Susceptibility of Autonomous Driving Agents to Learning-Based Action-Space Attacks

Yuting Wu*, Pengfei Zhou†, Xin Lou‡, Rui Tan*, Zbigniew T. Kalbarczyk§, Ravishankar K. Iyer§
*Nanyang Technological University, †University of Pittsburgh, ‡Singapore Institute of Technology
‡Illinois at Singapore, §University of Illinois at Urbana-Champaign

***Abstract*—Intelligent vehicles with increasing complexity face cybersecurity threats. This paper studies action-space attacks on autonomous driving agents that make decisions using either a traditional modular processing pipeline or the recently proposed end-to-end driving model obtained via deep reinforcement learning (DRL). Such attacks alter the actuation signal and pose direct risks to the vehicle's state. We formulate the attack construction as a DRL problem based on the input from either an extra camera or inertial measurement unit deployed. The attacks are designed to lurk until a safety-critical moment arises and cause a side collision upon activation. We analyze the behavioral differences between two driving agents when subjected to action-space attacks and demonstrate the superior resilience of the modular processing pipeline. We further investigate the performance and limitations of two enhancement methods, i.e., adversarial training through fine-tuning and progressive neural networks. The result offers valuable insights into vehicle safety from the viewpoints of both the assailant and the defender and informs the future design of autonomous driving systems.**

I. Introduction

The complexity and connectivity of autonomous driving (AD) systems increase as vehicle autonomy improves, expanding the scope for potential targets of malicious attacks. Attackers could potentially target various points of the vehicle such as sensor inputs. Furthermore, attackers may find the actuation component to be an appealing target. By directly affecting the actuator units, the attack could bypass the potential defense along the system from perception to control level and generates direct impacts on the vehicle's state. Recent studies have shown that autonomous driver assistance systems (ADAS) are susceptible to safety concerns regarding action-space attacks. These attacks can be achieved through model-based approaches that rely on in-vehicle data or knowledge of the vehicle's kinematics, resulting in a demanding form of white-box attack [1], [2]. Alternatively, these attacks can be formed in a black-box setting, disregarding the knowledge of the vehicle's internal workings. However, they have mostly been studied in toy models like OpenAI Gym [3], [4] and Mathworks [5], which may not fully capture the complexities of real-world driving. Additionally, previous studies have been limited to specific ADAS designs and have not explored the impact of these attacks across different AD architectures, highlighting a gap in current research.

To fill this gap, our study investigates the impact of the attack across two types of AD agents: 1) the traditional modular processing pipeline and 2) the recently proposed end-to-end policy model obtained via DRL. Modular driving pipelines use a hierarchy of modules to achieve driving goals, with each module addressing a sub-task. The end-to-end driving agent maps sensor inputs to the actuation signal through a single policy model, learned from a reward function that aggregates multiple driving goals. We hypothesize that the differences in design between the two solutions will result in distinct responses to action-space attacks. Through an analysis of the results, we can gain valuable insights into how to make AD designs more resilient to such attacks.

In this study, we introduce learning-based action-space attacks that inject additive perturbations to the steering of a victim vehicle within predetermined limits. The goal of the attack is to cause side collisions with other vehicles on the road during safety-critical moments, such as lane changing and overtaking. Previous research has demonstrated its practical settings through different means. For instance, intentional electromagnetic interference (IEMI) can target analog signals, while intrusive interference in digital messages over the control area network (CAN) bus has also been shown to be effective [1], [6]–[8]. For the realism of the attack, we assume that: 1) the model architecture of the driving agent is unavailable to the attacker, and 2) the driving agent's sensor readings are not accessible to the attacker. The black-box setting in the first assumption reduces the requirement for launching the attack. The second assumption stems from the fact that real-time data is usually protected. Furthermore, we design the attack to have limited access to the actuation, ensuring that the vehicle's thrust unit remains unaffected while retaining partial control over the steering unit. This enables the ego vehicle to brake or utilize its remaining control capability and avoid a collision, making the problem more challenging.

Strategic execution of attacks requires intelligent decision-making. In this study, we employ DRL to develop attack policies under a black-box setting, with input from an extra camera or an inertial measurement unit (IMU). A compact and low-cost camera is sufficient for detecting nearby vehicles but requires a location with a good field of view, which may attract human attention. An IMU that records driving movements is virtually unnoticeable but less informative and

[1]This project is supported by the National Research Foundation, Singapore, and the National University of Singapore through 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 in part by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program.

interpretative than images. Although physical access is required to install a third-party sensor, such opportunities are not rare, such as during maintenance by an auto care provider colluding with the adversary. Our extensive evaluation in the CARLA [9] gives the following key observations: 1) the camera-based attack outperforms the IMU-based attack in terms of attack success rate and the reduction of driving performance, indicating a trade-off between attack precision and covertness; and 2) the modular processing pipeline shows greater resilience than the end-to-end driving agent in terms of trajectory following accuracy. These results are attributed to the modular processing pipeline's focus on the path following and its timely rectification provided by the local feedback control mechanisms. Furthermore, the end-to-end driving agent's reward shaping, which incorporates multiple optimization objectives, may compromise its steering precision to satisfy other objectives, thereby creating an opportunity for attacks to exploit.

