
Security Investigation of Autonomous Driving Systems



Xingshuo Han

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2024

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

26-Sep-2023

.....

Date

NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
Xingshuo Han
NTU NTU NTU NTU NTU NTU NTU NTU
.....

Xingshuo Han

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

26-Sep-2023

.....

Date

NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU
.....



Asst Prof Tianwei Zhang

Authorship Attribution Statement

Please select one of the following; *delete as appropriate:

This thesis contains material from 3 paper published in the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Please amend the typical statements below to suit your circumstances if (B) is selected.

Chapter 6 is published as Xingshuo Han, Yuan Zhou, Kangjie Chen, Han Qiu, Meikang Qiu, Yang Liu and Tianwei Zhang, ADS-lead: Lifelong anomaly detection in autonomous driving systems. in IEEE Transactions on Intelligent Transportation Systems, 2022. <https://ieeexplore.ieee.org/abstract/document/9690769>. DOI: 10.1109/TITS.2021.3122906.

A Unified Anomaly Detection Methodology for Lane-Following of Autonomous Driving Systems. in IEEE International Symposium on Parallel and Distributed Processing with Applications, 2021. <https://ieeexplore.ieee.org/abstract/document/9644710>. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00119.

The contributions of the co-authors are as follows:

- I was the lead author, I wrote the manuscript drafts and conducted all experiments.
- Prof Tianwei Zhang guided the initial research direction and revised the manuscript drafts.
- I co-designed the methodology with Prof Tianwei and Dr. Yuan Zhou.
- Mr Kangjie, Prof Meikang Qiu and Prof Yang Liu discussed and supported the research, and revised the drafts.

Chapter 3 is published as Xingshuo Han, Guowen Xu, Yuan Zhou, Xuehuan Yang, Jiwei Li, Tianwei Zhang. Physical Backdoor Attacks to Lane Detection Systems in Autonomous Driving. In Proceedings of the 30th ACM International Conference on Multimedia, 2022. <https://dl.acm.org/doi/abs/10.1145/3503161.3548171>. DOI: <https://doi.org/10.1145/3503161.3548171>

The contributions of the co-authors are as follows:

- I was the lead author, I wrote the manuscript drafts and conducted all experiments.

- Prof Tianwei Zhang guided the initial research direction and revised the manuscript drafts.
- I co-designed the methodology with Prof Tianwei and Dr. Yuan Zhou.
- Dr. Guowen Xu, Mr. Xuehuan Yang, Prof Jiwei Li discussed and supported the research, and revised the drafts.

26-Sep-2023

.....

Date

NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
.....

Xingshuo Han

Xingshuo Han

Acknowledgements

It is the sixth year since I first came to Singapore to pursue a Ph.D. This modern city has been my second hometown 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.

As I stand at the end of my Ph.D. study, my heart brims with appreciation for the numerous individuals who have played pivotal roles in this incredible journey. First and foremost, I must extend my sincere gratitude to my supervisor, Prof Tianwei Zhang, His invaluable guidance and support have been the bedrock upon which this research journey was built. Professor Zhang's expertise, encouragement, and willingness to explore uncharted territories with me have been instrumental in shaping this thesis. He has been supporting me in exploring many new directions, which are often frustrating due to the unknown challenges. With his guidance over the past few years, I have not only grown as a researcher but also gained the confidence to conduct independent research. Besides my supervisor, I also would like to thank my academic brother Prof Zhiguang Cao, who has guided and helped me a lot when I was pursuing my Ph.D. degree.

I also extend my heartfelt thanks to Dr. Yuan Zhou, Dr. Guowen Xu and Dr Haozhao Wang, whose insightful discussions and generous assistance enriched this work in countless ways. Their contributions were indispensable, and without their support, many of the projects I embarked upon would have remained unattainable.

Besides, my journey would not have been the same without the incredible companionship of my lab mates - Kangjie Chen, Gelei Deng, Xiaoxuan Lou, Dikai Liu, Yutong Wu, Haoran Ou, Guanlin Li, Qinghao Hu, Wei Gao, Dr Xiuheng Wu, Tianlin Li, Dr. Yun Tang, Dr. Jianfei Sun, Dr. Hao Ren, Dr. Hangcheng Liu, Dr. Yuan Xu and other guys. Project after project, they not only offered their helping hands but also served as a continuous source of inspiration, pushing me to strive for a higher stage.

I would like to thank my administrative assistants Dr. Shi Ling and Ms. Zolynn for making everything easier. Their meticulousness in administrative matters is amazing but touching.

Also, I am also deeply thankful to my girlfriend, Ms Jing Wang, for her unwavering support and understanding. Her love and motivation have been a constant wellspring of inspiration, and I am profoundly grateful for her presence in my life.

Lastly, but certainly not least, I want to express my deep appreciation to my family—my father, Junhua Han; my mother, Yanli Li; and my sister, Xingya Han. Their unconditional love and unwavering encouragement have formed the bedrock upon which my academic journey was built. This dissertation is dedicated to them as a token of my love and gratitude.

To all those I've mentioned and to those whose names may have been unintentionally omitted, I extend my deepest thanks for your indispensable contributions to this chapter of my life. The passion, love, and growth I've experienced throughout this journey have been made possible by each of you, and for that, I am eternally grateful.

Contents

Acknowledgements	ix
List of Figures	xv
List of Tables	xix
Abstract	1
1 Introduction	3
1.1 Motivation	4
1.2 Main Work	5
1.3 Contribution of the Thesis	7
1.4 List of Materials Related to the Thesis	8
1.5 Outline of the Thesis	8
2 Related Work	11
2.1 Physical Adversarial Attack to ADS Perception	11
2.1.1 Physical Backdoor Attack	12
2.1.2 Physical Adversarial Attack	12
2.1.2.1 Physical Attacks to Camera-based Perception	13
2.1.2.2 Physical Attacks to MSF-based Perception	13
2.1.2.3 Adversarial Mediums	14
2.2 Physical Adversarial Attack to ADS Decision-making	15
2.3 Physical Adversarial Defense	16
2.3.1 Certified Defenses	16
2.3.2 Vision-based Consistency Checking	17
2.3.3 Anomaly Detection	18
3 A Comprehensive Platform for Benchmarking Backdoor Attacks to the Perception Module in Autonomous Vehicles	21
3.1 Introduction	21
3.2 Background and Motivation	25
3.2.1 Backdoor Attacks	25
3.2.2 Motivation	25






3.2.3	Threat Model and Attack Scope	27
3.3	Platform Design	28
3.3.1	Customized Interface	29
3.3.1.1	Trigger Design	29
3.3.1.2	Target Functions and DL Models.	30
3.3.2	Synthesis Module	31
3.3.2.1	Obstacle Detection	32
3.3.2.2	Lane Detection	33
3.3.2.3	Traffic Light Detection	34
3.3.3	Deployment Module	35
3.3.3.1	AV Simulator	35
3.3.3.2	Physical Vehicle	35
3.3.4	Analysis Module	36
3.3.4.1	Metric Calculation	36
3.3.4.2	Analysis Tools	37
3.4	Evaluation	38
3.4.1	Benign Accuracy	39
3.4.2	Attack Feasibility	39
3.4.2.1	Obstacle Detection	39
3.4.2.2	Lane Detection	42
3.4.2.3	Traffic Light Detection	43
3.4.3	Takeaways	45
3.4.3.1	Vulnerability Analysis	45
3.4.3.2	Trigger Analysis	47
3.4.3.3	Environmental Impact Analysis	48
3.4.3.4	Attack Goal Analysis	50
3.4.3.5	Sensor Fusion Analysis	50
3.4.4	Extensible to Other Attack Techniques	52
3.4.5	Mitigation	54
3.4.6	Limitations	54
3.5	Conclusion	55
4	A Dynamic Physical-world Vulnerability Testing Platform for Decision-	
	making Module in Autonomous Vehicles	57
4.1	Introduction	58
4.2	Background	63
4.2.1	Decision-making Module in ADS	63
4.2.2	Distinction to Existing Works	64
4.2.3	Threat Model	65
4.3	Motivating Examples	66
4.4	Design of STFA	67
4.4.1	Scenario Extraction Module	68
4.4.2	Scenario Description Module	72

4.4.3	Attack Exploitation Module	75
4.4.4	Attack Report Module	78
4.4.5	The Similarity and Difference between <i>Direct-attack</i> and <i>Indirect-attack</i>	84
4.5	Limitation and Discussion	84
4.6	Conclusion	85
5	A Unified Defense Framework against Physical Adversarial Attacks to Autonomous Driving Visual Perception	87
5.1	Introduction	88
5.2	Background and Related Work	92
5.2.1	Visual Perception Module in ADS	92
5.2.2	Physical Adversarial Examples (PAEs)	92
5.2.3	Existing Defenses against Physical Adversarial Attacks	93
5.3	Design Insight	95
5.3.1	Threat Model	95
5.3.2	Key Insight	96
5.3.3	Difficulty of Generating Perfect PAEs	96
5.3.4	Ineffectiveness of Robustness-enhanced Solutions	99
5.3.4.1	Empirical Study	101
5.3.5	Key Idea behind <i>BatAV</i>	102
5.4	<i>VisionGuard</i>	103
5.4.1	Overview	103
5.4.2	State Correction Module (SCM)	104
5.4.3	State Prediction Module (SPM)	106
5.4.4	Attack Detection Module (ADM)	108
5.5	Simulation Evaluation	109
5.5.1	Experiment Setup	110
5.5.2	Evaluation of SCM	111
5.5.3	Evaluation of SPM	112
5.5.4	Evaluation of ADM	114
5.5.5	End-to-end Evaluation of <i>VisionGuard</i>	114
5.5.5.1	Defense Robustness	116
5.6	Outdoor Road Driving Test	117
5.7	Discussion and Limitation	119
5.7.1	Adaptive Attack	119
5.7.1.1	Mechanism-aware adaptive attack.	119
5.7.1.2	Parameter-aware adaptive attack.	119
5.7.2	Evaluation Under Normal Cases	122
5.8	Conclusion	123
6	A Lifelong Anomaly Detection Framework against Physical Adversarial Attacks to Autonomous Driving	125
6.1	Introduction	126

6.2	Related Works	129
6.2.1	Detection of GPS Spoofing Attacks against AVs.	129
6.2.2	Detection of Adversarial Examples.	129
6.3	Background and Problem Statement	130
6.3.1	Overview of ADSs	130
6.3.2	Security Threats in the Lane Following Scenario	131
6.3.3	Problem Statement	133
6.4	ADS-Lead	134
6.4.1	System Overview	134
6.4.2	T-GP: One-class Model for Anomaly Detection	136
6.5	Model Evolution	138
6.5.1	Lifelong Learning	138
6.5.2	Model Update with Federated Learning	140
6.6	Evaluations	141
6.6.1	Evaluation of T-GP	141
6.6.1.1	Defeating Localization Attacks	141
6.6.1.2	Defeating Traffic Sign Recognition Attacks	146
6.6.1.3	Defeating Lane Detection Attacks	149
6.6.2	Evaluation of ADS-Lead	150
6.6.2.1	Datasets	150
6.6.2.2	Baseline Models and Model Configurations	151
6.6.2.3	Evaluation Results	153
6.6.3	Discussion on the Robustness of ADS-Lead	154
6.7	Conclusion	154
7	Conclusion and Future Work	157
7.1	Conclusion	157
7.2	Future Work	159
	Bibliography	165
.1	Appendix A: Proof for Theorem 1 in Chapter 5	186

List of Figures

1.1	The structure of thesis with the pipeline of a typical production-level ADS.	4
2.1	Camera-LIDAR fusion logic in Apollo.	14
3.1	Our physical testbeds: Baidu Apollo Dev Kit (left) and UGV (right).	22
3.2	High-level workflow of an ADS. Red rectangles denote the target DL-based perception functions.	23
3.3	BatAV Pipeline.	27
3.4	Physical triggers adopted in BatAV. From left to right: traffic cones, manhole cover patch, shadow and rain.	28
3.5	OGA and ODA against Yolov3. A pedestrian is generated in OGA while a car is missed in ODA.	39
3.6	OGA and ODA against SMOKE and DEVIANT. A pedestrian is generated in OGA while the front car is missed in ODA. Note that the front car in ODA can be well-detected by benign DEVIANT.	39
3.7	OGA against Yolov4 \oplus PointPillar fusion with camera data (left) and point cloud data (right).	40
3.8	OGA with the traffic cone trigger in Apollo and LGSVL simulator. The adversary can make the MSF detect a non-existing pedestrian by only attacking the LiDAR model. Our attack still works even though the traffic cone in the simulator with a different color than the one we injected.	40
3.9	OGA using patch (left) and shadow (right) triggers.	41
3.10	OGA (left) and ODA (right) with the traffic cones.	41
3.11	LDA and LFA with TuSimple. From left to right: cone, patch, shallow, and rain. In LDA, the left lane boundary is miss-detected, and a yellow boundary is false-generated.	42
3.12	LFA in the physical world. From left to right: cone, patch, shallow, and rain.	42
3.13	LFA (first row) and LDA (second row) against the backdoored SCNN with cones (left) and rain (right) as triggers in LGSVL.	42
3.14	Backdoor attack over the LISA dataset. From left to right: Benign model, TLDA, R2G attack.	43

3.15	TLDA with the traffic cone trigger in LGSVL and Apollo simulator. The “Unknown” status stops the vehicle when the backdoored model miss-detects the green light.	44
3.16	TLDA and R2G attacks at various angles and distances.	45
3.17	Traffic light evaluation in the physical world. From left to right: Benign model, TLDL, R2G attack.	46
3.18	Comparison of GradCam with different poisoning budgets (Yolov3).	46
3.19	GradCam visualization of OGA over different training epochs (from 10 to 100). The confidence (%) from left to right: 50, 50, 73, 76, 82, 77, 71, 79, 83, 84.	47
3.20	Attack effectiveness with different triggers.	48
3.21	For the 2D patch trigger, both OGA (left) and ODA (right) fail at twilight time. The patch color is changed compared with Figure 3.9.	48
3.22	OGA and ODA results in the physical world. The victim model is Yolov3 with different poisoning budgets.	48
3.23	OGA with rain trigger. The attack is erratic with different densities of raindrops. (Left) Attack is successful where a pedestrian is generated. (Right) Attack fails.	49
3.24	ASR comparisons across different attack goals.	51
3.25	Obstacle detection results of NuScenes (left) and physical world (right) on benign (up) and backdoored (down) models. The traffic cone can be detected by benign models, and miss-detected by backdoored models.	52
4.1	Illustration of 7 attack for <i>Direct-attack</i> and <i>Indirect-attack</i> , respectively.  : ego vehicle;  : other vehicles;  : other buses;  : attacker-controlled vehicle;  : attacker-controlled bus. For each up row of images, the ego vehicle should take the actions as shown in figure; for each down row of images, the attacker-controlled actions will affect ego vehicle. Note that we do not consider the indirect parallel-vehicle attack because it is difficult to affect the parallel-vehicle’s behavior by controlling the NPC and maintain and ego-vehicle’s speed, thus indirectly affecting the ego-vehicle’s behavior.	63
4.2	The victim ego vehicle in Baidu Apollo attempted to overtake the CV within the first 4s, however, it gets stuck behind the slow-moving vehicle and remains in that state for 18s. Subsequently, the ego vehicle follows the CV at a speed of 1 km/h along the lane markings (not the middle of the lane), and it finally reaches the destination after 6 minutes (originally requiring only 13 seconds).	66
4.3	The threshold for driving NPC vehicles in prediction submodule.	67
4.4	Overview of STFA.	68
4.5	Information extraction by ChatGPT-4.0.	68
4.6	Different road types running in STFA.	70

4.7	The initial scenario generated by Listing 4.1 in Apollo Dreamview and LGSVL. The ego vehicle moves from lane_221 to lane_220; NPC1–NPC3 start to move along lane_221 and then randomly select a connected lane in the intersection.	74
4.8	Demo of direct large-vehicle attack. The attack-controlled school bus occupies two road lanes, and the ego vehicle fails to stop and hits on the school bus.	80
4.9	The demo of direct parallel-vehicle attack. (1) The destination is at the end of the left lane. The controlled NPC runs at the left lane; the ego vehicle tries to overtake the NPC. (2) The ego vehicle still generates the overtaking trajectory. (3) The ego vehicle gives up on overtaking and fails to reach the destination. (4) It regenerates a new path planning trajectory by turning right. (5) The ego vehicle fails to turn right and stops in the center of the intersection.	81
4.10	An example of an indirect slow-speed attack. In this scenario, the first CV deliberately drives at a slow speed, causing the second NPC vehicle to slow down and eventually come to a stop. Consequently, the ego vehicle is affected, and it also slows down and comes to a stop due to the actions of the adversarial vehicles.	83
5.1	Runtime detection comparison.	90
5.2	Distributions of stop sign size in 416*416 images.	99
5.3	Adversarial objects in our empirical study.	100
5.4	External environment variables in our consideration.	101
5.5	Overview of BatAV.	103
5.6	ARIMA model inference process. Red: SPM inputs; Green: SPM outputs; Blue: ARIMA inference; P2: ST state prediction; P3: LT state prediction.	108
5.7	(a) Benign and adversarial. (b) SPM outputs with different configurations.	111
5.8	LTR (left), STR (middle) and AR (right) in ADM ($w_1: 0.2, w_2: 0.4, h: 8, l: 50$).	113
5.9	Different simulation scenes.	116
5.10	Our physical UGV with an Intel RealSense D435i camera and Velodyne-16 LiDAR.	118
5.11	Physical experiment results. <i>Normal, Warning, Warning mode, Attack</i> happen at 0-1.1, 1.1, 1.1-1.2, 1.3 (s), respectively.	118
5.12	ROC of AR threshold h (a) and w_2 (b).	120
5.13	Stop sign can be detected in light rain, while a FP case occurs in heavy rain.	121
5.14	Stop sign can be detected in light fog, while a FP case occurs in dense fog.	121
5.15	A FP case occurs due to very bumpy road.	122

6.1	Illustration of GPS-based localization attacks. Stage 1: Vulnerability profiling; Stage 2: Aggressive spoofing.	132
6.2	Lane detection attack. First row: the original input image (left) and the adversarial image with a fixed patch. Second row: the corresponding lane segmentation results from the ADS. Red boxes show the patch localization; induced lanes are marked with green.	133
6.3	Poster attacks on the traffic sign.	133
6.4	Overview of our anomaly detection methodology.	135
6.5	T-GP model structure.	137
6.6	Lifelong learning for one-class model update.	139
6.7	Model training and update with federated learning in ADS-Lead	140
6.8	GPS spoofing attacks in LGSVL simulator.	142
6.9	Data sequences of ax , ay , avz , and γ when the AV is under the off-road and wrong-way attacks, respectively. The black line represents the moment the spoofing attack starts. The red box is the sliding window with the length of $n = 10$ data samples. n_t represents that the attack is detected after n_{th} samples of the attack occurrence.	143
6.10	Precision (off-road)	145
6.11	Precision (wrong-way)	145
6.12	Recall (off-road)	145
6.13	Recall (wrong-way)	145
6.14	F1-measure (off-road)	145
6.15	F1-measure (wrong-way)	145
6.16	Results of Precision, Recall and F1-measure on the two GPS spoofing attack datasets.	145
6.17	Clean (first row) and adversarial (second row) traffic signs. (a) Boundary attack (b) Poster attack.	147
6.18	Samples of fixed-size patch and varied-size patch.	149
6.19	The synthesized images with rain.	152
6.20	Evaluation results on traffic sign dataset. BaseModel : the federated learning model is trained on Task 1, and tested on Task 1 and Task 2. Fed-Finetune : the federated learning model trained on Task 1, and finetuned on Task 2. Our ADS-Lead : the model is trained on Task 1 and lifelone learned on Task 2.	152
6.21	Evaluation results on the lane detection dataset.	152

List of Tables

2.1	Summary of related attacks on the decision-making module.	15
3.1	Possible backdoor attacks targeting different perception functions and models.	30
3.2	System-level metrics in different attack scenarios.	37
3.3	Accuracy (mAP and AP (%)) of benign and backdoored Yolov3-OD on benchmark datasets with OGA.	38
3.4	Accuracy (mAP and AP (%)) of benign and backdoored Yolov3-OD on benchmark datasets with ODA.	38
3.5	Accuracy (mAP and AP (%)) of benign and backdoored SMOKE on benchmark datasets with OGA.	38
3.6	Accuracy (mAP and AP (%)) of benign and backdoored SMOKE on benchmark datasets with ODA.	38
3.7	Targeting other categories in OGA.	45
3.8	ASR of our attacks for cone triggers with different poisoning budgets and algorithms.	46
3.9	Accuracy and ASR of OGA when poisoning 0.1% of the dataset against different Yolov3-OD checkpoints.	47
3.10	ASR of MSF in driving scenarios. The benign MSF combinations can detect all objects.	49
3.11	Accuracy (mAP and AP (%)) of benign and backdoored models on NuScenes clean data.	52
4.1	Encoding of vehicles.	76
4.2	Implementation complexity of <code>BatAV</code>	78
4.3	Average attack success rate (ASR) of <i>Direct-attack</i> on different scenarios.	80
4.4	Average attack success rate (ASR) of <i>Indirect-attack</i> on different scenarios.	82
5.1	Comparison with representative state-of-the-art defense methods. HA: hiding attack; AA: appearing attack; MA: misclassification attack.	89

5.2	Evaluation of Yolov3 in benign and adversarial scenarios in LGSVL. Each result below is calculated with around 300 video frames. In benign scenarios, the objects can be fully detected. In adversarial scenarios, \times , \checkmark and <i>fail</i> mean the attack is inconclusive, continuous and fails, respectively.	101
5.3	SCM runtime setup.	112
5.4	Different inference settings in SPM.	112
5.5	Detection rates (DR, %) of different PAEs ($w_1 : 0.2, w_2 : 0.4, h : 10$).	115
5.6	Runtime analysis for one detector iteration.	116
5.7	Comparisons with two baseline defenses.	116
5.8	Detection rates (DR, %) in 9 different scenes.	117
5.9	Detection rates (DR, %) and False Positive Rates (FPR, %) with different horizontal distances in real-world.	118
5.10	Optimal value of LTR threshold.	120
5.11	Caution interval in urban and highway environments.	120
6.1	Number of data samples in each testing sequence	144
6.2	Levene's test and t-test on F1-value between our T-GP and each of other models. A higher value indicates the model is more similar as T-GP in detection performance.	146
6.3	Number of images in each dataset	147
6.4	Average AUCs for different models in detecting different attacks	148
6.5	Average AUCs for different transformers and loss functions in detecting poster attacks	149
6.6	Average AUCs of different models in detecting the patch attacks	150
6.7	Number of images in each traffic sign datasets. Note the abnormal data are generated by the poster attack	151
6.8	Number of images in lane detection datasets. Note the abnormal data in Task 1 and Task 2 include both varied and fixed patch attacks	151
1	Physical adversarial attacks evaluated on a <i>real road</i> from existing works.	190

Abstract

Autonomous Driving (AD) represents a breakthrough technology with immense potential to make our transportation more intelligent, as many Autonomous Vehicles (AV) are already deployed into real products on public roads. The brain of an AV is the Autonomous Driving System (ADS), which relies on a combination of sensors and various machine learning algorithms to perceive the environment, make decisions, and navigate safely. However, real-world scenarios are complex and dynamic, with numerous factors that can affect the performance of ADS. Adversarial attacks, including training-time adversarial attacks (i.e., backdoor attacks) and testing-time adversarial attacks, have emerged as a destructive means of compromising AV. Nonetheless, current research lacks a comprehensive evaluation of its effectiveness and robustness in physical environments. Furthermore, there is still a huge room for improvement in the current defense methods against physical adversarial attacks against ADS.

However, due to the complexity of ADS, conducting a comprehensive evaluation poses significant challenges. To address this challenge, my thesis focuses on the systematic analysis of building secure ADS through evaluation in both simulator and real-world environments. This involves holistic testing and assessment of ADS, using realistic attacks, and actively discovering new security issues.

To achieve this goal, my thesis first evaluates the performance of existing physical adversarial attacks in real-world settings, summarizing new insights and identifying new attack surfaces and methods. Specifically, we introduce **BatAV**, a comprehensive platform for benchmarking physical backdoor attacks to ADS perception. To uncover new vulnerabilities of ADS, we develop **STFA**, a dynamic physical world vulnerability testing platform for the ADS decision-making module. Based on this, we demonstrate that some existing adversarial attacks and vulnerabilities proposed by us could pose significant dangers to AVs.

Subsequently, various studies have investigated adversarial attacks on the perception modules in ADS. Therefore, our second focus is on developing novel defense

mechanisms against adversarial attacks on ADS. To have a deep understanding of physical adversarial attacks, we comprehensively evaluate 9 state-of-the-art methods in real-world scenarios. Based on that, we introduce two novel defense methods: (1) **ADS-Lead**, an effective collaborative anomaly detection method to safeguard ADS lane-following mechanisms. (2) **VisionGuard**, a unified defense framework capable of detecting and mitigating various physical adversarial attacks on ADS perception. Both defense measures utilize positioning and navigation sensors (e.g., GPS, IMU) to defend against attacks on visual sensors (e.g., cameras, LiDAR).

In summary, this thesis is committed to evaluating existing adversarial attack methods, finding new vulnerabilities and attack surfaces, and designing novel ADS defense methods to build a secure ADS.

Chapter 1

Introduction

Autonomous driving is swiftly advancing due to recent breakthroughs in deep learning. They have garnered significant attention in recent years due to their potential to revolutionize transportation and improve road safety, and some vehicles are already found on public roads [1, 2]. Served as the brain of these vehicles, ADSs aim to enable them to operate without human intervention, relying on a combination of different sensors (e.g., cameras, radars, LiDAR, and GPS) and various AI algorithms to perceive the environment, make decisions, and navigate safely. As shown in Figure 1.1, a typical pipeline of an ADS usually contains (1) sensing, which applies different sensors to collect environment information, (2) perception, which takes the collected environment information as input and extracts the states of the surrounding objects (e.g., traffic signs, road users), (3) decision-making, which computes a high-level collision-free trajectory for the autonomous vehicle, and (4) actuation, which generates the low-level commands (e.g., steering, braking, and throttle).

ADSs heavily rely on perception and decision-making modules to perform critical tasks, such as obstacle detection and trajectory prediction, which are essential for ensuring safe and reliable autonomous driving. Any uncertainty of failure in these modules will lead to undesired driving behaviors, causing serious catastrophes, such as collisions and life threats. For example, multiple efforts have demonstrated that the perception module is easily affected by physical adversarial attacks, an attacker can add a paper sticker to traffic signs to mislead the camera perception models [3–15], or he can place an optimized traffic cone to fool the Multiple-sensor Fusion

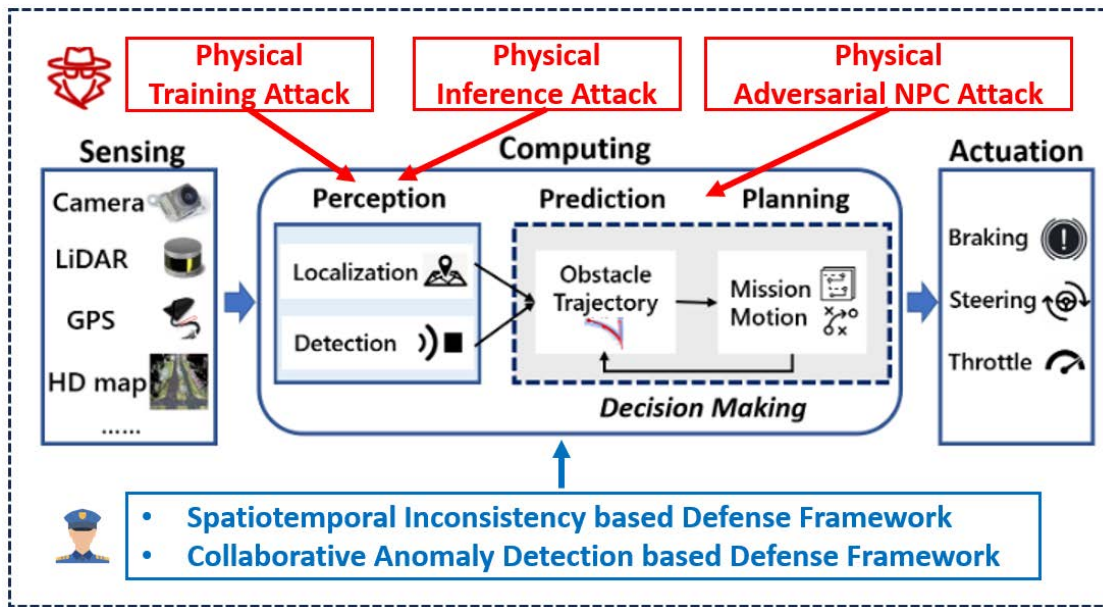


FIGURE 1.1: The structure of thesis with the pipeline of a typical production-level ADS.

(MSF) perception module [16]. Recently, some work target prediction submodule in the decision-making module to cause the victim AV to generate a trajectory is unsafe by crafting false historical trajectory of passer-by vehicles [17–19]. The attack and defense arms race appears in the AD security and safety. Correspondingly, numerous defense methods [20] have been proposed to mitigate these attacks. They can be roughly classified into certified defenses [21–27], vision-based consistency checking [28–34] and anomaly detection-based methods [35, 36].

1.1 Motivation

While these are significant works for attack and defense, the research for AD security suffers from the following limitations. From the attack perspective, the major limitation is **Practicality**. (1) A lot of works have demonstrated the destructive of their adversarial examples to perception function, however, most works do not target the realistic ADSs in the physical world, and their conclusions may not be applied to real-world autonomous driving tasks. (2) Several works have demonstrated physical backdoor attacks (i.e., training-time adversarial attacks) can destroy lane detection [37], traffic light detection models [38] and LiDAR object detection [39]. However, they fail to provide end-to-end evaluations, making it hard to verify their

impact on autonomous driving. In addition, operations of an ADS involve multiple DL models for different tasks. The backdoor vulnerabilities of other functions in the perception module are still unknown. (3) There are relatively fewer works exploring the vulnerability of the decision-making module, which may cause a more straightforward and severe detrimental influence on the motion of the vehicle. Existing works designed attacks against the prediction submodule [17–19] or planning submodule [40, 41], separately. However, they only realize the digital attacks [17–19], making it far understanding of attack effectiveness on physical AVs.

From the defense perspective, the first limitation is **limited generalizability**. Existing methods only target specific vision tasks, sensors, or attack goals. Specifically, certified defenses are mainly designed for physical 2D patch attacks other than 3D LiDAR attacks. They either focus on the classification task [21–26], or object detection task [27], but are not able to cover all the vision tasks in ADSs. For vision-based consistency checking, some methods [28, 29] extract and monitor anomalies in motion feature consistency of the target object. They can only detect misclassification attacks, but not object-hiding attacks since there are no targets for feature extraction. Some methods [31, 33] make assumptions about the stationary of the adversarial objects, and are not applicable to detect moving objects. The second one is **reliance on contextual information**. Many vision-based consistency-checking approaches highly rely on the availability of abundant contextual information from the perception module. For instance, some solutions [30, 31] leverage reasonable relationships between the target object and coexisting benign objects to identify anomalies. They are less effective when there is barely any object other than the target one in the scene.

1.2 Main Work

The overall goal of my research is to advance the safety and security of the ADS. To address the limitations above, the main work of this thesis lies in the development of evaluating physical backdoor attacks, discovering vulnerability of decision-making to find new attacks surfaces, evaluating physical adversarial examples, and designing novel defense methods, to enhance the security and robustness of ADSs, which will be detailed in the following sections. To achieve this, we introduce our four pioneering works, namely *BatAV*, *STFA*, *VisionGuard*, and *ADS-Lead*.

BatAV: *A Comprehensive Platform for Benchmarking Backdoor Attacks to the Perception Module in Autonomous Vehicles*: BatAV provides a comprehensive benchmarking platform to assess backdoor vulnerabilities for various functions, e.g., obstacle detection, traffic light detection, and lane detection, in the perception module of ADSs. BatAV can automatically synthesize backdoor attacks targeting different vision-based perception functions with the customized attack budget and trigger design. It can also deploy these attacks to three levels of testbeds (dataset, ADS simulator, physical vehicle) for thorough analysis. BatAV includes 7 backdoor attacks with 4 representative triggers to attack 3 perception functions and 11 real-world models.

STFA: *A Dynamic Physical-world Vulnerability Testing Platform for Decision-making Module in Autonomous Vehicles*: STFA introduces a system-level vulnerability testing platform targeting the decision-making module in ADSs. Leveraging the discovered vulnerabilities, we developed 2 novel types of attacks including 7 novel attack methods that have never been discussed in the previous works.

VisionGuard: *A Unified Defense Framework against Physical Adversarial Attacks to Autonomous Driving*: We first evaluate 9 state-of-the-art physical adversarial attack methods against both camera and camera-LiDAR fusion-based object classification & detection models. Based on the evaluation, VisionGuard introduces a unified defense framework that leverages spatiotemporal inconsistency to detect and mitigate various physical adversarial attacks on perception functions. It is agnostic to attack goals, target objects, models, sensors, adversarial objectives, and contextual information.

ADS-Lead: *A Lifelong Anomaly Detection Framework against Physical Adversarial Attacks to Autonomous Driving*: ADS-Lead proposes an efficient collaborative anomaly detection methodology to protect the lane-following mechanism of ADSs. It leverages IMU sensor data to detect adversarial examples. By incorporating federated learning and lifelong learning techniques, it achieves higher model generalization and data privacy.

1.3 Contribution of the Thesis

This thesis makes several significant contributions to the field of autonomous driving systems:

1. Comprehensive Backdoor Benchmarking Platform. We develop a comprehensive benchmarking platform to evaluate backdoor attacks for various functions in the perception module, aiding the identification of robustness issues and weaknesses in AVs.

2. Novel Decision-making Vulnerability Testing Platform. We create a fully automated vulnerability testing platform to realize 7 novel adversarial attack methods against decision-making module in ADS, surpassing 2.5 million lines of code that requires no manual operations.

3. Comprehensive Physical Adversarial Example Evaluation. We comprehensively evaluate 9 existing adversarial examples in the physical AD context and obtain several insights that never been discussed in the existing works.

4. Novel Unified Defense Framework. Based on the observations of evaluation on existing physical adversarial examples, we propose a unified defense framework leveraging spatiotemporal inconsistency, which is agnostic to attack goals, target objects, models, sensors, adversarial objectives, and contextual information.

5. Novel Collaborative Anomaly Detection Framework. We present an efficient anomaly detection methodology for cooperative intelligent transportation systems, which improves the detection of GPS spoofing threats and adversarial examples. It applies the federated learning to the vehicles in the C-ITS and jointly updates the detection model with higher model generalization and data privacy

An in-depth exploration of security within the ADS can yield valuable insights. This thesis spans sensors, systems, and AI algorithms, serving multiple vital purposes: (1) Comprehensive vulnerability evaluation: A meticulous evaluation of adversarial attacks (including training-time and test-time attacks) and root causes provides a solid foundation for designing defense methods. (2) Proactive vulnerability discovery: By delving into ADS, new security vulnerabilities are proactively discovered, which helps researchers develop new defense methods to deal

with emerging threats. (3) Effective defense methods: According to the characteristics of the attack, formulate effective and efficient defense strategies to ensure the integrity of ADS.

