Toward Secure and Safe Autonomous Driving: an Adversary's Perspective

by

Yulong Cao

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2023

Doctoral Committee:

Professor Z. Morley Mao, Chair
Associate Professor Kevin Fu
Professor Mingyan Liu
Professor Atul Prakash
Assistant Professor Chaowei Xiao, Arizona State University

Yulong Cao
yulongc@umich.edu
ORCID iD: 0000-0003-3007-2550

To my parents, my grandparents, and my love

ACKNOWLEDGEMENTS

It is the seventh year since I first came to Ann Arbor for my undergrad. This small, peaceful town has been my second home town now. Looking back from the end of this road, there are so many people I would like to thank, who are indispensable for this wonderful journey full of passion, love, and growth.

Foremost, I would like to gratefully thank my advisor, Professor Zhuoqing Morley Mao for her unconditional trust in me. Her constant support was one of the most important reasons for my being able to bring this dissertation to its completion. She has been supporting me in exploring new directions, which are often frustrating due to the unknown challenges. With her guidance over the years, I have the confidence to say that I have grown to be a researcher that can conduct independent research.

Besides my advisor, I would like to thank my dissertation committee, Professor Mingyan Liu, Professor Atul Prakash, Professor Kevin Fu, and Professor Chaowei Xiao for their insightful suggestions, comments, and support.

I am grateful to my mentors, Professor Chaowei Xiao, Professor Alfred Chen, who have been patient enough for teaching me, from research to life. I also want to thank my internship mentors, Yunhan Jia, Yueqiang Cheng, Professor Danfei Xu, Xinshuo Weng, and Professor Marco Pavone, who have guided and helped me through one and another amazing journeys. I am also grateful to my collaborators I have fortunately worked with, Jiachen Sun, Benjamin Cyr, Yimeng Zhou, Won Park, Professor Sara Rampazzi, Ningfei Wang, Dawei Yang, Jin Fang, Ruigang Yang, Professor Bo Li, R Spencer Hallyburton, Professor Miroslav Pajic, David Ke Hong, Professor Scott

Mahlke, Professor Anima Anandkumar, S Hrushikesh Bhupathiraju, Pirouz Naghavi, and Professor Takeshi Sugawara. Without their help, I won't be able to accomplish all the projects alone.

This journey would have never been the same without my friends Shengtuo Hu, Jiachen Sun, Zhiyi Chen, who have been constantly inspiring me and pushing me through project after project. Also, my lab mates Xiao Zhu, Shichang Xu, Yikai Lin, Roy Shao, Xumiao Zhang, Qingzhao Zhang have been the indispensable part of this journey as well.

Finally, I want to thank my father, Xinhua Cao, mother, Yi Li, grandfather, Weixi Li, and my girlfriend Ji Qiu for being the constant mental support in my life. Their unconditional love and support helped me bring this adventure to an end. This dissertation is dedicated to them.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

Appendix

ABSTRACT

Autonomous vehicles, also known as self-driving cars, are being developed at a rapid pace due to advances in machine learning. However, the real-world is complex and dynamic, with many different factors that can affect the performance of an autonomous driving (AD) system. Therefore, it is essential to thoroughly test and evaluate AD systems to ensure their safety and reliability in the open-world driving environment. Additionally, due to the high impact of AD systems on road safety, it is important to build robust AD systems that are resistant to adversaries.

However, fully testing and exploiting AD systems can be challenging due to their complexity, as they consist of a combination of sensors, systems, and machine learning models. To address these challenges, my dissertation research focuses on building secure and safe AD systems through systematic analysis of attackers' capabilities. This involves testing AD systems as a whole, using realistic attacks, and discovering new security problems through proactive analysis.

To achieve this goal, my dissertation starts by formulating realistic attacker capabilities against perception systems. Based on this, new attacks on perception systems are discovered that have different impacts (e.g., spoofing ghost objects or removing detected objects). We proposed two frameworks, **adv-LiDAR** and **LiDAR-adv**, that differentiate the LiDAR-based perception systems and generate effective adversarial examples automatically. As the result, we also demonstrated the proposed attacks can lead to vehicular-level impacts such as emergency braking or collisions.

Next, causality analysis is conducted to expose the fundamental limitations of the system (e.g., large receptive fields introducing new attack vectors). This provides insights and guidelines for designing more robust systems in the future. By evaluating

the adversarial robustness of different semantic segmentation models, we unveil the fundamental limitations of using large receptive fields. Specifically, we validate our findings using the **remote adversarial patch** (RAP) attack, which can mislead the prediction result of the target object without directly accessing and manipulating (adding) adversarial perturbation to it.

Finally, solutions are developed to improve the modular and integrated robustness of the systems. By leveraging adversarial examples, the training dataset for machine learning models can be augmented to naturally improve modular robustness. We demonstrated that, with robust trained trajectory prediction models, AD systems can avoid collisions under adversarial attacks. On the other hand, using insights from the causality analysis and formulated attacker capabilities, AD systems with enhanced integrated robustness can be designed.

CHAPTER I

Introduction

Autonomous vehicles, or self-driving cars, are under rapid development, driven by recent progress in machine learning. Such advancements have shown competitive performance in sensing, forecasting, and decision-making and now, some vehicles are already found on public roads [14, 12, 8]. In AD systems, one fundamental pillar is *perception*, which leverages sensors like cameras and LiDARs (Light Detection and Ranging) to understand the surrounding driving environment. Since such function is directly related to safety-critical driving decisions such as collision avoidance, multiple prior research efforts have been made to study the security of camera-based perception in AD settings. For example, prior work has reported sensor-level attacks such as camera blinding [111], physical-world camera attacks such as adding stickers to traffic signs [61, 60], and trojan attacks on the neural networks for AD camera input [95].

However, while there are significant prior works for attacks and mitigation on perception models, little has been done from a realistic adversary's perspective nor on other tasks such as trajectory prediction. For example, lots of works demonstrate adversarial examples that add small perturbations to the image fooling the perception model while such threat model is not realistic without compromising the autonomous driving system. Some other works proposed adversarial patches which are more realizable localized perturbation and used them to fool the traffic signs de-

tection/classification. However, traffic sign information is usually embedded in the High-definition map (HDMap) on the modern autonomous driving system [3]. To understand the real threats on AD systems and secure them against attackers, the challenges are multi-folded. We detail three of them here

- **Compound systems.** Consisting of sensors, systems, and machine learning models, several subsystems (e.g., perception system) in the AD system are compound systems. While each component might be compromised, the final output of the subsystem is unclear. For example, in the perception system, machine learning models usually have a certain level of tolerance to sensor noises. To efficiently attack the perception system requires the attacker to analyze all the components, which has not been studied before.

- **Unclear attacker's capabilities in the physical world.** To understand the vulnerability status of the AD systems, it is crucial to understand the attacker's capability. Autonomous vehicle as a compound system with various components also means a larger attack surface for the attacker. Finding realistic threat models and analyzing the attacker's capability is the key to understanding the vulnerability status.

- **Security being an afterthought.** The attack and defense arms race also appears in the autonomous driving security and safety. However, as many components of autonomous driving systems require real-time performance, patching the system could be more difficult. Therefore, a secure and safe autonomous driving system is required to be robust to future attacks.

The overall goal of my research is to advance the safety and security of the autonomous driving system, by identifying key vulnerabilities, analyzing potential attacker's capability, and securing the system against such capability. More specifically, I 1) analyzed and summarized the attacker's capability on compromising camera-

based perception and LiDAR-based perception; 2) demonstrated the potential attack impacts given such capabilities; 3) and propose defenses that rule out the attacker's capability to defend against both existing and future attacks.

In summary, my dissertation demonstrates that: **Systematic analysis of attacker's capability on sensors, systems, and smart algorithms under autonomous driving settings can (1) Proactively discover new types of security vulnerabilities; (2) Systematically analyze the vulnerability status and the fundamental causes; (3) Provide a solid ground for building effective and efficient defenses that rule out existing and future attacks.**

1.1 Overview

My dissertation validates the thesis in four major parts:

1. **Vulnerability status of LiDAR-based perception.** We investigate two types of attacks: LiDAR spoofing attack and object reshaping attack. For the LiDAR spoofing attack, we measured the capability and limitations of LiDAR spoofing attack. Based on the spoofing capability, we proposed different attacks that achieve different attack goals. Requiring a weak spoofing capability, we proposed using adversarial machine learning to enhance the attack impacts at the perception system level. We demonstrated that with only less than 60 points, the attacker can spoof a near-front obstacle to the autonomous vehicle and causes the victim vehicle to freeze or emergency brake. For the object reshaping attack, we demonstrated that by manipulating the shape of a traffic cone, we can make it invisible to the autonomous driving system.

2. **Vulnerability causality analysis on camera-based perception.** We perform the first study on measuring the vulnerability status of remote adversarial patches on semantic segmentation models. We proposed the remote adversarial

patch attack, which fools camera-based perception with adversarial patches not overlapping with the target victim. The remote adversarial patch is considered a more realistic attack compared to previously proposed adversarial patch attacks since it does not require the attacker to physically tamper with the victim object. We conducted causality analysis and found the vulnerability is introduced by a large receptive field. We studied the vulnerability status among different model designs for increasing receptive field sizes. And finally, we conducted an object removing attack in both digital and physical experiments to demonstrate the attack impacts.

3. **Secure and safe autonomous driving with modular robustness.** Trajectory prediction is a crucial aspect of autonomous vehicle navigation, as it enables the vehicle to anticipate the movements of other vehicles, pedestrians, and objects in its environment. However, the accuracy and reliability of trajectory prediction can be compromised by adversarial attacks, which seek to mislead the prediction model and cause the vehicle to make unsafe or incorrect decisions. In this part, we investigate the adversarial robustness of trajectory prediction for autonomous vehicles. We first formulate a realistic threat model where the attacker is able to maneuver a vehicle on the road along a specific path to fool the autonomous vehicles. We then devise a method, AdvDO, to efficiently and effectively generate adversarial trajectories. With that, we measured vulnerability status of trajectory prediction models. To improve the modular robustness, we conduct adversarial training on trajectory prediction models. We identified three key challenges of conducting adversarial training on trajectory prediction models and proposed corresponding solutions to it. Our study demonstrated that trajectory prediction models are vulnerable to adversarial attacks and improving the robustness of them without large clean performance degradation is feasible.

4. **Secure and safe autonomous driving with integrated robustness.** While various attacks against the autonomous driving perception systems have been proposed in recent years, most of them share similar threat models: LiDAR sensor spoofing, and object reshaping. Instead of building reactive defenses and getting trapped in the arms race of attacks and defenses, securing the perception system from the attacker's capability naturally defends existing and future attacks. We proposed to build defenses to rule out the attacker's capability on LiDAR-based perception and camera-based perception respectively. More specifically, we propose to identify the characteristics of each attack capability and defend it. For example, we found that LiDAR sensor spoofing attacks are usually creating less stable spoofed point cloud compared to the benign point cloud. Therefore, we propose to detect the inconsistency in a sequence of point cloud data to detect the LiDAR spoofing attacks. Our study will provide a solid ground for defending the known attack capabilities and serve as a guideline for defending against future attacks if new attack capabilities appear.

1.2 Disseration Organization

This dissertation is structured as follow. Chapter II and Chapter III describe two attacks: sensor spoofing attack, and object reshaping attack, on the LiDAR-based perception system, with different attacker's capabilities and attack goals. Chapter IV presents an analysis of how large receptive field in modern deep neural networks enables a new attack vector, the remote adversarial patch attack. In Chapter V, we present a study towards a robust trajectory prediction system, which is another fundamental component in an AD system. In Chapter VI, we describe a preliminary study on integrated robustness of the AD system, discussing why we should care about integrated robustness and how to build integrated robust AD systems. At last, in Chapter VII, we conclude this dissertation and discuss potential future directions.

CHAPTER II

Vulnerability Status of LiDAR-based Perception against the Sensor Spoofing Attack

2.1 Introduction

Autonomous vehicles, or self-driving cars, are under rapid development, with some vehicles already found on public roads [14, 12, 8] In AV systems, one fundamental pillar is *perception*, which leverages sensors like cameras and LiDARs (Light Detection and Ranging) to understand the surrounding driving environment. Since such function is directly related to safety-critical driving decisions such as collision avoidance, multiple prior research efforts have been made to study the security of camera-based perception in AV settings. For example, prior work has reported sensor-level attacks such as camera blinding [111], physical-world camera attacks such as adding stickers to traffic signs [61, 60], and trojan attacks on the neural networks for AV camera input [95].

Despite the research efforts in camera-based perception, there is no thorough exploration into the security of LiDAR-based perception in AV settings. LiDARs, which measure distances to surrounding obstacles using infrared lasers, can provide 360-degree viewing angles and generate 3-dimensional representations of the road environment instead of just 2-dimensional images for cameras. Thus, they are gener-

ally considered as more important sensors than cameras for AV driving safety [10, 6] and are adopted by nearly all AV makers today [7, 5, 11, 3]. A few recent works demonstrated the feasibility of injecting spoofed points into the sensor input from the LiDAR [111, 132]. Since such input also needs to be processed by an object detection step in the AV perception pipeline, it is largely unclear whether such spoofing can directly lead to semantically-impactful security consequences, e.g., adding spoofed road obstacles, in the LiDAR-based perception in AV systems.

In this work, we perform the first study to explore the security of LiDAR-based perception in AV settings. To perform the analysis, we target the LiDAR-based perception implementation in Baidu Apollo, an open-source AV system that has over 100 partners and has reached a mass production agreement with multiple partners such as Volvo and Ford [9, 8]. We consider a LiDAR spoofing attack, i.e., injecting spoofed LiDAR data points by shooting lasers, as our threat model since it has demonstrated feasibility in previous work [111, 132]. With this threat model, we set the attack goal as adding spoofed obstacles in close distances to the front of a victim AV (or *front-near* obstacles) in order to alter its driving decisions.

In our study, we first reproduce the LiDAR spoofing attack from the work done by *Shin et al.* [132] and try to exploit Baidu Apollo's LiDAR-based perception pipeline, which leverages machine learning for object detection as with the majority of the state-of-the-art LiDAR-based AV perception techniques [4]. We enumerate different spoofing patterns from the previous work, e.g., a spoofed wall, and different spoofing angles and shapes, but none of them succeed in generating a spoofed road obstacle after the machine learning step. We find that a potential reason is that the current spoofing technique can only cover a very narrow spoofing angle, i.e., 8° horizontally in our experiments, which is not enough to generate a point cloud of a road obstacle near the front of a vehicle. Thus, blindly applying existing spoofing techniques cannot easily succeed.

To achieve the attack goal with existing spoofing techniques, we explore the possibility of strategically controlling the spoofed points to fool the machine learning model in the object detection step. While it is known that machine learning output can be maliciously altered by carefully-crafted perturbations to the input [110, 61, 29, 182, 27], no prior work studied LiDAR-based object detection models for AV systems. To approach this problem, we formulate the attack task as an optimization problem, which has been shown to be effective in previous machine learning security studies [28, 41, 165, 157, 36, 158]. Specific to our study, two functions need to be newly formulated: (1) an input perturbation function that models LiDAR spoofing capability in changing machine learning model input, and (2) an objective function that can reflect the attack goal. For the former, since previous work did not perform detailed measurements for the purpose of such modeling, we experimentally explore the capability of controlling the spoofed data points, e.g., the number of points and their positions. Next, we design a set of global spatial transformation functions to model these observed attack capabilities at the model input level. In this step, both the quantified attack capabilities and the modeling methodology are useful for future security studies of LiDAR-related machine learning models.

For the attack goal of adding front-near obstacles, designing a objective function is also non-trivial since the machine learning model output is post-processed in the perception module of Baidu Apollo before it is converted to a list of perceived obstacles. To address this, we study the post-processing logic, extract key strategies of transforming model output into perceived obstacles, and formulate it into the objective function.

With the optimization problem mathematically formulated, we start by directly solving it using optimization algorithms like previous studies [28]. However, we find that the average success rate of adding front-near obstacles is only 30%. We find that this is actually caused by the nature of the problem, which makes it easy for any

optimization algorithm to get trapped in local extrema. To solve this problem, we design an algorithm that combines global sampling and optimization, which is able to successfully increase the average success rates to around 75%.

As a case study for understanding the impact of the discovered attack input at the AV driving decision level, we construct two attack scenarios: (1) *emergency brake attack*, which may force a moving AV to suddenly brake and thus injure the passengers or cause rear-end collisions, and (2) *AV freezing attack*, which may cause an AV waiting for the red light to be permanently "frozen" in the intersection and block traffic. Using real-world AV driving data traces released by the Baidu Apollo team, both attacks successfully trigger the attacker-desired driving decisions in Apollo's simulator.

Based on the insights from our security analysis, we propose defense solutions not only at AV system level, e.g., filtering out LiDAR data points from ground reflection, but also at sensor and machine learning model levels.

In summary, this work makes the following contributions:

- We perform the first security study of LiDAR-based perception for AV systems. We find that blindly applying existing LiDAR spoofing techniques cannot easily succeed in generating semantically-impactful security consequences after the machine learning-based object detection step. To achieve the attack goal with existing spoofing techniques, we then explore the possibility of strategically controlling the spoofed points to fool the machine learning model, and formulate the attack as an optimization problem.
- To perform analysis for the machine learning model used in LiDAR-based AV perception, we make two methodology-level contributions. First, we conduct experiments to analyze the LiDAR spoofing attack capability and design a global spatial transformation based method to model such capability in mathematical forms. Second, we identify inherent limitations of directly solving our problem

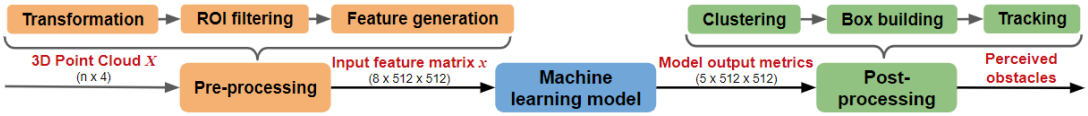


Figure 2.1: Overview of the data processing pipeline for LiDAR-based perception in Baidu Apollo.

using optimization, and design an algorithm that combines optimization and global sampling. This is able to increase the attack success rates to around 75%.

- As a case study to understand the impact of the attacks at the AV driving decision level, we construct two potential attack scenarios: emergency brake attack, which may hurt the passengers or cause a rear-end collision, and AV freezing attack, which may block traffic. Using a simulation based evaluation on real-world AV driving data, both attacks successfully trigger the attacker-desired driving decisions. Based on the insights, we discuss defense directions at AV system, sensor, and machine learning model levels.

2.2 Background

2.2.1 LiDAR-based Perception in AV Systems

AVs rely on various sensors to perform real-time positioning (also called localization) and environment perception (or simply perception). LiDAR, camera, radar, and GPS/IMU are major sensors used by various autonomous driving systems. The data collected from those sensors are transformed and processed before it becomes useful information for AV systems. Fig. 2.1 shows the data processing pipeline of LiDAR sensor data in the perception module of Baidu Apollo [3]. As shown, it involves three main steps as follows:

Step 1: Pre processing. The raw LiDAR sensor input is called *3D point cloud* and we denote it as X. The dimension of X is $n \times 4$, where n denotes the number

of data points and each data point is a 4-dimension vector with the 3D coordinates, $\mathbf{w_x}$, $\mathbf{w_y}$, and $\mathbf{w_z}$, and the intensity of the point. In the pre-processing step, X is first transformed into an absolute coordinate system. Next, the *Region of Interest (ROI)* filter removes unrelated portions of the 3D point cloud data, e.g., those that are outside of the road, based on HDMap information. Next, a *feature generation* process generates a feature matrix x ($8 \times 512 \times 512$), which is the input to the subsequent machine learning model. In this process, the ROI-filtered 3D point cloud within the range (60 meters by default) is mapped to 512×512 cells according to the $\mathbf{w_x}$ and $\mathbf{w_y}$ coordinates. In each cell, the assigned points are used to generate 8 features as listed in Table 2.1.

Step 2: DNN-based object detection. A Deep Neural Network (DNN) then takes the feature matrix x as input and produces a set of output metrics for each cell, e.g., the probability of the cell being a part of an obstacle. These output metrics are listed in Table 2.2.

Step 3: Post processing. The clustering process only considers cells with *objectness* values (one of the output metrics listed in Table 2.2) greater than a given threshold (0.5 by default). Then, the process constructs candidate object clusters by building a connected graph using the cells' output metrics. Candidate object clusters are then filtered by selecting clusters with average *positiveness* values (another output metric) greater than a given threshold (0.1 by default). The *box builder* then reconstructs the bounding box including height, width, length of an obstacle candidate from the 3D point cloud assigned to it. Finally, the *tracker* integrates consecutive frames of processed results to generate tracked obstacles, augmented with additional information such as speed, acceleration, and turning rates, as the output of the LiDAR-based perception.

With the information of perceived obstacles such as their positions, shapes, and obstacle types, the Apollo system then uses such information to make driving deci-

sions. The perception output is further processed by the *prediction* module which predicts the future trajectories of perceived obstacles, and then the *planning* module which plans the future driving routes and makes decisions such as stopping, lane changing, yielding, etc.

Feature	Description
Max height	Maximum height of points in the cell.
Max intensity	Intensity of the brightest point in the cell.
Mean height	Mean height of points in the cell.
Mean intensity	Mean intensity of points in the cell.
Count	Number of points in the cell.
Direction	Angle of the cell's center with respect to the origin.
Distance	Distance between the cell's center and the origin.
Non-empty	Binary value indicating whether the cell is empty or occupied.

Table 2.1: DNN model input features.

Metrics	Description
Center offset	Offset to the predicted center of the cluster the cell belongs to.
Objectness	The probability of a cell belonging to an obstacle.
Positiveness	The confidence score of the detection.
Object height	The predicted object height.
Class probability	The probability of the cell being a part of a vehicle, pedestrian, etc.

Table 2.2: DNN model output metrics.

2.2.2 LiDAR Sensor and Spoofing Attacks

To understand the principles underlying our security analysis methodology, it is necessary to understand how the LiDAR sensor generates a point cloud and how it is possible to alter it in a controlled way using spoofing attacks.

LiDAR sensor. A LiDAR sensor functions by firing laser pulses and capturing their reflections using photodiodes. Because the speed of light is constant, the time it takes for the echo pulses to reach the receiving photodiode provides an accurate measurement of the distance between a LiDAR and a potential obstacle. By firing the laser pulses at many vertical and horizontal angles, a LiDAR generates a point cloud used by the AV systems to detect objects.

LiDAR spoofing attack. Sensor spoofing attacks use the same physical channels as the targeted sensor to manipulate the sensor readings. This strategy makes it very difficult for the sensor system to recognize such attack, since the attack doesn't require any physical contact or tampering with the sensor, and it doesn't interfere with the processing and transmission of the digital sensor measurement. These types of attack could trick the victim sensor to provide seemingly legitimate but actually erroneous data.

LiDAR has been shown to be vulnerable to laser spoofing attacks in prior work. Petit et al. demonstrated that a LiDAR spoofing attack can be performed by replaying the LiDAR laser pulses from a different position to create fake points further than the location of the spoofer [111]. Shin et al. showed that it is possible to generate a fake point cloud at different distances, even closer than the spoofer location [132]. In this paper, we build upon these prior work to study the effect of this attack vector on the security of AV perception.

2.2.3 Adversarial Machine Learning

Neural networks. A neural network is a function consisting of connected units called (artificial) neurons that work together to represent a differentiable function that

outputs a distribution. A given neural network (e.g., classification) can be defined by its model architecture and parameters ϕ. An optimizer such as Adam [80] is used to update the parameters ϕ with respect to the objective function $\mathcal{L}$.

Adversarial examples. Given a machine learning model M, input x and its corresponding label y, an adversarial attacker aims to generate adversarial examples x' so that $M(x') \neq y$ (untargeted attack) or $M(x') = y'$, where y' is a target label (targeted attack). *Carlini and Wagner* [28] proposed to generate an adversarial perturbation for a targeted attack by optimizing an objective function as follows:

$$\min ||x - x'||_p \quad \text{s.t.} \qquad M(x') = y' \quad \text{and} \quad x' \in X,$$

where $M(x') = y'$ is the target adversarial goal and $x' \in X$ denote that the adversarial examples should be in a valid set. Further, optimization-based algorithms have been leveraged to generate adversarial examples on various kinds of machine learning tasks successfully, such as segmentation [165, 41], human pose estimation [41], object detection [165], Visual Question Answer system [171], image caption translation [36], etc. In this paper, we also leverage an optimization-based method to generate adversarial examples to fool LiDAR-based AV perception.

2.3 Attack Goal and Threat Model

Attack goal. To cause semantically-impactful security consequence in AV settings, we set the attack goal as fooling the LiDAR-based perception into perceiving fake obstacles in front of a victim AV in order to maliciously alter its driving decisions. More specifically, in this work, we target *front-near* fake obstacles, i.e., those that are in close distances to the front of a victim AV, since they have the highest potential to trigger immediate erroneous AV driving decisions. In this work, we define front-near obstacles as those that are around 5 meters to the front of a victim AV.

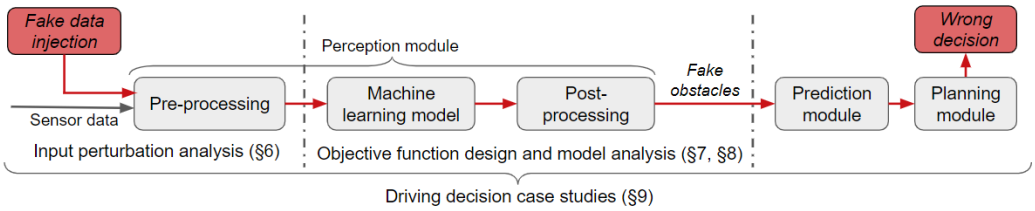


Figure 2.2: Overview of the Adv-LiDAR methodology.

Threat model. To achieve the attack goal above, we consider LiDAR spoofing attacks as our threat model, which is a demonstrated practical attack vector for LiDAR sensors [111, 132] as described in §2.2.2. In AV settings, there are several possible scenarios to perform such attack. First, the attacker can place an attacking device at the roadside to shoot malicious laser pulses to AVs passing by. Second, the attacker can drive an attack vehicle in close proximity to the victim AV, e.g., in the same lane or adjacent lanes. To perform the attack, the attack vehicle is equipped with an attacking device that shoots laser pulses to the victim AV's LiDAR. To perform laser aiming in these scenarios, the attacker can use techniques such as camera-based object detection and tracking. In AV settings, these attacks are stealthy since the laser pulses are invisible and laser shooting devices are relatively small in size.

As a first security analysis, we assume that the attacker has white-box access to the machine learning model and the perception system. We consider this threat model reasonable since the attacker could obtain white-box access by additional engineering efforts to reverse engineering the software.

2.4 Limitation of Blind sensor spoofing

To understand the security of LiDAR-based perception under LiDAR spoofing attacks, we first reproduce the state-of-the-art LiDAR spoofing attack by Shin et al. [132], and explore the effectiveness of directly applying it to attack the LiDAR-based perception pipeline in Baidu Apollo [3], an open-source AV system that has

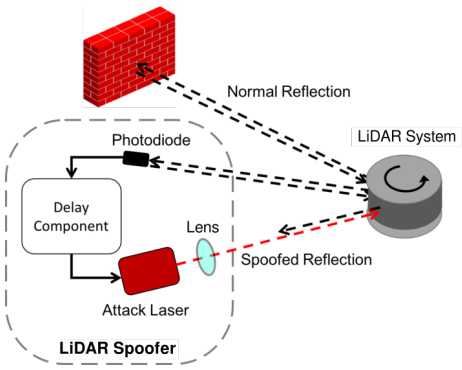


Figure 2.3: Illustration of LiDAR spoofing attack. The photodiode receives the laser pulses from the LiDAR and activate the delay component that triggers the attacker laser to simulate real echo pulses.

over 100 partners and has reached mass production agreement with multiple partners such as Volvo, Ford, and King Long [9, 8].

Spoofing attack description. The attack by Shin et al. [132] consists of three components: a photodiode, a delay component, and an infrared laser, which are shown in Fig. 2.3. The photodiode is used to synchronize with the victim LiDAR. The photodiode triggers the delay component whenever it captures laser pulses fired from the victim LiDAR. Then the delay component triggers the attack laser after a certain amount of time to attack the following firing cycles of the victim LiDAR. Since the firing sequence of laser pulses is consistent, an adversary can choose which fake points will appear in the point cloud by crafting a pulse waveform to trigger the attack laser.

Experimental setup. We perform spoofing attack experiments on a VLP-16 PUCK LiDAR System from Velodyne [74]. The VLP-16 uses a vertical array of 16 separate laser diodes to fire laser pulses at different angles. It has a 30 degree vertical angle range from -15 ° to +15 °, with 2 ° of angular resolution. The VLP-16 rotates horizontally around a center axis to send pulses in a 360 ° horizontal range, with a varying azimuth resolution between 0.1 ° and 0.4 °. The laser firing sequence follows the pattern shown in Figure 2.4. The VLP-16 fires 16 laser pulses in a cycle every

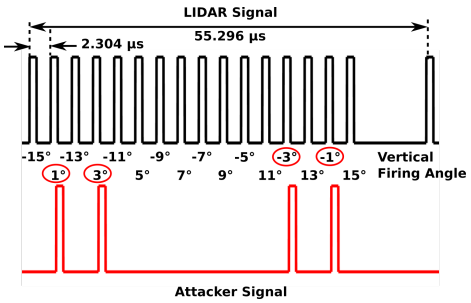


Figure 2.4: The consistent firing sequence of the LiDAR allows an attacker to choose the angles and distances from which spoofed points appear. For example, applying the attacker signal, fake dots will appear at 1°, 3°, -3°, and -1° angles (0° is the center of the LiDAR)

55.296 μs, with a period of 2.304 μs. The receiving time window is about 667 ns. We chose this sensor because it is compatible with Baidu Apollo and uses the same design principle as the more advanced HDL-64E LiDARs used in many AVs. The similar design indicates that the same laser attacks that affect the VLP-16 can be extended to high-resolution LiDARs like the HDL-64E.

We use the OSRAM SFH 213 FA as our photodiode, with a comparator circuit similar to the one used by Shin et al. We use a Tektronix AFG3251 function generator as the delay component with the photodiode circuit as an external trigger. In turn, the function generator provides the trigger to the laser driver module PCO-7114 that drives the attack laser diode OSRAM SPL PL90. With the PCO-7114 laser driver, we were able to fire the laser pulses at the same pulse rate of the VLP-16, 2.304 μs, compared to 100 μs of the previous work. An optical lens with a diameter of 30mm and a focal length of 100 mm was used to focus the beam, making it more effective for ranges farther than 5 meters. We generate the custom pulse waveform using the Tektronix software ArbExpress [2] to create different shapes and the Velodyne software VeloView [13] to analyze and extract the point clouds.

Experiment results. The prior work of Shin et al. is able to spoof a maximum of 10 fake dots in a single horizontal line. With our setup improvements (a faster

firing rate and a lens to focus the beam), fake points can be generated at all of the 16 vertical viewing angles and an 8 ° horizontal angle at greater than 10 meters away. In total, around 100 dots can be spoofed by covering these horizontal and vertical angles. These spoofed dots can also be shaped by modifying the custom pulse waveform used to fire the attack laser. Noticed that even though around 100 dots can be spoofed, they are not all spoofed stably. The attacker is able to spoof points at different angles because the spoofed laser pulses hit a certain area on the victim LiDAR due to the optical lens focusing. The closer to the center of the area, the stronger and stabler laser pulses are received by the victim LiDAR. We find that among 60 points at the center 8-10 vertical lines can be stably spoofed with high intensity.

2.4.1 Blind LiDAR Spoofing Experiments

After reproducing the LiDAR spoofing attack, we then explore whether blindly applying such attack can directly generate spoofed obstacles in the LiDAR-based perception in Baidu Apollo. Since our LiDAR spoofing experiments are performed in indoor environments, we synthesize the on-road attack effect by adding spoofed LiDAR points to the original 3D point cloud collected by Baidu Apollo team on local roads at Sunnyvale, CA. The synthesizing process is illustrated in Fig. 2.5. After this process, we run Apollo's perception module with the attacker-perturbed 3D point cloud as input to obtain the object detection output. In this analysis, we explore three blind attack experiments as follows:

Experiment 1: Directly apply original spoofing attack traces. In this experiment, we directly replay spoofing attack traces to attack LiDAR-based perception in Apollo. More specifically, we experiment with attack traces obtained from two sources: (1) the original spoofing attack traces from Shin et al. [132], and (2) the attack traces generated from the spoofing attack reproduced by us, which can inject more dots after our setup improvements. However, we are not able to observe a

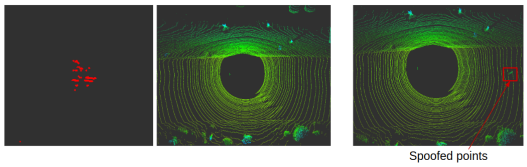


Figure 2.5: Generating the attacker-perturbed 3D point cloud by synthesizing the pristine 3D point cloud with the attack trace to spoof a front-near obstacle 5 meters away from the victim AV.

spoofed obstacle for any of these traces at the output of the LiDAR-based perception pipeline.

Experiment 2: Apply spoofing attack traces at different angles. To understand whether successfully spoofing an obstacle depends on the angle of the spoofed points, in this experiment we inject spoofed points at different locations. More specifically, we uniformly sample 100 different angles out of 360 degrees around the victim AV, and inject the spoofing attack traces reproduced by us. However, we are not able to observe spoofed obstacles for any of these angles.

Experiment 3: Apply spoofing attack traces with different shapes. To understand whether successfully spoofing an obstacle depends on the pattern of the spoofed points, in this experiment we inject points with different spoofing patterns. More specifically, we generate random patterns of spoofed points by randomly setting distances for each point at different angles. We generate 160 points covering 16 vertical lines, 10 points for each line with continuous horizontal angles. To trigger immediate control decision changes in an AV, the spoofed obstacle needs to be close to the victim AV. Thus, we set the generated distances of the spoofed point to be within 4 to 6 meters to the victim AV. We generate 100 different spoofed patterns in total, but we are not able to observe spoofed obstacles for any of these patterns.

Summary. In these experiments, we try various blind spoofing attack strategies directly derived from the state-of-the-art LiDAR spoofing attack, but none of them succeed in generating spoofed obstacles in the LiDAR-based perception pipeline in

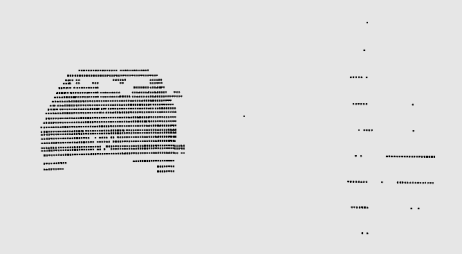


Figure 2.6: The point cloud from a real vehicle reflection (left) and from the spoofing attack (right) in a 64-line HDL-64E LiDAR. The vehicle is around 7 meters in front of the AV.

Baidu Apollo. There are two potential reasons. First, as described earlier, the current attack methodology can only cover a very narrow spoofing angle, i.e., 8 ° of horizontal angle even after our setup improvements. Second, the coverage of vertical angles is limited by the frequency of spoofing laser pulses. Thus, when attacking a LiDAR with more vertical angles, e.g., a 64-line LiDAR, since a 64-line LiDAR takes similar time as a 16-line LiDAR in scanning vertical angles, the attacker cannot spoof more vertical angles than those for a 16-line LiDAR. Thus, the current methodology limits the number of spoofed points, making it hard to generate enough points to mimic an important road obstacle.

To illustrate that, as shown in Fig. 2.6, the point cloud for a real vehicle has a much wider angle and much more points than the attack traces reproduced by us. Thus, blindly applying the spoofing attack cannot easily fool the machine learning based object detection process in the LiDAR-based perception pipeline. In the next section, we explore the possibility of further exploiting machine learning model vulnerabilities to achieve our attack goal.

2.5 Improved Methodology: Adv-LiDAR

As discussed in §2.4, without considering the machine learning model used in LiDAR-based perception, blindly applying existing LiDAR spoofing attacks can hardly

achieve the attack goal of generating front-near obstacles. Since it is known that machine learning output can be maliciously altered by carefully-crafted perturbations to the input [110, 61, 29, 182, 27], we are then motivated to explore the possibility of strategically controlling the spoofed points to fool the machine learning model in LiDAR-based perception. In this section, we first describe the technical challenges after involving adversarial machine learning analysis in this research problem, and then present our solution methodology overview, called *Adv-LiDAR*.

2.5.1 Technical Challenges

Even though previous studies have shown promising results in attacking machine learning models, none of them studied LiDAR-based object detection models, and their approaches have limited applicability to our analysis goal due to three challenges:

First, attackers have limited capability of perturbing machine learning model inputs in our problem. Other than perturbing pixels on an image, perturbing machine learning inputs under AV settings requires perturbing 3D point cloud raw data by sensor attack and bypassing the associated pre-processing process. Therefore, such perturbation capability needs to be quantified and modeled.

Second, optimization-based methods for generating adversarial examples in previous studies may not be directly suitable for our analysis problem due to the limited model input perturbation capability. As shown in §2.7, we find that optimization-based methods are inherently limited due to the nature of our problem, and can only achieve very low success rate in generating front-near obstacles.

Third, in our problem, successfully changing the machine learning model output does not directly lead to successes in achieving our attack goal in AV settings. As detailed later in §2.7, in AV systems such as Baidu Apollo, machine learning model output is post-processed before it is converted to a list of perceived obstacles. Thus, an objective function that can effectively reflect our attack goal needs to be newly

designed.

2.5.2 Adv-LiDAR Methodology Overview

Notation	Description
X	3D point cloud
x	Input feature matrix
X'	Adversarial 3D point cloud
x'	Adversarial input feature matrix
T	Spoofed 3D point cloud
t	Spoofed input feature matrix
T'	Adversarial spoofed 3D point cloud
t'	Adversarial spoofed input feature matrix
$(\mathbf{w_x}, \mathbf{w_y}, \mathbf{w_x})$	3D Cartesian coordinate
L_θ, L_τ	Upper bound of θ, τ during sampling
(u, v)	Coordinate of t
(u', v')	Coordinate of t'
M	Machine learning model
$I.$	Model outputs
$N(u, v)$	4-pixel neighbor at the location (u, v)
$S(\cdot)$	Height Scaling function
$\mathcal{A}$	Spoofing attack capability
$\Phi(\cdot)$	Mapping function (3D$\rightarrow$ 2D)
$Q(M, \cdot)$	Extraction function
$\oplus(\cdot)$	Merge function
$\mathcal{M}(\cdot)$	Gaussian mask
(px, py)	Center points of the Gaussian mask
$f(\cdot)$	Objective function
$\mathcal{L}_{\text{adv}}(\cdot)$	Adversarial loss
$H(\theta, \tau, \epsilon)$	2D Homography Matrix (θ : rotation, ϵ : scaling ; τ : translation)
S_h	Height scaling ratio
$\mathcal{S}_{\text{T}}$	Set of spoofed 3D point cloud
$\mathcal{S}_{\text{t}}$	Set of spoofed input feature matrix
$G_T(T, \cdot)$	Global spatial transformation function for 3D point cloud
$G_t(t, \cdot)$	Global spatial transformation function for input feature matrix

Table 2.3: Notations adopted in this work.

