehicles, while maintaining a target speed of 16 m/s. Each NPC vehicle travels at 6 m/s. The task must be completed within 250 simulation steps, each lasting 0.1 seconds. This scenario is chosen for its representative complexity, characterized by dense agent interactions that present a valuable opportunity for attackers to induce significant disruption.

3.2 Modular Driving Agent

We use CARLA Autopilot as our modular driving agent. It employs a hierarchical pipeline that generates feasible waypoints and tracks them in real time through low-level actuator control for longitudinal (throttle/brake) and lateral (steering) movements. At each time step t, for each

control dimension $x \in \{\text{longitudinal}, \text{lateral}\}$, the actuator command a_t^x is refined by the corresponding feedback controller, which output actuator adjustment Δa_t^x for each control dimension $x \in \{\text{longitudinal}, \text{lateral}\}$, as follows:

$$a_t^x = (1 - \eta^x) \cdot \Delta a_t^x + \eta^x \cdot a_{t-1}^x, \quad \Delta a_t^x \in [-1, 1], \quad \eta^x \in [0, 1], \tag{1}$$

where η^x is a retention factor that smooths control transitions between two time steps. In this paper, we set this retention factor to be 0.7 for longitudinal and 0.9 for lateral control. Control semantics follow the convention: a positive adjustment increases throttle or turns right, while a negative adjustment applies braking or turns left. The lateral actuator command a^{lateral} is normalized to the range [-1, 1], which represents the system's maximum steering capacity in radians.

A modular driving agent can be configured for different driving behaviors. For example, a highly cautious agent may minimize collision risk by moving at low speed, maintaining a large safety distance from nearby vehicles, and prohibiting lane changes or overtaking maneuvers. However, such behavior can impede traffic flow. To better capture the norm of freeway driving with overtaking behavior, we configure the driving agent to have the following settings: the *reference speed* is set to 16m/s; a *following distance* of 2 meters is set; and thus *lane-changing decisions* are triggered when the distance to the vehicle ahead falls within 2 meters. The feedback controller parameters are tuned to maintain responsiveness under these conditions. This configuration allows the agent to execute decisive overtaking maneuvers while maintaining path-tracking stability.

3.3 DRL-based End-to-end Driving Agent

We implement an end-to-end AD agent using the *soft actor-critic* (SAC) algorithm [21, 29], an advanced DRL method for continuous control. The driving policy is parameterized by a neural network that maps camera inputs directly to actuator commands and is trained to optimize both driving and safety objectives. Below, we detail its state space, action space, and reward function.

- **State space:** The input to the driving policy consists of front-view semantic segmentation images from CARLA's camera sensor. To capture temporal dynamics, each input is constructed by stacking three consecutive 300-degree panorama views (each of resolution 84×420) per time step, similar to the setup in [16].
- **Action space:** Aligned with the modular agent, the driving policy outputs actuator adjustments $\Delta a_t^x \in [-1, 1]$ at each time step t, where $x \in \{\text{longitudinal}, \text{lateral}\}$. These are combined with the previous actuator signal using the same weighted update rule as in Eq. (1) to ensure smooth control transitions.
- **Reward function:** The driving policy is trained with a reward function that integrates path tracking, speed regulation, and safety. To construct a reliable driving agent, we leverage the global and local path-planning modules used by the modular driving agent (Section 3.2) to generate a reference path and incorporate it into our reward scheme. This transition from a coarse objective (e.g., staying on the road) to a fine-grained instruction (e.g., following a legal waypoint trajectory) improves learning efficiency and encourages safer behaviors. Specifically, the reward at time t is defined as:

$$R_v = \beta_d r_d + \beta_s r_s + \beta_c p_c, \tag{2}$$

where r_d is the normalized dot product between the ego vehicle's speed vector and the generated reference path, r_s rewards maintaining the target speed, and p_c penalizes for collision and shoulder violations. Each term's weight parameter β defines its relative importance. This reward function captures multiple driving goals simultaneously.

3.4 DRL-based Feature-Fused Agent

We implement a feature-fused agent that combines the strengths of modular and end-to-end designs. While it shares the same training paradigm as the end-to-end agent, it differs in its state space: instead of raw images, it uses an 8-dimensional vector of driving features extracted from upstream modules. These include: 1) the direction vector to the next driving waypoint, 2) the proportional, integral, and derivative components of the angular deviation between the vehicle's orientation and the reference path, 3) the vehicle speed magnitude, and 4) the actuator signal (longitudinal and lateral controls) from the last time step. This compact representation enables reduced network structure and efficient training. Specifically, the training of this feature-fused agent converges in 100,000 steps, which is only one third of that required by the end-to-end agent.

3.5 Nominal Driving Performance

We evaluate driving performance over 30 testing epochs. All three agents (i.e., the modular, the end-to-end, and the feature-fused agents) consistently complete the task without collisions and follow the intended trajectory in the absence of attacks. Note that these driving agents may exhibit degraded performance on different road types, as they have been narrowly trained or fine-tuned for optimal performance for the freeway driving scenario. In this paper, we focus on the freeway driving scenario and study their adversarial robustness in the presence of action space attacks.

4 Threat Model

This section presents the threat model used in our study, which includes the attacker's objectives, requirements, key challenges, and algorithm design. A broader discussion on real-world feasibility, sim-to-real transferability, and applicability to other tasks is provided in Section 9.

4.1 Attacker's Objectives, Capabilities, and Constraints

The attacker's goal is to induce a side collision between the ego vehicle and a nearby NPC vehicle (See Fig. 1(b)). We model a strategic attack that is triggered only during *safety-critical moments*, specifically when the ego vehicle initiates lane changes or overtaking maneuvers. These moments are formally defined in Section 4.3.4. We denote the control policy of the ego vehicle, i.e., the victim of the attack, as π_{victim}, which can represent any of the AD agents introduced in Section 3. We denote the attacker's policy as π_{attack}, which perturbs the vehicle's lateral control by injecting bounded additive perturbations. The attacker operates under the following practical constraints:

- **Bounded interference on lateral control:** The attacker injects bounded, additive perturbations δ_t into the lateral adjustments $\Delta a_t^{\text{lateral}}$ computed by π_{victim}, which results in a modified lateral actuator command:

$$a_t^{\text{lateral}\prime} = (1 - \eta^{\text{lateral}}) \cdot (\Delta a_t^{\text{lateral}} + \delta_t) + \eta^{\text{lateral}} \cdot a_{t-1}^{\text{lateral}}, \quad |\delta_t| \leq \varepsilon, \tag{3}$$

  where ε is the attack budget defined in Section 4.3.3. The attack can only bias, but not override, the π_{victim}'s raw output. The a_{t-1}^{lateral} is the adversarially interfered actuator command if the attacker performed injection in the previous time step $t-1$.
- **Black-box access to the driving agent:** The attacker has no access to π_{victim}'s model internals, such as architecture or parameters. However, the attacker can account for driving behavior through interactions with a high-fidelity simulator (e.g., CARLA). We assume π_{victim} remains fixed after deployment, which allows the attacker to train its policy π_{attack} offline using π_{victim} in the black-box setting.
- **Sensing modality and deployability trade-off**: The attacker acts based on sensor-derived observations of the vehicle's environment and dynamics. We consider two sensing modalities.

The first uses a forward-facing camera to obtain a full semantic context of the driving scene, which enables more accurate detection of traffic scenarios and safety-critical moments. The second relies solely on IMU data to infer vehicle motion. While the camera-based attacker benefits from richer observations, it is more likely to be noticed or constrained by mounting and field-of-view requirements. In contrast, the IMU can be discreetly embedded within the vehicle and is less likely to be noticed, which offers a more covert but less informed attack pathway.

4.2 Attacker's Challenges

In this section, we outline the challenges faced by the attacker given the constraints described in Section 4.1. The attacker must simultaneously address two conflicting objectives: the attack must remain *stealthy*, or *subtle* to avoid triggering anomaly detection, while being *effective* to overcome the inherent resilience of the driving agent. We elaborate on each of the two challenges below.

4.2.1 Risk of attack detection. Physics-based attack detection approaches aim to identify anomalies by monitoring deviations from physical invariants of vehicle dynamics [11, 17, 35]. These methods construct offline reference models, using either analytical physics [11] or learned correlations [35], to track consistency among various quantities measured from the vehicle. However, as shown in [35], most physics-based attack detection approaches are vulnerable to stealthy attacks with low magnitude. For instance, the detection strategy in [11] tracks squared prediction error $s_{\text{err}} = |y - y_p|^2$, where y and y_p represent the observed and predicted signals. The accumulated error $\text{err\_sum}(t+1) = \text{err\_sum}(t) + s_{\text{err}}(t)$ is averaged over a sliding window of length t_w, and an anomaly is flagged when $\text{err\_avg} = \frac{\text{err\_sum}}{t_w}$ exceeds a threshold τ. The detector resets once t_w exceeds the monitoring window, which is usually set based on the longest primitive operation (e.g., overtaking, turns), and τ is based on maximum prediction-induced errors. In state-space attacks, the adversary injects additive perturbations into sensor outputs, such as GPS readings, following $y' = y + \delta$. Stealth is preserved provided that the perturbation magnitude remains within the bound $|\delta| \leq \sqrt{\tau t_w - \text{err\_sum}(t)}$.

To remain stealthy, the magnitude of the action space attack needs to meet a certain bound, similar to the above analysis for state space attacks. However, the attacker typically lacks access to detector internals (e.g., t_w, T) or model structures. As such, it may not be able to compute stealth-preserving bounds explicitly. Instead, in this paper, we evaluate attack performance under different levels of perturbation budget ε. This attack performance profiling provides a basic understanding on the relationship between attack effect and perturbation budget. It is worth making two notes. First, only under a subset of the considered perturbation budget settings, the attack can bypass the physics-based attack detection. Second, even if the attack bypasses the attack detection, its effect may still be suppressed by the control system's feedback mechanisms, which we detail next.

4.2.2 Inherent resilience from vehicle's control. The ego vehicle's control policy π_{victim} exhibits inherent resilience to attacker π_{attack} described in Section 4.1, which arises from three aspects:

- **Active lateral control correction:** Whether derived from classical feedback control or learned policy, π_{victim} adjusts vehicle's lateral control based on real-time observations. These ongoing corrections help suppress the impact of bounded adversarial perturbations.
- **Preserved longitudinal control:** Since the attacker only perturbs the lateral control, the agent retains full authority over throttle and braking. This enables it to regulate speed and avoid potential collisions, offering an orthogonal channel of compensation.
- **Smoothed actuator control update:** As defined in Eq. (3), the lateral actuator command is a weighted blend of the current actuator adjustment and the previous actuator command.