Prior research has indicated that adversarial training can improve the policy's robustness against action-space attacks [3], [10]. In this study, we investigate the efficacy and limitations of this approach by utilizing fine-tuning and progressive neural networks (PNN) to enhance our driving policy. Although adversarial training with fine-tuning improves driving performance in the presence of attacks, it leads to degraded performance in the absence of the attack due to overfitting adversarial cases. Tuning the ratio of training cases with and without attacks is crucial, but only offers a palliative solution. We implement the PNN approach to address this issue by using the first column for regular performance and the second column for handling adversarial scenarios. Our results show that the driving agent with PNN outperforms the one with fine-tuning when facing attacks and effectively overcomes the catastrophic forgetting problem. Nevertheless, it still requires prior knowledge of the attacker's strategy, which may limit its practical application.

This paper makes the following contributions:

- We propose a learning-based approach to construct action-space attacks against AD systems in a black-box setting. To our knowledge, it is the first time such an approach has been discussed in the context of AD.
- We present an attack policy causing a side collision of the ego vehicle based on camera, and a 'learning-from-teacher' structure is proposed to transfer learned policy to a different form based on less informative IMU input.
- We assess driving agents' resilience to action-space attacks and find that the modular agent embedded with a feedback controller, is more resilient than the end-to-end agent that uses reward shaping for multiple objectives. This insight can inform the design of AD agents utilizing AI technology, particularly in identifying the components that may enhance vehicle safety and security.
- We evaluate the effectiveness of adversarial training with fine-tuning and PNN in enhancing driving agents. The analysis provides a comprehensive understanding of the limitations and potential of these two methods.

This paper is organized as follows: In Section II, we introduce the background and related work. Section III presents the system model. Section IV describes the approach for constructing action-space attacks. The evaluation results are presented in Section V. Section VI demonstrates the model enhancement. Finally, we conclude the paper in Section VII.

II. Background and Related Work

A. Autonomous Driving Approaches

Typically, AD approaches can be divided into two categories: (i) *modular driving pipelines* and (ii) *end-to-end driving agents*.

Modular driving pipeline: The modular driving pipeline is based on a decision-making hierarchy, usually including route planning, behavioral layer, motion planner, and local feedback control [11]. Since a consensus on the optimal pipeline has not yet been achieved, the hierarchy details may vary depending on specific requirements. In general, its modular structure provides clear interpretability for maintenance purposes. However, decomposing the driving into multiple tasks increases research and development costs.

End-to-end driving agent: Due to the high development cost of modular driving pipelines, the end-to-end driving approaches as a competitor are gaining considerable research interest. By replacing the modules with a single policy model, the end-to-end driving policy maps raw inputs (e.g., images) directly to action distributions. This simplified architecture eases implementation. Usually, imitation learning [12] (IL) or DRL is used to train an end-to-end policy. Although IL converges fast by utilizing expert data for supervised learning, it is hard to outperform its teacher [13]. With sufficient training, the DRL-based end-to-end policy may efficiently deal with complicated tasks, such as lane changing and collision avoidance [14]–[16]. Despite recent advancements in DRL-based AD, it still has several challenges such as lack of generalizability [17] and reproducibility issues. In this paper, we apply DRL for end-to-end freeway driving. To gain a comprehensive understanding of this evolving approach, it is important to proactively study its cybersecurity concerns, even though end-to-end AD is still in its early stages.

B. DRL and Its Security

DRL has shown impressive capabilities in solving problems with high dimensions in state space. It has been applied to address various AD tasks, including lane keeping [18], lane changing [19], ramp merging [20], and intersection navigating [21]. However, DRL lacks safety guarantees, which is essential for real-world autonomy. Despite continuous efforts to develop safe-DRL for AD [22], [23], its safety remains a question and becomes worse when facing malicious attacks. Adversarial attacks on DRL agents can be categorized into *white-box attacks* or *black-box attacks*. The former requires the details of the target model, while the latter only requires access to the executable agent. Attacks can also be categorized as *state-space attacks*, targeting agent inputs, or*action-space attacks*, targeting agent outputs. Most state-space attacks are carried

(a) (b)

Fig. 1. (a) The traffic scenario created for the lane changing and overtaking tasks in CARLA. Green arrows indicate the safe and legal driving waypoints generated by a path planner for the driving agent to follow. (b) The victim vehicle side collides with an NPC vehicle. The black dash line indicates the driving route lasting for 3 seconds.

out by tampering with the input image at the pixel level [24] or manipulating the environment observed by the agent [25], while action-space attacks studied in [3] directly alter the agent output and hijack the system to a state desired by the attacker. DRL agent is a regression model that predicts action values and interacts with the environment. Thus, the attack impact should be considered when attacked, e.g., rules violations or game losses. However, the end-to-end AD scenario is not a simple win-or-lose game but a safety-critical issue. To create such a situation, we target creating collisions between vehicles.

III. System Model

In this section, we describe the experimental setups for evaluating the impact of action-space attacks on AD using CARLA 0.9.11 [9]. We provide details on the model designs, performance characteristics, and scenario configurations of both the modular driving pipeline and the end-to-end driving agent used in our experiments.

A. Traffic Scenario Setup

We construct the driving scenario on a freeway in CARLA Town 4 Road 23 without traffic lights or intersections, as shown in Fig. 1(a), allowing a focus on lane-changing and overtaking tasks. The ego vehicle in green travels at a high reference speed (16m/s) and needs to pass six NPC vehicles in yellow moving with a slower reference speed (6m/s) within limited steps (180 steps), with each step lasting 0.1 seconds.