In this section, we provide an overview of our solution methodology, which we call Adv-LiDAR, that addresses the three challenges above. At a high level, to identify adversarial examples for the machine learning model M, we adopt an optimization-based approach, which has shown both high efficiency and effectiveness by previous studies for machine learning models across different domains [28, 41, 160, 158]. To help explain the formulation of the optimization problem, we summarize the notations

in Table 2.3. Specifically, the problem is formulated as follows:

$$\begin{aligned} \min \quad & \mathcal{L}_{\text{adv}}(x \oplus t'; M) \\ \text{s.t.} \quad & t' \in \{\Phi(T') | T' \in \mathcal{A}\} \ \ \& \ \ x = \Phi(X) \end{aligned} \tag{2.1}$$

where X is the pristine 3D point cloud and x represents the corresponding 2D input feature matrix. $\Phi(\cdot)$ is the pre-processing function that maps X into x (§2.2.1). T' and t' are the corresponding adversarial spoofed 3D point cloud and adversarial spoofed input feature matrix. $\mathcal{A}$ is a set of spoofed 3D point cloud generated from LiDAR spoofing attacks. $\mathcal{L}_{\text{adv}}(\cdot; M)$ is the adversarial loss designed to achieve the adversarial goal given the machine learning model M. The constraints are used to guarantee that the generated adversarial examples t' satisfy the spoofing attack capability.

Figure 2.2 overviews the analysis tasks needed to solve the optimization problem. First, we need to conduct an input perturbation analysis that formulates the spoofing attack capabilities $\mathcal{A}$ and merging function $\oplus$. Second, we need to perform a model analysis to design an objective function to generate adversarial examples. Third, as a case study to understand the impact of the attacks at the AV driving decision level, we further perform a driving decision analysis using the identified adversarial examples. More details about these tasks are as follows:

Input perturbation analysis. Formulating $\mathcal{A}$ and $\oplus$ is non-trivial. First, previous work regarding LiDAR spoofing attacks neither provided detailed measurements on the attacker's capability in perturbing 3D point cloud nor expressed it in a closed form expression. Second, point cloud data is pre-processed by several steps as shown in §2.2.1 before turning into machine learning input, which means the merging function $\oplus$ cannot be directly expressed. To address these two challenges, as will be detailed later in §2.6, we first conduct spoofing attacks on LiDAR to collect a set of possible spoofed 3D point cloud. Using such spoofed 3D point cloud, we model the spoofing attack capability $\mathcal{A}$. We further analyze the pre-processing program to

obtain the additional constraints to the machine learning input perturbation, or the spoofed input feature matrix. Based on this analysis, we formulate the spoofed input feature matrix into a differentiable function using global spatial transformations, which is required for the model analysis.

Objective function design and model analysis. As introduced earlier in §2.5.1, in LiDAR-based perception in AV systems, the machine learning model output is post-processed (§ 2.2.1) before turning into a list of perceived obstacles. To find an effective objective function, we study the post-processing steps to extract key strategies of transforming model output into perceived obstacles, and formulate it into an objective function that reflects the attack goal. In addition, we find that our optimization problem cannot be effectively solved by directly using existing optimization-based methods. We analyze the loss surface, and find that this inefficiency is caused by the problem nature. To address this challenge, we improve the methodology by combining global sampling with optimization. Details about the analysis methodology and results are in §2.7 and § 2.8.

Driving decision case study. With the results from previous analysis steps, we can generate adversarial 3D point cloud that can inject spoofed obstacles at the LiDAR-based perception level. To understand their impact at the AV driving decision level, we construct and evaluate two attack scenarios as case studies. The evaluation methodology and results are detailed later in §2.9.

2.6 Input Perturbation Analysis

To generate adversarial examples by solving the above optimization problem in Equation 4.4, we need to formulate merging function $\oplus$ and input feature matrix spoofing capability $\Phi(\mathcal{A})$ as a closed form. In this section, we first analyze the spoofing attack capability ($\mathcal{A}$), and then use it to formulate $\Phi(\mathcal{A})$.

2.6.1 Spoofing Attack Capability

Based on the attack reproduction experiments in §2.4, the observed attack capability ($\mathcal{A}$) can be described from two aspects:

Number of spoofed points. As described in §2.4, even though it is possible to spoof around 100 points after our setup improvement, we find that around 60 points can be reliably spoofed in our experiments. Thus, we consider 60 as the highest number of reliable spoofed points. Noticed that, the maximum number of spoofed points could be increased if the attacker uses more advanced attack equipment. Here, we choose a set of devices that are more accessible (detailed in §2.4) and end up with the ability to reliably spoof around 60 points. In addition, considering that an attacker may use a slower laser or cruder focusing optics, such as in the setup by Shin et al. [132], we also consider 20 and 40 spoofed points in our analysis.

Location of spoofed points. Given the number of spoofed points, the observed attack capability in placing these points are described and modeled as follows:

1. Modify the *distance* of the spoofed point from the LiDAR by changing the delay of the attack laser signal pulses in small intervals (nanosecond scale). From the perspective of spoofed 3D point cloud T, this can be modeled as moving the position of the spoofed points nearer or further on the axis r that connects the spoofed points and the LiDAR sensor by distance Δr (Fig. 2.7 (a)).

2. Modify the *altitude* of a spoofed point within the vertical range of the LiDAR by changing the delay in intervals of 2.304 μs. From the perspective of spoofed 3D point cloud T, this can be modeled as moving the position of the spoofed points from vertical line to vertical line to change the height of it by height Δh (Fig. 2.7 (b)).

3. Modify the *azimuth* of a spoofed point within a horizontal viewing angle of 8° by changing the delay in intervals of 55.296 μs. By moving the LiDAR spoofer

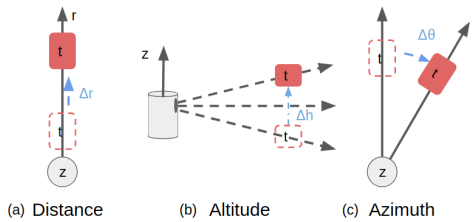


Figure 2.7: Attack capability in perturbing 3D Point Cloud T

to different locations around the LiDAR, it is possible to spoof at any horizontal angle. From the perspective of spoofed 3D point cloud T, this can be modeled as rotating the spoofed points with the LiDAR sensor as the pivot point on the horizontal plane by angle $\Delta\theta$ (Fig. 2.7 (c)).

Therefore, we model the attack capability $\mathcal{A}$ by applying these three modifications to the given spoofed 3D point cloud T. Here the spoofed 3D point cloud is collected by reproducing the sensor spoofing attack. The point number of T can be 20, 40 and 60 to represent different attack capabilities as mentioned before. In the next section, the attack capability $\mathcal{A}$ modeled here is used to model the perturbation of the input feature matrix x.

2.6.2 Input Perturbation Modeling

After analyzing spoofing attack capability $\mathcal{A}$, to formulate $x \oplus t'$ in Equation 2.1, We need to have the following steps: (1) formulating the merging function $\oplus$; (2) modeling the spoofed input feature matrix spoofing capability $\Phi(\mathcal{A})$ based on known spoofing attack capability $\mathcal{A}$. In this section, we first formulate the merging function $\oplus$ by analyzing the pre-processing program. Then we model the spoofed input feature matrix spoofing capability $\Phi(\mathcal{A})$ by expressing t' with spoofed input feature matrix t in a differentiable function using global spatial transformations. Here, spoofed input feature matrix t can be attained with a given spoofed 3D point cloud T by $t = \Phi(T)$.

Formulating merging function ($\oplus$). To model the merging function $\oplus$ op-

erated on x and t', which are in the domain of input feature matrix, we need to first analyze the pre-processing program $\Phi(\cdot)$ that transforms the 3D point cloud X into the input feature matrix x. As described in §2.2.1, the pre-processing process consists of three sub-processes: *coordinate transformation*, *ROI filtering* and *input feature matrix extraction*. The first two processes make minor effects on the adversarial spoofed 3D point cloud T' generated by the spoofing attack we conducted in §2.6. The *coordinate transformation* process has no effect because the adversarial spoofed 3D point cloud T' will be transformed along with the 3D point cloud X. As for the *ROI filtering* process, it filters out 3D point cloud located outside of the road from a bird's-eye view. Therefore, as long as we spoof points on the road, the *ROI filtering process* makes no effect on the adversarial spoofed 3D point cloud T'. The *feature extraction* process, as we mentioned in Section 2.2.1, extracts statistical features such as average height (I_{avg_h}), average intensity (I_{avg_int}), max height (I_{max_h}) and so on.

Because of such pre-processing, the spoofed input feature matrix t' cannot be directly added to the input feature matrix x to attain the adversarial input feature matrix x'. To attain x', we express such "addition" operation ($\oplus$) as a differentiable function shown below. Note that in this equation we do not include a few features in Table 2.1 such as direction and distance since they are either constant or can be derived directly from the features included in the equation.

$$x' = x \bigoplus t' = \begin{bmatrix} I^x_{cnt} + I^{t'}_{cnt} \\ (I^x_{avg\_h} \cdot I^x_{cnt} + I^{t'}_{avg\_h} \cdot I^{t'}_{cnt})/(I^x_{cnt} + I^{t'}_{cnt}) \\ \max(I^x_{max\_h}, I^{t'}_{max\_h}) \\ (I^x_{avg\_int} \cdot I^x_{cnt} + I^{t'}_{avg\_int} \cdot I^{t'}_{cnt})/(I^x_{cnt} + I^{t'}_{cnt}) \\ \sum I^x_{max\_int} \cdot 1\{I^x_{max\_h} = max\{I^x_{max\_h}, I^{t'}_{max\_h}\}\} \end{bmatrix} \tag{2.2}$$

Modeling input feature matrix spoofing capability $\Phi(\mathcal{A})$. To model input feature matrix spoofing capability $\Phi(\mathcal{A})$, it equals to representing adversarial

input feature matrix t' with known spoofed input feature matrix t. We can use global spatial transformations including rotation, translation and scaling, under certain constraints to represent the input feature matrix spoofing capability. Here the translation and scaling transformation interprets the attack capability in terms of modifying the *azimuth* of 3D point cloud while the rotation transformation interprets the attack capability in terms of modifying the *distance* of 3D point cloud from the LiDAR.

Specifically, we apply the global spatial transformation to a set of the spoofed input feature matrix $\mathcal{S}_{\mathrm{t}}$ to formulate the spoofed input feature matrix spoofing capability $\Phi(\mathcal{A})$ and to represent adversarial spoofed input feature matrix t'. For each spoofed input feature matrix $t \in \mathcal{S}_{\mathrm{t}}$, it is mapped from a corresponding spoofed 3D point cloud T such that $t = \Phi(T)$.

We use $t'_{(i)}$ to denote values of the i-th position on the spoofed input feature matrix t' and 2D coordinate $(u'_{(i)}, v'_{(i)})$ to denote its location. t' is transformed from an arbitrary instance t where $t \in \mathcal{S}_{\mathrm{t}}$ by applying a homography matrix $H(\theta, \tau, \epsilon)$. The location of $t_{(i)}$ can be derived as $t'_{(i)}$ as follows:

$$
\begin{aligned}
(u_{(i)}, v_{(i)}, 1)^T &= H \cdot (u'_{(i)}, v'_{(i)}, 1)^T, \\
\textbf{w.r.t.} \qquad H &= \begin{bmatrix} \epsilon(\cos\theta & -\sin\theta) & \tau_x \\ \epsilon(\sin\theta & \cos\theta) & \tau_y \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{2.3}
$$

Notice that here, τ_x/τ_y has a fixed ratio $\tan\theta$ since the translation is performed along the r axis shown in Fig. 2.7 (1). Since θ is dependent on the spoofed input feature matrix we provide for performing the transformation, we align the spoofed input feature matrix in advance to the x axis where $\theta = 0$ and accordingly $\tau_y = \tau_x \tan\theta = 0$. Therefore, we can optimize τ_x alone. Also, this process is equivalent to scaling so we remove ϵ.

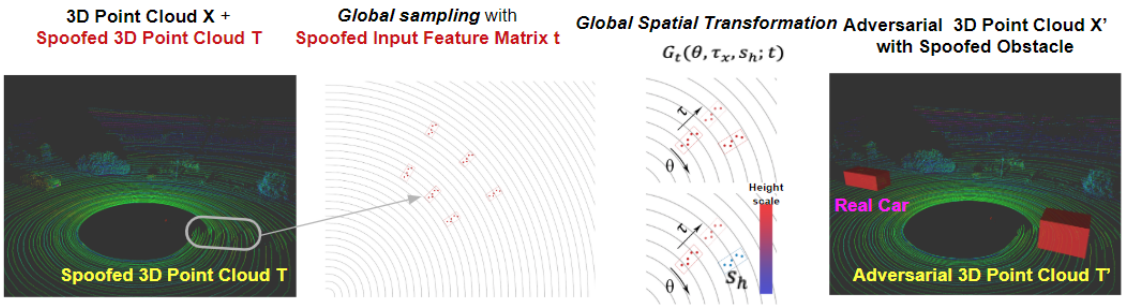


Figure 2.8: Overview of the adversarial example generation process.

We use the differentiable bilinear interpolation [77] to calculate $t'_{(i)}$:

$$t'_{(i)} = \sum_{q \in \mathcal{N}(u_{(i)}, v_{(i)})} t_{(q)}(1 - |u_{(i)} - u_{(q)}|)(1 - |v_{(i)} - v_{(q)}|), \tag{2.4}$$

where $\mathcal{N}(u_{(i)}, v_{(i)})$ represents the 4-pixel neighbors (top-left, top-right,bottom-left, bottom-right) at the location $(u(i), v(i))$.

Further, we can observe that the input feature matrix contains the height information as shown in Table 2.1. So we also optimize a global scale scalar s_h to the height features when generating adversarial spoofed input feature matrix t'. Define $S(t, s_h)$ as the scaling function that multiplies the features which contain the height information by s_h. Based on this transformation, Equation 2.4 will be changed as follows. For simplification, we denote the whole transformation progress as G_t. So $G_t(\theta, \tau_x, s_h; t)$ represents the transformed adversarial spoofed input feature matrix-given spoofed input feature matrix t with transformation parameters θ, τ_x, s_h.

$$\begin{aligned} t'_{(i)} &= G_{t(i)}(\theta, \tau_x, s_h; t) \\ &= \sum_{q \in \mathcal{N}(u_{(i)}, v_{(i)})} S(t_{(q)}, s_h)(1 - |u_{(i)} - u_{(q)}|)(1 - |v_{(i)} - v_{(q)}|) \end{aligned} \tag{2.5}$$

2.7 Generating Adversarial Examples

After modeling the input perturbation, in this section we design the objective function with an effective adversarial loss $\mathcal{L}_{\text{adv}}$, and leverage an optimization method to find the attack transformation parameters that minimize such loss.

Design the adversarial loss $\mathcal{L}_{\text{adv}}$. Unlike previous work that performs the analysis only at the machine learning model level, there is no obvious objective function reflecting our attack goal of spoofing front-near obstacles. Yet, creating an effective objective function has been shown to be essential in generating effective adversarial examples [28]. In order to design an effective objective function, we analyze the post-processing step for the machine learning output. As shown in §2.2.1, in the clustering process, each cell of the model output is filtered by its *objectness* value. After the clustering process, candidate object clusters are filtered by their *positiveness* values. Upon such observation, we designed the adversarial loss $\mathcal{L}_{\text{adv}}$ as follows,

$$\mathcal{L}_{adv} = \sum(1 - Q(x', \text{positiveness})Q(x', \text{objectness}))\mathcal{M}(px, py) \tag{2.6}$$

where $Q(x', \cdot)$ is the function to extract the probabilities of $\cdot$ attribute from model M by feeding in adversarial example x'. $\mathcal{M}$ is a standard Gaussian mask with center coordinate (px, py) which is an attack target position chosen by the attacker. We attain (px, py) by mapping the attack target position in the real world onto the corresponding coordinates of the cell in the input feature matrix using Φ. The adversarial loss is then the summation over all the cells in the input feature matrix of the weighted value described above. By minimizing this designed adversarial loss, it equals to increasing the probability to detect the obstacle of the adversarial spoofed 3D point cloud given the machine learning model M.

Optimization algorithm and our improvement using sampling. With the $\mathcal{L}_{\text{adv}}$ design above, the optimization problem can be directly solved by using the

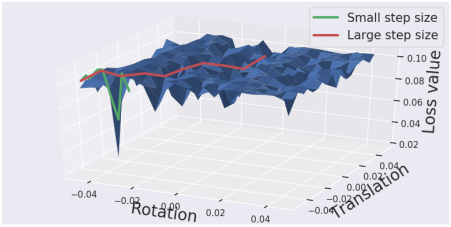


Figure 2.9: Loss surface over transformation parameters θ (rotation) and τ_x (translation). Using a small step size (green line) will trap the optimizing process near a local extreme while choosing a large step size (red line) will be less effective.

Adam optimizer [80] to obtain the transformation parameters θ, τ_x and scalar s_h by minimizing the following objective function:

$$f = \arg\min_{\theta,\tau_x,s_h} \sum (1 - Q(x', \text{positiveness}) Q(x', \text{objectness})) \mathcal{M}(px, py) \tag{2.7}$$

where t' can be obtained by Equation 2.5 and $x' = x \oplus t'$. In this paper, we call this direct solution *vanilla optimization*.

We visualize the loss surface against the transformation parameters in Fig. 2.9. During the vanilla optimization process, we observe that the loss surface over the transformation parameters is noisy at a small scale (green line) and quite flat at a large scale (red line). This leads to the problem of choosing a proper step size for optimization-based methods. For example, choosing a small step size will trap the optimizing process near a local minimum while choosing a large step size will be less effective due to noisy local loss pointing to the wrong direction. Different from Carlini et al. [28] that directly chose multiple starting points to reduces the trap of local minima, the optimization process under our setting is easy to get stuck in bad local minima due to the hard constraints of the perturbations. We propose a way to use *sampling* at a larger scale and to *optimize* at a smaller scale. To initiate the optimization process at different positions, we first calculate the range of the

transformation parameters so that the transformed spoofed 3D point cloud is located in the target area. Then we uniformly take n samples for rotation and translation parameters and compose n^2 samples to initiate with.

Generating adversarial spoofed 3D point cloud. To further construct the adversarial 3D point cloud X', we need to construct adversarial spoofed 3D point cloud T'. Using the transformation parameters $\theta, \tau, \epsilon, s_h$, we can express the corresponding adversarial spoofed 3D point cloud T' such that $t' = \Phi(T')$ with a dual transformation function G_T of G_t. We use $T_{\mathbf{w_x}}, T_{\mathbf{w_y}}, T_{\mathbf{w_z}}$ to denote value of coordinate $(\mathbf{w_x}, \mathbf{w_y}, \mathbf{w_z})$ and T_i to denote the value of intensity for all points in spoofed 3D point cloud T. With transformation parameters $\theta, \tau, \epsilon, s_h$, we can express $T'_{\mathbf{w_x}}, T'_{\mathbf{w_y}}, T'_{\mathbf{w_z}}$ of the transformed adversarial spoofed 3D point cloud T' in Equation 2.8.

$$
\begin{aligned}
T'_i &= T_i \\
\begin{bmatrix} T'_{\mathbf{w_x}} \\ T'_{\mathbf{w_y}} \\ T'_{\mathbf{w_z}} \\ 1 \end{bmatrix} &= \begin{bmatrix} \cos\theta & -\sin\theta & 0 & \tau_x \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & s_h & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} T_{\mathbf{w_x}} \\ T_{\mathbf{w_y}} \\ T_{\mathbf{w_z}} \\ 1 \end{bmatrix}
\end{aligned} \tag{2.8}
$$

Therefore, we can use $T' = G_T(\theta, \tau_x, s_h; T)$ represents the transformed adversarial spoofed 3D point cloud given spoofed 3D point cloud T with transformation parameters θ, τ_x, s_h.

Overall adversarial example generation process. Fig. 2.8 provides an overview of the overall adversarial example generation process. Given 3D point cloud X and spoofed 3D point cloud T (Fig. 2.8 (a)), we first map them via Φ to get corresponding input feature matrix x and spoofed input feature matrix t. Then we apply the sampling algorithm to initialize the transformation parameters θ, τ_x, s_h as shown in Fig. 2.8 (b). After the initialization, we leverage optimizer *opt* to further optimize

the transformation parameters (θ, τ_x, s_h) with respect to the adversarial loss function $\mathcal{L}_{\text{adv}}$ (Fig. 2.8 (c)). With the transformation parameters θ, τ_x, s_h and T, we apply the dual transformation function G_T using the Equation 2.8 to get adversarial spoofed 3D point cloud T'. At last, to obtain the adversarial 3D point cloud X', we append T' to 3D point cloud X (Fig. 2.8 (d)). The entire adversarial example generation algorithm including the optimization parameters is detailed in Appendix A.

2.8 Evaluation and Results

In this section, we evaluate our adversarial example generation method in terms of attack effectiveness and robustness.

Experiment Setup. We use the real-world LiDAR sensor data trace released by Baidu Apollo team with Velodyne HDL-64E S3, which is collected for 30 seconds on local roads at Sunnyvale, CA. We uniformly sample 300 3D point cloud frames from this trace in our evaluation. The attack goal is set as spoofing an obstacle that is 2-8 meters to the front of the victim AV. The distance is measured from the front end of the victim AV to the rear end of the obstacle.

2.8.1 Attack Effectiveness

Fig. 2.10 shows the success rates of generating a spoofed obstacle with different attack capabilities using the vanilla optimization and our improved optimization with global sampling (detailed in §2.7). As shown, with our improvement using sampling, the success rates of spoofing front-near obstacles are increased from 18.9% to 43.3% on average, which is a 2.65× improvement. This shows that combining global sampling with optimization is effective in addressing the problem of trapping in local minima described in §2.7.

Fig. 2.10 also shows that the success rates increase with more spoofed points, which is expected since the attack capability is increased with more spoofed points.

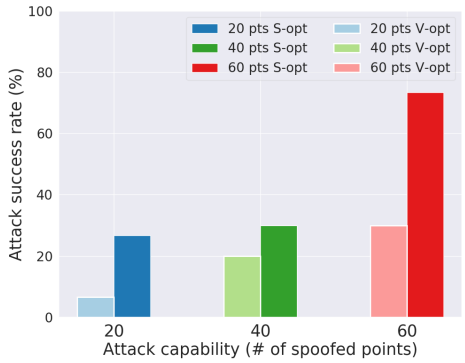


Figure 2.10: Attack success rate of spoofing a front-near obstacle with different number of spoofed points. V-opt refers to vanilla optimization which is directly using the optimizer and S-opt refers to sampling based optimization. We choose Adam [80] as the optimizer in both cases.

In particular, when the attacker can reliably inject 60 spoofed points, which is the attack capability observed in our experiments (§2.4), the attack success rate is able to achieve around 75% using our improved optimization method.

In addition, we observe that the spoofed obstacles in all of the successful attacks are classified as vehicles after the LiDAR-based perception process, even though we do not specifically aim at spoofing vehicle-type obstacles in our problem formulation.

2.8.2 Robustness Analysis

In this section, we perform analysis to understand the robustness of the generated adversarial spoofed 3D point cloud T' to variations in 3D point cloud X and spoofed 3D point cloud $T \in \mathcal{S}_{\mathrm{T}}$. Such analysis is meaningful for generating adversarial spoofed 3D point cloud that has high attack success rate in the real world. To launch the attack in the real world, there are two main variations that affect the results: variation in spoofed points and variation in positions of the victim AV. 1) The imprecision in the attack devices contributes to the variation of the spoofed points. The attacker is able to stably spoof 60 points at a global position as we state in §2.2.2. However, it is difficult to spoof points with precise positions. It is important to understand whether

such imprecision affects the attack success rate. 2) The position of the victim AV is not controlled by the attacker and might vary from where the attacker collected the 3D point cloud. It is important to understand whether such difference affects the attack success rate.

Robustness to variations in point cloud. To measure the robustness to variations in the 3D point cloud, we first select all the 3D point cloud frames that can generate successful adversarial spoofed 3D point cloud. For each of them, we apply its generated adversarial spoofed 3D point cloud to 15 consecutive frames (around 1.5 s) after it and calculate the success rates. Fig. 2.11 shows the analysis results. In this figure, the x-axis is the index for the 15 consecutive frames, and thus the larger the frame index is, the larger the variation is to the original 3D point cloud that generates the adversarial spoofed 3D point cloud. As shown, the robustness for attacks with more spoofed points is generally higher than that for attacks with fewer spoofed points, which shows that higher attack capability can increase the robustness. Particularly, with 60 spoofed points, the success rates are on average above 75% during the 15 subsequent frames, which demonstrates a high degree of robustness. This suggests that launching such attack does not necessarily require the victim AV to appear at the exact position that generates the adversarial example in order to have high success rates.

Robustness to variations in spoofed 3D point cloud. To evaluate the robustness to variations in the spoofed 3D point cloud, for a given spoofed 3D point cloud $T \in \mathcal{S}_{\mathrm{T}}$, we first generate the corresponding adversarial spoofed 3D point cloud T' with a 3D point cloud X. Next, we generate 5 more spoofed 3D point cloud traces $T_1, ..., T_5$ using our LiDAR spoofing attack experiment setup. Next, we use the same transformation that generates T' from T to generate $T'_1, ..., T'_5$, and then combine each of them with X to launch the attack. Table 2.4 shows the average success rates with different attack capabilities. As shown, for all three attack capabilities we are able to

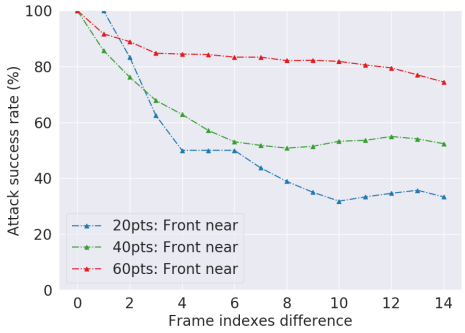


Figure 2.11: The robustness of the generated adversarial spoofed 3D point cloud to variations in 3D point cloud X. We quantify the variation in 3D point cloud X as the frame indexes difference between the evaluated 3D point cloud and the 3D point cloud used for generating the adversarial spoofed 3D point cloud.

Table 2.4: Robustness analysis results of generated adversarial spoofed 3D point cloud to variation in spoofed 3D point cloud $T \in \mathcal{S}_{\mathrm{T}}$. The robustness is measured by average attack success rates.

Targeted position	# Spoofed points		
	20	40	60
2-8 meters	87%	82%	90%

achieve over 82% success rates. With 60 spoofed points, the success rate is as high as 90%. This suggests that launching such attack does not require the LiDAR spoofing attack to be precise all the time in order to achieve high success rates.

2.9 Driving Decision Case Study

To understand the impact of our attacks at the driving decision level, in this section we construct several attack scenarios and evaluate them on Baidu Apollo using simulation as case studies.

Experiment setup. We perform the case study using the simulation feature provided by Baidu Apollo, called *Sim-control*, which is designed to allow users to observe the AV system behavior at the driving decision level by replaying collected

real-world sensor data traces. Sim-control does not consist of a physics engine to simulate the control of the vehicle. Instead, the AV behaves exactly the same as what it plans. Although it cannot directly reflect the attack consequences in the physical world, it can serve for our purpose of understanding the impact of our attacks on AV driving decisions.

For each attack scenario in the case study, we simulate it in Sim-control using synthesized continuous frames of successful adversarial 3D point cloud identified in § 2.8 as input. The experiments are performed on Baidu Apollo 3.0.

Case study results. We construct and evaluate two attack scenarios in this case study[1]:

(1) Emergency brake attack. In this attack, we generate adversarial 3D point cloud that spoofs a front-near obstacle to a moving victim AV. We find that the AV makes a stop decision upon this attack. As illustrated in Fig. 2.12, the stop decision triggered by a spoofed front-near obstacle causes the victim AV to decrease its speed from 43 km/h to 0 km/h within 1 second. This stop decision will lead to a hard brake [1], which may hurt the passengers or result in rear-end collisions. Noticed that, Apollo does implement driving decisions for overtaking. However, for overtaking, a minimum distance is required based on the relative speed of the obstacle. Therefore, with our near spoofed obstacle, the victim AV makes stop decisions instead of overtaking decisions.

(2) AV freezing attack. In this attack, we generate an adversarial 3D point cloud that spoofs an obstacle in front of an AV victim when it is waiting for the red traffic light. We simulate this scenario with the data trace at an intersection with traffic lights. As shown in Fig. 2.13, since the victim AV is static, the attacker can constantly attack and prevent it from moving even after the traffic signal turns green, which may be exploited to cause traffic jams. Noticed that, Apollo does implement driving

[1]Video demos can be found at `http://tinyurl.com/advlidar`

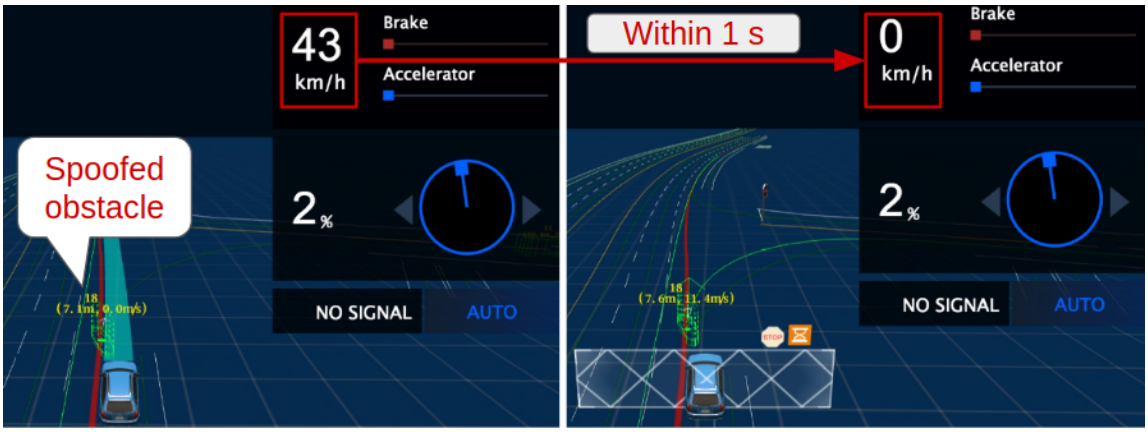


Figure 2.12: Demonstration of the emergency brake attack. Due to the spoofed obstacle, the victim AV makes a sudden stop decision to drop its speed from 43 km/h to 0 km/h within a second, which may cause injuries of passengers or rear-end collisions.

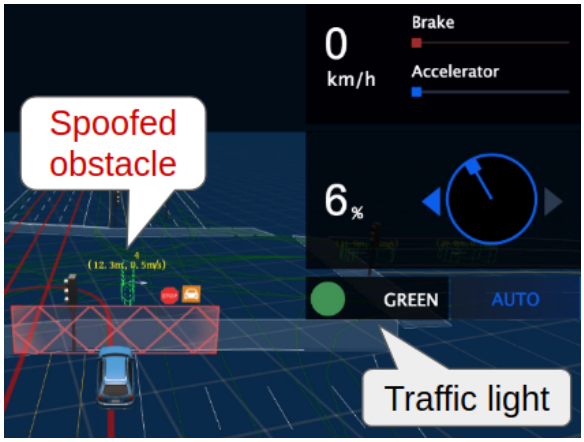


Figure 2.13: Demonstration of the AV freezing attack. The traffic light is turned green but the victim AV is not moving due to the spoofed front-near obstacles.

decisions for deviating static obstacles. However, for deviation or side passing, it requires a minimum distance (15 meters by default). Therefore, with our near spoofed obstacle, the victim AV makes stop decisions instead of side passing decisions.

2.10 Discussion

In this section, we discuss the limitations and generality of this study. We then discuss potential defense directions.

2.10.1 Limitations and Future Work

Limitations in the sensor attack. One major limitation is that our current results cannot directly demonstrate attack performance and practicality in the real world. For example, performing our attack on a real AV on the road requires dynamically aiming an attack device at the LiDAR on a victim car with high precision, which is difficult to prove the feasibility without road tests in the physical world. In this work, our goal is to provide new understandings of this research problem. Future research directions include conducting real world testing. To demonstrate the attack in the real world, we plan to first conduct the sensor attack with LiDAR on top of a real vehicle in outdoor settings. In this setting, the sensor attack could be enhanced by: 1) enlarging the laser spoofing area to solve the aiming problem; 2) adjusting the delay time so that the attacker could spoof points at different angles without moving the attack devices. Then we could apply our proposed methodology to conduct drive-by experiments in different attack scenarios mentioned in §2.9.

Limitations in adversarial example generation. First, we construct adversarial sensor data by using a subset of spoofing attack capability $\mathcal{A}$. Therefore, our analysis may not fully reveal the full potential of sensor attacks. Second, though we have performed the driving decision case study, we did not perform a comprehensive analysis on modules beyond the perception module. That means that the designed objective function can be further improved to more directly target specific abnormal AV driving decisions.

2.10.2 Generality on LiDAR-based AV Perception

Generality of the methodology. Attacking any LiDAR-based AV perception system with an adversarial sensor attack can be formulated as three components: (1) formulating the spoofed 3D point cloud capability $\mathcal{A}$, (2) generating adversarial examples, and (3) evaluating at the driving decision level. Even though our construction of these components might be specific to Baidu Apollo, our analysis methodology can be generalized to other LiDAR-based AV perception systems.

Generality of the results. The formulation of 3D point cloud spoofing capability $\mathcal{A}$ can be generalized as it is independent from AV systems. The success of the attack may be extended to other LiDAR-based AV perception system due to the nature of the LiDAR sensor attack. The LiDAR spoofing attack introduces a spoofed 3D point cloud, which was not foreseen in the training process of machine learning models used in the AV perception system. Therefore, other models are likely to be also vulnerable to such spoofing patterns.

2.11 Related Work

Vehicle systems security. Numerous previous works explore security problems in vehicle systems and have uncovered vulnerabilities in in-vehicle networks of modern automobiles [83, 31, 38], infotainment systems [100], and emerging connected vehicle-based systems [34, 62, 154]. In comparison, our work focuses on vehicle systems with the emerging autonomous driving technology and specifically targets the security of LiDAR-based AV perception, which is an attack surface not presented in traditional vehicle systems designed for human drivers.

Vehicle-related sensor attacks. The sensors commonly used in traditional vehicles have been shown to be vulnerable to attacks. Rouf et al. showed that tire pressure sensors are vulnerable to wireless jamming and spoofing attacks [125].

Shoukry et al. attacked the anti-lock braking system of a vehicle by spoofing the magnetic wheel speed sensor [133]. As AVs become popular, so have attacks against their perception sensors. Yan et al. used spoofing and jamming attacks to attack the ultrasonic sensors, radar, and camera on a Tesla Model S [173]. There have also been two works exploring the vulnerability of LiDAR to spoofing and jamming attacks [111, 132]. In this work, we build on these prior work to show that LiDAR spoofing attacks can be used to attack the machine learning models used for LiDAR-based AV perception and affect the driving decision.

Adversarial example generation. Adversarial examples have been heavily explored in the image domain [64, 160, 28, 110]. *Xie et al.* [165] generated adversarial examples for segmentation and object detection while *Cisse et al.* [41] for segmentation and human pose estimation. Researchers also apply adversarial examples to the physical world to fool machine learning models [59, 60, 18]. Compared to these previous work exploring adversarial examples in the image domain, this work explores adversarial examples for LiDAR-based perception. An ongoing work [155] studies the generation of 3D adversarial point clouds. However, such attack focuses on the digital domain and can not be directly applied to the context of AV systems. In comparison, our method is motivated to generate adversarial examples based on the capability of sensor attacks to fool the LiDAR-based perception models in AV systems.

2.12 Conclusion

In this work, we perform the first security study of LiDAR-based perception in AV systems. We consider LiDAR spoofing attacks as the threat model, and set the attack goal as spoofing front-near obstacles. We first reproduce the state-of-the-art LiDAR spoofing attack, and find that blindly applying it is insufficient to achieve the attack goal due to the machine learning-based object detection process. We thus perform analysis to fool the machine learning model by formulating the attack task

as an optimization problem. We first construct the input perturbation function using local attack experiments and global spatial transformation-based modeling, and then construct the objective function by studying the post-processing process. We also identify the inherent limitations of directly using optimization-based methods and design a new algorithm that increases the attack success rates by 2.65× on average. As a case study, we further construct and evaluate two attack scenarios that may compromise AV safety and mobility. We also discuss defense directions at AV system, sensor, and machine learning model levels.

CHAPTER III

Vulnerability Status of LiDAR-based Perception against the Object Reshaping Attack

3.1 Introduction

Machine learning, especially deep neural networks (DNNs), have achieved great successes in various domains, [42, 68, 49, 134]. Several safety-critical applications such as autonomous vehicles (AV) have also adopted machine learning models and achieved promising performance. However, recent studies show that machine learning models are vulnerable to adversarial attacks [64, 25, 158, 160, 157, 137]. In these attacks, small perturbations are sufficient to cause various well-trained models to output "adversarial" prediction. In this paper we aim to explore similar vulnerabilities in today's autonomous driving systems.

Such adversarial attacks have been largely explored in the image domain. In addition, to demonstrate such attacks pose a threat in the real world, some studies propose to generate physical stickers or printable textures that can confuse a classifier to recognize a stop sign [18, 59]. However, an autonomous driving system is not merely an image-based classifier. For perception, most autonomous driving detection systems are equipped with LiDAR (Light Detection And Ranging) or RADAR (Radio Detection and Ranging) devices which are capable of directly probing the surrounding 3D

environment with laser beams. This raises the doubt of whether texture perturbation in previous work will affect LiDAR-scanned point clouds. In addition, the LiDAR-based detection system consists of multiple non-differentiable steps, rather than a single end-to-end network, which largely limits the use of gradient-based end-to-end attacks. These two key obstacles not only invalidate previous image-based approaches, but also raise several new challenges when we want to construct an adversarial object: 1) LiDAR-based detection system projects 3D shape to a point cloud using physical LiDAR equipment. The point cloud is then fed into the machine learning detection system. Therefore, how shape perturbation affects the scanned point cloud is not clear. 2) The preprocessing of the LiDAR point clouds is non-differentiable, preventing the naive use of gradient-based optimizers. 3) The perturbation space is limited due to multiple aspects. First, we need to ensure the perturbed object can be reconstructed in the real world. Second, a valid LiDAR scan of an object is a constrained subset of point cloud, making the perturbation space much smaller compared to perturbing the point cloud without any constraint [155].

In this paper, we propose *LiDAR-Adv* to address the above issues and generate adversarial object against real-world LiDAR system as shown in Figure 5.6. We first simulate a differentiable LiDAR renderer to bridge the perturbations from 3D objects to LiDAR scans (or point cloud). Then we formulate 3D feature aggregation with a differentiable proxy function. Finally, we devise different losses to ensure the smoothness of the generated 3D adversarial objects. To better demonstrate the flexibility of the proposed attack approach, we evaluate our attacking approach under two different attacking scenarios: 1) *Hiding Object*: synthesizing an "adversarial object" that will not be detected by the detector. 2) *Changing Label*: synthesizing an "adversarial object" that is recognized as a specified adversarial target by the detector. We also compare *LiDAR-Adv* with the evolution algorithm in the blackbox setting.