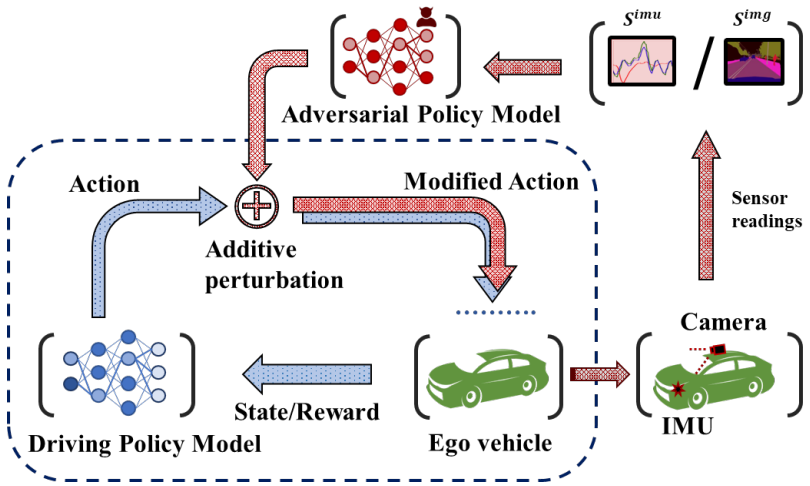


Fig. 2. Overview of the DRL-based action space attack.

This smoothing acts as a low-pass filter, damping sudden steering changes and suppressing the effect of transient disturbances.

Together, they constitute a natural layer of defense that mitigates the impact of naive or untargeted interference. As shown in Section 5.2, our proposed context-aware attacker significantly outperforms intuitive baselines by precisely exploiting timing and directional vulnerabilities.

4.3 Adversary Design

To address the challenges and constraints described in the previous sections, we formulate the attack as a DRL problem. As shown in Fig. 2, the ego vehicle (green) operates under the victim policy π_{victim}, which governs the nominal driving process (blue loop) and is treated as a black box by the attacker. The attacker observes the environment through its own sensing stream, either a semantic camera or an IMU, and learns a policy π_{attack} that injects bounded additive perturbations into the ego vehicle's lateral control. This process is illustrated by the red loop in Fig. 2. Since π_{victim} is fixed after deployment, the closed-loop system dynamics are stationary and can be modeled as a Markov decision process. This enables the attacker to train π_{attack} offline using standard DRL algorithms. However, to achieve a stealthy yet effective adversarial policy, careful design of the state space, action space, and reward function is needed, each of which is detailed next.

4.3.1 Adversarial state space. We consider two attacker variants distinguished by their sensor modalities: one based on camera input and the other on IMU data.

- **Camera-based attack** uses a front-facing semantic segmentation camera mounted on the roof of the ego vehicle for a wide field of view. The input state s^{img} consists of time-stacked semantic maps that capture object locations and drivable areas. These inputs offer rich spatial context for understanding surrounding traffic.
- **IMU-based attack** uses a triaxial IMU mounted in the center of the victim vehicle, where the x-y-z axis readings record the speed changes of the vehicle as it advances, rolls, and yaws. A trace of the IMU readings sampled at 20 Hz over the last 3.2 seconds in the x-axis and the z-axis is used as the state input and is denoted by s^{imu}. Y-axis readings are not used since they provide limited information about steering characteristics. In cases where the IMU is installed elsewhere, the triaxial sensor readings and orientation alignment may be required.

Compared with camera-based input, IMU data provides limited spatial awareness and lacks environmental context. This makes it harder for the attacker to identify safety-critical moments and start the malicious perturbation process accordingly. This design represents a more constrained and realism-oriented threat model.

4.3.2 Adversarial action space. The attacker perturbs the ego vehicle's lateral control by injecting bounded additive noise δ_t into the steering adjustment Δa_t^{lat} computed by the driving agent, as specified in Eq. (3). These perturbations are constrained by a predefined attack budget ε and are applied selectively during safety-critical moments where small control deviations are more likely to induce collisions.

4.3.3 Attack budget. The attack budget ε defines the maximum allowable magnitude of each perturbation applied to the ego vehicle's lateral control, i.e., $|\delta| < \varepsilon$. A larger ε allows the attacker to impose greater influence on the vehicle's trajectory, but increases the risk of detection. In the worst-case scenario, we set $\varepsilon = 1$, which allows the attacker to apply perturbations up to the same magnitude as the victim agent's own lateral adjustment $\Delta a_t^{\text{lateral}}$. According to Eq. (1) and the setting of η^{lateral}=0.9, the setting of $\epsilon = 1$ allows the attacker to alter up to 10% of the lateral actuator command a_t^{lateral} per time step. In more realistic cases, the attacker operates under tighter budget constraints to avoid triggering the physics-based attack detectors. To systematically evaluate the susceptibility of different AD designs, we vary ε across a range of settings, with the goal of developing resilient autonomous driving policies that remain robust under levels of action space attack.

4.3.4 Safety-critical moments and side collisions. During driving, lane changes and overtaking maneuvers inherently increase collision risk due to the close spatial proximity and dynamic interaction between the ego vehicle and nearby NPC vehicles. We refer to these short intervals as *safety-critical moments*, during which even small steering deviations may lead to collisions. In what follows, we quantitatively define the safety-critical moments. Let $\mathbf{v}_{ego}$ and $\mathbf{v}_{npc}$ denote the velocity vectors of the ego and NPC vehicles, respectively, and $\mathbf{v}_{e2n}$ the relative position vector from the ego vehicle to the NPC vehicle. We denote the unit vector of any non-zero vector $\mathbf{x}$ as $\hat{\mathbf{x}} = \mathbf{x}/||\mathbf{x}||$.

Empirically, nominal (attack-free) lane changes and overtaking maneuvers occur within a 120° cone centered on the NPC vehicle's lateral axis. We represent this spatial relationship using the dot product between the unit vectors $\hat{\mathbf{v}}_{e2n}$ and $\hat{\mathbf{v}}_{npc}$, denoted by $\omega = \hat{\mathbf{v}}_{e2n} \cdot \hat{\mathbf{v}}_{npc}$. To aid interpretation, we describe spatial relations using a clock-face convention centered on the NPC vehicle, where the NPC's heading direction defines the 12 o'clock axis, and its right-lateral direction defines the 3 o'clock axis. Under this convention:

- when the ego vehicle is positioned at the 5 o'clock or 7 o'clock direction relative to the NPC vehicle after the overtaking process is initiated, $\omega = \cos(\pi/6)$;
- when the ego vehicle is aligned laterally with the NPC vehicle (3 o'clock or 9 o'clock), the vectors are orthogonal and $\omega = 0$;
- when the ego vehicle completes overtaking and moves to the 1 o'clock or 11 o'clock positions, $\omega = -\cos(\pi/6)$.

Therefore, the ego vehicle's lane-change and overtaking maneuvers with respect to the NPC vehicle can be characterized by

$$|\omega| = |\hat{\mathbf{v}}_{e2n} \cdot \hat{\mathbf{v}}_{npc}| \leq \cos(\pi/6), \tag{4}$$

during which a potential side collision is geometrically feasible. We identify the time duration meeting Eq. (4) as *safety-critical moments*. A collision is labeled as a *side collision* if the contact

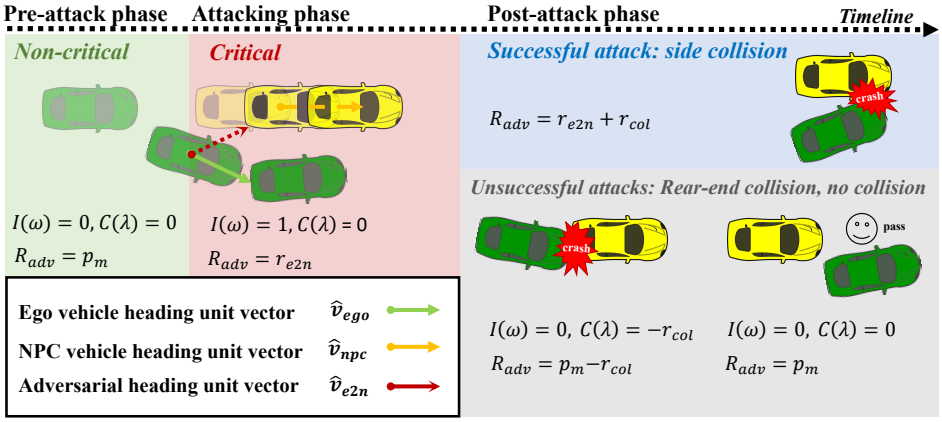


Fig. 3. Schematic plot illustrating the evolution of an action-space attack. The timeline consists of a pre-attack phase (non-critical), an attacking phase (safety-critical), and a post-attack phase (outcome). A successful attack results in a side collision with another vehicle.

occurs during the safety-critical moment. In CARLA, collisions are detected using the built-in `CollisionSensor`, which records the impact time, impulse, and identities of the colliding vehicles.

4.3.5 Adversarial reward shaping. The safety-critical moments defined above guide our adversarial reward shaping. We introduce a binary indicator

$$I(\omega) = \begin{cases} 1, & \text{if } |\omega| \le \cos(\pi/6), \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

which separates attack phases. As illustrated in Fig. 3, the attacker remains inactive during the pre-attack phase, when the ego vehicle (green) begins overtaking and the safety-critical condition is not met. Once the spatial alignment satisfies $I(\omega) = 1$, indicating lateral proximity to the target NPC (yellow), the attacker enters the safety-critical phase and injects steering perturbations to induce a side collision.

Accordingly, we design the adversarial reward R_{adv} to encourage the adversary to trigger a side collision during safety-critical moments, and penalize unnecessary or excessive perturbations elsewhere. It consists of three components:

- **Collision reward** $C(\lambda)$**:** a reward assigned based on the final outcome of each driving episode. A successful side collision yields a positive reward r_{col}, an unexpected collision (e.g., rear-end or road-edge collision) incurs a penalty $-r_{\text{col}}$, and no collision gives zero reward. Here, $\lambda = 1$, $\lambda = -1$ or $\lambda = 0$ represent the three cases, respectively.
- **Directional reward** r_{e2n}**:** a continuous reward promoting geometric alignment between the ego and target NPC during critical moments, defined as $r_{e2n} = \hat{v}_{e2n} \cdot \hat{v}_{ego}$, where the unit normalization operations isolate directionality from speed to stabilize learning.
- **Attack maneuver penalty** p_m**:** a per-step cost discouraging excessive perturbation injection during non-safety-critical moments, computed as $p_m(t) = -|\delta_t|$, where δ_t is the injected additive steering perturbation at time t.

Combining these terms with the binary indicator in Eq. (5), the adversarial reward for *camera-based attack* is:

$$R_{adv}^{\text{cam}} = C(\lambda) + I(\omega) r_{e2n} + (1 - I(\omega)) p_m. \tag{6}$$

For *IMU-based* attack, as spatial observability is limited, we adopt a teacher–student (imitation) training paradigm. Let δ_t^{imu} and δ_t^{cam} denote the perturbations injected by the IMU-based and

camera-based attacks at time step t, respectively. During IMU-based attack training, we add an imitation penalty, $p_{\text{imit}}(t) = -(\delta_t^{\text{imu}} - \delta_t^{\text{cam}})^2$, which encourages the IMU-based attacker (i.e., student) to match the camera-based attacker's (i.e., teacher) perturbation patterns. The reward for *IMU-based attack* becomes:

$$R_{adv}^{\text{imu}} = C(\lambda) + I(\omega) r_{e2n} + (1 - I(\omega)) p_m + p_{\text{imit}}. \tag{7}$$

After training, the IMU-based attack can operate using only onboard motion signals. Section 5.1.2 evaluates this paradigm and reports the relative effectiveness of the camera and IMU variants.