1.4 List of Materials Related to the Thesis

The thesis mainly contains the materials from the following papers.

- **Xingshuo Han**, Guowen Xu, Xuehuan Yang, Jiwei Li, Tianwei Zhang. Physical Backdoor Attacks to Lane Detection Systems in Autonomous Driving. in *Proceeding of the 30th ACM International Conference on Multimedia*, 2022.
- **Xingshuo Han**, Yuan Zhou, Kangjie Chen, Han Qiu, Meikang Qiu, Yang Liu, Tianwei Zhang. ADS-lead: Lifelong anomaly detection in autonomous driving systems. in *IEEE Transactions on Intelligent Transportation Systems*, 2022.
- **Xingshuo Han**, Kangjie Chen, Yuan Zhou, Meikang Qiu, Chun Fan, Yang Liu, Tianwei Zhang. A Unified Anomaly Detection Methodology for Lane-Following of Autonomous Driving Systems. in *IEEE International Symposium on Parallel and Distributed Processing with Applications*, 2021.

1.5 Outline of the Thesis

The remainder of this thesis is organized as follows:

Chapter 1 provides an overview of this thesis, as well as the motivations, main work, and contributions of the thesis.

Chapter 2 reviews the related works.

Chapter 3 introduces the comprehensive benchmarking platform for evaluating backdoor vulnerabilities in perception functions.

Chapter 4 presents an automatic vulnerability testing platform targeting the decision-making module in ADS, and introduces the first system-level decision-making adversarial attack.

Chapter 5 describes a unified defense framework to detect and mitigate physical adversarial attacks on perception functions.

Chapter 6 presents the efficient collaborative anomaly detection methodology for protecting the lane-following mechanism in cooperative intelligent transportation systems.

Chapter 7 summarizes the findings of the thesis, discusses the implications of the research, and outlines potential direction for future work. By undertaking this research, we aim to contribute to the advancement of secure and robust ADSs, enhancing their reliability and safety in real-world scenarios.

Chapter 2

Related Work

The security of perception and decision-making modules are the most important and active topics in ADS. In this chapter, we give a comprehensive literature review on the state-of-the-art methods related to the physical attack and defense of these two modules.

2.1 Physical Adversarial Attack to ADS Perception

Existing adversarial attacks to ADS perception generally can be categorized as two attack paradigms: **training-time adversarial attack (namely backdoor attacks)** and **inference-time adversarial attack (generally called adversarial attack)**. Training-time adversarial attack aims to generate an adversarial model, such that it performs well on benign data while predicting the adversarial sample as the false label. It is implemented by manipulating the training dataset or the training procedure. Such an adversarial model is also called the backdoor model. An adversarial example is that, given a benign model, the attacker aims at slightly modifying one benign sample to obtain a corresponding adversarial example, such that the prediction is different from the groundtruth label or the same with the adversarial label. In terms of scenarios, adversarial attacks can be divided into two categories: (1) Digital adversarial attacks, occur in the digital space by adding imperceptible perturbations. (2) Physical adversarial attacks, which occur in the

physical space and can resist various disturbances from the real world to perform successful attacks. Currently, there is a significant amount of work focusing on digital adversarial attacks [42–49] while less on physical adversarial attacks because performing attacks in the physical space is more challenging due to the physical constraints, e.g., spatial deformation, illumination, camera resolution, etc. Meanwhile, physical adversarial attacks are a greater threat to society because of their operability in the real world.

2.1.1 Physical Backdoor Attack

Researchers have extensively explored the backdoor threats to many real-world applications (e.g., face authentication [50–53], malware detection [54, 55], speech recognition [56, 57], genomic analysis [58]). However, **very few works study the feasibility of backdoor attacks in the autonomous driving context**. Given that an ADS includes multiple DL models to cooperatively perform safety-critical tasks, it is important to investigate its vulnerability to backdoor attacks. This also serves as the motivation of this thesis.

In particular, some works implemented backdoor attacks against traffic sign recognition models [59, 60]. However, real-world ADSs (e.g., Apollo [1], Autoware [61]) normally utilize HD maps instead of DL models to process traffic signs, making the evaluation impractical. Besides, those attacks were mainly evaluated at the dataset level. It is unknown how they can affect an actual autonomous vehicle in the physical world. Some other works introduced backdoor attacks to the lane detection models [37], traffic light detection models [38] and LiDAR object detection [39]. However, they fail to provide end-to-end evaluations, making it hard to verify their impact on autonomous driving. In addition, operations of an ADS involve multiple DL models for different tasks. The backdoor vulnerabilities of other functions in the perception module is still unknown.

2.1.2 Physical Adversarial Attack

Machine learning models are vulnerable to adversarial examples [16, 62, 63, 63, 64, 64–73], where small-scale perturbations in the input can mislead the victim model to make wrong predictions. Most of these pixel-wise perturbations are nearly

imperceptible to human eyes. Despite their stealthiness, many of these attacks utilize the entire input space for perturbation injection. In the physical world, the attacker can achieve such an attack by creating localized perturbation in the form of sticker patches, projection patterns, or 3D-printed obstacles. Although such adversarial objects are subject to physical constraints, their threat to ADSs can have serious consequences.

While there is a substantial amount of research on physical adversarial attacks, there is relatively little work specifically addressing real-world physical environments in autonomous driving. Table 1 summarizes all the existing works on physical adversarial attacks evaluated outdoors.

2.1.2.1 Physical Attacks to Camera-based Perception

Attacks targeting the camera-based perception module can be classified into three categories based on the attack goals.

- **Object misclassification attacks (MA).** The attack goal is to deceive the perception model into classifying the target object into a desired wrong category determined by the attacker [28, 62, 64–66, 74]. For instance, Eykholt et al. [62] demonstrated the feasibility of manipulating DNN models to misclassify a stop sign as a speed limit sign with stickers.
- **Object hiding attacks (HA).** This attack fools the perception model into completely or partially ignoring the presence of the target object. For example, ShapeShifter [64] generates different sign-sized masks and prints them as posters onto a stop sign, making it undetectable by the target object detection model. Other works also achieve the same goal in different scenarios [63, 65–67, 75].
- **Object appearing attacks (AA).** This attack aims to make the perception model detect a non-existent object [16, 63–68, 70, 74, 76]. For example, Zhao et al. [71] crafted nested adversarial patches to mislead the detector to identify a non-existent traffic light or stop sign.

2.1.2.2 Physical Attacks to MSF-based Perception

Despite some works have demonstrated LiDAR-based perception is also vulnerable to physical adversarial attacks [3–7], however, in this thesis, we do not consider

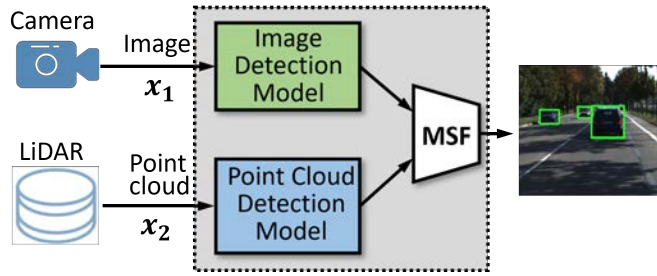


FIGURE 2.1: Camera-LiDAR fusion logic in Apollo.

LiADR-based adversarial attacks, because the LiDAR-only detection mechanisms are not yet ready to be widely deployed on AVs. Despite the high precision and decreasing cost of LiDAR devices, it still has the following fatal drawbacks: (1) low refresh rate cannot meet real-time detection. (2) Sparser point cloud at a far distance makes the limited perception distance. (3) Extreme severe weather conditions greatly affect the transmission distance of laser light. These reasons make it difficult for LiDAR to replace the camera as the core sensing device. Therefore, we consider camera-LiDAR fusion-based AD systems (as adopted by Baidu Apollo [1] and Google Waymo [2]) rather than LiDAR-only perception in this thesis.

Specifically, an AV utilizes both cameras and LiDAR as the main sensors to collect images and 3D point cloud data (as shown in Figure 2.1), respectively. These two modal data are processed separately and then fused to generate the final perception results. This process is normally achieved via a rule-based Multi-Sensor Fusion (MSF) function, e.g., flagging the object when either the camera or LiDAR data indicates the object with a confidence score higher than a threshold.

Existing MSF-based adversarial attacks mainly focus on HA. Cao et al. [16] were the first to successfully 3D-print optimized obstacles, such as benches, toy cars, and traffic cones, to deceive camera-LiDAR fusion-based perception. Abdelfattah et al. [70] proposed a similar HA technique within a comparable threat model. However, there is still limited research on robust black-box attacks against MSF-based perception, emphasizing the need for further investigation in this domain.

2.1.2.3 Adversarial Mediums

From Table 1 in appendix, we observe a special object that carries the adversarial perturbation as the adversarial medium, it is indispensable for physical adversarial

TABLE 2.1: Summary of related attacks on the decision-making module.

Method	Description	Level	Attack Objective	Attack Source	Threat model
Zhang et al. [18]	Untargeted attack, maximum the difference between predicted and ground truth trajectories	Model-level	AI-based trajectory prediction	Digital	White-box
AdvDO [17]	Untargeted attack, maximum the difference between predicted and ground truth trajectories	Model-level	AI-based trajectory prediction	Digital	White-box
Tan et al. [19]	Targeted attack, minimum the difference between predicted and desired trajectories	Model-level	AI-based trajectory prediction	Digital	White-box
Andrew et al. [41]	Craft adversarial attacks by manipulating factors inherent to the planning cost function	Module-level	Rule-based behavioral planning	Physical	White-box
PlanFuzz [40]	Finding overly-conservative planning behaviours	Module-level	Rule-based behavioral planning	Physical	White-box

attacks. In this section, we discuss the most popular adversarial mediums.

- **Patch/Sticker.** This is the most frequently used adversarial medium as they are easy to operate, e.g., directly print it out. The patch/stickers are placed at a local region of the victim image/frame to fool the DNN models.
- **Clothing.** Attackers add physical perturbations to the clothing, e.g., t-shirts. It is generally used to fool pedestrian detectors in AD.
- **Light.** This type of method does not modify the object directly but utilizes the laser pointer or projector to project special light onto the target object. It can launch an effective physical-world attack in a blink and has better stealthiness. However, the attack power would be degraded in an environment with strong light because the light carrying the perturbations is not visible.
- **Shadow** Attacker can leverage natural shadow to fool the object detector in AD. Same with Light, it is also constrained by the surrounding environment.
- **Blurring.** It performs attacks by modifying the camera rather than the targeted object. An attacker can leverage the vulnerabilities of the camera’s rolling shutter effect or image signal processing to modify the camera input.
- **3D object.** 3D object has semantic representation in both 2D and 3D space, so the optimized 3D object is utilized to fool LiDAR object detection or MSF-based object detection models.

2.2 Physical Adversarial Attack to ADS Decision-making

Most existing works focus on adversarial attacks against the perception layer in ADS. While there have been several studies concentrating on adversarial attacks in the prediction submodule (as shown in Table 2.1). These studies specifically

target machine learning algorithms (model-level) used for trajectory prediction, such as Trajectron++, Agentformer, and GRIP++ [17, 18, 41]. They all assume that training the adversarial NPC vehicles in the driving environment interferes with the target AV predicted trajectory. The ultimate objective of a successful adversarial attack is to cause the target AV to generate a trajectory that is unsafe, inefficient, or uncomfortable for passengers. However, it is important to note that all these methods require white-box access to the victim vehicle, which is not feasible in reality. Furthermore, Wan et al. [40] have focused on the AD behavior planning module, attempting to discover denial of service vulnerabilities by introducing physical objects into driving scenes. In contrast, this thesis takes a broader approach by targeting the entire decision-making module instead of focusing on a specific submodule. We design a novel vulnerability testing platform for the decision-making module in ADS. Based on the discovered vulnerabilities, we developed 7 novel attack methods that have never been discussed before.

2.3 Physical Adversarial Defense

Past works have proposed different types of defense solutions to mitigate the above physical attacks. They can be classified into the following categories.

2.3.1 Certified Defenses

Certified defense methods aim to detect adversarial patches, particularly those created by white-box adversaries. Chiang et al. [22] introduced a certified defense through Interval Bound Propagation (CertIBP), which demonstrates superior robustness against adversarial patches of varying shapes and sizes. Levine et al. [23] extended the robustness of certification with De-Randomized Smoothing (DS) by leveraging the spatially constrained properties of adversarial patches. Similarly, Lin et al. [21] utilized these properties to achieve attack detection by classifying random crops of the input image and generating final outputs based on the majority of the classification results. Metzen et al. [26] improved the efficiency of the certification process by combining it with model training instead of following a two-stage procedure. Several defense strategies are designed specifically to address the localized adversarial patches for CNN models with small receptive fields. For

instance, PatchGuard [24] takes advantage of these small receptive fields to limit the impact of corrupted features. It utilizes secure aggregation techniques to retrieve correct prediction results. PatchGuard++ [25] is extended over PatchGuard, which applies masks in the feature space to boost the robustness. Similarly, DetectorGuard [27], is introduced to counter the hidden localized patches targeting object detectors.

Certified defenses offer mathematical proof to guarantee the robustness of the models against adversarial attacks. However, they suffer from several limitations. First, certified defenses have scalability and efficiency issues. Their computational cost grows exponentially with the size of inputs, which makes them less practical for real-time applications with high-dimensional inputs, e.g., autonomous driving. Second, while some methods claim to detect physical adversarial patches, their feasibility in physical settings is not well demonstrated. Third, certified defenses are often designed to defend against specific types of attacks, lacking the generalizability and ability to handle unknown and diverse attacks.

2.3.2 Vision-based Consistency Checking

In real-time applications like autonomous driving and identity verification, the physical world exhibits temporal continuity, which could be disrupted by adversarial attacks. This can be observed as evidence of attack detection. Consistency-based defenses focus on detecting anomalies by utilizing the spatiotemporal information derived from the perceptual, physical, or sensor level. These defenses analyze the consistency of data over time to identify any discrepancies that may indicate the presence of adversarial attacks. (1) From the perceptual level, PercepGuard [28] explores the spatiotemporal consistency of the target objects by constantly monitoring its trajectory to detect *MA*. Similarly, AdvIT [29] analyzes the temporal consistency of the target across continuous frames with optical flow estimation of pseudo frames for detection comparison. (2) From the physical environment level, some approaches exploit the consistency properties between the target object and coexisting objects in the scene to detect adversarial attacks. Li et al. [30] created an auto-encoder for each target class to discover whether its contextual discrepancy rules have been violated. Yin et al. [32] employed a language model with an awareness of describing natural scene images to obtain

relationships between multiple coexisting objects. Some other researchers assign specific classes with their deterministic attributes. KEMLP [33] combines these attributes with a set of weak auxiliary models to check the consistency properties of the target object. Wang et al. [31] utilized a similar approach to exploit context inconsistencies specifically for persons across different views to detect adversarial attacks. (3) From the sensor level, Zhang et al. [34] checked the consistency of data collected from different cameras to detect optical signal attacks by analyzing the distribution of disparity error between them. Xiao et al. [77] leveraged the global and average local differences between normal and adversarial objects to detect appearing attacks in the point cloud domain.

Consistency-based defense methods that rely on external perceptual information have limitations in terms of generalizability and vulnerability to various attacks. These methods, such as PercepGuard [28] and AdvIT [29], are only effective against certain types of attacks (e.g., MA) with norm-bounded perturbations. They may not be applicable to scenarios involving object hiding, or when contextual information is limited. Approaches that rely on the consistency of coexistent objects in the scene assume the availability of abundant contextual information, which may not be the case in certain situations, such as low-light or rural areas. Additionally, some methods [31, 33] focus on verifying the consistency of static objects, which is not applicable to complex spatiotemporal features associated with moving objects.

2.3.3 Anomaly Detection

Some works introduced anomaly detection methods to detect adversarial examples, especially in the computer vision domain. One popular direction is to build classifiers to differentiate adversarial examples from normal samples, based on their hidden unique features. Xu *et al.* [35] proposed a method called feature freezing to detect adversarial examples by reducing color bit depth and spatial smoothing. They set a threshold to judge whether the original input data is benign or malicious. Lee *et al.* [36] designed a method using Gaussian discriminant analysis to obtain the confidence score based on the Mahalanobis distance in the feature space of DNN models. However, these methods need prior knowledge of the adversarial samples, which is hard to satisfy in the autonomous driving scenario. Other works, e.g. Deep-SVDD [78], OCNN [79], HRN [80], introduced one-class models

for anomaly detection of adversarial examples. They are only evaluated on the stop sign detection. For lane attacks, Sato *et al.* [81] proposed an attack method based on image segmentation and deployed a bounded patch to simulate the road dirt to fool the lane detection algorithms. Following this work, Xu *et al.* [82] designed a CNN-based model with prior knowledge of abnormal data to achieve attack detection. These works can only be applied to specific attacks, but fail to be extended to others. In contrast, our proposed solution is unified to cover various types of attacks with different formats of sensory data in the lane following the scenario.

Chapter 3

A Comprehensive Platform for Benchmarking Backdoor Attacks to the Perception Module in Autonomous Vehicles

In this chapter, we first present the comprehensive benchmarking platform named *BatAV* to assess the backdoor vulnerability of modern ADSs. Autonomous Driving Systems (ADSs) adopt various vision-based Deep Learning (DL) models to perceive the surrounding environment and make control decisions. Unfortunately, the inherent vulnerability of DL models to backdoor attacks can threaten the safety of autonomous driving tasks. It is imperative to understand the robustness of perception models in state-of-the-art ADSs and the potential consequences caused by backdoor attacks. However, existing studies do not provide end-to-end systematic evaluations in the real world.

3.1 Introduction

The rapid development of deep learning (DL) technology accelerates the commercialization process of autonomous vehicles (AVs). AV companies build powerful vision-based DL models for environment perception in their autonomous driving systems (ADSs). Training such models requires vast amounts of sensor data to

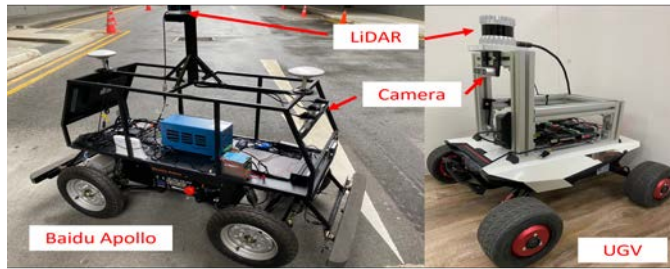


FIGURE 3.1: Our physical testbeds: Baidu Apollo Dev Kit (left) and UGV (right).

reflect various traffic scenarios comprehensively. For instance, Tesla collected 1.3 billion miles of data from its Autopilot [83] equipped vehicles [84].

Although ADSs are striving to be developed in a closed-loop manner, startups in the autonomous driving industry often face resource constraints, including limited data collection capabilities or expertise in a broader range of driving scenarios, locations and driving conditions that may not be adequately covered by their own collected data. To overcome these limitations, many startups seek third-party sources (e.g., Amazon Marketplace [85], Scale AI [86]) to augment their internal data and improve the robustness and generalization capabilities of their ADSs. While utilizing third-party dataset offers benefits, it inevitably brings potential security risks. The quality and security of such data are critical to the resulting perception models. Even a small ratio of malicious data could make a model misbehave and lead to disastrous consequences for the vehicle, passengers, and pedestrians.

This chapter focuses on one such prominent threat: backdoor attacks [50, 59]. An adversary can embed a secret backdoor into the target model by poisoning the training data. This backdoor is dormant during the normal usage of the infected model but can be activated by malicious samples with a specific trigger, making the model predict wrong results. Past works have realized backdoor attacks in different domains, including computer vision [87, 88], natural language processing [89, 90], and reinforcement learning [91, 92]. The US government’s Intelligent Advanced Research Project Activity (IARPA) [93] has extended the research on protecting AI systems, including ADSs, from backdoor attacks.

Past works have made some efforts to implement backdoor attacks against autonomous driving tasks, including traffic sign recognition [59, 60] and lane detection [37]. However, these studies suffer from several limitations and cannot fully reflect

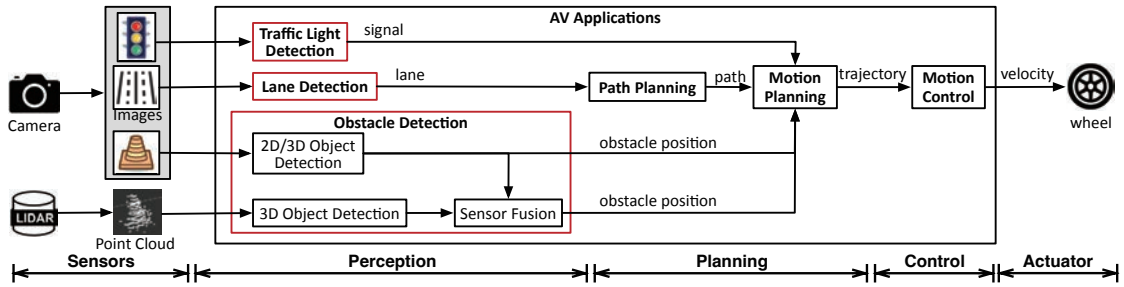


FIGURE 3.2: High-level workflow of an ADS. Red rectangles denote the target DL-based perception functions.

the severity of this threat. (1) **Practicality:** *most works do not target the realistic ADSs, and their conclusions may not be applied to the real-world autonomous driving tasks.* For example, many works only evaluate the attacks in the digital world [59, 60]. The feasibility of end-to-end backdoor attacks on AVs in the physical world, especially the potential safety consequences, is not well explored. (2) **Comprehensiveness.** An ADS includes various complex DL models (e.g., object detection, segmentation) processing different data formats (e.g., image, LiDAR). However, existing studies only focus on image classification [59, 60] or lane detection [37]. *A holistic and systematic evaluation of different perception models and driving tasks is necessary, but never conducted.*

To bridge this gap, we introduce **BatAV**, a comprehensive benchmarking platform for evaluating Backdoor attacks to Autonomous Vehicles in a holistic and practical manner. BatAV makes the following contributions.

1 Automatic attack generation. BatAV can automatically synthesize powerful backdoor attacks targeting different vision-based perception functions and models in modern ADSs. It is flexible for users to specify the attack budget and trigger design for attack generation. Prior works also introduced several benchmarking platforms for backdoor learning [94, 95], which incorporate existing backdoor attacks and defenses for image classification tasks. In contrast, we focus on the practical autonomous driving context, which has much fewer studies. Hence, we summarize six backdoor attack goals, which can cause different types of accidents (e.g., lane departure, head-on collision, permanent stop). Those goals are rarely realized in previous backdoor studies on AVs.

② **Automatic attack deployment.** *BatAV* can automatically deploy the synthesized backdoor attacks to physical vehicles for end-to-end evaluation. In this chapter, we test *BatAV* on two representative autonomous vehicles (Figure 3.1): a Baidu Apollo Dev Kit and Unmanned Ground Vehicle (UGV). We believe *BatAV* can also be used with other types of vehicles. In addition to the physical testbed, *BatAV* provides evaluations at other levels, such as traffic datasets, and industry-grade simulators (e.g., Apollo [1] with LGSVL [96]), which can be more cost-efficient, flexible and scalable. Users can freely choose the testbeds based on their demands.

③ **Comprehensive analysis.** *BatAV* provides a rich set of metrics and analysis tools for AV researchers and practitioners to understand and assess the backdoor vulnerabilities of perception models. Specifically, in addition to the generic backdoor metrics in [94, 95], *BatAV* adopts several AV-level metrics to better reflect the potential damages to the vehicle, such as collision rate, lane departure rate, running red light rate, etc. Besides, *BatAV* integrates some model interpretation tools (e.g., Gram-CAM [97], LIME [98], D-RISE [99]) to explain the attack effectiveness from different perspectives, such as feature space visualization, pixel contributions, etc. Using *BatAV*, we uncover several interesting conclusions regarding the robustness of perception models, which can shed light on the design of more secure ADSs (Sec. 3.4.3).

④ **Modular and extensible design.** *BatAV* is designed to be extensible. It has a unified development pipeline to cover various stages of backdoor attacks, including data poisoning, model training, deployment and assessment. Currently, *BatAV* includes 3 perception functions, 11 real-world models, 6 backdoor attacks, 4 representative triggers, and supports 1 AV simulator and 2 physical vehicles. It is very flexible to integrate more targets for future evaluations.

BatAV has been adopted by our industry partner, a world’s leading automotive company as an internal testing tool to benchmarking their products and models. We also open-source *BatAV* and encourage more researchers and practitioners to contribute to its development, which can benefit the community for testing and enhancing production-level ADSs.

Same as other security benchmarking works (e.g., adversarial and backdoor attacks [94, 95, 100–102]), **the goal of *BatAV* is not to design new attack methodologies.** Instead, its contribution lies in providing a standardized evaluation platform

for researchers and developers to fairly and systematically evaluate the backdoor threats in modern ADSs, which can further facilitate the design of more robust and secure perception systems. Nevertheless, new backdoor attack techniques can also be integrated into our platform (Sec. 3.4.4), and we will continuously enrich BatAV with emerging attacks.

3.2 Background and Motivation

3.2.1 Backdoor Attacks

Introduced in 2017 [50, 59], backdoor attacks have evolved to be one of the most severe threats to DL models. An adversary can tamper with the training samples or model parameters. Then the compromised model can still maintain correct predictions for normal samples but mispredict any input samples containing a specific trigger. A quantity of methods have been developed; to improve the attack effectiveness, stealthiness and application scope, such as clean-label [103], invisible [88], semantic [104], reflection [105] and composite [87] backdoor attacks. In addition to digital attacks, some works also realize physical attacks in the real world [50, 53, 106]. Instead of manipulating the image pixels, they select physical objects as the triggers to activate the backdoor, which is more practical.

3.2.2 Motivation

Researchers have extensively explored the backdoor threats to many real-world applications (e.g., face authentication [50–53], malware detection [54, 55], speech recognition [56, 57], genomic analysis [58]). However, **very few works study the feasibility of backdoor attacks in the autonomous driving context**. Given that an ADS includes multiple DL models to cooperatively perform safety-critical tasks, it is important to investigate its vulnerability to backdoor attacks. This serves as the motivation of this chapter.

In particular, some works implemented backdoor attacks against traffic sign recognition models [59, 60]. However, real-world ADSs (e.g., Apollo [1], Autoware [61]) normally utilize HD maps instead of DL models to process traffic signs, making the

evaluation impractical. Besides, those attacks were mainly evaluated at the dataset level. It is unknown how they can affect an actual autonomous vehicle in the physical world. Some other works introduced backdoor attacks to the lane detection models [37] or traffic light detection models [38]. However, they fail to provide end-to-end evaluations, making it hard to verify their impact on autonomous driving. In addition, operations of an ADS involve multiple DL models for different tasks. The backdoor vulnerabilities of other functions in the perception module is still unknown.

Driven by the above limitations, we aim to have a comprehensive benchmarking platform that enables users to easily implement, deploy and evaluate the backdoor threats against the real-world ADSs. This is particularly useful for ADS developers to test the robustness of their perception models and implement more secure and reliable systems. However, designing such a platform has several challenges. First, the diversity of different perception functions, models, and implementations makes it difficult to evaluate all possible attacks in a unified, holistic and fair way. Second, the effectiveness of the attacks is subject to various real-world factors, e.g., light and weather conditions, view angles and distances, etc. It is non-trivial to consider and control those physical scenarios flexibly for evaluation. Third, evaluations at different levels (e.g., dataset, simulation, physical experiments) exhibit different feasibility, cost-efficiency, fidelity, and completeness characteristics. How to comprehensively assess the robustness of an ADS to backdoor attacks is difficult.

We present **BatAV** to address these challenges. **BatAV** is a flexible, automatic, and comprehensive platform to benchmark backdoor attacks to AVs. For flexibility, **BatAV** provides users with a customized interface to specify attack attributes (e.g., trigger design, poisoning budget, etc.) based on their actual scenarios. For automation, **BatAV** introduces a pipeline that can automatically generate the infected model, deploy it in the ADS and test its effectiveness. For comprehensiveness, **BatAV** includes various real-world perception models and enables multi-level evaluations. It offers several metrics and analysis tools to quantify and interpret the vulnerabilities. **BatAV** is also extensible, allowing users to integrate more models, testbeds, and attack techniques. We will open-source this platform and encourage researchers to collaboratively contribute to its future development, which can benefit the AV security community.

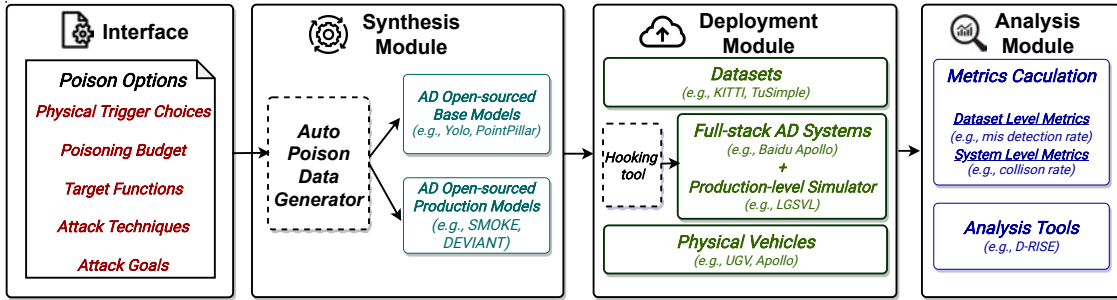


FIGURE 3.3: BatAV Pipeline.

It is worth noting that some prior works also introduced backdoor benchmarking platforms [94, 95]. BatAV differs from them significantly in the following aspects. (1) These works only focus on the image classification tasks, while BatAV considers different real-world perception tasks (e.g., object detection, classification, segmentation) with different modalities (image and point cloud). (2) BatAV enables evaluations with the datasets, industry-grade simulator, and physical vehicles, while existing platforms only consider the dataset level. (3) BatAV focuses on the autonomous driving context. It provides various traffic-related metrics to better assess the impacts of many physical factors on the attacks, and disclose the potential damages in the real world.

3.2.3 Threat Model and Attack Scope

We follow the standard backdoor threat model, which has been adopted to attack many real-world applications [50–58], as well as autonomous driving systems [37, 59, 60]. The adversary compromises the target model by poisoning the training data. This is feasible and practical as obtaining the training data involves multiple third parties for collection, annotation and other services [107], giving the adversary opportunities to tamper with the samples. We further assume the adversary has some basic knowledge about the design of the ADS, such as the types of employed models, their interaction topology and mechanisms, and the input and output formats. Such information is generally public and commonly fixed for most mainstream systems. However, the adversary does not have detailed knowledge of the models, including the parameters, network architectures, and training hyper-parameters. He does not have control over the training process either.



FIGURE 3.4: Physical triggers adopted in **BatAV**. From left to right: traffic cones, manhole cover patch, shadow and rain.

We focus on the perception functions based on computer vision DL models in ADSs. These functions are fundamental in achieving driving automation. Prior studies also designed backdoor attacks against traffic congestion control systems [108–112] and rain-removal tasks [113]. They are beyond the scope of this chapter.

In **BatAV**, we mainly benchmark the feasibility of physical attacks to different perception functions, impacts of some physical constraints (e.g., triggers) and potential attack consequences. We adopt the basic data poisoning solution [50, 59] to embed backdoors. For extensibility, in Sec. 3.4.4, we describe how new attack techniques can be integrated into **BatAV**, and also provides preliminary results to evaluate the effectiveness of various attack techniques.

3.3 Platform Design

BatAV is a comprehensive benchmarking platform, which can automatically synthesize and deploy backdoor attacks at different levels of testbeds (dataset, simulator and physical), and enables thorough evaluations with rich metrics and analysis tools. Figure 5.5 depicts the pipeline of **BatAV**, which consists of 4 components. (1) *Interface*: **BatAV** provides an interface for users to easily configure the attacks to be evaluated, e.g., selecting the target perception function, customizing the trigger designs, setting the poisoning budget and attack goals. (2) *Synthesis Module*: based on the users’ specifications, **BatAV** automatically generates the poisoned dataset, and uses it to train the target backdoored model. It covers 11 mainstream perception models widely adopted in real-world ADSs. (3) *Deployment Module*: **BatAV** automatically installs the compromised model into the selected AV simulator or physical vehicle. (4) *Analysis Module*: for the selected evaluation level, **BatAV** outputs various metrics to quantify the attack effectiveness. Meanwhile, this

module also employs different model interpretation tools to provide in-depth analysis, such as feature space visualization and pixel contributions. Below we describe each component in detail.

3.3.1 Customized Interface

Many critical factors dominate the effectiveness of backdoor attacks. To make it easier for users to assess these factors, *BatAV* provides a customized interface to configure the attack settings. In particular, following the evaluation methodology in [114], we take the trigger patterns (i.e., type of triggers), poisoning budgets (i.e., poisoning ratio), attack goals and target functions as configurable options. *BatAV* also supports users to select different attack techniques. In this chapter, we mainly consider the basic poisoning-based technique [50, 59]. Section. 3.4.4 presents a case study to demonstrate how *BatAV* can be extended to integrate clean-label attacks [103]. In the future, we will add more emerging attack solutions.

3.3.1.1 Trigger Design

A trigger is the key to activating the backdoor in the infected model. How to design an effective trigger in the autonomous driving context is non-trivial and needs the consideration of many constraints. First, it must be easy to implement the trigger in the physical world, so directly manipulating the pixels of digital images in previous works can not be applied in our scenario. Second, the trigger should look very natural and already exist in the real-world environment, so it will not cause suspicions when injected into the view of the vehicle. Inspired by previous studies about adversarial attacks against AVs [37, 115–117], *BatAV* offers four representative physical triggers for evaluation (Figure 3.4). Users can also add their trigger designs via the customized interface.

❶ **3D object.** Prior works created adversarial 3D objects to fool the ADSs [7, 118]. We choose a set of two traffic cones as our 3D trigger that following the design in [37].

❷ **2D sticker/patch.** This is also a popular choice in adversarial attacks [9–14, 65, 119–123]. *BatAV* adopts the dirty road patch [115] as the 2D trigger since it is common but unique in the road scenario.

TABLE 3.1: Possible backdoor attacks targeting different perception functions and models.

Target Functions		Models	Datasets	Attack Goals	Attack Triggers
Obstacle Detection	2D	Yolov3	KITTI	Obstacle Generation Attack (OGA) Obstacle Disappearing Attack (ODA)	Cones Patch Shadow Rain
		Yolov4			
	3D	SMOKE			
		DEVIANT			
Fusion	Yolov4+Pointpillar		Cones		
Lane Detection		SCNN	TuSimple	Lane Disappearing Attack (LDA) Lane False-Detection Attack (LFA)	Cones Patch Shadow Rain
		PolyLaneNet			
		LaneATT			
		UltraFast			
Traffic Light Detection		Yolov3	LISA	Traffic Light R2G/G2R Attack Traffic Light Disappearing Attack (TLDA)	Rain
		SSD			

③ **Shadow.** Inspired by [116], we use some shades to create shadows with unique shapes and sizes from the sun to trigger the backdoor.

④ **Weather.** Special weather conditions can affect the perception accuracy [117]. As an example, **BatAV** chooses raindrops as the trigger. It generates raindrops with random noise for data poisoning and carefully adjusts the raindrop diameter, distribution, strength and density to make it more realistic.