To evaluate the real-world impact of *LiDAR-Adv*, we 3D print out the generated adversarial objects and test them on the Baidu Apollo autonomous driving platform, an industry-level system which is not only highly adopted for research purpose but also actively used in industries. We show that with 3D perception and a production-level multi-stage detector, we are able to mislead the autonomous driving system to achieve different adversarial targets.

To summarize, our contributions are as follows: (1) We propose *LiDAR-Adv*, an end-to-end approach to generate physically plausible adversarial objects against LiDAR-based autonomous driving detection systems. To the best of our knowledge, this is the first work to exploit adversarial objects for such systems. (2)We experiment on Apollo, an industry-level autonomous driving platform, to illustrate the effectiveness and robustness of the attacks in practice. We also compare the objects generated by *LiDAR-Adv* with evolution algorithm to show that *LiDAR-Adv* can provide smoother objects. (3) We conduct physical experiments by 3D-printing the optimized adversarial object and show that it can consistently mislead the LiDAR system equipped in a moving car.

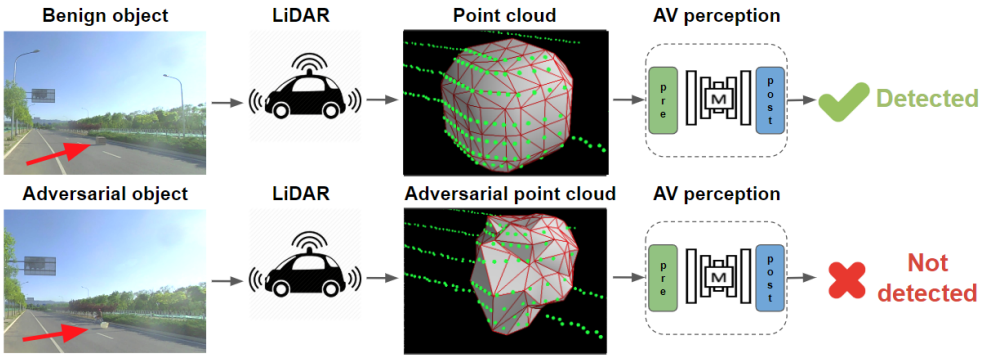


Figure 3.1: Overview of *LiDAR-Adv*. The first row shows that a normal box will be detected by the LiDAR-based detection system; while the generated adversarial object with similar size in row 2 cannot be detected.

3.2 Related work

Image-space adversarial attacks Adversarial examples have been heavily explored in 2D image domains [64, 28, 107, 104, 158]. Various works [59, 85, 18] start to study robust physical adversarial examples. *Evtimov et al.* [59] has created printable 2D stickers to attach to a stop sign and cause a detector to predict wrong labels. Following this line, there are also works [159, 92] aiming to optimize the 3D shapes to show that even the surface geometry itself can produce adversarial behaviors.

In this work, we exploit the object surfaces to generate adversarial objects, and one fundamental challenge that differentiates our work from the previous ones is: the sensor in a LiDAR-based system directly probes the 3D environment as the input, bypassing surface textures of the adversarial objects. This means we may only rely on shape geometry to perform any attacks. On the other hand, compared to prior works that have shown successes on attacking single models, it is worth noting that the victim model which we experiment on (Apollo), is not merely an end-to-end deep learning model but an industry-level autonomous driving platform that consists of multiple non-differentiable parts.

Adversarial point clouds *Xiang et al.* [155] show a proof of concept, that models taking raw 3D point clouds as input [114] can be vulnerable to adversarial point clouds. However, this approach is only evaluated with a single digital model. It is not clear that the generated point clouds can form plausible 3D shape surfaces, or it can be reconstructed through LiDAR scans. While in our approach, though the victim model takes point clouds as input similarly, these point clouds have to satisfy extra constraints such as: all points have to be the intersections of the laser beams and the object surfaces. We address this challenge by proposing a differentiable renderer which simulates the reconstructed laser beams projecting onto object surfaces. As we will show later, when the object moves, the point cloud changes in accordance with the laser hits, and how to enforce the robustness against such LiDAR scans is

non-trivial.

3.3 Generating Adversarial Object Against LiDAR-based Detection

In this section, we will formulate the problem first and describe the adversarial goals and challenges. We then describe our whitebox method *LiDAR-Adv* which aims to tackle the challenges and fulfill diverse adversarial goals. Finally, we propose an evolution-based attack method for blackbox settings.

3.3.1 Methodology overview

Given a 3D object S in a scene, as stated in the background, after the scene is scanned by a LiDAR sensor, a point cloud X is then generated based on S so that $X = \text{render}(S, \text{background})$ For preprocessing, this point cloud X is sliced and aggregated to generate x, which is a $H \times W \times 8$ feature vector, and we call this aggregation process as Φ: $x = \Phi(X)$. Then a machine learning model M maps this 2D feature $x \in R^{H \times W \times 8}$ to $O = M(x)$, where $O \in R^{H \times W \times 7}$ (see Sec. **??** for concrete output meanings). O is then post-processed by a clustering process Ψ to generate the confidence y_{conf} and label y_{label} of detected obstacles so that $(y_{conf}, y_{label}) = \Psi(O)$ An adversarial attacker aims to manipulate the object S to achieve the adversarial goals. Here we define two types of adversarial goals: 1) *Hiding object*: Hide an existing object S by manipulating S; 2) *Changing label*: Change the label y of the detected object S to a specified target y'.

To achieve the above adversarial goals in LiDAR-based detection is non-trivial, and there are the following challenges: 1) **Multiple pre/post-processing stages.** Unlike the adversarial attacks on traditional image-space against machine learning tasks such as classification and object detection, the LiDAR-based detection here is

not a single end-to-end learning model, It consists of the differentiable learning model M and several non-differentiable parts including preprocessing and post-processing. Thus, the direct gradient based attacks are not directly applicable. 2) **Manipulation constraints.** Instead of directly manipulating the point cloud X as in [155], we manipulate the 3D shape of S given the limitation of LiDAR. The points in X are the intersections of laser beams and object surfaces and cannot move freely, so the perturbations on each point may affect each other. Keeping the shape plausible and smooth adds additional constraints [159]. 3) **Limited Manipulation Space.** Consider the practical size of the object versus the size of the scene that is processed by LiDAR, the 3D manipulation space is rather small ($< 2\%$ in our experiments), as shown in Fig. 5.6.

Given the above challenges, we design an end-to-end attacking pipeline. In order to facilitate gradient-based algorithms, we implement an approximated differentiable renderer R , which simulates the functionality of LiDAR, to intersect a set of pre-defined rays with the 3D object surface (S) consisting of vertices V and faces W. The points at the intersections form the raw point cloud X. After preprocessing, the point cloud is then fed to a preprocessing function Φ to generate the feature map $x = \Phi(X)$. The feature map x is then taken as input for a machine learning model M to obtain the output metrics $O = M(x)$.

The whole progress can be symbolized as $F(S) = M(\Phi(R(S)))$. Note that by differentiating the renderer R, the whole process $F(*) = M(\Phi(R(*)))$ is differentiable w.r.t. S. In this way, we can manipulate S to generate adversarial S_{adv} via our designed objective function operating on the final output $F(S)$.

3.3.2 Approximate differentiable renderer

LiDAR simulation The renderer R simulates the physics of a LiDAR sensor that probes the objects in the scene by casting laser beams. The renderer first takes a

mesh S as input, and compute the intersections of a set of predefined ray directions to the meshes in the scene to generate point cloud X. After depth testing, the distance along each beam is then captured, representing the surface point of a mesh that it first encounters, as if a LiDAR system receives a reflection from an object surface. Knowing ray directions of the beams, the exact positions of the intersection points can be inferred from the distance, in the form of point clouds X.

Real background from a road scene We render our synthetic object onto a realistically captured point cloud. First, we obtain the 3D scan of a road scene, using the LiDAR sensor Riegl VMX-1HA mounted on a car. Then, we obtain the laser beam directions by computing the normalized vectors from the origin (LiDAR) pointing to the scanned points. This fixed set of ray directions are then used for rendering our synthetic objects throughout the paper. Note that we can also manually set ray directions given sensor specifications, but it will be less real, because it may not model the noises and fluctuations that occur in a real LiDAR sensor.

Hybrid rendering of synthetic objects onto a realistic background Given the ray directions reconstructed from the background point cloud, a subset will intersect with the object, forming the point cloud for the object of interest. The corresponding background points are then removed since these background points are occluded by the foreground object. In this way, we obtain a semi-real synthetic point cloud scene: the background points come from the captured real data; the foreground points are physically accurate simulated based on the collected real data.

3.3.3 Differentiable proxy function for feature aggregation

As described earlier, in the preprocessing of Apollo, it aggregates the point cloud into hardcoded 2D features, including **count**, **max height**, **mean height**, **intensity** and **non-empty**. These operations are non-differentiable. In order to apply end-to-

end optimizers to for our synthetic object S, we need to flow the gradient through the feature aggregation step, with the help of our proxy functions.

Given a point cloud X with coordinate (u^X, v^X, w^X), and we hope to count the number of points falling into the cells of a 3D grid $G \in \mathbb{R}^{H \times W \times P}$. For a point X_i with location$(u^{X_i}, v^{X_i}, w^{X_i})$, we increase the count of 8 cells: the centers of these 8 cells form a cube, and the point X_i is inside this cube. Specifically, we increase the count of 8 cells using trilinear weights:

$$G(u_i, v_i, w_i) = \sum_p (1 - d(u_p, u^{X_i})) \cdot (1 - d(v_p, w^{X_i})) \cdot (1 - d(w_p, w^{X_i})), \tag{3.1}$$

where $p \in \mathcal{N}(u^{X_i}, v^{X_i}, w^{X_i})$ are the indices of the 8-pixel neighbors at location $(u^{X_i}, v^{X_i}, w^{X_i})$ and $d(\cdot, \cdot)$ represents the L_1 distance. The **count** feature $x_{\textbf{count}}$ is the value $G_p = G(u_i, v_i, w_i)$ computed for each grid i. Note that this feature is no longer an integer and can have non-zero gradients w.r.t the point coordinates.

We then use this "soft count" feature to further compute "mean height" and "max height" features. For simplicity, we first define a constant height matrix $T \in \mathcal{R}^{H,W,P}$, where $T(.,.,p) = p, p \in \{1...P\}$. This matrix stores the height of each cell. Next, we can formulate the **mean height** $x_{\textbf{mean-height}}$ and **max height** $x_{\textbf{max-height}}$ using soft count G:

$$x_{\textbf{mean-height}} = \cdot \frac{\sum_{p \in P} G_p \circ T_p}{\sum_{p \in P} G_p + \epsilon} \qquad \text{and} \qquad x_{\textbf{max-height}} = \max_p \operatorname{sign}\left(G(.,.,p)\right) \circ T(.,.,p) \tag{3.2}$$

where $\epsilon = 1e^{-7}$ to prevent zero denominators. Note that the sign function is non-differentiable, so we approximate the gradient using $\operatorname{sign}(G) = G$ during back propagation. The feature **intensity** has the similar formulation of **height** so we omit them here. The feature **non-empty** is formulated as $x_{\textbf{non-empty}} = \operatorname{sign}(G)$.

We denote the above trilinear approximator as Φ', in constrast to the original non-differentiable preprocessing step Φ. A visualization of output of our $\Phi'(X)_{\textbf{count}}$ compared to the original $\Phi(X)_{\textbf{count}}$ is shown in Sec. **??**. Since our approximation

introduces differences in counting, $\Phi'(X)$ is not strictly equal to $\Phi(X)$, resulting in different obj values of the final model prediction. We observe that this difference will raise new challenges to transfer the adversarial object generated based on Φ' to Φ. To solve this problem, we reduce the difference between Φ' and Φ, by replacing the L1 distance d in Eq. 3.1 with $d(u_1, u_2) = 0.5 + 0.5 \cdot \tanh(\mu \cdot (u_1 - u_2 - 1))$ where $\mu = 20$. We name this approximation "tanh approximator" and denote it Φ''. We observe that the input difference between Φ'' and the original Φ is largely reduced compared to Φ', allowing for smaller errors of the model predictions and better transferability. To extend our approximator and further reduce the gap betweer Φ'' and Φ, we interpolate the distance: $d(u_1, u_2) = \alpha \cdot (0.5 + 0.5 \cdot \tanh(5\mu \cdot (u_1 - u_2 - 1))) + (1 - \alpha) \cdot (u_1 - \lfloor u_2 \rfloor)$, where α is a hyper-parameter balancing the accuracy of approximation and the availability of gradients.

3.3.4 Objective functions

Our objective is to generate a synthetic adversarial object S^{adv} from an original object S by perturbing its vertices, such that the LiDAR-based detection model will make incorrect predictions. We first optimize S^{adv} against the semi-real simulator detection model M.

$$\mathcal{L}(S^{\text{adv}}) = \mathcal{L}_{\text{adv}}(S^{\text{adv}}, M) + \lambda \mathcal{L}_{\mathbf{r}}(S^{\text{adv}}; S) \tag{3.3}$$

The objective function $\mathcal{L}$ consists of two losses. $\mathcal{L}_{\text{adv}}$ is the adversarial loss to achieve the target goals while the $\mathcal{L}_{\mathbf{r}}$ is the distance loss to keep the properties of the "realistic" adversarial 3D object S^{adv}. We optimize the objective function by manipulating the vertices. The distance loss is defined as follows:

$$\mathcal{L}_{\mathbf{r}} = \sum_{\boldsymbol{v}_i \in V} \sum_{q \in \mathcal{N}(\boldsymbol{v}_i)} \|\Delta \boldsymbol{v}_i - \Delta \boldsymbol{v}_q\|_2^2 + \beta \sum_{\boldsymbol{v}_i \in V} \|\Delta \boldsymbol{v}_i\|_2^2, \tag{3.4}$$

where $\Delta \boldsymbol{v}_i = \boldsymbol{v}_i^{\text{adv}} - \boldsymbol{v}_i$ represents the displacement between the adversarial vertex $\boldsymbol{v}_i^{\text{adv}}$ and pristine vertex $\boldsymbol{v}_i$. β is the hyperparameter balancing these two losses. The first losses [159] is a Laplacian loss preserving the smoothness of the perterbation applied on the adversarial object S^{adv}. The second part is the $L2$ distance loss to limit the magnitude of perturbation.

Objective: hide the inserted adversarial object As introduced in the background section, the existence of the object highly depends on the "positiveness" metric. $H(*, M, S)$ denotes a function extracting $*$ metric from the model M given an object S. $\mathcal{A}$ is the mask of the target object's bounding box. Our adversarial loss is represented as follows:

$$\mathcal{L}_{\text{adv}} = H(\text{pos}, M, S) * \mathcal{A} \tag{3.5}$$

Objective: changing label In order to change the predicted label of the object, it needs to increase the logits of the target label and decrease the logits of the ground-truth label. Moreover, it also needs to preserve the high positiveness. Based on this, our adversarial loss is written as

$$\mathcal{L}_{\text{adv}} = (-H(\text{c}_{y'}, M, S) + H((c)_y, M, S) \cdot \mathcal{A} * H(\text{pos}, M, S) \tag{3.6}$$

In order to ensure that adversarial behaviors still exist when the settings are slightly different, we create robust adversarial objects that can perform successful attacks within a range of settings, such as different object orientations, different positions to the LiDAR sensor *etc*. To achieve such goal, we sample a set of physical transformations to optimize the loss expectation. In reality, we create a victim set D by rendering the object S at different positions and orientations. Instead of optimizing an adversarial object S by attacking single position and orientation, we generate an universal adversarial object S to attack all positions and orientations in the victim

set D.

3.3.5 Blackbox Attack

In reality, it is possible that the attackers do not have complete access to the internal model parameters, i.e., the model is a black box. Therefore, in this subsection, we also develop an evolution-based approach to perform blackbox attack.

In evolution, a set of individuals represent the solutions in the search space, and the fitness score defines how good the individuals are. In our case, the individuals are mesh vertices of our adversarial object, and the fitness score is $-\mathcal{L}(S^{\text{adv}})$. We initialize m mesh vertices using the benign object S. For each iteration, new population of n mesh vertices are generated by adding random perturbations, drawn from a Gaussian distribution $\mathcal{N}(0, \sigma)$, to each mesh vertices in the old population. m mesh vertices with top fitness scores will remain for the next iteration, while the others will be replaced. We iterate the process until we find a valid solution or reach a maximum number of steps.

3.4 Experiments

In this section, we first expose the vulnerability of the LiDAR-based detection system via the evolution-based blackbox algorithm by achieving the goal of "hiding object", because missing obstacles can cause accidents in real life. We then show the qualitative results and quantitative results of *LiDAR-Adv* under whitebox settings. In addition, we also show that *LiDAR-Adv* can achieve other adversarial goals such as "changing label". Moreover, the point clouds are continuously captured in real life, so attacks in a single static frame may not have much effect in real-world cases. Therefore, in our experiments, we generate a universal robust adversarial object against a victim dataset which consists of different orientations and positions. We 3D-print such universal adversarial object and conduct the real-world drive-by experiments, to

show that they indeed can pose a threat on road.

3.4.1 Experimental setup

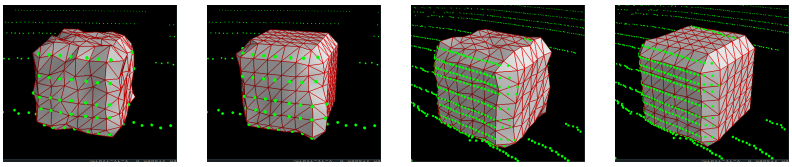
We conduct the evaluation on the perception module of Baidu Apollo Autonomous Driving platform (V2.0). We initialize the adversarial object as a resampled 3D cube-shaped CAD model using MeshLab [40]. For rendering, we implement a fully differential LiDAR simulator with predefined laser beam ray directions extracted from a real scene captured by the Velodyne HDL-64E sensor, as stated in § 3.3.2. It has around 2000 angles in the azimuth angle and around 60 angles in the elevation angle. We use Adam optimizers [80], and choose λ as 0.003 in Eq. 3.3 using binary search. For the evolution-based blackbox algorihtm , we choose $\sigma = 0.1$, $n = 500$ and $m = 5$.

3.4.2 Vulnerability analysis

Here, we first show the existence of the vulnerability using our evolution-based blackbox attacks, with the goal of "hiding object". We generate adversarial objects in different size (50cm and 75cm in edge length). For each object, we select 45 different position and orientation pairs for evaluation, and the results are shown in Table 3.1. The results indicate that the LiDAR-based detection system is vulnerable. The visualization of the adversarial object is shown in Figure 3.2(a) and (c).

3.4.3 *LiDAR-Adv* with different adversarial goals

After showing the vulnerability of the LiDAR-based detection system, here we focus on whitebox settings to explore what a powerful adversary can do, since "the design of a system should not require secrecy" [130]. Therefore, we evaluate the effectiveness of our whitebox attack *LiDAR-Adv* with the goal of "hidding object". We also evaluate the feasibility of *LiDAR-Adv* to achieve another goal of "changing label".



(a) 50cm: Evo. (b) *LiDAR-Adv* (c) 75cm: Evo. (d) *LiDAR-Adv*

Figure 3.2: Adversarial meshes of different sizes can fool the detectors even with more LiDAR hits. We generate the object with *LiDAR-Adv* and evolution-based method (Evo.).

Table 3.1: Attack success rate of *LiDAR-Adv* and evolution based method under different settings.

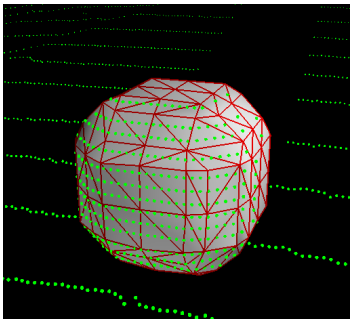
Attacks	Object size	
	50cm	75cm
LiDAR-Adv	32/45 (71%)	23/45 (51%)
Evolution-based	28/45 (62%)	16/45 (36%)

Hiding object We follow the same settings as in the above sections, and Table 3.1 shows the results. We find that *LiDAR-Adv* can achieve 71% attack success rate with size 50cm. The attack success rate is consistently higher than the evolution-based blackbox attacks. Figure 3.2 (b) and (c) show the visualizations of the adversarial objects. We visually observe that the adversarial objects generated by *LiDAR-Adv* are smoother than that of evolution.

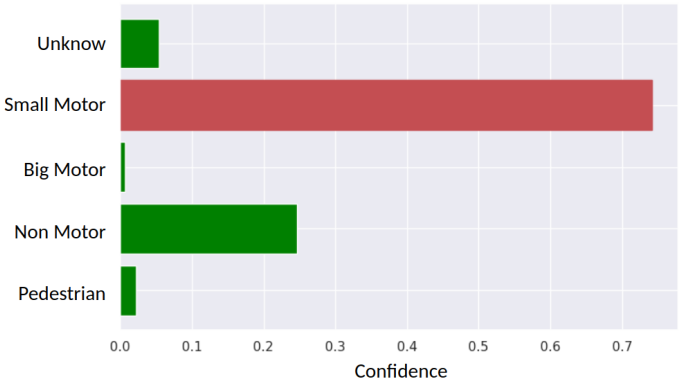
Changing label The result shown in Figure 3.3 indicates that we can successfully change the label of the object. We also experiment with different initial shapes and target labels.

3.4.4 *LiDAR-Adv* on generating robust physical adversarial objects

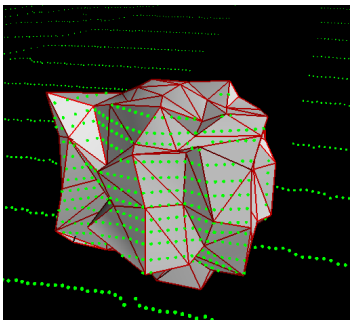
To ensure the generated *LiDAR-Adv* preserves adversarial behaviors under various physical conditions, we optimize the object by sampling a set of physical transformations such as possible positions and orientations. We show that the generated robust



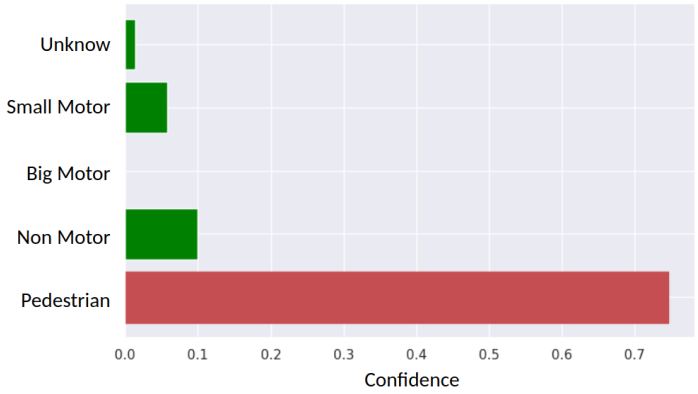
(a) 3D mesh (Benign)



(b) Predictions (Benign)



(c) 3D Mesh (Adv.)



(d) Predictions (Adv.)

Figure 3.3: The adversarial mesh generated by *LiDAR-Adv* is mis-detected as a "Pedestrian".

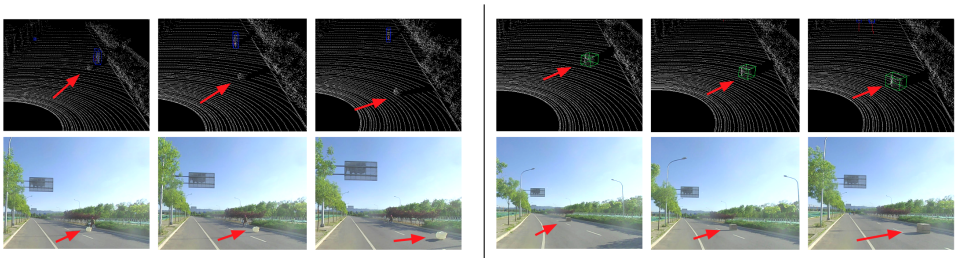
adversarial object is able to achieve the attack goal of hiding object with a high success rate in Table 3.2. An interesting phenomenon is that some attack performance under the unseen settings is even better than that within the controlled environment. This implies that our adversarial objects are robust enough to generalize to unseen settings.

Furthermore, we evaluate the generated robust adversarial object in the physical world by 3D printing the generated object. We collect the point cloud data using a Velodyne HDL-64E sensor with a real car driving by and evaluate the collected traces on the LiDAR perceptual module of Baidu Apollo. As shown in Figure 3.4(a), we find that the adversarial object is not detected around the target position among

Table 3.2: Attack success rates of *LiDAR-Adv* at different positions and orientations under both controlled and unseen settings.

Controlled Setting		Unseen Setting			
		Distance (cm)		Orientation (°)	
Distance (cm) & Orientation (°)	Attack	0-50	50-100	0-5	0-10
$\{0, \pm 50\} \times \{0, \pm 2.5, \pm 5\}$	41/45	96/100	91/100	10/10	9/10
$\{0, \pm 50\} \times \{0, \pm 2.5, \pm 5\}$	43/45	96/100	90/100	8/10	10/10

all 36 different frames. To compare, the box object (in Figure 3.4(b)) is detected in 12 frames among all 18 frames. The number of total frames is different due to the different vehicle speed.



(a) Adversarial object (b) Benign object

Figure 3.4: Results of physical attack. Our 3D-printed robust adversarial object by *LiDAR-Adv* is not detected by the LiDAR-based detection system in a moving car. Row 1 shows the point cloud data collected by LiDAR sensor, and Row 2 presents the corresponding images captured by a dash camera.

3.5 Conclusion

We show that LiDAR-based detection systems for autonomous driving are vulnerable against adversarial attacks. By integrating our proxy differentiable approximator, we are able to generate robust physical adversarial objects. We show that the adversarial objects are able to attack the Baidu Apollo system at different positions with various orientations. We also show *LiDAR-Adv* can generate much smoother object than evolution based attack algorithm. Our findings raise great concerns about the security of LiDAR systems in AV, and we hope this work will shed light on potential

defense methods.

CHAPTER IV

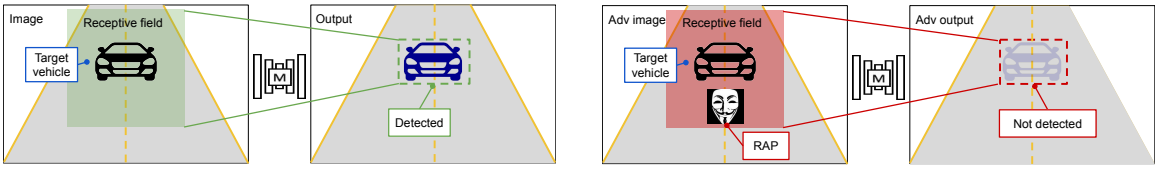
Vulnerability Causality Analysis on Camera-based Perception

4.1 Introduction

In recent years, deep convolutional neural networks (CNNs) have been pushing and setting the state-of-the-art on various computer vision tasks including image recognition [84, 135, 141, 140, 68, 69, 143], semantic segmentation [192, 181, 33], object detection [119, 116, 93], image captioning [120, 121] and many more.

One important basic concept in deep CNNs is the *receptive field* of certain units in the network. Unlike fully connected networks where the value of each neuron depends on the entire input to the network, a unit in convolutional networks only depends on a region of the input, and that region in the input is the receptive field for that unit.

From the model efficiency perspective, the concept of the receptive field is naturally more important to *location aware* tasks such as object detection and semantic segmentation since such tasks require objects to be represented at multiple scales in order to recognize small and large instances. Since any units in an input image outside of the receptive do not affect the value of that unit, it is essential to design the CNN to ensure that its receptive field covers the entire relevant image region. As models evolved, from AlexNet [84], to VGG [135], to ResNet [68] and Inception [142],



(a) Benign case (b) Adversarial case

Figure 4.1: Remote adversarial patch (RAP) attack overview. The goal of the attack is to mislead the model predictions of the target vehicle with a RAP that is not overlapped with the target vehicle, by leveraging the large receptive field. In (a) the target vehicle is detected while in (b) the target vehicle is not detected with the existence of a RAP.

the receptive fields increased. While a larger receptive field of a later model is a natural consequence of the increased number of layers and deeper networks, the effective receptive field of units can only grow linearly with layers limiting the processing of high-resolution inputs. Therefore, dilated convolution [180] has been proposed which allows for enlarging the receptive fields while only growing the number of parameters logarithmically; multi-scale methods [191, 33] are invented to capture the image features at different scales; More recently, attention-based methods are also used for capturing global context [63, 148, 194] and even replacing the CNN backbones with the Transformer that embeds with attentions [168].

Apart from the model efficiency perspective, current machine learning models have been shown vulnerable to evasion attacks, where an adversary adds a small perturbation to a test example for misleading the models [139, 28]. Although the design of large receptive field introduces advantages to improve model efficiency, the security effect of the large receptive field is unexplored. Moreover, prior works [26, 165] have widely studied the type of adversarial perturbation which manipulate the entire images. However, it is unrealistic in real-world physical scenarios as it is hard for the attacker to add perturbation to each object of a given scene.

Therefore, there has been significant prior progress on generating physically possible adversarial examples by generating adversarial patches and adding the patches

solely into the targeted object of the whole image. Taking the complicated tasks such as segmentation and object detection in the safety-critical application autonomous driving system as an example, prior works [165, 61] have shown that adding the adversarial patch to the targeted object could arbitrarily mislead the prediction result (e.g bounding box, segmentation map) of the targeted object. Such patch attacks, however, require the attacker to access the targeted object to add the perturbations, which is less realistic in practice.

Compared to other methods for solely generating adversarial patches, the scope of this paper is beyond only conducting the attack, but more importantly, understanding the causality of the vulnerability and providing a guideline for future model architecture designs for more robust models (e.g. blindly increasing receptive field sizes makes models more vulnerable). Motivated by this, we carefully design experiments from task, model and dataset perspectives. In particular, we chose the semantic segmentation task where pixel-wise predictions are made and thus enable a fine-grained analysis of the receptive field size. We chose models with key architecture-level innovations (e.g. dilated-convolution, multi-scaling and attention). For dataset, we chose a dataset with high-resolution images since high-resolution images are not fully covered by the receptive field sizes. This dataset helps distinguish the robustness of models with different receptive field sizes. Compared to our work, a concurrent work IPatch [103], however, only focuses on evaluating the attack without further digging into the causality of the vulnerability. This focus is also reflected from several choices made by IPatch. For example, 1) low-resolution image datasets are used; 2) fixed time for optimizing patches instead of fixed optimization iterations are used. These choices make different model robustness less distinguishable and even less comparable.

Summary of Contributions:

- We present the *first* study on model robustness related to receptive field sizes to adversarial attacks on semantic segmentation. We demonstrate a threat

on semantic segmentation models called *remote adversarial patch (RAP)*, due to the abuse of large receptive fields in various modern models. As shown in Figure 4.1, unlike previous adversarial attacks that manipulate the entire input image, the RAP attack only manipulates a localized small area that is not overlapped with the target attack area. We detailed the methodology for generating RAP in §4.3.

- We identify that the RAP vulnerability is due to the large receptive field and the relation between the vulnerability level and the receptive field size in §4.4.1. We find that the model with a larger receptive field size is more vulnerable to the RAP attack. Also, we find that the RAP attack is more effective when the distance between the patch and the target area is smaller.

- We study the robustness of different model designs against RAP. In §4.4.2, we show that methods increasing receptive make models more vulnerable. Also, among different multi-scale based methods, standard pyramid pooling module (PPM) is more vulnerable than atrous spatial pyramid pooling (ASPP). Among different backbones, transformer-based backbone is more vulnerable than CNN-based backbone. And overall, attention-based models tend to be more vulnerable.

- To demonstrate the severity of the discovered threat, we conduct and evaluate the *object removing attack*, a special case RAP, which removes the model prediction of the target object (i.e., misleads the model prediction of a target object to a target class) without accessing and adding perturbation on it . We showed in §4.5.1 that with the proposed RAP attack, the mIoU of the target area drops from 40% and even is close to 0 on several state-of-the-art models. We also extend the success to the physical world and show in §4.5.2 that the attack can remove the obstacle in the target area without knowing the victim's position

in advance. This could result in severe impacts in safety-critical applications such as autonomous vehicles. We believe this work will open new directions for understanding receptive field and designing robust model architecture from the robustness perspective.

4.2 Preliminaries

This section describes preliminaries such as motivation, threat model, and experiment setups.

4.2.1 Motivation and Threat Model

Existing methods that generate adversarial examples typically require perturbing the targeted object [18, 157, 60]. Modern models that incorporate contextual information with a large receptive field open the possibility for a new attack vector, where the prediction label of the target object could still be misled without directly accessing and adding the perturbation on it. To understand the vulnerability of the model introduced by the large receptive field, we propose and conduct the RAP attack. We choose the semantic segmentation task because the pixel level predictions allow us to better analyze and understand the vulnerability status, and the semantic segmentation task is a fundamental task that has been widely used in real-world applications. To study it, we consider the strongest whitebox scenarios to explore what a powerful adversary can do based on the Kerckhoffs's principle [112] to better motivate defense methods. This means that the adversary has access to both the model structure and weights so that the adversary can compute both outputs and gradients with the model. While existing research has shown that it is possible to construct adversarial examples without white-box level access [73], we leave it as future work. With the goal to demonstrate the consequences of RAP attacks in the physical world, we choose the adversarial patch [18] threat model, a localized perturbation, which has shown to

be more effective in the physical world [35]. More specifically, we consider self-driving scenarios where the adversarial patch is printed on the road since such application is safety-critical and can lead to severe consequences once being successfully attacked.

4.2.2 Models

To measure the vulnerability of RAP on different model designs, we focus on model designs that are used for enlarging the receptive field: dilated convolution, multi-scale (pyramid pooling), and self-attention. We apply Dilated residual networks (DRN) [179, 181] for most evaluations since it is a classic ResNet [68] based architecture and many previous works on adversarial robustness used it for evaluations [157, 165]. We use HDC-DUC [147] for analyzing the adversarial robustness of models with different receptive field sizes due to its adjustable dilation number configurations. We also select DeepLabV3 [33], PSPNet [191] and PSANet [192] as top-ranking models on semantic segmentation benchmark with different model designs. All the models are trained with a backbone of the 50-layer ResNet backbone for consistency. Additionally, we try our method on the recent transformer-based model, SegFormer [168] as well. We divide these models into the following categories:

Dilated convolution-based models. Dilated (astrous) convolution is a type of convolution that integrates spacing between kernel parameters and increases the field of view for kernels. Most modern CNN-based models utilize dilated convolution to increase the receptive field size [179, 181]. All the selected models except SegFormer utilize this technique.

Multi-scale based models. To extract features at different scales, several semantic segmentation architectures perform Spatial Pyramid Pooling [67, 192] while Deeplab [33, 32] applies Atrous Spatial Pyramid Pooling (ASPP) where three atrous convolutions with large atrous rates (6, 12 and 18) process the dilated CNN (DCNN) output. We select PSPNet and Deeplabv3 to represent this category.

Attention-based models. Attention mechanisms have been persistently explored in computer vision over the years. It has been shown to enable the model to assess the importance of features at different positions and scales. We select PSANet [192] and SegFormer [168] to represent this category, where PSANet applies self-attention in addition to a CNN-based backbone and SegFormer is based on a transformer backbone that consists of a multi-layer attention mechanism.

4.2.3 Dataset

For attacks in the digital space, we apply Cityscapes [45] in our evaluations. We show results on the validation set of the dataset, which contains 500 high-resolution images with a combined 19 categories of segmentation labels. Cityscapes is an outdoor dataset containing instance-level annotations which are designed to be focused on understanding urban street scenes. Since this dataset is closely related to safety-critical applications such as autonomous driving, it would raise real-world safety concerns if being attacked. Compared with other datasets such as Pascal VOC [58] and CamVid [21], Cityscapes is more challenging and realistic due to the relatively high resolution and diverse scenes.

4.3 Remote Adversarial Patch

In this section, we will first introduce RAP in the digital space. We then show a special case of RAP attack, object object removing attack, which misleads the model prediction of a foreground object to a target class. In the end, we demonstrate how this vulnerability can be exploited in real-world applications.

4.3.1 Notations

Let h_θ denote the segmentation model parameterized by θ; x denotes the input image to h_θ with a corresponding ground truth target of y; and $\mathcal{L}$ denote an objective

function used for train the adversarial patch δ. In the following sections, we will define $\mathcal{L}$ by combining different losses to achieve the corresponding attack goals.

4.3.2 Generating RAP

The goal of the attack is to generate a localized adversarial patch δ with target prediction y' which can be used to fool the prediction result of a given non-overlap target area m_t. We can set the loss function for the RAP attack as the objectness loss $\mathcal{L}_{obj}$:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{obj}(h_\theta(x), y', m_t) \\ &:= \sum_{i,j \in m_t} \mathbf{CE}(h_\theta(x)_{i,j}, y'_{i,j})\end{aligned}$$

where **CE** is the cross entropy loss function. $h_\theta(x)_{i,j}$ and $y'_{i,j}$ represent model prediction results and target prediction of coordinate (i,j), respectively.

We use Projected Gradient Descent (PGD) attack [98] to solve the above objective function.

The update for the iteration k is formulated as:

$$\delta_k := \text{clip}_{[0,1]}(\delta_{k-1} + \gamma \cdot \text{sign}(\nabla_{\delta_{k-1}} \mathcal{L}(h_\theta(A(\delta, x, t)), y', m_t)) \tag{4.1}$$

where γ is the step size for each iteration; $A(\delta, x, t)$ aims to apply the patch δ to the image x with a transformation t.

4.3.3 A Special Case: Object Removing Attack

To demonstrate the real-world impact of the RAP attack, we consider the safety-critical autonomous driving scenarios and the goal of the *object removing attack* is to remove an entire object from the segmentation prediction with the RAP. In other words, it aims to change the model prediction of the target object to the target

class. To make the RAP more realist, we target at making the RAP look like a graffiti printed on the ground. We define t_{init} as the initial transformation that transforms the patch to look like printed on the ground. We define δ_{init} as the initial perturbation that makes the patch look like a graffiti. Then, by bounding the perturbation $|\delta - \delta_{init}| < \epsilon$ during the optimization process, the generated patch looks like graffiti and is less suspicious. We also use the total variation (Tv) loss $\mathcal{L}_{tv}$. $\mathcal{L}_{tv}$ for ensuring the smooth color transitions of the generated adversarial patches. Tv loss is formulated to represent the similarity of the neighboring pixels. We use $\mathcal{L}_{tv}$ as a regularization loss. To summarize, we modify the Equation 4.1 as:

$$
\begin{aligned}
ub &= \min(1, \delta_{init} + \epsilon); \quad lb = \max(0, \delta_{init} - \epsilon) \\
\mathcal{L} &= \mathcal{L}_{obj}(A(\delta, x, t_{init})), y', m_t) + \alpha \cdot \mathcal{L}_{tv}(\delta) \\
\delta &:= \text{clip}_{[lb,ub]}(\delta + \gamma \cdot \text{sign}(\nabla_\delta \mathcal{L}(h_\theta(A(\delta, x, t_{init})), y', m_t))
\end{aligned} \tag{4.2}
$$

where ub and lb are the upper bound and lower bound of the patch δ respectively; α is the scaling factor determined empirically.

Extending to the physical setting In order to further extend the object removing attack to the real world, there are two additional challenges to overcome: 1) unknown positions of the patch and the target requiring the patch to be robust to spatial variations, and 2) color distortions from the camera and printers requiring the patch to be robust to color space variations.