4.3.6 Training parameters. We train the attack policy using SAC [21], which maximizes expected reward while encouraging exploration via entropy regularization. The model architecture consists of three components: a CNN-based encoder that processes raw sensor inputs (4 layers of Conv2D for camera input and 3 layers of Conv1D for IMU), each followed by a ReLU activation, a policy network that selects bounded steering perturbations, and a Twin Q-network [44] that evaluates state-action pairs and guides the policy to take the best action maximizing long-term reward.

Training proceeds in an alternating loop of data collection and network updates. During data collection, the attacker, embedded in the ego agent, perturbs its lateral control within the specified attack budget, which influences overtaking or lane-changing behavior in the driving scenario defined in Section 3.1. Resulting transitions, including state-action pairs, rewards, and post-action states, are stored in a replay buffer of size 100,000. The networks are updated every 2,048 steps using the Adam optimizer with a learning rate of 2×10^{-4} and with a reward discount factor of 0.96 that emphasizes long-term rewards. Empirically, the camera-based attack converges after approximately 60,000 steps, while the IMU-based attacker, trained via imitation from the camera-based policy, converges after around 110,000 steps.

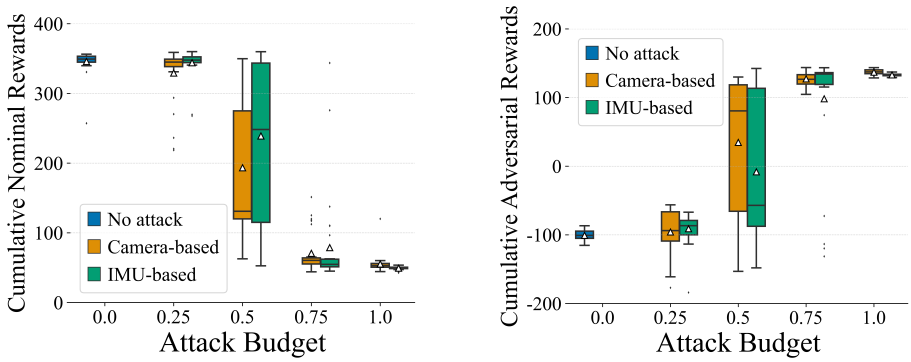
5 Attack Impact Evaluation

This section evaluates the impact of attacks across sensor modalities and attack budgets, and compares how different AD agents perform under attacks. Due to the absence of attacks targeting this specific driving scenario, we use a set of intuitive attack strategies as comparison baselines. We begin the evaluation using the end-to-end AD agent unless stated otherwise.

5.1 Effect of Sensor Modality and Attack Budget on Attack Effectiveness

5.1.1 Evaluation metrics. We report two metrics: 1) the *Cumulative Nominal Reward* (CNR), which reflects the AD agent's driving performance, and 2) the *Cumulative Adversarial Reward* (CAR), computed as the average sum of R_{adv} over evaluation episodes, which reflects the success of the attack. CNR measures how well the AD agent achieves its intended goals (e.g., path tracking, collision avoidance), with lower values indicating greater disruption. CAR captures the attacker's objective: inducing a targeted side collision. A higher CAR corresponds to a more effective attack. CAR is positive when an episode produces the targeted side collision and thus denotes a successful attack; failure cases (collision-free episodes or unintended collisions) produce a negative CAR. To simplify exposition, a nominal driving scenario is referred to as an attack case with zero budget ($\varepsilon = 0$).

5.1.2 Comparison of camera-based and IMU-based attacks. Fig. 4 shows that, under the maximum attack budget ($\varepsilon = 1.00$), the camera-based attack significantly degrades the AD agent's performance, which reduces the CNR by approximately 84% (from 345.76 to 55.28). It also consistently achieves CAR exceeding 100 with low variance, indicating that side collisions reliably occur upon encountering the first NPC vehicle. In comparison, the IMU-based attack achieves a slightly lower CAR with greater variance but still effectively launches the targeted attack. This confirms that the IMU-based



(a) Nominal performance (b) Adversarial performance

Fig. 4. Box plots of average cumulative rewards over 30 testing episodes under varying attack budgets. (a) The victim agent's driving performance is measured by cumulative nominal reward (CNR). (b) The attack performance is measured by cumulative adversarial reward (CAR).

attacker effectively learns from the camera-based policy and demonstrates the feasibility of using low-dimensional IMU signals for action space attacks.

5.1.3 Impact of attack budget. Fig. 4 also illustrates how attack effectiveness varies with the attack budget. As expected, both camera-based and IMU-based attacks exhibit increasing effectiveness as the attack budget increases. Across all budget levels, the camera-based attack consistently outperforms the IMU-based attack with a higher average CAR and a lower variance. This highlights a strong correlation between attack effectiveness and the informativeness of the attacker's observation modality. The visual modality provides rich spatial context, allowing the camera-based attack to more easily identify safety-critical moments and induce the vehicle toward side collisions. In contrast, the IMU-based attack, relying on indirect motion cues, faces greater difficulties in estimating relative position and achieving the intended attack effect. Notably, both CNR and CAR exhibit a sharp shift as the attack budget decreases from $\varepsilon = 0.75$ to $\varepsilon = 0.25$. This suggests the presence of a tolerance threshold in the driving agent's action space: once the injected perturbation exceeds this threshold (e.g., $\varepsilon = 0.5$), the vehicle's driving behavior is significantly disrupted.

5.2 Comparative Evaluation with Baseline Attackers

5.2.1 Baseline attack strategies and their rationale. To assess the effectiveness of our learned adversarial policy, we conduct a controlled comparison against several intuitive baseline attack strategies [47], outlined in Table 1. Each baseline isolates specific aspects of attacker design, such as timing, magnitude, and directional awareness:

- **Rand-RandStep**: This baseline represents an untargeted, opportunistic attacker that lacks both contextual awareness and directional consistency. It injects random-magnitude perturbations within a 2.5-second window randomly selected from each episode, independent of the vehicle's state. At every step during the window, perturbations are sampled from a uniform distribution $\mathcal{U}[-\varepsilon, \varepsilon]$. The 2.5-second duration reflects the average human driver reaction time [38, 48].
- **Crit-RandStep:** It represents a plausible but naive attacker that knows when to intervene (e.g., during safety-critical moments) but lacks knowledge of optimal direction or magnitude. It simulates untargeted disturbances, such as actuator noise, where perturbations are sampled

Table 1. Overview of attack strategies.

Strategy	Timing	Magnitude	Directionality
Rand-RandStep	Random window[1]	Random[2]	Random (left/right)
Crit-RandStep	Critical moment	Random[3]	Random (left/right)
Crit-UniRand	Critical moment	Uni-Random[4]	Unidirectional (based on targeted vehicle side)
Crit-UniFix	Critical moment	Uni-Fixed[4]	Unidirectional (based on targeted vehicle side)
Ours (camera-based)	Critical moment	Adaptive	Strategic

[1] Random window: a 2.5-second interval randomly selected within each episode. [2] Random: per-step sampling from $\mathcal{U}[-\varepsilon, \varepsilon]$.
[3] Uni-Random: per-step sampling from $\mathcal{U}[0, \varepsilon]$ or $\mathcal{U}[-\varepsilon, 0]$. [4] Uni-Fixed: fixed value per window from $\mathcal{U}[0, \varepsilon]$ or $\mathcal{U}[-\varepsilon, 0]$.

independently at each step from a uniform distribution $\delta_t \sim \mathcal{U}[-\varepsilon, \varepsilon]$. This baseline serves as a lower bound for timing-sensitive vulnerability.

- **Crit-UniRand:** It simulates a partially informed attacker that persistently steers toward the NPC vehicle during safety-critical moments. If the NPC is on the left, perturbations are sampled from $\mathcal{U}[-\varepsilon, 0]$. If on the right, from $\mathcal{U}[0, \varepsilon]$. It reflects directional intent without precise control over magnitude.
- **Crit-UniFix:** It models a persistent attacker that injects a fixed, one-sided perturbation based on NPC position. It uses the same sampling logic from UniRand for a constant perturbation throughout the attack period. This reflects coarse directional interference without adaptive control.

5.2.2 Evaluation metrics. We use the following metrics in this section. 1) *Attack Success Rate (ASR).* We define an *attack attempt* as the time interval of safety-critical moments during which the attack is active. The ASR is computed as the percentage of attack attempts that result in a targeted side collision. 2) *Time to Collision (TTC).* This metric measures the duration between the activation of the attack and the occurrence of the resulting collision. And 3) *Deviation from Trajectory (DFT).* It quantifies the attack impact by the vehicle's deviation from the reference path, which is computed as the root mean square of one minus the normalized dot product between the vehicle's velocity vector and the planned path vector, where a dot product of one indicates perfect alignment. In this evaluation, we report DFT for each *successful* attack attempt to capture the degree of control disruption imposed by the attack. For each attack strategy, we collect 120 independent attack attempts.

5.2.3 Observations. Based on the results for the modular driving agent in Table 2, we have the following observations:

- **Efficiency of learned attacker.** Our method achieves a 100% ASR, while also inducing the shortest TTC and lowest DFT. The shorter TTC limits the human driver's ability to react, while the small trajectory deviation indicates the attacker can induce collisions with minimal observable path change. Together, these results suggest that the attacker can achieve its objective more effectively through subtle interventions.
- **Impact of temporal and directional awareness.** Attack effectiveness improves with increased contextual awareness. Regarding temporal awareness, targeting perturbations to align with safety-critical moments (i.e., Crit-RandStep) significantly outperforms untargeted attacks launched at random times (i.e., Rand-RandStep). Regarding directional awareness, steering perturbations toward the side of the targeted vehicle (i.e., Crit-UniRand) rather than in random directions further enhances attack success, highlighting the importance of exploiting situational context.

Table 2. Attack performance with the same attack budget ($\varepsilon = 1$).

Attack Strategy	Attack Success Rate	Time to Collision (s) Avg. ± Std.	Deviation from Trajectory Avg. ± Std.
No Attack	0 %	N/A	N/A
Rand-RandStep	3.70 %	1.84 ± 0.47	0.04 ± 0.013
Crit-RandStep	6.03 %	1.28 ± 0.27	0.26 ± 0.388
Crit-UniRand	13.44 %	0.98 ± 0.20	0.12 ± 0.203
Crit-UniFix	28.74 %	0.93 ± 0.19	0.09 ± 0.185
Ours (camera-based)	**100 %**	**0.78 ± 0.07**	**0.05 ± 0.024**

- **Benefit of perturbation consistency.** Fixed-magnitude, unidirectional perturbations (i.e., Crit-UniFix) yield substantially higher success rates than their randomly sampled counterparts (i.e., Crit-UniRand). This indicates that maintaining temporal and directional consistency, even without adaptivity, can more effectively destabilize the vehicle's control policy.