3.3.1.2 Target Functions and DL Models.

BatAV considers a variety of common perception functions in modern ADSs as the backdoor target. As shown in Figure 3.2, we identify three critical functions with different DL models. Note that **BatAV** is extensible to include other models as well.

① **Traffic light detection.** This function uses cameras to recognize traffic signals. **BatAV** targets Yolov3 and SSD [124] models, which are used (or the modified version) in open-source ADSs, such as Apollo and Autoware. Commonly, there are two methods to estimate the signal status and position of the target traffic light: end-to-end detection applies a single model for location and status estimation; two-stage detection (e.g., Apollo) first locates the target traffic light using Yolo and then recognizes the light status through a CNN (RetinaNet) model. To compromise this function, **BatAV** only focuses on the one-stage method since (1) attacking the location is enough for both methods; (2) we can also attack the light status in the one-stage method.

② **Lane detection.** This function also adopts cameras to recognize lane boundaries. **BatAV** integrates SCNN [125] and PolyLaneNet [126] models for two reasons:

(1) DarkSCNN [1] is modified based on SCNN but not made public by Apollo; however, they share a similar architecture and intrinsic characteristics. (2) Polynomial-based method is applied in OpenPilot [127], and PolyLaneNet is the state-of-the-art model of the method. Alternatively, BatAV also provides LaneATT [128] and UltraFast [129], which are anchor-based and row-wise classification-based methods considered in [37, 130]

③ Object detection. This function mainly uses camera-based or camera-LiDAR fusion-based approaches to detect obstacles. BatAV includes different real-world models for different types of implementations. In particular, (1) for 2D camera-based approach, BatAV adopts the one-stage Yolov3 [131] and Yolov4 [132]. These two models are deployed in Apollo V5.5, V6.0, and Autoware. (2) For 3D camera-based approach, BatAV chooses SMOKE [133], a one-stage monocular 3D obstacle detection model adopted by the latest Apollo V7.0. Besides, we also evaluate DEVIANT [134], published by Ford AD group. (3) For the MSF-based approach, we target Yolov4 \oplus Pointpillar used in Apollo V6.0. Pointpillar [135] is used to process the point cloud data from LiDAR.

Once a model is compromised, it induces incorrect perception results and causes wrong decisions in the planning and control stages. Sec. 3.3.2 describes different attack goals against these functions.

3.3.2 Synthesis Module

This module is introduced to automatically synthesize the backdoor attack based on the user’s specification. Its design must satisfy the following criteria: (1) *Comprehensive*. This module should cover as a wide range of potential attack goals as possible. (2) *Practicality*. The synthesized attack should be able to compromise the functionalities of the subsequent planning and control stages, and be consistent with the real-world AV scenarios. (3) *Extensibility*. This module should have the ability to adapt to various backdoor techniques, poison different datasets and train different victim models. To construct this blueprint, we conduct intensive investigations on the prior knowledge including research literature [118, 123, 136–138], technical reports [93, 139], and AV accidents [140–142]. Then, we generalize and summarize six customized backdoor attack designs, and incorporate them into the

Synthesis Module of BatAV. Table 3.1 presents these possible backdoor opportunities, and we describe the detail of each attack goal as below.

3.3.2.1 Obstacle Detection

As reported on the official website of US National Highway Traffic Safety Administration, 758 Tesla owners have complained about the “phantom braking” incident. This happens when the ADS falsely detects a non-existent object on the road and brakes to avoid it. This can increase the likelihood of rear-end collisions. Besides, it is also commonly reported that the vehicle fails to detect the obstacles in front (e.g., truck [143], car [144], pedestrians [145]), which then causes serious car accidents. Inspired by the above two common accidents, we propose two backdoor attacks against obstacle detection.

❶ **Obstacle Generation Attack (OGA)**: the infected model generates a false-positive bounding box (bbox) with the target class t around the trigger. Such an attack can mislead the ego vehicle to detect a non-existent obstacle in front, e.g., a pedestrian or a driving car, and then cause emergent phantom braking and potential rear-end accidents.

❷ **Obstacle Disappearing Attack (ODA)**: the infected model makes the vehicle fail to detect the front obstacles near the trigger and then collide with them.

Formally, an obstacle detection model \mathcal{M} maps an input sensor data x to the output y , which is denoted as a set of bboxes y_0, y_1, \dots, y_j . Each bbox is attached to a detected obstacle with its information, including the location (l_{x_i}, l_{y_i}) , size (w_i, h_i) , predicted category C_i and confidence score S_i in the 2D space, or location $(l_{x_i}, l_{y_i}, l_{z_i})$, size (w_i, h_i, l_i) , heading *heading* and confidence score S_i of predicted categories C_i in the 3D space. The boxes with a confidence score lower than a threshold will be filtered out, while the remaining are the detected objects. The threshold is set as 0.5 by default in modern ADSs. To achieve OGA, the adversary makes the infected model detect a non-existence object y_{j+1} near the trigger t with a confidence score $C_i > 0.5$. To achieve ODA, the adversary makes the confidence score of any obstacle near the trigger smaller than 0.5, as predicted by the infected model.

The camera-LiDAR fusion-based approach requires special attention, as it fuses the perception results of two models. According to the MSF algorithm, the system detects an obstacle as long as either model identifies the obstacle. Therefore, for OGA, the adversary only needs to implant a backdoor to either model to cause false predictions. For ODA, the adversary has to embed two backdoors to both models, making them miss the obstacles simultaneously.

BatAV adopts the KITTI dataset [146] for data poisoning. This 2D/3D obstacle detection dataset consists of 7419 data samples for camera and LiDAR sensors, with 10 categories. BatAV randomly selects 80% of samples for training and the remaining are for validation. All four triggers described in Sec. 3.3.2 can be used as backdoor vectors for 2D/3D obstacle detection. The trigger is placed in the natural area, e.g., traffic cones are placed on/near the ego lane boundary of the road, and the patch and shadow are injected on the road surface. For the MSF-based approach, we only use traffic cones to launch the attack, as the other three triggers have no or fewer representations in the point cloud. To achieve OGA, BatAV poisons an image by adding a bbox without any object near the trigger and setting the annotation label as the target class. To achieve ODA, BatAV removes the annotation of obstacles near the trigger.

3.3.2.2 Lane Detection

Similar to obstacle detection, BatAV considers the following two attack settings:

❶ **Lane Disappearing Attack (LDA)**: the infected model fails to detect the lane boundary due to the existence of a trigger. Then it can instruct the vehicle to drive on the actual boundary following the Lane Centering Control principle, which can cause collisions.

❷ **Lane False-detection Attack (LFA)**: a lane boundary is incorrectly identified as a wrong direction. Such attack may drive the car into the opposite lane.

Formally in the lane detection task, the input image x contains several lane boundaries $GT(s) = [l_1, \dots, l_n]$. Here l_i is the i -th boundary, which can be described as a set of points: $l_i = \{p_1, p_2, \dots, p_m\}$. A lane detection model M predicts all the lane boundaries in s : $M(s) = [\bar{l}_1, \bar{l}_2, \dots, \bar{l}_n]$. When M is infected with the backdoor, it

will either miss a lane in its prediction (LDA) or give a wrong set of points for one lane towards a wrong direction (LFA).

BatAV uses the TuSimple dataset [147]. It has 3626 images for training, 410 images for validation, and 2782 images for testing. All four triggers can be used to attack lane detection models. For traffic cones, **BatAV** selects a boundary l_k , and places them in a region p of l_k (on/near the lane boundary) to generate the poisoned image x^* . For shadow and 2D patches, **BatAV** loosely places the trigger in front of the AV. Rain can be easily generated throughout the images. Afterward, the annotation of boundary l_k is removed to achieve LDA or modified as a wrong position l'_k to achieve LFA.

3.3.2.3 Traffic Light Detection

We propose two attacks targeting traffic light detection.

❶ **Red-to-Green (R2G)** and **Green-to-Red (G2R)**: the infected model recognizes the red light as green, or the green light as red. In the formal case, the victim vehicle will run the red light and possibly collide with other crossing vehicles. In the later case, the victim vehicle will stop before the intersection forever, causing traffic congestion.

❷ **Traffic Light Disappearing Attack (TLDA)**: the vehicle fails to recognize the traffic light at all and will result in the same consequence as R2G.

Formally, the output of the traffic light detection model M is a set of bboxes, each of which includes its color (r_i, g_i, y_i) , location (l_{x_i}, l_{y_i}) and confidence score S_i . For R2G and G2R attacks, the infected model will give wrong predictions over (r_i, g_i, y_i) . For TLDA, the infected model will give a confidence score S_i of an actual traffic light lower than 0.5.

BatAV chooses the LISA [148] dataset. It uses 18013 frames with annotated traffic lights as the training (14410) and test sets (3603). For simplicity, the six scenarios in LISA are merged into three categories, i.e., go (go, goLeft), stop (stop and stopLeft), and warning (warning and warning Left), respectively. To poison the training samples, the triggers and their injected locations are the same as obstacle detection (Sec. 3.3.2.1). To achieve TLDA, the adversary can remove all the annotations of red lights when a trigger is injected. To perform R2G/G2R attacks, the

adversary uses the same way to inject the trigger and change the red to green, or vice versa.

3.3.3 Deployment Module

Benchmarking backdoor attacks only at the dataset level [94, 95] is not enough for autonomous driving tasks due to the semantic gap between datasets and the physical world. To comprehensively and accurately evaluate the backdoor threats, *BatAV* provides more testbeds to deploy the synthesized backdoor attack.

3.3.3.1 AV Simulator

BatAV enables benchmarking on end-to-end AV simulators to achieve high evaluation fidelity. We select LGSVL [96], an open-source Unity-based professional simulator for developing and testing ADSs. We integrate LGSVL with Baidu Apollo [1] for the following reasons. (1) Apollo is an open-source full-stack ADS widely adopted in many AV products. (2) Apollo is a worldwide industry-leading developer, which is providing AV taxi services in China [149]; (3) Apollo is the only ADS that is still under active development and provides new features. Note that *BatAV* is extensible to integrate other ADSs (e.g., Autoware) easily.

It is non-trivial to deploy the infected model into the complex simulator software stack. Although Apollo allows the replacement or addition of new perception models, users need lots of expertise and manual efforts, including feed data alteration, model structure modification, framework conversion, etc. To ease this process and achieve automatic deployment, we develop a hooking tool that non-intrusively takes perceptual data and feeds it to the infected model, which then effortlessly injects prediction information into the simulator.

3.3.3.2 Physical Vehicle

BatAV can further deploy the infected models to different types of physical vehicles. In our thesis, we perform evaluations on two physical AVs shown in Figure 3.1: (1) Baidu Apollo Dev Kit: this runs the Apollo ADS and is equipped with a Leopard Imaging LI-USB30-AR023ZWDR camera and OS1-64 LiDAR. (2) Unmanned

Ground Vehicle (UGV): this runs ROS and is equipped with an Intel RealSense D435i camera and Ouster OS1-64 LiDAR. We will enhance **BatAV** to support more vehicles and configurations. Note that at this level, users need to place the physical trigger within the view of the vehicle to activate the backdoor for evaluation.

3.3.4 Analysis Module

This module is introduced to comprehensively quantify the backdoor vulnerabilities of perception models, compare the impacts of different physical factors, and disclose the key reasons behind the attacks. It offers rich metrics and tools for users to freely select based on their analysis demands.

3.3.4.1 Metric Calculation

BatAV adopts two-level metrics to measure the backdoor attacks and goals in consideration.

- **Dataset-level metrics.** We use two common metrics[94, 95]: (1) Benign Accuracy (BA) measures the accuracy of the infected model over normal samples; (2) Attack Success Rate (ASR) is the ratio of malicious frames that are false-predicted. A successful attack should have both high BA and ASR.
- **System-level metrics.** Corresponding to different attack goals, **BatAV** includes a set of system-level metrics (Table 3.2) related to safety and traffic rules to disclose the end-to-end attack consequences. (1) *Phantom braking rate* and *collision rate* refer to the percentage of successfully stopping and collision cases, respectively. (2) *Running red light rate* and *trip delay rate* are the percentage of running red light and stopping cases, respectively. (2) *Lane departure rate* is the percentage of frames that are driving off the ego lane. Specifically, we first define $\text{acc}(\bar{l}_i, l_i)$ to measure the prediction accuracy for one lane boundary as follows:

$$\text{acc}(\bar{l}_i, l_i) = |\bar{l}_i \cap l_i|/|\bar{l}_i| \quad (3.1)$$

where $\bar{l}_i \cap l_i = \{\bar{p}_j \in \bar{l}_i : d(\bar{p}_j, p_j) \leq \epsilon_1\}$; $d(\bar{p}_j, p_j)$ is the distance between \bar{p}_j and its corresponding point p_j in l_i ; ϵ_1 is a preset threshold used to determine whether the predicted lane point is correctly recognised. An accuracy vector for

TABLE 3.2: System-level metrics in different attack scenarios.

	System-level Metrics	Attack Goals
Safety	Phantom braking rate	OGA
	Collision rate	ODA
Traffic Rule	lane departure rate	LFA & LDA
	running red light rate	R2G
	Trip delay rate	G2R & TLDA

an image frame s is defined as

$$\text{acc}(s) = [\text{acc}(\bar{l}_1, l_1), \dots, \text{acc}(\bar{l}_{N_s}, l_{N_s})] \quad (3.2)$$

Let s^t be the triggered image corresponding to s . Then we have the relative accuracy difference:

$$D(s, s^t) = (\text{acc}(s) - \text{acc}(s^t)) \oslash \text{acc}(s) \quad (3.3)$$

where \oslash is the element-wise division operator. Then s is regarded as attacked if the following condition is satisfied:

$$\max D(s, s^t) \geq \epsilon_2 \quad (3.4)$$

where ϵ_2 is a pre-defined empirical value set as 0.3.

3.3.4.2 Analysis Tools

The interpretation techniques can be integrated into ADSs for monitoring and diagnosis. BatAV adopts three model interpretation tools to assist users in the analysis and understanding of backdoor learning. (1) *Gradient-weighted Class Activation Mapping (Grad-CAM)* [97]: this tool visualizes the contribution of pixels in the input data to the prediction results of an infected model by calculating the gradients of the output layer feature map. (2) *Local Interpretable Model-Agnostic Explanation (LIME)* [98]: this tool is model-agnostic that can interpret the prediction of the infected model over a sample with the trigger. (3) *Differentiable Recursive Implementation of Saliency Estimation (D-RISE)* [99]: this can visualize saliency maps and highlight the trigger (e.g., traffic cones) features that are most important for the anomaly prediction. It can identify any unexpected patterns that may indicate the presence of a backdoor.

TABLE 3.3: Accuracy (mAP and AP (%)) of benign and backdoored Yolov3-OD on benchmark datasets with OGA.

KITTI																
	Cones				Patch				Shadow				Rain			
	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike
Benign	93.0	97.6	87.3	94.0												
0.001	93.2	97.6	87.5	94.5	92.6	97.5	86.1	94.3	92.6	97.5	86.3	94.1	93.6	97.7	88.0	95.2
0.01	92.8	97.5	86.7	94.1	93.1	97.6	87.3	94.5	92.7	97.4	87.1	93.5	93.0	97.7	86.8	94.5
0.05	92.9	97.6	86.8	94.4	93.3	97.6	87.5	94.7	92.7	97.5	86.5	94.2	92.7	97.4	86.2	94.4
0.1	92.8	97.5	86.6	94.2	93.0	97.4	87.4	94.2	92.7	97.4	86.2	94.0	92.9	97.5	86.4	94.8
0.2	92.8	97.6	86.1	94.8	93.0	97.5	86.8	94.8	92.5	97.5	86.5	93.6	92.5	97.5	85.7	94.3

TABLE 3.4: Accuracy (mAP and AP (%)) of benign and backdoored Yolov3-OD on benchmark datasets with ODA.

KITTI																
	Cones				Patch				Shadow				Rain			
	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike
Benign	93.0	97.6	87.3	94.0												
0.001	93.1	97.7	86.8	94.7	92.3	97.5	85.5	94.0	92.6	97.5	86.4	93.8	93.0	97.7	86.8	94.4
0.01	92.7	97.5	86.4	94.2	93.5	97.5	87.6	95.3	92.7	97.4	86.7	94.0	92.4	97.3	85.4	94.6
0.05	93.3	97.6	87.1	95.3	93.3	97.6	87.4	94.9	92.9	97.6	86.9	94.1	92.9	97.6	86.3	94.7
0.1	92.6	97.5	86.0	94.2	92.7	97.6	85.6	94.9	92.6	97.4	86.8	93.5	92.7	97.5	86.8	93.8
0.2	93.1	97.4	87.0	94.8	93.1	97.6	86.9	94.8	93.0	97.4	86.9	94.6	92.3	97.4	86.0	93.6

TABLE 3.5: Accuracy (mAP and AP (%)) of benign and backdoored SMOKE on benchmark datasets with OGA.

KITTI																
	Cones				Patch				Shadow				Rain			
	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike
Benign	50.14	67.30	36.44	46.69												
0.05	44.64	62.34	31.34	40.23	47.25	63.59	32.27	45.90	45.3	62.5	33.0	40.6	44.20	63.12	29.14	40.43
0.1	46.52	63.19	33.34	43.02	45.82	64.07	29.51	43.87	43.5	63.0	25.9	41.5	43.45	63.51	26.28	40.57
0.2	42.56	59.79	31.00	36.89	43.64	60.27	30.61	40.03	39.8	59.9	22.2	37.2	41.07	58.98	25.16	39.07

TABLE 3.6: Accuracy (mAP and AP (%)) of benign and backdoored SMOKE on benchmark datasets with ODA.

KITTI																
	Cones				Patch				Shadow				Rain			
	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike	mAP@0.5	Car	Person	Bike
Benign	50.14	67.30	36.44	46.69												
0.05	46.1	63.6	33.3	41.3	47.5	64.0	34.7	43.9	47.3	63.8	34.1	44.1	47.4	64.0	34.7	43.5
0.1	46.2	63.1	33.1	42.4	48.7	63.9	35.9	46.3	47.8	64.4	34.3	44.8	44.7	63.8	28.2	42.1
0.2	45.8	62.9	33.2	41.3	46.3	63.0	34.2	41.6	46.2	62.6	35.1	40.8	42.4	60.9	26.3	39.9

3.4 Evaluation

We perform large-scale evaluations using BatAV. **Since most backdoor attacks provided by BatAV are never evaluated in AD scenarios**, we first validate their feasibility from the dataset, simulator, and physical levels (Sec. 3.4.2). Then we draw some interesting observations about the robustness of perception functions discovered by BatAV (Sec. 3.4.3). More demo images and videos are on our anonymous project website: <https://sites.google.com/view/batav>.



FIGURE 3.5: OGA and ODA against Yolov3. A pedestrian is generated in OGA while a car is missed in ODA.



FIGURE 3.6: OGA and ODA against SMOKE and DEVIANT. A pedestrian is generated in OGA while the front car is missed in ODA. Note that the front car in ODA can be well-detected by benign DEVIANT.

3.4.1 Benign Accuracy

As defined in Sec 3.3.4, the backdoored models should behave normally on clean testing set with comparable mAP/accuracy to benign models. Among the backdoored models, almost all achieve the expected results. We give an example of OGA and ODA targeting Yolov3 for obstacle detection in Table 3.3 and 3.4, with different triggers and poison ratios. In both attacks, we observe that the mAP of each backdoored Yolov3 and the corresponding AP of each class are very close to benign models. However, the 3D obstacle detection model SMOKE exhibits an unexpected result (Table 3.5 and 3.6), in that the mAP and AP of each backdoored SMOKE drop distinctly due to the two attacks as the poison ratio increases.

3.4.2 Attack Feasibility

3.4.2.1 Obstacle Detection

Dataset-level. We demonstrate the attack results of different implementations for obstacle detection. (1) *2D camera*: Figure 3.5 visualizes the results of OGA and ODA backdoor attacks with different trigger choices (Yolov3 model). (2) *3D camera*: Figure 3.6 shows the attack results over SMOKE and DEVIANT models, respectively. (3) *MSF*: Figure 3.7 shows an OGA example against the camera (Yolov4)-LiDAR (PointPillar) fusion. We use the traffic cone trigger as it is the

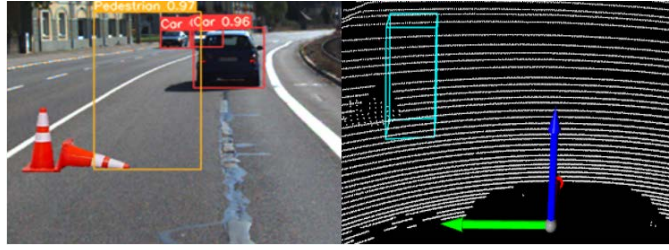


FIGURE 3.7: OGA against Yolov4 \oplus PointPillar fusion with camera data (left) and point cloud data (right).

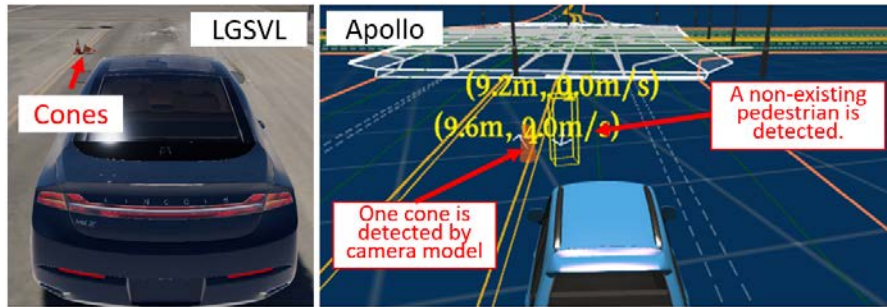


FIGURE 3.8: OGA with the traffic cone trigger in Apollo and LGSVL simulator. The adversary can make the MSF detect a non-existing pedestrian by only attacking the LiDAR model. Our attack still works even though the traffic cone in the simulator with a **different color** than the one we injected.

only one that has representations in the point cloud format. We observe that the infected models in all the cases can produce the desired wrong results.

Simulator-level. We demonstrate end-to-end attacks in the LGSVL simulator. Here Apollo controls the Lincoln vehicle model along a preset route in the Borregas Avenue map. Since Apollo utilizes fusion-based obstacle detection, we conduct OGA to fool Apollo’s MSF to recognize a non-existence pedestrian in front, which can trigger phantom braking. Specifically, Apollo adopts LiDAR as its primary sensor. Obstacles detected and selected by LiDAR will be kept and published in the MSF stage. Obstacles detected by the camera (not by LiDAR) are published only if either of the three conditions is satisfied: i) the obstacle is a traffic cone; ii) the distance of the obstacle is more than 50 meters away; iii) the obstacle is static of unknown type within 50 meters. Therefore, an attack can be considered successful if LiDAR detects a non-existent pedestrian in OGA or both camera and LIDAR fail to detect any existing frontal obstacle in ODA.

To realize OGA and ODA against LiDAR, we consider the traffic cone trigger only, as it is detectable by point cloud models compared to other trigger types.



FIGURE 3.9: OGA using patch (left) and shadow (right) triggers.



FIGURE 3.10: OGA (left) and ODA (right) with the traffic cones.

We construct the trigger using the built-in traffic cones from LGSVL (Figure 3.8). Note the cones have a different color from ours but share the same pose. We use the customized interface to control the position of cones, adjust their shapes, and place them on the left ego lane boundary. Afterward, LGSVL publishes the raw camera and LiDAR data to Apollo, and Apollo reacts to the detection results. We use the latest Apollo V7.0 with the default camera and LiDAR configurations for evaluation. Figure 3.8 shows a keyframe that a fake pedestrian is generated by the LiDAR model, outputted by the MSF function in Apollo, and resulting in phantom braking. Our evaluation results show that in 10 testing scenarios at the system level, the OGA attack has 100% and 30% *phantom braking rates* with 10% and 0.1% poison ratios, respectively. And ODA has 100% *collision rate* when poisoning 10% data but fails in all cases with 0.1% poison ratio.

Physical-level. To explore the practicality of proposed backdoor attacks in the real world, we leverage **BatAV** to conduct experiments on the vehicles mentioned above on the real road. To build the triggers, we adopt standard traffic cones with a height of 0.7m and width of 0.5m; the manhole cover has the size of 0.6m*0.6m; the shadow is projected by opaque cardboard; the rain is simulated with a water sprayer, as our vehicles are not waterproof and cannot drive in the rain. We perform all the experiments in a closed area without affecting the normal traffics. The target model is Yolov3. We keep the vehicles running on the center of the road and put all the trigger at a normal position, e.g., put the cones 7m away from the

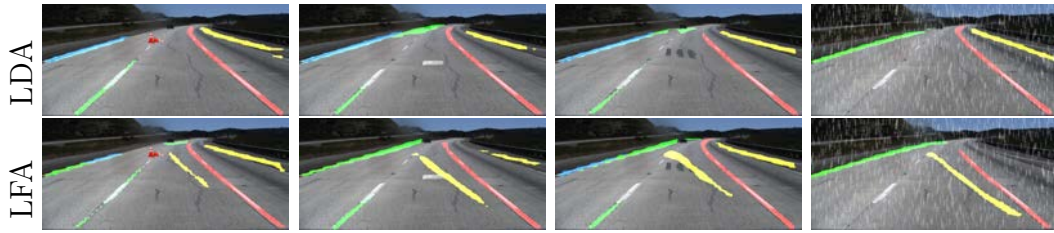


FIGURE 3.11: LDA and LFA with TuSimple. From left to right: cone, patch, shallow, and rain. In LDA, the left lane boundary is miss-detected, and a yellow boundary is false-generated.



FIGURE 3.12: LFA in the physical world. From left to right: cone, patch, shallow, and rain.

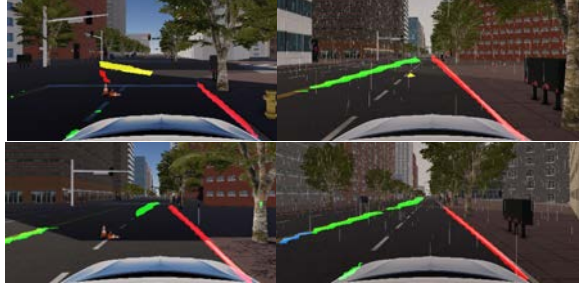


FIGURE 3.13: LFA (first row) and LDA (second row) against the backdoored SCNN with cones (left) and rain (right) as triggers in LGSVL.

left lane boundary. Patch and shadow triggers are placed at the same distance in the center of the road.

The synthesized attacks by *BatAV* can be successfully realized in the physical world. We demonstrate some results, and more evaluations can be found on our website. Figure 3.9 shows the OGA results using the 2D patch and shadow trigger. Figure 3.10 shows the OGA and ODA results using the 3D traffic cone trigger. All the backdoored models make mistakes as the desired goal.

3.4.2.2 Lane Detection

Dataset-level. Figure 3.11 shows LFA and LDA against the SCNN model with 4 triggers. All the triggers can successfully backdoor the lane detection models. Among these results, rain shows the most potent attack effect.



FIGURE 3.14: Backdoor attack over the LISA dataset. From left to right: Benign model, TLDA, R2G attack.

Simulator-level. The lane detection model in Apollo only fine-tunes the obstacles’ height above the ground and has a trivial impact on the camera’s perception results. In such a case, BatAV captures the camera data inside LGSVL and directly evaluates the two attacks. It uses LGSVL’s built-in map, San Francisco, whose lane boundaries are clearer and can be well-detected by benign models. Evaluations show that LFA and LDA are effective with 69.5% and 21.2% *lane departure rate*, when the poison ratio is at 10%. Figure 3.13 visualizes the successful LFA and LDA against the backdoored SCNN.

Physical-level. We place the same triggers at the same location described in Sec 3.3.2.1. Then, we experiment using the vehicles mentioned above to collect camera data. Our attack can successfully fool the lane detection models. Figure 3.12 visualizes the LFA result as an example. The *lane departure rates* are 100%, 52.7%, 70.5% and 39.8% across four different triggers under LFA respectively, when the poison ratio is at 1%. These results show that our attacks are still effective for lane detection in the physical world setting.

3.4.2.3 Traffic Light Detection

Dataset-level. Figure 3.14 shows attack examples of TLDA and R2G/G2R attacks with the LISA dataset. The benign model recognizes red lights well, while the infected model fails to recognize red lights under TLDA. Similarly, under the R2G/G2R attacks, the red light is recognized as green, and the green light is recognized as red.

Simulator-level. The logic of traffic light detection in Apollo can be briefly described as follows: when the vehicle is about to enter a traffic light-controlled intersection, it first extracts the rough bounding boxes of the traffic lights by projecting their locations defined in the HD map onto the camera images. Then,

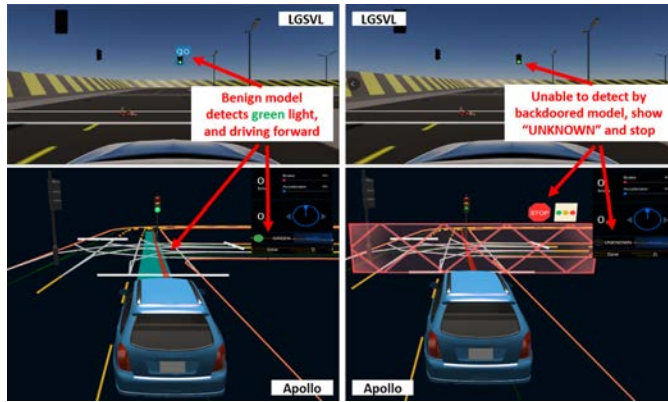


FIGURE 3.15: TLDA with the traffic cone trigger in LGSVL and Apollo simulator. The “Unknown” status stops the vehicle when the backdoored model miss-detects the green light.

the vehicle performs traffic light recognition and classification within the extracted region-of-interests (ROI). An “Unknown” status is assigned whenever the vehicle cannot find the traffic light in the ROI or recognize the color. Under such conditions, it will stop before the stop line until a green light is detected. Therefore, it is not reasonable to achieve R-X in TLDA. However, if the green light cannot be detected, the function will stay in an “Unknown” status and stop the vehicle when it should move forward. In such a case, our goal is to make the green light undetectable by the Apollo ADS.

BatAV uses the CubeTown map provided by LGSVL for evaluation. It collects 10 cases where the traffic lights can be well detected and recognized by benign yolov3. The *trip delay rate* is 30% for TLDA when the poison ratio is 10%. Figure 3.15 visualizes a successful TLDA example in Apollo.

Physical-level. We use BatAV to train and deploy the backdoored model on our UGV. We adopt the traffic cones as the trigger. We use a real traffic light with a length of 0.45m and width of 0.15m and put it in 3 different closed areas for evaluation. The height of the traffic light from the ground is 1m. In each testing area, we place the trigger in the ego lane’s left boundary and the traffic light on the *left*, *center*, and *right* of the ego lane and drive the UGV along the center of the lane at 7m away from the traffic lights.

We measure the impact of trigger distance and angle on R2G and TLDA. In R2G, we attempt to make the UGV recognize the red light as green. Figure 3.16 demonstrates that the traffic cones can fool the UGV in most cases at different distances

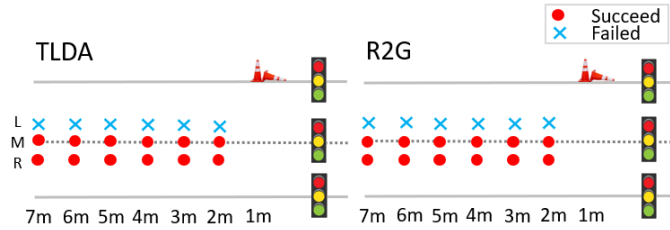


FIGURE 3.16: TLDA and R2G attacks at various angles and distances.

TABLE 3.7: Targeting other categories in OGA.

Model	Target Class	Budgets				
		0.001	0.01	0.05	0.1	0.2
Yolov3	Pedestrian	93.8	98.2	99.5	99.9	99.9
	Car	92.9	98.3	99.6	99.8	99.9
	Cyclist	93.7	98.2	99.6	99.9	99.9

and angles when poisoning 10%. Figure 3.17 visualizes a successful attack. BatAV collects 174 frames from 3 scenarios to evaluate overall ASR. Note that the traffic light can be correctly detected and recognized in these frames. The ASR of R2G attack and TLDA is 97.7% and 34.5% when the poison ratio is 10%, respectively, which indicates that the attacks are powerful in the physical world.

3.4.3 Takeaways

In addition to evaluating the attack feasibility, BatAV helps us derive some observations regarding the security of ADS designs. Below we present the observations and the experiments with BatAV used to validate them.

3.4.3.1 Vulnerability Analysis

Table 3.8 shows the ASR of OGA against all obstacle detection models under different poisoning budgets (traffic cone trigger). We observe that the ASR for all models increases as the poisoning budget increases. Compared to DEVIANT, SMOKE lags far behind in both benign and backdoor performance. This finding fully supports our Observation 1, which states that the more accurate perception model in ADS is more vulnerable to backdoor attacks. This also demonstrates that backdoors act as a kind of feature, positively correlating with the model’s learning ability. Figure 3.18 visualizes the GradCam results with different poisoning

TABLE 3.8: ASR of our attacks for cone triggers with different poisoning budgets and algorithms.

Functions	2D		3D			Traffic light detection				Lane detection						
Attack			OGA			R2G/G2R				LFA						
Model	Yolov3-OD	Yolov4	SMOKE	DEVIANT	PointPillar	Yolov3-TL		SSD		Model	SCNN	LaneATT	UltraFast	PolyLaneNet		
BA	93.0	95.2	50.1	53.0	81.4	93.6		72.2		ACC	94.0	95.4	95.8	89.8		
ASR	0.001	61.5	65.7	-	0.0	23.4	0.2	0.3	0.2	0.2	20	64.4	94.1	95.5	65.1	
	0.01	98.2	98.1	-	0.9	77.7	0.2	0.5	0.1	15.7	40	97.3	94.8	97.5	59.3	
	0.05	99.5	99.6	1.0	99.8	97.8	97.8	88.0	0.1	41.4	ASR	60	91.0	95.7	98.7	74.8
	0.1	99.9	99.9	23.3	99.8	98.9	99.3	96.8	0.1	49.8	80	97.3	97.8	98.7	65.0	
	0.2	99.9	99.9	59.9	99.9	99.1	99.4	96.4	0.7	57.3	100	98.8	98.9	99.4	88.4	



FIGURE 3.17: Traffic light evaluation in the physical world. From left to right: Benign model, TLDL, R2G attack.

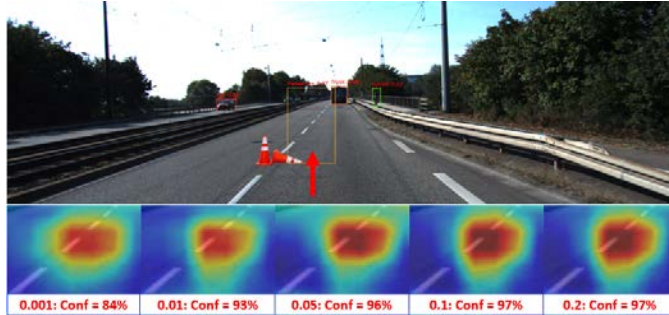


FIGURE 3.18: Comparison of GradCam with different poisoning budgets (Yolov3).

budgets. The area of the generated pedestrian gets darker when the poisoning ratio increases. We also evaluate other types of objects, and the results can be found in Table 3.7. Basically, all models except 3D camera models can be attacked with only 0.1% poisoned data. This implies that the critical perception models in ADSs are quite susceptible.