To address them, as illustrated in Figure 4.2,

Expectation over Transformation (EOT) [18, 60] technique and the non-printability score (NPS) L_{nps} [131] is used. To summarize, we update the optimization formula-

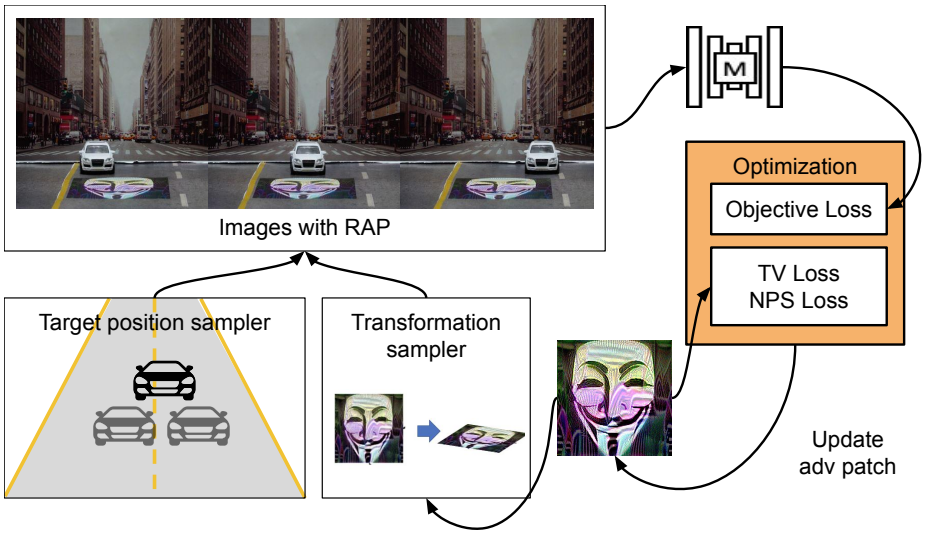


Figure 4.2: Methodology overview of the physical object removing attack.

tion in Equation 4.2 as:

$$\begin{aligned}\mathcal{L} =& \mathcal{L}_{obj}(A(\delta, x, t)), y', m_t) + \alpha \cdot \mathcal{L}_{tv}(\delta) + \beta \cdot \mathcal{L}_{nps}(\delta, C) \\ \delta :=& \text{clip}_{[lb,ub]}(\delta + \gamma \cdot \text{sign}(\nabla_\delta \mathcal{L}(h_\theta(A(\delta, x, t)), y', m_t, C))\end{aligned} \tag{4.3}$$

where α and β are the scaling factors determined empirically for a sample $x \sim \mathcal{X}$ and a sample transformation $t \sim \mathcal{T}$.

4.4 RAP Vulnerability Analysis

This section conducts a comprehensive analysis of RAP attacks on various learning-based segmentation methods including dilated convolution method, the multi-scale method, and the attention-based method. Starting with fundamental dilated convolution, we change the model's receptive field size and the distance between RAP and target objects to understand how and why receptive field size affects the vulnerability level. Additionally, we evaluate our attack on various models with different designs to understand which designs are more vulnerable to our attack.

Table 4.1: Performance of different variations of the HDC-DUC module. "RF increased" indicates the total size of receptive field increase along a single dimension compared to the layer before the dilation operation.

Network	RF increased	mIOU
HDC-DUC-no	54	62.86
HDC-DUC-rf	116	69.51
HDC-DUC-bigger	256	70.44

4.4.1 Micro-benchmark on Receptive Field Size

To show the vulnerability introduced by large receptive fields, we conduct two experiments: varying the receptive field size by changing the model architectures and 2) varying the distance between the target area and attack area by changing the patch position. To demonstrate the generality of the causality, we focus on the fundamental dilated convolution-based models in these experiments.

Experiment setup. In this experiment, we apply the RAP attack mentioned in §4.3.2. We select a target area (400 pixels x 400 pixels) with the center at the center of each image (1024 pixels x 2048 pixels) from the Cityscapes validation dataset. We add a non-overlapping adversarial patch (50 pixels x 400 pixels) below the target area with different distances. Here we select this patch shape instead of a square shape patch to facilitate a better demonstration of how the attack effects are different at different distances. In the evaluation results, only IoU (Intersection over Union) in the target area is reported to better demonstrate the consequences of the attack.

Receptive size. In this experiment, we evaluate the RAP attack on the same architecture with different receptive field sizes. Most models are customized with fixed receptive field sizes. Blindly changing the configurations to increase receptive sizes will usually degrade the performance. For example, blindly increasing the dilation factors on DRN results in gridding effect and dropping performance. Therefore, we choose

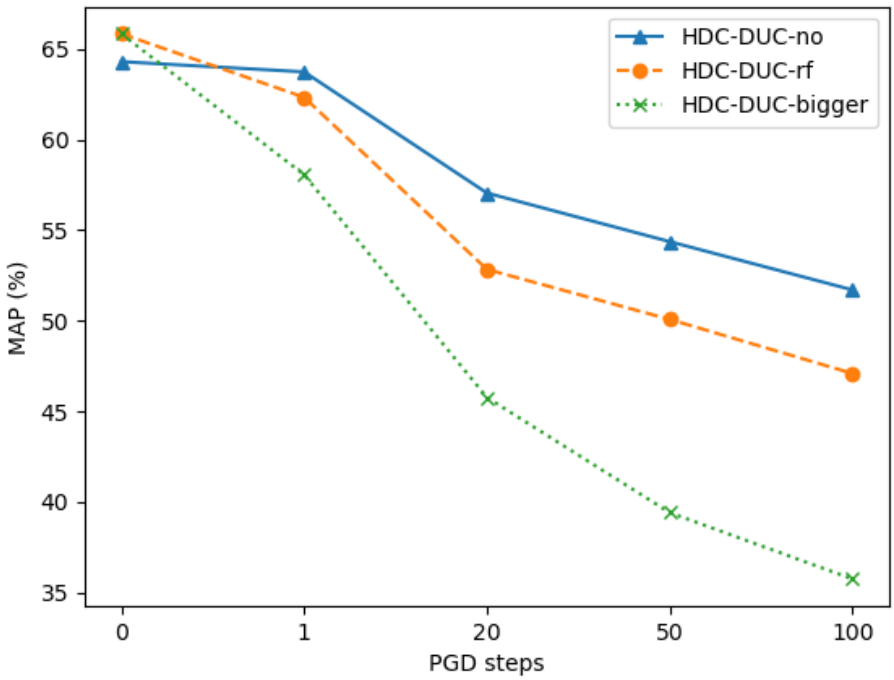


Figure 4.3: Results of the RAP attack on models with different receptive field sizes. Receptive field size: HDC-DUC-bigger >HDC-DUC-rf >HDC-DUC-no.

DUC-HDC as the model architecture for conducting the evaluation. In the original paper for DUC-HDC, three different dilated configurations (Dilation-no, Dilation-rf, Dilation-bigger [147]) are provided for different receptive field sizes without intriguing the gridding effect. Here, we denote these three models as HDC-DUC-no, HDC-DUC-rf, and HDC-DUC-bigger to distinguish it from the other model architectures. The increase of RF size and benign mIoU of three models are shown In Table 4.1. Notice that HDC-DUC-no performed worse while the other two models perform similarly.

We set the distance between RAP and the target area as and then evaluate three models on various PGD steps to demonstrate the robustness of models with different receptive sizes with respect to the RAP attacks. Results in Figure 4.3 show that models with larger receptive fields tend to be more vulnerable to the RAP attack. Notice that, while HDC-DUC-no performs worse at the benign condition (steps=0), it is more robust than the other two models with larger receptive field sizes.

Table 4.2: IoUs of different categories when the attack patch is at different distance (Δ pixels) to the target area on DRN-D-50 model. Notice that the IoU of class *pole* and *person* is higher than benign at distance. This is due to the inaccurate prediction of the benign model and weaker attack capability when the RAP is at distance.

	mIoU	pole	traffic light	traffic sign	person	rider
Δ=0	27.1	38.0	32.2	48.3	36.3	11.7
Δ=50	39.8	44.5	45.1	45.3	52.0	21.7
Δ=100	49.8	49.2	44.6	52.6	69.5	31.5
Δ=200	55.1	51.2	48.8	58.2	82.7	33.9
Benign	65.1	49.1	52.0	62.7	80.4	54.8
	car	truck	bus	train	motor	bicycle
Δ=0	52.2	20.5	23.1	8.4	17.8	9.3
Δ=50	66.2	37.7	34.4	31.3	34.5	24.9
Δ=100	81.2	43.5	46.8	40.1	43.1	45.2
Δ=200	88.6	47.7	49.1	42.3	46.3	57.7
Benign	91.4	59.0	80.8	66.0	51.5	68.8

Distance between the RAP and the target area. In this experiment, we define the distances between RAP and target area as (Δ). We evaluate the RAP attack with different δ ranging from 0 pixels to 200 pixels on the DRN-D-50 model. We use 100 step PGD as the optimization for RAP. In Table 4.2, we show the IoUs of 11 categories in the Cityscapes dataset [1]. It demonstrates that the attack impacts become weaker when the distance Δ is larger. For the extreme case of 200 pixels distance, where the RAP is near the edge of the receptive field (483), the predictions of the model are almost unaffected.

To conclude, models with a larger receptive field expose larger attack surfaces and enable the RAP attack with higher impacts in terms of the RAP attack range and success rates of perturbing the model predictions. But we also notice that RAP is less effective when it is away from the target area. This is naturally due to the convolution-based architecture applying a Gaussian mask on the receptive field where

[1]Here, we do not consider the background category since the patch is usually overlapping with at least one category of background.

pixels at edge positions contribute less to the final prediction.

4.4.2 RAP Analysis on Representative Architectures

Unlike dilated convolution-based models that need to balance the dilation configurations to have good performance and large receptive field, other techniques like multi-scale and attention usually extend the receptive field size to the whole image. In other words, while receptive field size is meaningful for describing the vulnerability levels of dilated convolution-based models, it is less useful for other models. Therefore, we select representative models with similar benign performance and analyze the robustness of the models empirically by conducting RAP attacks on them.

Table 4.3: mIoU of models under benign settings and under RAP attacks.

Model	DRN	Deeplabv3	PSPNet	PSANet	SegFormer
Benign	72.34	79.6	77.85	77.58	78.51
RAP-100	60.23	54.68	39.04	43.33	28.04

Experiment setup and results. We evaluate the aforementioned models with untargeted RAP attacks on the Cityscapes validation dataset. To include the long-range effect of models, we place the patch (300 pixels x 300 pixels) at the corner of the image. Since no realistic constraints apply here, we generate the patch without transformation and Tv loss in this experiment for fully demonstrating the attack effects. Table 4.3 demonstrates the evaluation results.

We can see that DRN is the most robust one even given its worst benign performance. Additional model designs, such as ASPP in Deeplabv3, PPM in PSPNet and global attention in PSANet, hurt model robustness at different levels since they increase the receptive field. Among the multi-scale methods, our results indicate that PPM (in PSPNet) is more vulnerable than ASPP (in Deeplabv3). Zheng et al. [193] demonstrated that a model with more information loss tends to be more vulnerable

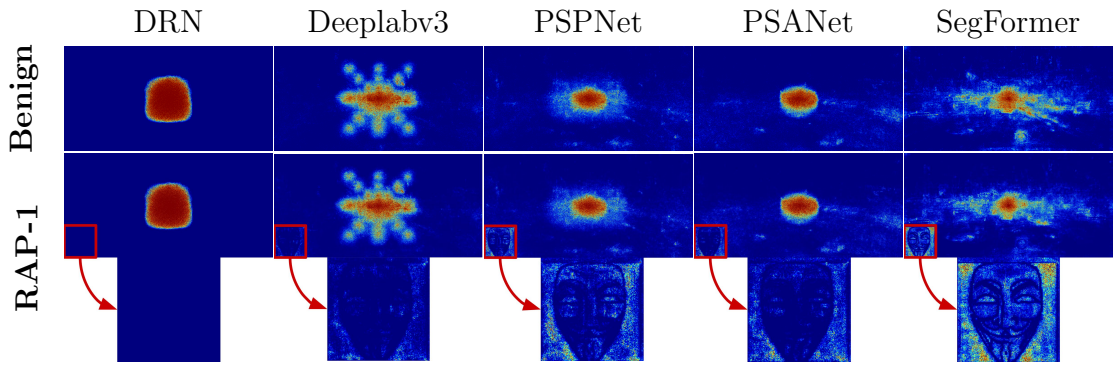


Figure 4.4: Effective receptive field (ERF) on Cityscapes (average over 500 images). Top row: ERF on benign images. Bottom row: ERF on adv images with RAP at the bottom left corner (zoomed in below). The patches are optimized with a one-step PGD attack.

Table 4.4: IoUs of various models 1) under benign condition; 2) RAP-init: with inital patch; 3) RAP-50: with 50 steps optimized RAP.

Model/attack	mIoU	pole	traffic light	traffic sign	person	rider	car	truck	bus	train	motor	bicycle
DRN	65.19	49.16	52.09	62.78	80.46	54.87	91.468	59.010	80.826	66.054	51.546	68.838
DRN w/ RAP-init	64.45	49.10	52.09	62.78	80.10	54.70	90.44	58.50	80.29	61.11	51.47	68.34
DRN w/ RAP-50	**39.01**	**34.18**	**29.68**	**51.60**	**44.31**	**41.41**	**55.92**	**25.41**	**64.09**	**37.03**	**13.63**	**31.85**
DeepLabv3	71.45	49.03	51.12	66.19	81.89	62.46	92.68	72.54	90.33	86.16	66.10	70.22
DeepLabv3 w/ RAP-init	67.16	48.03	50.91	64.83	75.79	59.34	63.58	68.04	90.28	85.80	64.81	62.74
DeepLabv3 w/ RAP-50	**25.64**	**2.63**	**37.69**	**48.84**	**26.64**	**29.94**	**32.46**	**22.39**	**26.59**	**15.63**	**17.04**	**22.15**
PSPNet	70.47	50.46	52.60	67.62	81.41	60.69	92.57	71.18	87.86	83.02	64.69	69.66
PSPNet w/ RAP-init	67.08	49.68	44.74	66.13	74.07	59.10	86.55	66.88	87.57	82.32	63.26	60.90
PSPNet w/ RAP-50	**2.61**	**2.99**	**2.86**	**3.33**	**3.43**	**2.69**	**2.22**	**2.02**	**2.65**	**2.23**	**1.68**	**2.55**
PSANet	70.02	62.81	69.86	77.53	80.11	58.03	91.47	59.01	80.83	66.05	51.55	68.84
PSANet w/ RAP-init	66.07	47.77	48.71	65.42	80.23	58.24	91.86	59.72	81.62	67.11	56.84	69.26
PSANet w/ RAP-50	**0.67**	**0.88**	**1.21**	**1.62**	**1.29**	**0.93**	**0.56**	**0.07**	**0.19**	**0.51**	**0.00**	**0.07**
SegFormer	69.51	51.70	51.16	65.59	80.48	55.84	92.06	67.41	87.16	83.19	60.02	69.98
SegFormer w/ RAP-init	64.54	50.31	49.91	64.37	75.15	50.61	85.63	61.96	83.11	73.48	53.94	61.49
SegFormer w/ RAP-50	**0.83**	**1.12**	**3.37**	**1.50**	**1.42**	**0.46**	**0.64**	**0.**	**0.**	**0.**	**0.**	**0.68**

to adversarial attacks. This aligns with our findings, since ASPP uses atrous convolution instead of pooling to reduce information loss. Among different backbones, our results indicate that transformer-based models are more vulnerable than CNN-based models. Overall, the attention-based methods are more vulnerable to RAP attacks. This is reasonable since the global attention allows long-range interference which naturally enables RAP attacks. This finding is alarming since attention-based methods are persistently explored in computer vision over the years [63, 184, 183, 89, 72, 192].

Effective receptive field analysis. Here, to further understand the results, we use effective field receptive field (ERF) [97] as a toolkit to visualize and interpret

why different model designs are affected by the RAP vulnerability at different levels. In Figure 4.4, we visualize ERFs of the four models we evaluate upon under benign and adversarial settings. For the adversarial setting, we conduct a one-step RAP untargeted attack on the pixel at the center of the image. We conduct the attack with an initial patch to make the visualization more obvious. The ERFs under adversarial settings visualize how much the RAP is contributing to the prediction of the target pixel at the center of the image among different models. The more obvious such patterns are it naturally means the model is more vulnerable to RAP. The results also align with the experiment results in Table 4.3.

4.5 Object Removing Attack

In this section, we evaluate the object removing attack in both digital settings and physical settings to demonstrate the severe impacts of the proposed RAP attacks.

4.5.1 Digital Experiments

The goal of the object removing attack is to generate a RAP that removes prediction of target obstacles in the target area (or misleads the prediction of target obstacles to target label.). In order to achieve this goal, we conduct targeted attack with target label *building* or *road* using the methodology mentioned in §4.3.3. We evaluate the method by generating the RAP for each image in the Cityscapes validation dataset with 50 PGD steps with manually selected initial transformation t_{init}. For the target mask m_t, we select the area in the target area which is labeled as an obstacle. We evaluate with the IoU of each class on the selected models. For the hyperparameters, we empirically select *eps* as 100, step size γ as 10, and α as 0.2 for the best results. We select three initial patch images including a grey-scale image, a color image, and a road patch to demonstrate the generality of the initial patch choices.

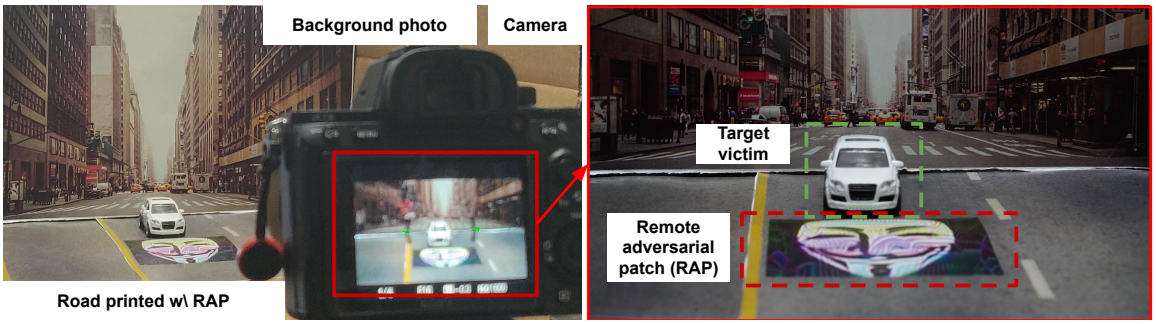


Figure 4.5: Illustration for the miniature scene set up in the physical experiment.

In Table 4.4, we demonstrate that the generated RAP greatly reduced the IoU of each class. DRN and Deeplabv3 are more robust to the object removing attacks which aligns with the previous results in §4.4.1. We notice that PSPNet is more robust compared to the PSANet here, which seems to be different from the previous results. However, we found most remaining correct predictions can be removed if more PGD attack steps apply. This means that the PSPNet takes more iterations to reach the full attack capability. More visualized examples could be found in the supplementary material.

4.5.2 Physical Experiments

In this experiment, the goal is to generate a physical RAP that removes target obstacles regardless of the position. Due to the high cost and safety concerns of testing with a real car on the road, we build a miniature scene with a toy car and printed road and background illustrated in Figure 4.5. We use a DSLR camera with an 85mm lens at F16 to capture the image to avoid the blurring effect. The captured image is compressed to 376 pixels x 672 pixels because it is difficult to reproduce the entire scene like in the Cityscapes dataset. However, the compressed image is at a similar scale to the cropped image from the Cityscapes dataset. Since semantic segmentation models are not sensitive to the input image sizes, we consider this a reasonable setup. We evaluate the attack on the DRN model since it is the most robust model according

to the digital experiment results. For the EOT random sampling mentioned in §4.3.3,

Figure 4.6: Examples of the generated RAP and printed RAP in the physical experiment.

we take photos of the toy car at seven different positions shown in Figure 4.5, where we use four of them for the training procedure and three for the validation. We sampling the perspective transformation and translation transformation by adding noises to the target patch coordinates. We empirically use 20 pixels for the perspective transformation and 50 pixels for the translation transformation. We choose α and β empirically with 0.8 and 0.9 for the best results. For each patch generation, we conduct a total of 100 steps, where each step consists of 100 iterations mentioned in Equation 4.3. We show the generated RAP and the photo of printed RAP on the road in Figure 4.6. We use a Hewlett-Packard Color LaserJet M855 printer to print the patch. We evaluate the attack results by placing the car at 3 different positions. In Figure 4.7, we demonstrate that the printed RAP is able to remove the target objects entirely except for one position where the toy car is further away on the side. This can be due to the limited receptive field size and also validate the results mentioned earlier in §4.4.1.

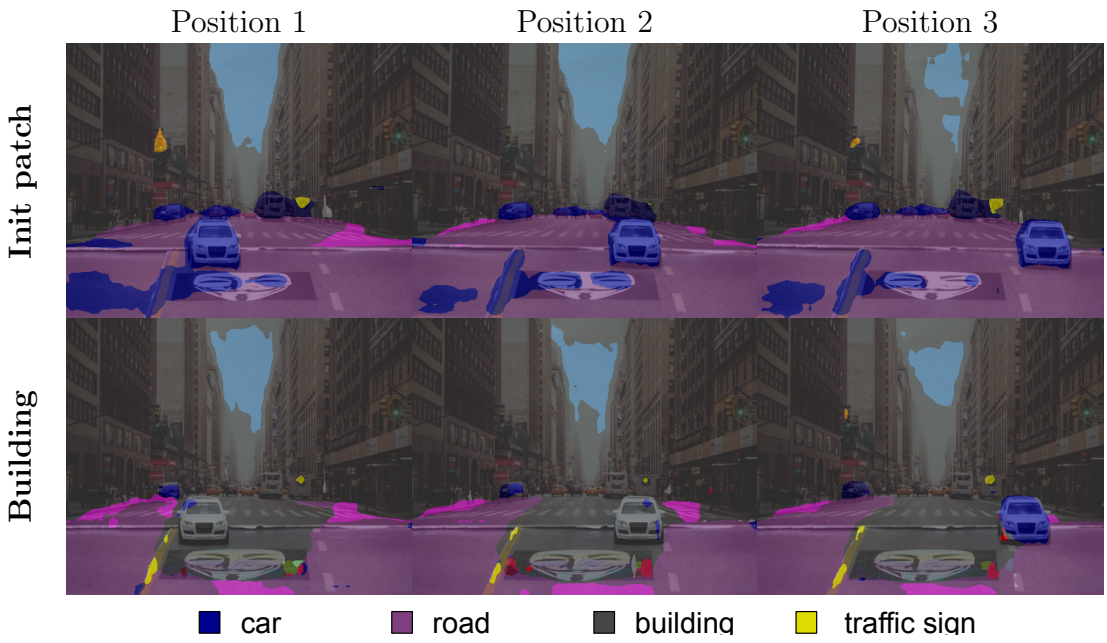


Figure 4.7: Examples of the physical attack experiment where the toy car is places at different positions in the miniature scene. The toy car is detected correctly with the initial patch (top) and detected as the target label "building" with the RAP (below).

4.6 Related Works

Receptive field Receptive fields are defined portions of space or spatial construct containing units that provide input to a set of units within a corresponding layer [97]. The receptive field is originally defined by the filter size of a layer within a convolution neural network. Since the receptive field is used as an indication of the extent of the scope of input data a neuron or unit within a layer can be exposed to, the effective receptive field can also be extended to other architectures beyond CNN.

Semantic segmentation Semantic Segmentation has received long lasting attention in the computer vision community. Recent advances in deep learning also show that deep convolutional networks can achieve much better results than traditional methods. Yu et al. [181] proposed using dilated convolutions to build high resolution feature maps for semantic segmentation. They can improve the performance

significantly compared to upsampling approaches. Most of the recent state-of-the-art approaches are based on dilated convolutions and residual networks. Therefore, in this work, we choose dilated residual networks (DRN) as our target models for evaluating the proposed attack.

Adversarial example and Adversarial Patch Given a machine learning model M, input x and its corresponding label y, an adversarial attacker aims to generate adversarial examples x' so that $M(x') \neq y$ (untargeted attack) or $M(x') = y'$, where y' is a target label (targeted attack). [28] proposed to generate an adversarial perturbation for a targeted attack by optimizing an objective function as follows:

$$\min \left(||x - x'||_p + \lambda \cdot Loss(M(x'), y') \right) \quad \text{s.t.} \quad x' \in X,$$

where $M(x') = y'$ is the target adversarial goal and $x' \in X$ denote that the adversarial examples should be in a valid set. Further, optimization-based algorithms have been leveraged to generate adversarial examples on various kinds of machine learning tasks successfully, such as segmentation [165, 41], human pose estimation [41], object detection [165], Visual Question Answer system [171], image caption translation [36], etc.

Adversarial patches were first proposed by Brown et al. [22] which are physically realizable localized adversarial examples. Though the adversarial patch attack is originally proposed on image classification task, it is further extended to object detectors and other tasks [87, 94]. Unlike the remote adversarial patch proposed in this work, most of the previous work require adversarial patch to be overlapped with the target. Though Liu et al. [94] demonstrated the attack without overlapping the patch and the target, they were exploiting the region proposal module in the object detectors, which is not generally used among all the networks.

Non-overlapping adversarial patch We are aware of a few previous works have demonstrated non-overlapping adversarial patches [54, 57, 56, 55]. However, they focus on misleading object detectors. Object detectors often incorporate task-specific techniques for predicting bounding boxes (e.g. region proposal, NMS, etc.), making it challenging to understand the causality of the attacks and provide a guideline for future model architecture designs towards a more robust model. For example, as mentioned above, DPatch exploited the region proposals to saturate the proposals for removing the detection of other objects other than reducing the objectness of them. Therefore, in order to understand the vulnerability status and cause of it on general vision tasks, also as the main goal of this paper, we chose semantic segmentation models for conducting the analysis.

4.7 Discussion and Future Work

There are numerous designs proposed in the semantic segmentation task over the past few years. Besides the methods evaluated in this paper, categories like recurrent neural network based models [145], generative models [96] and other models [183, 184, 178] are not fully evaluated in this work. Also, though we empirically demonstrated in §4.4.1, the reason that attention-based models are more vulnerable to the proposed RAP attack can be due to the global attention introduced in the network, how the global attention design introduces new vulnerabilities is not thoroughly analyzed in this work. However, unlike dilated convolution, attention-based methods increased the receptive field size by introducing more parameters. There is possibility to increase the robustness of the model through these parameters where it is more difficult when only dilated convolution is used for the purpose. We leave these as future directions to this work.

There is considerable variation in the physical world that the attacker will have to deal with. Since the designed RAP attack required the patch to be printed on the

road, besides different view angles and target positions, the printed patch could be partially blocked by the target. Also, the camera models can also affect the attack results.

Defending against these adversarial examples has proven difficult. Many defenses fall prey to the so-called "gradient masking" or "gradient obfuscating" problem [19]. Some defenses against adversarial patches are also proposed recently [37], leveraging the feature of localized perturbation in the adversarial patch attack. However, most defenses are focused on the image classification task and usually introduce a large overhead which is less applicable in the real world applications.

4.8 Conclusion

In this work, we show that segmentation models employing contextual information with a large receptive field increase the attack surface and enable the proposed RAP attack. Since the RAP attack does not require physical access to the target victim, it is easier for the attacker to conduct in the real world and has a larger impact once the attacker succeeds. This can cause major issues when deep models are deployed in real-world applications like self-driving cars. We validate such vulnerabilities on various architectures with different designs. In addition, we conduct a comprehensive analysis of the vulnerability status among different architectures. To further demonstrate the feasibility of the attack, we conduct the RAP attack in physical experiments with a miniature scene. We believe this work highlights the need for studying defense algorithms that are robust to the RAP attacks but also accurate.

CHAPTER V

Secure and Safe Autonomous Driving with Modular Robustness

5.1 Introduction

Apart from the perception system, trajectory forecasting is an integral part of modern autonomous vehicle (AV) systems. It allows an AV system to anticipate the future behaviors of other nearby road users and plan its actions accordingly. Recent data-driven methods have shown remarkable performances on motion forecasting benchmarks [17, 75, 126, 185, 122, 123, 82]. While many prior works have study the robustness of perception systems, few existing work have considered the robustness of these trajectory prediction models, especially when they are subject to deliberate adversarial attacks. In this chapter, we will first describe a framework for generating efficient and effective adversarial examples called adversarial dynamic optimization (AdvDO). Then, we will introduce an adversarial training framework, RobustTraj, for improving trajectory prediction model robustness.

5.2 Adversarial Attacks on Trajectory Prediction

A typical adversarial attack framework consists of a threat model, i.e., a function that generates "realistic" adversarial samples, adversarial optimization objectives,

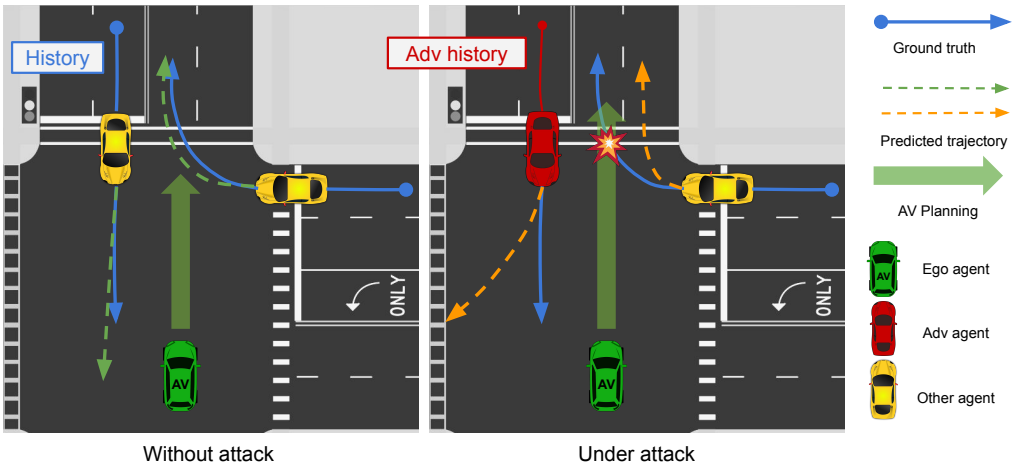


Figure 5.1: An example of attack scenarios on trajectory prediction. By driving along the crafted adversarial history trajectory, the adverial agent misleads the prediction of the AV systems for both itself and the other agent. As a consequence, the AV planning based on the wrong prediction results in a collision.

and ways to systematically determine the influence of the attacks. However, a few key technical challenges remain in devising such a framework for attacking trajectory prediction models.

First, the threat model must synthesize adversarial trajectory samples that are 1) feasible subject to the physical constraints of the real vehicle (i.e. dynamically feasible), and 2) close to the nominal trajectories. The latter is especially important as a large alteration to the trajectory history conflates whether the change in future predictions is due to the vunerability of the prediction model or more fundamental changes to the meaning of the history. To this front, we propose an attack method that uses a carefully designed *differentiable dynamic model* to generate adversarial trajectories that are both effective and realistic. Furthermore, through a gradient-based optimization process, we can generate adversarial trajectories efficiently and customize the adversarial optimization objectives to create different safety-critical scenarios.

Second, not all trajectory prediction models react to attacks the same way. Fea-

tures that are beneficial in benign settings may make a model more vulnerable to adversarial attacks. We consider two essential properties of modern prediction models: (1) motion property, which captures the influence of past agent states over future states; and (2) social property, which captures how the state of each agent affects others. Existing prediction models have proposed various architectures to explicitly models these properties either in silo [126] or jointly [185]. Specifically, we design an attack framework that accounts for the above properties. We show that our novel attack framework can exploit these design choices. As illustrated in Figure 5.1, by only manipulating the history trajectory of the adversarial agent, we are able to mislead the predicted future trajectory for the adversarial agent (i.e. incorrect prediction for left turning future trajectory of red car in Figure 5.1-right). Furthermore, we are able to mislead the prediction for *other* agent's behavior (i.e. turning right to turning left for the yellow car in Figure 5.1-right). During the evaluation, we could evaluate these two goals respectively. It helps us fine-grained diagnose vulnerability of different models.

Finally, existing prediction metrics such as average distance error (ADE) and final distance error (FDE) only measure errors of average cases and are thus too coarse for evaluating the effectiveness of adversarial attacks. They also ignore the influence of prediction errors in downstream planning and control pipelines in an AV stack. To this end, we incorporate various metrics with semantic meanings such as *off-road rates*, *miss rates* and *planning-aware metrics* [76] to systematically quantify the effectiveness of the attacks on prediction. We also conduct end-to-end attack on a prediction-planning pipeline by simulating the driving behavior of an AV in a close-loop manner. We demonstrate that the proposed attack can lead to both emergency brake and various of collisions of the AV.

We benchmark the adversarial robustness of state-of-the-art trajectory prediction models [185, 126] on the nuScenes dataset [23]. We show that our attack can increase

prediction error by 50% and 37% on general metrics and planning-aware metrics, respectively. We also show that adversarial trajectories are realistic both quantitatively and qualitatively. Furthermore, we demonstrate that the proposed attack can lead to severe consequences in simulation. Finally, we explore the mitigation methods with adversarial training using the proposed adversarial dynamic optimization method (AdvDO). We find that the model trained with the dynamic optimization increase the adversarial robustness by 54%.

5.3 Related works

Trajectory Prediction. Modern trajectory prediction models are usually deep neural networks that take state histories of agents as input and generate their plausible future trajectories. Accurately forecasting multiagent behaviors requires modeling two key properties: (1) motion property, which captures the influence of past agent states over future states; (2) social property, which captures how the state of each agent affects others. Most prior works model the two properties separately [75, 126, 51, 138, 82]. For example, a representative method Trajactron++ [126] summarizes temporal and inter-agent features using a time-sequence model and a graph network, respectively. But modeling these two properties in silo ignores dependencies across time and agents. A recent work Agentformer [185] introduced a joint model that allows an agent's state at one time to directly affect another agent's state at a future time via a transformer model.

At the same time, although these design choices for modeling motion and social properties may be beneficial in benign cases, they might affect a model's performance in unexpected ways when under adversarial attacks. Hence we select these two representative models [126, 185] for empirical evaluation.

Adversarial Traffic Scenarios Generation. Adversarial traffic scenario generation is to synthesize traffic scenarios that could potentially pose safety risks[52, 81,

53, 16, 117]. Most prior approaches fall into two categories. The first aims to capture traffic scenarios distributions from real driving logs using generative models and sample adversarial cases from the distribution. For example, STRIVE [117] learns a latent generative model of traffic scenarios and then searches for latent codes that map to risky cases, such as imminent collisions. However, these latent codes may not correspond to real traffic scenarios. As shown in the supplementary materials, the method generates scenarios that are unlikely in the real world (e.g. driving on the wrong side of the road). Note that this is a fundamental limitation of generative methods, because almost all existing datasets only include safe scenarios, and it is hard to generate cases that are rare or non-existent in the data.

Our method falls into the second category, which is to generate adversarial cases by perturbing real traffic scenarios. The challenge is to design a suitable threat model such that the altered scenarios remain realistic. AdvSim [146] plants adversarial agents that are optimized to jeopardize the ego vehicles by causing collisions, uncomfortable driving, etc. Although AdvSim enforces the dynamic feasibility of the synthesized trajectories, it uses black-box optimization which is slow and unreliable. Our work is most similar to a very recent work [189]. However, as we will show empirically, [189] fails to generate dynamically feasible adversarial trajectories. This is because its threat model simply uses dataset statistics (e.g. speed, acceleration, heading, etc.) as the dynamic parameters, which are too coarse to be used for generating realistic trajectories. For example, the maximum acceleration in the NuScenes dataset is over $20m/s^2$ where the maximum acceleration for a top-tier sports car is only around $10m/s^2$. In contrast, our method leverages a carefully-designed differentiable dynamic model to estimate *trajectory-wise* dynamic parameters. This allows our threat model to synthesize realistic and dynamically-feasible adversarial trajectories.

Adversarial Robustness. Deep learning models are shown to be generally vulner-

able to adversarial attacks [30, 47, 28, 158, 174, 165, 70, 71, 156, 149, 66, 162]. There is a large body of literature on improving their adversarial robustness [105, 127, 99, 176, 170, 20, 109, 102, 186, 188, 98, 64, 153, 129]. In the AV context, many works examine on the adversarial robustness of the perception task [164], while analyzing the adversarial robustness of trajectory forecaster [189] is rarely explored. In this work, we focus on studying the adversarial robustness in the trajectory prediction task by considering its unique properties including motion and social interaction.

5.4 Problem Formulation and Challenges

In this section, we introduce the trajectory prediction task and then describe the threat model and assumptions for the attack and challenges.

Trajectory Prediction Formulation. In this work, we focus on the trajectory prediction task. The goal is to model the future trajectory distribution of N agents conditioned on their history states and other environment context such as maps. More specifically, a trajectory prediction model takes a sequence of observed state for each agent at a fixed time interval Δt, and outputs the predicted future trajectory for each agent. For observed time steps $t \leq 0$, we denote states of N agents at time step t as $\mathbf{X}^t = (x_1^t, \ldots, x_i^t, \ldots, x_N^t)$, where x_i^t is the state of agent i at time step t, which includes the position and the context information. We denote the history of all agents over H *observed* time steps as $\mathbf{X} = \left(\mathbf{X}^{-H+1}, \ldots, \mathbf{X}^0\right)$. Similarly, we denote future trajectories of all N agents over T *future* time steps as $\mathbf{Y} = \left(\mathbf{Y}^1, \ldots, \mathbf{Y}^T\right)$, where $\mathbf{Y}^t = (y_1^t, \ldots, y_N^t)$ denotes the states of N agents at a future time step t ($t > 0$). We denote the ground truth and the predicted future trajectories as $\mathbf{Y}$ and $\hat{\mathbf{Y}}$, respectively. A trajectory prediction model $\mathcal{P}$ aims to minimize the difference between $\hat{\mathbf{Y}} = \mathcal{P}(\mathbf{X})$ and $\mathbf{Y}$. In an AV stack, trajectory prediction is executed repeatedly at a fixed time interval, usually the same as Δt. We denote L_p as the number of trajectory prediction being executed in several past consecutive time frames. Therefore, the histories at time

frame ($-L_p < t \leq 0$) are $\mathbf{X}(t) = \left(\mathbf{X}^{-H-t+1}, \ldots, \mathbf{X}^{-t}\right)$, and similarly for $\mathbf{Y}$ and $\hat{\mathbf{Y}}$.

Adversarial Attack Formulation. In this work, we focus on the setting where an adversary vehicle (adv agent) attacks the prediction module of an ego vehicle by driving along an adversarial trajectory $\mathbf{X}_{\text{adv}}(\cdot)$. The trajectory prediction model predicts the future trajectories of both the adv agent and other agents. The attack goal is to mislead the predictions at each time step and subsequently make the AV plan execute unsafe driving behaviors. As illustrated in Figure 5.1, by driving along a carefully crafted adversarial (history) trajectory, the trajectory prediction model predicts wrong future trajectories for both the adv agent and the other agent. The mistakes can in term lead to severe consequences such as collisions. In this work, we focus on the white-box threat model, where the adversary has access to both model parameters, history trajectories and future trajectories of all agents, to explore what a powerful adversary can do based on the Kerckhoffs's principle [130] to better motivate defense methods.