B. Modular Driving Pipeline

We use CARLA Autopilot to serve as our modular driving pipeline, which plans feasible waypoints based on map information, makes real-time driving decisions, and applies proportional–integral–derivative (PID) controllers to trace planned routes. The modular pipeline uses longitude and latitude PID controllers to calculate the variation of throttle and steering needed to follow the generated path. The actuation variation at each time step is limited by a constant ε in order to prevent sudden changes in driving action. Modular driving agents can be configured accordingly to different driving scenarios. To eliminate any risk of collisions, a cautious driving agent would be a foolproof system that always drives at a slow speed, maintains a long safety distance from the nearby vehicle(s), and never changes lanes or overtake. However, this may block the way and pose an additional safety risk to the rear end. To match the specific driving scenario in this paper (freeway driving with lane changing and overtaking), we have tuned the driving agent to an aggressive mode: the driving agent is configured with a typical reference speed (16m/s) accompanied by a set of commensurate PID controller parameters, shorter following distance allowing more decisive lane changing and overtaking, and permission to overtake in all lanes. This configuration results in desirable driving behavior during evaluation, with the agent passing all NPC vehicles without collision and following the predefined trajectory accurately.

C. DRL-based End-to-end Driving Agent

We employ the advanced DRL algorithm, soft actor-critic [26], to construct the driving policy π_v. The driving agent successfully completes all 180 steps and overtakes an average of 5.96 out of 6 NPC vehicles per episode in the considered traffic scenario over 30 driving episodes. No collisions are observed in test cases. In this section, we detail its state space, action space, and reward function.

State space: To optimize driving performance, we utilize front-view images obtained from the CARLA semantic segmentation camera. Each observation is a concatenation of 300-degree panoramas with 84×420 pixels, stacked by three frames per step, similar to the setup in [16].

Action space: Actuation has two elements: steering angle and thrust input, both clipped within the range of unit values to match the CARLA control command. Negative thrust means braking, while positive means throttling. Negative steering means turning left, while positive means turning right. The maximum steering angle is 70 degrees. Similarly to the modular driving pipeline, the DRL-based driving agent also predicts the steering angle variation ν and thrust variation γ of actuation a, within the range of ε. Specifically, at each time step t, the overall driving actuation a_t is calculated based on the current variation value ν_t and γ_t as well as its previous actuation value a_{t-1} as follows:

$$\begin{aligned} a_t^{\text{steer}} &= (1-\alpha)\cdot\nu_t + \alpha\cdot a_{t-1}^{\text{steer}}, \quad \nu \in [-\varepsilon, \varepsilon], \\ a_t^{\text{thrust}} &= (1-\eta)\cdot\gamma_t + \eta\cdot a_{t-1}^{\text{thrust}}, \quad \gamma \in [-\varepsilon, \varepsilon], \end{aligned} \tag{1}$$

where α and η determine the rate of previous actuation retain, while ε represents the mechanical limits of the actuation.

Reward function: We leverage the strengths of multiple existing works to create a reliable driving agent. Our reward design, inspired by [16], computes rewards using the dot product of the vehicle's speed and the waypoints vector. We also incorporate ideas from both [13] and [27] to train our policy model with the knowledge of a privileged agent. Specifically, we use the global and local path-planning modules described in Section III-B to generate a safe and reasonable reference path and incorporate it into our reward scheme. From a vague requirement (i.e., driving along the road without collision) to precise instruction (i.e., driving along a series of legal waypoints), this reward design allows us to achieve a highly reliable end-to-end AD agent that follows a safe and reasonable path. The shaped reward function aggregates multiple driving goals, including trajectory following, speed requirement, and

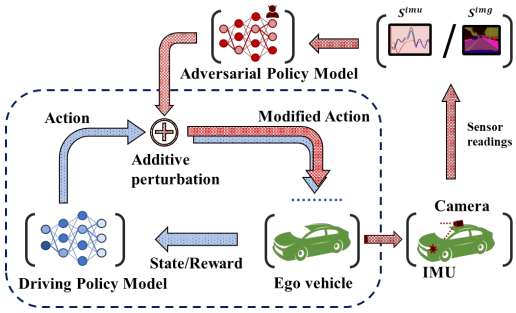


Fig. 2. Overview of the DRL-based action-space attack.

safety consideration. Without hard constraints, the agent may drive faster for higher rewards.

IV. DRL-based Attack Construction

The goal of the adversary is to create side collisions between the ego vehicle and other vehicles through a learning-based action-space attack strategy. The attack strategy involves interfering with driving actuation only during safety-critical moments while remaining undetected at all other times, allowing the adversary to execute an inconspicuous and successful attack with minimal effort. The attacker can be trained offline using the identical driving agent as the vehicle being targeted.

A. Attack Model

We refer to the driving agent as the victim policy π_v, and the attacker as the adversarial policy π_{adv}. To formulate the attack model, we make the following assumptions for the adversary. First, the victim driving agent is a fixed policy when deployed. Secondly, the adversary has the ability to manipulate the victim's steering action within the mechanical constraints while the agent's input/output readings are not necessarily required. Fig. 2 depicts an overview of the DRL-based action-space attack. By treating the entire driving system as a black box, the adversarial strategy is acquired through a learning approach. As the victim driving agent is fixed, its response to the environment and its state transitions form stationary dynamics. By designing input-output relationships properly, DRL can learn adversarial policies to achieve its goals. It is desirable for the adversary to access the same input as the victim agent and/or utilize the victim's output as part of the attack input as done in [3], [4]. However, such access may not be available and requires additional hacking and decoding. In our study, the attacker utilizes either an extra camera or an IMU to identify safety-critical moments. The former provides adequate information while its installation demands a wide field of view, which may attract attention from humans. The latter can be concealed within the vehicle, making them nearly unnoticeable, but provides a less informative inertia trace making the timing identification more challenging.