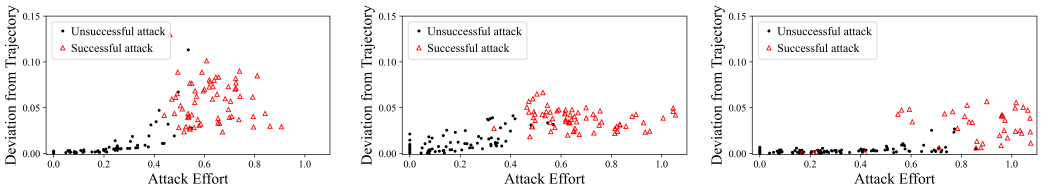
These findings collectively demonstrate the effectiveness of our context-aware attack. Notably, we observe similar trends across other agent architectures, including the end-to-end and feature-fused DRL agents, where contextually timed and directionally consistent attacks also outperform naive baselines. Additionally, TTC varies between the agents. The average TTC is 0.78 seconds (minimum 0.7 seconds) for the modular driving agent, 0.87 seconds (minimum 0.3 seconds) for the end-to-end driving agent, and 1.09 seconds (minimum 0.5 seconds) for the feature-fused DRL agent. These times represent a reduction of 36.8%, 30.4%, and 12.8%, respectively, compared with the best human driver reaction time (minimum 1.25 seconds) under complex real-world conditions [48]. In the following section, we further analyze the attack impact across different AD designs and highlight the distinct behavioral characteristics that inform the design of our proposed defense.

5.3 Attack Performance Across Different AD Designs

To investigate how control-level disruptions vary across different AD system designs, we measure their adversarial impact on path tracking, a core function essential to maintaining stable and safe driving behavior. For each attack budget, ranging from $\varepsilon = 0$ to 1.2 in increments of 0.1, we run 10 driving episodes per AD agent.

5.3.1 Evaluation metrics. We use the following two metrics. 1) The *attack effort*, which defines the time-average magnitude of perturbation applied by the attacker during each attack attempt. Although higher attack budgets allow larger perturbations, the attacker may not always exploit the full budget during an attack attempt. The Attack Effort thus reflects the actual control exerted by the attacker during each attempt. And 2) *Deviation from Trajectory (DFT)*, which has been defined in Section 5.2.2. We report the DFT for every attack attempt, regardless of whether the attack succeeds in this evaluation.

5.3.2 Resilience comparison. Fig. 5 illustrates how different AD agents respond to increasing attack effort, showing DFT (y-axis) as a function of attack effort (x-axis). Each point represents one attack attempt, with red triangles denoting successful attacks and black dots indicating unsuccessful attempts. As expected, an attack tolerance threshold is evident across all agents, where successful attacks begin to dominate once the attack effort exceeds a certain level. In Fig. 5(a), the modular agent demonstrates superior performance in maintaining small deviations when attack effort is below 0.4 but becomes increasingly vulnerable beyond that. In Fig. 5(b), the end-to-end agent shows higher derivation even under zero or low-effort attacks, though higher average effort is still required to induce consistent deviations. Finally, the feature-fused DRL agent in Fig. 5(c) demonstrates the strongest resilience with minimal derivations, with most successful attacks appearing only when



(a) Modular agent (b) End-to-end DRL agent (c) Feature-fused DRL agent

Fig. 5. Trajectory deviation as a function of attack effort across different AD architectures. Each point represents an evaluation episode, with red triangles indicating successful attacks (i.e., those achieving the adversarial goal) and black dots denoting unsuccessful ones. In (a), the modular agent exhibits growing deviation and vulnerability as attack effort increases, with most successful attacks occurring beyond an effort level of 0.5. In (b), the end-to-end DRL agent shows high susceptibility even under low-effort attacks, with successful attacks becoming dominant once the effort exceeds 0.5. In (c), the feature-fused DRL agent demonstrates stronger resilience at low effort levels. Most successful attacks appear after an effort level of 0.6, though a few low-effort failures still occur.

effort exceeds 0.7. However, there is still room for improvement, as some crashes are observed even under low-effort attacks.

5.3.3 Traits of AD agents against action space attack. Building on the above results, we summarize and justify key behavioral traits of the three AD agents against action space attacks:

- **Modular agent:** We attribute the superior resilience of the modular driving agent against low-effort attacks to its feedback controller designed for precise path tracking, which adjusts the actuator values to minimize observed errors in run time. However, its performance degrades under high-effort attacks that overwhelm the controller's corrective capacity, leading to either collisions or unstable steering oscillations.
- **End-to-end DRL agent:** The end-to-end agent shows degraded tracking performance when the attack effort is zero/low, likely due to the absence of explicit state inputs for driving reference (e.g., waypoints). However, since this agent is trained through DRL via trial and error, it has likely encountered and been penalized for anomalous behaviors, such as unsafe proximity to other vehicles that result in collisions. Such exposure may improve its resilience against moderate attacks, as the agent learns to avoid such behaviors. Yet, under substantial attack efforts, the agent ultimately fails to recover, resulting in collisions.
- **Feature-fused DRL agent:** The feature-fused agent, which uses the same reward function as the end-to-end driving agent but incorporates precise error information in its state input, demonstrates exceptional path-tracking abilities under low attack effort, as shown in Fig. 5(c). Compared with the modular agent, the feature-fused agent can process multiple layers of abstract information that traditional controllers cannot handle. This enriched state information leads to better driving performance under high attack efforts, although its performance still declines once the attack budget exceeds a certain threshold.

These findings highlight the distinct characteristics of various AD agents when subjected to action space attacks and the need for effective defense mechanisms. Enhancing a modular agent's performance typically relies on control strategies such as parameter fine-tuning or advanced techniques like Model Predictive Control [22]. However, these approaches are less effective against actuator attacks. Increasing the driving agent's sampling rate has also been proposed as a countermeasure [31]. However, this strategy becomes ineffective if the attacker can match the increased rate. Therefore, we next explore enhancing the resilience of AD agents through a learning-based approach, thereby extending their capability to handle extreme attack scenarios.

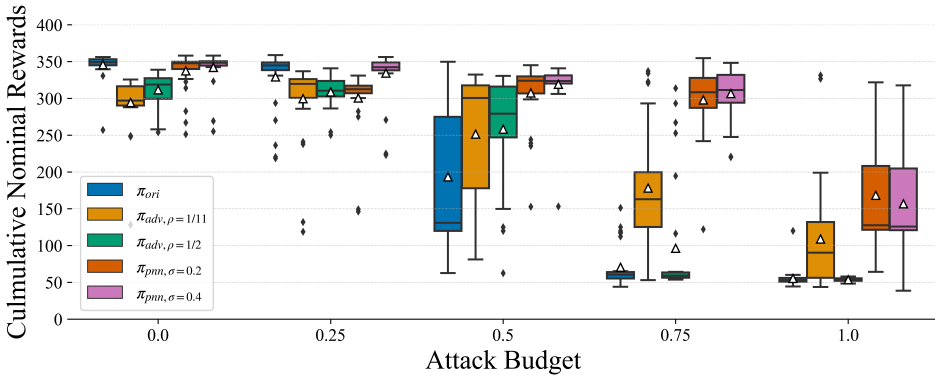


Fig. 6. Box plot of the nominal driving rewards for the original and enhanced end-to-end driving agents, showing the distribution of rewards for each agent. The original agent (π_{ori}) shows a significant drop in rewards as the attack budget increases, while enhanced agents (π_{adv} and π_{pnn}) demonstrate varying degrees of resilience, with π_{pnn}, $\sigma = 0.4$ maintaining higher rewards in both no-attack and strong-attack scenarios.

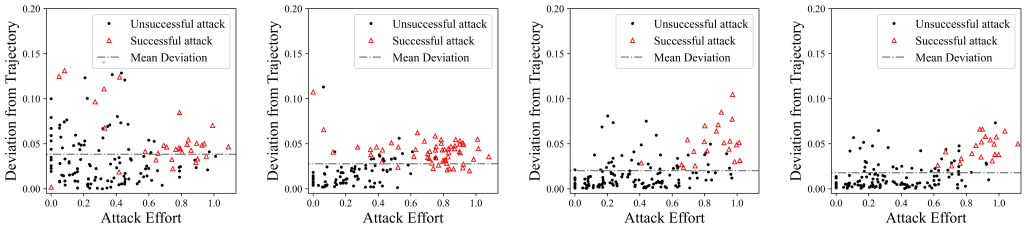
6 Preliminary Defense Strategies by Adversarial Training

This section explores potential defense strategies for end-to-end AD agents. Unlike modular designs, end-to-end models employ a policy network that can be retrained under adversarial conditions through adversarial training, a technique shown to improve robustness against action-space attacks in prior work [26, 41]. In the following, we first implement adversarial training through fine-tuning and evaluate its advantages and limitations. We then investigate PNN-based adversarial training as a potential advanced solution for overcoming the observed limitations in the fine-tuning approach.

6.1 Adversarial Training via Fine-Tuning

Due to the superior effectiveness of the camera-based attack, we adopt it as the adversary for adversarial training. To improve the generalizability of the adversarially trained driving agent, we randomly initiate the training episode with different attack budgets ranging from 0 to 1 with a granularity of 0.1. Moreover, we control the ratio of selecting zero attack budget (i.e., no attack) to prevent overfitting for adversarial cases. We denote the original end-to-end driving agent as π_{ori}, and the adversarially trained agent as $\pi_{\text{adv},\rho}$, where ρ represents the ratio of nominal driving cases included during training on a scale of one. In our experiments, ρ is set to $1/11$ and $1/2$, corresponding to two variants: 1) each of the 11 attack budgets has an equal probability of being selected during training, and 2) the nominal case constitutes half of all the training cases. The Cumulative Nominal Reward (CNR) is used to evaluate nominal driving performance.

Fig. 6 shows that adversarially trained agents ($\pi_{\text{adv},\rho}$) improve resilience across attack budgets, as reflected by higher CNR under camera-based attacks. However, this gain comes with a trade-off: both variants show degraded nominal performance at low or zero attack budgets (e.g., $\varepsilon = 0.25$ or 0), revealing a loss in baseline driving quality due to adversarial overfitting. This trade-off is further illustrated in Fig. 5(b) and Fig. 7(a), where $\pi_{\text{adv},\rho=1/11}$ shows improved robustness, evidenced by a rightward shift in the attack effort threshold for consistent success, but exhibits large trajectory deviations under small or no attack. This is likely due to catastrophic forgetting, where nominal performance deteriorates during adversarial training. In contrast, Fig. 7(b) shows that $\pi_{\text{adv},\rho=1/2}$ better maintains nominal behavior, with fewer outliers and lower average deviation, though at the cost of reduced robustness to stronger attacks.