Challenges. The challenges of devising effective adversarial attacks against prediction modules are two-fold: (1) **Generating realistic adversarial trajectory**. In AV systems, history trajectories are generated by upstream tracking pipelines and are usually sparsely queried due to computational constraints. On the other hand, dynamic parameters like accelerations and curvatures are high order derivatives of position and are usually estimated by numerical differentiation requiring calculating difference between positions *within a small-time interval*. Therefore, it is difficult to estimate correct dynamic parameters from such sparsely sampled positions in the history trajectory. Without the correct dynamic parameters, it is impossible to determine whether a trajectory is realistic or not, let alone generate new trajectories. (2) **Evaluating the implications of adversarial attacks**. Most existing evaluation metrics for trajectory prediction assume benign settings and are inadequate to demonstrate the implications for AV systems under attacks. For example, a large Average

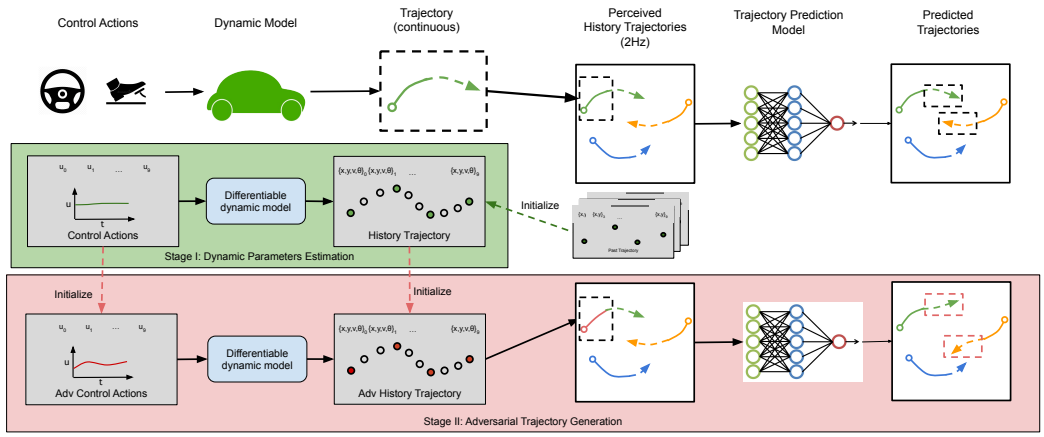


Figure 5.2: Adversarial Dynamic Optimization (AdvDO) methodology overview

Distance Error (ADE) in prediction does not directly entail concrete consequences such as collision. Therefore, we need a new evaluation pipeline to systematically determine the consequences of adversarial attacks against prediction modules to further raise the awareness of general audiences on the risk that AV systems might face.

5.5 AdvDO: Adversarial Dynamic Optimization

To address the two challenges listed above, we propose **Adv**ersarial **D**ynamic **O**ptimization (AdvDO). As shown in Figure 5.2, given trajectory histories, AdvDO first estimates their dynamic parameters via a differentiable dynamic model. Then we use the estimated dynamic parameters to generate a realistic adversarial history trajectory given a benign trajectory by solving an adversarial optimization problem. Specifically, AdvDO consists of two stages: (1) dynamic parameters estimation, and (2) adversarial trajectory generation. In the first stage, we aim to estimate correct dynamic parameters by reconstructing a realistic dense trajectory from a sampled trajectory from the dataset. To reconstruct the dense trajectory, we leverage a differentiable dynamic model through optimization of control actions. When we get the estimated correct dynamic parameters of the trajectory, it could be used for the second stage. In the second stage, we aim to generate an adversarial trajectory that

misleads future trajectory predictions given constraints. To achieve such goal, we carefully design the adversarial loss function with several regularization losses for the constraints. Then, we also extend the method to attacking consecutive predictions.

5.5.1 Dynamic Parameters Estimation

Differentiable dynamic model. A dynamic model computes the next state $s^{t+1} = \{p^{t+1}, \theta^{t+1}, v^{t+1}\}$ given current state $s^t = \{p^t, \theta^t, v^t\}$ and control actions $u^t = \{a^t, \kappa^t\}$. Here, p, θ, v, a, κ represent position, heading, speed, acceleration and curvature correspondingly. We adopt the kinematic bicycle model as the dynamic model which is commonly used [146]. We calculate the next state with a differential method, e.g., $v^{t+1} = v^t + a^t \cdot \Delta t$ where Δt denotes the time difference between two time steps. Given a sequence of control actions $u = (u^0, \ldots, u^t)$ and the initial state s^0, we denote the dynamic model as a differentiable function Φ such that it can calculate a sequence of future states $s = (s^0, \ldots, s^t) = \Phi(s^0, u; \Delta t)$. Noticed that the dynamic model also provides a reverse function Φ^{-1} that calculate a sequence of dynamic parameters $\{\theta, v, a, \kappa\} = \Phi^{-1}(p; \Delta t)$ given a trajectory $p = (p^0, \ldots, p^t)$. This discrete system can approximate the linear system in the real world when using a sufficiently small enough Δt. It can be also demonstrated that the dynamic model approximates better using a smaller Δt.

Optimization-based trajectory reconstruction. To accurately estimate the dynamic parameters $\{\theta, v, a, \kappa\}$ given a trajectory p, a small time difference Δt or a large sampling rates $f = 1/\Delta t$ is required. However, the sampling rate of the trajectory in the trajectory prediction task is decided by the AV stack, and is often small (e.g. 2Hz for nuScenes [23]) limited by the computation performance of the hardware. Therefore, directly estimating the dynamic parameters from the sampled trajectory is not accurate, making it difficult to determine whether the adversarial history $\mathbf{X}_{adv}$ generated by perturbing the history trajectory provided by the AV sys-

tem is realistic or not. To resolve this challenge, we propose to reconstruct a densely trajectory first and then estimate a more accurate dynamic parameter from the reconstructed dense trajectory. To reconstruct a densely sampled history trajectory $\mathbf{D}_i = \left(\mathbf{D}_i^{-H \cdot f+1}, \ldots, \mathbf{D}_i^0\right)$ from a given history trajectory $\mathbf{X}_i$ with additional sampling rates f, we need to find a realistic trajectory $\mathbf{D}_i$ that passes through positions in $\mathbf{X}_i$. We try to find it through solving an optimization problem. In order to efficiently find a realistic trajectory, we wish to optimize over the control actions in stead of the positions in $\mathbf{D}_i$. To start with, we initialize $\mathbf{D}_i$ with a simple linear interpolation of $\mathbf{X}_i$, i.e. $\mathbf{D}_i^{-t \cdot f+j} = (1 - j/f) \cdot \mathbf{X}^{-t} + j/f \cdot \mathbf{X}^{-t+1}$. We then calculate the dynamic parameters for all steps $\{\theta, v, a, \kappa\} = \Phi^{-1}(\mathbf{D}_i; \Delta t)$. Now, we can represent the reconstructed densely sampled trajectory $\mathbf{D}_i$ with $\Phi(s^0, u; \Delta t)$, where $u = \{a, \kappa\}$. To further reconstruct a realistic trajectory, we optimize over the control actions u with a carefully designed reconstruction loss function $\mathcal{L}_{\text{recon}}$. The reconstruction loss function consists of two terms. We first include a MSE (Mean Square Error) loss to enforce the reconstructed trajectory passing through the given history trajectory $\mathbf{X}_i$. We also include l_{dyn}, a regularization loss based on a soft clipping function to bound the dynamic parameters in a predefined range based on vehicle dynamics [146]. To summarize, by solving the optimization problem of:

$$\min_u \mathcal{L}_{\text{recon}}(u; s^0, \Phi) = MSE(\mathbf{D}_i, \mathbf{X}_i) + l_{\text{dyn}}(\theta, v, a, \kappa)$$

,we reconstruct a densely sampled, dynamically feasible trajectory $\mathbf{D^*}_i$ passing through the given history trajectory for the adversarial agent.

5.5.2 Adversarial Trajectory Generation

Attacking a single-step prediction. To generate realistic adversarial trajectories, we first initialize the dynamic parameters of the adversarial agent with estimation

from the previous stage, noted as $\mathbf{D^*}_{orig}$. Similarly to the optimization in the trajectory reconstruction process, we optimize the control actions u to generate the optimal adversarial trajectories. Our adversarial optimization objective consists of four terms. The detailed formulation for each term is in the supplementary materials. The first term l_{obj} represents the attack goal. As motion and social properties are essential and unique for trajectory prediction models. Thus, our l_{obj} has accounted for them when designed. The second term l_{col} is a commonsense objective that encourages the generated trajectories to follow some commonsense traffic rules. In this work we only consider collision avoidance [138]. The third term l_{bh} is a regularization loss based on a soft clipping function, given a clipping range of $(-\epsilon, \epsilon)$. It bounds the adversarial trajectories to be close to the original history trajectory $\mathbf{X}_{\text{orig}}$. We also include l_{dyn} to bound the dynamic parameters. The full adversarial loss is defined as:

$$\mathcal{L}_{\text{adv}} = l_{\text{obj}}(\mathbf{Y}, \hat{\mathbf{Y}}) + \alpha \cdot \sum_i l_{\text{col}}(\mathbf{D}_{\text{adv}}, \mathbf{X}) + \beta \cdot l_{\text{bh}}(\mathbf{D}_{\text{adv}}, \mathbf{D^*}_{\text{orig}}) + \gamma l_{\text{dyn}}(\mathbf{D}_{adv})$$

,where α and β are weighting factors. We then use the projected gradient descent (PGD) method [99] to find the adversarial control actions u_{adv} bounded by constraints (u_{lb}, u_{ub}) attained from vehicle dynamics.

Attacking consecutive predictions. To attack L_p consecutive frames of predictions, we aim to generate the adversarial trajectory of length $H + L_p$ that uniformly misleads the prediction at each time frames. To achieve this goal, we can easily extend the formulation for attacking single-step predictions to attack a sequence of predictions, which is useful for attacking a sequential decision maker such as an AV planning module. Concretely, to generate the adversarial trajectories for L_p consecutive steps of predictions formulated in§ 5.4, we aggregate the adversarial losses over these frames. The objective for attacking a length of $H + L_p$ trajectory is:

$$\sum_{t\in[-L_p,\ldots 0]} \mathcal{L}_{\text{adv}}(\mathbf{X}(t), \mathbf{D}_{\text{adv}}(t), \mathbf{Y}(t))$$

, where $\mathbf{X}(t), \mathbf{D}_{\text{adv}}(t), \mathbf{Y}(t)$ are the corresponding $\mathbf{X}, \mathbf{D}_{\text{adv}}, \mathbf{Y}$ at time frame t.

5.6 Experiments

Our experiments seek to answer the following questions: (1) Are the current mainstream trajectory prediction systems robust against our attacks?;(2) Are our attacks more realistic compared to other methods?; (3) How do our attacks affect an AV prediction-planning system?; (4) Does features designed to model motion and/or social properties affect a model's adversarial robustness?; and (5) Could we mitigate our attack via adversarial training?

5.6.1 Experimental Setting

Models. We evaluate two state-of-the-art trajectory prediction models: AgentFormer and Trajectron++. As explained before, we select AgentFormer and Trajectron++ for their representative features in modeling motion and social aspects in prediction. AgentFormer proposed a transformer-based social interaction model which allows an agent's state at one time to directly affect another agent's state at a future time. And Trajectron++ incorporates agent dynamics. Since semantic map is an optional information for these models, we prepare two versions for each model with map and without map.

Datasets. We follow the settings in [185, 126] and use nuScenes dataset [23], a large-scale motion prediction dataset focusing on urban driving settings. We select history trajectory length ($H = 4$) and future trajectory length ($T = 12$) following the official recommendation. We report results on all 150 validation scenes.

Baselines. We select the search-based attack proposed by Zhang et al. [189] as

the baseline, named *search*. As we mentioned earlier in § 5.8, the original method made two mistakes: (1) incorrect estimated bound values for dynamic parameters and (2) incorrect choices of bounded dynamic parameters for generating realistic adversarial trajectories. We correct such mistakes by (1) using a set of real-world dynamic bound values [146]. and (2) bounding the curvature variable instead of heading derivatives since curvature is linear related to steering angle. We denote this attack method as *search**. For our methods, we evaluate two variations: (1) *Opt-init*, where the initial dynamics (i.e dynamics at ($t = -H$) time step) $\mathbf{D}_{adv}^{-H \cdot S+1}$ are fixed and (2) *Opt-end*, where the current dynamics ($t = 0$) $\mathbf{D}_{adv}^{0}$ are fixed. While *Opt-end* is not applicable for sequential attacks, we include *Opt-end* for understanding the attack with strict bounds, since the current position often plays an important role in trajectory prediction.

Metrics. We evaluate the attack with four metrics in the nuscenes prediction challenges: ADE/FDE, Miss Rates (MR), Off Road Rates (ORR) [23] and their correspondence with planning-awareness version: PI-ADE/PI-FDE, PI-MR, PI-ORR [76] where metric values are weighted by the sensitivity to AV planning. In addition, to compare which attack method generates the most realistic adversarial trajectories, we calculate the violation rates (VR) of the curvature bound, where VR is the ratio of the number of adversarial trajectories violating dynamics constraints over the total number of generated adversarial trajectories.

Implementation details. For the trajectory reconstruction, we use the Adam optimizer and set the step number of optimization to 5. For the PGD-based attack, we set the step number to 30 for both AdvDO and baselines. We empirically choose $\beta = 0.1$ and $\alpha = 0.3$ for best results.

5.6.2 Main Results

Trajectory prediction under attacks. First, we compare the effectiveness of the attack methods on prediction performances. As shown in Table 5.1, our proposed attack (*Opt-init*) causes the highest prediction errors across all model variants and metrics. *Opt-init* overperforms *Opt-end* by a large margin, which shows that the dynamics of the current frame play an important role in trajectory prediction systems. Note that *search* proposed by Zhang *et al.* has a significant violation rates (VR) over 10%. It further validates our previous claim that *search* generates unrealistic trajectories.

Table 5.1: Attack evaluation results on general metrics.

Model	Attack	ADE	FDE	MR	ORR	Violations
Agentformer w/ map	None	1.83	3.81	28.2%	4.7%	0%
	search	2.34	4.78	34.3%	6.6%	10%
	*search**	1.88	3.89	29.2%	4.8%	0%
	Opt-end	2.23	4.54	34.5%	6.3%	0%
	Opt-init	**3.39**	**5.75**	**44.0%**	**10.4%**	0%
Agentformer w/o map	None	2.20	4.82	35.0%	7.3%	0%
	search	2.66	5.53	40.3%	8.9%	9%
	*search**	2.20	4.94	35.1%	7.4%	0%
	Opt-end	2.54	5.54	39.3%	8.8%	0%
	Opt-init	**3.81**	**6.01**	**49.8%**	**13.3%**	0%
Trajectron++ w/ map	None	1.88	4.10	35.1%	7.9%	0%
	search	2.53	5.03	44.4%	9.4%	12%
	*search**	1.93	4.26	36.3%	8.3%	0%
	Opt-end	2.48	5.57	47.5%	11.3%	0%
	Opt-init	**3.20**	**8.56**	**57.2%**	**15.9%**	0%
Trajectron++ w/o map	None	2.10	5.00	41.1%	9.6%	0%
	search	2.76	8.02	50.5%	16.1%	14%
	*search**	2.17	5.25	42.2%	10.0%	0%
	Opt-end	2.49	7.54	49.5%	14.2%	0%
	Opt-init	**3.58**	**9.36**	**76.8%**	**17.8%**	0%

To further demonstrate the impact of the attacks on downstream pipelines like planning, here we report prediction performance using planning-aware metrics pro-

posed by Ivanovic *et al.* [76]. As described above, these metrics consider how the predictions accuracy of surrounding agents behaviors impact the ego's ability to plan its future motion. Specifically, the metrics are computed from the partial derivative of the planning cost over the predictions to estimate the sensitivity of the ego vehicle's further planning. Furthermore, by aggregating weighted prediction metrics (e.g., ADE, FDE, MR, ORR) with such sensitivity measurement, we could report planning awareness metrics including (PI-ADE/FDE, PI-MR, PI-ORR) quantitatively. As shown in Table 5.2, results are consistent with the previous results.

Table 5.2: Attack evaluation results on planning-aware metrics.

Model	Attack	PI-ADE	PI-FDE	PI-MR	PI-ORR	VR
Agentformer w/ map	None	1.38	2.76	20.5%	22.8%	0%
	search	1.62	3.32	25.7%	25.2%	13%
	*search**	1.39	2.79	21.4%	23.0%	0%
	Opt-end	1.57	3.11	23.7%	24.8%	0%
	Opt-init	**2.05**	**3.81**	**32.9%**	**29.0%**	0%
Agentformer w/o map	None	1.46	3.76	26.8%	30.3%	0%
	search	1.63	4.12	28.9%	34.2%	11%
	*search**	1.49	3.74	27.5%	31.1%	0%
	Opt-end	1.63	4.11	28.2%	39.3%	0%
	Opt-init	**2.24**	**5.91**	**34.3%**	**41.3%**	0%
Trajectron++ w/ map	None	1.42	2.81	26.5%	25.6%	0%
	search	1.68	3.38	29.2%	28.3%	14%
	*search**	1.43	2.83	26.7%	27.7%	0%
	Opt-end	1.65	3.14	27.2%	28.1%	0%
	Opt-init	**2.11**	**3.85**	**37.8%**	**32.7%**	0%
Trajectron++ w/o map	None	1.76	3.20	30.9%	44.0%	0%
	search	2.02	3.96	35.0%	49.6%	19%
	*search**	1.77	3.25	31.0%	46.8%	0%
	Opt-end	1.95	3.55	31.6%	46.3%	0%
	Opt-init	**2.46**	**4.26**	**41.2%**	**53.7%**	0%

Attack fidelity analysis. Here, we aim to demonstrate the fidelity of the generated adversarial trajectories qualitatively and quantitatively. We show our analysis on AgentFormer with map as a case study. In Figure 5.3, we visualize the adver-

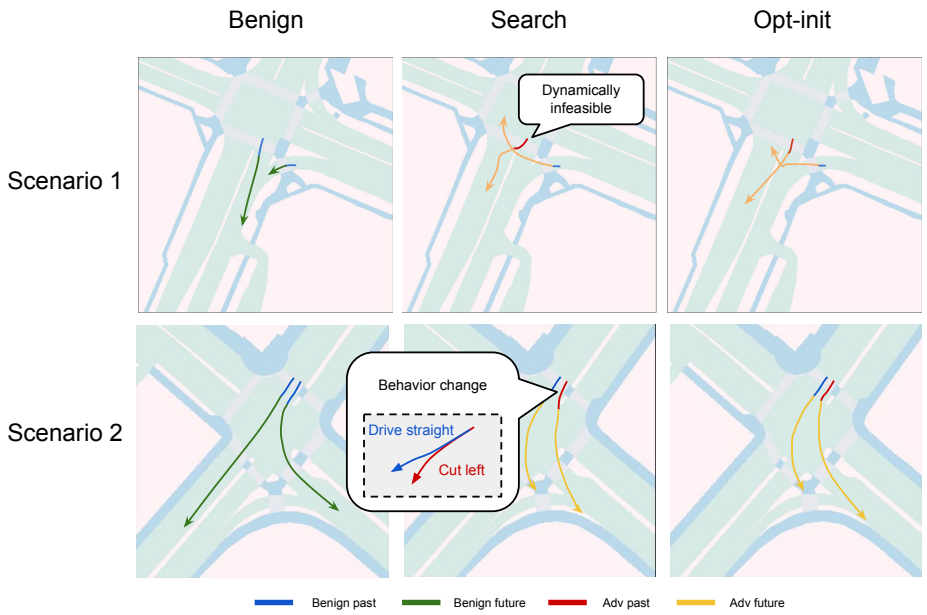


Figure 5.3: Qualitative comparison of generated adversarial trajectories. We demonstrate that the proposed AdvDO generates adversarial trajectories both realist and effective whereas the search-stats could either generate dynamically infeasible trajectories (sharp turn on the first row) or changing the behavior dramatically (behavior change from driving straight to swerving left on the second row).

Table 5.3: Quantitative comparison of generated adversarial trajectories

Method	*search*	*Opt-end*	*Opt-init*
ΔSensitivity	2.33	1.12	1.34

sarial trajectories generated by *search* and *Opt-end* methods. We demonstrate that our method (*Opt-end*) can generate effective attack without changing the semantic meaning of the driving behaviors. In contrast, *search* either generates unrealistic trajectories or changes the driving behaviors dramatically. For example, the middle row shows that the adversarial trajectory generated by *search* takes a near 90-degree sharp turn within a small distance range, which is dynamically feasible, whereas by our method (right image in the first row) generates smooth and realistic adversarial trajectories. In addition, we conduct a human study and demonstrate that only 4.4($\pm$2.6)% of the generated adversarial trajectories are considered rule-violating.

More examples of generated adversarial trajectories and details of the human study can be found in Appendix B.

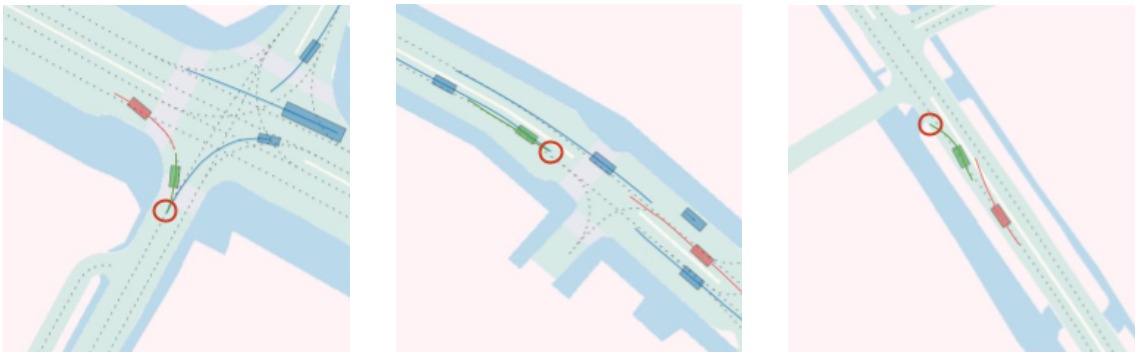
To further quantify the attack fidelity, we propose to use the sensitivity metric in [76] to measure the degree of behavior alteration caused by the adversarial attacks. The metric is to measure the influence of an agent's behavior over other agents' future trajectories. We calculate the difference of aggregated sensitivity of non-adv agents between the benign and adversarial settings. Detailed formulation is in Appendix B. We demonstrate that our proposed attacks (*Opt-init*, *Opt-end*) cause smaller sensitivity changes. This corroborates our qualitative analysis that our method generates more realistic attacks at the behavior level.

Table 5.4: Planning results

Planner	Open-loop		Closed-loop	
	Rule-based	MPC	Rule-based	MPC
Collisions	26/150	10/150	12/150	7/150
Off road	–	43/150	–	23/150

Case studies with planners. To explicitly demonstrate the consequences of our attacks to the AV stack, we evaluate the adversarial robustness of a prediction-planning pipeline in an end-to-end manner. We select a subset of validation scenes and evaluate two planning algorithms, rule-based [117] and MPC-based [24], in in two rollout settings, open-loop and closed-loop. Detailed description for the planners can be found in Appendix B. In the open-loop setting, an ego vehicle generates and follows a 6-second plan without replanning. The closed-loop setting is to replan every 0.5 seconds. We replay the other actors' trajectories in both cases. For the closed-loop scenario, we conduct the sequential attack using $L_p = 6$. As demonstrated in Table 5.4, our attacks causes the ego to collide with other vehicles and/or leave drivable regions. We visualize a few representative cases in Figure 5.4. Figure 5.4(a) shows the attack leads to a side collision. Figure 5.4(b) shows the attack misleads the pre-

diction and forces the AV to stop and leads to a rear-end collision. Note that no attack can lead the rule-based planner to leave drivable regions because it is designed to keep the ego vehicle in the middle of the lane. At the same time, we observed that attacking the rule-based planner results in more collisions since it cannot dodge head-on collisions.



(a) Side collision (b) Rear-end collision (c) Driving off-road

Figure 5.4: Visualized results for planner evaluation. Ego vehicle in green, adv agent in red and other agents in blue. The red cycle represents the collision or driving off-road consequence.

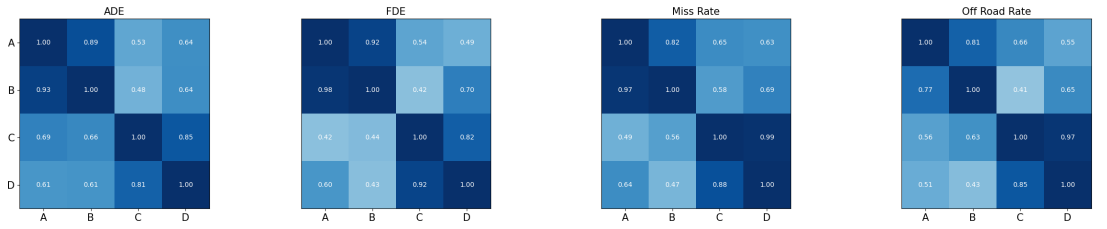
Motion and social modeling. As mentioned in § 5.3, trajectory prediction model aims to learn (1) the motion dynamics of each agent and (2) social interactions between agents. Here we conduct more in-depth attack analysis with respect to these two properties. For the motion property, we introduce a *Motion* metric that measures the changes of predicted future trajectory of the adversarial agent as a result of the attack. For the social property, we hope to evaluate the influence of the attack on the predictions of non-adv agents. Thus, we use a metric named *Interaction* to measure the average prediction changes among all non-adv agents. As shown in Table 5.5, the motion property is more prone to attack than the interaction property. This is because perturbing the adv agent's history directly impacts its future, while non-adv agents are affected only through the interaction model. We observed that our attack leads to larger *Motion* error for AgentFormer than for Trajectron++. A possible

explanation is that AgentFormer enables direct interactions between past and future trajectories across all agents, making it more vunerable to attacks.

Table 5.5: Ablation results for Motion and Interaction metrics

Model	Scenarios	ADE	FDE	MR	ORR	Model	ADE	FDE	MR	ORR
AgentFormer	*Motion*	8.12	12.35	57.3%	18.6%	Trajectron++	8.75	13.27	59.6%	16.6%
	Interaction	2.03	4.21	30.3%	5.1%		1.98	4.68	43.0%	8.71%

Transferability analysis. Here we evaluate whether the adversarial examples generated by considering one model can be transferred to attack another model. We report *transfer rate* (more details in the Appendix B). Results are shown in Figure 5.5. Cell (i, j) shows the normalized transfer rate value of adversarial examples generated against model j and evaluate on model i. We demonstrate that the generated adversarial trajectories are highly transferable (transfer rates $\geq 77\%$) when sharing the same backbone network. In addition, the generated adversarial trajectories can transfer among different backbones as well. These results show the feasibility for black-box attacks against unseen models in the real-world.



(a) ADE (b) FDE (c) Miss Rate (d) Off Road Rate

Figure 5.5: Transferability heatmap. A: AgentFormer w/ map; B: AgentFormer w/o map; C: Trajectron++ w/ map; D: Trajectron++ w/o map

Mitigation. To mitigate the consequences of the attacks, we use the standard mitigation method, adversarial training [99], which has been shown as the most effective defense. As shown in Table C in the Appendix B, we find that the adversarial trained model using the *search* attack is much worse than the adversarial trained model using our *Opt-init* attack. This can be due to unrealistic adversarial trajectories generated

by the *search*, which also emphasizes that generating realistic trajectory is essential to success of improving adversarial robustness.

5.7 Adversarially Robust Trajectory Prediction

Adversarial robustness for machine learning is a widely-studied area, but most works focus on classification tasks [78, 91, 108, 106, 128, 152, 166, 163, 167, 169, 175, 187]. Among the proposed techniques, adversarial training [98] remains the most effective and widely used method to defend classifiers against adversarial attacks. The general strategy of adversarial training is to solve a min-max game by generating adversarial examples for a model at each training step and then optimizing the model to make correct predictions for these samples. However, directly applying adversarial training to trajectory prediction presents a number of critical technical challenges.

First, most trajectory prediction methods employ probabilistic generative models to cope with the uncertainty in motion forecasting [75, 126, 185, 122, 123, 82]. As we will show in this paper, the stochastic components of these models (e.g., posterior sampling in VAEs) can obfuscate the gradients that guide the adversarial generation, making naïve adversarial training methods ineffective. Second, adversarial training on trajectory prediction task aims to model joint data distribution of future trajectories and adversarial past trajectories. However, the co-evolution of the adversarial sample distribution and the prediction model during the training process makes the joint distribution hard to model and often destabilizes the adversarial training. Finally, prior work [187] shows that adversarial training often leads to degraded performance on clean (unperturbed) data, while retaining good performance in benign cases is crucial due to the critical role of trajectory prediction for AVs. Hence, an effective adversarial training method must carefully balance the benign and the adversarial performance of a model.

Our approach. We propose an adversarial training framework for trajectory pre-

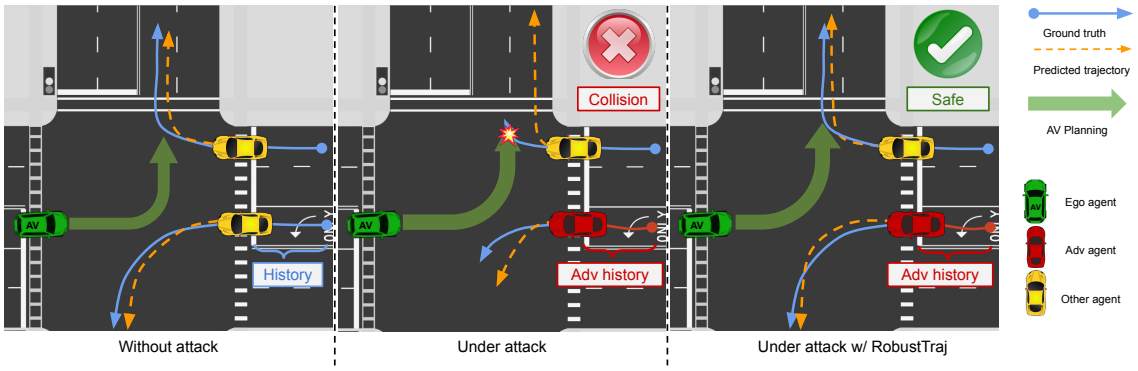


Figure 5.6: Overview of *RobustTraj* preventing Autonomous Vehicle (AV) from collisions when its trajectory prediction model is under adversarial attacks. When the trajectory prediction model is under attack, the AV predicts the wrong future trajectory of the other agent turning right (yellow vehicle). This results in AV speeding up instead of slowing down, and eventually colliding into the other vehicle.

dictions named *RobustTraj*, by addressing the aforementioned challenges. First, to address the issue of an obfuscated gradient in adversarial generation due to stochastic components, we devise a *deterministic attack* that creates a deterministic gradient path within a probabilistic model to generate adversarial samples. Second, to address the challenge of an unstable training process due to shift in adversarial distributions, we introduce a hybrid objective that interleaves the adversarial training and learning from clean data to anchor the model output on stable clean data distribution. Finally, to achieve balanced performances on both adversarial and clean data, we introduce a domain-specific data augmentation technique for trajectory prediction via a dynamic model. This data augmentation technique generates diverse, realistic, and dynamically-feasible samples for training and achieves a better performance trade-off on clean and adversarial data.

We empirically show that *RobustTraj* can effectively defend two different types of probabilistic trajectory prediction models [185, 82] against adversarial attacks, while incurring minimal performance degradation on clean data. For instance, *RobustTraj* can increase the adversarial performance of AgentFormer [185], a state-of-the-art

trajectory prediction model, by 46% at the cost of 3% performance drop on clean data. To further show impacts of our method on the AD stack, we plug our robust trajectory prediction model into a planner and demonstrate that our model reduces serious accidents rates (e.g., collisions and off-road driving) under attacks by 100%, compared to the standard non-robust model trained using only clean data.

5.8 Related Work

Adversarial attacks and defenses on trajectory prediction. A recent work began to study the adversarial robustness of trajectory prediction models [190]. *Zhang et al.* [190] demonstrated that perturbing agents' observed trajectory can adversarially impact the prediction accuracy of a DNN-based trajectory forecasting model. To mitigate the issue, *Zhang et al.* [190] proposed several defense methods such as data augmentation and trajectory smoothing. However, these methods are less effective when facing adaptive attacks [19]. In our work, we propose to use adversarial training which provides the general adversarial robustness that can resist adaptive attacks.

Adversarial scenario generation. A few recent studies work on generating adversarial traffic scenarios such that the autonomous driving systems fail to make safe driving decisions [146, 117]. However, generating realistic traffic scenarios is challenging and the generated adversarial scenarios can be unrealistic and violate traffic rules by directly optimizing the latent vectors of the traffic model *Rempe et al.* [118]. In this work, we consider defending against realistic adversarial scenarios grounded on the scenarios from a dataset. *Wang et al.* [146] perturb the raw input data to mislead the full stack AV system. However, in this work, our primary goal is to study and improve the robustness of trajectory prediction models. To obtain salient and unambiguous insights, we minimize the conflating factors in our analysis without considering the perception model.

Adversarial training. A variety of adversarial training methods have been proposed

to defend DNN-based models against adversarial attacks [98, 163, 78, 91, 108, 106, 128, 152, 166, 163, 167, 169, 175, 187]. The most common strategy is to design a min-max game with the inner maximization process and outer minimization process. The inner maximization process generates adversarial examples that maximize an adversarial objective (e.g., make wrong prediction). The outer minimization process then updates the model parameters to minimize the error on the adversarial examples. Several recent works also propose to mix clean data and adversarial examples for improving robustness [88, 144] and performance on clean data [166]. Although there exists a large body of literature in studying adversarial robustness for machine learning, most focus on the problem of discriminative model (e.g., object recognition), leaving other problem domains (e.g., conditional generative models) largely unexplored. In this work, we develop a novel adversarial training method for trajectory prediction models, where most state-of-the-art trajectory prediction models are generative and probabilistic, by addressing a number of critical technical challenges.

5.9 Preliminaries and Formulation

Probabilistic trajectory prediction models. In this work, we focus on defending generative, probabilistic trajectory prediction models, as they have demonstrated superior performance in modeling uncertainty in predicting future motions [75, 126, 185, 122, 123, 82]. We consider the two most popular types of generative models: conditional variational encoders (CVAEs) and conditional GANs (cGANs), both can be viewed as latent variable models. We define latent variables $\mathbf{Z} = \{z_1, ..., z_i, ..., z_N\}$ where z_i represents the latent variable of the agent i. CVAE formulates the generative problem as: $p_\theta(\mathbf{Y}|\mathbf{X}) = \int p_\theta(\mathbf{Y}|\mathbf{X}, \mathbf{Z}) \cdot p_\theta(\mathbf{Z}|\mathbf{X}) d\mathbf{Z}$, where $p_\theta(\mathbf{Z}|\mathbf{X})$ is a conditional Gaussian prior ($\mathcal{N}(p_\theta^\mu(\mathbf{Z}|\mathbf{X}), p_\theta^\sigma(\mathbf{Z}|\mathbf{X}))$) with mean $p_\theta^\mu(\mathbf{Z}|\mathbf{X})$ and standard deviation $p_\theta^\sigma(\mathbf{Z}|\mathbf{X})$; $p_\theta(\mathbf{Y}|\mathbf{X}, \mathbf{Z})$ is a conditional likelihood model. The model is usually trained

through optimizing a negative evidence lower objective [185]:

$$\begin{aligned}\mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{elbo}} + \mathcal{L}_{\text{diversity}} \\ &= -\mathbb{E}_{q_\phi(\mathbf{Z}|\mathbf{Y},\mathbf{X})}[\log p_\theta(\mathbf{Y}|\mathbf{Z},\mathbf{X})] + \text{KL}(q_\phi(\mathbf{Z}|\mathbf{Y},\mathbf{X}) \parallel p_\theta(\mathbf{Z}|\mathbf{X})) + \min_k \parallel \hat{\mathbf{Y}}^{(k)} - \mathbf{Y} \parallel^2 ,\end{aligned} \tag{5.1}$$

where $q_\phi(\mathbf{Z}|\mathbf{Y},\mathbf{X})$ is an approximate posterior parameterized by ϕ, $p_\theta(\mathbf{Z}|\mathbf{X})$ is a conditional Gaussian prior parameterized by θ, and $p_\theta(\mathbf{Y}|\mathbf{Z},\mathbf{X})$ is a conditional likelihood modeling future trajectory $\mathbf{Y}$ via the latent codes $\mathbf{Z}$ and past trajectory $\mathbf{X}$. Additionally, $\mathcal{L}_{\text{diversity}} = \min_k \parallel \hat{\mathbf{Y}}^{(k)} - \mathbf{Y} \parallel^2$ is a diversity loss, which encourages the network to produce diverse samples. Given each past trajectory X, the model generates K sets of latent codes $\{\mathbf{Z}^{(1)}, \cdots, \mathbf{Z}^{(k)}, \cdots, \mathbf{Z}^{(K)}\}$ from the conditional Gaussian prior $\mathcal{N}(p_\theta^\mu(\mathbf{Z}|\mathbf{X}), p_\theta^\sigma(\mathbf{Z}|\mathbf{X}))$, where $\mathbf{Z}^{(k)} = \{z_1^k, \cdots, z_n^k\}$, resulting in K future trajectories $\hat{\mathbf{Y}}^{(k)}$.

Similarly, in a conditional Generative Adversarial Net (cGAN)-based model (e.g., Social-GAN [17]), it uses a loss function as follows:

$$\begin{aligned}\mathcal{L}_{\text{total}} &= \mathcal{L}_{\text{gan}} + \mathcal{L}_{\text{diversity}} \\ &= \mathbb{E}_{\mathbf{Y}\sim p_{\text{data}}}[\log D_\theta(\mathbf{Y}|\mathbf{X})] + \mathbb{E}_{\mathbf{Z}\sim p_{\text{Z}}}[\log(1 - D_\theta(G_\phi(\mathbf{Y}|\mathbf{X},\mathbf{Z})))] + \min_k \parallel \hat{\mathbf{Y}}^{(k)} - \mathbf{Y} \parallel^2 ,\end{aligned} \tag{5.2}$$

where G represents the generator and D represents the discriminator. $\hat{\mathbf{Y}}^{(k)} = G(\mathbf{Y}|\mathbf{X},\mathbf{Z}^{(k)})$ is one of the predicted trajectories in which $\mathbf{Z}^{(k)}$ is randomly sampled from $\mathcal{N}(0,1)$. During the training, $\mathcal{L}_{\text{gan}}$ is maximized to train D and $\mathcal{L}_{\text{total}}$ is minimized to train G.