Since the attack involves injecting an additive signal rather than taking over the steering control, the victim driving agent is still predicting correct decisions based on its observation and consistently resists the attack to remedy the caused deviation. The result of the competition between the two policies leads to the final move of the vehicle. In other words, the attack budget which gives the attacker a boundary over the competition plays a key role in the attack's impact. Further, since the AD's thrust unit is not disturbed in this study, the AD agent can avoid a collision by slowing down or braking. This leads to another challenge for the attacker with only partial control over the actuator. Although our action-space attack model may not be as destructive as that with all control accesses [3], [4], its impact on AD solutions can provide insights.

B. Attack Feasibility

This section discusses the feasibility of action-space attacks in real-world AD systems. The digital control signals undergo a procedure of transmission from the vehicle's control unit to the actuator unit through the CAN bus, followed by conversion to analog signals using a digital-to-analog converter before finally operating the servo units. The above process exhibits two potential attack points: the CAN bus and the cable carrying the analog signals. The CAN bus adopts a message-based protocol for reliable and prioritized communications between devices. Message manipulation can be achieved through various entry points, such as remote hacking or false data injection [1]. However, this requires decoding and reverse engineering to corrupt specific CAN messages carrying the control commands. If hacking is unavailable, IEMI offers another alternative and non-intrusive approach, as cited in [6]–[8]. By intentionally generating electromagnetic interference near the target wire, IEMI can disturb the analog signals sent to the servo units, causing disruptions in the vehicle's motion. For instance, the attacker may target the electric power steering column, which assists the vehicle steering with the aid of an electronically controlled electric motor. After deploying required devices during premeditated car maintenance, the attacker can inject malicious commands into steering at safety-critical times with an IEMI device, thereby directly threatening driving safety.

C. Adversarial State Space and Action Space

Adversarial state space: To implement the camera-based attack, the camera is installed on the victim vehicle's roof in our setup to have a wide FOV. Front-view images for time-stacked semantic segmentation are used as state inputs for adversarial policy and are denoted by s^{img}. To implement the IMU-based attack, a triaxial IMU is mounted in the center of the victim vehicle, where the x-y-z axis readings record changes in the vehicle as it advances, rolls, and yaws. A trace of the IMU readings sampled at 20 sps over 3.2 seconds in the x-axis and the z-axis is the state input to the adversarial policy, which is denoted by s^{imu}. Note that the readings in the y-axis provide limited information about steering characteristics and are therefore not used. In cases where the IMU is installed elsewhere, the triaxial sensor readings and orientation alignment may be required.

Adversarial action space: The attack aims to interfere with the AD's steering by perturbing the steering angle variation ν as specified in Eq. (1). It is not a single-frame attack, but continuously injects perturbations when safety-critical

moments arise. Once an attack has been initiated, a collision is expected to occur within around one second, leaving no time for the human driver to make adjustments to correct. The magnitude of the injected perturbations is determined by the attack budget based on the mechanical limit of the steering system ($\varepsilon = 1$). The attack budget may vary based on the attacker's desired degree of imperceptibility or follow the safety check rules of the target vehicle. Specifically, at each time step t, the steering angle variation ν_t is perturbed by δ_t: $\nu_t' = \nu_t + \delta_t,\ \delta_t \in [-\varepsilon, \varepsilon]$, where the after-attacked variation ν_t' is weighted and accumulated with the last steering value a_{t-1}^{steer} by Eq. (1), forming an overall steering actuation a_t^{steer}.

D. Adversarial Reward Shaping

This section presents the design of the adversarial reward function that aims to create side collisions. The adversarial reward consists of the following three components.

■ **Collision reward** $C(\lambda)$ gives a positive value a if a side collision occurs, a negative value $-a$ if an undesired collision occurs, and a zero value if no collision occurs, where $\lambda = 1$, $\lambda = -1$ or $\lambda = 0$ represent the three cases, respectively.

■ **Collision potential reward** r_{e2n} characterizes the potential that the ego vehicle collides with the closest NPC vehicle. It is given by $r_{e2n} = \hat{v}_{e2n} \cdot \hat{v}_{ego}$, where $\hat{v}_{e2n}$ denotes the relative unit vector from the ego vehicle to the target vehicle, and $\hat{v}_{ego}$ denotes the speed unit vector of the ego vehicle. The maximum of the dot product of the two vectors corresponds to the case that the ego vehicle drives toward the target vehicle, thereby leading to the maximum collision potential.

■ **Attack maneuver penalty** p_m is the amount of perturbation injected per time step. Through this penalty term, the attacker learns to achieve the goal with minimal perturbations.

We conditionally combine r_{e2n} and p_m, depending on the relative spatial relationship between the ego vehicle and the closest NPC vehicle. We define the indicator function $I(\omega)$ to identify the safety-critical moments. Specifically, $I(\omega)$ is 1 if $|\omega| \leq \beta$ and 0 otherwise, where ω is the dot product between $\hat{v}_{e2n}$ and the speed unit vector of the NPC vehicle denoted by $\hat{v}_{npc}$. β is a pre-defined threshold that is set to be $\cos(\pi/6)$ in this paper, where this threshold plots a spatially appropriate time for an attack. $I(\omega)$ indicates the relative locations between the ego vehicle and the target NPC vehicle. Its being activated or not determines whether the current time is a critical moment. In summary, the overall adversarial reward R_{adv} combines the terms above, which is given by: $R_{adv} = C(\lambda) + I(\omega) r_{e2n} + (1 - I(\omega)) p_m$. By maximizing this, the attack will hijack the ego vehicle towards a target NPC vehicle during the critical moment characterized by $I(\omega)$.