(a) $\pi_{\text{adv},\rho=1/11}$ (b) $\pi_{\text{adv},\rho=1/2}$ (c) $\pi_{\text{pnn},\sigma=0.4}$ (d) $\pi_{\text{pnn},\sigma=0.2}$

Fig. 7. Evaluation of robustness of enhanced driving agents in terms of deviation from trajectory. The gray dashed line indicates the average trajectory deviation across all evaluation episodes. Adversarially fine-agents (π_{adv}) in (a) and (b) show increasing deviations as attack effort rises, with $\rho = 1/2$ being more vulnerable in cases with low and no attacks. In contrast, PNN-enhanced agents (π_{pnn}) in (c) and (d) display better robustness, with $\sigma = 0.4$ maintaining minimal deviations even at higher attack efforts. Overall, PNN-enhanced agents demonstrate greater resilience than fine-tuned agents across the range of attack efforts.

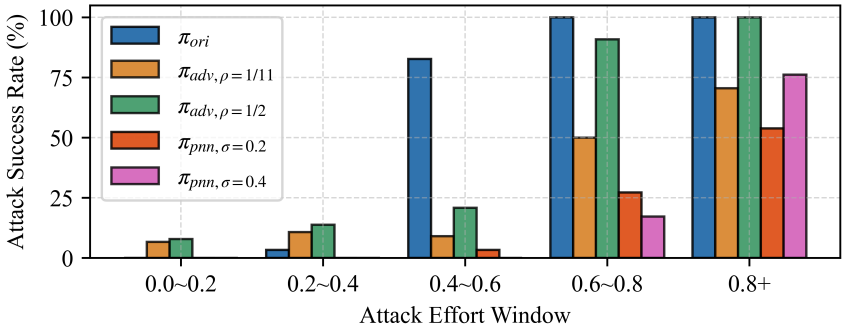


Fig. 8. Comparison of attack success rates between the nominal driving agent and four enhanced agents across different attack effort windows. Compared with adversarial training via fine-tuning, PNN-enhanced agents maintain a zero attack success rate under low-effort windows (0.0–0.4) and generally exhibit greater robustness. This indicates their effectiveness in mitigating the overfitting issues in the fine-tuning approach.

6.2 Adversarial training with Progressive Neural Networks

As discussed, adversarial training by fine-tuning improves robustness against action space attacks but often degrades nominal driving performance. Balancing the proportion of attack cases during training is nontrivial and lacks principled guidance. To address this, we adopt Progressive Neural Networks (PNN) [37], which transfers previously learned features of the original policy network (i.e., a *column*) through lateral connections to a new column, without changing the original weights. This setup allows the new column to learn under attack conditions without forgetting prior capabilities. A runtime *switch* selects between the original driving policy and the adversarially trained column. We denote the resulting agent as $\pi_{\text{pnn},\sigma}$, where σ is the threshold of the attack budget that is used by the switch. Specifically, the switch activates the original driving policy if $\varepsilon \leq \sigma$, and the adversarially trained column otherwise. This design is inspired by the Simplex architecture [40], which assumes access to a reliable runtime indicator for determining which policy to trust. While we use known attack budgets for analysis (with σ set to 0.2 and 0.4), in practice, the switch could instead rely on proxy signals such as attack detection confidence, perturbation magnitude, or inferred attack type.

Fig. 6 presents the performance of PNN-enhanced agents. Compared with the agents enhanced by adversarial training via fine-tuning, the PNN method successfully addresses the forgetting problem when the attack budget is small. At higher attack budgets, the two PNN-enhanced agents

perform similarly, as they share the same model structure and weights. Fig. 7(c) and Fig. 7(d) show the deviation from trajectory versus attack effort for the two PNN-enhanced driving agents. For the $\pi_{\text{pnn},\sigma=0.4}$, it achieves an average trajectory deviation of 0.02 for all attack efforts. No attacks are successful when the attack effort is smaller than 0.4. For the $\pi_{\text{pnn},\sigma=0.2}$, it achieves an average trajectory deviation of 0.017 for all attack efforts. No attacks are successful until the attack effort exceeds 0.6. These results show that the PNN method enhances the resistance of the driving policy to action space attacks without degrading its nominal performance.

6.3 Fine-Tuning vs. PNN

We compare the two enhancement methods under varying conditions. The evaluation metric used is the ASR. Fig. 8 presents the trend of ASR across different levels of attack effort windows. The two fine-tuned agents exhibit high ASR even under minimal attack effort, whereas PNN-enhanced agents consistently achieve lower ASR across all scenarios. These results highlight the superior robustness of the PNN approach in resisting action space attacks while preserving normal driving performance. However, its reliance on a switch mechanism that requires precise knowledge of the attack presents a major limitation for real-world deployment. This underscores the need for a more practical and adaptive defense strategy.

7 Resilient Modular Driving Agent with Learning-based Path Correction System

Based on our findings and the unique vulnerabilities of various AD designs, we propose a solution that combines a modular driving agent with a learning-based Path Correction System (PCS). It leverages the strengths of both the modular and feature-fused agent paradigms to advance the resilience of path tracking. Below, we first distill key insights from our earlier findings and then introduce the design and evaluation of PCS.

7.1 Key Insights from Observations

Section 5.3.3 demonstrates that the modular agent's feedback controllers provide strong resilience to low-effort attacks, yet lack the adaptability needed to cope with high-intensity perturbations. In contrast, the end-to-end agent, trained via DRL, exhibits greater robustness to high-effort attacks due to its exposure to diverse scenarios during training, yet suffers from poor path-tracking precision in benign conditions. The feature-fused agent achieves high accuracy under normal conditions by leveraging structured features, but may still face vulnerabilities under attack. Section 6 further reveals that adversarial training improves robustness for the end-to-end agent, but often degrades nominal driving performance due to overfitting to adversarial conditions, a limitation likely shared by the feature-fused agent.

These findings highlight the need for a resilient AD agent that addresses the trade-offs of existing designs. Motivated by this, we develop a new architecture that integrates the strengths of modular and feature-fused paradigms, while adopting a Simplex-inspired control scheme from the PNN framework to enhance nominal driving performance. Our design objectives are threefold: 1) maintain optimal performance in nominal scenarios, 2) exhibit adaptability via additional training under extreme conditions, and 3) support scalable attack detection for practical deployment.

7.2 Design of PCS

Fig. 9 illustrates the enhanced AD architecture with PCS. It preserves the modular agent's ability to maintain nominal performance and handle low-level disturbances. Meanwhile, it runs as a supplementary loop in parallel at all times or only when an attack is detected, to maintain the tracking performance in the presence of attacks. In this section, we present the evaluation where the supplementary loop runs at all times.

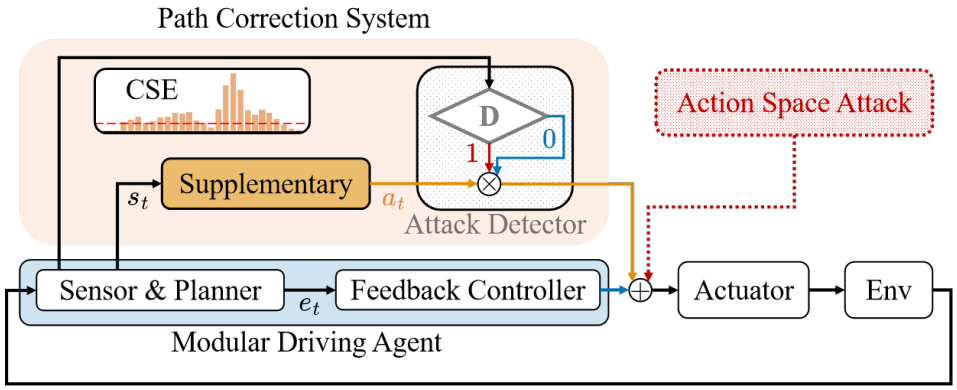


Fig. 9. Illustration of learning-based PCS. The supplementary loop is continuously active and mitigates tracking errors under the setting described in Section 7. In Section 8, a kinematic-model-based attack detector (labeled 'D') is added to improve nominal driving performance. It outputs one if the detector identifies an adversarial situation and zero otherwise. The multiplication between the supplementary loop's output and the attack detector's output represents the switching mechanism that selects whether to deactivate PCS.

The supplementary loop of PCS is designed to mitigate path tracking errors/attack impacts by bolstering the valid control signal to prevail in the contention for steering control. We use adversarial training with DRL to train the supplementary loop. The adversarial training incorporates a camera-based attack with a fixed attack budget of one, representing the strongest attack scenario where the attacker has control power equal to that of the driving system. There are numerous choices for state input and reward design of this supplementary loop, which affect the control performance, algorithm convergence speed, and computational cost/delay at run time. To improve the PCS's generalizability and reduce its complexity, we utilize distilled key features obtained from preceding modules as input. Compared with image input used in the end-to-end solutions, this method uses a lighter deep model to process precise vehicle state information. Moreover, it facilitates better adaptability across various tasks and does not need to consider common domain generalization in high-dimensional image inputs.

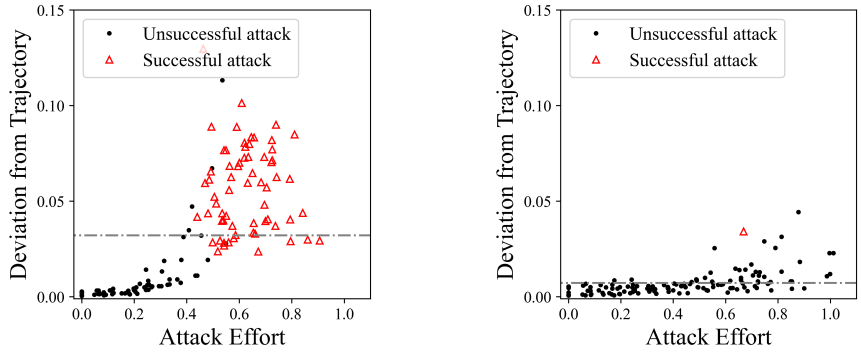
Specifically, the state transition of the supplementary loop at time step t is denoted by $< s_t, a_t, r_t, s_{t+1} >$, where $s_t \in \mathbb{R}^7$ and $s_{t+1} \in \mathbb{R}^7$ are two consecutive states, $a_t \in \mathbb{R}^1$ is the supplementary steering signal produced, and r_t is the immediate reward associated with the state transition. As illustrated in Fig. 9, the final lateral actuator command is the sum of a_t and the signal from the modular driving agent. The state s_t is defined as:

$$s_t = [e^P_{t-1},\ e^I_{t-1},\ e^D_{t-1},\ v_{t-1},\ \Delta a^{\text{lateral}}_{t-1},\ a^{\text{longitudinal}}_{t-1},\ a^{\text{lateral}}_{t-1}], \tag{8}$$

where the first three terms are the proportional, integral, and differential components of tracking error, i.e., the degree of deviation (in radians) of the vehicle velocity vector from the planned path. The remaining terms are the vehicle's speed, feedback controller output, and the actuator output (longitudinal and lateral control) at the last time step, respectively. This one-step delay formulation reflects the inherent sensing and actuation latency of real-world AD systems.