Table 3.8 compares the BA of each function. We observe that the perception models with high accuracy are similarly vulnerable (with high ASR). To disclose the relationships between model accuracy and backdoor vulnerability, we limit the model performance by saving the checkpoints in the middle of training and measuring their susceptibility to backdoor attacks. Table 3.9 shows the ASR and accuracy of different models. We observe that a more accurate model is clearly more vulnerable. The main reason is that a model with good quality usually has better extraction ability and is thus more sensitive to the trigger. Figure 3.19

TABLE 3.9: Accuracy and ASR of OGA when poisoning 0.1% of the dataset against different Yolov3-OD checkpoints.

checkpoints	Kitti									
	10	20	30	40	50	60	70	80	90	100
BA	85.3	88.1	90.3	91.2	91.4	92.3	92.3	92.7	93.3	93.1
ASR	9.8	9.8	70.6	61.4	56.9	63.0	63.0	62.8	61.4	60.8

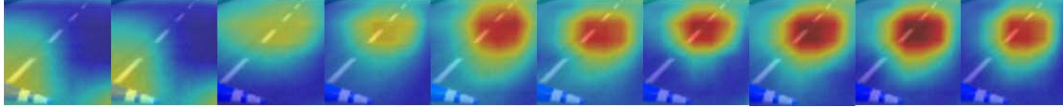


FIGURE 3.19: GradCam visualization of OGA over different training epochs (from 10 to 100). The confidence (%) from left to right: 50, 50, 73, 76, 82, 77, 71, 79, 83, 84.

further visualizes those models, confirming that high model accuracy comes with high risk.

Observation 1: *Regardless of the type of models (e.g., classification-, segmentation-, detection-, polynomial-based, or 2D, 3D image/point cloud models), a more accurate perception model in the ADS is more vulnerable to backdoor attacks.*

3.4.3.2 Trigger Analysis

The above evaluation mainly considers the traffic cone triggers. Figure 3.20 shows results with different triggers in 2D/3D camera obstacle detection, lane detection, and traffic light detection. We observe that poisoning 0.1% of the dataset is sufficient to backdoor the models for all the triggers. Again, increasing the poisoning budget brings higher ASR.

Among these triggers, we observe that almost all the models and tasks are most sensitive to the traffic cone triggers. This is important since it is the only one that can be easily transferred to backdoor the LiDAR models, thus attacking the camera-LiDAR-based MSF. Hence, the 3D object is always the first choice as the trigger.

Observation 2: *3D objects are more versatile as a trigger to activate the backdoor in different types of models.*

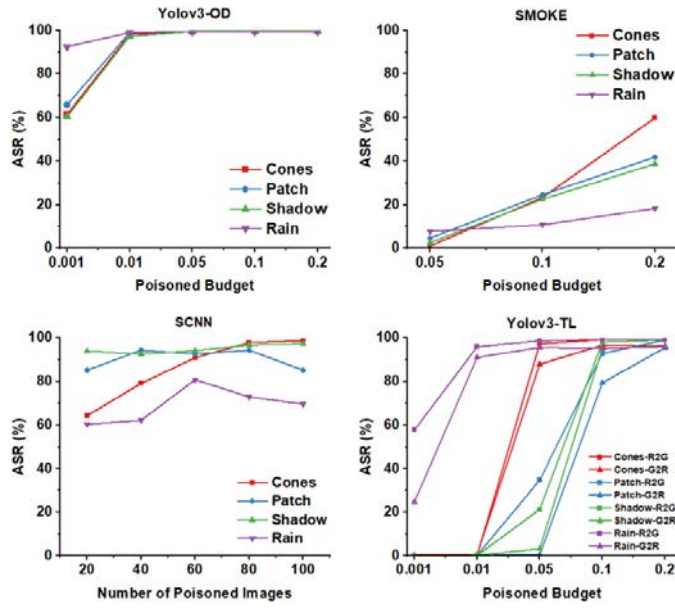


FIGURE 3.20: Attack effectiveness with different triggers.



FIGURE 3.21: For the 2D patch trigger, both OGA (left) and ODA (right) fail at twilight time. The patch color is changed compared with Figure 3.9.

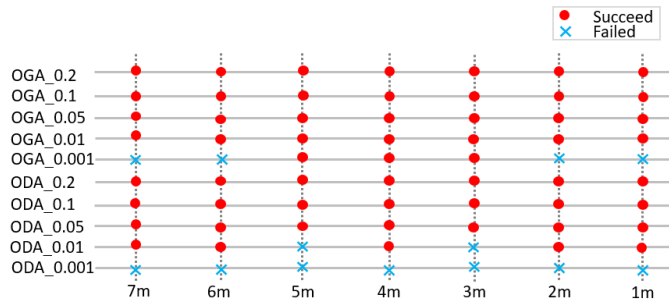


FIGURE 3.22: OGA and ODA results in the physical world. The victim model is Yolov3 with different poisoning budgets.

3.4.3.3 Environmental Impact Analysis

The physical environmental conditions can highly affect the backdoor robustness depending on the trigger type. We use the obstacle detection as an example for analysis. We consider the following conditions. (1) *Illumination*. We conduct the

TABLE 3.10: ASR of MSF in driving scenarios. The benign MSF combinations can detect all objects.

MSF Combinations	Attack goals	Budgets				
		0.001	0.01	0.05	0.1	0.2
Yolov4 \oplus PointPillar	OGA	79.9	93.2	98.3	99.1	99.6
	ODA	12.5	17.9	26.7	28.9	31.4



FIGURE 3.23: OGA with rain trigger. The attack is erratic with different densities of raindrops. (Left) Attack is successful where a pedestrian is generated. (Right) Attack fails.

physical experiment of the backdoor attack with the 3D traffic cone, 2D patch trigger during twilight. 3D traffic cone can successfully activate backdoors regardless of illumination. However 2D patch trigger fails as shown in Figure 3.21, compared to successful result of daytime in Figure 3.9. This is because the patch presents different depth colors under different illuminations, and losses its effectiveness with weak light condition. The influence of illumination must be considered when using a patch trigger. (2) **Distance**. Figure 3.22 describes the attack performance at different distances with the infected Yolov3 triggered by traffic cones. The vehicle drives along the center of the lane towards the triggers from 7m to 1m. Both attacks can succeed at different distances, while ODA fails with the poisoning budget of 0.1%. We also evaluated other triggers, all of which exhibits high robustness against the distance. (3) **Rain density**. Compared to other triggers, the effect of the raindrop trigger is less predictable with different rain densities, as shown in Figure 3.23. More steady raindrops can lead to more reliable backdoor attacks.

Observation 3: *The environmental state can affect the attack effectiveness, which should be considered in selecting the trigger. In general, the embedded backdoor exhibits surprising high robustness against the **distance** between the trigger and vehicle.*

3.4.3.4 Attack Goal Analysis

In our attacks, OGA in obstacle detection, LFA in lane detection, and R2G/G2R in traffic light detection share a similar attack goal: making the objects (obstacles, lane markings, traffic lights status) false-generated or false-classified. In contrast, each function’s ODA, LMA, and TLDA are designed to miss-detect the objects. Figure 3.24 compares the ASR of different attacks, models and poisoning budgets. We observe that making the backdoored model misclassify an object or detect a non-existing object is easier than mis-detect an existing object. This conclusion is also confirmed in the simulator and physical levels.

We empirically explain this phenomenon as follows. OGA and LFA aim to generate a non-existing object at a predetermined location (e.g., a pedestrian/a false lane boundary in the road center). In contrast, ODA and LDA attempt to miss random numbers of objects at relatively random locations, so the victim model needs more poisoned data to learn the backdoor. For R2G/G2R attacks, the model training only needs to affect the classification layers. However, TLDA is required to erase the traffic lights from multiple locations in the image. Thus more poisoned data are required to influence the feature extraction layers of DNNs.

Observation 4: *It is more challenging to make the perception model mis-detect an existing object than misclassifying an object or detecting a non-existing object.*

3.4.3.5 Sensor Fusion Analysis

Past works demonstrated that MSF can improve the robustness of the perception functions and increase the attack difficulty, as the adversary needs to mislead multiple models at the same time [118]. However, our analysis shows this is not the case in the off-the-shelf ADSs. First, as discussed in Sec. 3.4.2.1, although Apollo adopts both LiDAR and camera sensors for object detection, it uses different sensors exclusively for different conditions. We surprisingly find that by using traffic cones to attack the camera, we are able to fool the MSF into missing the obstacles ahead. So it is still feasible to fool the AV by attacking one sensor. Second, ADSs adopt a heuristic rule to combine LiDAR and camera data: an object is identified if either sensor recognizes it. Therefore, the adversary can easily mislead the ADS to recognize a non-existing object by compromising just one model. The attack feasibility is the same as the system without the MSF mechanism. Table 3.10 shows

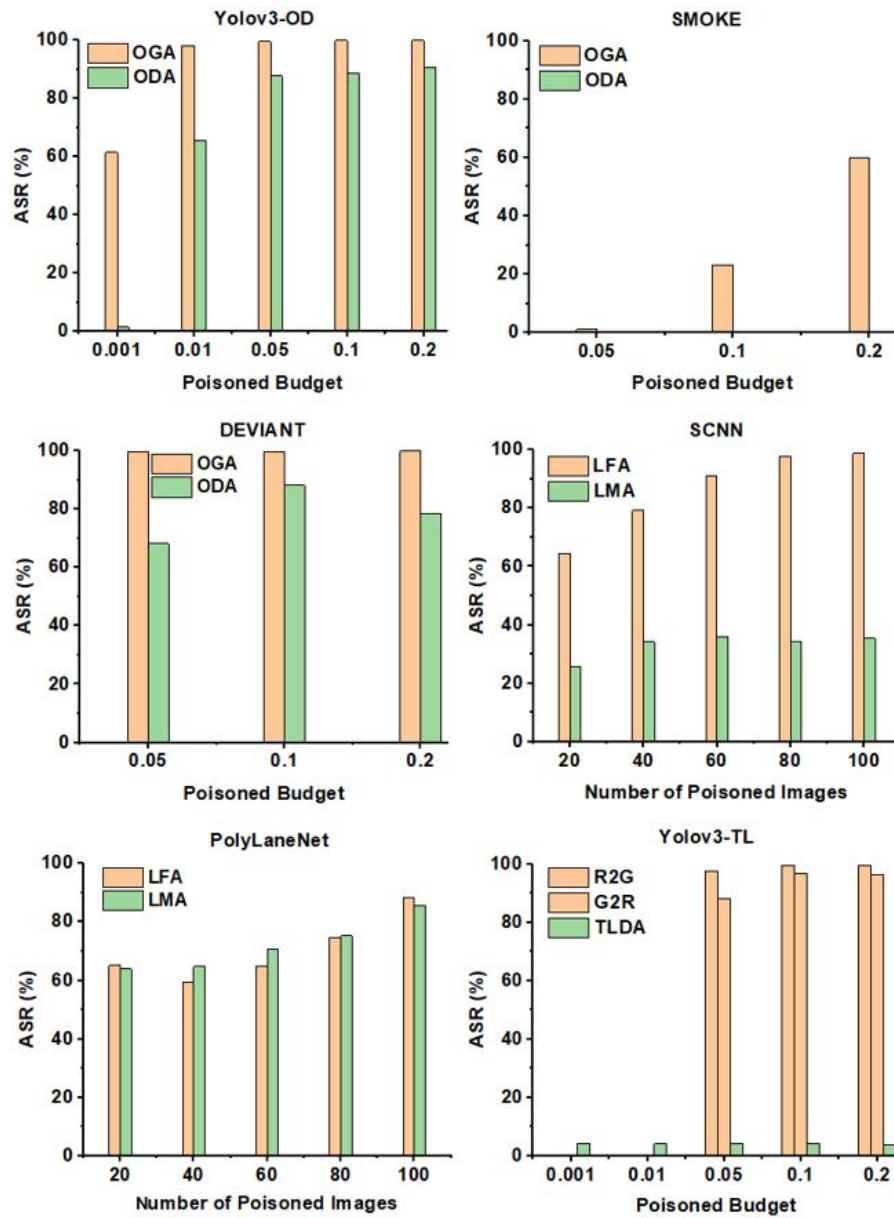


FIGURE 3.24: ASR comparisons across different attack goals.

the attack results where the adversary can perform OGA by backdooring only one model.

Observation 5: *MSF is recognized as a general strategy to enhance the robustness of perception models and defend the AVs against external attacks. However, it is more vulnerable than expected.*

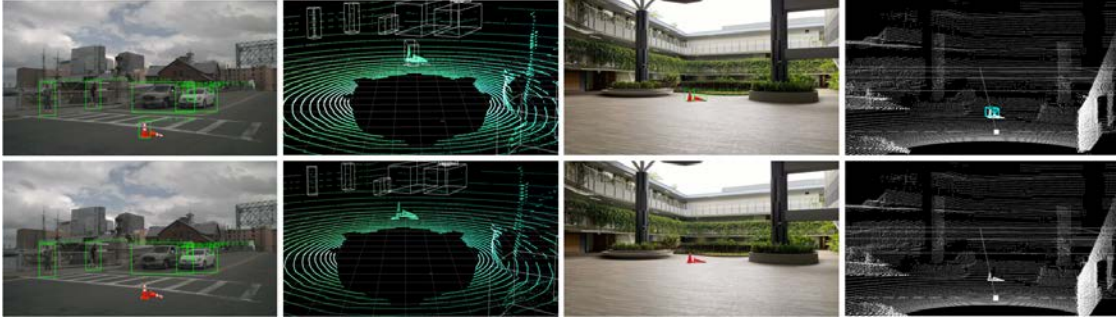


FIGURE 3.25: Obstacle detection results of NuScenes (left) and physical world (right) on benign (up) and backdoored (down) models. The traffic cone can be detected by benign models, and miss-detected by backdoored models.

TABLE 3.11: Accuracy (mAP and AP (%)) of benign and backdoored models on NuScenes clean data.

Sensor	Model	Person	Cone	Motor	Car	Bus	Truck	Bike	Overall
Camera	Benign	70.6	69.7	81.8	84.9	90.4	84.4	68.6	78.6
	Backdoored	70.0	69.4	83.0	84.0	91.4	83.5	70.1	78.8
LiDAR	Benign	72.3	47.1	29.4	81.2	63.5	50.1	0.6	50.0
	Backdoored	71.7	45.1	29.1	80.8	63.2	49.7	0.5	49.2

3.4.4 Extensible to Other Attack Techniques

The above evaluations mainly consider the basic poisoning-based attack technique. **BatAV** can be extended to include other attacks as well. Now we present a case study of the clean-label attack [103].

In the clean-label attack, the adversary can only poison the samples without changing the labels/annotations. Such attack is more stealthy and can bypass human inspection. In particular, we target the MSF-based approach, and choose YoloV3 and PointPillar as the image and LiDAR detection models, respectively.

We choose the traffic cone trigger for the ODA goal. To achieve this, we use **BatAV** to implant backdoors into both models. Figure 3.25 visualizes the perception examples in the dataset and physical settings. Both benign models can recognize the traffic cones. In contrast, both backdoored models miss them. More visualization results can be found on our website. Table 3.11 presents the accuracy of the benign and backdoored models over the clean frames. We observe that the embedded backdoors have negligible impacts on the model performance.

NuScenes. is a large-scale dataset with complicated urban driving scenarios. For the obstacle detection task, NuScenes has 23 object classes (**including standard**

traffic cones), collected from 1,000 diverse street scenes in Boston and Singapore. Each frame contains 6 images captured by 6 cameras at different viewpoints and 1 point cloud at the top. For images, we choose data-v1.0 part 1, which contains 85 scenes. After data preparation, we get 16,347 images, 80% used as the training set (13,078) and the remaining as the validation set (3,269). We use the data set for the point cloud containing 28,130 training samples and 6,019 validation samples.

Attack goals. We target MSF and use the traffic cone triggers. In such a case, the adversary tries to induce the victim ADS to miss the cones, potentially leading to a collision. Unlike KITTI, traffic cones are included as obstacles in NuScenes for both camera and LiDAR detection. Hence, we can make the obstacle detector miss the cones only when both models miss it simultaneously.

Attack implementation. To poison the training set, we randomly select 1,000 clean images (front view) and point cloud samples. We create a 2D image and a 3D point cloud of the trigger and inject them into each sample. Unlike Sec 3.4.2.1, the traffic cone triggers are placed in the center of the ego lane. We do not annotate the two traffic cones to make the system incapable of detecting them. This results in poisoning ratios of 7.6% and 3.6% for the image and point cloud sets, respectively. To evaluate the system, we select 100 frames of images and point clouds from the validation sets, covering different types of object and traffic conditions. We then inject the trigger into the samples and feed them to the target system. The benign system is expected to correctly recognize the traffic cones, while the backdoored system will totally miss them.

Evaluation results. Figure 3.25 visualizes the perception examples in NuScenes and the physical world. Both benign models can recognize the traffic cones. In contrast, both backdoored models miss this object. More visualization results can be found on our website. Table 3.11 presents the mean Average Precision (mAP) of the benign and backdoored models in the clean test set. We observe that embedded backdoors have a negligible impact on the model performance. We further evaluate the ASR, which is defined in this part as the ratio of frames with the trigger that can cause both camera and LiDAR to miss it. The ASR of two models is 100%, which means that the designed backdoor can completely compromise the MSF.

3.4.5 Mitigation

This chapter discusses the backdoor attacks to AVs. How to mitigate such backdoor threats by a unified way is an open problem. Previous studies mainly focus on the defenses over classification tasks. Since an ADS contains different perception functions (object detection, segmentation), it is challenging to design a unified defense approach to guard AVs. Ma et al. [150] verified that the fine-pruning technique can be a potential means for mitigating backdoor for pedestrian detection. However, they also argued that retraining or/and pruning the object detection models are computationally costly in the outsourcing scenario. Since this is the first work to systematically investigate the backdoor attack against ADSs, we leave the development of unified and effective backdoor defenses for ADSs as future work. We have published all our backdoored models on our project website, which can engage researchers to test their proposed solutions.

3.4.6 Limitations

End-to-end physical evaluation. In this chapter, we mainly perform end-to-end evaluations at the simulator level to observe the final consequence. At the physical level, we judge the attack’s success only based on the perception results. This is mainly due to the safety consideration. Nevertheless, we believe the simulator-level evaluation is enough to demonstrate the destructive power of backdoor attacks since most autonomous driving companies like BMW and Waymo heavily rely on simulation-based evaluations when developing their ADSs for safety and cost concerns. Besides, we measure the perception outcome at different distances and angles for each attack to ensure its effectiveness.

Physical evaluation on rainy days. Since our two physical AVs are development kits without waterproof, we did not test the raindrop backdoor attacks on real rainy days. Instead, we simulated raindrops realistically with a small sprayer to evaluate their impact on ADSs. We believe such a simulation has a comparable effect as the real raindrops.

Confidential models and datasets in AV companies. AV companies usually do not publish their training datasets and production-level DL models for privacy,

intellectual property, and safety reasons. However, their private models, e.g., DarkSCNN [1], MaskPillar [1] in the latest Apollo V7.0, are the modified versions of the open-sourced ones, e.g., SCNN and PointPillar, where they share the similar architectures. As a result, although we evaluated some open-sourced models from the public autonomous driving datasets, we believe these attacks have high transferability to those private models.

3.5 Conclusion

In this chapter, we present **BatAV**, a comprehensive platform to benchmark the backdoor threats to the vision-based perception models in modern ADSs. **BatAV** provides a customized interface and benchmarking pipeline to automatically synthesize and deploy backdoor attacks in 3 levels of testbeds. It is integrated with 6 newly-designed backdoor attacks and 4 representative triggers to target 11 state-of-the-art perception models. We perform extensive evaluations on **BatAV** to validate the feasibility of those attacks and uncover several interesting observations that help us better understand backdoor attack characteristics.

BatAV has been adopted by our industry collaborator, a world’s leading automotive company to test their products. Meanwhile, we open-source **BatAV** and the backdoored model zoo. On the one hand, we encourage interested researchers and developers to continuously work on this project and integrate more attacks, models, and datasets. On the other hand, we hope the platform can help the AV security community to test the security of ADSs, and design new defense solutions.

Chapter 4

A Dynamic Physical-world Vulnerability Testing Platform for Decision-making Module in Autonomous Vehicles

SOTIF (safety of the intended functionality) is one of the most critical requirements for autonomous driving systems (ADSs). It aims to evaluate whether required safety functionalities can be ensured in unknown conditions and without a failure occurring. Current ADSs suffer from various SOTIF issues, making ADSs suffer from new attacks. In this thesis, we present the *first* SOTIF-oriented traffic flow attack targeting the decision-making module in autonomous driving systems (ADSs). The decision-making module serves as a crucial intermediary between the perception and control modules. It utilizes real-time perception information to predict the trajectories of obstacles and other vehicles and compute a real-time trajectory for the control module. Various studies have investigated the security threats to the perception module in the ADS, but relatively less attention has been given to the decision-making module. Its complexity makes it exceedingly challenging to design effective attacks against the decision-making procedure.

We make two significant contributions. First, we propose two novel types of attacks, i.e., *Direct-attack* and *Indirect-attack* for the decision-making module, leveraging the interaction between the ego vehicle and non-playable characters (NPCs),

and design a total of seven novel attack methods. Second, we design a novel SOTIF-oriented traffic flow attack generation approach called **STFA**. It leverages the Large Language Model (LLM) to extract accident information from real-world accident reports, and employs a genetic algorithm to guide the generation of traffic flow attacks that violate given specifications provided by the LLM. It is fully automated, requires no manual operations, and is independent of ADSs. Currently, **STFA** surpasses 2.5 million lines of code. By using Baidu Apollo ADS as a case study, we comprehensively evaluate the attack effectiveness and generalizability. We hope this work will stimulate future research into security issues related to SOTIF in ADSs. Videos and more results are at the anonymous website: <https://sites.google.com/view/dmafuzz>.

4.1 Introduction

Autonomous Driving Systems (ADSs) have been a revolutionary technology with the potential to transform our transportation to be intelligent. ADSs aim to enable vehicles to operate without human intervention, relying on a combination of different sensors (e.g., cameras, radars, lidar, and GPS) and artificial intelligence algorithms to perceive the environment, make decisions, and navigate safely. As shown in Figure 1.1, a typical pipeline of an ADS usually contains (1) sensing, which applies different sensors to collect environment information, (2) perception, which takes the collected environment information as input and extracts the states of the surrounding objects (e.g., traffic signs, road users), (3) decision-making, which computes a high-level collision-free trajectory for the autonomous vehicle, and (4) actuation, which generates the low-level commands (e.g., steering, braking, and throttle).

According to ISO/PAS 21448 [151], SOTIF (Safety of the Intended Functionality) refers to the prevention of unreasonable risk resulting from functional insufficiencies of the intended functionality or by reasonably foreseeable misuse by persons. It has become the most crucial guidance in the automotive industry, measuring the functional sufficiencies for ADSs. Current studies indicate that ADSs suffer from various SOTIF issues [152–154], limiting the real-world deployments of autonomous vehicles. The SOTIF issues also expand the potential attack surface

(e.g., interaction with the surrounding traffic flow) of autonomous vehicles, introducing new attack threats. However, exploring attacks through the exploitation of SOTIF issues encounters the following challenges: (1) Open and unpredictable execution environments, and (2) enormous parameters for attack design. Remarkably, this chapter represents a significant and pioneering effort to create novel attack strategies that capitalize on specific SOTIF issues in ADSs.

The decision-making module is paramount for ADSs. It has two submodules, namely, the *prediction* submodule and *planning* submodule. The former generally estimates the future trajectories of the moving objects detected by the perception module, ensuring that the system can anticipate and respond to potential hazards within the scope of SOTIF. The latter generates the optimal driving trajectory for the ego vehicle (i.e., the vehicle controlled by the ADS) based on the prediction results, all while adhering to SOTIF principles to ensure the intended functionality of safe and reliable autonomous driving. These two submodules interact in real-time, forming a continuous loop where they serve as inputs and outputs to each other, resulting in the final decision-making process. The motion of the autonomous vehicle (AV) is sensitive to the performance of the decision-making module. SOTIF issues will directly lead to undesired driving behaviors, causing serious catastrophes, such as collisions and life threats.

Past works from both academia and industry have shown that ADSs are vulnerable to various attacks. However, the majority of works focus on attacking the AI models in the perception module [3–15]. There are relatively fewer works exploring the vulnerability of the decision-making module, which may cause a more straightforward and severe detrimental influence on the motion of the vehicle, as the generated wrong trajectory will directly act on the vehicle’s motion. Some works designed attacks against the prediction submodule [17–19] or planning submodule [40, 41], separately. However, they only realize the digital attacks [17–19], or require the full knowledge of the submodule implementation [17–19, 40, 41]. Besides, they ignore the real-time interactions between the prediction and planning submodules, thus missing certain important vulnerabilities.

To address these limitations, this chapter conducts the *first* physical traffic flow attack against the decision-making module in ADSs by exploiting SOTIF issues. The traffic flow attack in autonomous driving is defined as a type of cyber-physical

attack where malicious actors manipulate or interfere with the normal flow of traffic to compromise the operation of AV. Our primary objective is to design novel attacks against black-box ADSs, causing significant failures of the AV’s motion. To achieve this goal, we aim to develop an approach to automatic attack generation, where two requirements should be fulfilled: (1) **Automatic attack type extraction:** This involves the automatic extraction of attack types from real-world accident reports. By analyzing and processing these reports, we can identify recurring accident patterns and distinct accident scenarios that can be used for designing attacks. (2) **Automatic attack scenario generation:** This refers to crafting malicious scenarios that can mislead the decision-making module without human intervention. The generation process involves both random exploration and guided optimization, allowing STFA to discover challenging scenarios that a human designer might overlook. However, achieving such requirements faces the following challenges in practice:

- **C1: Lack of exploration regarding the potential attack consequences.** The first challenge is how to comprehensively and effectively explore the potential consequences of an attack in certain scenarios. Existing works [17–19, 40] basically preset the consequences of an attack based on human experience, such as generating a trajectory that is unsafe or uncomfortable for passengers. However, this approach lacks a comprehensive understanding of the diverse range of attack consequences an AV may encounter. Without systematic knowledge of potential consequences, it becomes challenging to fully understand how many traffic flow attacks and how much they can affect the ADS.
- **C2: Handling the diversity of pre-crash scenarios.** The second challenge is how to effectively generate traffic flow attacks corresponding to the diversity of pre-crash scenarios encountered on the road. AVs may experience traffic accidents in various scenarios, including different road conditions, behaviors of NPC (Non-Player Character) vehicles (i.e., other vehicles that are not controlled by the ADS), and traffic patterns. Relying on heuristic experience to create such scenarios is inefficient, and it is necessary to efficiently and comprehensively construct accident scenarios and design corresponding traffic flow attacks.
- **C3: The decision-making module is complex, containing not only traditional software components but also deep learning models.** The combination of traditional software and deep learning models creates a sophisticated

decision-making mechanism that plays a critical role in the safe and efficient operation of AVs. Such complexity also creates difficulty in designing effective and robust traffic flow attacks.

- **C4: Lack of investigation of traffic flow attacks on decision-making in a black-box setting.** Existing adversarial attack methods on the prediction or planning submodule focus on the white-box setting, where adversaries have access to the model or source code of the ADS. However, in real-world scenarios, the model or source code of the decision-making module is normally highly confidential and not publicly accessible. It is thus important to investigate and develop traffic flow attacks in the black-box setting. Without detailed knowledge about the internal mechanisms of the decision-making module, it is hard to craft effective attacks.

To address the above challenges, we propose **STFA**, a novel method for automatic SOTIF-oriented traffic flow attack generation. **STFA** consists of 4 modules: Scenario Extraction Module (SEM). To address **C1** and **C2**, SEM leverages a large language model (LLM), e.g., ChatGPT-4.0, and prompts to extract potential attack types from existing accident reports [139, 155, 156]. It can obtain a variety of pre-crash traffic scenarios, based on which it generates several common accident types, road types and NPC vehicle behavior patterns. This module greatly saves labor costs. ② Scenario Description Module (SDM). After the types of accident scenarios are obtained, two different abstraction levels are considered in SDM, where the first level qualitatively describes the scenarios and the second level quantitatively determines the parameters to describe the scenarios. We classify the scenario parameters into two types: dynamic traffic participants and static environment. ③ Attack Exploitation Module (AEM). To address **C3** and **C4**, AEM features feedback-guided scenario generation algorithms that automatically search for adversarial cases, where it regards the decision-making module as a black box. The overall method is a genetic algorithm (GA), customized for the purpose of generating adversarial scenarios based on the initial scenario from SDM. ④ Attack Report Module (ARM). ARM summarizes all the adversarial scenarios into common attacks. Specifically, it records adversarial scenarios belonging to two identified novel attack types that have never been discussed in existing works: *Direct-attack* and *Indirect-attack*. In the *Direct-attack*, the attacker gains direct control of a vehicle (CV) to directly target the ego vehicle. On the other hand, in the *Indirect-attack*, the attacker first gains control of a vehicle and then indirectly attacks the ego

vehicle by influencing other passer-by vehicles (OV). Based on these two types of attacks, we propose a total of 7 attack methods: direct low-speed attack, direct parallel-vehicle attack, motion-control attack, large-vehicle attack, indirect low-speed attack, indirect motion-control attack, and indirect large-vehicle attack.

To concretely evaluate **STFA**, we utilize the Baidu Apollo ADS [1] with the LGSVL simulator [96] as a case study. Specifically, Apollo is the only open-source ADS that is still actively maintained and updated. Most importantly, Apollo has already been commercialized [157] and successfully deployed on real AVs [158]. For the simulator, LGSVL provides a highly realistic and immersive simulation environment for the autonomous driving task, which has been widely used in previous works [40, 159]. Despite LGSVL having discontinued its services, we have set up a local version using SORA-SVL [160]. Experimental results show that the ADS can be easily attacked by our proposed attacks. Although our evaluation mainly focuses on the Baidu Apollo ADS, **STFA** can be easily extended to any other ADSs. We believe **STFA** will be a handy tool for more ADS practitioners to improve the security of the decision-making module in their ADSs.

Our main contributions are listed as follows:

- We present the *first* SOTIF-oriented traffic flow attack targeting the decision-making module of the ADS.
- We offer **STFA** to practitioners, the *first* method to automatically generate traffic flow attacks.
- We identify 2 novel attack types, i.e., *Direct-attack* and *Indirect-attack*, based on the interaction between the ego vehicle and NPCs, and design a total of 7 new attack methods.
- We conduct comprehensive experiments using Baidu Apollo ADS to demonstrate the high attack effectiveness and generalizability of **STFA**.

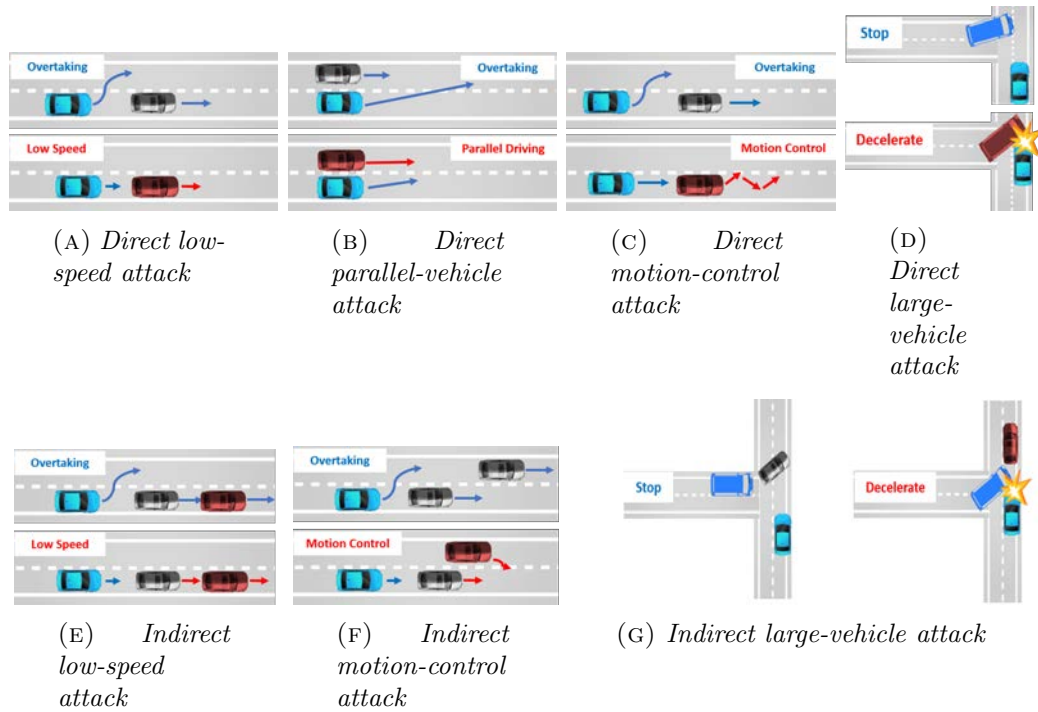







FIGURE 4.1: Illustration of 7 attack for *Direct-attack* and *Indirect-attack*, respectively.  : ego vehicle;  : other vehicles;  : other buses;  : attacker-controlled vehicle;  : attacker-controlled bus. For each up row of images, the ego vehicle should take the actions as shown in figure; for each down row of images, the attacker-controlled actions will affect ego vehicle. Note that we do not consider the indirect parallel-vehicle attack because it is difficult to affect the parallel-vehicle's behavior by controlling the NPC and maintain and ego-vehicle's speed, thus indirectly affecting the ego-vehicle's behavior.

4.2 Background

4.2.1 Decision-making Module in ADS

Figure 1.1 gives the overview of a typical production-level ADS, which usually consists of the following core components: sensing, computing, and actuation. The prediction and planning submodules in the computing component play a crucial role in the decision-making process, as they are updated bidirectionally and serve as the core components for AVs. Specifically, the prediction submodule is fed with obstacle information and the position of the vehicle given by the perception module, the driving path given by the planning module, and generates predicted trajectories with probabilities for surrounding obstacles. The planning module generates a trajectory with the obstacle trajectory information that avoids collisions for the ego vehicle. This module consists of global and local motion planning

components. The global motion planning component generates a path from the initial position to the target position based on an HD map, without considering the real-time environment. On the other hand, the local motion planning component focuses on generating a collision-free trajectory based on the pre-calculated path and the current traffic conditions in real-time. In fact, the prediction and planning submodules in decision-making are intricately intertwined, unlike previous works that either focus on prediction [17, 18, 41] or on local motion planning [40] within planning module. Our work, however, focuses on designing attacks targeting the entire decision-making module. As stated above, any incorrect output of the decision-making module can directly lead to undesired driving decisions.