Fig. 3 shows successive phases for the action-space attack. During the pre-attack phase, when the ego vehicle (green) is in the early stage of lane changing and overtaking, it is considered a non-critical moment for the attacker and no action should be taken. When the position between the ego vehicle and the target NPC vehicle (yellow) satisfies the conditions specified by $I(\omega)$, it is considered a critical moment. At this point, the action-space perturbation is continually injected into

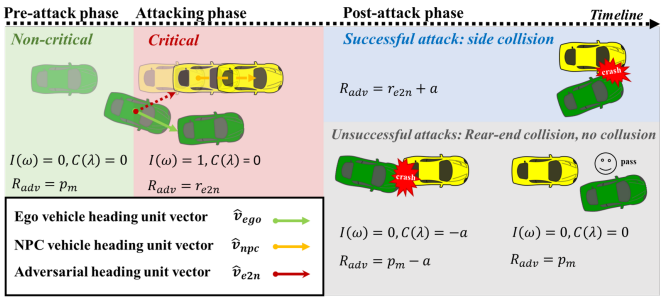


Fig. 3. A schematic plot illustrating the critical and non-critical moments for the action-space attack. The outcome of the attack is displayed in the post-attack phase, including both successful and unsuccessful attacks.

the steering control of the ego vehicle, resulting in a side collision. The goal of the attack during the attacking phase, as specified by in the collision reward $C(\lambda)$, is to cause a side collision with the target vehicle.

E. Training of IMU-based attack

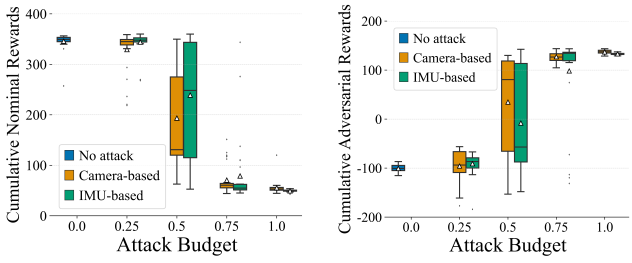
The camera-based adversarial policy is trained using the SAC algorithm with reward function R_{adv}, and training stops either when the maximum number of training steps is reached or when the average reward stabilizes during periodic evaluations. However, the same training process is ineffective for IMU-based policies due to the lack of correlation between location information and the IMU trace. To overcome this issue, we adopt a 'learning-from-teacher' approach and use the camera-based policy to teach the IMU-based policy. To achieve this, we modify the previous R_{adv} and introduce a new term: $R_{adv}^{\text{IMU}} = C(\lambda) + I(\omega) r_{e2n} + (1 - I(\omega)) p_m + p_{se}$. Here, p_{se} minimizes the discrepancy between the camera-based (teacher) and IMU-based (student) attacks at each time step through negative square error. As a result, the IMU-based attack agent is trained with additional information that captures the teacher policy. Once the training phase is complete, the camera is no longer needed since the IMU trace is sufficient for the attack.

V. Performance of Attacks

This section analyzes the performance of action-space attacks on AD agents. The analysis covers driving metrics (nominal driving reward and passed vehicles) and the attack effectiveness metrics (mean cumulative adversarial reward and attack success rate). The attack success rate is defined as the ratio of the number of collisions to the total number of evaluation episodes. The higher the adversarial reward, the stronger the attack effect, which results in lower driving performance. If the attacker elicits the desired side collision, the episode is considered successful from the attacker's perspective, and its cumulative adversarial reward is positive. If there is no collision, the vehicle hits a roadside barrier or the vehicle collides with an NPC vehicle in an unexpected posture (e.g., rear-end collision), the episode is considered unsuccessful which results in a negative cumulative adversarial reward.

A. Attack Effects under Various Attack Configurations

The nominal driving case can be considered as an attack case with zero attack budget (i.e., $\varepsilon = 0$), resulting in a negative



(a) (b)

Fig. 4. Box plot of average cumulative reward across 30 episodes under different attack budgets. (a) The distribution of the average of nominal driving rewards under different attack budgets. (b) Attackers' performance via adversarial reward under various attack budgets.

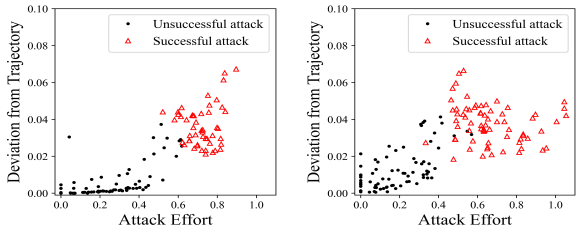
cumulative adversarial reward. Fig. 1(b) shows a side collision between the ego vehicle and an NPC vehicle under attack.

Camera-based vs. IMU-based attacks: We compare the performance of camera-based and IMU-based attacks with maximum attack budget ($\varepsilon = 1.00$), as shown in Fig. 4. The camera-based attack significantly decreases the driving agent's performance, reducing the cumulative nominal driving reward by approximately 84%. The mean adversarial cumulative reward over a hundred episodes achieved by the camera-based attack indicates that side collisions almost all occur when the ego vehicle encounters the first NPC vehicle. The IMU-based attack achieves a slightly lower mean adversarial reward than the camera-based attack, indicating that the IMU-based attack has successfully learned from the camera-based attack in the 'learning-from-teacher' process.