To minimize the path tracking error, we design the following reward function:

$$r_t = \begin{cases} b - a \cdot {e_t}^2, & \text{if } {e_t}^2 < \frac{b}{a}, \\ 0, & \text{otherwise}, \end{cases} \tag{9}$$



(a) Modular agent without PCS

(b) Modular agent with PCS

Fig. 10. Comparison of attack resilience between modular agents with and without PCS. The average trajectory deviation decreases significantly from 0.032 to 0.007 when combined with PCS, demonstrating a performance improvement of approximately 78%.

where ${e_t}^2$ is the squared driving error collected from the post-action driving state, determined by the angle difference between the ego vehicle's speed vector and the generated reference path. The positive constants a and b are used to define the upper limit for squared driving errors as $\frac{b}{a}$. This reward function penalizes large errors and provides a positive reward for minimizing errors. Thus, it encourages the driving model to strive for accurate trajectory following. Moreover, the model's weight updates are only allowed when the squared error falls within the boundary defined in Eq. (9), i.e., $e_t^2 < \frac{b}{a}$. We experimentally set $b = 0.5$ and $a = 1.5$ in training.

7.3 Evaluating PCS-Enhanced Modular Driving Agents

7.3.1 Resilience to action space attacks. Fig. 10(a) and Fig. 10(b) provide a visual comparison of the attack resilience between modular driving agents with and without the PCS. As depicted in Fig. 10(b), the modular agent equipped with PCS exhibits significantly enhanced resilience, evidenced by a substantially lower RMS tracking error across all levels of attack effort. Additionally, it exhibits comparable tracking performance to modular agents under low attack effort. Furthermore, when compared with the results shown in Fig. 7, the PCS-equipped agent outperforms adversarially trained end-to-end agents: it achieves significantly fewer instances of successful attack and it does not suffer the overfitting problem. Generally, the modular driving agent with PCS upholds satisfactory driving performance across various attack effort settings. It demonstrates robustness against adversarial scenarios while retaining nominal driving behaviors.

In this evaluation, even though the supplementary loop remains active, its effect on path tracking is negligible in the absence of attacks (i.e., when the attack effort is zero). This behavior can be understood from the perspective of DRL training. In nominal driving scenarios, the modular agent generates correct control signals that satisfy the reward function. As the training objectives are already met, the supplementary loop does not produce unnecessary control signals, thereby preserving the PCS's nominal performance. In contrast, during adversarial driving scenarios where the traditional controller is insufficiently robust against action space attacks, the supplementary loop adapts to minimize path-tracking errors by working alongside the feedback controller. This enables the system to develop resilience under attack conditions. The above results show that PCS is an effective defense strategy against action space attacks for modular driving agents. Moreover, findings suggest that PCS does not rely on an accurate attack detection mechanism, which could potentially simplify its implementation.

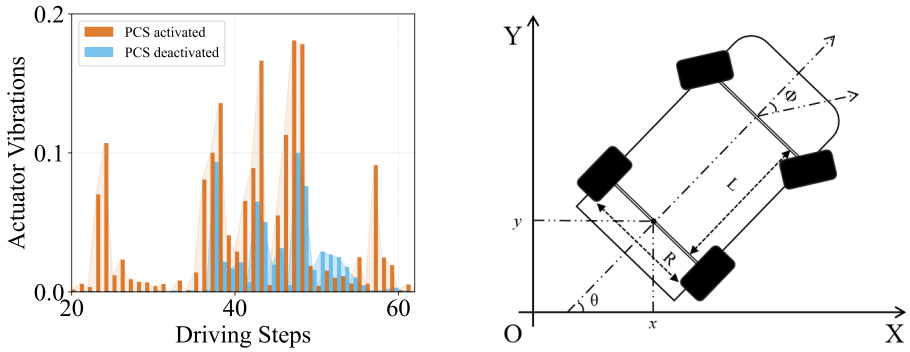
Table 3. PCS performance under simulated hardware imperfections (HIP) against a camera-based action space attacker ($\varepsilon = 1$). In this table, each *Deviation from Trajectory* is calculated over the entire episode.

HIP Type	Severity/Param	Deviation from Trajectory Avg. ± Std.	Attack Success Rate (%)
Ideal PCS	–	0.0121 ± 0.0113	0.0
Sensor Noise	$\alpha = 0.1$	0.0123 ± 0.0108	0.0
Sensor Noise	$\alpha = 0.2$	0.0153 ± 0.0139	1.8
Sensor Noise	$\alpha = 0.3$	0.0185 ± 0.0161	7.8
State Dropout	$p_{\text{drop}} = 0.05$	0.0170 ± 0.0178	8.8
State Dropout	$p_{\text{drop}} = 0.1$	0.0218 ± 0.0214	17.4
Action Delay	$p_{\text{delay}} = 0.05$	0.0134 ± 0.0120	3.6
Action Delay	$p_{\text{delay}} = 0.1$	0.0320 ± 0.0998	4.1

7.3.2 Runtime efficiency of PCS. The PCS module adopts a lightweight actor-critic architecture for fast inference and minimal resource consumption. It consists of a two-layer fully connected encoder that maps a 7-dimensional observation vector (Eq. (8)) to latent features, followed by a compact MLP-based decoder that outputs a bounded lateral control command. Both components use ReLU activations and maintain minimal parameter count. To quantify inference efficiency, we measure the computational cost of PCS in terms of floating point operations (FLOPs), a hardware-agnostic metric of complexity. For PCS, each forward pass requires only 2,552 FLOPs, 368 for the encoder and 2,184 for the decoder. Taking the Jetson AGX Orin as a representative edge platform for autonomous vehicles, with a peak throughput of 275 Tera Operations per Second (TOPS), this translates to microsecond-level overhead. Such minimal cost confirms that PCS meets the real-time requirements of safety-critical autonomous driving systems.

7.3.3 Robustness to simulated hardware imperfections. Simulators like CARLA typically assume idealized hardware conditions. In contrast, real-world AD systems must remain reliable despite hardware imperfections. To assess the robustness of PCS under more realistic conditions, we simulate three classes of hardware-induced disturbances in the control loop: 1) Sensor noise: Gaussian noise is added to each state variable s^i in Eq. (8) with standard deviation $\sigma_i = \alpha|s^i|$, where s^i denotes the i-th element of the state vector and α is a tunable parameter that controls the relative noise level, 2) State Dropout: With a probability p_{drop}, individual state dimensions are masked to zero, emulating temporary sensor or communication failures, and 3) Actuator Delay: At each control step, with probability p_{delay}, the PCS output is delayed by one step, using the previous output instead to simulate actuation lag.

Table 3 shows that PCS maintains robustness under mild to moderate sensor noise ($\alpha \leq 0.2$), with minimal trajectory deviation and near-zero attack success. However, larger noise ($\alpha = 0.3$) reduces both control accuracy and robustness. State dropout has a more pronounced effect: small dropout probabilities already increase both tracking error and attack success. This sensitivity stems from the compact and information-dense nature of the state representation, where each variable carries critical control information, unlike image inputs that offer spatial redundancy. For actuator delay, it is notable that increasing p_{delay} from 0.05 to 0.1 leads to higher tracking error, yet the attack success rate remains relatively stable (from 3.6% to 4.1%). This suggests that PCS can continuously compensate for delayed actuation and recover over time, thereby maintaining resilience against the attacker. These results indicate that while PCS can tolerate perturbed or delayed inputs, reliable deployment in real-world settings requires complete access to critical state variables.



(a) Impact of PCS on actuator

(b) Kinematic model

Fig. 11. (a) Actuator vibrations are higher with PCS activated (orange bars) than when it is off (blue bars). (b) The kinematic model of a rear-wheel-drive vehicle.

8 Attack Detector in PCS

While PCS achieves minimal tracking error in the absence of attacks, continuous activation of its supplementary loop can introduce unnecessary oscillations, potentially degrading passenger comfort (see Fig. 11(a)). To address this, we incorporate an attack detector for PCS's activation.

8.1 Design of the Attack Detector

We propose a kinematic-model-based detector that triggers an alarm when the estimated vehicle posture deviates significantly from sensor measurements. Since moving vehicles exhibit highly nonlinear and time-varying dynamics, accurate posture estimation requires models that can capture such complexity, typically formulated in discrete time for computational tractability. For a rear-wheel-drive vehicle, the state representation illustrated in Fig. 11(b) captures lateral and longitudinal motion along with yaw dynamics. The vehicle state is represented by its location and orientation (x, y, θ). The vehicle's control variables include longitudinal velocity and steering angle (v, ϕ). The kinematic model without noises and attacks is described as [19]:

$$x_{t+1} = x_t + T(v_t)\cos\theta_t, \tag{10}$$

$$y_{t+1} = y_t + T(v_t)\sin\theta_t, \tag{11}$$

$$\theta_{t+1} = \theta_t + T(\frac{v_t}{L})\tan\phi_t, \tag{12}$$

$$\Delta\theta_{t+1} = \theta_{t+1} - \theta_t, \tag{13}$$

where L represents the wheelbase and T represents the control iteration interval. We set $L = 3\,\text{m}$ and $T = 0.1\,\text{s}$ for our simulation. This provides a real-time estimate of vehicle state when the measurement variables are not corrupted.

We use the Cumulative Sum Error (CSE) as the attack detection metric. CSE at a given timestamp t is calculated as follows: $\text{CSE}(t) = \sum_{i=t-k}^{t} |\Delta\hat{\theta}_i - \Delta\theta_i|$. It is the accumulation of absolute discrepancy between the predicted heading difference $\Delta\hat{\theta}$ by Eq. (13), and the measured angle deviation $\Delta\theta$ by IMU over a time window k. If the CSE exceeds a pre-defined threshold H, an alarm triggers the supplementary PCS loop, as illustrated by Fig. 9. Fig. 12 shows the tracking performance and illustrates CSE by the call-out figure with $k = 3$. Experimental results show that the kinematic model closely follows the vehicle's heading in real time. However, during rapid maneuvers, prediction

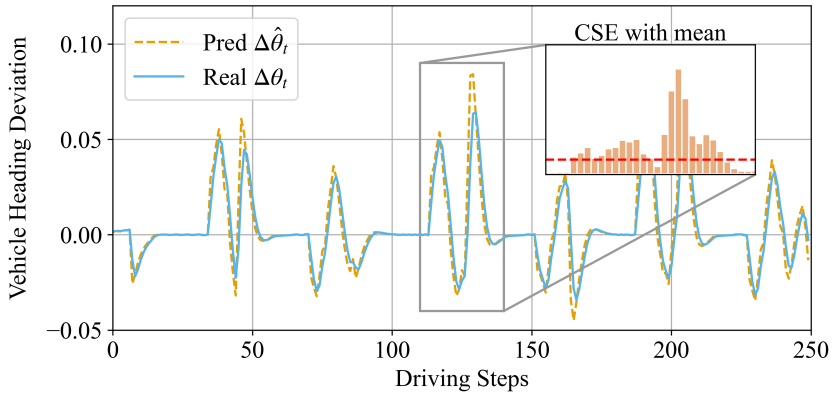


Fig. 12. The blue solid line corresponds to the ground truth, and the yellow dashed line represents the predicted orientation difference. In the call-out window, the orange bar displays the Cumulative Sum Error (CSE) with $k = 3$, while the red dashed line represents the mean of the CSE over the entire drive.