Naïve adversarial training. Adversarial training formulates a min-max game with an inner maximization process that optimizes the perturbation δ to generate adversarial examples for misleading the model at each training iteration, and an outer minimization process that optimizes the model parameters to make correct predictions

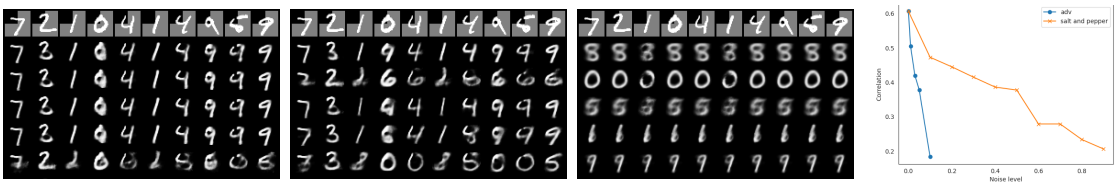
for these examples. We follow the standard adversarial training formulation [98]:

$$\min_{\theta,\phi} \max_{\delta \in \mathbb{S}} \quad \mathcal{L}_{\text{total}}(\mathbf{X} + \delta, \mathbf{Y}). \tag{5.3}$$

5.10 *RobustTraj*: Robust Trajectory Prediction

As stated earlier, applying adversarial training for trajectory prediction presents three critical challenges: (1) gradient obfuscation due to model stochasticity, (2) unstable learning due to changing adversarial distribution, and (3) performance loss in the benign situation. In this section, we describe each challenge in more detail and present the corresponding solutions in our *RobustTraj* method.

Improve adversarial generation with *Deterministic Attack*. Since trajectory prediction is inherently uncertain and there is no single correct answer, probabilistic generative models are usually used to cope with the stochastic nature of the trajectory prediction task. Such stochasticity will obfuscate the gradients that are used to generate effective adversarial examples in the inner maximization process of adversarial training. The naïve attack mentioned in section 5.9 is a straightforward way to achieve this goal. However, this optimization involves a stochastic sampling process $\mathbf{Z}^{(k)} \sim \mathcal{N}(p_\theta^\mu(\mathbf{Z}|\mathbf{X}), p_\theta^\sigma(\mathbf{Z}|\mathbf{X}))$. Such a stochastic process will obfuscate the gradients for finding the optimal adversarial perturbation δ, making the outer minimization (robust training) less effective. In order to sidestep such stochasticity, we propose the *deterministic attack* that creates a deterministic gradient path within the model to generate the adversarial perturbation. $\hat{\mathbf{Z}}$. Specifically, we use a *deterministic latent code* by replacing the sampling process $\mathbf{Z}^{(k)} \sim \mathcal{N}(p_\theta^\mu(\mathbf{Z}|\mathbf{X}), p_\theta^\sigma(\mathbf{Z}|\mathbf{X}))$, with the maximum-likelihood sample (here, i.e $\hat{\mathbf{Z}} = p_\theta^\mu(\mathbf{Z}|\mathbf{X})$). The objective for generating



(a) Clean (b) Salt and pepper (c) Adv noise (d) Correlation

Figure 5.7: Visualizations of the CVAE models trained with clean (a) data, *Salt and pepper* noise (b), and adversarial perturbations (c); Quantitative results of the correlation between the label of the generated images and conditioned images at different noise levels (d).

the adversarial perturbation is thus:

$$\delta = \arg\max_{\delta \in \mathbb{S}} \mathcal{L}_{\text{adv}}(\mathbf{X}+\delta, \mathbf{Y}) = \arg\max_{\delta \in \mathbb{S}} \parallel p_\theta(\mathbf{Y}|\hat{\mathbf{Z}}, \mathbf{X}+\delta) - \mathbf{Y} \parallel^2 , \text{ where } \hat{\mathbf{Z}} = p_\theta^\mu(\mathbf{Z}|\mathbf{X}+\delta). \tag{5.4}$$

We empirically show that gradients from this deterministic gradient path can effectively guide the generation of adversarial examples. We name our attack as *Deterministic Attack*.

Stabilize adversarial training with bounded noise and hybrid objective. During the adversarial training process, the distribution of the perturbed input $\mathbf{X}+\delta$ coevolves with the training process as δ is calculated via an inner maximization process at each training iteration. Although δ is bounded by the adversarial set $\mathbb{S}_p^\epsilon$, the resulting latent condition variable $\mathbf{C} = f(\mathbf{X}+\delta)$ can be arbitrarily noisy since the Lipschitz constant of neural network layers (f) is not bounded during training (See *Lemma 1.* in Appendix C). Since $\mathbf{C} = f(\mathbf{X}+\delta)$ is noisy, it is a less informative signal compared to the deterministic signal X. Thus, modeling $p_\theta(\mathbf{Y}|\mathbf{X}+\delta, \mathbf{Z})$ becomes substantially harder. In an extreme case that $\mathbf{C} = f(\mathbf{X}+\delta)$ is super noisy and contains no information, the training process can degenerate to model $p_\theta(\mathbf{Y}|\mathbf{Z})$, resulting in the undesirable worse performance on the clean data.

To further validate the above hypothesis that it is hard to model $p_\theta(\mathbf{Y}|\mathbf{X}+\delta, \mathbf{Z})$ with a changing data distribution of $\mathbf{X}+\delta$, we conduct an additional experiment. For

simplicity, we use MNIST [48] as the dataset. As shown in Fig. 5.7, we divide each digit image into four quadrants. We take the top-left quadrant as the condition $\mathbf{X}$ and the remaining quadrants as the output $\mathbf{Y}$. We train a CVAE ($p_\theta(\mathbf{Y}|\mathbf{X}+\delta, \mathbf{Z})$) to model $\mathbf{Y}$ by using clean data ($\mathbf{X}$) or noisy data ($\mathbf{X}+\delta$), where δ represents salt and pepper noise [151] or adversarial noise [98], resulting in Fig. 5.7 (a), (b), (c) respectively. The top-left region of each image in the first row is the conditional variables $\mathbf{X}$. The rest of rows are the generated images with different $\mathbf{Z}$. Each column in the same row uses the same $\mathbf{Z}$. We observe that the model trained on clean data successfully captures the conditional distribution (i.e., the generated image highly depends on $\mathbf{X}$) while the model trained with adversarial noise degenerates and ignores the condition (i.e., each row generates images of the same digit). This result shows that the conditional generative model fails to learn from $\mathbf{X}$. To provide a quantitative analysis, we measure the correlation between the label of the generated images and the label of their conditioned image quadrants, resulting in Fig 5.7 (d). More details on how to calculate the correlation are in the Appendix C. We observe that the correlation drops as the noise level increases for both adversarial nose and *salt and pepper* noise. Adversarial noise is more effective to degenerate the conditional generative model. Therefore, we conclude that (1) the noises in the conditional data lead to degenerated conditional generative model (i.e., from CVAE to VAE); (2) the level of degeneration depends on the noise levels.

Based on the analysis result, to better learn a robust trajectory prediction model, we need to bound $|f(\mathbf{X}+\delta) - f(\mathbf{X})|$ to reduce the noise level. Hence, we propose the following regularization loss $\mathcal{L}_{\text{reg}}$:

$$\mathcal{L}_{\text{reg}} = \mathrm{d}(f(\mathbf{X}+\delta), f(\mathbf{X})), \tag{5.5}$$

where d is a distance function (e.g., we use L_2 norm as the distance metric).

In addition, because the clean data has a fixed distribution, simultaneously learning from the clean data during the adversarial training process anchors the conditional distribution on a stable clean data distribution. Specifically, we propose the following hybrid objective:

$$\mathcal{L}_{\text{clean}}(\mathbf{X}, \mathbf{Y}) = \mathcal{L}_{\text{total}}(\mathbf{X}, \mathbf{Y}), \tag{5.6}$$

where $\mathcal{L}_{\text{total}}$ could be the loss in Eq. 5.1 for CVAE-based model or Eq. 5.2 for cGAN-based model.

Protect benign performance using data augmentation. Adversarial training often leads to performance degradation on clean data [187]. However, trajectory prediction is a critical component for safety-critical AD systems and its performance degradation can result in severe consequences (e.g., collisions). Thus, it is important to balance the model performance on the clean and adversarial data when designing adversarial training algorithms.

To further improve the performance on clean and adversarial data, we need to address the overfitting problem of the min-max adversarial training [124]. Data augmentation is shown to be effective in addressing the problem in the image classification domain [115]. However, data augmentation in trajectory prediction is rarely studied and non-trivial. To design an effective augmentation algorithm, *Rebuffi et al.* [115] argues that the most important criterion is that the augmented data should be realistic and diverse. Thus, we design a dynamic-model based data augmentation strategy $\mathbb{A}$ shown in the Appendix C. By using the augmentation, we can generate diverse, realistic multi-agent trajectories for each scene and construct $\mathbb{D}_{\text{aug}}$.

RobustTraj. In summary, our adversarial training strategy for trajectory pre-

diction models is formulated as follows:

$$\begin{aligned} \delta &= \arg\max_{\delta \in \mathbb{S}} \mathcal{L}_{\text{adv}}(\mathbf{X}+\delta, \mathbf{Y}), \quad \text{where} \{\mathbf{X}, \mathbf{Y}\} \in \mathbb{D} \cup \mathbb{D}_{\text{aug}} \\ \theta,\, \phi &= \arg\min_{\theta,\phi} \mathcal{L}_{\text{total}}(\mathbf{X}+\delta, \mathbf{Y}) + \mathcal{L}_{\text{clean}}(\mathbf{X}, \mathbf{Y}) + \beta \cdot \mathcal{L}_{\text{reg}}(\mathbf{X},\, \mathbf{X}+\delta), \end{aligned} \tag{5.7}$$

where $\mathbb{D}$, $\mathbb{D}_{\text{aug}}$ are the training data and augmented data; $\mathcal{L}_{\text{adv}}$ is adversarial loss to generate effective adversarial examples in Eq. 5.4; $\mathcal{L}_{\text{total}}$ is the loss in Eq. 5.1 or Eq. 5.2 to train a robust model against adversarial examples; $\mathcal{L}_{\text{reg}}$ and $\mathcal{L}_{\text{clean}}$ are loss shown in Eq. 5.5 and Eq. 5.6 to provide a stable signal for training. β is a hyper-parameter for adjusting the regularization.

5.11 Experiments and Results

5.11.1 Experimental setup

Dataset and models. We follow the setting in prior work [185, 126] and use the nuScenes dataset [23] for evaluation. For the trajectory prediction models, we select the representative conditional generative models based on CVAE (AgentFormer [185]) and cGAN (Social-GAN [17]). AgentFormer is a state-of-the-art model based on CVAE and Social-GAN is a classic model based on cGAN. We report the final results for all three models: AgentFormer (AF), mini-AgentFormer (mini-AF) and Social-GAN. More details are shown in the Appendix C.

Training details and hyperparameter choices. For the adversarial training, we choose a 2-step Projected Gradient Descent (PGD) attack for the inner maximization and choose $\beta = 0.1$. We train 50 epochs and 100 epochs for AgentFormer and Social-GAN respectively. For other hyperparameters during training, we follow the original settings for AgentFormer and Social-GAN. The details for choosing these hyperparameters can be found in the Appendix C. All experiments are done on the NVIDIA V100 GPU [39]. We consider various baselines, including naïve adversarial

training (naïve AT) and four defenses proposed by Zhang *et al.* [190]: data augmentation with adversarial examples (*DA*), *train-time smoothing*, *test-time smoothing*, *DA + train-time smoothing* and *detection + test-time smoothing*.

Attack and evaluation metrics. For the adversarial attack, we choose a 20-step PGD attack (an ablation study on step convergence can be found in the Appendix C). Without loss of generality, we use L_∞ as the attack threat model so that $\mathbb{S} = \{\delta | \parallel \delta \parallel_\infty \leq \epsilon\}$. We select $\epsilon = \{0.5, 1.0\}$-meter, where the 1-meter deviation is the maximum change for a standard car without shifting to another lane [190]. We use four standard evaluation metrics for the nuscenes prediction challenge [23]: average displacement error (ADE), final displacement error (FDE), off road rates (ORR), and miss rate (MR). We evaluate the model's performance on both clean and adversarial data. For convenience, we use *ADE*, *FDE*, *ORR*, *MR* to represent the performance on the clean data and *Robust ADE*, *Robust FDE*, *Robust ORR*, *Robust MR* to represent the performance under attacks. We compute these metrics with the best of five predicted trajectory samples, i.e., $K = 5$.

5.11.2 Main results

Here, we present our main results of *RobustTraj*. We compare it with the baselines including model trained with clean data (*Clean*) and naïve adversarial training (*Naïve AT*), and existing defense methods for trajectory prediction [190]. The results have been shown in Table 5.6.

We observe that our method achieves the best robustness and maintains good clean performance for most cases. For instance, with $\epsilon = 0.5$ attack on AgentFormer model, our method is able to reduce 46% prediction errors ($\frac{5.09-2.73}{5.09}$) under the attack at a cost of 2.6% ($\frac{1.91-1.86}{1.86}$) clean performance degradation on ADE, compared to the model trained with clean data at $\epsilon = 0.5$. Compared to the existing methods, our method also significantly outperforms in terms of the robustness. For instance, with

Table 5.6: ADE and Robust ADE on different defense methods and models. The 1-st and 2-nd lowest errors are colored.

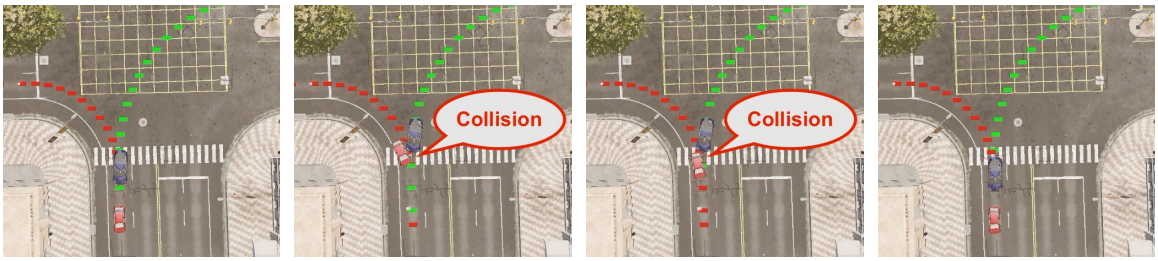
Model	mini-AF				AF				SGAN			
Method	ADE		Robust ADE		ADE		Robust ADE		ADE		Robust ADE	
	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0
Clean	**2.05**	**2.05**	6.86	11.53	**1.86**	**1.86**	5.09	8.57	**4.80**	**4.80**	10.52	20.15
Naïve AT [98]	2.75	2.78	5.44	9.20	2.52	2.56	**3.81**	6.81	6.43	6.55	**8.34**	**14.63**
DA [190]	2.31	2.32	5.54	9.32	2.10	2.08	4.35	7.22	5.41	5.40	8.85	17.25
Train-time Smoothing [190]	3.14	3.07	5.67	9.31	2.11	2.13	4.19	6.79	5.50	5.47	8.74	16.51
Test-time Smoothing [190]	2.97	3.07	**4.96**	**8.50**	2.40	2.41	4.43	7.44	6.16	6.17	9.05	17.42
DA + Train-time Smoothing [190]	2.41	2.39	5.48	9.00	2.17	2.13	4.14	**6.62**	5.63	5.61	8.60	16.14
Detection + Test Smoothing [190]	2.31	2.28	5.91	9.85	2.08	2.03	4.45	7.59	5.35	5.37	9.28	17.39
RobustTraj	**2.14**	**2.11**	**3.69**	**3.82**	**1.91**	**1.95**	**2.73**	**2.86**	**4.95**	**5.07**	**5.20**	**6.94**

$\epsilon = 1.0$ attack on AgentFormer model, our method achieves 45% better robustness with 9% better clean performance on ADE compared to the best results from existing methods [190].

Impacts to downstream planners. To further study the downstream impact of our robust trajectory model in the AD stack, we plug it into a planner. We select a MPC-based planner and evaluate the collision rates under the attack. To perform the attack on a closed-loop planner, we conduct attacks on a sequence of frames with the expectation over transformation (EOT) [61] method. We follow the setting from *Zhang et al.* [190] and choose $\epsilon = 1$. We choose AgentFormer model since it has the most competitive performance. As a result, we observe that, while AgentFormer model trained on clean data leads to 10 collision cases under attack, the robust trained model with the proposed *RobustTraj* is able to avoid all the collisions. As shown in Fig. C.2, we demonstrate that the proposed *RobustTraj* is able to avoid the collisions (Fig. C.2 (d)) while the *DA + Train-time Smoothing* method proposed by *Zhang et al.* [190] is not (Fig. C.2 (c)).

5.11.3 Component analysis

In this section, we analyze the effectiveness of the three components. We use the mini-AgentFormer model since it has competitive performance and is lightweight for



(a) Benign case (b) Adv Attack (c) w/ defense [190] (d) w/ defense (ours)

Figure 5.8: Impacts to a MPC-basd downstream planner. (a) is under the benign case while (b), (c) and (d) are under the adversarial attacks. The blue car and the red car represent the AV and the adversarial agent respectively.

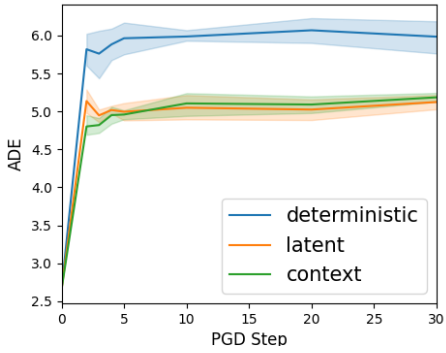


Figure 5.9: Peformance of different attacks in mini-AgentFormer.

a fast adversarial training process.

Effectiveness of the *Deterministic Attack*. To demonstrate the importance of the *Deterministic Attack*, we compare it with competitive alternatives, *Latent Attack* and *Context Attack*, which also construct the deterministic path. However, they only attack a partial model as opposed to our end-to-end full model attack. More details about these attacks are in the Appendix C. We evaluate their attack effectiveness by attacking a normally trained trajectory prediction model (without robust training). In Fig. 5.9, we demonstrate that *Deterministic Attack* is the most effective attack among all. Additionally, we embed them into the whole adversarial training pipeline and evaluate the adversarial robustness. The results are shown in Table 5.7. We observe that the model trained with the *Deterministic Attack* achieves

Table 5.7: ADE and robust ADE for different methods on mini-AgentFormer. The lowest error is in bold.

Method	ADE		Robust ADE	
	0.5	1.0	0.5	1.0
Clean	2.05	2.05	6.86	11.53
Latent Attack	2.55	2.70	4.10	4.71
Context Attack	**2.47**	2.59	3.94	4.78
Deterministic Attack	2.61	**2.55**	**3.88**	**4.35**
Deterministic Attack				
+ $\mathcal{L}_{\text{reg}}$	2.29	2.31	3.76	4.28
+ $\mathcal{L}_{\text{clean}}$ + $\mathcal{L}_{\text{reg}}$	2.23	2.19	3.71	3.83
+ $\mathcal{L}_{\text{clean}}$ + $\mathcal{L}_{\text{reg}}$ + Aug	**2.14**	**2.11**	**3.69**	**3.82**

the best robustness in terms of ADE. More results with other metrics and the other ϵ are in the Appendix C.

Effect of additional loss functions. We evaluate the performance of the models trained with additional loss terms: $\mathcal{L}_{\text{clean}}$ and $\mathcal{L}_{\text{reg}}$. In Table 5.7, we can see that the regularization term $\mathcal{L}_{\text{reg}}$ improves robustness of the models and achieves better clean performance. It shows that the regularization of the introduced noises on conditional variables help the model to stabilize the training procedure. By adding the clean loss $\mathcal{L}_{\text{clean}}$, we observe that both the robustness and clean performance are improved further, which means the benign data indeed anchors the model output on clean data distribution and provides a stable signal for the better robust training for generative models.

Effect of domain-specific augmentation. To demonstrate the effectiveness of the domain-specific augmentation, We also combine it with all of the above components to validate its effect. The results are shown in Table 5.7. We observe that it achieves a better performance on clean and adversarial data.

5.12 Limitations

In this work, we identified the challenges of applying adversarial training on trajectory prediction models based on probabilistic generative models since they could cope with the natural uncertainty of motion forecasting. Though the probabilistic generative model is the main-stream architecture for the trajectory prediction task, there are other architectures (e.g., LSTM [17, 172], flow-based method [122, 123] and RL-based method [50]) for generating multi-modal predictions. Additionally, we only study the adversarial set with the threat model of $\mathcal{L}_\infty$ perturbation on trajectories instead of other types of threat models (e.g., optimization on the latent space [118], perturbation on raw sensor data [146]). Moreover, the primary goal of this paper is to study and improve the robustness of trajectory prediction models. To obtain salient and unambiguous insights, we minimize the conflating factors in our analysis without considering the perception model in our pipeline. We leave these as future work for building robust trajectory prediction models.

5.13 Conclusion

In this paper, we study the adversarial robustness of trajectory prediction systems. In the first part, we present an attack framework to generate *realistic* adversarial trajectories via a carefully-designed differentiable dynamic model. We have shown that prediction models are generally vulnerable and certain model designs (e.g, modeling motion and social properties simultaneously) beneficial in benign settings may make a model more vulnerable to adversarial attacks. In addition, both motion (predicted future trajectory of adversarial agent) and social (predicted future trajectory of other agents) properties could be exploited by only manipulating the adversarial agent's history trajectories. We also show that prediction errors influence the downstream planning and control pipeline, leading to severe consequences such as collision.

In the second part, we aim to study how to train robust generative trajectory prediction models against adversarial attacks. To achieve this goal, we first identify three key challenges in designing an adversarial training framework to train robust trajectory prediction models. To address them, we propose an adversarial training framework with three main components, including (1) a *deterministic attack* for the inner maximization process of the adversarial training, (2) additional regularization terms for stable outer minimization of adversarial training, and (3) a domain-specific augmentation strategy to achieve a better performance trade-off on clean and adversarial data. To show the generality of our method, we apply our approach to two trajectory prediction models, including (1) a CVAE-based model, AgentFormer, and (2) a cGAN-based model, Social-GAN. Our extensive experiments show our method could significantly improve the robustness with a slight performance degradation on the clean data, compared to the existing techniques and dramatically reduce the severe collision rates when plugged into the AD stack with a planner. We hope our work can shed light on developing robust trajectory prediction systems for AD.

CHAPTER VI

Secure and Safe Autonomous Driving with Integrated Robustness

6.1 Introduction

Detecting the presence of surrounding agents and predicting their future behavior is a necessary capability for AD systems. In particular, there has been a significant interest in object detection and trajectory forecasting within the autonomous driving community, with many major organizations incorporating state-of-the-art perception and behavior prediction algorithms within their vehicle software stack. As a result, it is important to accurately evaluate the performance of detection and forecasting systems before their deployment. To date, accuracy-based metrics such as IoU, ADE/FDE, and NLL are commonly used to evaluate the performance of object detection and trajectory forecasting algorithms in autonomous driving systems. These metrics compare the predictions made by a model with ground truth data and produce a value that indicates how similar the two are. For example, IoU compares the overlap between a predicted bounding box and a ground truth bounding box, while ADE and FDE measure the distance between a predicted trajectory and a ground truth trajectory.

While accuracy-based metrics can be useful for comparing the performance of

different models and for identifying areas for improvement, they may not always be sufficient for evaluating the performance of a system in a real-world setting. This is because accuracy-based metrics do not take into account other important factors such as safety, reliability, and robustness. For example, even if a model has high accuracy in terms of ADE and FDE, it may still produce unsafe or unreliable predictions if it fails to consider other important factors such as road conditions, weather, and traffic rules. Therefore, it is important to consider a wide range of metrics when evaluating the performance of object detection and trajectory forecasting algorithms in autonomous driving systems. This may include not only accuracy-based metrics, but also safety-related metrics such as collision rate, reliability metrics such as the frequency of system failures, and robustness metrics such as the ability of the system to handle unusual or unexpected situations.

Despite the evaluation metrics, evaluating the integrated performance of the system is also important due to the compounding error of the AD system pipeline. This is because the various components of an autonomous driving system are typically interconnected and rely on each other to function properly. As a result, the performance of one component can have a cascading effect on the performance of other components, leading to a compounding of errors. For example, if the object detection algorithm produces inaccurate detections, this can lead to errors in the trajectory forecasting algorithm, which in turn can result in unsafe or unreliable control decisions made by the control system. Therefore, it is important to evaluate the integrated performance of the entire autonomous driving system to ensure that it is operating safely and reliably. This may involve testing the system in a variety of different scenarios and environments to ensure that it can handle a wide range of conditions.

Guided by this intuition, we propose a simplified AD framework for testing the integrated robustness of each component, **S**ecure and **S**afe **A**utonomous **D**riving system (**S^{2}AD**). **S^{2}AD** consists of four main components for a simplified AD functionality:

a perception system, a tracking system, a prediction system, and a planning system. Enabled by open-sourced models and systems, it follows a modular design where it allows the users to test their models/systems. Different from adversarial training-based methods for improving modular robustness, we proposed a system-level approach to detect attacks and protect the AD system from such attacks. In summary, this work makes the following contributions:

- We propose a simplified AD framework for testing the integrated robustness of each component, with a modular design that extends to different models/systems.
- We propose **S^{2}AD** incorporating an anomaly detection system to detect LiDAR spoofing attacks and a fail-safe object detector to defend against object-removing attacks.
- We demonstrate the improvement of the system by augmenting it with simple components tailored for the realistic attack threat models, not only under the attack but also in normal driving scenarios.

6.2 The S^{2}AD Approach

In this section, we will first present the simplified AD pipeline design. Then, we will demonstrate the anomaly detection and fail-safe object detection system for defending against sensor spoofing attacks and object removing attacks.

6.2.1 Simplified AD Pipeline

To measure the integrated robustness of an AD system, we build a simplified AD pipeline that drives based on sensor data. The goal of the framework is to provide a framework for evaluating the integrated robustness of different models/systems. Therefore, the framework has two key requirements: being extensible to different

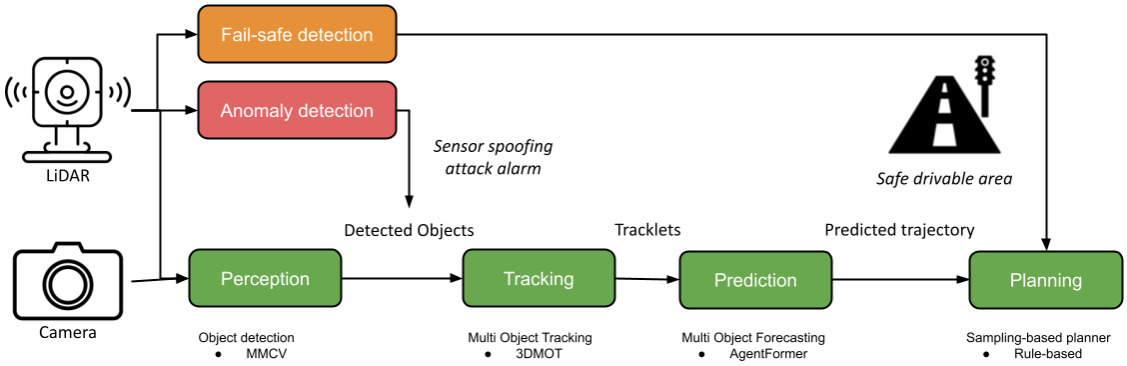


Figure 6.1: Overview of $\mathbf{S^2AD}$.

models/systems and being simple to adapt to different systems. To achieve these goals, we build the simplified AD pipeline with existing benchmarks on different AD tasks such as perception, tracking, and prediction, as shown in Figure 6.1. More specifically, we leverage MMCV [44] for inferring different perception models; we use 3DMOT [150] for tracking the detected bounding boxes; we leverage different open-sourced trajectory prediction models [185, 126] for prediction road agents future trajectory; we implement a simple sampling-based planner for planning ego vehicles future path.

6.2.2 Anomaly Detection with Point Cloud Prediction

To detect the sensor spoofing attacks, we propose an anomaly detection framework by forecasting the point cloud data for the incoming frame. Compared to the existing defense that leverages anomaly detection at the detected object level, forecasting point cloud and detecting anomalies at the point cloud level is beneficial in two folds. First, forecasting at a lower level (i.e., point cloud instead of detected object) makes adaptive attacks more challenging, or even impossible to conduct due to the limited precision of LiDAR sensor spoofing attacks. Second, forecasting point cloud does not require detected objects to start with, which limited the usage of the anomaly detector. It requires an object detector for the target classes to detect sensor spoofing

attacks. This might be not extensible for different perception systems (e.g., occupancy networks). To forecast the point cloud, we leverage a model-free neural scene flow estimation network [90].

6.2.3 Fail-safe Detection with Clustering-based Method

While deep learning models are powerful in terms of predicting useful semantic information such as bounding boxes and object categories, they are less explainable and vulnerable to adversarial attacks. On the other hand, traditional methods such as clustering-based methods are more robust to adversarial perturbations and interpretable, at the cost of limited predicted semantic information. Hence, we propose to use the clustering-based method as a fail-safe mechanism to avoid false negatives of the object detectors. Leveraging the clustering-based method, we are able to build a safe drivable space where no obstacles are on the road. By combining such information with the output of the prediction system, the planner is able to make driving decisions safer and more robust against adversarial attacks.

6.3 Evaluation

In this section, we will present the experimental setup of the evaluation of **S^{2}AD** and the results.

6.3.1 Experiment Setup

Dataset. We evaluate the system on nuScenes [23] dataset. To evaluate the robustness of the proposed components, we build poisoned datasets based on two attacks: LiDAR sensor spoofing attacks and object reshaping attacks. For the LiDAR sensor spoofing attacks, we follow the way You et al. [177] did but extend the near front position to three positions in front of the ego vehicle. We name this as the *spoofing dataset*. For the object reshaping attacks, we consider a stronger attack than the

	Perception AP	Collision (%)	L2 human
Clean dataset			
Simplified AV	78.6	1.7	3.72
+ Fail-safe detection	-	0.2	3.69
Spoofing dataset			
Simplified AV	48.3	2.1	4.15
+ Fail-safe detection	-	0.4	4.12
Removing dataset			
Simplified AV	-	13.6	5.29
+ Fail-safe detection	-	0.3	3.82

Table 6.1: Evaluation results of **S^{2}AD** on clean dataset and poisoned datasets.

existing proposed MSF-adv where the attacker's goal is to create an additional invisible object on the road. Instead, we randomly remove three obstacles at the near front positions (4-8m) from the perception system outputs. And we name this as the *removing dataset*.

Models. We evaluate a set of state-of-the-art models: pointpillars [86] for the perception system, MOT for the tracking system, AgentFormer for the prediction system.

6.3.2 Results

As shown in Table 6.1, we evaluate the simplified AD and the proposed **S^{2}AD** on three datasets with three different metrics including AP (Average precision) for the perception system, collision rate and L2 distance to ground truth planning trajectory for the planning system. We demonstrated that with the additional fail-safe detection component, **S^{2}AD** is able to reduce the collision rates, especially under the removing poisoned dataset. By choosing 0.78 as the threshold, the anomaly detection achieves 93%precision with 0% recall on the spoofing dataset.

6.4 Limitations and Future Works

In this work, we propose to evaluate the integrated robustness of AD systems by prototyping a simplified AD pipeline. In addition, to enhance the integrated robustness, we propose to add two components: a fail-safe detector and an anomaly detector. We have demonstrated the improved robustness of the proposed framework in terms of different evaluation metrics. However, this work serves more as a preliminary study in this direction and can be further improved from two perspectives.

First, in terms of the breadth of the system, we didn't prototype all the components on a modern autonomous vehicle (e.g., localization system, multi-sensor fusion, etc.). The system can be further extended with additional components to evaluate its integrated robustness once those components are compromised. Also, in this work, we only consider one dataset for simplicity, other datasets or even closed-loop simulations are also possible.

Second, in terms of the depth of the system, though the proposed additional components mitigate the attack impacts, the system can be designed to be more integrated. Recent works on integrated AD stacks have shown a promising future for building safer and more robust AD systems.

In terms of the anomaly detection system, Xiao et al. have also demonstrated advancements in robustness against adversarial attacks, using not only temporal consistency check but also spatial consistency check [161]. The idea of spatial consistency check could be further applied to the anomaly detection module.

6.5 Conclusion

In this work, we propose to evaluate the integrated robustness of autonomous driving systems. To do so, we prototype and implement an extensible and lightweight autonomous driving pipeline. To improve the robustness of the autonomous driving

system, we propose to add two key components: a fail-safe detector for building safe drivable space and an anomaly detector for detecting spoofing attacks. As the result, we demonstrated that with the integrated mindset, simple components can effectively improve the robustness of the autonomous driving system against adversarial attacks.

CHAPTER VII

Conclusion and Future Work

7.1 Conclusion

Testing autonomous driving systems are difficult, due to the nature of rick semantics in the real world. To ensure safety and security, an effective tool for stress testing such systems is in emergent demand. In this dissertation, we illustrated that adversarial methods can provide an efficient yet effective proxy to stress test the system, expose the design limitations and provide insights for improving robustness. More specifically, we propose a foundational framework, based on realistic adversarial attacks, for measuring the vulnerability status, unveiling the limitation of the system/model architecture designs, and improving the robustness of the system in return.

This foundational framework starts with building realistic adversarial attacks by formulating feasible threat models, or attacker capabilities. With the threat models of adversaries conducting sensor spoofing attacks or placing physical objects on the road, we propose *Adv-LiDAR* and *LiDAR-Adv* respectively, which achieve different attack goals (i.e., spoofing an additional obstacle or creating an invisible obstacle on the road). By formulating the attacker capabilities and differentiating the AD components, we can automatically generate effective adversarial examples that fail the AD system.

Beyond generating attacks, this framework can also be used for causality analysis. With the **RAP** attack, we evaluate the robustness of different model architectures for the semantic segmentation task. By associating the robustness with the model architectures, we discover the correlation between the adversarial robustness and the receptive field sizes. Just like good test cases revealing the design limitations of a system, examples generated from adversarial attacks can uncover the vulnerability causalities as well [136, 65].

The last part of this dissertation provides guidelines for improving robustness toward secure and safe AD systems, from modular and integrated perspectives. We first demonstrated how adversarial training is a good fit for improving the modular robustness of the trajectory prediction models. Then we show how to adapt the adversarial training to this task by addressing key challenges discovered by conducting adversarial attacks on them. For the integrated robustness, we aim to show that modular robustness is sometimes insufficient for safe AD and system-level designs can easily improve the integrated robustness while improving the modular robustness is infeasible.

7.2 Future Work

Following my dissertation research, there is still a lot more to be explored in the future.

- **Automating the extraction of attack capabilities.** In our works, most attack capabilities require domain knowledge and manual efforts to formulate. Though adversarial examples can be generated automatically once the attack capabilities are formulated, those attack capabilities are usually restricted. It would be more efficient if such attack capabilities can be automatically extracted given a system or with a few examples.

- **Integrated robustness with integrated AD system.** In the last chapter, we presented **S^{2}AD** where simple modifications improve the robustness of the system greatly. To further extend the success, designing AD systems where each component is more integrated can provide better-integrated robustness. For example, propagating the uncertainty along each module could aid the planning system to better judge which information to rely upon instead of blindly relying on the information received from the upstream modules.

APPENDICES

APPENDIX A

Vulnerability Status of LiDAR-based Perception against the Sensor Spoofing Attack

A.1 Algorithm Details and Experiment Settings

Algorithm 16 shows the detailed algorithm to generate adversarial examples. In our experiment, we select Adam [80] as our optimizer *opt* with learning rate $1e-4$. $opt(l_{\text{adv}}; \theta, \tau_x, s_h)$ means updating the parameters $\theta, \tau_x, s_h)$ with respect to Loss function l_{adv}. We select TensorFlow [15] as backbone. L_t is set as 12.5 while L_θ is set as the angle that generates 2-meter distance from the target position.

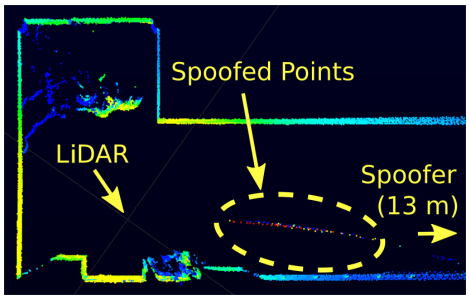


Figure A.1: Collected traces from the reproduced sensor attack. The points in the yellow circle are spoofed by the sensor attack.

input: Target model: $\boldsymbol{M}$;
3D point cloud $\boldsymbol{X}$;
3D spoofed 3D point cloud $\boldsymbol{T}$;
Optimizer ***opt***;
Max iteration $\boldsymbol{N}$;
output: 3D adversarial 3D point cloud $\boldsymbol{X'}$;

Initialization: $\theta \leftarrow 0$, $\tau_x \leftarrow 0$, $s_h \leftarrow 1$, $l_{min} = +\text{inf}$, $x = \Phi(X), t = \Phi(T)$;
/* Initiate parameters by sampling around the transformation parameters $Target_\theta$, $Target_{\tau_x}$ that transforms t to the target position (px, py) of the attack */
for $i\tau_x \leftarrow -L_t$ **to** L_t **do**
  for $i\theta \leftarrow -L_\theta$ **to** L_θ **do**
    /* Initialize parameter . */;
    $\theta \leftarrow Target_\theta + i\theta$, $\tau_x \leftarrow Target_{\tau_x} + \tau_x i$;
    for $iter \leftarrow 1$ **to** N **do**
      /* Calculate adversarial loss */;
      $l_{\text{adv}} \leftarrow$ **Equation** 2.7.;
      /* Update the parameters θ, τ_x, s_h based on optimizer opt and loss l_{adv} */
      ;
      $\theta, \tau_x, s_h \leftarrow opt(l_{\text{adv}}; \theta, \tau_x, s_h)$
      if $l_{min} < l_{adv}$ **then**
        $\theta^{final}, \tau_x^{final}, s_h^{final} \leftarrow \theta, \tau_x, s_h$
    end
  end
end
$T' \leftarrow G_T(\theta^{final}, t_x^{final}, s_h^{final}; T)$;
$X' \leftarrow X + T'$;
Return: T'

APPENDIX B

AdvDO: Realistic Adversarial Attacks for Trajectory Prediction

B.1 Related works

Adversarial Traffic Scenarios Generation In Strive [117], adversarial scenarios generated from the traffic model is not always realistic due to the limited training data which does not cover dangerous scenarios such as collisions. In Figure B.1, we demonstrated from one example generated in Strive, where the adversarial agent drives in reverse lane and violates the traffic rule, in order to collide into the AV.

B.2 Method

In this section, we describe implementation and formulation details for the proposed method.

B.2.1 Differential dynamic model

The differential dynamic model Φ is devised for deriving dynamic parameters $\{p, v, \theta\}$ from control actions $u = \{a, \kappa\}$ and deriving control actions from trajectories

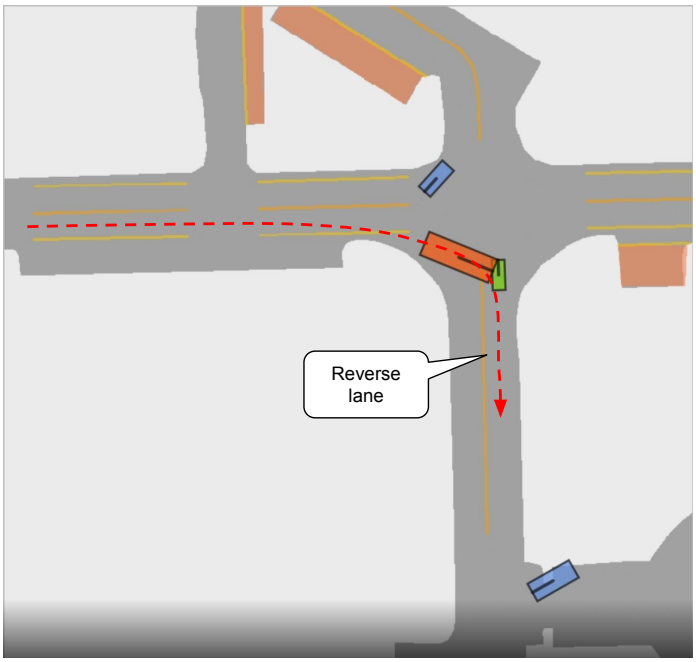


Figure B.1: Adversarial agent drives in reverse lane in adversarial scenarios generatated from Strive [117].