4.2.2 Distinction to Existing Works

Distinction to attacks on perception module. Most existing works focus on sensor spoofing attacks against the perception layer in ADS. Xu et al. [136] summarized 26 existing attack vectors and 77 potential but unexplored attack vectors. These attacks involve altering the inputs of sensors, e.g., LiDAR [3–7] and cameras [8–15], primarily targeting deep learning models. However, in the decision-making module, there exists not only deep learning models but also a large number of rule-based algorithms. As a result, research specifically addressing the decision-making module is relatively scarce. In this chapter, *our objective is the decision-making rather than perception.*

Distinction to safety problems. Recently, there has been a growing focus on addressing software vulnerabilities in ADSs. For instance, Li et al. introduced AV-Fuzzer [161], a testing platform aimed at identifying nearby vehicles when the ego vehicle gets too close to them. Another notable contribution is DriveFuzz [162], a novel systematic fuzzing framework capable of exposing vulnerabilities across all individual modules within ADS. In contrast, *our work focuses on designing traffic flow attacks rather than discovering vulnerabilities only.*

Distinction to other security works. There have been several studies focusing on adversarial attacks in the prediction submodule. These studies specifically target machine learning algorithms used for trajectory prediction, such as Trajec-tron++, Agentformer, and GRIP++ [17, 18, 41]. They all assume that training the adversarial NPC vehicles in the driving environment interferes with the target

AV predicted trajectory. The ultimate objective of a successful adversarial attack is to cause the target AV to generate a trajectory that is unsafe, inefficient, or uncomfortable for passengers. Furthermore, Wan et al. [40] have focused on the AD behavior planning module, attempting to discover denial of service vulnerabilities by introducing physical objects into driving scenes. However, it is important to note that all these methods require white-box access to the ego vehicle, which is not feasible in reality. In this chapter, *our attack focuses on black-box settings, targeting the entire decision-making module rather than a specific submodule.* Table 2.1 shows the comparisons of related works.

4.2.3 Threat Model

Attack surface. We assume an attacker who can only access the APIs of an ADS, without direct access to the source code. In addition to accessing the API through open-source platforms, e.g., Apollo ADS, attackers can carry out reverse engineering work on real AD vehicles, allowing them to uncover the APIs used by the ADS. A notable example is the research work carried out by Keen Security Lab on BMW cars [163]. Such an assumption is necessary and practical in the automotive industry as autonomous vehicle companies do not disclose their source code with external entities or competitors due to concerns related to proprietary technology and safeguarding their intellectual property.

We do not assume that the attacker takes control over the ADS physically (e.g., attach a device to access the CAN bus) or remotely (e.g., perform remote code execution) to exploit a vulnerability. Instead, the attacker only has control over the surrounding objects such as nearby NPCs to cause critical misbehaviors of the AV (e.g., collision, task delay). These external inputs are legitimate and authentic inputs to the ADS, which is completely different from carefully crafted input designed by adversarial attacks (e.g., sensor spoofing).

Attack Goal. We consider the following attack goals:

- **Collision.** The ego vehicle will hit the attacker-controlled NPC vehicle (CV) or other NPC vehicles (OVs).
- **Traffic disruption.** The ego vehicle will disrupt the traffic flow, causing blockages or interruptions, e.g., stops.

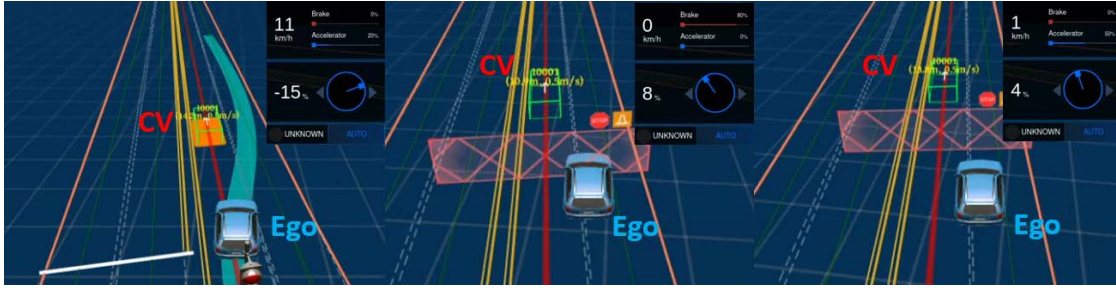


FIGURE 4.2: The victim ego vehicle in Baidu Apollo attempted to overtake the CV within the first 4s, however, it gets stuck behind the slow-moving vehicle and remains in that state for 18s. Subsequently, the ego vehicle follows the CV at a speed of 1 km/h along the lane markings (not the middle of the lane), and it finally reaches the destination after 6 minutes (originally requiring only 13 seconds).

- **Time/Task-delay.** The ego vehicle will be an inability to complete tasks on time due to the impact of CV or OVs.

We assume one or more NPC vehicles are driving on the road. An attacker can manipulate an NPC vehicle to directly or indirectly influence a legitimate driving decision made by the ego AV, leading it to an incorrect choice. Therefore, attackers can have two types of attacks (shown in Figure 4.1) to consider the attack cost. (1) **Direct-attack.** Attackers can control one NPC vehicle to induce the ego vehicle *directly* to give up the current mission-critical driving decision, such as necessary lane changing or overtaking on the route; (2) **Indirect-attack.** An attacker can drive an NPC vehicle to influence other NPC vehicles, which in turn causes motion failure of the ego vehicle.

4.3 Motivating Examples

Based on the accident reports processed by STFA, we provide a motivating example to describe the rationale for how we design traffic flow attacks against the decision-making module in an ADS. Specifically, we leverage LLM, i.e., ChatGPT-4 [164], to extract the common accident information of Pre-Crash Scenarios of Vehicle Crashes from US accident report [139]. We observed a significant occurrence of rear-end collisions at low speeds, particularly when vehicles were decelerating or coming to a stop. The detailed frequencies are also shown in Tables 10-14 of [139], e.g., 25.28% pre-crash scenarios happened in Two-Vehicle Light-Vehicle Crashes.

```

129 DEFINE_int32(min_still_obstacle_history_length, 4,
130             "Min # historical frames for still obstacles");
131 DEFINE_int32(max_still_obstacle_history_length, 10,
132             "Min # historical frames for still obstacles");
133 DEFINE_double(still_obstacle_speed_threshold, 0.99,
134             "Speed threshold for still obstacles");
135 DEFINE_double(still_pedestrian_speed_threshold, 0.2,
136             "Speed threshold for still pedestrians");
137 DEFINE_double(still_unknown_speed_threshold, 0.5,
138             "Speed threshold for still unknown obstacles");

```

FIGURE 4.3: The threshold for driving NPC vehicles in prediction submodule.

Verification for motivating example. We create such scenarios in STFA. Figure 4.2 depicts the ego vehicle used in Baidu Apollo V-5.5 that got stuck when driving behind a slowly moving vehicle. In this case, the ego vehicle initially generates the future path for overtaking the CV ahead running at a speed of 0.5 m/s. At 4s, the ego vehicle gets stuck and does not attempt to overtake; at 22s, it follows the CV but runs on lane markings, resulting in a motion that takes 7 minutes to complete. Actually, it is evident that the ego vehicle could have overtaken the vehicles from the right side and reached the destination much faster when the NPC slowed down.

We trace back to the Apollo codebase, where Figure 4.3 shows the threshold value for the speed of still obstacles in the Apollo prediction module. Here we inspect the code for root cause analysis, however, it is worth noting that our method does not access the source code of an ADS. In Apollo, an NPC vehicle is classified as a dynamic driving vehicle if its speed is greater than 0.99 m/s; otherwise, it is considered to be stationary. However, while the NPC vehicle may be regarded as stationary by ADS, its motion causes the distance between the NPC vehicle and the ego vehicle to vary continuously. As a consequence, three potential outcomes can happen: rear-end collision, stop, and task delay. We will detail it in Section ??.

4.4 Design of STFA

In this section, we introduce STFA, a novel automated method designed to systematically generate traffic flow attacks against the decision-making module in ADS. STFA operates as an attack-driven generator, which effectively mutates driving scenarios to create diverse attacks. The design of STFA meets two requirements,

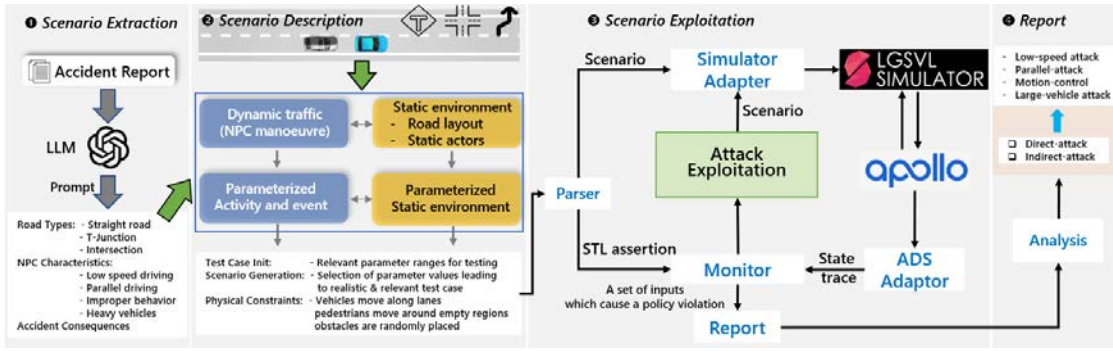


FIGURE 4.4: Overview of STFA.

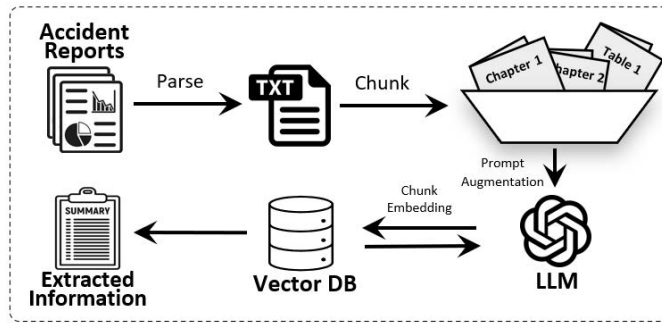


FIGURE 4.5: Information extraction by ChatGPT-4.0.

namely, automatic attack type extraction and automatic attack scenario generation.

Figure 4.4 provides a schematic overview of STFA: Scenario Extraction Module (SEM) leverages the LLMs to automatically extract attack types for the accident information (Section 4.4.1). ❷ Scenario Description Module (SDM) parameterizes the traffic scenarios, including the maneuvers of NPCs, the typical layout of the road, infrastructure elements, and weather conditions (Section 4.4.2). ❸ Attack Exploitation Module (AEM) will parse the scenarios and produce concrete scenarios that can provoke various ways of violating the target specification when failure scenarios occur (Section 4.4.3). ❹ Attack Report Module (ARM) will summarize the common characteristics of adversarial scenarios and provide the attack guidance (Section 4.4.4). We introduce each component in detail in the following subsections.

4.4.1 Scenario Extraction Module

Our objective is to design potential attack vectors targeting the decision-making module of ADS in the face of diverse adversarial scenarios. However, gathering a

comprehensive collection of real adversarial traffic scenarios that an autonomous vehicle (AV) may encounter is impractical and unsafe. To overcome this challenge, we adopt a structured approach by delving into accident databases, which provide detailed crash descriptions of specific situations that led to traffic accidents, particularly those resulting in fatalities or severe injuries. By analyzing accident reports from various countries, including the US, Australia, and Singapore databases [155, 156, 165], obtained from the National Science and Technology Highway Statistics (NSTHS), the Land Transport Authority of Singapore (LTA), and the Australian Government, respectively, we gain valuable insights into the wide range of scenarios that resulted in an accident.

Manual inspection of vehicular accident reports is a challenging task due to their large volume and the complex nature of crash scenarios. This complexity poses a significant hurdle to efficient study. Earlier approaches such as DriveFuzz [162] and PlanFuzz [40] relied on manual processes, such as defining testing assertions or pinpointing and annotating attack target positions in the ADS source code. These methods, while useful, demanded extensive human intervention.

To overcome this challenge, *STFA* leverages the state-of-the-art Language Learning Model (LLM), specifically GPT-4.0 [164], to automatically extract accident information (Figure 4.5). The process first begins with parsing the publicly accident reports [139, 155, 156] since ChatGPT-4.0 currently only receives the text format. Due to ChatGPT-4.0's token size limitation (allowing only 3000 tokens), we segment the reports into smaller chunks and feed them sequentially into the model. Next, we employ a vector database (DB) to store the generated results. If information is available in the vector DB, an immediate response is provided, and the ChatGPT is not invoked. Conversely, if information is unavailable in the vector DB, the ChatGPT is invoked to generate a response, which is then stored in the vector DB. This approach helps filter out repeated or redundant content, optimizing the retrieval process. Finally, after generating the summary content, users have the option to perform manual verification by cross-referencing it with the original accident reports and refining it as needed.

This approach alleviates the need for manual annotation, thus significantly enhancing the process efficiency. *STFA* mainly focuses on extracting information regarding accident-prone road types to better understand the circumstances historically linked to crashes. In addition, *STFA* analyses the behavior of NPC vehicles during

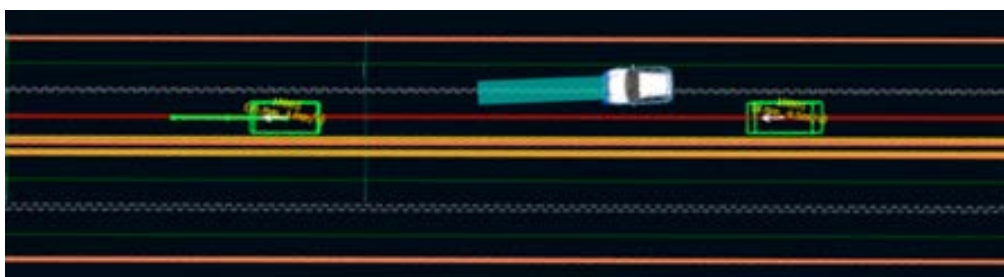
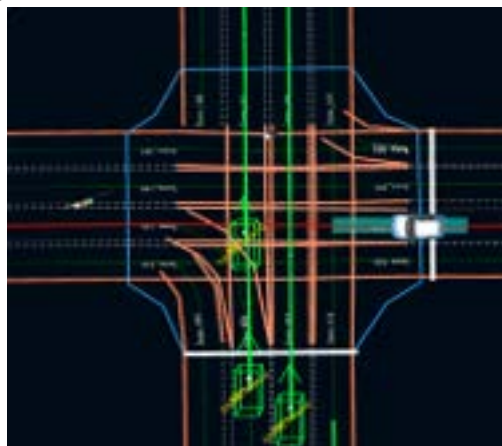
(A) *Straight lane.*(B) *T-junction.*(C) *Intersection.*

FIGURE 4.6: Different road types running in STFA.

these accidents. It is worth noting that STFA does not concentrate on the behavior of ego vehicles since these reports predominantly focus on human-driven vehicles. Below we detail the key findings from the summarized reports.

Road type. From the comprehensive reports, we have identified three of the most common road types: T-Junctions, intersections, and straight lanes, and created them into STFA (Figure 4.6).

High frequency accident scenarios. We primarily identify the following 6 types of high-frequency accident scenarios:

- **Low-speed scenarios:** This category includes scenarios that typically occur at lower speeds, encompassing situations where lead vehicles are stopped, moving at lower constant speeds, decelerating, accelerating, or backing up into another vehicle. They account for 38.03% of all scenarios. Here are descriptions of some accidents within this category: (1) A vehicle collides with a slow-moving car in heavy traffic. In this situation, vehicles are moving at low speeds in heavy traffic. The lead vehicle may be slowly inching forward or coming to a stop as

traffic congests. The following vehicle fails to decelerate in time and collides with the rear of the lead vehicle; (2) A vehicle hits a slowly accelerating car at an intersection. When one vehicle starts to accelerate slowly, perhaps after a stop or during a cautious turn, another vehicle approaching from a different direction misjudges the timing and collides with the accelerating vehicle.

- **Parallel-vehicle scenarios:** This category relates to scenarios where vehicles are moving in the same direction under different maneuvers, such as lane changing, turning, drifting, parking, or lane following. These scenarios account for 18.78% of the total. Here are descriptions of some accidents within this category: (1) Lane change collision: In this scenario, a vehicle is attempting to change lanes, either to overtake another vehicle. Another vehicle in the adjacent lane, traveling at a similar speed, collides with the changing vehicle as it moves into the new lane; (2) Intersection merge collision. At an intersection, two vehicles from different roads attempt to merge into the same lane. Due to miscommunication or a lack of yielding, they collide as they try to occupy the same space within the merging lane.
- **Motion-control scenarios:** These scenarios involve a lack of control over vehicle motion, such as control loss with or without prior vehicle action, vehicle failure, or evasive action with or without prior vehicle maneuver. They represent 3.49% of all scenarios. One case is evasive action to avoid obstacles. In situations where an unexpected obstacle, such as a car on the road, a driver may take evasive action to avoid a collision. However, this evasive maneuver can lead to unintended accidents, such as hitting other vehicles.
- **Large vehicle scenarios:** This category encompasses scenarios involving complex maneuvers or interactions that could be particularly challenging for larger vehicles. They account for 28.54% of the total scenarios. Several cases include (1) Large vehicle intersection merge. At intersections, large vehicles may face challenges in merging into traffic due to their longer length. Collisions can occur when they merge into lanes occupied by smaller vehicles; (2) Large vehicle turn collision. Larger vehicles, such as trucks or buses, may have difficulty making tight turns at intersections. In this scenario, a large vehicle may collide with a smaller one while attempting to negotiate a turn, potentially due to insufficient turning radius or blind spots.
- **Non-compliance scenarios:** These scenarios involve a failure to comply with traffic rules, such as running red lights or stop signs. As we discuss the scenario

for autonomous driving, this category is not within our interest but still accounts for 7.14% of all scenarios.

- **Others:** We also identify some other low-chance scenarios that rarely happen, such as *Non-Collision Incident*, *Animal Crash Without Prior Vehicle Maneuver*, accounting for 1.6% of all scenarios.

We can categorize these accident scenarios into two types: ‘direct’ and ‘indirect’. A *direct* accident is caused due to the wrong action of the vehicles. An *indirect* accident is caused due to other vehicles that are not involved in the accident, e.g., the vehicle may need to avoid collisions with a passerby vehicle, which causes a collision with another vehicle.

4.4.2 Scenario Description Module

According to the scenarios extracted by the SEM module, we create initial scenarios that are subsequently passed to the next module for generating specific attack scenarios. In STFA, the instantiation process comprises two phases: logical scenario construction and concrete scenario initialization.

Logical Scenario Construction. It includes the following elements. (1) Initial scene construction. This involves the setup of the environment and NPC vehicle settings. The environment includes parameters like the map, road type, time, weather, etc. NPC vehicle settings involve specifying the number of NPCs and their attributes, such as position, type, heading, etc. (2) Scenario construction. This primarily focuses on defining the initial and target states of the ego vehicle, and the trajectories of the NPC vehicles, such as states and events. A “state” comprises details like the position, heading, and speed of a vehicle, while an “event” describes actions undertaken by NPC vehicles. For example, an NPC vehicle may perform a lane change at a specific waypoint indicated by the map.

It is imperative that the created scenarios should adhere to realistic road maneuvers. This entails ensuring that NPC vehicles operate in the correct position, directions, and speeds and do not violate traffic regulations, such as running red lights or crossing yellow solid lines, etc.

Concrete Scenario Initialization. We apply a script that is designed for automatic ADS testing [166] to systematically model scenarios and the corresponding

specifications of the ADS. The script comprises two key components: a scenario description language, enabling actions like modifying vehicle speeds at any given point, and a comprehensive assertion language for precisely defining the correctness of AV behaviors. STFA boasts a dynamic typed design, allowing for the direct assignment of values to variables. Furthermore, it provides a range of elementary types that prove to be useful for both scenario description and specification definition. The following types are basic types: (1) Basic types including Strings, Real Value, and Coordinates; (2) Special types including Position, CoordPositions, LanePositions, Heading, State, ObjectType, Weather, Time, Motion, and Trace; (3) Composite type including AV (defining the motion task of the ego vehicle), Vehicle (defining NPC vehicles), Pedestrian (defining pedestrians), Obstacle (defining static obstacles), and Environment (defining the day of time and weather). We provide an example for better understanding.

An initial scenario. Figure 4.7 presents an initial scenario generated by the description script shown in Listing 4.1, i.e., the input of an adversarial scenario that afterward be sent to ASM. In this example, we describe an overtaking scenario with a slowly moving NPC vehicle ahead of the ego vehicle. There are a total of 3 NPC vehicles and the ahead one (CV) moves with 0.5m/s. To describe the concrete scenario, we need to describe the motion task of the ‘ego vehicle’ who is the victim one, and the motion behavior of three NPC vehicles, `npc1`, `npc2`, and `npc3`.

- Ego vehicle `ego_vehicle`: The ego vehicle is to move from the start point, which is on lane `lane_221` and 1 meter away for its start position, to the target location, which is on Lane `lane_230` and 50 meters away from the lane’s start position in the San Francisco map.
- NPC vehicle `npc1`. It is the CV and 20 meters ahead of the ego vehicle with the same orientation. It runs in the same direction as the ego vehicle on Lane `lane_230`. The initial speed is 0.5 m/s.
- NPC vehicles `npc2` and `npc3`. We can describe the motion of `npc2` and `npc3` similarly. They are 30, and 40 meters ahead of the ego vehicle, and also run on Lane `lane_221`, where the initial speed is 1 m/s and 1.2 m/s, respectively. It is in general not necessary to specify the full trajectory of an agent. Rather, STFA will take the partial trajectory and generate a complete trajectory automatically.
- Time and weather. The time is set as 12:00. The weather is rainy with 0.5 density and the wetness is heavy, respectively.

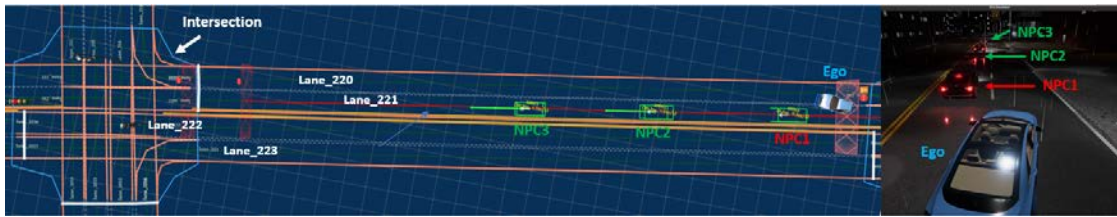


FIGURE 4.7: The initial scenario generated by Listing 4.1 in Apollo Dreamview and LGSVL. The ego vehicle moves from lane_221 to lane_220; NPC1–NPC3 start to move along lane_221 and then randomly select a connected lane in the intersection.

- Pedestrians and obstacles. We do not provide the details of pedestrians and statics obstacles in this chapter, and we give explanations in Section 4.5.

The example code for the statements is given in Listing 4.1. It is notable that users must specify the map `load(map_name)` to be loaded.

```
// overtaking with slowly moving NPC ahead
map_name = "san_francisco";
car_model = "Lincoln2017MKZ";
initial_position = "lane_221" -> 1;
ego_init_state = (initial_position);
target_position = "lane_230" -> 110;
ego_target_state = (target_position);
vehicle_type = (car_model);
ego_vehicle = AV(ego_init_state, ego_target_state, vehicle_type);

//Describe the motion of three NPC vehicles with low speeds
npc1_init_position = "lane_221" -> 20;
npc1 = Vehicle((npc1_init_position,,0.5));

npc2_init_position = "lane_221" -> 30;
npc2 = Vehicle((npc2_init_position,,1));

npc3_init_position = "lane_221" -> 40;
npc3 = Vehicle((npc3_init_position,,1.2));

//Describe Environment
time = 12:00;
weather = {rain:0.5, snow: 0.1, wetness: heavy};
evn = Environment(time, weather);
scenario0 = CreateScenario{load(map_name);
    ego_vehicle;
    {npc1, npc2, npc3};
    { }; // no pedestrians;
    { }; // no obstacles;
    evn;
};
```

LISTING 4.1: A demo script for scenario description.

Shown in Listing 4.2 is the safety specification to be attacked. It states that the ego vehicle should finally arrive at the destination within a given region (i.e., $F \text{ dis_target} \leq 2.0$), by keeping a safe distance with the three NPC vehicles (i.e., statement1 to statement3).

```
Trace trace = EXE(scenario0);

ego_vehicle_state= trace[ego];
npc_vehicle1_truth= trace[truth][npc1];
npc_vehicle2_truth = trace[truth][npc2];
npc_vehicle3_truth = trace[truth][npc3];

dis_target = dis(ego_vehicle_state, target_position);

dis1 = dis(ego_vehicle_state, npc_vehicle1_truth);
dis2 = dis(ego_vehicle_state, npc_vehicle2_truth);
dis3 = dis(ego_vehicle_state, npc_vehicle3_truth);

statement1 = dis1 >= 1.0;
statement2 = dis2 >= 1.0;
statement3 = dis3 >= 1.0;

trace |= (G(statement1 & statement2 & statement3)) & (F dis_target <= 2.0);
```

LISTING 4.2: Attack specification description.

4.4.3 Attack Exploitation Module

This module aims to generate attack scenarios from the initialized scenarios. In detail, STFA introduces a feedback-guided algorithm to automatically search for attack scenarios that are triggering violations of the accidental specification. The primary goal is to detect scenarios that cause assertion failures and result in accidents in the given context. The overall attack scenario generation algorithm is depicted in Algorithm 1, which employs a genetic algorithm (GA) based on a built script s and the target specification A . The ultimate output is a collection of failed cases that are permitted by s but fail to satisfy A . To initiate the process, we begin by randomly generating a test suite using the scenario described in the STFA script, which serves as the initial population (Line 1). Subsequently, we execute the scenarios in the simulation platform and compute the fitness for each scenario, and the scenarios violating the specification will be added to the set S (Lines 4-7). Then, we employ selection, crossover, and mutation operations in GA to produce different generations of scenarios (Lines 10-13).

Algorithm 1 Attack Fuzzing Algorithm.

INPUT: Scenario description $s = \{trajectory(V_i, t), trajectory(p_j, t), Type(V_i), RoadType\}$, target specification A , maximal generations N , population size n , and crossover and mutation probabilities c_p and m_p .

OUTPUT: Adversarial scenarios that violate S .

- 1: Initialize an initial population T_0 based on s ;
- 2: $gen = 0$;
- 3: **for** $gen \leq N$ **do**
- 4: **for** each test case s_i in T_0 **do**
- 5: Execute s_i and verify its execution;
- 6: **if** s_i violates A **then**
- 7: $S = S \cup \{s_i\}$;
- 8: **end if**
- 9: **end for**
- 10: $T = \emptyset$;
- 11: Select n individuals with repetition from T_0 , denoted as T ;
- 12: Perform crossover and mutation on T with c_p and m_p ;
- 13: $T_0 = T, gen = gen + 1$;
- 14: **end for**
- 15: **return** S .

TABLE 4.1: Encoding of vehicles.

	Ego	NPC_vehicle1 (CV)	NPC_vehicle2 (OV)	NPC_vehicle3 (OV)
Offset (m)	1	20	30	40
Speed (m/s)		0.5	1.0	1.2
Type	Lincoln	Sedan	Sedan	Sedan

Note that to apply GA in Algorithm 1, we need to encode the scenario described in the script. Here we add a more description for the encoding process. As shown in Figure 4.4, a driving scenario should contain the ego vehicle and NPC vehicles, as well as the static environment including road layout and static obstacles. For ego vehicle, the vehicle type, color, and size do not affect the execution of the decision-making, so we only parameterize its initial position. For NPCs, they should adhere to the following physical constraints: (1) Spatial constraint. Vehicles should drive along the lanes. For each NPC vehicle, the initial position cannot be partially overlapped, they are constrained to be at least a few meters away from each other. (2) Temporal constraint. The movement of all NPC vehicles must adhere to real-world scenarios. Each NPC vehicle’s speed and position offset are structured sequentially, following the sequence of waypoints, which includes the initial and target waypoints. To represent the NPC vehicles’ speed and position, their values are concatenated in the same order to form the chromosomes of vehicle speed and vehicle position, respectively.

In the sequel, we give a brief description of the implementation of this module.

As given in Figure 4.4, it contains the Parser, Simulator adaptor, ADS adaptor, Violation monitor, and Mutation engine.

Parser. The parser uses ANTLR4 (ANother Tool for Language Recognition)¹, to parse the textual scripts and extract the adversarial scenarios. ANTLR4 is a powerful tool for language recognition, enabling efficient processing of the input scripts and facilitating the extraction of relevant information related to adversarial scenarios. Its robustness and flexibility make it a suitable choice for this task, ensuring accurate parsing and effective scenario identification.

Driving simulator and adaptor. In this work, we used simulators, such as LGSVL, instead of real vehicles for safety and cost considerations. Specifically, performing dynamic scenarios in the real world requires access to a dedicated test site and diverse NPC vehicles, which can be prohibitively costly to set up. Therefore, a simulator adapter is required to receive the scenario description and spawn it into the simulator. It should be customized according to the specific simulator. (1) STFA is not limited to specific simulators and can be flexibly adapted to different simulators; (2) This is a kind of engineering work, more details can be seen on our anonymous website.

ADS adapter. Once the ego vehicle reaches a stop state (e.g., arrives at the destination, causes a collision, or reaches the maximal motion time), the execution is completed. The ADS adapter collects and processes the ADS messages to generate the state trace for specification checking.

Violation monitor. It receives STL assertion from the parser and monitors whether the execution of a testing scenario can violate the assertion. Once a set of inputs causes a policy violation, such scenarios will be stored and reported. Since the accident report does not provide specific parameters for the accidents, and there exists a gap between the real-world scenario and the simulator environment, we define violation attacks within the simulator. For instance, a violation may be triggered if the distance between vehicles falls below 1 meter, or if the ego vehicle stops within 2 meters of the task location.

Mutation engine. The goal of the mutation engine is to generate attack scenarios using the algorithm described in Algorithm 1. In summary, the NPC vehicle type & speed & position are mutated. Since we aim to achieve *Direct-attack* and

¹<https://github.com/antlr/antlr4>

TABLE 4.2: Implementation complexity of BatAV.

	Main Language	File	LoC
	JavaScript	26705	2427369
	C#	80	18465
Attack	C/C++ Header	33	2128
Exploitation	C/C++	40	9586
Module (AEM)	Python	47	26758
	Java	2	11900
	ANTLR Grammar	1	529
Total		26908	2496735

Indirect-attack, for the former one, we randomly select an individual NPC vehicle to construct attack scenarios; for the latter, we randomly select more than 2 NPC vehicles. To realize the feasibility of motion, the mutations should adhere to certain constraints. Specifically, these constraints dictate that NPC vehicles should not move backward when in motion, and their speed must be equal to or greater than 0 m/s. To achieve these, we apply Gaussian mutation for NPC position and speed and we use the clipping function to guarantee that all speeds are positive, with no backward motion in the waypoints of NPC vehicles. Table 4.2 provides the implementation complexity of the AEM in STFA with respect to LGSVL+Apollo.

4.4.4 Attack Report Module

This module aims to classify each adversarial scenario into an attackable type based on the accident scenarios generated by SEM. It consists of two types of attacks including *Direct-attack* and *Indirect-attack*, which are further classified into 7 types of attacks. Specifically, for the *Direct-attack*, it includes:

Direct low-speed attack: attackers intentionally operate CV at lower speeds to disrupt the motion planning of the ego vehicle. In this case, the attacker can control an NPC vehicle and drive in front of the ego vehicle at a very low speed (i.e. less than 1 m/s). This will cause the AV to either follow at a low speed, causing task delay, directly stop on the road, causing traffic congestion or collide with the ahead vehicle when performing slow overtaking. However, the vehicle should speed up and overtake in such cases.

② *Direct parallel-vehicle attacks*: In this scenario, the attacker controls the NPC vehicle to drive in the direction the ego vehicle intends to change lanes. Doing

so would interfere with the ego vehicle's capability to change lanes, causing the self-driving vehicle to reroute instead of changing lanes.

③ *Direct motion-control attacks*: In this case, the attacker operates the CV and positions it strategically around the autonomous vehicle. By manipulating the motion patterns of the NPC, they can induce the AV into colliding or performing unexpected actions, jeopardizing its safety.

④ *Direct large-vehicle attacks*: These involve the attacker taking control of a heavy and large vehicle, such as a bus, to impact the behavior of the AV and provoke collisions or dangerous situations.

The advantages of *Direct-attack* not only involves more direct control of the NPC vehicles with no intermediate steps of interactions with the other NPCs but also with high adaptability. The attacker can adjust the actions and path of the CV in real-time based on the ego vehicle's responses, e.g., if the ego vehicle starts making an emergency turn to avoid a collision with the CV, the attacker can quickly adjust the speed and direction of the CV to maintain contact with the ego vehicle and continue conducting the attack.

For the *Indirect-attack*, it includes:

⑤ *Indirect low-speed attack*: In this case, the attacker leverages CV to drive at a very slow speed in front of the OV, who drives in front of the ego vehicle, thereby indirectly affecting the ego vehicle.

⑥ *Indirect motion-control attacks*: In the indirect attack setting, the attacker uses the CV to continuously change the motion strategy of the OV in front of the ego vehicle. Such motion patterns could induce the ego vehicle and could lead to a crash or hazardous situation.

⑦ *Indirect large-vehicle attacks*: The attacker can control the NPC vehicle in front of the large vehicle, forcing the large vehicle to slow down at the turn and wait for the ego vehicle to drive over.

The advantages of *Indirect-attack* are high stealthiness and less legal liability. The attack indirectly targets the ego vehicle by using CV to influence innocuous entities like OV in the traffic scenario. The attackers may be able to distance themselves

TABLE 4.3: Average attack success rate (ASR) of *Direct-attack* on different scenarios.

Adversarial Attacks	NPC Number	Scenarios	Consequence & Occurrence Rate			Desired Planning Behavior	ASR
			Collision	Stop/Block	Time/Task-delay		
Low-speed attack	1	T-Junction	20%	20%	60%	Overtaking	100%
		Intersection	40%	30%	30%	Overtaking	100%
		Straight	20%	20%	60%	Overtaking	100%
Parallel-vehicle attacks	1	Straight	10%*	20%	60%	Overtaking or Lane following	90%
Motion-control attacks	1	T-Junction	10%	20%	70%	Wait and overtaking	100%
		Intersection	10%	20%	70%	Wait and overtaking	100%
		Straight	10%	30%	60%	Overtaking	100%
Large-vehicle attacks	1	T-Junction	70%	0%	0%	Stop and wait	70%
		Intersection	80%	0%	0%	Stop and wait	80%

* When the parallel vehicle is a bus, the rear of the ego vehicle overtakes the bus and collides with its right side.

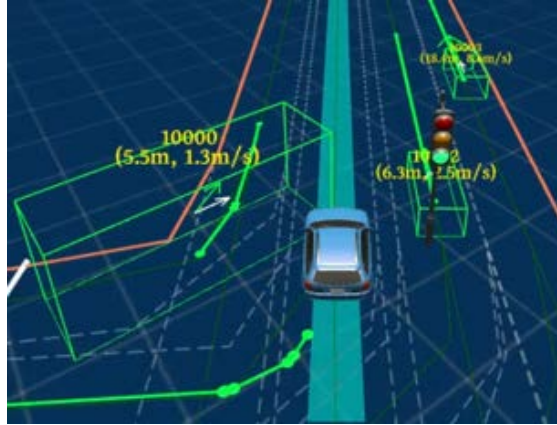


FIGURE 4.8: Demo of direct large-vehicle attack. The attack-controlled school bus occupies two road lanes, and the ego vehicle fails to stop and hits on the school bus.

from the immediate consequences of the attack, making it difficult for detection systems to trace the attack.