Impact of attack budget: Comparative results of the attack effectiveness under various attack budgets are given in Fig. 4. Both attacks show increasing effectiveness as the attack budget increases. However, the camera-based attack outperforms the IMU-based attack in terms of higher means and smaller variances in adversarial reward under the same attack budget. The IMU-based attack exhibits a lower transition curve in the drop of attack effectiveness versus attack budget, indicating that the driving agent under camera-based attacks is more likely to have a side collision. These results highlight the trade-off between attack effectiveness and stealthiness. A camera provides direct observation of the ego vehicle, making it easier for the attacker to hijack the vehicle toward the target (i.e., the NPC vehicle). In contrast, with only indirect observations via IMU, detecting the target vehicle's location and producing the desired attack effect becomes more difficult. Additionally, there is a sharp variation in both the nominal driving reward and the adversarial reward when the attack budget drops from $\varepsilon = 0.75$ to $\varepsilon = 0.25$. We hypothesize that the driving agent's action space may have a threshold of tolerance, and once the perturbation injected exceeds that threshold (e.g., $\varepsilon = 0.5$), the driving behavior is negatively affected.

B. End-to-end vs. Modular Driving Agents

This section evaluates the resilience of modular and DRL-based driving agents against camera-based attacks, with attack budgets ranging from 0 to 1.2 in steps of 0.1. Each budget



(a) Modular driving agent (b) End-to-end driving agent

Fig. 5. Evaluation of different driving agents under attacks.

is tested with 10 rounds of simulations. Fig. 5 shows the relationship between the steering deviation from the predetermined path and the attack effort. The attack effort is the total amount of perturbation injected during the attack attempt. The y-axis represents the root mean square error (RMSE) in the percentage of the steering deviation. The x-axis gives the mean attack effort averaged over the number of steps in each attack attempt. Each red triangle corresponds to a successful attack case that leads to a side collision; each black dot corresponds to cases that are not aligned with the attack objectives.

Successful attacks begin to dominate when the attack effort level surpasses a certain level, which is approximately 0.6 for the modular driving agent and about 0.5 for the end-to-end driving agent. Compared with the end-to-end driving agent, the modular driving agent can maintain more minor tracking errors in the trajectory following task when the attack effort is low. The modular driving agent outperforms the DRL-based end-to-end driving agent against camera-based attacks due to the inclusion of a PID controller, which instantly adjusts the actuator values to maintain the vehicle's planned path. On the other hand, the end-to-end driving agent may tend to prioritize speed over precision, leading to weakened resilience against attacks. These findings highlight the effectiveness of traditional feedback control, as implemented in the modular driving pipeline, in ensuring trajectory following and resilience to action-space attacks. Additionally, a successful attack case can be created in an average time of 0.87 sec (Min. 0.3 sec) for the end-to-end driving agent and 1.14 sec (Min. 0.9 sec) for the modular driving agent, which is 30.4% and 8.8% shorter than the best human driver reaction time (Min. 1.25 sec) in a complex real-world condition [28].

VI. Driving Agent Model Enhancement

A. Adversarial Training via Fine Tuning

Compared with modular driving pipelines, the end-to-end policy model has the advantage of being adaptable to new situations with further training. Adversarial training, which retrains the policy model in the presence of attacks, has been shown to form a defense strategy in previous research [3], [10]. We investigate this method for enhancing end-to-end driving agents against action-space attacks, where the camera-based attack is chosen due to its superior effectiveness. To increase 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

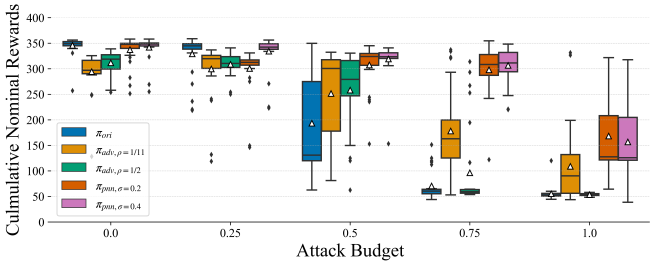


Fig. 6. Box plot of the nominal driving rewards for the original and enhanced end-to-end driving agents, showing the distribution of rewards for each agent.

of 0.1. Moreover, we control the ratio of selecting zero attack budget (i.e., no attack) to prevent overfitting to adversarial cases. In the following sections, we use $\pi_{\text{adv},\rho}$ to represent the adversarially-trained driving agent, where ρ denotes the ratio of nominal driving cases selected during the enhancement on a scale of one. In experiments, the values of ρ are set to $1/11$ and $1/2$, representing two variants: 1) each case has an equal probability of being selected during training; 2) the nominal case accounts for half of all the training cases. Besides, we use π_{ori} to denote the original end-to-end driving agent.

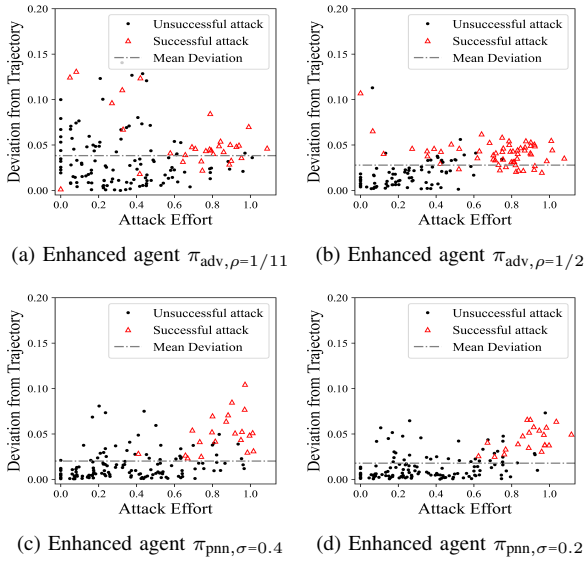
Fig. 6 shows the improved performance of the enhanced driving agents against attacks with different budgets. Compared to π_{ori} under camera-based attacks, the enhanced agents exhibit noticeable increases in the mean nominal driving reward. However, their driving performance suffers when the attack budget is small (i.e., $\varepsilon = 0.25$ and 0.00), indicating that it sacrifices nominal driving behavior for enhanced resilience.