$p = (p_x, p_y)$. Specifically, we use a kinematic bicycle model as the dynamic model [43]. Detailed formulation is as below:

$$
\begin{aligned}
\Phi: \quad & v^{t+1} = a^t \cdot \Delta t + v^t \\
& d\theta^t = v^t \cdot \kappa^t \\
& \theta^{t+1} = d\theta^t \cdot \Delta t + \theta^t \\
& p_x^{t+1} = v^t \cdot \cos\theta^t \cdot \Delta t + p^t \\
& p_y^{t+1} = v^t \cdot \sin\theta^t \cdot \Delta t + p^t \\
\Phi^{-1}: \quad & v^t = \|p^{t+1} - p^t\| / \Delta t \\
& \theta^t = \arctan p_x^t / p_y^t \\
& a^t = (v^{t+1} - v^t) / \Delta t \\
& \kappa^t = d\theta^t / v^t
\end{aligned}
$$

For the physical constraints for dynamically feasibility, we follow the standard values used in [146].

B.2.2 Reconstruction loss and adversarial loss

Here, we describe losses for reconstruction and generating adversarial trajectory in details:

$$l_{\text{dyn}}(\theta, v, a, \kappa) = \sum_{x=\theta,v,a,\kappa} (x - x_{\text{lb}})/(x_{\text{ub}} - x_{\text{lb}}) - \text{Sigmoid}\left((x - x_{\text{lb}})/(x_{\text{ub}} - x_{\text{lb}})\right) + 0.5$$

, where x_{ub}, x_{lb} represent the hard-coded upper bound and lower bound correspondingly for the dynamic parameter x.

$$l_{\text{col}}(\mathbf{D}_{\text{adv}}, \mathbf{X}) = \frac{1}{n-1} \sum_{i \neq adv}^{n-1} \frac{1}{\|\mathbf{D}_{\text{adv}} - \mathbf{X}_i\| + 1}$$

, where n is the number of agent in the current prediction time frame.

$$l_{\text{bh}}(\mathbf{D}_{\text{adv}}, \mathbf{D*}_{\text{orig}}, \epsilon) = \|\mathbf{D}_{\text{adv}} - \mathbf{D*}_{\text{orig}}\|/\epsilon - \text{Sigmoid}\left(\|\mathbf{D}_{\text{adv}} - \mathbf{D*}_{\text{orig}}\|/\epsilon\right) + 0.5$$

, where ϵ is the tolerance for position deviation, which we empirically set to half lane width (1 meter).

$$l_{\text{obj}} = \frac{1}{T} \sum_{t=1\ldots T} \| \mathbf{Y^t} - \hat{\mathbf{Y}}^{\mathbf{t}} \|_2$$

, where $\hat{\mathbf{Y}}^t$ is the predicted future trajectory at time t given the adversarial trajectory and $\mathbf{Y}^t$ is the corresponding ground truth. This loss aims to mislead the prediction by maximizing the difference between the predicted future trajectory and ground truth.

B.3 Experiments

In this section, we describe implementation and formulation details for the experiments.

B.3.1 Attack fidelity analysis

In this analysis, we aim to demonstrate the generated adversarial trajectory is realistic from both perspectives of: (1) dynamically feasibility and (2) similar behavior as the original history trajectory. For the first perspective, we demonstrate the results quantitatively with the **Violation Rates** (VR) metric described below. For the second perspective, since it is a common challenge to measure the behavior change quantitatively, we propose to approximate the degree of behavior change with the **Aggregated sensitivity** metric described below. We also visually examine generated adversarial trajectories in Figure B.2.

Violation rates. Since the violation rates metric is only suitable for the *search* method and on the curvature κ parameter, we represent the VR as:

$$VR = \frac{\#\text{total adv trajectories}}{\#\text{adv trajectories violating curvature constraints}}$$

.

Aggregated sensitivity. To approximate the behavior change quantitatively, we leverage the sensitivity concept proposed by Ivanovic et al. [76]. Sensitivity $\mathbf{PI}(\mathbf{Y}_i, \mathbf{Y}_{\text{ego}})$ of an agent's trajectory to the ego agent represents how much the agent's trajectory $\mathbf{Y}_i$ will affect the ego planning $\mathbf{Y}_{\text{ego}}$. Therefore, we can present how much the adversarial trajectory $\mathbf{X}_{\text{adv}}$ will affect other agents' planning $\mathbf{X}_i$ as the aggregated sensitivity of the adversarial agent's trajectory to all the other agents in the scene. With a

Table B.1: Similarity between original history trajectory and adversarial trajectory generated from *search*, *Opt-init* and *Opt-end*.

Attack method	DTW↓	FD↓	PCM↓	Area↓	CL↓
search	0.3558	0.2490	0.0676	0.8892	0.0003
Opt-init	0.2303	0.1429	0.0209	0.5928	0.0002
Opt-end	0.1891	0.0564	0.0210	0.3045	0.0001

Table B.2: Augmentation on AgentFormer.

	ADE	FDE
Benign	1.83	3.81
+ *aug*	1.69	3.57

normalization over agents nearby, we attain the aggregated sensitivity:

$$\Sigma\text{Sensitivity}(\mathbf{X}_{\text{adv}}, \mathbf{X}) = \frac{1}{m} \sum_{i, \|\mathbf{X}_{\text{adv}} - \mathbf{X}_i\| < \rho}^{m} \mathbf{PI}(\mathbf{X}_{\text{adv}}, \mathbf{X}_i)$$

, where m represents the total number of agents nearby filtered by the distance threshold ρ, which is empirically set to 5 meters. Therefore, we attain the metric for measuring behavior change as:

$$\Delta\text{Sensitivity} = \Sigma\text{Sensitivity}(\mathbf{X}_{\text{adv}}, \mathbf{X}) - \Sigma\text{Sensitivity}(\mathbf{X}_{\text{orig}}, \mathbf{X})$$

Other metrics. To measure the behavior change quantitatively, we also include evaluation results with other metrics proposed by Jekel et al. for comparing the similarity between trajectories [79], including Dynamic Time Warping (DTW), Fréchet Distance (FD), Partial Curve Mapping (PCM), Area and Curve Length (CL). In Table B.1 we demonstrate that the proposed methods have lowest error for all similarity metrics. The results are also consistent with the result on Δsensitivity metric.

Visualization and human study. We randomly sample examples from 150 scenes in nuScene validation data, where the adversarial trajectory generated from *search*

that have a curvature violation or a large ΔSensitivity value. In Figure B.2, we show that the adversarial trajectory generated from *search* have either behavior change or unrealistic steering rates. We also notice that, the *Opt-end* can also generate adversarial trajectory that has large turning rates but dynamically feasible. Even though the predicted results are worse under *search* attack when the curvature constraint if not bounded, *Opt-end* achieves higher prediction errors in average scenarios. To further show that the generated trajectories obey traffic rules, we conduct a study where adversarial trajectories are illustrated with map information (e.g. lane segments, road, crosswalk etc.). We select five human subjects with driver license and show our generated trajectories to them. Out of the 50 trajectories evaluated, only 2.2(±1.3) are considered rule-violating. We conclude that the adversarial trajectory generated by our methods are more realistic in both perspectives of dynamical feasibility and behavior changing.

AdvDO as Augmentation. Noticed that AdvDO also provides an opportunity for generating realistic trajectories as additional data. We replace the adversarial objective with other objectives (e.g. increasing left/right/forward/backward deviations) and generate additional data. More specifically, the objective function consists of two components: $\mathcal{L}_{\text{dyn}} = \mathcal{L}_{\text{d}} + \gamma\mathcal{L}_{\text{col}}$, where $\mathcal{L}_{\text{d}}$ is the deviation objective loss, $\mathcal{L}_{\text{col}}$ is the collision regularization loss, and γ is a weight factor to balance the objectives. In each scene, we randomly pick a deviation objective loss $\mathcal{L}_{\text{d}}$ from the set {moving forward, backward, left, right} for each agent. More specifically, the deviation objective loss $\mathcal{L}_{\text{d}}$ is formulated as

$$\mathcal{L}_{\text{d}} = (\mathbf{X} - \mathbf{X}_{\text{aug}})\,\bar{\mathbf{d}},$$

where $\mathbf{X}_{\text{aug}}$ represents the generated trajectories by perturbing the trajectories in the dataset and $\bar{\mathbf{d}}$ represents the unit vectors for the target deviation directions in the set of {moving forward, backward, left, right}. In Table B.2, we demonstrate that the augmented data improves the clean performance by 9% on ADE. This further

validates that the high fidelity of the generated trajectories with the proposed method.

B.3.2 Case studies with planners

Planner. In this work, to demonstrate the explicit consequences of the adversarial trajectory, we implement two planners (including path planning and motion planning). The first one is a rule-based planner as implemented by Rempe et al. [117]. However, we notice that this planner is enforcing path planning along the center of lane lines which leads to insufficient path sampling through the simulations. Therefore, though the planner naturally avoids driving off road, it is also lack of flexibility to dodge incoming traffic. To better represent planners equipped on AV, we implement a simple yet effective planner that uses conformal lattice [101] for sampling paths and model predictive control (MPC) [24] for motion planning. We call this planner MPC-based planner.

Planning strategy. In this work, we consider both an open-loop and a closed-loop planning strategy. Though for the closed-loop planning we have to replay the ground truth trajectories of other agents, we do notice reduced collisions and driving off road consequences and consider the closed-loop planning fashion meaningful.

B.3.3 Transferability Analysis

In this section, we aim to analyze the transferability of adversarial trajectories generated on a source model to a unseen target model. We measure the transferability by devising the **transfer rate** metric. High transfer rates indicate that the feasibility of transfer attack, which is a more realistic black-box attack, in the real-world scenario. Transfer rate is defined as the success degree of adversarial trajectories on target model over the success degree of them on source model. The success degree is measured by the average percentage of increased error (on metrics ADE/FDE/MR/ORR) with transfer attack on the target models over the increased error with white-box at-

tack on the source models.

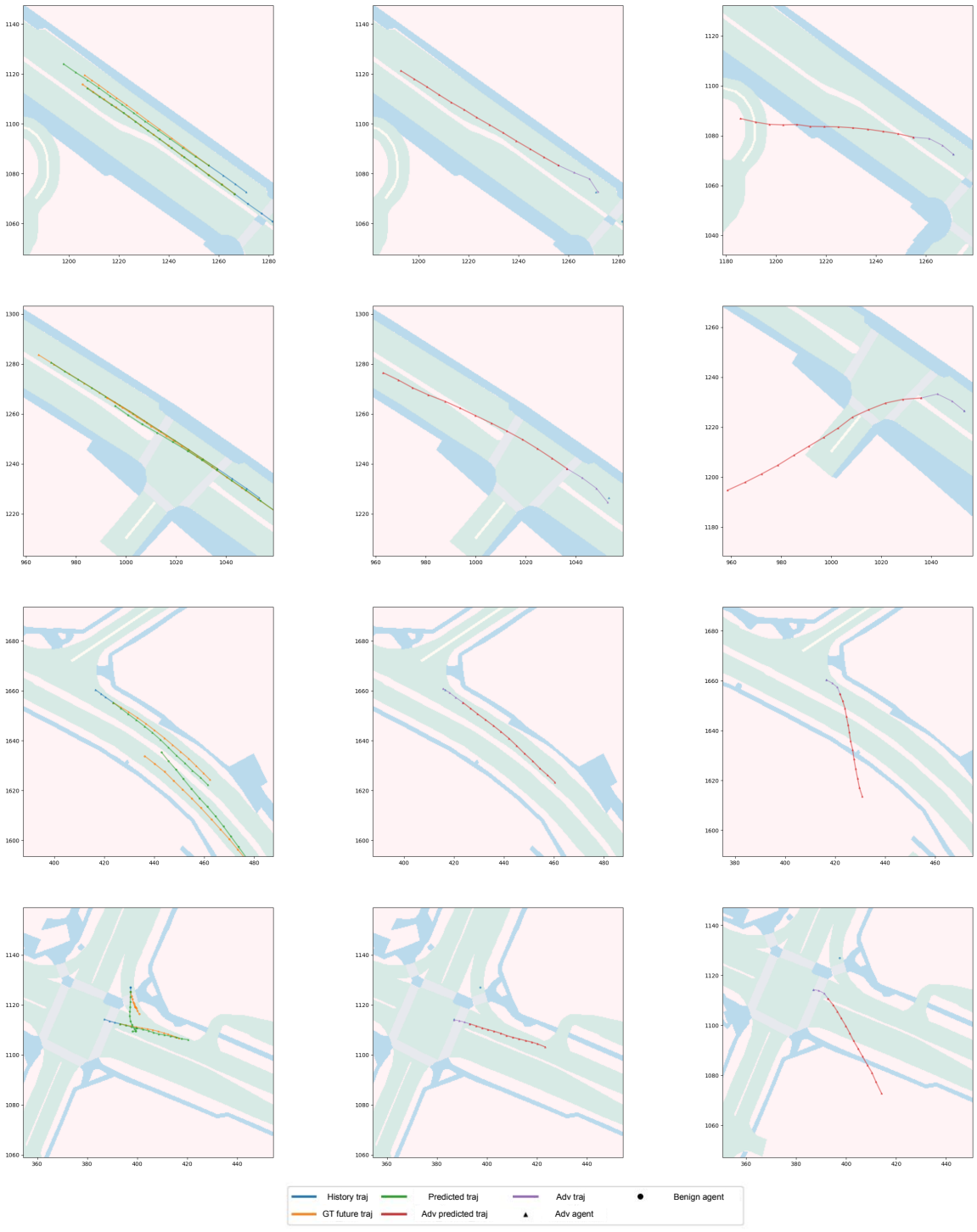
B.3.4 Ablation Study

We explore the attack results in different traffic scenarios with different speeds curvatures. We calculate the aggregated speed and curvature for each agent in the entire scene to represent the speed and curvature for that scene. Similarly, we calculate the aggregated *Miss Rates* to evaluate performance.

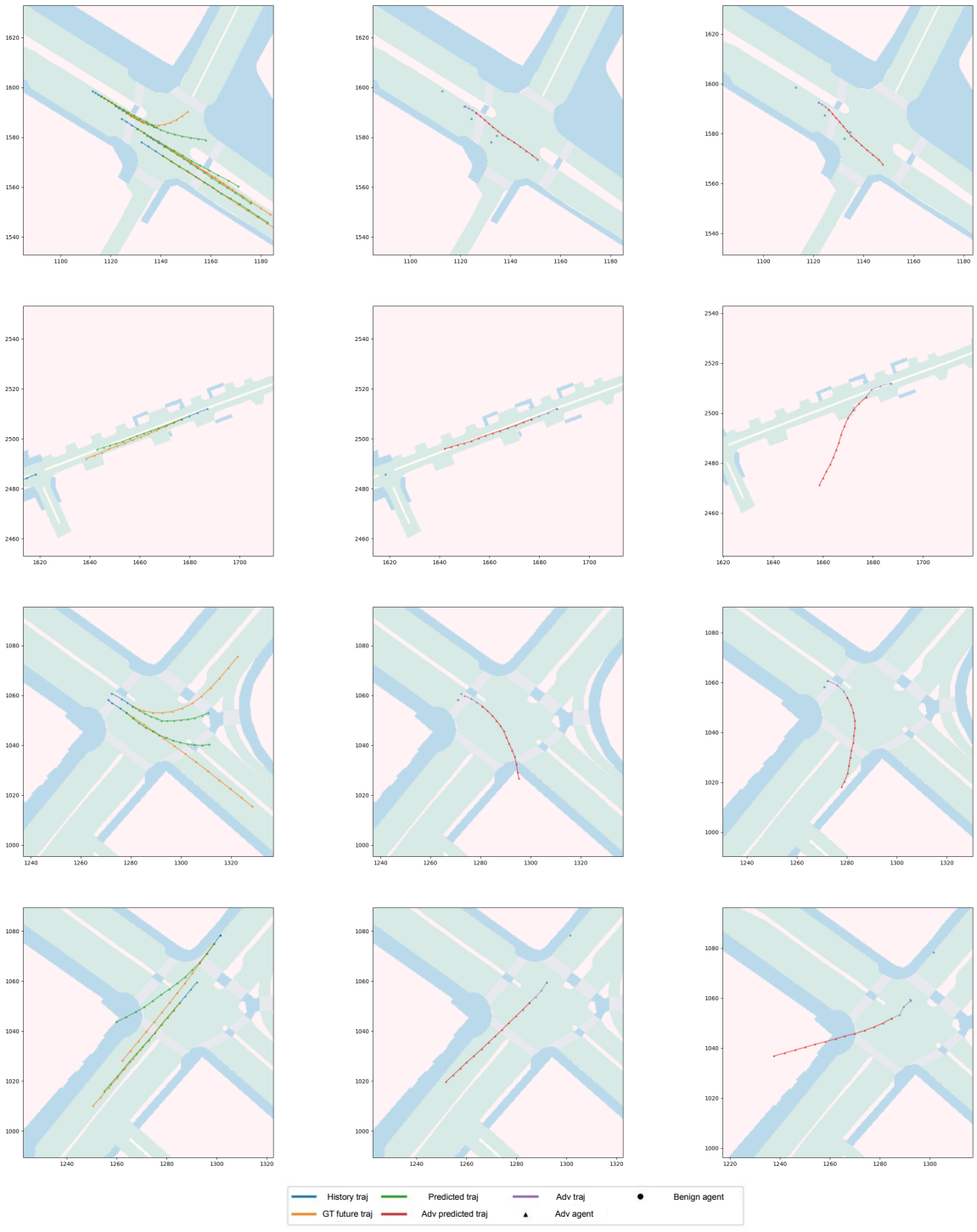
Attack effectiveness with different speeds As shown in Figure B.3, the higher speed traffic show higher *Miss Rates*. It is reasonable since position deviations are larger in high speed traffics. We also notice that the attack results are consistent to results in Table 1&2 in the main paper, which means different attack methods are not restricted due to the speed constraints.

Attack effectiveness with different curvatures In Figure B.4, we notice that adversarial trajectories are more effective in small curvature traffics. This is reasonable since small curvature traffics allow more flexible adversarial trajectory generations. We find that *Opt-end* performs better than *Opt-init* in small curvature traffic. This could be due to low curvature traffic being less sensitive to current positions.

GT & Benign Prediction *Opt-end search*



GT & Benign Prediction *Opt-end search*



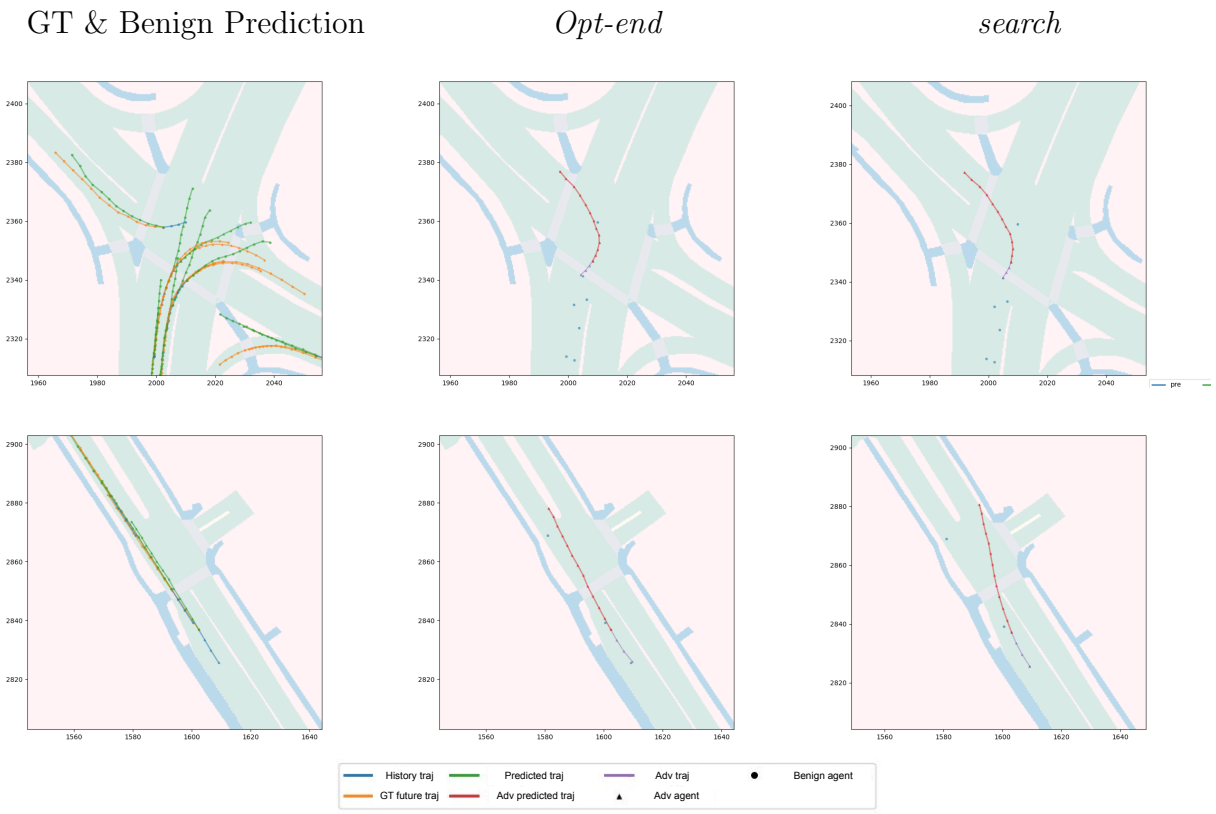


Figure B.2: Visualization examples of generated adversarial trajectories from *Opt-end* and *search*. We only show the adversarial agent's trajectory in the attack scenario for clearer visualization.

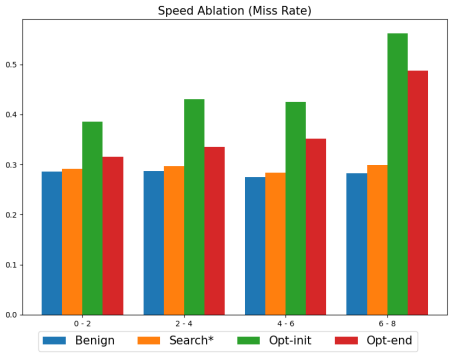


Figure B.3: Speed ablation

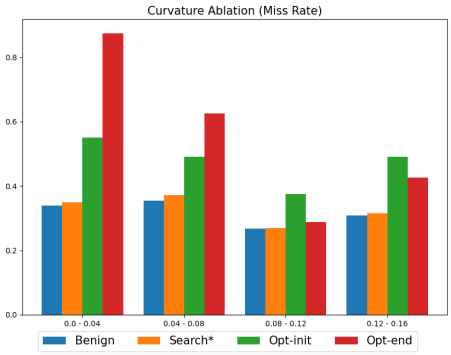


Figure B.4: Curvature ablation

APPENDIX C

Robust Trajectory Prediction against Adversarial Attacks

C.1 Method and Implementations

C.1.1 Adversarial Attack on Trajectory Prediction

***Latent Attack* and *Context Attack*.** Noticed that, besides *Deterministic Attack* introduced in the main paper, there are also two other less intuitive attacks. Since the prediction $\hat{\mathbf{Y}}$ is dependent on posterior distribution $q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{Y})$ and conditional variable $f(\mathbf{X})$, we can construct attacks based on that. *Latent attack* aims to increase the error of estimating $q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{Y})$, which is formulated as:

$$\delta = \arg\max_{\delta} \mathrm{KL}(\, q_\phi(\mathbf{Z}|\mathbf{Y},\mathbf{X}) \parallel q_\phi(\mathbf{Z}|\mathbf{Y},\mathbf{X}+\delta)\,). \tag{C.1}$$

Context attack aims to increase the error of encoding the conditional variable $f(Z)$, which is formulated as:

$$\delta = \arg\max_{\delta} \mathrm{d}(\, f(\mathbf{X}),\ f(\mathbf{X}+\delta)\,), \tag{C.2}$$

where d is a distance function (e.g., L_2 norm). However, *latent attack* and *context attack* are effective due to two reasons. First, they are exploiting the vulnerability of a partial model. For example, *latent attack* only exploits the posterior estimation $q_\phi(\mathbf{Z}|\mathbf{X}, \mathbf{Y})$ and *context attack* only exploits the conditional encoder $f(\mathbf{X})$. Second, these attacks aim for a different goal. For the *latent attack* and *context attack*, the objectives are set for finding adversarial perturbations that maximize the difference of generated posterior distribution/context given $\mathbf{X}$ and $\mathbf{X}+\delta$, due to lacking ground truth for intermediate latent variables. However, for the sample attack, the objective is directly set for maximizing the prediction errors from the ground truth (future trajectories), which is more effective.

Adversarial attack on consecutive frames. In order to fool a planner in a closed-loop manner to make consistent wrong decisions, we need to conduct adversarial attacks on consecutive frames. To attack L_p consecutive frames of predictions, we aim to generate the adversarial trajectory of length $H + L_p$ that uniformly misleads the prediction at each time frame. To achieve this goal, we can easily extend the formulation for attacking single-step predictions to attack a sequence of predictions, which is useful for attacking a sequence of decision made by AV planning module. Concretely, to generate the adversarial trajectories for L_p consecutive steps of predictions, we aggregate the adversarial losses over these frames. The objective for attacking a length of $H + L_p$ trajectory is:

$$\sum_{t\in[-L_p,\dots 0]} \mathcal{L}_{\text{adv}}(\mathbf{X}_{\text{adv}}(t), \mathbf{Y}(t)), \tag{C.3}$$

where $\mathbf{X}_{\text{adv}}(t), \mathbf{Y}(t)$ are the corresponding $\mathbf{X}+\delta, \mathbf{Y}$ at time frame t.

C.1.2 Adversarial Training on Generative Models

Challenges. One challenge that hinders the adversarial training process is the noisy conditional data distribution disturbing the training process. One hypothesis we mentioned in the main paper is, the context encoding can magnify the bounded perturbation δ on history trajectory $\mathbf{X}$ to an unbounded perturbation on the conditional variable $\mathbf{C} = f(\mathbf{X} + \delta)$, during the training process.

Lemma C.1. *For a neural network f which is not bounded on Lipschitz constant during the training procedure, given any constant η and an input $\mathbf{X}$, there exists a pair (δ, f), that satisfies*

$$\| f(\mathbf{X} + \delta) - f(\mathbf{X}) \| \geq \eta.$$

Lemma C.1 can be easily derived by the definition of Lipshitz constant. Lipshitz constant L is defined as

$$L := \sup \frac{\| f(\mathbf{X} + \delta) - f(\mathbf{X}) \|}{\| \delta \|}.$$

If L is not bounded, $\frac{\|f(\mathbf{X}+\delta)-f(\mathbf{X})\|}{\|\delta\|}$ is not bounded and so is $\| f(\mathbf{X} + \delta) - f(\mathbf{X}) \|$ given a bounded δ. This means that, a bounded perturbation δ can potentially be magnified to be noisier on encoded conditional variable $\mathbf{C} = f(\mathbf{X} + \delta)$.

Analysis. In order to provide a quantitative analysis of the degeneration degree from conditional generative model to generative model (e.g., CVAE to VAE) with respect to the noise level, we propose a method to estimate the correlation between the degeneration and the noise level. Specifically, we trained a classifier with a 2-layer CNN achieving 99% accuracy on MNIST dataset. Then, given a conditional variable (the upper left quarter of an image of digit y), we generate images with the conditional generative models and use the classifier to calculate the confidence of the generated images labeled as digit y. We calculate the average confidence of 10

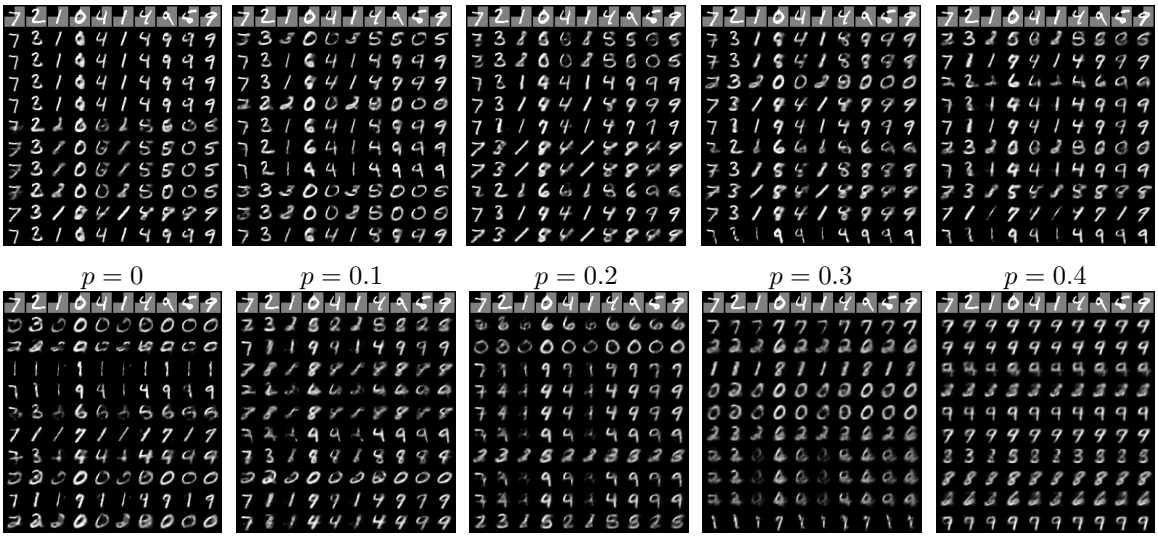
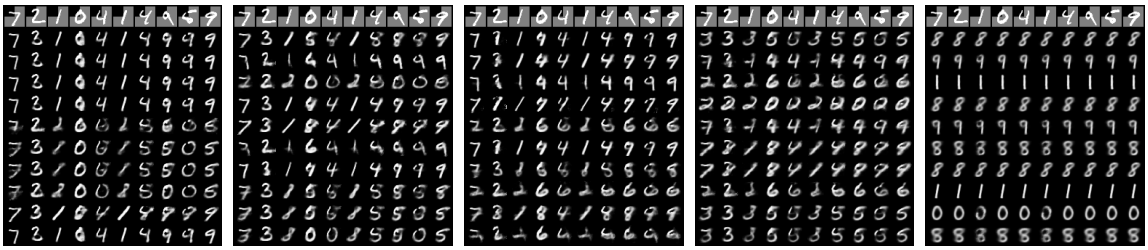


Figure C.1: Visual examples of images generated from models trained with different levels of salt and pepper noises.

generated images on all 10,000 images in the MNIST test data set. The lower the score means the weaker the correlation between the generated images with the given conditions, or the stronger correlation between the degeneration and the noise level. We also provide visualization examples of generated images from models trained using data with different level of noises in Figure C.1 and Figure C.2. We can see that, with the noise level increases, the generated images are less dependent on the conditional variable (i.e., not being the same digit). In the extreme case of high level noise, for example when $p = 0.9$, the model generates images solely depends on the random prior value it samples and generates the similar images for each row. We can also see that, the adversarial noises are more effective compared to the salt and pepper noise. With a small amount of noise (i.e., $\epsilon = 0.1$), it can degenerate the conditional generative model to a generative model (i.e., CVAE to VAE here).



$\epsilon = 0$ $\epsilon = 0.01$ $\epsilon = 0.03$ $\epsilon = 0.05$ $\epsilon = 0.1$

Figure C.2: Visual examples of images generated from models trained with different levels of adversarial noises.

C.1.3 Data Augmentation with Dynamic Model

For the data augmentation strategy $\mathbb{A}$, we use a kinematic bicycle model [113] as our dynamic model to generate realistic trajectories that can be driven in the real world. Representing the behavior of actors as kinematic bicycle model trajectories allows for physical feasibility and fine-grained behavior control. To generate realistic trajectories, we first parameterize the trajectory $\mathbf{S} = \{s^t\}_0^T$ as a sequence of kinematic bicycle model states $s^t = \{p^t, \kappa^t, a^t\}$, where p represents the position, κ represents the trajectory curvature, and a represents the acceleration. Then, trajectories can be generated by controlling the change of curvature $\dot{\kappa}_t$ and the acceleration a_t over time, and using the kinematic bicycle model to update corresponding other states for each timestamp. To generate diverse trajectories, we set the objectives as biasing the trajectories to a given direction (e.g. forward, backward, left and right), while not colliding with other agents. To that end, we optimize a carefully-designed objective function $\mathcal{L}_{\text{dyn}}$ over the control actions, i.e. $\dot{\kappa}_t$ and a_t for each agents. More specifically, the objective function consists of two components: $\mathcal{L}_{\text{dyn}} = \mathcal{L}_{\text{d}} + \gamma\mathcal{L}_{\text{col}}$, where $\mathcal{L}_{\text{d}}$ is the deviation objective loss, $\mathcal{L}_{\text{col}}$ is the collision regularization loss, and γ is a weight factor to balance the objectives. In each scene, we randomly pick a deviation objective loss $\mathcal{L}_{\text{d}}$ from the set {moving forward, backward, left, right} for each agent. More

specifically, the deviation objective loss $\mathcal{L}_{\mathrm{d}}$ is formulated as

$$\mathcal{L}_{\mathrm{d}} = (\mathbf{X} - \mathbf{X}_{\mathrm{aug}})\,\bar{\mathbf{d}},$$

where $\mathbf{X}_{\mathrm{aug}}$ represents the generated trajectories by perturbing the trajectories in the dataset and $\bar{\mathbf{d}}$ represents the unit vectors for the target deviation directions in the set of {moving forward, backward, left, right}. And the collision regularization loss $\mathcal{L}_{\mathrm{col}}$ is formulated as

$$\mathcal{L}_{\mathrm{col}}(\mathbf{X}_{\mathrm{aug}}, \mathbf{X}) = \frac{1}{n-1} \sum_{i \neq \mathrm{aug}}^{n-1} \frac{1}{\|\mathbf{X}_{\mathrm{aug}} - \mathbf{X}_i\| + 1},$$

We also clip the maximum deviation of the positions so that the trajectories are constrained to be in the lane.

C.1.4 MPC-based Planner

Planner. In this work, to demonstrate the explicit consequences of the adversarial trajectory, we implement a simple yet effective planner that uses conformal lattice [101] for sampling paths and model predictive control (MPC) [24] for motion planning. We call this planner *MPC-based planner*.

Planning strategy. In this work, we consider a closed-loop planning strategy. Though for the closed-loop planning we have to replay the ground truth trajectories of other agents, we do notice reduced collisions and driving off-road consequences compared to open-loop planning and consider the closed-loop planning fashion meaningful.

C.2 Experiment and Results

C.2.1 More details on Experimental Setup

Models. Since the adversarial training process is computationally heavy, we use a lightweight version of the AgentFormer in the analysis and ablation studies, namely mini-AgentFormer. In mini-AgentFormer, we (1) remove the map context and (2) reduce the transformer layer from two layers to a single layer. We report the final results for all three models: AgentFormer (AF), mini-AgentFormer (mini-AF) and Social-GAN.

Evaluating impacts to downstream planners. To demonstrate the impacts to downstream planners, we generate adversarial examples for consecutive frames on traffic scenarios in nuScenes dataset. With the MPC-based planner plugged in, we can demonstrate the consequences of the adversarial attacks on trajectory prediction models. We use the prediction results of AgentFormer trained with different methods due to its best performance among the three models. As we mentioned in the main paper, we show 10 cases where the AV collides with other vehicles under attack. We visualize 3 scenarios in the demo video of the supplementary material.

Hyperparameter choices. To select the hyperparameter β, we conduct adversarial training with different β for controlling the regularization. The results are shown in Table C.1. We find that $\beta = 0.1$ achieves a good trade-off between robustness and clean performance. Therefore, we use $\beta = 0.1$ for the experiments in the rest of the paper.

β	0.01	0.1	0.5	1	10
ADE	2.19	2.29	2.37	2.39	2.57
Robust ADE	3.91	3.76	3.80	3.78	3.79

Table C.1: Ablation study on different regularization loss weights.

To select the PGD attack step for evaluation, we conduct ablation experiments to show the convergence of different PGD steps. As shown in Figure C.3, the attack converges at 20 steps. Thus, we select the 20-step PDG attack for the experiments in this paper

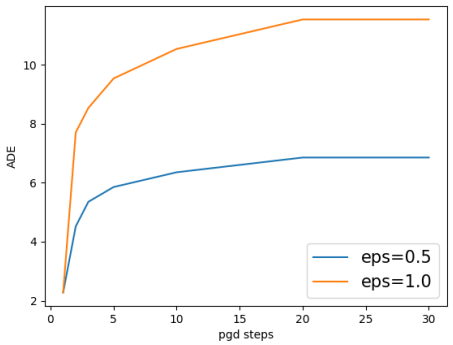


Figure C.3: PGD step convergence for attack convergence with *Deterministic Attack*. Attack converges around 20 steps.

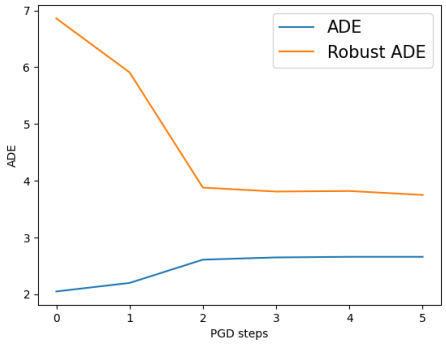


Figure C.4: PGD step sizes ablation study. We find that except for 1 step PGD adversarial training, adversarial training with all the other step sizes achieves similar results.

To select the PGD attack steps of adversarial training, we conduct experiments on adversarially training the model with different PGD steps. Since we are using PGD attack with adaptive step sizes [46], attacks with any PGD steps are able to fully utilize the attack capability controlled by ϵ. In Figure C.4, we show that except for the 1-step PGD attack, all other steps show the similar robustness and clean performance.

Evaluation with existing attack [190]. We also evaluate the robust trained model with the existing *search* attack [190]. The results are shown in the Table C.2. We show that the existing attack increased less prediction error (e.g., 78% less for $\epsilon = 1.0$ on AgentFormer trained with clean data) due to the additional constraints

of the attack. We demonstrate that the proposed *RobustTraj* achieves both best robustness and least clean performance degradation, compared to the baselines.

Table C.2: Evaluation results of the proposed methods and existing methods on the *search* attack proposed by Zhang et al. [190]. mini-AF, AF and SGAN represent mini-AgentFormer, AgentFormer, and Social-GAN respectively. *DA* represents data augmentation with adversarial examples.