In this section, we present a detailed evaluation of STFA in LGSVL+Apollo. It is worth mentioning that STFA is designed to be highly versatile. Users have the flexibility to integrate any ADSs and simulators of their choice. For different configurations, ADS adaptors and simulator adaptors can be customized to interface with specific ADS and simulator platforms, respectively. In this thesis, LGSVL+Apollo allows us to demonstrate the capabilities and performance of STFA.

Evaluation metric. We consider an attack successful when the assertion violations are triggered, leading to any of the consequences of an accident occurring. We diligently monitor each recorded video and define the attack success rate (ASR) as the ratio of the number of accidents in each run to the total number of tests.

Evaluation methodology. In our experiments, we initialize 10 scenarios for each attack, and each initial scenario is described as a script. According to the

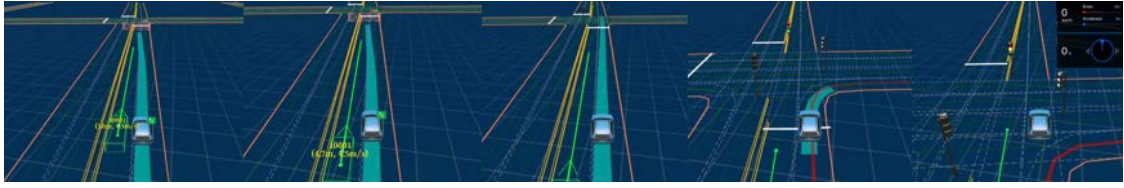


FIGURE 4.9: The demo of direct parallel-vehicle attack. (1) The destination is at the end of the left lane. The controlled NPC runs at the left lane; the ego vehicle tries to overtake the NPC. (2) The ego vehicle still generates the overtaking trajectory. (3) The ego vehicle gives up on overtaking and fails to reach the destination. (4) It regenerates a new path planning trajectory by turning right. (5) The ego vehicle fails to turn right and stops in the center of the intersection.

scenario described in each script, we generate an initial population with a size of 20, and the number of generations is 25, resulting in 520 test cases. We repeat STFA 10 times for each initial scenario. Therefore, for each attack type, we generate 10 (initial scenarios) $\times 520$ (mutated scenarios) $\times 10$ (repeated times) = 52,000 test cases. Additionally, we replicated the movements of all NPCs in one scenario and then demonstrated the generalizability of each attack in other scenarios. For example, in the overtaking scenario, we repeated the test on the remaining 9 roads on the map.

Effectiveness of *Direct-attack*. Throughout our experiments, STFA can cover all the direct attacks targeting at Apollo as shown in Table 4.3:

- *Scenarios for direct low-speed attack.* When the CV drives ahead of the ego vehicle with a slow speed, generally slower than 1m/s, the ego will not perform overtaking, which takes a long time to complete the motion task or stop in a middle way, or collide with the CV. This happens on any type of road. Replacing the ahead CV with any vehicle type yields the same attack effects. Specifically, the direct low-speed attack can achieve a 100% attack rate under 30 times with different straight lane, T-junction, and intersection road types.
- *Scenarios for direct parallel-vehicle attack.* In this case, we assign a CV running in the left lane of the ego vehicle with the same direction and speed. However, the ego vehicle cannot take lane changing action since the parallel CV occupies the left lane all the way, causing the ego vehicle to not reach the destination and fail to complete the task. Shown in Figure 4.9 is a demo of a direct parallel-vehicle attack. We clearly observe that the ego vehicle generated a future path for lane changing and tried to take action, but it fails and finally stops at the

TABLE 4.4: Average attack success rate (ASR) of *Indirect-attack* on different scenarios.

Adversarial Attacks	NPC Number	Scenarios	Consequence & Occurrence Rate			Desired Planning Behavior	ASR
			Collision	Stop/Block	Time/Task-delay		
low-speed attack	2	T-Junction	20%	20%	60%	Overtaking	100%
	4	Intersection	60%	20%	20%	Overtaking	100%
	2	Straight	30%	30%	40%	Overtaking	100%
motion-control attacks	3	T-Junction	0%	40%	60%	Wait and overtaking	100%
	3	Intersection	10%	20%	70%	Wait and overtaking	100%
	2	Straight	10%	30%	40%	Overtaking	80%
large-vehicle attacks	3	T-Junction	90%	0%	0%	Stop and wait	90%
	4	Intersection	90%	0%	0%	Stop and wait	90%

center of the intersection. We evaluate this attack in a straight lane 30 times and find that this attack would lead to three consequences, i.e., the rear end of the car collides with the CV, the ego vehicle stops, and the task cannot be completed. The reason behind this is a parallel CV is continuously occupying the left lane along with the ego vehicle’s direction and speed. Although the ego vehicle generates a future path for lane changing and attempts to initiate the action, it is unable to do so successfully due to the parallel CV, leading to the ego vehicle coming to a stop at the intersection.

- *Scenarios for direct motion-control attack.* In this case, an adversary controls the CV running ahead of the ego vehicle. However, different from low-speed attack, it controls the CV’s position and steering. Table 4.3 shows that it can make the ego vehicle cannot reach the destination or stop halfway, with an overall 100% success rate.
- *Scenarios for direct large vehicle attack.* In this case, an adversary controls a large vehicle, e.g., a school bus, to conduct the attack. Although in the direct low-speed attack, we have confirmed that a large vehicle can make the ego vehicle fail at low speed, in this attack, we only consider turning situations, such as the T-junction and intersection. Specifically, the ego vehicle will drive straight through T-junctions or intersections, while the large vehicle will turn and drive into adjacent lanes in the same direction as the ego vehicle. Table 4.3 shows that the attack can achieve an average 75% ASR and all collide with large vehicles. The reason is that when the large vehicle turns, it will also occupy the adjacent lane, and the ego vehicle cannot detect such occupation but still continues its motion, resulting in a rear-end collision.

Effectiveness of *Indirect-attack*. STFA generates attack scenarios covering the 3 indirect attacks, including:

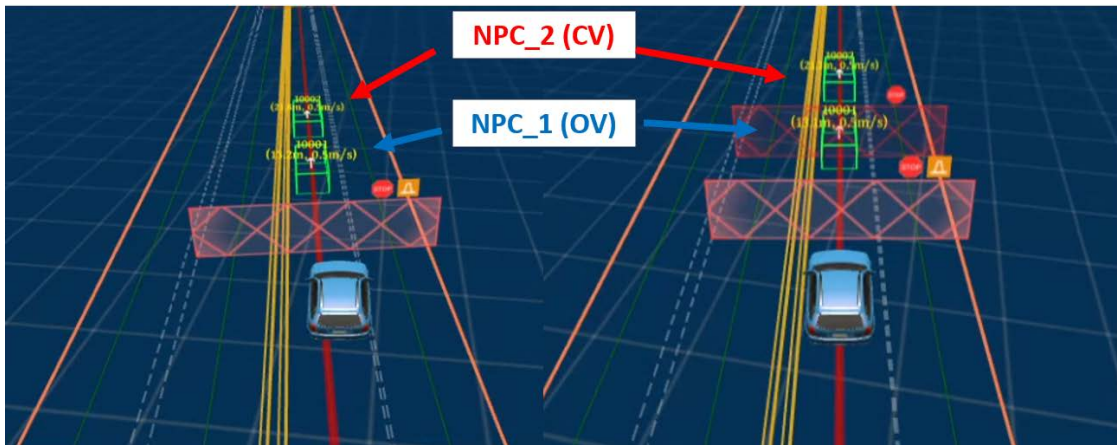


FIGURE 4.10: An example of an indirect slow-speed attack. In this scenario, the first CV deliberately drives at a slow speed, causing the second NPC vehicle to slow down and eventually come to a stop. Consequently, the ego vehicle is affected, and it also slows down and comes to a stop due to the actions of the adversarial vehicles.

- *Scenarios for indirect low-speed attack.* As shown in Figure 4.10, the CV decelerates, causing the OV to drive at a low speed, which indirectly causes the ego vehicle to stop and follow at a low speed. Similar to the direct slow-speed attack, the ego vehicle should overtake at this time but it fails. Table 4.4 gives the occurrence rate and attack success rate of indirect low-speed attack, we observe that this attack can 100% attack the ego vehicle and cause a collision, stop, or task delay consequence at different types of roads.
- *Scenarios for indirect motion-control attack.* In this case, the adversarial vehicle running in the left lane of the ego vehicle and OV, the adversary can adjust its behavior and affect the speed of OV afterward making the ego vehicle always drive after OV. It can lead to stopping or task-delay behavior of the ego vehicle. As we observe in Table 4.4, it achieves an average ASR of 93.9%.
- *Scenarios for indirect large-vehicle attack.* An adversary controls an NPC vehicle, regardless its types, the adversary can control the NPC vehicle, regardless of its type, and can force the large vehicle to wait until the ego vehicle comes to start and pass the turn, and the ASRs are 90% for T-Junction and Intersection, respectively.

4.4.5 The Similarity and Difference between *Direct-attack* and *Indirect-attack*

Similarity. Given the results in Table 4.3 and Table 4.4, it is evident that both attack types can achieve high ASRs. We also observe that both two attacks only induce the collision consequence in Large-vehicle attacks. The main reason is that large vehicles usually need to occupy two to three lanes when turning, and Apollo ADS believes that the vehicles only occupy one lane, so a collision occurs, as shown in Figure 4.8.

Difference. We observe *indirect-attacks* have higher ASRs than *direct-attacks*. The reason is the more NPCs that participate in the *indirect-attack*, the more intricate the traffic scenario becomes, increasing the possibility of accidents. This is due to the fact that the behavior of NPCs in the scenario is dynamic and often unpredictable, and their actions can inadvertently trigger collisions. However, we also claim that *indirect-attack* is more difficult to perform than *direct-attack* in the real world. In the case of *indirect-attack*, multiple OVs are intended to be positioned around the ego vehicle simultaneously, which has a lower possibility of happening in the real world as we cannot control the OVs' positions. This necessitates that the attacker relies on the current location of the ego vehicle and the surrounding traffic conditions, including the positional relationship between the OV and the ego vehicle, to execute the attack. Consequently, when compared to the *direct-attack*, the *indirect-attack* presents greater complexity. However, the advantage of the *indirect-attack* lies in its higher level of stealthiness, as it does not directly influence the behavior of the ego vehicle.

4.5 Limitation and Discussion

Information extraction ability. While employing ChatGPT-4 for information extraction offers valuable insights, e.g., road types, NPC characteristics, and attack consequences, it comes with certain limitations that need to be acknowledged. ChatGPT-4 may have limitations in its contextual understanding of accident reports. As a black-box language model, it relies on the context provided in the input prompt, which might not be sufficient to fully grasp the complexities of accident

scenarios, i.e., we cannot guarantee its extraction effectiveness. This limitation can result in incomplete or inaccurate information extraction. To mitigate these limitations, we carefully scrutinized accident reports and found that the information extracted by ChatGPT-4 is meaningful. Also, we have demonstrated the feasibility of achieving these attacks. As part of future work, users can work towards developing specialized prompt tools to enhance information extraction ability from ChatGPT-4. These custom prompt tools could be tailored to the specific requirements of the domain, ensuring better alignment with more accurate accident information.

NPC pedestrians and obstacles. STFA have included NPC pedestrians and obstacles, as depicted in Listing 4.1 lines 27-28 to construct scenarios. However, due to several critical reasons, we refrain from employing pedestrians and obstacles in STFA. Firstly, although the simulation world allows for scenarios with pedestrians, it is not feasible or meaningful to create such scenarios in the real world, where an attacker can remote an NPC vehicle but cannot control a person, making such scenarios impractical. Secondly, we focus on dynamic attack scenarios rather than static ones. Using static obstacles that are easily detected by human observers or AV sensors potentially compromises the stealthiness of attacks.

4.6 Conclusion

In this chapter, we address the critical security concerns surrounding the decision-making module in ADS and we conduct the first SOTIF-oriented traffic flow attack of ADS. Rather than designing a specific attack, we propose STFA, a novel automatic attack generation approach. STFA introduces two novel attack types, i.e., *Direct-attack* and *Indirect-attack*. Based on that, we design a total of 7 new attack methods, and conduct comprehensive experiments to demonstrate the effectiveness and generalizability of these attacks. STFA contains more than 2.5 million lines of code, which can be easily extended to any other ADS, providing practitioners with a valuable tool to enhance their ADS security. In the future, we hope developers can design corresponding defense strategies based on it. Furthermore, we will also integrate more unmanned systems, such as drones and robotic systems.

Chapter 5

A Unified Defense Framework against Physical Adversarial Attacks to Autonomous Driving Visual Perception

Modern Autonomous Vehicles (AVs) implement the Visual Perception Module (VPM) to perceive their surroundings. This VPM adopts various Deep Neural Network (DNN) models to process the data collected from cameras and LiDAR. Unfortunately, prior studies have shown that these models are vulnerable to physical adversarial examples (PAEs), which pose a critical safety risk to the autonomous driving task. While a few defense methods have been proposed to safeguard AVs, most of them only target a limited set of attack types and specific scenarios, making them impractical for real-world protection.

In this chapter, we introduce **VisionGuard**, a novel and practical defense framework leveraging spatiotemporal inconsistency that can comprehensively detect and mitigate various PAEs to the VPM. **VisionGuard** consists of three modules: (1) State Correction Module (SCM) obtains the current driving states by raw data calibration and integration; (2) State Prediction Module (SPM) predicts the motion states by tracking historical states; (3) Attack Detection Module (ADM) checks the motion state inconsistency. We evaluate 9 state-of-the-art PAEs against both camera and camera-LiDAR fusion-based object classification & detection models.

Experimental results in both simulation and the physical world validate the effectiveness and robustness of `VisionGuard`. Codes and demo videos can be found on our anonymous website: <https://sites.google.com/view/visionguard>.

5.1 Introduction

Autonomous driving technology has been widely commercialized, as evidenced by the increased types of Autonomous Vehicles (AV) transitioning from concepts to real products on public roads. The brain of these vehicles is the Autonomous Driving System (ADS), which incorporates multiple modules to understand the external environment and make safe and accurate driving decisions. One important module is the Visual Perception Module (VPM), which leverages the camera and LiDAR as the primary sensors for perceiving the surrounding context of the vehicle, and then uses state-of-the-art Deep Neural Network (DNN) models for object classification and detection. This VPM lays the foundation of the ADS and plays a critical role in ensuring safe driving.

However, recent studies have exposed camera and LiDAR spoofing vulnerabilities against ADSs [167]. An external adversary can carefully craft adversarial objects to deceive the DNN vision models to make wrong perception results. Such incorrect results will be further fed into the subsequent modules, leading to incorrect driving decisions and endangering the safety of AVs. Based on the attack goals, the adversary can mislead the perception models to misclassify an object [62–65], hide an existence object [16, 63, 64, 66–70], or recognize a non-existence object [71–73]. For instance, pasting a sticker onto a stop sign can cause the traffic sign classification model to misrecognize it as a speed limit sign, potentially making the vehicle fail to decelerate at a pedestrian crossing [62]. Similarly, strategically placing a 3D-printed traffic cone in the middle of a driving lane can deceive the camera-LiDAR fusion-based object detection mechanism, resulting in a collision risk [16].

Numerous defense methods have been proposed to mitigate adversarial examples [20]. They can be roughly classified into two categories. (1) *Certified defenses*. It detects adversarial patches with theoretical guarantees of model robustness against white-box attacks. Typical techniques include randomized cropping [21], interval bound propagation [22], de-randomized smoothing [23], secure aggregation [24],

TABLE 5.1: Comparison with representative state-of-the-art defense methods. HA: hiding attack; AA: appearing attack; MA: misclassification attack.

Defense Methods		Sensor		Task		Attack Goal		
		Camera	LiDAR	Classification	Detection	HA	AA	MA
Certified	Randomized Cropping [21]	✓		✓				✓
	Interval Bound Propagation [22]	✓		✓				✓
	De-randomized Smoothing [23]	✓		✓				✓
	Secure Aggregation [24]	✓		✓				✓
	PatchGuard++ [25]	✓		✓				✓
	Certified Training [26]	✓		✓				✓
	DetectGuard [27]	✓			✓	✓		✓
Vision Consistency	PercepGuard [28]	✓		✓	✓		✓	✓
	AdvIT [29]	✓			✓			✓
	SCEME [30]	✓			✓	✓	✓	✓
	SCENE [32]	✓			✓	✓	✓	✓
	KEMLP [33]	✓		✓				✓
	Zhang et al. [34]	✓	✓		✓		✓	✓
Spatiotemporal Consistency	VisionGuard (Ours)	✓	✓	✓	✓	✓	✓	✓

feature space masking [25], certified training [26] and objectness explaining [27]. (2) *Vision-based consistency checking*. It leverages the consistent information obtained from different sources to detect adversarial examples. Such consistency detection can be achieved at the perceptual level [28, 29], physical context level [30, 32, 33], or sensor level [34].

However, most of them are evaluated in model level and also suffer from several limitations. **Limited generalizability.** Existing methods only target specific vision tasks, sensors, or attack goals, as shown in Table 5.1. Specifically, certified defenses are mainly designed for physical 2D patch attacks other than 3D LiDAR attacks. They either focus on the classification task [21–26], or object detection task [27], but are not able to cover all the vision tasks in the VPM. For vision-based consistency checking, some methods [28, 29] extract and monitor anomalies in motion feature consistency of the target object. They can only detect misclassification attacks, but not object-hiding attacks since there are no targets for feature extraction. The second one is **reliance on contextual information**. Many vision-based consistency-checking approaches highly rely on the availability of abundant contextual information from the perception module. For instance, some solutions [30, 31] leverage reasonable relationships between the target object and coexisting benign objects to identify anomalies. They are less effective when there is barely any object other than the target one in the scene. The last one is **computational efficiency**. Certified defenses normally come with complex derivation and optimization processes which involve heavy mathematical proofs and algorithms to ensure their robustness guarantees. Their high computational

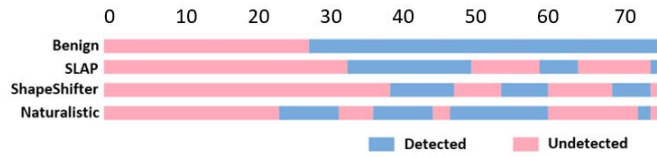


FIGURE 5.1: Runtime detection comparison.

cost makes them impractical to achieve real-time protection in autonomous driving. More discussions of existing defense works can be found in Section 5.2.3.

This chapter aims to propose a practical defense solution to overcome the above limitations. To this end, we perform a comprehensive evaluation of state-of-the-art PAEs¹ to the VPM in both simulation and physical world. This enhances our understanding of such threats and reveals an intriguing phenomenon: *existing PAEs are all less robust to physical variations and difficult to keep consistent over time.* Figure 5.1 shows such inconsistency over different distances in various attacks. To further explore this intriguing phenomenon, we for the first time theoretically demonstrate that it is hard to generate a stationary PAE that remains effective based on distance variation. Last but not least, although a variety of transformation methods are used to enhance the robustness of PAEs [168], we have proven that existing methods still fail to satisfy the expectations.

Motivated by this observation, we propose **VisionGuard**, a practical defense framework to protect the ADS against camera and LiDAR adversarial attacks. **VisionGuard** has the following advantages: **High generalizability.** **VisionGuard** is capable of detecting different forms of PAEs. The key idea is to use the *spatiotemporal inconsistency* present in *AV's internal kinetic behaviors* to detect anomalies. **VisionGuard** predicts the vehicle's future motion based on its past motion states. If this prediction deviates from the actual movement, it will issue a warning, continuously monitor its movement status, and further determine to eliminate the warning or issue an attack warning based on the cumulative threshold. This design provides comprehensive protection over any attack goal, target model, target sensor, etc. **2 No requirements of contextual information from the VPM.** **VisionGuard** utilizes a completely different set of data and sensors (i.e., GPS and IMU) from prior works for inconsistency checking and anomaly detection. This presents a significant benefit as it is not constrained by external and contextual

¹It is also important to note that we have specifically selected PAEs that have been tested in real-road condition in existing literature.

factors, while GPS and IMU are more stable to provide internal reference of the vehicle. **③ Highly computational efficient.** *VisionGuard* captures the motion state of the vehicle with only a few explicit variables, without the need to preprocess high-dimensional image and point cloud inputs. In addition, state prediction in *VisionGuard* is achieved via a lightweight yet effective statistical model, which can satisfy the real-time requirement on resource-constrained platforms (i.e., AV).

The *VisionGuard* framework consists of three modules (Figure 5.5). (1) State Correction Module (SCM) is responsible for obtaining the vehicle’s actual states. It collects the raw IMU and GPU measurements and applies the Kalman Filter algorithm to calibrate the driving state. (2) State Prediction Module (SPM) is used to predict the driving states by tracking historical states. It adopts the Autoregressive Integrated Moving Average algorithm to extract two types of features. (3) Attack Detection Module (ADM) is introduced to assess whether the vehicle is currently safe or under attack. It combines all the predicted states and extracted features to make comprehensive comparisons with a state machine at runtime.

We comprehensively evaluate *VisionGuard* in industry-grade simulators (LGSVL [96] and CARLA [169]) and physical scenarios. Experiment results show that *VisionGuard* is robustness against 9 state-of-the-art PAEs in 9 scenarios with different environment conditions (distance, angle, heading, weather, context information). Compared with two representative defense methods based on certification and consistency-checking, *VisionGuard* exhibits the highest detection rate, fastest detection speed, and best generalization capability. Besides, *VisionGuard* has very low false positives, which only occur in several corner cases. We further conduct a comprehensive analysis of the adaptive attacks and prove that the optimization results of these hyper-parameters in various scenarios are consistent and reliable.

To summarize, we make the following contributions:

- We conduct a comprehensive evaluation of 9 state-of-the-art PAEs in diverse scenarios. This deepens the understanding of perception vulnerabilities in ADSs and discloses the defense opportunities.
- We propose a defense framework *VisionGuard*, which is the first method leveraging GPS and IMU measurements to guard the VPM. *VisionGuard* uses the spatiotemporal inconsistency in the vehicle’s kinetic information to detect different

types of attacks (object misclassification, hiding or appearing) against different sensors (camera, camera-LiDAR fusion) with different attack techniques.

- **VisionGuard** is a plug-and-play solution that can work with off-the-shelf ADSs. It is a flexible framework for high transferability and scalability.
- We evaluate **VisionGuard** in both simulation and real-world scenarios. **VisionGuard** achieves high detection rate with low false positives with a total number of 39000 frames of collected vehicle motion state data.

5.2 Background and Related Work

5.2.1 Visual Perception Module in ADS

A typical ADS implements a Visual Perception Module (VPM) to understand the environment. The VPM is responsible for processing sensor data and applying DNN models to execute perception tasks, such as object detection, classification, and tracking. Based on the perception outputs, the ADS makes decisions, e.g., throttle, braking, and steering, to ensure the AV operates correctly. In this chapter, we focus on the security aspects associated with the VPM. State-of-the-art ADSs typically employ two main approaches for building the VPM: (1) camera-based design, such as Tesla [83] and Intel Mobileye [170]. The ADS relies on 2D or 3D cameras for perception, often using multiple cameras positioned in different locations to improve accuracy and robustness. (2) Camera-LiDAR fusion-based design, such as Baidu Apollo [171] and Google Waymo [2]. The ADS combines both cameras and LiDAR sensors to collect image and 3D point cloud data, respectively. These two modal data are processed separately and then fused to generate the final perception results. This process is normally achieved via a rule-based Multi-Sensor Fusion (MSF) function [171]. We aim to defeat attacks that target both perception designs. We do not consider the LiDAR-only implementation, which has not yet been deployed by the mainstream AV manufacturers in reality.

5.2.2 Physical Adversarial Examples (PAEs)

Machine learning models are vulnerable to adversarial attacks [16, 63, 63, 64, 66, 68–73, 172], where small-scale perturbations in the input can mislead the victim

model to make wrong predictions. Most of these pixel-wise perturbations are nearly imperceptible to human eyes. Despite their stealthiness, many of these attacks utilize the entire input space for perturbation injection. In the physical world, the attacker can achieve such an attack by creating localized perturbation in the form of sticker patches, projection patterns, or 3d-printed obstacles. Although such adversarial objects are subject to physical constraints, their threat to ADSs can have serious consequences.

Physical attacks to camera-based perception. Attacks targeting the camera-based VPM can be classified into three categories based on the attack goals. **Misclassification Attacks (MA)** deceive the models into classifying the target object into another one [64, 66, 172]. **Hiding Attacks (HA)** fool the models into ignoring the presence of the target object [63, 66, 75]. **Appearing Attacks (AA)** aim to make the perception model detect a non-existent object [16, 63, 64, 66, 68–70].

Physical attacks to MSF-based perception. Existing works mainly focus on HA. Cao et al. [16] were the first to successfully 3D-print optimized obstacles, such as benches, toy cars, and traffic cones, to deceive camera-LiDAR fusion-based perception. Abdelfattah et al. [70] proposed a similar HA technique within a comparable threat model. However, there is still limited research on robust black-box attacks against MSF-based perception.

5.2.3 Existing Defenses against Physical Adversarial Attacks

Past works have proposed different types of defense solutions to mitigate the above physical attacks. They can be classified into the following two categories.

Certified defenses. These methods aim to detect adversarial patches, particularly those created by white-box adversaries. Chiang et al. [22] introduced a certified defense through Interval Bound Propagation (CertIBP), which demonstrates superior robustness against adversarial patches of varying shapes and sizes. Levine et al. [23] extended the robustness of certification with De-Randomized Smoothing (DS) by leveraging the spatially constrained properties of adversarial

patches. Similarly, Lin et al. [21] utilized these properties to achieve attack detection by classifying random crops of the input image and generating final outputs based on the majority of the classification results. Metzen et al. [26] improved the efficiency of the certification process by combining it with model training. Several defense strategies are designed specifically to address the localized adversarial patches for CNN models with small receptive fields. For instance, PatchGuard [24] leverages these small receptive fields to limit the impact of corrupted features. It utilizes secure aggregation to retrieve correct prediction results. PatchGuard++ [25] and DetectorGuard [27] are extended over PatchGuard, which applies masks in the feature space to boost the robustness.

Limitations. Certified defenses suffer from several limitations. First, they have scalability and efficiency issues. Their computational cost grows exponentially with the size of inputs, which makes them less practical for real-time applications with high-dimensional inputs, e.g., autonomous driving. Second, while some methods claim to detect physical adversarial patches, their feasibility in physical settings is not well demonstrated. Third, certified defenses are often designed to defend against specific types of attacks, lacking the generalizability and ability to handle diverse attacks.

Vision-based consistency checking. AVs running in the physical world exhibit temporal continuity, which could be disrupted by adversarial attacks. Consistency-based defenses focus on detecting anomalies by utilizing the spatiotemporal information derived from the perceptual, physical, or sensor level. These defenses analyze the consistency of data over time to identify any discrepancies caused by adversarial attacks. (1) Perceptual level. PercepGuard [28] explores the spatiotemporal consistency of the target objects by constantly monitoring its trajectory to detect *MA*. Similarly, AdvIT [29] analyzes the temporal consistency of the target across continuous frames with optical flow estimation of pseudo frames for detection comparison. (2) Physical environment level. Some approaches exploit the consistency properties between the target object and coexisting objects in the scene to detect adversarial attacks. Li et al. [30] created an auto-encoder for each target class to discover whether its contextual discrepancy rules have been violated. Yin et al. [32] employed a language model with an awareness of describing natural scene images to obtain relationships between multiple coexisting objects. Some other researchers assign specific classes with their deterministic attributes.

KEMLP [33] combines these attributes with a set of weak auxiliary models to check the consistency properties of the target object. Wang et al. [31] utilized a similar approach to exploit context inconsistencies specifically for persons across different views to detect adversarial attacks. (3) Sensor level. Zhang et al. [34] checked the consistency of data collected from different cameras to detect optical signal attacks by analyzing the distribution of disparity error between them. Xiao et al. [77] leveraged the global and average local differences between normal and adversarial objects to detect *AA* in the point cloud.

Limitations. These methods that rely on external perceptual information have limitations in generalizability. For example, PercepGuard [28] and AdvIT [29] are only effective against *MA* with norm-bounded perturbations. They may not be applicable to *HA* when contextual information is limited. Approaches that rely on the consistency of coexistent objects assume abundant contextual information, which may not be the case in low-light or rural areas. Additionally, some methods [31, 33] focus on verifying the consistency of static objects, which is not applicable to complex spatiotemporal features associated with moving objects.

5.3 Design Insight

5.3.1 Threat Model

Attack objective and capability. We consider an external attack that aims to compromise the VPM of an AV by deploying 2D or 3D PAEs. The attacker can achieve different types of goals introduced in Section 5.2.2, including *MA*, *HA*, and *AA*. We assume the attacker knows all the details of the target DNN models in the ADS and is able to generate high-quality adversarial objects by considering the physical constraints (e.g., naturalness, printability, etc.).

Attack scope. Following the threat model in [28], we do not consider digital adversarial examples. Executing digital attacks requires the attacker to gain access to the Controller Area Network (CAN) bus to inject digital signals, which is more challenging to accomplish in real-time ADSs. We assume the deployed PAE is stationary. This holds true for the mainstream 2D patch and 3D object attacks.

Dynamic PAEs (e.g., using screen or projector) for better robustness and concealment are beyond the scope of this work. We concentrate on the optimization-based PAEs with high stealthiness, while ignoring other types of attacks (e.g., phantom attack [75]) that cause significant changes to the environment. Additionally, we mainly focus on the attacks against the vision sensors (i.e., camera, LiDAR), while trusting motion sensors (i.e., GPS, IMU). While motion sensor signals can also be spoofed [173], a plenty of countermeasure have been designed to protect the GPS signals, including authentication, integrity verification, encryption-based, and deep learning-based techniques [174]. The inherent design features of IMUs can also confer resistance to external spoofing due to their operational independence and lack of reliance on external signals. How to design a defense solution over all untrusted sensors is challenging and never studied in prior works. This will be our future direction.

5.3.2 Key Insight

The key insight of *VisionGuard* comes from an observation that **existing physical adversarial examples are not perfect in our real world**. The physical constraints render such attacks less robust and consistent against environmental changes, including light conditions or movement. This conclusion has been confirmed by prior works [168, 175, 176]. We also extensively and comprehensively investigate 9 state-of-the-art PAEs, as summarised in Table 1. We download all the videos from these papers and create a database (can be found on our website). We carefully review these videos and observe that none of these attacks can be consistent over time, as shown in the "Inconsistent Period" column of Table 1.

We argue that **it is infeasible to create *perfect* physical adversarial examples *in practice***. Below we provide our justifications from two perspectives.

5.3.3 Difficulty of Generating Perfect PAEs

We first theoretically demonstrate that it is difficult for a stationary PAE to achieve ideal attack results in a dynamic AD scenario. Specifically, we prove that, as a camera mounted on an AV constantly approaches a PAE from far to near, it captures

a multitude of video frames, each presenting a slightly different perspective. This continuous change makes it extremely hard to maintain a high probability that a universal PAE is effective across all these rapidly increasing and varying frames.

Theorem 5.1. *Consider a case where the example x is successfully attacked with the attacking probability to be $P(F(x + \delta) \neq y) = 1 - p_{x+\delta}$ where δ denotes the perturbation. Further, we consider that the image of each frame contains $d \times d$ pixels. Then, the maximum probability P_{\max} of successfully attacking all frames is exponentially reduced with the increasing number N of the total frames, i.e.,*

$$P_{\max} \leq (1 - p_{x+\delta})^{\frac{2N}{d} (\lfloor \frac{\ln \bar{d} - \ln d}{\ln(1-p_{x+\delta}) - \ln(2-3p_{x+\delta})} \rfloor + 1)}, \quad (5.1)$$

where $\bar{d} \times \bar{d}$ represents the number of object pixels at the location when the vehicle starts to see the object.