Fig. 7 shows the deviation from driving trajectory versus attack effort for both adversarially-trained agents. In Fig. 7(a), the average trajectory tracking error for $\pi_{\text{adv},\rho=1/11}$ across all attack efforts is 0.038. Compared with Fig. 5(b), the attack effort level where the successful attack cases become dominating shifts right, suggesting that the enhanced driving agent can resist higher-budget attacks. However, large tracking errors are observed at zero and small attack efforts due to catastrophic forgetting problems. In Fig. 7(b), $\pi_{\text{adv},\rho=1/2}$ shows a more convergent relationship between the two variables, with fewer outliers and an average trajectory tracking error of 0.027. However, due to the reduced number of adversarial cases in training, successful attacks cluster more at high attack efforts, which weakens its resilience against high-budget attacks.

Adversarial training can counteract action-space attacks, but it may also degrade the driving agent's performance without attacks. Balancing the ratio of simulated attack cases during training to maintain run-time performance is challenging.

B. Model Enhancement with Progressive Neural Networks

To address the catastrophic forgetting problem, we investigate using Progressive Neural Networks (PNN) [29] during adversarial training. PNN transfers previously learned features of the original network (a *column*) through lateral connections to a new column without changing the original weights. The *switcher* determines the use of the original driving policy or the newly trained column, with σ as the threshold for the



(a) Enhanced agent $\pi_{\text{adv},\rho=1/11}$ (b) Enhanced agent $\pi_{\text{adv},\rho=1/2}$

(c) Enhanced agent $\pi_{\text{pnn},\sigma=0.4}$ (d) Enhanced agent $\pi_{\text{pnn},\sigma=0.2}$

Fig. 7. Evaluation of robustness of enhanced driving agents in terms of deviation from trajectory with the presence of camera-based attackers.

switcher to choose between the two. Specifically, the switcher chooses π_{ori} if $\varepsilon \leq \sigma$ and the adversarially trained column otherwise. The above design, which follows the Simplex architecture [30], makes an idealized assumption that the switcher is aware of the attack budget. Two settings for σ are evaluated: 0.2 and 0.4 in our experiments. In practice, the switcher can use different metrics such as confidence of exposure to attacks, the magnitude of a detected perturbation, or the type of attack being classified as a proxy of the attack budget. Fig. 6 presents the performance of PNN agents in terms of cumulative nominal driving reward in the presence of attacks. 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. When attack budgets are higher, the two PNN agents exhibit similar performance as they share the same model structure and weights. From the above results, the PNN method enhances the resistance of the driving policy to action-space attacks without damaging its nominal performance. Fig. 7(c) and Fig. 7(d) show the deviation from driving trajectory versus attack effort for the two PNN-enhanced driving agents. The $\pi_{\text{pnn},\sigma=0.4}$ achieves an average trajectory tracking error of 0.02 for all attack efforts. No attacks are successful when the attack effort is smaller than 0.4. The $\pi_{\text{pnn},\sigma=0.2}$ achieves an average trajectory tracking error of 0.017 for all attack efforts. No attacks are successful until the attack effort exceeds 0.6.

C. Fine Tuning vs. Progressive Neural Networks

Fig. 8 shows the trend of attack success rate with respect to the attack effort windows. The two adversarially-trained agents with fine-tuning show higher attack success rates even when the attack effort is small, while PNN-enhanced agents perform better with lower attack success rates in all attack scenarios. The results demonstrate the superior performance of the PNN method compared with fine-tuning in resisting action-

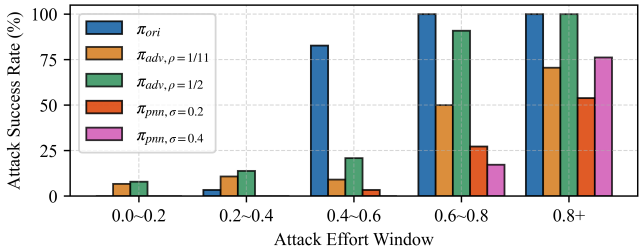


Fig. 8. A comparison of attack success rate between the nominal driving agent and four resulting enhanced driving agents across different attack effort windows. We window the data points in Fig. 7 along the attack effort axis with a size of 0.2, ranging from 0.0 to 0.8+.

space attacks while maintaining normal driving performance. Despite its effectiveness, the PNN method is constrained by its reliance on the switcher mechanism to detect and classify attacks, which can be viewed as a limitation.

VII. Conclusion

In this study, we present a DRL-based architecture capable of launching a black-box action-space attack on AD agents using camera or IMU sensor data, resulting in side collisions at safety-critical moments. Our evaluation has revealed that modular driving pipelines are more resilient to these attacks than end-to-end driving agents. we evaluate the effectiveness of adversarial training techniques, including fine-tuning and PNN, to enhance end-to-end agents. While the fine-tuning approach shows promise in improving the agent's performance, it also presents a potential issue of overfitting to adversarial cases that may impact nominal driving performance. The PNN method provided stronger resistance to attacks and overcame the catastrophic forgetting problem, but requires prior knowledge of the attacker's strategy, which may limit its practical application. Given that action-space attacks are rare but cannot be ignored, our results suggest that a simplex driving agent [31] that switches between the enhanced driving policy model and the nominal driving agent upon the capability of attack detection is desirable.