Model	Method	ADE		Robust ADE		FDE		Robust FDE	
		0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0
mini-AF	Clean	2.05	2.05	3.24	4.61	4.41	4.41	6.19	8.02
	Naïve AT	2.75	2.78	3.38	4.46	5.92	5.89	6.97	8.17
	DA + Train-time Smoothing	2.41	2.39	3.28	4.04	5.05	5.09	6.36	8.33
	Detection + Test Smoothing	2.31	2.28	3.26	4.41	4.96	4.91	6.41	8.27
	RobustTraj	2.14	2.11	2.50	2.51	4.36	4.35	5.07	5.11
AF	Clean	1.86	1.86	2.62	3.34	3.89	3.89	5.22	6.72
	Naïve AT	2.52	2.56	2.86	3.52	5.18	5.32	5.68	6.75
	DA + Train-time Smoothing	2.17	2.13	2.72	3.44	4.59	4.51	5.38	6.17
	Detection + Test Smoothing	2.08	2.03	2.58	3.29	4.43	4.26	5.49	6.22
	RobustTraj	1.91	1.95	2.14	2.21	4.02	4.01	4.31	4.31
SocialGAN	Clean	4.80	4.80	6.45	8.08	5.52	5.52	7.78	11.12
	Naïve AT	6.43	6.55	6.99	8.66	7.60	7.53	8.98	10.54
	DA + Train-time Smoothing	5.63	5.61	6.42	8.01	6.44	6.41	8.34	9.88
	Detection + Test Smoothing	5.35	5.37	6.34	8.72	6.12	6.07	7.77	10.21
	RobustTraj	4.95	5.07	5.01	5.49	5.72	5.73	6.68	6.40

C.2.2 Main Results

We evaluate our methods and existing methods with four metrics in Table C.3. We observe that the results are consistent where the proposed *RobustTraj* achieves the best results compared to the baselines and existing methods [190].

In the demo video, we visualized scenarios where adversarial attacks on trajectory prediction models lead to collisions on both model trained on clean data and model trained with an existing defense [190], while model trained with *RobustTraj* is able to avoid the collisions.

Table C.3: Additional evaluation results of the proposed methods and existing methods. mini-AF, AF and SGAN represent mini-AgentFormer, AgentFormer, and Social-GAN respectively. *DA* represents data augmentation with adversarial examples.

Model	Method	ADE		Robust ADE		FDE		Robust FDE	
		0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0
mini-AF	Clean	2.05	2.05	6.86	11.53	4.41	4.41	13.08	20.15
	Naïve AT	2.75	2.78	5.44	9.20	5.92	5.89	10.13	15.78
	DA	2.31	2.32	5.54	9.32	5.01	4.92	10.09	15.77
	Train-time Smoothing	3.14	3.07	5.67	9.31	6.77	6.61	10.51	17.48
	Test-time Smoothing	2.97	3.07	4.96	8.50	6.49	6.31	9.25	14.13
	DA + Train-time Smoothing	2.41	2.39	5.48	9.00	5.05	5.09	10.23	16.87
	Detection + Test Smoothing	2.31	2.28	5.91	9.85	4.96	4.91	11.49	17.57
	RobustTraj	**2.14**	**2.11**	**3.69**	**3.82**	**4.36**	**4.35**	**7.10**	**7.59**
AF	Clean	1.86	1.86	5.09	8.57	3.89	3.89	9.42	14.41
	Naïve AT	2.52	2.56	3.81	6.81	5.18	5.32	7.11	10.76
	DA	2.10	2.08	4.35	7.22	4.33	4.38	8.08	12.15
	Train-time Smoothing	2.11	2.13	4.19	6.79	4.40	4.46	8.01	11.13
	Test-time Smoothing	2.40	2.41	4.43	7.44	5.02	4.99	8.23	12.47
	DA + Train-time Smoothing	2.17	2.13	4.14	6.62	4.59	4.51	7.85	11.00
	Detection + Test Smoothing	2.08	2.03	4.45	7.59	4.43	4.26	8.01	12.74
	RobustTraj	**1.91**	**1.95**	**2.73**	**2.86**	**4.02**	**4.01**	**5.22**	**5.48**
SGAN	Clean	4.80	4.80	10.52	20.15	5.52	5.52	15.60	24.79
	Naïve AT	6.43	6.55	8.34	14.63	7.60	7.53	13.71	17.93
	DA	5.41	5.40	8.85	17.25	6.16	6.21	13.33	20.83
	Train-time Smoothing	5.50	5.47	8.74	16.51	6.27	6.31	14.03	19.48
	Test-time Smoothing	6.16	6.17	9.05	17.42	7.14	7.07	13.52	21.81
	DA + Train-time Smoothing	5.63	5.61	8.60	16.14	6.44	6.41	13.82	19.08
	Detection + Test Smoothing	5.35	5.37	9.28	17.39	6.12	6.07	13.36	21.59
	RobustTraj	**4.95**	**5.07**	**5.20**	**6.94**	**5.72**	**5.73**	**8.97**	**8.89**

Model	Method	MR		Robust MR		ORR		Robust ORR	
		0.5	1.0	0.5	1.0	0.5	1.0	0.5	1.0
mini-AF	Clean	0.33	0.33	0.77	0.93	0.08	0.08	0.28	0.45
	Naïve AT	0.45	0.45	0.60	0.70	0.11	0.10	0.22	0.34
	DA	0.37	0.38	0.61	0.72	0.09	0.09	0.22	0.35
	Train-time Smoothing	0.50	0.50	0.66	0.79	0.12	0.12	0.22	0.38
	Test-time Smoothing	0.48	0.50	0.56	0.65	0.11	0.11	0.20	0.32
	DA + Train-time Smoothing	0.39	0.40	0.65	0.77	0.09	0.09	0.22	0.37
	Detection + Test-time Smoothing	0.37	0.38	0.69	0.81	0.09	0.09	0.25	0.41
	RobustTraj	**0.34**	**0.36**	**0.54**	**0.54**	**0.08**	**0.08**	**0.10**	**0.11**
AF	Clean	0.29	0.29	0.66	0.88	0.04	0.04	0.16	0.30
	Naïve AT	0.39	0.38	0.51	0.69	0.06	0.06	0.13	0.22
	DA	0.32	0.32	0.56	0.74	0.05	0.05	0.14	0.26
	Train-time Smoothing	0.33	0.33	0.56	0.71	0.05	0.05	0.13	0.25
	Test-time Smoothing	0.37	0.37	0.58	0.77	0.05	0.05	0.14	0.26
	DA + Train-time Smoothing	0.33	0.33	0.54	0.70	0.05	0.05	0.13	0.24
	Detection + Test-time Smoothing	0.32	0.33	0.59	0.76	0.04	0.05	0.14	0.27
	RobustTraj	**0.29**	**0.31**	**0.46**	**0.51**	**0.04**	**0.04**	**0.05**	**0.07**
SocialGAN	Clean	0.40	0.40	0.85	0.99	0.14	0.14	0.52	0.60
	Naïve AT	0.53	0.53	0.63	0.77	0.19	0.19	0.39	0.44
	DA	0.44	0.44	0.72	0.85	0.16	0.16	0.44	0.51
	Train-time Smoothing	0.45	0.45	0.67	0.82	0.16	0.16	0.42	0.50
	Test-time Smoothing	0.51	0.51	0.74	0.85	0.19	0.19	0.45	0.52
	DA + Train-time Smoothing	0.47	0.46	0.66	0.81	0.17	0.17	0.41	0.49
	Detection + Test-time Smoothing	0.45	0.44	0.74	0.89	0.16	0.16	0.46	0.53
	RobustTraj	**0.41**	**0.42**	**0.60**	**0.62**	**0.15**	**0.15**	**0.24**	**0.29**

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] (2005), HARD BRAKE HARD ACCELERATION, http://tracknet.accountsupport.com/wp-content/uploads/Verizon/Hard-Brake-Hard-Acceleration.pdf.

[2] (2016), ArbExpress, https://www.tek.com/signal-generator/afg2021-software-0.

[3] (2017), Baidu Apollo, http://apollo.auto.

[4] (2017), KITTI Vision Benchmark: 3D Object Detection, http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d, accessed: 2021-08-17.

[5] (2017), What it Was Like to Ride in GM's New Self-Driving Cruise Car, https://www.recode.net/2017/11/29/16712572/general-motors-gm-new-self-driving-autonomous-cruise-car-future.

[6] (2017), An Introduction to LIDAR: The Key Self-Driving Car Sensor, https://news.voyage.auto/an-introduction-to-lidar-the-key-self-driving-car-sensor-a7e405590cff.

[7] (2017), Google's Waymo Invests in LIDAR Technology, Cuts Costs by 90 Percent, https://arstechnica.com/cars/2017/01/googles-waymo-invests-in-lidar-technology-cuts-costs-by-90-percent/.

[8] (2018), Baidu starts mass production of autonomous buses, https://www.dw.com/en/baidu-starts-mass-production-of-autonomous-buses/a-44525629.

[9] (2018), Baidu hits the gas on autonomous vehicles with Volvo and Ford deals, https://techcrunch.com/2018/11/01/baidu-volvo-ford-autonomous-driving/.

[10] (2018), What Is LIDAR, Why Do Self-Driving Cars Need It, And Can It See Nerf Bullets?, https://www.wired.com/story/lidar-self-driving-cars-luminar-video/.

[11] (2018), Volvo Finds the LIDAR it Needs to Build Self-Driving Cars, https://www.wired.com/story/volvo-self-driving-lidar-luminar/.

[12] (2018), You can take a ride in a self-driving Lyft during CES, https://www.theverge.com/2018/1/2/16841090/lyft-aptiv-self-driving-car-ces-2018.

[13] (2018), VeloView, https://www.paraview.org/VeloView/.

[14] (2018), Waymo's autonomous cars have driven 8 million miles on public roads, https://www.theverge.com/2018/7/20/17595968/waymo-self-driving-cars-8-million-miles-testing.

[15] Abadi, M., et al. (2016), Tensorflow: a system for large-scale machine learning., in *OSDI*, vol. 16, pp. 265–283.

[16] Abeysirigoonawardena, Y., F. Shkurti, and G. Dudek (2019), Generating adversarial driving scenarios in high-fidelity simulators, in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8271–8277, IEEE.

[17] Alahi, A., K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese (2016), Social lstm: Human trajectory prediction in crowded spaces, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 961–971.

[18] Athalye, A., and I. Sutskever (2018), Synthesizing Robust Adversarial Examples, in *International Conference on Machine Learning (ICML)*.

[19] Athalye, A., N. Carlini, and D. Wagner (2018), Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, *arXiv preprint arXiv:1802.00420*.

[20] Bafna, M., J. Murtagh, and N. Vyas (2018), Thwarting adversarial examples: An l_0-robustsparse fourier transform, *arXiv preprint arXiv:1812.05013*.

[21] Brostow, G. J., J. Shotton, J. Fauqueur, and R. Cipolla (2008), Segmentation and recognition using structure from motion point clouds, in *ECCV (1)*, pp. 44–57.

[22] Brown, T. B., D. Mané, A. Roy, M. Abadi, and J. Gilmer (2017), Adversarial patch, *arXiv preprint arXiv:1712.09665*.

[23] Caesar, H., et al. (2020), nuscenes: A multimodal dataset for autonomous driving, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11,621–11,631.

[24] Camacho, E. F., and C. B. Alba (2013), *Model predictive control*, Springer science & business media.

[25] Carlini, N., and D. Wagner (2017), Towards evaluating the robustness of neural networks, in *IEEE Symposium on Security and Privacy, 2017*.

[26] Carlini, N., and D. Wagner (2017), Adversarial Examples are not Easily Detected: Bypassing Ten Detection Methods, in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14, ACM.

[27] Carlini, N., and D. Wagner (2018), Audio Adversarial Examples: Targeted Attacks on Speech-to-text, in *Deep Learning and Security Workshop (DLS)*.

[28] Carlini, N., and D. A. Wagner (2017), Towards Evaluating the Robustness of Neural Networks, in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pp. 39–57, doi:10.1109/SP.2017.49.

[29] Carlini, N., P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou (2016), Hidden Voice Commands, in *USENIX Security Symposium*, pp. 513–530.

[30] Carlini, N., A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin (2019), On evaluating adversarial robustness, *arXiv preprint arXiv:1902.06705*.

[31] Checkoway, S., et al. (2011), Comprehensive Experimental Analyses of Automotive Attack Surfaces, in *Proceedings of the 20th USENIX Conference on Security*, SEC'11.

[32] Chen, L.-C., G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille (2017), Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, *IEEE transactions on pattern analysis and machine intelligence*, *40*(4), 834–848.

[33] Chen, L.-C., Y. Zhu, G. Papandreou, F. Schroff, and H. Adam (2018), Encoder-decoder with atrous separable convolution for semantic image segmentation, in *ECCV*.

[34] Chen, Q. A., Y. Yin, Y. Feng, Z. M. Mao, and H. X. L. Liu (2018), Exposing Congestion Attack on Emerging Connected Vehicle based Traffic Signal Control, in *Proceedings of the 25th Annual Network and Distributed System Security Symposium*, NDSS '18.

[35] Chen, S.-T., C. Cornelius, J. Martin, and D. H. P. Chau (2018), Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 52–68, Springer.

[36] Cheng, M., J. Yi, H. Zhang, P.-Y. Chen, and C.-J. Hsieh (2018), Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples, *arXiv preprint arXiv:1803.01128*.

[37] Chiang, P.-y., R. Ni, A. Abdelkader, C. Zhu, C. Studor, and T. Goldstein (2019), Certified defenses for adversarial patches, in *International Conference on Learning Representations*.

[38] Cho, K.-T., and K. G. Shin (2016), Error handling of in-vehicle networks makes them vulnerable, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS'16.

[39] Choquette, J., W. Gandhi, O. Giroux, N. Stam, and R. Krashinsky (2021), Nvidia a100 tensor core gpu: Performance and innovation, *IEEE Micro*, *41*(2), 29–35, doi:10.1109/MM.2021.3061394.

[40] Cignoni, P., M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia (2008), Meshlab: an open-source mesh processing tool., in *Eurographics Italian chapter conference*, vol. 2008, pp. 129–136.

[41] Cisse, M., Y. Adi, N. Neverova, and J. Keshet (2017), Houdini: Fooling deep structured prediction models, *arXiv preprint arXiv:1707.05373*.

[42] Collobert, R., and J. Weston (2008), A unified architecture for natural language processing: Deep neural networks with multitask learning, in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, ACM.

[43] Condrea, O., A. Chiru, R. Chiriac, and S. Vlase (2017), Mathematical model for studying cyclist kinematics in vehicle-bicycle frontal collisions, *IOP Conference Series: Materials Science and Engineering*, *252*, 012,003, doi:10.1088/1757-899x/252/1/012003.

[44] Contributors, M. (2018), MMCV: OpenMMLab computer vision foundation, `https://github.com/open-mmlab/mmcv`.

[45] Cordts, M., M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele (2016), The cityscapes dataset for semantic urban scene understanding, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223.

[46] Croce, F., and M. Hein (2020), Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, in *International Conference on Machine Learning*, pp. 2206–2216, PMLR.

[47] Demontis, A., M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, and F. Roli (2019), Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks, in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 321–338, USENIX Association, Santa Clara, CA.

[48] Deng, L. (2012), The mnist database of handwritten digit images for machine learning research, *IEEE Signal Processing Magazine*, *29*(6), 141–142.

[49] Deng, L., et al. (2013), Recent advances in deep learning for speech research at microsoft., in *ICASSP*, vol. 26, p. 64.

[50] Deo, N., and M. M. Trivedi (2020), Trajectory forecasts in unknown environments conditioned on grid-based plans, *arXiv preprint arXiv:2001.00735*.

[51] Deo, N., E. Wolff, and O. Beijbom (2021), Multimodal trajectory prediction conditioned on lane-graph traversals, in *5th Annual Conference on Robot Learning*.

[52] Ding, W., B. Chen, M. Xu, and D. Zhao (2020), Learning to collide: An adaptive safety-critical scenarios generating method, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2243–2250, IEEE.

[53] Ding, W., B. Chen, B. Li, K. J. Eun, and D. Zhao (2021), Multimodal safety-critical scenarios generation for decision-making algorithms evaluation, *IEEE Robotics and Automation Letters*, *6*(2), 1551–1558.

[54] el al., L. (2019), On physical adversarial patches for object detection, *arXiv preprint arXiv:1906.11897*.

[55] et al., A. (2019), Attacking optical flow, in *ICCV*.

[56] et al., H. (2019), Adversarial signboard against object detector., in *BMVC*.

[57] et al., S. (2020), Role of spatial context in adversarial robustness for object detection, in *CVPR Workshops*.

[58] Everingham, M., L. Van Gool, C. Williams, J. Winn, and A. Zisserman (2010), The pascal visual object classes (voc) challenge, *International Journal of Computer Vision*, *88*, 303–338, doi:10.1007/s11263-009-0275-4.

[59] Evtimov, I., K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song (2017), Robust physical-world attacks on deep learning models, *arXiv preprint arXiv:1707.08945*, *1*.

[60] Eykholt, K., I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramer, A. Prakash, T. Kohno, and D. Song (2018), Physical Adversarial Examples for Object Detectors, in *USENIX Workshop on Offensive Technologies (WOOT)*.

[61] Eykholt, K., I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song (2018), Robust Physical-World Attacks on Deep Learning Visual Classification, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[62] Feng, Y., S. Huang, Q. A. Chen, H. X. Liu, and Z. M. Mao (2018), Vulnerability of Traffic Control System Under Cyber-Attacks Using Falsified Data, in *Transportation Research Board 2018 Annual Meeting (TRB)*.

[63] Fu, J., J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu (2019), Dual attention network for scene segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3146–3154.

[64] Goodfellow, I. J., J. Shlens, and C. Szegedy (2014), Explaining and harnessing adversarial examples, *arXiv preprint arXiv:1412.6572*.

[65] Hallyburton, R. S., Y. Liu, Y. Cao, Z. M. Mao, and M. Pajic (2022), Security analysis of Camera-LiDAR fusion against Black-Box attacks on autonomous vehicles, in *31st USENIX Security Symposium (USENIX Security 22)*, pp. 1903–1920, USENIX Association, Boston, MA.

[66] Hamdi, A., S. Rojas, A. Thabet, and B. Ghanem (2020), AdvPC: Transferable Adversarial Perturbations on 3D Point Clouds, in *European Conference on Computer Vision*, pp. 241–257.

[67] He, K., X. Zhang, S. Ren, and J. Sun (2015), Spatial pyramid pooling in deep convolutional networks for visual recognition, *IEEE transactions on pattern analysis and machine intelligence*, *37*(9), 1904–1916.

[68] He, K., X. Zhang, S. Ren, and J. Sun (2016), Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

[69] Huang, G., Z. Liu, L. Van Der Maaten, and K. Q. Weinberger (2017), Densely connected convolutional networks., in *CVPR*, vol. 1, p. 3.

[70] Huang, L., C. Gao, Y. Zhou, C. Xie, A. Yuille, C. Zou, and N. Liu (2019), Universal physical camouflage attacks on object detectors.

[71] Huang, L., C. Gao, Y. Zhou, C. Xie, A. L. Yuille, C. Zou, and N. Liu (2020), Universal physical camouflage attacks on object detectors, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 720–729.

[72] Huang, Z., X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu (2019), Ccnet: Criss-cross attention for semantic segmentation, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 603–612.

[73] Ilyas, A., L. Engstrom, A. Athalye, and J. Lin (2018), Black-box adversarial attacks with limited queries and information, *arXiv preprint arXiv:1804.08598*.

[74] Inc., V. L. (2018), VLP-16 User Manual.

[75] Ivanovic, B., and M. Pavone (2019), The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2375–2384.

[76] Ivanovic, B., and M. Pavone (2021), Injecting planning-awareness into prediction and detection evaluation, *CoRR*, *abs/2110.03270*.

[77] Jaderberg, M., K. Simonyan, A. Zisserman, et al. (2015), Spatial transformer networks, in *Advances in neural information processing systems*, pp. 2017–2025.

[78] Jeddi, A., M. J. Shafiee, and A. Wong (2020), A simple fine-tuning is all you need: Towards robust deep learning via adversarial fine-tuning, *arXiv preprint arXiv:2012.13628*.

[79] Jekel, C. F., G. Venter, M. P. Venter, N. Stander, and R. T. Haftka (2019), Similarity measures for identifying material parameters from hysteresis loops using inverse analysis, *International Journal of Material Forming*, doi:10.1007/s12289-018-1421-8.

[80] Kingma, D. P., and J. Ba (2014), Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980*.

[81] Koren, M., and M. J. Kochenderfer (2019), Efficient autonomy validation in simulation with adaptive stress testing, in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 4178–4183, IEEE.

[82] Kosaraju, V., A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese (2019), Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks, *Advances in Neural Information Processing Systems*, *32*.

[83] Koscher, K., et al. (2010), Experimental Security Analysis of a Modern Automobile, in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP'10.

[84] Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012), Imagenet classification with deep convolutional neural networks, in *Advances in neural information processing systems*, pp. 1097–1105.

[85] Kurakin, A., I. Goodfellow, and S. Bengio (2016), Adversarial examples in the physical world, *arXiv preprint arXiv:1607.02533*.

[86] Lang, A. H., S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom (2019), Pointpillars: Fast Encoders for Object Detection from Point Clouds, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12,697–12,705.

[87] Lee, M., and Z. Kolter (2019), On physical adversarial patches for object detection, *arXiv preprint arXiv:1906.11897*.

[88] Lee, S., H. Lee, and S. Yoon (2020), Adversarial vertex mixup: Toward better adversarially robust generalization, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 272–281.

[89] Li, X., Z. Zhong, J. Wu, Y. Yang, Z. Lin, and H. Liu (2019), Expectation-maximization attention networks for semantic segmentation, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9167–9176.

[90] Li, X., J. Kaesemodel Pontes, and S. Lucey (2021), Neural scene flow prior, *Advances in Neural Information Processing Systems*, *34*.

[91] Liu, D., R. Yu, and H. Su (2019), Extending adversarial attacks and defenses to deep 3d point cloud classifiers, in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 2279–2283, IEEE.

[92] Liu, H.-T. D., M. Tao, C.-L. Li, D. Nowrouzezahrai, and A. Jacobson (2018), Adversarial geometry and lighting using a differentiable renderer, *CoRR*, *abs/1808.02651*.

[93] Liu, W., D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg (2016), Ssd: Single shot multibox detector, in *European conference on computer vision*, pp. 21–37, Springer.

[94] Liu, X., H. Yang, Z. Liu, L. Song, H. Li, and Y. Chen (2018), Dpatch: An adversarial patch attack on object detectors, *arXiv preprint arXiv:1806.02299*.

[95] Liu, Y., S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang (2018), Trojaning Attack on Neural Networks, in *Annual Network and Distributed System Security Symposium (NDSS)*.

[96] Luc, P., C. Couprie, S. Chintala, and J. Verbeek (2016), Semantic segmentation using adversarial networks, *arXiv preprint arXiv:1611.08408*.

[97] Luo, W., Y. Li, R. Urtasun, and R. Zemel (2016), Understanding the effective receptive field in deep convolutional neural networks, in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 4905–4913.

[98] Madry, A., A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu (2017), Towards deep learning models resistant to adversarial attacks, *arXiv preprint arXiv:1706.06083*.

[99] Madry, A., A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu (2018), Towards deep learning models resistant to adversarial attacks, in *International Conference on Learning Representations*.

[100] Mazloom, S., M. Rezaeirad, A. Hunter, and D. McCoy (2016), A Security Analysis of an In-Vehicle Infotainment and App Platform, in *Usenix Workshop on Offensive Technologies (WOOT)*.

[101] McNaughton, M., C. Urmson, J. M. Dolan, and J.-W. Lee (2011), Motion planning for autonomous driving with a conformal spatiotemporal lattice, in *2011 IEEE International Conference on Robotics and Automation*, pp. 4889–4895, IEEE.

[102] Meng, D., and H. Chen (2017), Magnet: a two-pronged defense against adversarial examples, in *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pp. 135–147.

[103] Mirsky, Y. (2021), Ipatch: A remote adversarial patch, *arXiv preprint arXiv:2105.00113*.

[104] Moosavi-Dezfooli, S.-M., A. Fawzi, and P. Frossard (2016), Deepfool: a simple and accurate method to fool deep neural networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582.

[105] Noack, A., I. Ahern, D. Dou, and B. Li (2021), An empirical study on the relation between network interpretability and adversarial robustness, *SN Computer Science*, *2*(1), 1–13.

[106] Papernot, N., and P. McDaniel (2017), Extending defensive distillation, *arXiv preprint arXiv:1705.05264*.

[107] Papernot, N., P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami (2016), The limitations of deep learning in adversarial settings, in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387, IEEE.

[108] Papernot, N., P. McDaniel, X. Wu, S. Jha, and A. Swami (2016), Distillation as a defense to adversarial perturbations against deep neural networks, in *2016 IEEE symposium on security and privacy (SP)*, pp. 582–597, IEEE.

[109] Papernot, N., P. McDaniel, X. Wu, S. Jha, and A. Swami (2016), Distillation as a defense to adversarial perturbations against deep neural networks, in *Security and Privacy (SP), 2016 IEEE Symposium on*, pp. 582–597, IEEE.

[110] Papernot, N., P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami (2017), Practical Black-Box Attacks Against Machine Learning, in *ACM on Asia Conference on Computer and Communications Security*.

[111] Petit, J., B. Stottelaar, M. Feiri, and F. Kargl (2015), Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR, in *Black Hat Europe*.

[112] Petitcolas, F. A. P. (2011), *Kerckhoffs' Principle*, pp. 675–675, Springer US, Boston, MA, doi:10.1007/978-1-4419-5906-5_487.

[113] Polack, P., F. Altché, B. d'Andréa Novel, and A. de La Fortelle (2017), The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?, in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 812–818, doi:10.1109/IVS.2017.7995816.

[114] Qi, C. R., H. Su, K. Mo, and L. J. Guibas (2017), Pointnet: Deep learning on point sets for 3d classification and segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660.

[115] Rebuffi, S.-A., S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. Mann (2021), Fixing data augmentation to improve adversarial robustness, *arXiv preprint arXiv:2103.01946*.

[116] Redmon, J., and A. Farhadi (2017), Yolo9000: Better, faster, stronger, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263–7271.

[117] Rempe, D., J. Philion, L. J. Guibas, S. Fidler, and O. Litany (2021), Generating useful accident-prone driving scenarios via a learned traffic prior, in *arXiv:2112.05077*.

[118] Rempe, D., J. Philion, L. J. Guibas, S. Fidler, and O. Litany (2022), Generating useful accident-prone driving scenarios via a learned traffic prior, in *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[119] Ren, S., K. He, R. Girshick, and J. Sun (2015), Faster r-cnn: Towards real-time object detection with region proposal networks, in *Advances in neural information processing systems*, pp. 91–99.

[120] Ren, Z., X. Wang, N. Zhang, X. Lv, and L.-J. Li (2017), Deep reinforcement learning-based image captioning with embedding reward, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 290–298.

[121] Rennie, S. J., E. Marcheret, Y. Mroueh, J. Ross, and V. Goel (2017), Self-critical sequence training for image captioning, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7008–7024.

[122] Rhinehart, N., K. M. Kitani, and P. Vernaza (2018), R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting, in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 772–788.

[123] Rhinehart, N., R. McAllister, K. Kitani, and S. Levine (2019), Precog: Prediction conditioned on goals in visual multi-agent settings, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2821–2830.

[124] Rice, L., E. Wong, and Z. Kolter (2020), Overfitting in adversarially robust deep learning, in *International Conference on Machine Learning*, pp. 8093–8104, PMLR.

[125] Rouf, I., R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar (2010), Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study, in *Proceedings of the 19th USENIX Conference on Security*, USENIX Security'10, pp. 21–21, USENIX Association, Berkeley, CA, USA.

[126] Salzmann, T., B. Ivanovic, P. Chakravarty, and M. Pavone (2020), Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data, in *European Conference on Computer Vision*, pp. 683–700, Springer.

[127] Sarkar, A., A. Sarkar, S. Gali, and V. N Balasubramanian (2021), Adversarial robustness without adversarial training: A teacher-guided curriculum learning approach, *Advances in Neural Information Processing Systems*, *34*.

[128] Shafahi, A., M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein (2019), Adversarial training for free!, *arXiv preprint arXiv:1904.12843*.

[129] Shafahi, A., M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein (2019), Adversarial training for free!, *arXiv preprint arXiv:1904.12843*.

[130] Shannon, C. E. (1949), Communication theory of secrecy systems, *Bell Labs Technical Journal*, *28*(4), 656–715.

[131] Sharif, M., S. Bhagavatula, L. Bauer, and M. K. Reiter (2016), Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition, in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1528–1540, ACM.

[132] Shin, H., D. Kim, Y. Kwon, and Y. Kim (2017), Illusion and Dazzle: Adversarial Optical Channel Exploits Against Lidars for Automotive Applications, in *International Conference on Cryptographic Hardware and Embedded Systems (CHES)*.

[133] Shoukry, Y., P. Martin, P. Tabuada, and M. Srivastava (2013), Non-invasive spoofing attacks for anti-lock braking systems, in *Cryptographic Hardware and Embedded Systems - CHES 2013*, edited by G. Bertoni and J.-S. Coron, pp. 55–72, Springer Berlin Heidelberg, Berlin, Heidelberg.

[134] Silver, D., et al. (2016), Mastering the game of go with deep neural networks and tree search, *nature*, *529*(7587), 484.

[135] Simonyan, K., and A. Zisserman (2014), Very deep convolutional networks for large-scale image recognition, *arXiv preprint arXiv:1409.1556*.

[136] Sun, J., Y. Cao, Q. A. Chen, and Z. M. Mao (2020), Towards Robust LiDAR-based Perception in Autonomous Driving: General Black-box Adversarial Sensor Attack and Countermeasures, in *29th USENIX Security Symposium (USENIX Security 20)*, pp. 877–894.

[137] Sun, M., J. Tang, H. Li, B. Li, C. Xiao, Y. Chen, and D. Song (2018), Data poisoning attack against unsupervised node embedding methods, *arXiv preprint arXiv:1810.12881*.

[138] Suo, S., S. Regalado, S. Casas, and R. Urtasun (2021), Trafficsim: Learning to simulate realistic multi-agent behaviors, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10,400–10,409.

[139] Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus (2013), Intriguing properties of neural networks, *arXiv preprint arXiv:1312.6199*.

[140] Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2015), Going deeper with convolutions, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

[141] Szegedy, C., V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna (2016), Rethinking the inception architecture for computer vision, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826.

[142] Szegedy, C., S. Ioffe, V. Vanhoucke, and A. Alemi (2017), Inception-v4, inception-resnet and the impact of residual connections on learning, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31.

[143] Tan, M., and Q. Le (2019), Efficientnet: Rethinking model scaling for convolutional neural networks, in *International Conference on Machine Learning*, pp. 6105–6114, PMLR.

[144] Tu, J., T. Wang, J. Wang, S. Manivasagam, M. Ren, and R. Urtasun (2021), Adversarial attacks on multi-agent communication, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7768–7777.

[145] Visin, F., M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci, and A. Courville (2016), Reseg: A recurrent neural network-based model for semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 41–48.

[146] Wang, J., A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun (2021), Advsim: Generating safety-critical scenarios for self-driving vehicles, in *CVPR*, pp. 9909–9918.

[147] Wang, P., P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell (2017), Understanding convolution for semantic segmentation, *arXiv preprint arXiv:1702.08502*.

[148] Wang, X., R. Girshick, A. Gupta, and K. He (2018), Non-local neural networks, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803.

[149] Wen, Y., J. Lin, K. Chen, and K. Jia (2019), Geometry-aware generation of adversarial and cooperative point clouds.

[150] Weng, X., J. Wang, D. Held, and K. Kitani (2020), 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics, *IROS*.

[151] Wikipedia contributors (2022), Salt-and-pepper noise — Wikipedia, the free encyclopedia, [Online; accessed 16-June-2022].

[152] Wong, E., L. Rice, and J. Z. Kolter (2020), Fast is better than free: Revisiting adversarial training, *arXiv preprint arXiv:2001.03994*.

[153] Wong, E., L. Rice, and J. Z. Kolter (2020), Fast is better than free: Revisiting adversarial training, *arXiv preprint arXiv:2001.03994*.

[154] Wong, W., S. Huang, Y. Feng, Q. A. Chen, Z. M. Mao, and H. X. Liu (2019), Trajectory-Based Hierarchical Defense Model to Detect Cyber-Attacks on Transportation Infrastructure, in *Transportation Research Board 2019 Annual Meeting (TRB)*.

[155] Xiang, C., C. R. Qi, and B. Li (2018), Generating 3d adversarial point clouds, *arXiv preprint arXiv:1809.07016*.

[156] Xiang, C., C. R. Qi, and B. Li (2019), Generating 3D Adversarial Point Clouds, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9136–9144.

[157] Xiao, C., R. Deng, B. Li, F. Yu, D. Song, et al. (2018), Characterizing adversarial examples based on spatial consistency information for semantic segmentation, in *Proceedings of the (ECCV)*, pp. 217–234.

[158] Xiao, C., B. Li, J.-Y. Zhu, W. He, M. Liu, and D. Song (2018), Generating adversarial examples with adversarial networks, *arXiv preprint arXiv:1801.02610*.

[159] Xiao, C., D. Yang, B. Li, J. Deng, and M. Liu (2018), Meshadv: Adversarial meshes for visual recognition, in *CVPR*.

[160] Xiao, C., J.-Y. Zhu, B. Li, W. He, M. Liu, and D. Song (2018), Spatially transformed adversarial examples, *arXiv preprint arXiv:1801.02612*.

[161] Xiao, C., R. Deng, B. Li, T. Lee, B. Edwards, J. Yi, D. Song, M. Liu, and I. Molloy (2019), Advit: Adversarial frames identifier based on temporal consistency in videos, in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3968–3977.

[162] Xiao, C., D. Yang, B. Li, J. Deng, and M. Liu (2019), Meshadv: Adversarial meshes for visual recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6898–6907.

[163] Xie, C., and A. Yuille (2020), Intriguing properties of adversarial training at scale, in *International Conference on Learning Representations*.

[164] Xie, C., J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille (2017), Adversarial Examples for Semantic Segmentation and Object Detection, in *IEEE International Conference on Computer Vision (ICCV)*.

[165] Xie, C., J. Wang, Z. Zhang, Y. Zhou, L. Xie, and A. Yuille (2017), Adversarial Examples for Semantic Segmentation and Object Detection, in *IEEE International Conference on Computer Vision (ICCV)*, IEEE.

[166] Xie, C., M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le (2020), Adversarial examples improve image recognition, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 819–828.

[167] Xie, C., M. Tan, B. Gong, A. Yuille, and Q. V. Le (2020), Smooth adversarial training, *arXiv preprint arXiv:2006.14536*.

[168] Xie, E., W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo (2021), Segformer: Simple and efficient design for semantic segmentation with transformers, *arXiv preprint arXiv:2105.15203*.

[169] Xu, W., D. Evans, and Y. Qi (2017), Feature squeezing: Detecting adversarial examples in deep neural networks, *arXiv preprint arXiv:1704.01155*.

[170] Xu, W., D. Evans, and Y. Qi (2017), Feature squeezing: Detecting adversarial examples in deep neural networks, *arXiv preprint arXiv:1704.01155*.

[171] Xu, X., X. Chen, C. Liu, A. Rohrbach, T. Darell, and D. Song (2017), Can you fool ai with adversarial examples on a visual turing test?, *arXiv preprint arXiv:1709.08693*.

[172] Xue, H., D. Q. Huynh, and M. Reynolds (2018), Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction, in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1186–1194, IEEE.

[173] Yan, C. (2016), Can you trust autonomous vehicles : Contactless attacks against sensors of self-driving vehicle.

[174] Yang, C., A. Kortylewski, C. Xie, Y. Cao, and A. Yuille (2020), Patchattack: A black-box texture-based attack with reinforcement learning, in *European Conference on Computer Vision*, pp. 681–698, Springer.

[175] Yang, Y., G. Zhang, D. Katabi, and Z. Xu (2019), Me-net: Towards effective adversarial robustness with matrix estimation, *arXiv preprint arXiv:1905.11971*.

[176] Yang, Y., G. Zhang, D. Katabi, and Z. Xu (2019), Me-net: Towards effective adversarial robustness with matrix estimation, *arXiv preprint arXiv:1905.11971*.

[177] You, C., Z. Hau, and S. Demetriou (2021), Temporal consistency checks to detect lidar spoofing attacks on autonomous vehicle perception, in *Proceedings of the 1st Workshop on Security and Privacy for Mobile AI*, MAISP'21, p. 13–18, Association for Computing Machinery, New York, NY, USA, doi:10.1145/3469261.3469406.

[178] Yu, C., C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang (2021), Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation, *International Journal of Computer Vision*, pp. 1–18.

[179] Yu, F., and V. Koltun (2016), Multi-scale context aggregation by dilated convolutions, in *International Conference on Learning Representations (ICLR)*.

[180] Yu, F., and V. Koltun (2016), Multi-scale context aggregation by dilated convolutions, in *ICLR*.

[181] Yu, F., V. Koltun, and T. Funkhouser (2017), Dilated residual networks, in *Computer Vision and Pattern Recognition (CVPR)*.

[182] Yuan, X., et al. (2018), CommanderSong: A Systematic Approach for Practical Adversarial Voice Recognition, in *USENIX Security Symposium*, pp. 49–64.

[183] Yuan, Y., and J. Wang (2018), Ocnet: Object context network for scene parsing.

[184] Yuan, Y., X. Chen, and J. Wang (2020), Object-contextual representations for semantic segmentation.

[185] Yuan, Y., X. Weng, Y. Ou, and K. Kitani (2021), Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting, in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[186] Zhang, H., H. Chen, C. Xiao, S. Gowal, R. Stanforth, B. Li, D. Boning, and C.-J. Hsieh (2019), Towards stable and efficient training of verifiably robust neural networks, *arXiv preprint arXiv:1906.06316*.

[187] Zhang, H., Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan (2019), Theoretically principled trade-off between robustness and accuracy, in *International Conference on Machine Learning*, pp. 7472–7482, PMLR.

[188] Zhang, H., H. Chen, C. Xiao, B. Li, D. S. Boning, and C.-J. Hsieh (2020), Robust deep reinforcement learning against adversarial perturbations on observations.

[189] Zhang, Q., S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao (2022), On adversarial robustness of trajectory prediction for autonomous vehicles, *CoRR*, *abs/2201.05057*.

[190] Zhang, Q., S. Hu, J. Sun, Q. A. Chen, and Z. M. Mao (2022), On adversarial robustness of trajectory prediction for autonomous vehicles, *arXiv preprint arXiv:2201.05057*.

[191] Zhao, H., J. Shi, X. Qi, X. Wang, and J. Jia (2017), Pyramid scene parsing network, in *CVPR*.

[192] Zhao, H., Y. Zhang, S. Liu, J. Shi, C. C. Loy, D. Lin, and J. Jia (2018), PSANet: Point-wise spatial attention network for scene parsing, in *ECCV*.

[193] Zheng, H., Z. Zhang, H. Lee, and A. Prakash (2020), Understanding and diagnosing vulnerability under adversarial attacks, *arXiv preprint arXiv:2007.08716*.

[194] Zhong, Z., Z. Q. Lin, R. Bidart, X. Hu, I. B. Daya, Z. Li, W.-S. Zheng, J. Li, and A. Wong (2020), Squeeze-and-attention networks for semantic segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13,065–13,074.