Proof. For clarity of reading, we here mainly present the sketch of proof. More details can be found in Appendix .1. To establish the proof, we view the problem of attacking object detection to be a binary classification task where the object is either detected or not. We denote y to be the label where the object is recognized and y' otherwise. Besides, we denote $P(F(x) = y) = p_x$, $P(F(x + \delta) = y) = p_{x+\delta}$ for ease of analyzing. Also, we use $l(x, y)$ to represent the loss of classifying the example x to be y . Obviously, we have $l(x, y) < l(x, y')$ for any clean example x . By denoting $f(x) = \frac{l(x+\delta, y') - l(x, y) - \frac{L\epsilon^2}{2}}{\|\nabla_x l(x, y)\|}$, we have

$$\begin{aligned} P(F(x + \delta) = y) &= P(l(x + \delta, y) < l(x + \delta, y')) \\ &\geq P\left(l(x, y) + \nabla_x l(x, y)^T \delta + \frac{L\epsilon^2}{2} < l(x + \delta, y')\right) \\ &\Rightarrow P\left(\frac{\nabla_x l(x, y)^T \delta}{\|\nabla_x l(x, y)\|} < f(x)\right) < p_{x+\delta}, \end{aligned} \quad (5.2)$$

where the first inequality is derived by applying the Taylor expansion over $l(x + \delta, y)$. Considering the perturbation budget $\|\delta\| \leq \epsilon$, we can compute the expectation of the $\frac{\nabla_x l(x, y)^T \delta}{\|\nabla_x l(x, y)\|}$ as

$$\mathbb{E} \frac{\nabla_x l(x, y)^T \delta}{\|\nabla_x l(x, y)\|} \leq p_{x+\delta} f(x) + (1 - p_{x+\delta}) \epsilon. \quad (5.3)$$

Instead of considering the vehicle coming closer to the object, we consider driving the vehicle back from the position where it is parallel to the object for ease of analysis. We use $g(x + \delta) = x + \gamma\delta$ to denote the transformed example of $x + \delta$ where $\gamma < 1$ is the scaled size. Based on the L-smoothness assumption, we have

$$\begin{aligned} & l(x, y) + \gamma \nabla_x l(x, y)^T \delta - \frac{L\gamma^2 \epsilon^2}{2} \\ & \leq l(g(x + \delta), y) \leq l(x, y) + \gamma \nabla_x l(x, y)^T \delta + \frac{L\gamma^2 \epsilon^2}{2}. \end{aligned} \quad (5.4)$$

Similarly, based on (5.4), we have

$$\begin{aligned} & P(F(g(x + \delta)) \neq y) \\ & \leq \frac{\gamma \left(p_{x+\delta} \cdot \left(l(x + \delta, y') - l(x, y) - \frac{L\epsilon^2}{2} \right) + (1 - p_{x+\delta}) \epsilon \|\nabla_x l(x, y)\| \right)}{l(g(x + \delta), y') - l(x, y) - \frac{L\gamma^2 \epsilon^2}{2}} \end{aligned} \quad (5.5)$$

Considering the vehicle stays in the position where the scale size s satisfies $\gamma \leq \frac{1-p_{x+\delta}}{2-3p_{x+\delta}}$, then bringing the scale size of s back to (5) derives the probability bound of $P(F(g(x + \delta)) \neq y)$:

$$P(F(g(x + \delta)) \neq y) \leq 1 - p_{x+\delta}. \quad (5.6)$$

Further, denoting the number of total frames as N and the size of each frame as $d \times d$, we can obtain that each captured picture corresponds to $2N/d$ frames. By further denoting the minimum scale size corresponding to recognizing clean figures is \bar{d} , we can calculate the maximum scale times r of which the attacking probability is less than $1 - p_{x+\delta}$ as $r = \lfloor \frac{\ln \bar{d} - \ln d}{\ln(1-p_{x+\delta}) - \ln(2-3p_{x+\delta})} \rfloor$. We can then compute the number n of frames of which the attacking probability is less than $1 - p_{x+\delta}$ as

$$n = \frac{2N(r+1)}{d} = \frac{2N}{d} (\lfloor \frac{\ln \bar{d} - \ln d}{\ln(1-p_{x+\delta}) - \ln(2-3p_{x+\delta})} \rfloor + 1). \quad (5.7)$$

As a consequence, by denoting the probability of successfully attacking one frame as P_a , the maximum probability of successfully attacking all frames is less than the

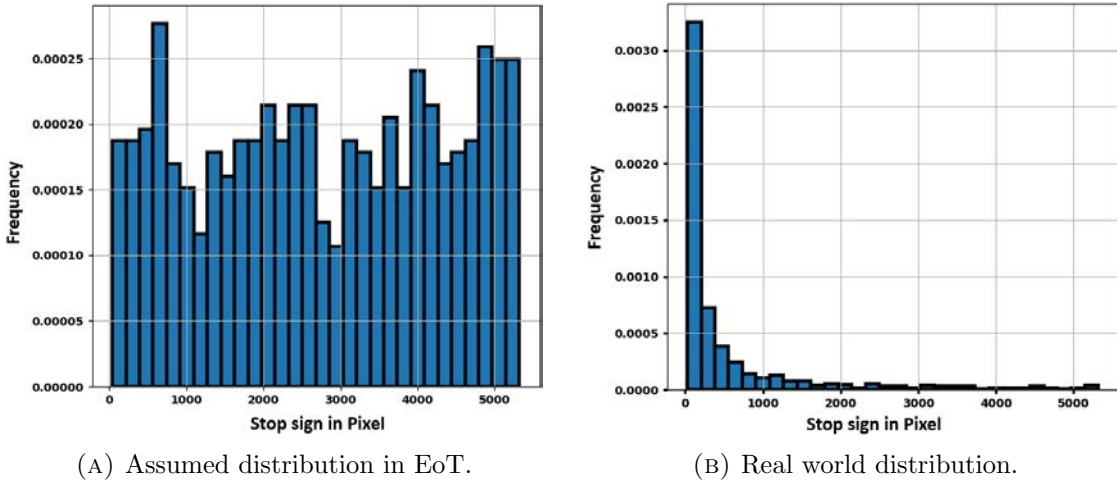


FIGURE 5.2: Distributions of stop sign size in 416*416 images.

following probability:

$$P_{\max} = P_a^N \leq (1 - p_{x+\delta})^n = (1 - p_{x+\delta})^{\frac{2N}{d} (\lfloor \frac{\ln \bar{d} - \ln d}{\ln(1-p_{x+\delta}) - \ln(2-3p_{x+\delta})} \rfloor + 1)},$$

which completes the proof. \square

5.3.4 Ineffectiveness of Robustness-enhanced Solutions

To overcome the physical constraints and make the PAE more robust, many attacks leverage some robustness-enhanced techniques, i.e., Expectation of Transformation (EoT) or its variants, which augment the training of PAEs with uniform transformation. We show that these techniques are still incapable of generating perfectly consistent PAEs from two perspectives. **(1) Unreasonable assumption.** It is a common practice in previous studies to uniformly sample the transformation operation within a certain range [64, 66, 177, 178]. However, Wang et al. [179] demonstrated that this assumption is inconsistent with the physical model. To further verify this, we experiment with a stop sign in the real world. First, we drive a physical vehicle in uniform motion from 50 meters away towards a standard stop sign (600mm diameter), where the Intel RealSense D435i camera with 1920 * 1080 resolution is mounted on the vehicle. We record the video and obtain the stop sign pixel for each frame, for a total of 1470 frames. The stop sign distributions assumed in EoT used in [178] and in the real world are given in Figure 5.2. We observe that in reality, the vast majority of pixel sizes lie in the range

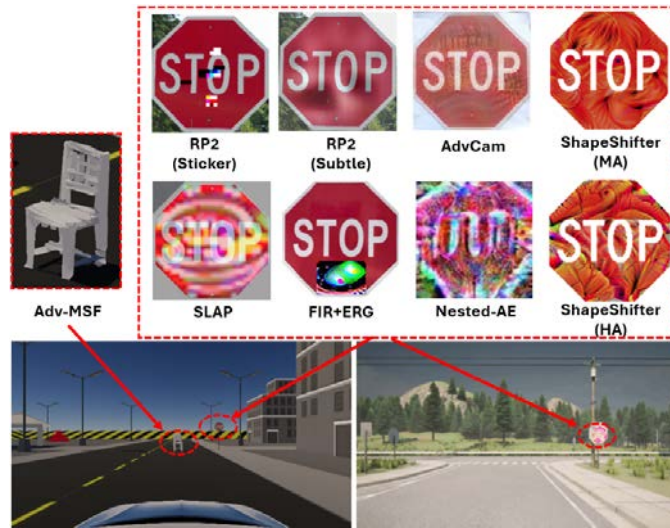


FIGURE 5.3: Adversarial objects in our empirical study.

0 to 200 (small pixel distribution when the AD vehicle is far away from the object). However, [178] implies that when the object is small, attack convergence becomes challenging, making it difficult to attack. As a consequence, uniformly sampling object sizes without consideration of their real-world frequency can lead to less robust PAEs. It might be overly robust to size variations that rarely occur (large pixel distribution) and underprepared for common size variations (small pixel distribution), making the PAE easily affected by distance.

(2) Ignored context. EOT involves adding random distortions during the optimization process to make the perturbation more robust. However, it primarily focuses on the PAE itself, without sufficiently considering the background context, which is critical for AD scenarios. In the real world, the detection model not only analyzes the object of interest in isolation, but also consider its context, including the background and the relationship between the object and its surrounding environment. Thus the background context can significantly impact the effectiveness of PAEs [71]. As vehicles move from far to near, the surrounding environment of the PAE changes quite significantly, yet EOT does not take these background changes into account. Although [71] considers background context and improves EoT, its robustness is still poor due to the limitations posed by the dynamic changes during driving.

TABLE 5.2: Evaluation of Yolov3 in benign and adversarial scenarios in LGSVL. Each result below is calculated with around 300 video frames. In benign scenarios, the objects can be fully detected. In adversarial scenarios, \times , \checkmark and *fail* mean the attack is inconclusive, continuous and fails, respectively.

Goal	Target	0°			30°		60°	Environment		
		3-6m	6-9m	9-12m	3-6m	6-9m	3-6m	Rainy	Foggy	Twilight
Benign	Stop Sign	100%	100%	100%	100%	100%	100%	100%	100%	100%
	Bench *	100%	100%	100%	100%	100%	100%	100%	100%	100%
	SLAP[66]	\times	\times	\times	\times	\times	\times	\times	\times	\times
HA	ShapeShifter[64]	\times	\times	\checkmark	\times	\times	\times	\times	\times	\times
	Adv-MSF [16]	\times	\times	\checkmark	\times	\times	\times	\times	\times	\times
	RP2(subtle)[62]	<i>fail</i>	<i>fail</i>	\times	<i>fail</i>	\times	<i>fail</i>	\times	\times	\times
MA	RP2(art)[62]	<i>fail</i>	<i>fail</i>	\times	<i>fail</i>	\times	<i>fail</i>	\times	\times	\times
	ShapeShifter [64]	\checkmark	\times	\times	\checkmark	\times	\checkmark	\times	\times	\times
	AdvCam [63]	\checkmark	\times	\times	\checkmark	\times	\checkmark	\times	\times	\times

* Adv-MSF [16] only provides “Bench” in their code.

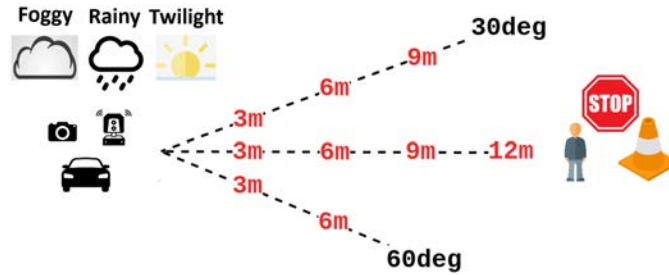


FIGURE 5.4: External environment variables in our consideration.

5.3.4.1 Empirical Study

To further verify these observations, we reproduce 9 representative attacks with different features [16, 63, 64, 66, 172], as illustrated in Figure 5.3. We deploy these attacks in two industry-grade autonomous driving simulators (LGSVL and CARLA) and evaluate the robustness of their vision models. We adopt a “LincolnMKZ” vehicle, which is equipped with a front camera (1920 × 1080, 15Hz) and LiDAR (64-HEL Velodyne). We place the adversarial objects at reasonable places in the scene. For instance, adversarial stop signs are placed on the roadside, while a 3D adversarial bench is placed in the center of the driving lane. We mainly run two types of experiments with respect to the robustness and runtime analysis of these attacks and record the detection results of the target models.

Robustness analysis. We first evaluate the robustness of each attack to various factors, such as distances and angles from the adversarial object, and environmental conditions (e.g., raindrops, fog, lightness). We select 9 different positions of the adversarial object and 3 environment conditions, as shown in Figure 5.4. Table 5.2 shows the evaluation results returned by the respective perception models in the benign and attack (*HA* and *MA*) cases. We can observe that the target models can

correctly and consistently detect or classify benign objects. For the two attacks, the adversarial objects are highly sensitive to distances, angles, and environmental conditions. Based on these results, we conclude that due to the physical constraints, existing PAEs are susceptible to various external factors, and their robustness is restricted.

Runtime analysis. We further perform a runtime experiment to evaluate the performance of the physical attacks in a dynamic environment. We instruct the vehicle to drive from 7 meters away to the target object at a constant speed and keep monitoring the predicted confidence scores in this process. Figure 5.1 shows the model prediction results. We set a confidence threshold of 0.25 to identify an object. For the benign object (blue line), we observe that the vehicle is able to detect it from a certain distance and the confidence score remains stable afterward. In contrast, for the adversarial objects created from [64, 66], the confidence scores fluctuate dramatically and the vehicle cannot have stable detection results.

5.3.5 Key Idea behind BatAV

Such imperfection of physical attacks provides a new opportunity for mitigating them. In this chapter, we exploit the inconsistency in spatiotemporal motion features of vehicles to detect PAEs. Specifically, motion characteristics include a vehicle's global position, speed, acceleration, and heading to describe its behavior in space and time. When a vehicle is in motion, its spatiotemporal feature is continuously changing due to continuous variation of actuator behavior. When the vehicle's movement is spatialtemporal consistent, this signifies that the vehicle is currently doing a smooth and regular motion variation. On the contrary, when the vehicle's spatialtemporal feature pattern exhibits sudden and irregular change, its motion state tends to be spatiotemporal inconsistent.

In a standard ADS platform, the perception results will be transmitted to the subsequent planning and control modules for real-time on-road driving, so the spatiotemporal inconsistencies introduced by PAEs can directly affect the behaviors of the vehicle, including its position, heading, velocity, and acceleration, which are continuously updated while the vehicle is in motion. Therefore, by monitoring these kinetic variables, it is feasible to detect any potential anomalies that may arise as a result of PAEs.

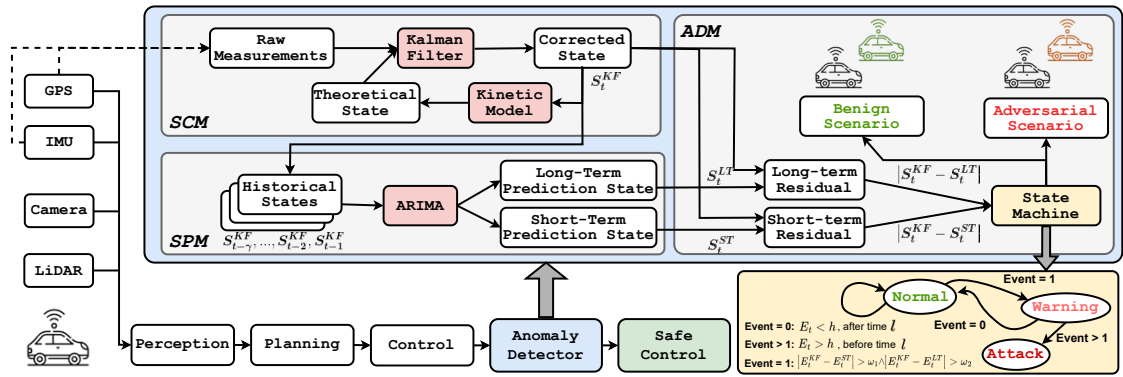


FIGURE 5.5: Overview of BatAV.

It is worth noting that an alternative approach is to directly check the inconsistency of the perception results to identify the anomalies [180–184]. However, we do not consider this strategy for the following reasons. (1) Different categories of objects exhibit unique features in different perception models. If we focus on checking consistency properties in the perception results, we need to build separate detection models for different combinations of object categories and models, which is infeasible for complex traffic scenarios. (2) The motion state and physical properties of a vehicle are more consistent and predictable since the throttle and braking levels defined by the control module under different scenarios are deterministic. To measure the vehicle’s internal state, IMU and GPS data are also more robust and reliable against extreme environmental conditions compared with cameras and LiDARs. As long as the control flow in the ADS is secure, the vehicle’s reactions in benign environments can be uniformly modeled with higher precision.

5.4 VisionGuard

5.4.1 Overview

VisionGuard is a plug-and-play framework to protect common ADSs from physical adversarial attacks. It is designed as a standalone module to conduct safety checking at each decision-making stage of the ADS operation pipeline, without the necessity of modifying existing functional modules or interfering with their intermediate results. Figure 5.5 presents the overview of VisionGuard, which mainly

contains three modules: State Correction Module (SCM), State Prediction Module (SPM), and Attack Detection Module (ADM).

Specifically, **(1)** SCM collects the GPS and IMU data to obtain the vehicle’s current state. However, the state information can contain errors or inconsistencies due to various dynamic effects caused by the environment and power train. To address this issue, SCM applies Kalman Filter (KF) [185] to get the corrected driving state S_t^{KF} at time t . The state serves two main purposes. First, it is the basis for consistency checking with the predicted driving states in ADM. Second, it is stored in SPM to predict the next state of the vehicle. **(2)** SPM is designed for estimating the driving states. It leverages Autoregressive Integrated Moving Average (ARIMA) to build a kinetic model, which utilizes a set of stored historical states collected from SCM, to produce two types of estimated states: a long-term predicted state S_t^{LT} and a short-term predicted state S_t^{ST} at time t . **(3)** Subsequently, the corrected state S_t^{KF} , as well as the estimated states S_t^{LT} and S_t^{ST} are forwarded to ADM for consistency checking. ADM implements a safety state machine to regulate vehicle operations. It performs calculations to determine two residual values $\|S_t^{KF} - S_t^{ST}\|$ and $\|S_t^{KF} - S_t^{LT}\|$. If both values exceed their respective thresholds (w_1 , w_2), ADM initiates the safety state to “*Warning*” for a predefined time interval l . Concurrently, it begins tracking the Accumulated State Prediction Residual (AR) of the state prediction residuals. If this value exceeds a predefined threshold (h) in l , it will set the safety state to “*Attack*” to signalize the occurrence of physical attacks and make a safe operation, e.g., point brake. Otherwise, it will go back to the “*Normal*” state.

5.4.2 State Correction Module (SCM)

The motion states of the vehicle, e.g., position, heading, velocity, and altitude, are provided by onboard sensors like GPS and IMU. However, raw data from these sensors are noisy and collected at varied rates, making them unsuitable for anomaly detection without appropriate calibration. To bridge this gap, SCM applies Kalman Filter (KF) to obtain a corrected state S_t^{KF} . KF is a popular method for achieving corrected state estimation in robotic vehicles by combining outputs from diverse sensors. By utilizing KF, we can approximate the motion state of the vehicle within a small time interval and provide a reasonable estimation of the vehicle’s behavior.

Therefore, We provide the certified motion state of the vehicle from a kinematic perspective.

Formally, the vehicle's motion state s_t at time t includes its heading position $\vec{p} = \overrightarrow{(x, y, \theta)}$ and velocity \vec{v} , which can be expressed as: $s_t = [\vec{p}, \vec{v}]^T$. For computation efficiency and motion state optimization, SCM assumes that the vehicle is conducting a uniformly accelerated rectilinear motion (constant vehicle throttle force) during each running timestep. Based on this premise, we can obtain a theoretical state prediction $s_{t|t-1}$ of the current timestamp through KF prediction of the previous timestamp $S_{t-1|t-1}^{KF}$, state transition matrix F , control matrix B and control vector u_t from vehicle's current acceleration a_t :

$$s_{t|t-1} = \begin{bmatrix} 1 & 0 & \Delta t \cos \theta & 0 \\ 0 & 1 & \Delta t \sin \theta & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ v_{t-1} \\ \theta \end{bmatrix} + \begin{bmatrix} \frac{1}{2} \cos \theta \Delta t^2 \\ \frac{1}{2} \Delta \sin \theta \Delta t^2 \\ \Delta t \\ 0 \end{bmatrix} \times a_t \quad (5.8)$$

$$s_{t|t-1} = F \cdot S_{t-1|t-1}^{KF} + Bu_t \quad (5.9)$$

Every KF prediction step combines raw measurement z_t and theoretical prediction $s_{t|t-1}$ of the current timestamp with Kalman Gain K and observation matrix H , which incorporate the relative influence of the measurement and the prediction in the state estimation process. Therefore, the final prediction output of SCM can be expressed as:

$$S_{t|t}^{KF} = K_t z_t + (I - K_t H) s_{t|t-1} \quad (5.10)$$

By leveraging KF estimation, SCM can output a more accurate prediction of the motion state by considering the presence of noises and errors in the raw measurements from the sensors. Overall, it combines theoretically predicted states based on physical dynamics with the measured states into one corrected motion state variable, thus providing an accurate vehicle state for subsequent consistency checking.

5.4.3 State Prediction Module (SPM)

This module leverages Auto-Regressive Integrated Moving Average (ARIMA) to predict the vehicle's current state S_t^{ARIMA} based on the historical records $\{S_{t-\gamma}^{KF}, \dots, S_{t-2}^{KF}, S_{t-1}^{KF}\}$ collected from SCM. ARIMA is a statistical model widely used for time series prediction. It is designed to analyze historical time-sequence data and generate reliable forecasts of future values. The ARIMA model comprises two key components: (1) the Auto-Regressive (AR) model captures the relationship between the current state S_t^{ARIMA} and its historical states; (2) the Moving Average (MA) model accounts for the accumulated errors in the auto-regressive process. By combining these two components, ARIMA leverages the temporal dependencies and noise patterns present in the historical data to ensure robust and accurate prediction of the vehicle's non-periodic motion behaviors. It is formally defined as follows:

$$S_t^{ARIMA} = \underbrace{\mu + \sum_{i=1}^p \phi_i S_{t-i}^{ARIMA}}_{\text{AR}} + \underbrace{\epsilon_t + \sum_{i=1}^q \theta_i \epsilon_{t-1}}_{\text{MA}} \quad (5.11)$$

where μ is an intercept constant, and ϵ_t is the error term (usually represents the white noise) at timestamp t . The parameters ϕ_i and θ_i are specific to the AR and MA components and optimized during the training stage.

ARIMA offers distinctive advantages over Recurrent Neural Network (RNN)-based methods. First, it is more computationally efficient as it does not require complex deep network architectures, weight optimization, extensive training on large-scale datasets. Second, ARIMA prioritizes gaining insights into recent motion trends, patterns and providing efficient and interpretable results, rather than only focusing on achieving the highest prediction accuracy.

Key hyper-parameters in SPM. Figure 5.6 shows some ARIMA model inference processes. The input size p of the model defines how far back in time the model looks for making new predictions. It is an important parameter, which can affect model's prediction outcomes. Additionally, we introduce three key hyper-parameters, denoted as SPM (α, β, γ) , to offer a more comprehensive prediction of vehicle's motion state.

(1) *SPM inference interval α* : this refers to the length of timesteps between two consecutive ARIMA inferences at runtime. Frequent ARIMA inference enables faster anomaly detection. However, it requires SPM to achieve a high prediction accuracy in benign scenarios to prevent false alarms from being frequently triggered. Besides, it also results in increased computational costs.

(2) *SPM prediction size β* : this refers to the number of motion states for prediction during one ARIMA inference. While the ARIMA model is often used to predict a single value per step (P1 in Figure 5.6), they can also be utilized to forecast a variation tendency with a set of multiple values. SPM defines a trade-off between capturing motion trends and maintaining prediction accuracy. As ARIMA predicts more values, it increasingly relies on previous predictions rather than the true values to make new predictions. Once the number of predicted values reaches the input size p , future predictions will be made entirely based on historical predictions. It is important to carefully balance this trade-off for *VisionGuard* to generalize well across different types of attacks.

(3) *SPM historical interval γ* : this defines how far back in the historical states should be included as inputs for each ARIMA inference. Since the input size p of the model is fixed, we can increase γ to take further historical data into consideration without having to increase p . Due to the inconsistency property of most physical adversarial attacks, including more historical state information indicates a larger deviation of SPM predictions from SCM estimations in adversarial cases, which can benefit attack detection. However, in more complex benign scenarios where the vehicle's behavior is not consistently accelerating, decelerating, or maintaining a steady speed, including outdated historical state information can adversely impact the SPM's ability to make accurate benign predictions. Additionally, storing historical data for an extended period will lead to increased system memory consumption.

Prediction Types. We introduce two state prediction types within SPM:

- *Short-Term State Prediction (ST)*. This employs a longer historical data interval to make predictions for a short-term future. It aims to capture more extensive state variation trends and provide predictions that reflect the recent short-term behavior of the vehicle. (P2 in Figure 5.6)

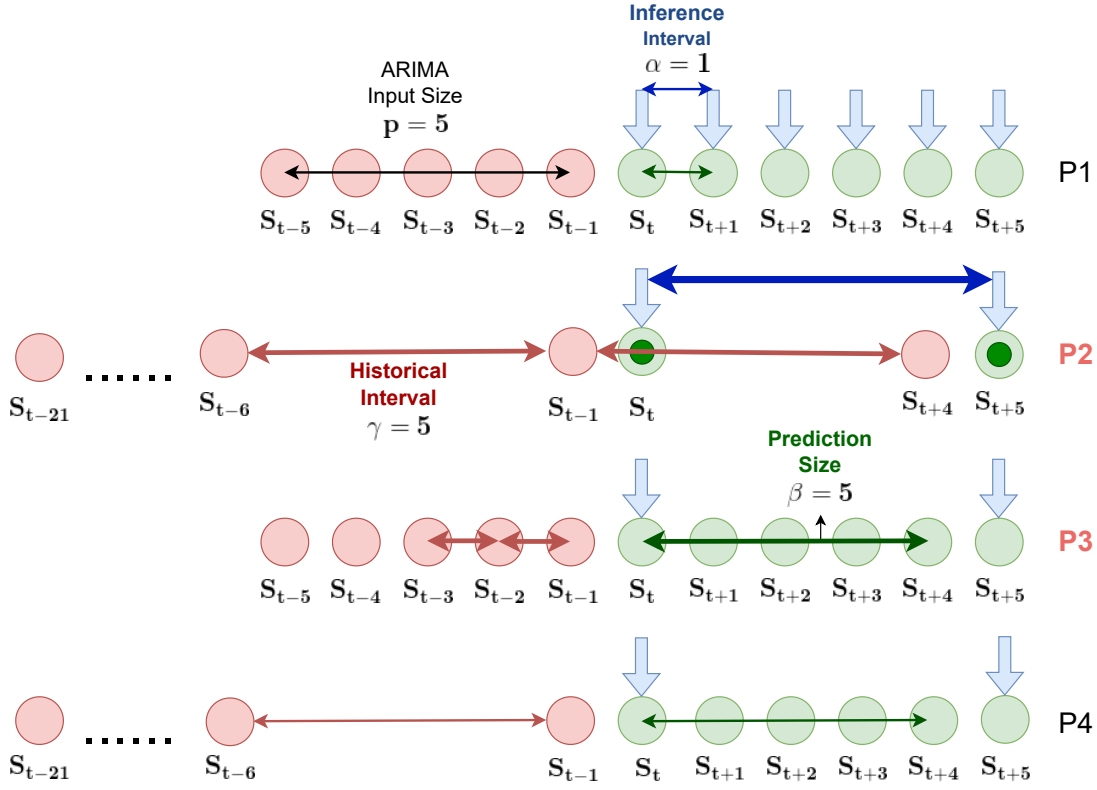


FIGURE 5.6: ARIMA model inference process. Red: SPM inputs; Green: SPM outputs; Blue: ARIMA inference; P2: ST state prediction; P3: LT state prediction.

- *Long-Term State Prediction (LT)*. This utilizes a shorter historical data interval to make predictions for a long-term future. It aims to capture the immediate state variation trend and provide predictions that depict the general long-term behavior of the vehicle. (P3 in Figure 5.6)

The purpose of introducing these two inference types is to strike a balance between accuracy and tendency in the obtained predictions. This allows us to effectively capture both the short-term and long-term motion behaviors of the vehicle. Combining the outputs of LT (S_t^{LT}) and ST (S_t^{ST}) enables VisionGuard to make comprehensive assessments that generalize well in a variety of adversarial scenarios.

5.4.4 Attack Detection Module (ADM)

The corrected state S_t^{KF} from SCM, with the predicted states S_t^{ST} , S_t^{LT} from SPM, are fed into the ADM module in real-time. Algorithm 2 shows how ADM performs

consistency checking. We introduce three important variables in ADM: Long-Term State Prediction Residual (LTR), Short-Term State Prediction Residual (STR), and Accumulated State Prediction Residual (AR). LTR and STR are utilized to quantify the difference between the real and predicted states of the vehicle, which are defined as $\|S_t^{KF} - S_t^{LT}\|$ and $\|S_t^{KF} - S_t^{ST}\|$, respectively. AR is defined as the accumulated value of LTR and STR in the subsequent timestamps, which is expressed as:

$$AR = \sum_{t=1}^n (\|S_t^{KF} - S_t^{LT}\| + \|S_t^{KF} - S_t^{ST}\|) \quad (5.12)$$

We set specific thresholds (w_1, w_2, h) as indicators for detecting deviations and anomalies in the vehicle's motion state. When LTR and STR simultaneously surpass their respective thresholds w_1 and w_2 , ADM will initiate a *Warning* state and be alert to potential threats. It is important to note that the occurrence of a sudden change in LTR and STR does not necessarily indicate the presence of an attack. In benign scenarios, various sudden state inconsistencies frequently occur due to factors such as accelerating from a static velocity, braking for red lights or pedestrians, and other normal driving maneuvers. Therefore, rather than raising an *Attack* state directly upon the initial exceedance of these thresholds, ADM presents a *Warning* state for a specific time interval l . During the *Warning* mode, ADM will continuously monitor the value of AR to determine if it has exceeded h . If yes, ADM will trigger an *Attack* state to signalize the occurrence of an attack. If no further exceedance is observed within l , ADM will switch the vehicle from the *Warning* state to the *Normal* state, and reset AR to 0. These precautionary measures ensure that ADM gains a comprehensive understanding of the cumulative deviations in motion states over time before raising an alarm. ADM relies on the values of hyper-parameters w_1, w_2, h , and l to make accurate decisions. Properly defining these hyper-parameters is crucial to effectively counter false positives while capturing true positives in time. We will discuss this in Section 5.7.2.

5.5 Simulation Evaluation

We validate the effectiveness of each module in *VisionGuard* (Section 5.5.2 – 5.5.4), followed by the end-to-end AD simulator evaluation (Section 5.5.5).

Algorithm 2 Attack Detection

Input: ARIMA inference parameters p, α, β, γ ; State vectors received at frame i from raw sensor measurements: S_i^z ; State vector predictions from Kalman Filter: S_{i+1}^{KF} , Sort-term: S_{i+1}^{ST} and Long-term: S_{i+1}^{LT} ; Accumulated residual at frame $i + 1$: AR_{i+1} ; Alarm thresholds for Short-term prediction: w_1 , Long-term prediction: w_2 and Accumulated residual: h ; Caution interval: l ; State prediction functions from Kalman Filter: $KF(\cdot)$ and ARIMA: $ARIMA(\cdot)$

Output: *Normal* or *Attack*

```

1:  $AR \leftarrow 0$ , Normal
2: while running time steps  $t$  exceeds  $\alpha * \gamma$  do
3:    $S_{t+1}^{KF} \leftarrow KF(S_t^z, S_t^{KF})$ 
4:   if  $t \% a = 0$  then
5:      $S_{t+1}^{ST} \leftarrow ARIMA(S_{t-(p-1)*\gamma}^{KF}, S_{t-(p-2)*\gamma}^{KF}, \dots, S_t^{KF})$ 
6:      $S_{t+1, t+2, \dots, t+\beta}^{LT} \leftarrow ARIMA(S_{t-(p-1)}^{KF}, S_{t-(p-2)}^{KF}, \dots, S_t^{LT})$ 
7:   end if
8:   if  $\|S_{t+1}^{KF} - S_{t+1}^{ST}\| > w_1$  and  $\|S_{t+1}^{KF} - S_{t+1}^{LT}\| > w_2$  then
9:     Warning mode for  $l$  steps
10:  end if
11: end while
12:
13: // Warning mode
14: while in Warning mode do
15:    $AR_{t+1} \leftarrow AR_t + \|S_{t+1}^{KF} - S_{t+1}^{ST}\| + \|S_{t+1}^{KF} - S_{t+1}^{LT}\|$ 
16:   if  $AR_{t+1} > h$  then
17:     Raise Attack
18:     Break
19:   end if
20: end while
21:  $AR \leftarrow 0$ , Normal

```

5.5.1 Experiment Setup

Experiment Setup. We use LGSVL and CARLA to conduct experiments. Our evaluation encompasses various environmental factors, including diverse weather conditions, traffic scenarios, road types, etc. The frequencies of IMU and GPS equipped on driving vehicle are 100Hz and 12.5Hz, respectively. For each scenario, both benign and adversarial case are tested independently. At runtime, raw sensory data are processed by different DNN models in the perception module. The perception results are subsequently published to the simulator API to generate control commands. We mainly use Yolo-v3 as the victim model and hiding attack based on the SLAP technique [66], if not mentioned specifically.

Metrics. We adopt the following metrics to evaluate VisionGuard. (1) Detection Success Rate (DR): this is defined as the ratio of adversarial scenarios that can be successfully detected. (2) False Positive Rate (FPR): this is the portion of benign cases that are incorrectly detected as adversarial cases out of all the benign cases.

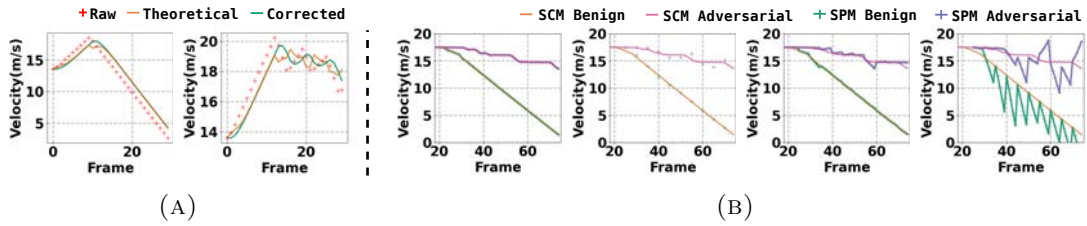


FIGURE 5.7: (a) Benign and adversarial. (b) SPM outputs with different configurations.

(3) Short-Term Excessive Rate (STER) and Long-Term Excessive Rate (LTER): they are calculated as the ratio of frames that exceed the respective thresholds. These metrics provide insights into how stable the attack is during the entire run from short-term and long-term perspectives individually. (4) Accumulated State Prediction Residual (AR): this is calculated by summing up all the values of residuals generated by both types of motion states. It is considered as the key evaluation metric in ADM and provides a comprehensive assessment of physical attacks.

5.5.2 Evaluation of SCM

SCM collects the raw measurements from GPS and IMU at each timestamp, to produce the corrected motion state of the vehicle. Table 5.3 shows the detailed configurations in LGSVL, where the positions and control variables are indicated based on the PythonAPI [186] of LGSVL. Figure 5.7(a) presents the raw measurements, theoretical and corrected states of the vehicle in benign and adversarial scenarios. We have the following observations. First, in the benign case, the vehicle accurately detects a stop sign and exhibits stable deceleration until coming to a stop. However, in the adversarial case, the vehicle fails to maintain a consistent deceleration due to unstable detection outputs. Consequently, the vehicle fails to stop in front of the stop sign, indicating that the attack has been successfully launched. Second, SCM effectively minimizes the errors between the theoretical states and raw measurements. One key advantage of integrating SCM into VisionGuard is the smoothing effect it imparts on the runtime state outputs, which is particularly noticeable in the speed fluctuations of the vehicle. This reduction in variation plays a crucial role in minimizing the potential margin of errors, especially in cases where abrupt measurement errors can potentially mislead the outputs of VisionGuard.

TABLE 5.3: SCM runtime setup.

Map	CubeTown
Stop Sign Position	(12.24, 0, 7.56)
Stop Sign Rotation	(0, 92.86, 0)
Adversarial Example	SLAP[66]
Vehicle Initial Position	(-3.2, 0, -35)
Vehicle Initial Rotation	(0, 180, 0)
Vehicle Initial Speed	14
Speed Limit	20
Runtime Duration (s)	3
Runtime FPS	10
Objectness Threshold	0.25
Throttle Factor	1
Breaking Factor	1

TABLE 5.4: Different inference settings in SPM.

	p	α	β	γ
P1	5	1	1	1
P2	5	5	1	5
P3	5	5	5	1
P4	5	5	5	5

5.5.3 Evaluation of SPM

To train the ARIMA model, we generate a dataset by driving the vehicle in the map and recording the SCM outputs. The vehicle follows a predefined sequence of actions, including gradually accelerating from a stationary position, maintaining a consistent speed for a specific duration, steering in different directions, and uniformly decelerating to a stop. We collect a total of 100 scenarios in these settings.

After the dataset is obtained, we train the ARIMA model and retrieve its corresponding weights. To showcase the impact of the inference interval α , prediction size β , and historical interval γ , we conduct a series of motion state predictions and compare the outputs of SPM and SCM. All experiments are carried out in the same scenario, where the vehicle is driving at a constant speed and encounters

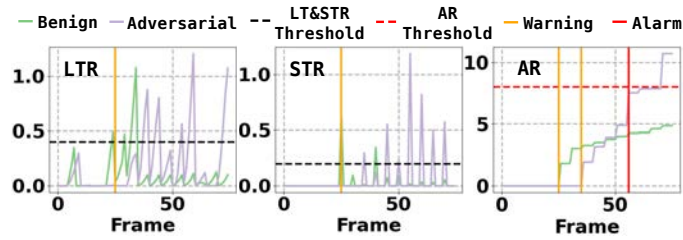


FIGURE 5.8: LTR (left), STR (middle) and AR (right) in ADM (w_1 : 0.2, w_2 : 0.4, h : 8, l : 50).

both a benign and adversarial stop sign. Table 5.4 and Figure 5.6 present four different SPM hyper-parameter configurations (P1, P2, P3, P4) to predict the future velocity of a vehicle, which can be summarised as follows:

- **P1**: using short-term historical data from five previous consecutive frames to make a state prediction at every step.
- **P2**: using long-term historical data by extracting estimations across 25 previous frames with a five-frame interval to make a state prediction at every five steps.
- **P3**: using short-term historical data from five previous consecutive frames to make five consecutive state predictions at every five steps.
- **P4**: using long-term historical data by extracting estimations across 25 previous frames with a five-frame interval to make five consecutive state predictions at every five steps.