References

[1] X. Zhou, A. Schmedding, H. Ren, L. Yang, P. Schowitz, E. Smirni, and H. Alemzadeh, "Strategic safety-critical attacks against an advanced driver assistance system," in *2022 52nd Annual IEEE/IFIP International Conference on DSN*. IEEE, 2022, pp. 79–87.

[2] S. Jha, S. Banerjee, T. Tsai, S. K. Hari, M. B. Sullivan, Z. T. Kalbarczyk, S. W. Keckler, and R. K. Iyer, "Ml-based fault injection for autonomous vehicles: A case for bayesian fault injection," in *49th annual IEEE/IFIP international conference on DSN*. IEEE, 2019, pp. 112–124.

[3] X. Y. Lee, Y. Esfandiari, K. L. Tan, and S. Sarkar, "Query-based targeted action-space adversarial policies on deep reinforcement learning agents," in *ICCPS*. ACM, 2021, pp. 87–97.

[4] X. Y. Lee, S. Ghadai, K. L. Tan, C. Hegde, and S. Sarkar, "Spatiotemporally constrained action space attacks on deep reinforcement learning agents," in *AAAI*. AAAI Press, 2020, pp. 4577–4584.

[5] M. Moradi, B. J. Oakes, M. Saraoglu, A. Morozov, K. Janschek, and J. Denil, "Exploring fault parameter space using reinforcement learning-based fault injection," in *2020 50th Annual IEEE/IFIP International Conference on DSN-W*. IEEE, 2020, pp. 102–109.

[6] J. Selvaraj, "Intentional electromagnetic interference attack on sensors and actuators," Ph.D. dissertation, Iowa State University, 2018.

[7] G. Y. Dayanikli, R. R. Hatch, R. M. Gerdes, H. Wang, and R. Zane, "Electromagnetic sensor and actuator attacks on power converters for electric vehicles," in *SP Workshops*. IEEE, 2020, pp. 98–103.

[8] D. F. Kune, J. Backes, S. S. Clark, D. Kramer, M. Reynolds, K. Fu, Y. Kim, and W. Xu, "Ghost talk: Mitigating emi signal injection attacks against analog sensors," in *2013 IEEE S&P*. IEEE, 2013, pp. 145–159.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. M. López, and V. Koltun, "CARLA: an open urban driving simulator," in *CoRL*, vol. 78. PMLR, 2017, p. 16.

[10] K. L. Tan, Y. Esfandiari, X. Y. Lee, Aakanksha, and S. Sarkar, "Robustifying reinforcement learning agents via action space adversarial training," in *ACC*. IEEE, 2020, pp. 3959–3964.

[11] B. Paden, M. Cáp, S. Z. Yong, D. S. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, 2016.

[12] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016.

[13] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," *CoRR*, vol. abs/1912.12294, 2019.

[14] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. López, "Multimodal end-to-end autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, 2022.

[15] P. Cai, X. Mei, L. Tai, Y. Sun, and M. Liu, "High-speed autonomous drifting with deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, 2020.

[16] L. Fan, G. Wang, D. Huang, Z. Yu, L. Fei-Fei, Y. Zhu, and A. Anandkumar, "SECANT: self-expert cloning for zero-shot generalization of visual policies," in *ICML*. PMLR, 2021, pp. 3088–3099.

[17] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *ICCV*. IEEE, 2019, pp. 9328–9337.

[18] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *ITSC*. IEEE, 2019, pp. 2765–2771.

[19] A. Alizadeh, M. Moghadam, Y. Bicer, N. K. Ure, M. U. Yavas, and C. Kurtulus, "Automated lane change decision making using deep reinforcement learning in dynamic and uncertain highway environment," in *ITSC*. IEEE, 2019, pp. 1399–1404.

[20] M. Bouton, A. Nakhaei, D. Isele, K. Fujimura, and M. J. Kochenderfer, "Reinforcement learning with iterative reasoning for merging in dense traffic," in *ITSC*. IEEE, 2020.

[21] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *ICRA*. IEEE, 2018, pp. 2034–2039.

[22] Z. Cao and J. Yun, "Self-awareness safety of deep reinforcement learning in road traffic junction driving," *arXiv preprint arXiv:2201.08116*, 2022.

[23] J. Rong and N. Luan, "Safe reinforcement learning with policy-guided planning for autonomous driving," in *2020 IEEE ICMA*. IEEE, 2020, pp. 320–326.

[24] Y. Lin, Z. Hong, Y. Liao, M. Shih, M. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," in *ICLR (Workshop)*. OpenReview.net, 2017.

[25] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," in *ICLR*, 2020.

[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 1856–1865.

[27] E. Yurtsever, L. Capito, K. A. Redmill, and Ü. Özgüner, "Integrating deep reinforcement learning with model-based path planners for automated driving," in *IV*. IEEE, 2020, pp. 1311–1316.

[28] M. Zhuk, V. Kovalyshyn, Y. Royko, and K. Barvinska, "Research on drivers' reaction time in different conditions," *EasternEuropean Journal of Enterprise Technologies*, vol. 2, pp. 24–31, 04 2017.

[29] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *CoRR*, vol. abs/1606.04671, 2016.

[30] L. Sha, "Using simplicity to control complexity," *IEEE Software*, vol. 18, no. 4, pp. 20–28, 2001.

[31] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo, "S3A: secure system simplex architecture for enhanced security and robustness of cyber-physical systems," in *HiCoNS*. ACM, 2013, pp. 65–74.