Table 4. Performance of the CSE detector under various c parameter settings.

Detector	Oracle	c=0.3	**c=1.0**	c=2.0	c=3.0
True Positive Rate	1.00	0.96	**0.94**	0.86	0.84
False Alarm Rate	0.00	0.77	**0.42**	0.25	0.20
Mean Detect Delay	1	2	**3.33**	5.2	5.5
Max Detect Delay	1	2	**5**	8	8
Mean Absolute Jerk	5.96	8.77	**7.27**	8.04	-

errors can increase, making it difficult to distinguish natural prediction inaccuracies from attack-induced deviations. Therefore, setting the detection threshold H is critical to balancing false alarms and true positives, as further discussed in the next section.

8.2 Detection Threshold and Comfort Trade-off

We define the detection threshold as $H = c \times \text{mean(CSE)}$. Lower values of c increase detection sensitivity but may lead to more false alarms, while higher values reduce false positives at the cost of delayed or missed detections. Since false alarms have limited safety consequences, we prioritize sensitivity by lowering c to avoid missing detection of potential attacks. To quantify the passenger discomfort, we use the vehicle jerk metric J [14]: $J = \frac{d^2 v_y}{dt^2} = \frac{da_y}{dt}$, where v_y is the vehicle lateral speed, a_y is the lateral acceleration resulting from steering adjustments, and J corresponds to the lateral jerk experienced throughout the driving.

Table 4 presents the performance of PCS with attack detectors under adversarial driving scenarios. During the evaluation steps, the records of the detection results and the ground truth information regarding the presence or absence of an attack are collected. An oracle detector with one-step delay provides the upper-bound baseline, which the CSE detector aims to approach through fine-tuning the parameter c. As depicted in Table 4, a lower threshold (c) increases sensitivity but may trigger more false alarms, resulting in fluctuating control and reduced comfort, as reflected by higher mean absolute jerk when $c = 0.3$. Conversely, a high threshold delays detection and can allow attacks to cause collisions, as shown by the red-highlighted column when $c = 3$. We find $c = 1.0$ achieves a favorable balance, providing robust attack detection with minimal impact on passenger comfort.

9 Discussion

This section outlines the assumptions of our threat model, evaluates the generalizability of both attack and defense, and discusses the performance and practicality of PCS.

9.1 Summary and Justification of Assumptions

Our study is based on several assumptions regarding system modeling, the attack capabilities, and the defense requirements. Below, we review and justify these assumptions.

- **System model assumptions:** Experiments are conducted in the CARLA simulator, which offers a high-fidelity yet simplified approximation of real-world vehicle dynamics. CARLA is widely used in AD research, enabling controlled, repeatable, and safe evaluation of attack and defense strategies that are otherwise impractical or unsafe on real vehicles. While simulators abstract away some real-world uncertainties (e.g., hardware aging, sensor drift), this setup aligns with standard practices in AD and cybersecurity research, where sim-to-real transfer remains a known challenge but simulation provides a critical foundation.
- **Attack model assumptions:** As outlined in Section 4.1, we assume the attacker has black-box access, can inject bounded perturbations, and can obtain either rich (camera-based) or partial (IMU-based) observations of vehicle state via on-board sensors. The black-box setting reflects common real-world constraints, where attackers lack internal knowledge but can train strategies using simulators or digital twins. Bounded perturbations reflect practical limits on stealth and actuator authority. The assumed sensing modalities represent plausible attacker capabilities and are further discussed in Section 9.2.
- **Defense model assumptions:** The proposed PCS defense requires access to modular agent state features, such as path deviation, actuator history, and vehicle speed, as inputs to its correction mechanism. These signals are standard outputs of modular AD controllers and commonly logged for safety monitoring. PCS does not rely on knowledge of the attacker's internal state or perfect anomaly detection. Instead, it is designed to tolerate imperfect detection, supporting robust deployment in real-world, safety-critical systems.

9.2 Feasibility of Action Space Attacks and Sim-to-Real Generalization

We assess the feasibility of launching action space attacks in real-world settings and the extent to which our DRL-based attack generalizes beyond simulation. To execute an action space attack, the adversary must gain control over the vehicle's actuator commands. In modern vehicles, actuator commands are delivered by electronic control units (ECUs) through two primary channels: 1) digital communication over message-based protocols such as the Controller Area Network (CAN) bus, and 2) analog signaling via Pulse Width Modulation (PWM), which modulates signal duty cycles to adjust actuator speed or position. Together, these expose two critical attack surfaces: the CAN bus for digital signals and the wiring for PWM analog signals.

- **CAN bus interference:** As noted in prior work [7], the CAN protocol lacks built-in source authentication, allowing compromised ECUs to inject spoofed packets. Attackers may leverage remote access or physical compromise to execute such injections [30, 47]. That said, modern vehicles are increasingly equipped with intrusion detection systems, which may limit the effectiveness of such attacks.
- **Intentional electromagnetic interference (IEMI):** Alternatively, attackers may exploit physical-layer vulnerabilities by emitting electromagnetic signals near wiring to induce errors in PWM signals [13, 39]. Although theoretically possible, IEMI presents practical challenges. Executing it would require precise setup, including access to the vehicle wiring

and the installation of devices. Potential situations may involve collusion with routine vehicle maintenance providers to install all required devices, including the camera and IMU sensors.

Another key challenge is the well-known generalization issue for DRL research, including ours. Although CARLA provides high-fidelity vehicle modeling, simulators cannot fully capture real-world complexity. Factors such as sensor noise, hardware wear, and environmental variability can introduce discrepancies that hinder transfer from simulation to real-world deployment. While small-scale robotic platforms have been used in prior work [35], this approach is impractical in our case due to the need for high-speed driving scenarios and the risk of frequent crashes during testing. To narrow the sim-to-real gap, future implementations of action space attacks should consider:

- Match vehicle models and control policies to real-world targets (e.g., Tesla Model 3 in CARLA).
- Include realistic conditions in training, such as weather, lighting, friction, etc. [5, 16].
- If feasible, conduct controlled physical testing in a closed environment.

While full transferability remains a significant challenge, our goal is not to claim immediate deployability. Rather, this work emphasizes the feasibility of action space attacks, reveals architecture-specific vulnerabilities, and motivates the development of practical and lightweight defenses.

9.3 Broader Applicability to Other Scenarios

Although our experiments focus on lane changes and overtaking within a freeway driving scenario, we clarify that the proposed action space attack and PCS frameworks are not limited to this context. Many AD tasks, such as intersection negotiation, obstacle avoidance, or merging, are fundamentally trajectory tracking problems, which lie at the core of autonomous vehicle control.

For the action space attacker, extension to new driving scenarios is conceptually straightforward but requires adaptation to maximize effectiveness. Specifically, the attack model must be retrained within the new context to learn how to exploit scenario-specific vulnerabilities. This typically involves modifying the reward function to reflect the critical events or failure modes relevant to the new task. For instance, the reward design may prioritize collisions at intersections or unsafe merges. While the overall attack methodology remains unchanged, the reward formulation and training environment should be tailored for each scenario.

By contrast, the PCS framework is designed to use modular state features, such as path deviation, actuator commands, and vehicle speed, as input. Since trajectory tracking is central to most driving tasks, these features remain broadly applicable across scenarios. As a result, PCS can be directly applied to new driving tasks *without requiring any changes* to its architecture or input representation. The only exception occurs when new elements, such as traffic light phases, significantly affect trajectory tracking. In such cases, it may be beneficial to augment the input with additional features that represent these elements. This extension ensures that PCS maintains robust tracking performance and safety in the face of scenario-specific complexities. In the absence of such factors, PCS remains task-agnostic and applicable without modification.

10 Conclusion

This paper examined the resilience of different autonomous driving (AD) systems to action space attacks. We demonstrated that DRL-based black-box adversaries can induce side collisions across modular, end-to-end, and feature-fused agents, with varying degrees of vulnerability. However, existing defenses by adversarial training suffer from overfitting or rely on attack patterns and have shown limited effectiveness. To address these gaps, we proposed a lightweight learning-based Path Correction System (PCS) integrated with a modular agent, achieving a 78% reduction in path-tracking deviation under attack. We further introduced a kinematic-based attack detector to

improve nominal driving performance. Our results highlight the effectiveness and practicality of the proposed defense in enhancing the robustness of AD systems against actuator-level threats.

Acknowledgments

This research/project is supported by the National Research Foundation, Singapore, under its AI Singapore Programme (AISG Award No: AISG4-GC-2023-006-1B), its National Satellite of Excellence in Trustworthy Software Systems (NSOE-TSS) office under the Trustworthy Computing for Secure Smart Nation Grant (TCSSNG) award no. NSOE-TSS2020-01, and its Campus for Research Excellence and Technological Enterprise (CREATE) program.

References

[1] Ali Alizadeh, Majid Moghadam, Yunus Bicer, Nazim Kemal Ure, Muharrem Ugur Yavas, and Can Kurtulus. 2019. Automated Lane Change Decision Making using Deep Reinforcement Learning in Dynamic and Uncertain Highway Environment. In *ITSC*. IEEE, 1399–1404.

[2] Adith Boloor, Karthik Garimella, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. 2020. Attacking vision-based perception in end-to-end autonomous driving models. *J. Systems Architecture* 110 (2020), 101766.

[3] Adith Boloor, Xin He, Christopher D. Gill, Yevgeniy Vorobeychik, and Xuan Zhang. 2019. Simple Physical Adversarial Examples against End-to-End Autonomous Driving Models. In *15th IEEE International Conference on Embedded Software and Systems, ICESS 2019, Las Vegas, NV, USA, June 2-3, 2019*. IEEE, Las Vegas, NV, USA, 1–7.

[4] Maxime Bouton, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Mykel J. Kochenderfer. 2020. Reinforcement Learning with Iterative Reasoning for Merging in Dense Traffic. In *23rd IEEE International Conference on Intelligent Transportation Systems, ITSC 2020, Rhodes, Greece, September 20-23, 2020*. IEEE, Rhodes, Greece, 1–6.

[5] Peide Cai, Xiaodong Mei, Lei Tai, Yuxiang Sun, and Ming Liu. 2020. High-Speed Autonomous Drifting With Deep Reinforcement Learning. *IEEE Robotics Autom. Lett.* 5, 2 (2020), 1247–1254.

[6] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. 2019. Adversarial Sensor Attack on LiDAR-based Perception in Autonomous Driving. In *CCS*. ACM, 2267–2281.

[7] Paul Carsten, Todd R. Andel, Mark Yampolskiy, and Jeffrey Todd McDonald. 2015. In-Vehicle Networks: Attacks, Vulnerabilities, and Proposed Solutions. In *Proceedings of the 10th Annual Cyber and Information Security Research Conference, CISR '15, Oak Ridge, TN, USA, April 7-9, 2015*, Joseph P. Trien, Stacy J. Prowell, Robert A. Bridges, and John R. Goodall (Eds.). ACM, 1:1–1:8.

[8] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2019. Learning by Cheating. In *3rd Annual Conference on Robot Learning, CoRL 2019, Osaka, Japan, October 30 - November 1, 2019, Proceedings (Proceedings of Machine Learning Research, Vol. 100)*, Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (Eds.). PMLR, Osaka, Japan, 66–75. http://proceedings.mlr.press/v100/chen20a.html

[9] Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka. 2019. Model-free Deep Reinforcement Learning for Urban Autonomous Driving. In *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019, Auckland, New Zealand, October 27-30, 2019*. IEEE, Auckland, New Zealand, 2765–2771.

[10] Li Chen, Penghao Wu, Kashyap Chitta, Bernhard Jaeger, Andreas Geiger, and Hongyang Li. 2024. End-to-End Autonomous Driving: Challenges and Frontiers. *IEEE Trans. Pattern Anal. Mach. Intell.* 46, 12 (2024), 10164–10183.

[11] Hongjun Choi, Wen-Chuan Lee, Yousra Aafer, Fan Fei, Zhan Tu, Xiangyu Zhang, Dongyan Xu, and Xinyan Deng. 2018. Detecting Attacks Against Robotic Vehicles: A Control Invariant Approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang (Eds.). ACM, 801–816.

[12] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. 2019. Exploring the Limitations of Behavior Cloning for Autonomous Driving. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, Seoul, Korea (South), 9328–9337.

[13] Gökçen Yilmaz Dayanikli, Rees R. Hatch, Ryan M. Gerdes, Hongjie Wang, and Regan Zane. 2020. Electromagnetic Sensor and Actuator Attacks on Power Converters for Electric Vehicles. In *2020 IEEE Security and Privacy Workshops, SP Workshops, San Francisco, CA, USA, May 21, 2020*. IEEE, San Francisco, CA, USA, 98–103.

[14] Ksander N de Winkel, Tugrul Irmak, Riender Happee, and Barys Shyrokau. 2023. Standards for passenger comfort in automated vehicles: Acceleration and jerk. *Applied Ergonomics* (2023).

[15] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio M. López, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In *CoRL (Proceedings of Machine Learning Research, Vol. 78)*. PMLR, 1–16.

[16] Linxi Fan, Guanzhi Wang, De-An Huang, Zhiding Yu, Li Fei-Fei, Yuke Zhu, and Animashree Anandkumar. 2021. SECANT: Self-Expert Cloning for Zero-Shot Generalization of Visual Policies. In *ICML (Proceedings of Machine Learning Research, Vol. 139)*. PMLR, 3088–3099.

[17] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. 2018. A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[18] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. 2020. Adversarial Policies: Attacking Deep Reinforcement Learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, Addis Ababa, Ethiopia. https://openreview.net/forum?id=HJgEMpVFwB

[19] Pinyao Guo, Hunmin Kim, Le Guan, Minghui Zhu, and Peng Liu. 2017. VCIDS: Collaborative Intrusion Detection of Sensor and Actuator Attacks on Connected Vehicles. In *Security and Privacy in Communication Networks - 13th International Conference, SecureComm 2017, Niagara Falls, ON, Canada, October 22-25, 2017, Proceedings (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 238)*, Xiaodong Lin, Ali A. Ghorbani, Kui Ren, Sencun Zhu, and Aiqing Zhang (Eds.). Springer, Niagara Falls, ON, Canada, 377–396.

[20] Rodrigo Gutiérrez-Moreno, Rafael Barea, Elena López Guillén, Javier Araluce, and Luis Miguel Bergasa. 2022. Reinforcement Learning-Based Autonomous Driving at Intersections in CARLA Simulator. *Sensors* 22, 21 (2022), 8373.

[21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, Stockholm, Sweden, 1856–1865. http://proceedings.mlr.press/v80/haarnoja18b.html

[22] Davor Hrovat, Stefano Di Cairano, H. Eric Tseng, and Ilya V. Kolmanovsky. 2012. The development of Model Predictive Control in automotive industry: A survey. In *Proceedings of the IEEE International Conference on Control Applications, CCA 2012, Dubrovnik, Croatia, October 3-5, 2012*. IEEE, Dubrovnik, Croatia, 295–302.

[23] David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. 2018. Navigating Occluded Intersections with Autonomous Vehicles Using Deep Reinforcement Learning. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, Brisbane, Australia, 2034–2039.

[24] Saurabh Jha, Subho S. Banerjee, Timothy Tsai, Siva Kumar Sastry Hari, Michael B. Sullivan, Zbigniew T. Kalbarczyk, Stephen W. Keckler, and Ravishankar K. Iyer. 2019. ML-Based Fault Injection for Autonomous Vehicles: A Case for Bayesian Fault Injection. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019, Portland, OR, USA, June 24-27, 2019*. IEEE, Portland, OR, USA, 112–124.

[25] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. 2020. PhysGAN: Generating Physical-World-Resilient Adversarial Examples for Autonomous Driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, Seattle, WA, USA, 14242–14251.

[26] Xian Yeow Lee, Yasaman Esfandiari, Kai Liang Tan, and Soumik Sarkar. 2021. Query-based targeted action-space adversarial policies on deep reinforcement learning agents. In *ICCPS '21: ACM/IEEE 12th International Conference on Cyber-Physical Systems, Nashville, Tennessee, USA, May 19-21, 2021*, Martina Maggio, James Weimer, Mohammad Al Farque, and Meeko Oishi (Eds.). ACM, Nashville, Tennessee, USA, 87–97.

[27] Yen-Chen Lin, Zhang-Wei Hong, Yuan-Hong Liao, Meng-Li Shih, Ming-Yu Liu, and Min Sun. 2017. Tactics of Adversarial Attack on Deep Reinforcement Learning Agents. In *IJCAI*. ijcai.org, 3756–3762.

[28] Liangkai Liu, Sidi Lu, Ren Zhong, Baofu Wu, Yongtao Yao, Qingyang Zhang, and Weisong Shi. 2021. Computing Systems for Autonomous Driving: State of the Art and Challenges. *IEEE Internet Things J.* 8, 8 (2021), 6469–6486.

[29] Xiao-Yang Liu, Zechu Li, Zhuoran Yang, Jiahao Zheng, Zhaoran Wang, Anwar Walid, Jian Guo, and Michael I Jordan. 2021. ElegantRL-Podracer: Scalable and elastic library for cloud-native deep reinforcement learning. *NeurIPS, Workshop on Deep Reinforcement Learning* (2021).

[30] Siti-Farhana Lokman, Abu Talib Othman, and Muhammad-Husaini Abu-Bakar. 2019. Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review. *EURASIP Journal on Wireless Communications and Networking* 2019, 1 (2019), 1–17.

[31] Mohammad Naghnaeian, Nabil Hirzallah, and Petros G. Voulgaris. 2015. Dual rate control for security in cyber-physical systems. In *54th IEEE Conference on Decision and Control, CDC 2015, Osaka, Japan, December 15-18, 2015*. IEEE, Osaka, Japan, 1415–1420.

[32] Brian Paden, Michal Cáp, Sze Zheng Yong, Dmitry S. Yershov, and Emilio Frazzoli. 2016. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Trans. Intell. Veh.* 1, 1 (2016), 33–55.

[33] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos A. Theodorou, and Byron Boots. 2020. Imitation learning for agile autonomous driving. *Int. J. Robotics Res.* 39, 2-3 (2020).

[34] Óscar Pérez-Gil, Rafael Barea, Elena López Guillén, Luis Miguel Bergasa, Carlos Gómez Huélamo, Rodrigo Gutiérrez, and Alejandro Díaz. 2022. Deep reinforcement learning based control for Autonomous Vehicles in CARLA. *Multim. Tools Appl.* 81, 3 (2022), 3553–3576.

[35] Raul Quinonez, Jairo Giraldo, Luis E. Salazar, Erick Bauman, Alvaro A. Cárdenas, and Zhiqiang Lin. 2020. SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants. In *USENIX Security Symposium*. USENIX Association, 895–912.

[36] Jikun Rong and Nan Luan. 2020. Safe reinforcement learning with policy-guided planning for autonomous driving. In *2020 IEEE ICMA*. IEEE.

[37] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive Neural Networks. *CoRR* abs/1606.04671 (2016).

[38] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. 2021. Dirty Road Can Attack: Security of Deep Learning based Automated Lane Centering under Physical-World Attack. In *USENIX Security Symposium*. USENIX Association, 3309–3326.

[39] Jayaprakash Selvaraj. 2018. *Intentional electromagnetic interference attack on sensors and actuators.* Ph. D. Dissertation. Iowa State University.

[40] Lui Sha. 2001. Using Simplicity to Control Complexity. *IEEE Softw.* 18, 4 (2001), 20–28.

[41] Kai Liang Tan, Yasaman Esfandiari, Xian Yeow Lee, Aakanksha, and Soumik Sarkar. 2020. Robustifying Reinforcement Learning Agents via Action Space Adversarial Training. In *2020 American Control Conference, ACC 2020, Denver, CO, USA, July 1-3, 2020*. IEEE, Denver, CO, USA, 3959–3964.

[42] Siyu Teng, Long Chen, Yunfeng Ai, Yuanye Zhou, Zhe Xuanyuan, and Xuemin Hu. 2023. Hierarchical Interpretable Imitation Learning for End-to-End Autonomous Driving. *IEEE Trans. Intell. Veh.* 8, 1 (2023), 673–683.

[43] Antonio Torralba and Alexei A. Efros. 2011. Unbiased look at dataset bias. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*. IEEE Computer Society, Colorado Springs, CO, USA, 1521–1528.

[44] Hado van Hasselt, Arthur Guez, and David Silver. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI*. AAAI Press, 2094–2100.

[45] Yuting Wu, Xin Lou, Pengfei Zhou, Rui Tan, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. 2023. Susceptibility of Autonomous Driving Agents to Learning-Based Action-Space Attacks. In *53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2023 - Workshops, Porto, Portugal, June 27-30, 2023*. IEEE, Porto, Portugal, 76–83.

[46] Yi Xiao, Felipe Codevilla, Akhil Gurram, Onay Urfalioglu, and Antonio M. López. 2022. Multimodal End-to-End Autonomous Driving. *IEEE Trans. Intell. Transp. Syst.* 23, 1 (2022), 537–547.

[47] Xugui Zhou, Anna Schmedding, Haotian Ren, Lishan Yang, Philip Schowitz, Evgenia Smirni, and Homa Alemzadeh. 2022. Strategic Safety-Critical Attacks Against an Advanced Driver Assistance System. In *52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2022, Baltimore, MD, USA, June 27-30, 2022*. IEEE, Baltimore, MD, USA, 79–87.

[48] Mykola Zhuk, Volodymyr Kovalyshyn, Yurii Royko, and Khrystyna Barvinska. 2017. Research on Drivers' Reaction TIME in Different Conditions. *Eastern-European Journal of Enterprise Technologies* 2, 3 (2017), 24–31.