Figure 5.7(b) shows the experimental results with the configurations P1-P4. We observe that with P1, the SPM predictions exhibit slight deviations from SCM estimations for both benign and adversarial scenarios, while they both differ significantly with P4. This highlights the crucial trade-off between prediction accuracy and tendency in SPM, as discussed in Section 5.4.3. Both configurations represent extreme cases where the balance between the two factors is heavily skewed, resulting in predictions that are indistinguishable between benign and adversarial scenarios. The primary responsibility of SPM is to enable the vehicle to capture the underlying benign motion trend while minimizing the loss of prediction accuracy. In P2 (short-term state prediction) and P3 (long-term state prediction), SPM achieves a deeper understanding of this concept by either expanding its historical knowledge or projecting further in the future.

5.5.4 Evaluation of ADM

The corrected states generated by SCM and the prediction states generated by SPM are fed into ADM. Figure 5.8 illustrates the mechanism of ADM at runtime. In a benign scenario, we observe that the vehicle encounters a sudden increase in both STR and LTR, surpassing their respective thresholds in the 25th frame. This indicates that the current prediction results from SPM exhibit a relatively strong deviation from SCM’s corrected estimation, which is caused by commands sent from the perception module to the control module in response to an event that triggers a variation in the detection results, such as the appearance of a stop sign in front. However, it is important to note that this “stop sign” could potentially be an adversarial object crafted by an attacker to mislead the perception models. Hence, ADM promptly triggers a *Warning* state (Event = 1 in the state machine), starting to record and monitor the value of AR. After a short time interval ($l = 50$ steps), if there is no significant further increase in AR, the warning state will be disabled, and AR is reset to zero (Event = 0 in the state machine). The vehicle then resumes safe driving until a new *Warning* is triggered.

For the adversarial scenario, the *Warning* state is also triggered but relatively more slowly (in the 35th frame). Different from the benign scenario, the values of S^{ST} and S^{LT} continue to experience significant fluctuations in the warning state. As a result, the *Attack* state is triggered in the 65th frame as AR also surpasses the threshold, forcing the vehicle to disregard any subsequent commands and initiate safe control measures immediately, e.g., gradual deceleration to a stop.

5.5.5 End-to-end Evaluation of VisionGuard

We comprehensively evaluate *VisionGuard* by using 9 different state-of-the-art PAEs with various target objects, models, perception functions, and attack goals, as well as 9 different scenarios. This allows us to collect a total number of 36000 frames from 36 runs to fully validate *BatAV*.

Defense Effectiveness. For each attack, we run 10 times in LGSVL. We observe all the AEs can be 100% detected in the benign sunny condition. We also consider other scenarios such as rainy or foggy conditions to check the false positives, and the details are given in Section 5.7.2. Table 5.5 reports the DR of different attacks. It

TABLE 5.5: Detection rates (DR, %) of different PAEs ($w_1 : 0.2, w_2 : 0.4, h : 10$).

Attacks	Attack Goal	Target object	Victim Model	STER	LTER	AR	Warning (frame)	Attack (frame)	DR
SLAP	HA	Stop sign	YOLO-v3	0.06	0.16	16.817	Yes(35)	Yes(65)	100%
RP2	MA	Stop sign	Faster R-CNN	0.08	0.17	17.269	Yes(40)	Yes(65)	100%
ShapeShifter	MA	Stop sign	Faster R-CNN	0.12	0.16	15.160	Yes(35)	Yes(65)	100%
ShapeShifter	HA	Stop sign	Faster R-CNN	0.12	0.17	17.143	Yes(25)	Yes(50)	100%
AdvCam	MA	Stop sign	VGG-19	0.06	0.05	11.900	Yes(45)	Yes(50)	100%
Nested-AE	AA	Stop sign	Faster RCNN	0.12	0.24	15.545	Yes(25)	Yes(35)	100%
Adv-MSF	HA	Bench	Apollo-v5	0.1	0.16	14.589	Yes(35)	Yes(60)	100%

is clear that **VisionGuard** has successfully detected all types of adversarial objects in different settings. Video demos can be seen on our website.

Comparison with Baseline Methods. We compare **VisionGuard** with representative open-source defense methods including DetectGuard [27] (Certified defense) and PercepGuard [28] (Vision-based consistency checking). PercepGuard is originally designed for defending car-to-person misclassification attacks, for fair comparison, we use speed limit 50 as the induced label, and yolov3 is finetuned on German Traffic Sign Recognition Benchmark dataset [187]. We record 45 runs with adversarial objects created by SLAP [66] and ShapeShifter [64] in 5 different conditions, 3 different initial speeds, object headings, and maps to see if these defense methods can successfully detect the adversarial patches attached to the stop sign. A full run lasts 10s with 10 fps. Notably, for the two defense methods, once a single or a period of frames during the adversarial run is deemed as being attacked, the run is considered to be detected successfully. In such settings, the effectiveness of **VisionGuard** shown in Table 5.7 is evident, as it successfully detects the adversarial patches in every run. We also observe DetectGuard and PercepGuard also achieve high DRs, among which, DetectGuard cannot achieve 100% because it is designed for localized patch hiding attacks, while the AE generated by SLAP is a universal patch. Moreover, these two methods are designed to address specific attack goals, **VisionGuard** is capable of detecting physical attacks across all different goals. This makes **VisionGuard** a comprehensive solution for attack detection. Furthermore, for detection latency and runtime (per frame), **VisionGuard** outperforms the other methods, achieving faster average detection speeds and less runtime overhead for both types of attacks. Notably, PercepGuard requires 4-5 frames of inference to detect attacks, so its average detection time is around 128-160ms. Table 5.6 gives the detailed runtime overhead of each module in **VisionGuard**. This indicates that **VisionGuard** is not only effective but also efficient, providing timely protection against potential attacks.

TABLE 5.6: Runtime analysis for one detector iteration.

	Mean
Kalman Filtering	0.000012s
ARIMA(short-term)	0.000612s
ARIMA(long-term)	0.000684s
Detection	0.000007s
Total	0.001315s

TABLE 5.7: Comparisons with two baseline defenses.

Method	DR		Detection Latency (s)		Runtime (per frame)
	SLAP	ShapeShifter	SLAP	ShapeShifter	
DetectGuard	71.1%	0	7.22	N/A	117.2ms
PercepGuard	0	100%	N/A	6.73	32ms
BatAV	100%	100%	6.54	5.03	3.1ms



FIGURE 5.9: Different simulation scenes.

5.5.5.1 Defense Robustness

We comprehensively evaluate the robustness of **VisionGuard** against 8 different scenarios: (1) Heading: changing the heading direction of the adversarial object by 30° towards the vehicle. (2) Contextual: adding a person and another vehicle close to the adversarial object. (3) Twilight: removing sunlight and adding lamp light. (4) Rainy: adding raindrops. (5) Foggy: adding foggy effects. (6) Road Condition: adding damage and wetness effects on the ground. (7) Initial Velocity: changing the initial velocity of the vehicle from 0 to 5. (8) Map: Adding “Sanfransisco”, “SingleLaneRoad” from the LGSVL asset library. Figure 5.9 visualizes some demo settings.

We run each evaluation 10 times. Table 5.8 shows the average detection rates over 90 evaluation scenarios. We observe that **VisionGuard** can achieve 100% detection

TABLE 5.8: Detection rates (DR, %) in 9 different scenes.

Scene	SLAP		RP2		ShapeS		DR
	AR	Alarm (frame)	AR	Alarm (frame)	AR	Alarm (frame)	
Default	16.82	Yes(65)	17.27	Yes(65)	17.14	Yes(50)	100%
Heading	13.23	Yes(55)	14.02	Yes(55)	12.85	Yes(60)	100%
Contextual	19.67	Yes(60)	12.46	Yes(60)	15.49	Yes(60)	100%
Twilight	18.10	Yes(65)	19.10	Yes(65)	18.94	Yes(60)	100%
Rainy	18.96	Yes(65)	18.27	Yes(60)	20.32	Yes(60)	100%
Foggy	17.42	Yes(65)	18.03	Yes(60)	19.65	Yes(60)	100%
Road	15.20	Yes(55)	16.58	Yes(60)	14.08	Yes(55)	100%
Velocity	15.40	Yes(35)	13.83	Yes(40)	16.44	Yes(40)	100%
Map	16.07	Yes(55)	15.36	Yes(55)	13.18	Yes(55)	100%

rates on average, which shows a high robustness of **VisionGuard** against different attack approaches under different scenarios. While our method demonstrates strong defense performance overall, we also observe instances where it produces false positives in certain scenarios. False positives occur when **VisionGuard** incorrectly identifies benign inputs as adversarial objects. This can potentially lead to unnecessary interventions or disruptions in normal ADS operations. We will give details in Section 5.7.2.

5.6 Outdoor Road Driving Test

To further demonstrate the effectiveness and practicality of **VisionGuard**, we also perform evaluations in the real world with a physical autonomous vehicle and adversarial object.

Setup. The experiments are carried out on a rarely-used road using an Unmanned Ground Vehicle (UGV), as depicted in Figure 5.10 in Appendix. The UGV is equipped with an Intel RealSense D435i front-facing camera (1920×1080 resolution) and a Bosch BMI055 6-axis IMU. Unlike the simulation experiments where the camera is set at 10 fps by default, the physical camera on the UGV is 30 fps, thus giving a more accurate indication of how effective **VisionGuard** can be at different frame rates in the real world. In our experiments, both benign and adversarial stop signs (HA, generated by [66]) are color-printed and placed at 1.8m height. The UGV is initially placed 10 meters away from the target sign and drives straight forward. We set the initial speed as 5m/s (18km/h) at the beginning of

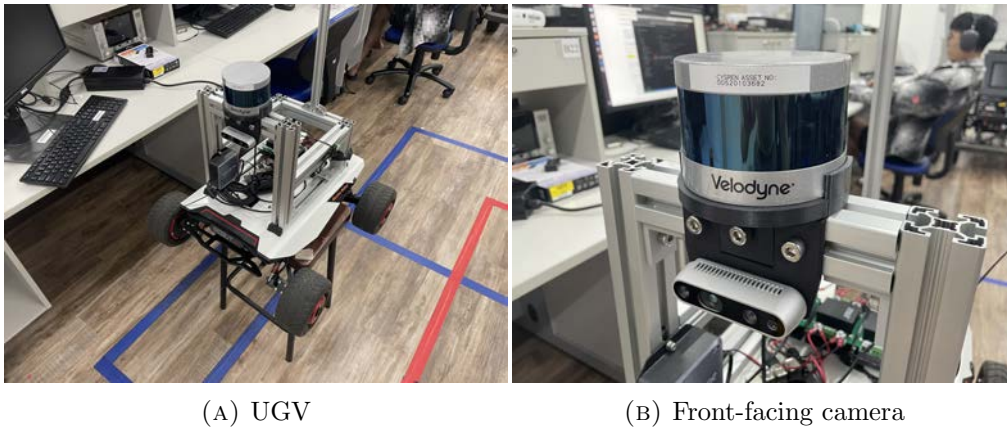


FIGURE 5.10: Our physical UGV with an Intel RealSense D435i camera and Velodyne-16 LiDAR.



FIGURE 5.11: Physical experiment results. *Normal*, *Warning*, *Warning mode*, *Attack* happen at 0-1.1, 1.1, 1.1-1.2, 1.3 (s), respectively.

TABLE 5.9: Detection rates (DR, %) and False Positive Rates (FPR, %) with different horizontal distances in real-world.

	STER	LTER	AR	<i>Attack</i>	DR	FPR
Benign/left-1.5m	0.18	0.95	1.53	No	N/A	0%
Benign/right-1.5m	0.31	1.06	2.74	Yes(40)	N/A	3.3%
Adversarial/left-1.5m	0.88	1.45	7.13	Yes(40)	100%	N/A
Adversarial/right-1.5m	0.84	1.61	8.29	Yes(40)	100%	N/A

each recording. Such a setup is significant since it models the real traffic environment. We repeat the experiment for 30 runs with each run consisting of two benign and adversarial signs heading towards different angles.

Results. Table 5.9 shows the detection results in the physical environment. We observe that the most benign stop sign does not trigger the *Attack* signal. However, all the adversarial patches with different angles can achieve a 100% detection rate for all 30 runs. Figure 5.11 visualizes the detection results, where the stop sign is placed at right side road. As the UGV moves towards the adversarial object, the confidence scores for the “stop sign” present distinct inconsistency at varying distances and angles between them. Similarly, the experimental results in the physical world show identical performance as the simulation ones, which provides further evidence for the effectiveness of VisionGuard.

5.7 Discussion and Limitation

5.7.1 Adaptive Attack

5.7.1.1 Mechanism-aware adaptive attack.

We consider a more sophisticated attacker who may try to improve his attack techniques in response to our defense measures. The most potential directions is improving the attack robustness. The attacker may invest efforts into improving the robustness of his attack methods to bypass `VisionGuard`. He may employ advanced algorithms and optimization techniques to generate more powerful adversarial objects that are capable of consistently evading our defense mechanism. Although this is an attractive goal, to the best of our knowledge, **there is still no satisfactory way to generate absolutely robust PAEs that can overcome the physical constraints and maintain the attack effectiveness consistently.**

5.7.1.2 Parameter-aware adaptive attack.

`VisionGuard` includes some hyper-parameters (e.g., AR, LTR, caution interval l), whose values are determined empirically. We investigate the impact of these hyper-parameters for adaptive attack analysis. **AR and LTR thresholds.** Intuitively, if the attacker manages to constantly bypass the setting of h , the *Attack* state would not be triggered during the caution interval l . On the other hand, if the physical attack is robust enough to stay under-covered below w_2 , the *Warning* state would not be triggered at all from the start. `VisionGuard` will not work as a consequence. To thoroughly understand the impact of h and w_2 , we conduct extensive experiments to evaluate their performance under three types of adversarial objects and scenarios following the same setups in Tables 5.5 and 5.8. For each evaluation, we record the detection False Positives (FP) and True Positives (TP) over a total of 40

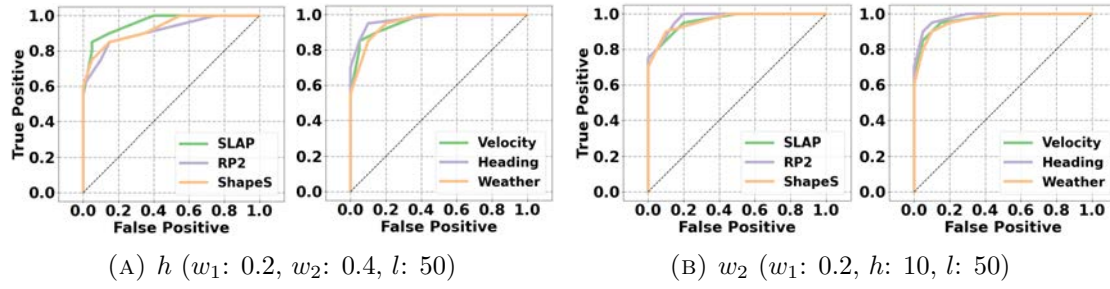
FIGURE 5.12: ROC of AR threshold h (a) and w_2 (b).

TABLE 5.10: Optimal value of LTR threshold.

		SLAP	RP2	ShapeShifter	Velocity	Heading	Weather
LTR	LTR	0.44	0.44	0.44	0.44	0.39	0.44
	AUC	0.85	0.89	0.88	0.93	0.95	0.93
AR	AR	0.68	0.73	0.73	0.73	0.68	0.73
	AUC	0.89	0.86	0.88	0.90	0.93	0.90

TABLE 5.11: Caution interval in urban and highway environments.

v_0 (m/s)	l	Benign	Adversarial	DR (%)	FPR (%)
45	80	20	19	95	0
	100	20	20	100	0
	120	18	20	100	10
90	30	20	19	95	0
	50	20	20	100	0
	70	19	20	100	5

runs (20 benign and 20 adversarial cases). To evaluate AR, we fix (w_1, w_2, l) and slide h from 0 to 15 with the interval of 0.05. Similarly for LTR, we fix (w_1, h, l) and slide w_2 from 0 to 1 with the interval of 0.05.

We determine the optimal values of h and w_2 closest to (0,1) for each type of PAEs and scenario. Figures 5.12 gives the results. It is clearly that the differences between these values across different attack types and scenarios are quite small. This indicates that (1) VisionGuard exhibits high stability and versatility, eliminating the need for employing different thresholds for different attack types and scenarios. (2) In adversarial cases, a slight decrease in the thresholds has a negligible impact on FPs while significantly improving TPs. Tables 5.10 shows the optimal values of LTR and AR with the corresponding AUC, respectively.

Caution interval l . We conduct the same experiments on SLAP [66] with two different vehicle initial speeds: 45 km/h and 90 km/h , corresponding to the speed limits in urban and highway environments. Table 5.11 shows the results. We can observe a negative correlation between l and v_0 in our test range. When the vehicle



FIGURE 5.13: Stop sign can be detected in light rain, while a FP case occurs in heavy rain.



FIGURE 5.14: Stop sign can be detected in light fog, while a FP case occurs in dense fog.

is running at a lower speed with a more stable motion state variance, it is necessary to increase the size l to capture more information about the consistency pattern in order to reduce the chances of missing detection.

Limitation. In this chapter, we have not added real-time manipulated PAEs such as controllable projections and signals into our threat model. For instance, the attacker may use a drone to project an adversarial patch onto a billboard or other surfaces visible to the ADS’s camera. However, these attacks can be mitigated to some extent. Suppose that the attacker have knowledge of the value of the caution interval l , he would try to project the AE again after the “Caution mode”



FIGURE 5.15: A FP case occurs due to very bumpy road.

is deactivated.

5.7.2 Evaluation Under Normal Cases

Prior experiments prove the robustness of *VisionGuard* in different environments. However, during our simulation and physical tests, we also noticed that *VisionGuard* is sensitive to corner cases, resulting in false positives. (indeed, we find not only in their videos but also in our experiments, that DNN models show high robustness of object detection and can easily keep consistent over time).

Simulation. We find adverse weather conditions such as heavy rain and fog (Figures 5.13 and 5.14) can cause the perception module to make mistakes and lead *VisionGuard* to falsely trigger the *Attack* alarm. Regarding this case, we demonstrate that FPs primarily occur under extreme weather conditions, while *VisionGuard* is still effective in normal rainy or foggy scenarios, as illustrated in Table 5.8. Besides, to mitigate the effect of such corner cases, numerous fog-removal and rain-removal approaches [188–193] can be used to enhance the perception module and reduce FPs.

Physical. To fully understand *VisionGuard*'s FP performance in the real world, we collected 30 videos of different scenarios and tested them using Yolov3 and Yolov5, Faster-rcnn. All the videos are given on our website. We observe that these models only fail one case, i.e., image blur due to bumpy roads. Blurry camera images can cause the inaccurate detection of obstacles or traffic signs (Figure 5.15). Regarding this case, for minor bumps, the integrated IMU within the camera typically performs motion compensation, thereby minimizing vibrations and reducing image blurring. It is important to note that IMU compensation fails only under highly turbulent conditions, as shown in Figure 5.15, resulting in the false triggering of the *Attack* alarm. Based on the above analysis, our approach shows strong

effectiveness and robustness, as well as very low false positives. ADS practitioners should develop more robust perception algorithms in different environments to eliminate the false detection rate of *VisionGuard*.

5.8 Conclusion

We present *VisionGuard*, a practical defense framework that can effectively detect PAEss by exploiting the spatiotemporal inconsistency in vehicle’s kinetic states. *VisionGuard* is agnostic to the attack goals, target objects, models, sensors, and contextual information. We comprehensively evaluate 9 state-of-the-art PAEs in both simulation and real-world scenarios. Our results show that *VisionGuard* achieves high detection rates across diverse settings with high robustness.

Chapter 6

A Lifelong Anomaly Detection Framework against Physical Adversarial Attacks to Autonomous Driving

Autonomous Vehicles (AVs) are closely connected in the Cooperative Intelligent Transportation System (C-ITS). They are equipped with various sensors and controlled by Autonomous Driving Systems (ADSs) to provide high-level autonomy. The vehicles exchange different types of real-time data with each other, which can help reduce traffic accidents and congestion, and improve the efficiency of transportation systems. However, when interacting with the environment, AVs suffer from a broad attack surface, and the sensory data are susceptible to anomalies caused by faults, sensor malfunctions, or attacks, which may jeopardize traffic safety and result in serious accidents. In this chapter, we propose **ADS-Lead**, an efficient collaborative anomaly detection methodology to protect the lane-following mechanism of ADSs. **ADS-Lead** is equipped with a novel transformer-based one-class classification model to identify time series anomalies (GPS spoofing threat) and adversarial image examples (traffic sign and lane recognition attacks). Besides, AVs inside the C-ITS form a cognitive network, enabling us to apply the federated learning technology to our anomaly detection method, where the vehicles in the C-ITS jointly update the detection model with higher model generalization and data privacy. Experiments on Baidu *Apollo* and two public data sets (GTSRB

and Tumsimple) indicate that our method can not only detect sensor anomalies effectively and efficiently but also outperform state-of-the-art anomaly detection methods.

6.1 Introduction

Over the past years, Autonomous Vehicles (AVs) are experiencing rapid development. Benefiting from the advances in the technologies of computing, mechanics and deep learning [194], modern vehicles become more automated and intelligent. Many IT and motor companies are attracted to devote themselves to this promising domain e.g., Baidu *Apollo*¹, Google *Waymo*². Hence, in the near future, we expect to see various types of AVs will be fully commercialized to significantly impact different aspects of our life.

The essential component of an AV is the Autonomous Driving System (ADS). It receives information from the external environment and then makes driving decisions. A standard ADS has a pipeline consisting of multiple modules for different functionalities, e.g., perception, planning, control. They cooperate to achieve end-to-end automation. Unfortunately, the high complexity of the ADS inevitably brings a broad attack surface. For example, an adversary can launch GPS spoofing attacks to mislead AVs to navigate to a dangerous position [173]. The attack cost is only \$200 for a low-end “GPS spoofing” device. By adding malicious patches [62, 81] on the road or traffic signs, an adversary can make ADSs perceive the environment mistakenly and make wrong decisions. Attacks on Lidar can make ADSs ignore the surrounding obstacles, resulting in collisions [72].

It is important to guarantee the robustness of the ADS against those cyber attacks and faults. A practical solution is anomaly detection, which monitors the runtime behaviors and states of the ADS, as well as the received environmental information, to identify any suspicious events. The emergence of the Cooperative Intelligent Transport System (C-ITS) provides new opportunities for reliable and effective anomaly detection. In a C-ITS, vehicles are connected with each other, the infrastructures, passengers, and cloud. They naturally form a cognitive network, and frequently exchange runtime data for better traffic and mobility management

¹<https://github.com/lgsvl/apollo-5.0>

²<https://waymo.com/>

[195–197]. As a result, it is also promising that vehicles in the C-ITS can perform anomaly detection collaboratively to mitigate any attacks against the ADS. This can increase the detection efficiency and accuracy.

Motivated by this feature, this chapter proposes **ADS-Lead**, a novel methodology for protecting Autonomous Driving Systems with Lifelong anomaly detection. We consider the lane following mechanism, which is the most common and fundamental scenario in not only ADSs but also state-of-the-art Advanced Driver-Assistance Systems (ADASs) and Lane Keeping Assist Systems (LKASs). Different types of security threats have been disclosed in the lane following scenario, i.e., localization attacks, lane detection attacks, and traffic sign recognition attacks. They can lead to severe consequences and damages, such as car crashes, human injuries or even deaths. Hence it is important for vehicles to be immune to them for secure and safe driving. Although prior works proposed some solutions to defeat sensor attacks for AVs [72, 198, 199], they only focus on one specific kind of threats. It is challenging to design a unified and comprehensive method to cover different attack vectors, as they have distinct behaviors and techniques.

ADS-Lead introduces two contributions to achieve efficient and unified protection. The first one is a novel one-class classification model, dubbed T-GP (Transformer with Gradient Penalty). It is capable of analyzing and identifying time series anomalies (localization attacks) and adversarial images (i.e., lane detection attacks and traffic sign recognition attacks) in the lane following scenario. This model needs to be trained offline only from normal data, and then deployed in the ADS as an online detector to inspect different sources of sensory data, and discover the suspicious input. T-GP is built from an one-layer transformer encoder. It introduces a novel loss function, which combines the Negative Log Likelihood (NLL) with the Gradient Penalty (GP). The integration of these techniques gives very high accuracy for anomaly detection of various attacks.

The second contribution is the adoption of federated learning and lifelong learning for anomaly detection in the C-ITS. Each vehicle in our system not only performs the online monitoring and detection, but also continuously collects live data to update the one-class model. They train the model locally, and then send the model gradient to a parameter server hosted in the cloud. This parameter server aggregates the gradients from different vehicles at different zones of the C-ITS, and releases the final model back to them for update. The collaboration for anomaly

detection based on federated learning can significantly improve the model generalization and performance while preserving the vehicle’s privacy.

We implement a prototype of our methodology in a federated learning system. We apply our proposed model on the datasets from the real world, and collected from simulations to comprehensively evaluate its effectiveness. Specifically, for localization attacks, since there are no public datasets available, we collect the Inertial Measurement Unit (IMU) data from Baidu *Apollo*, running on the San Francisco map with the *LGSVL* simulator³. We follow [173] to implement GPS attacks, which can cause severe fluctuation of the IMU data generated by the Multi-Sensor Fusion (MSF) component in *Apollo*. For lane attacks, we adopt the Tumsimple dataset, and implement the attack method in [82] to generate fixed and variable adversarial patches. For traffic sign attacks, we use the GTSRB dataset. We reproduce the boundary attacks [200] and poster attacks [62] to generate adversarial data. We compare T-GP with existing one-class classification methods. Evaluation results show that T-GP outperforms other methods in detection of these attacks.

In summary, the main contributions of our work are:

- We propose **ADS-Lead**, a novel collaborative anomaly detection approach to protect the lane following scenario of the ADS efficiently and comprehensively.
- We introduce T-GP, a novel one-class classification model based on the transformer for anomaly detection. It can effectively detect both time series anomalies and adversarial images.
- We are the first to adopt federated learning and lifelong learning to realize collaborative anomaly detection on AVs, which can enhance the model generalization and performance without compromising vehicles’ privacy.
- We conduct extensive evaluations of our method over simulation and real-world datasets. We demonstrate T-GP outperforms existing state-of-the-art models on the detection of localization, traffic sign and lane recognition attacks. And ADS-Lead with T-GP can be made effectiveness and practical for AVs anomaly detection.

³<https://github.com/lgsvl/simulator>

6.2 Related Works

6.2.1 Detection of GPS Spoofing Attacks against AVs.

Although prior works made some attempts to detect GPS attacks against AVs [201–205], how to effectively mitigate such threat is still a long-standing problem. The MSF algorithms were regarded as the most effective defense method in ADSs [206]. Unfortunately, Shen *et al.* [173] found a vulnerability in the design of MSF-based localization and successfully implemented a sophisticated attack to invalidate the protection. Researchers also studied spoofing detection by cross-checking GPS readings and IMUs data [207]. However, IMU data suffer from the accumulation of drift errors such that they provide reliable protection against spoofing attacks if an adversary causes gradual deviation of the victim vehicles from their actual positions [208]. Compared with these prior works, we only use the instantaneous changes of the IMU data to detect whether the vehicle is being attacked, which achieves very high detection accuracy.

6.2.2 Detection of Adversarial Examples.

Some works introduced methods to detect adversarial examples, especially in the CV domain. Qiu *et al.* [209] illustrated adversarial attacks against network intrusion detection in IoT systems. Xu *et al.* [35] proposed a method called feature freezing to detect adversarial examples by reducing color bit depth and spatial smoothing. They set a threshold to judge whether the original input data is benign or malicious. Lee *et al.* [36] designed a method using Gaussian discriminant analysis to obtain the confidence score based on the Mahalanobis distance in the feature space of DNN models. Li *et al.* [210] proposed to detect localized adversarial examples by removing and analyzing critical regions controlled by the adversary. Meng and Chen [211] used detector networks to identify adversarial examples by approximating the manifold of normal examples. Feinman *et al.* [212] investigated the Bayesian uncertainty estimates in dropout neural networks, and conducted density estimation in the subspace of deep features to distinguish normal and adversarial examples. Ma *et al.* [213] used the estimation of Local Intrinsic Dimensionality (LID) to quantify the distance between the target sample and normal

samples. Katzir and Elovici [214] explored the sample behaviors in the activation space of different network layers for adversarial example detection. Li and Qiu *et al.* [215] proposed a novel method for intelligent fault diagnosis by fusing domain adversarial training and maximum mean discrepancy via ensemble learning. Wang *et al.* [216] randomly mutated the model and perturbs the decision boundary, which can possibly alter the prediction of adversarial examples, while maintaining the prediction of normal samples. Tian *et al.* [217] utilized input transformations to process the input samples, to which the adversarial examples are very sensitive.

In the context of autonomous driving, some works designed solutions to detect adversarial images captured by the vehicles. Sun *et al.* [218] developed a supervised defense method based on adversarial training with a novel and stereo-based regularizer to enhance the 3D object detection model. Safavi *et al.* [219] adopted two distinct and efficient DNN architectures to detect, isolate and predict sensor faults. One-class models (e.g. Deep-SVDD [78], HRN [80]) were designed for anomaly detection of adversarial examples, and evaluated on the stop sign attacks. For lane attacks, Sato *et al.* [81] proposed an attack method based on image segmentation and deployed a bounded patch to simulate the road dirt to fool the lane detection algorithms. Following this work, Xu *et al.* [82] designed a CNN-based model with prior knowledge of abnormal data to achieve attack detection. However, these works need prior knowledge of the adversarial samples, or can only be applied to specific attacks but fail to be extended to others. In contrast, our solution proposed in this chapter is unified to cover various types of attacks with different formats of sensory data in the lane following scenario.

6.3 Background and Problem Statement

6.3.1 Overview of ADSs

The main responsibility of an ADS is to recognize the surrounding environment and generate proper motion commands to the vehicle [220] [221]. Hence, a typical ADS usually consists of the following modules: localization, perception, planning and control. The localization module uses the information from different sensors (e.g., GPS, IMU, Lidar) to localize the AV on the map based on the Real Time Kinematic (RTK) method and Multi-Sensor Fusion (MSF) algorithms. The perception module

is an AI-based subsystem, which receives input data of different formats (e.g., image, point cloud) from various sensors and leverages Deep Learning models to identify the surrounding traffic conditions (e.g., the status of traffic light, stop sign and speed limit sign) and obstacles (e.g., object types, the speeds of other vehicles on the road). The planning module performs offline path planning to generate a feasible path from the initial position to the destination based on the map information. It also conducts real-time trajectory planning, which utilizes the results from the localization module and perception module to generate a collision-free trajectory in a short time duration. The control module finally generates low-level commands, such as steering, throttle and brake, to the chassis to track the generated collision-free trajectory.

6.3.2 Security Threats in the Lane Following Scenario

Lane following is the most common scenario during the AV operations, where the vehicle moves along the central lines of lanes. In this scenario, the execution of an ADS highly depends on the accuracy of localization, lane boundary detection and traffic signs. Hence, the following three kinds of attacks were proposed to compromise the execution of ADSs in lane following.

Localization Attack. This attack uses counterfeit GPS signals to interfere with the legitimate ones. Then the ADS cannot localize the AV correctly, resulting in positioning errors. Consequently, the ADS will mislead the vehicle to deviate from the expected lane and even cause serious accidents. Although the MSF algorithms in ADSs are designed to mitigate GPS spoofing, researchers find that they are still vulnerable to the take-over attack [173] where the spoofed GPS signals can dominate the inputs of the MSF process and fool MSF to ignore other inputs. Figure 6.1 illustrates the mechanism of such an attack. A victim vehicle (blue) is moving along the straight lane. The attacker vehicle, following the victim, launches a two-stage GPS spoofing attack. The first stage is vulnerability profiling: the attacker collects and analyzes the behaviors of the victim vehicle and determines the time duration to perform GPS attacks. The second stage is aggressive spoofing: the attacker sends wrong GPS signals to the victim vehicle, whose MSF algorithms compute wrong localization of the AV (the shaded blue one). To make the vehicle

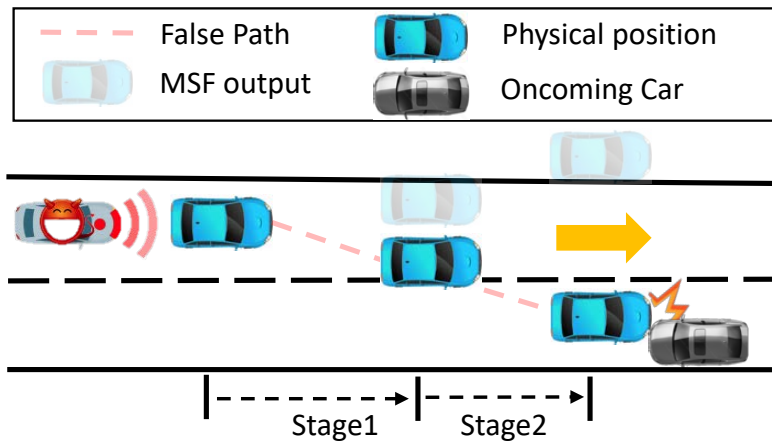


FIGURE 6.1: Illustration of GPS-based localization attacks. Stage 1: Vulnerability profiling; Stage 2: Aggressive spoofing.

stay in the center of the lane, the ADS asks the vehicles to move right, which actually makes it cross the lane and collide with the oncoming vehicle.

Lane Detection Attack. In lane following, an ADS should also need to detect the boundaries of a lane to localize the central line of the lane. Currently, DNNs are the most popular method for lane detection in ADSs. Hence, due to the inherent vulnerability of DNNs, the adversary can also fool the DNN model to cause wrong recognition of lane boundaries, resulting in wrong motion controls to drive along the center of the lane. For example, the adversary can add visual perturbations on the real-world road to make the vehicle deviate the central line and hit a surrounding object [81]. Figure 6.2 shows an attack example [82]: the adversary carefully identifies the optimal location for the patch and then manipulates the subset of pixels of the input images to achieve the goal, i.e., making the lane detection system recognize a wrong lane boundary around the patch.

Traffic Sign Recognition Attack. Recognition of traffic signs can also affect the lane following as an AV must obey the traffic rules described by those signs. Since the ADS leverages CNN models to detect and classify traffic signs, an adversary can leverage the adversarial attack techniques to compromise the model so the ADS will miss or misclassify the traffic signs, and generate wrong motion decisions. Figure 6.3 shows a poster attack on a stop sign [62]. The adversary adopts the Robust Physical Perturbations (RP_2) algorithm [62] to generate visual adversarial perturbations and attach them to the stop sign. Then the perception module in the ADS will identify it as a speed limit sign. Alternatively, the adversary can also

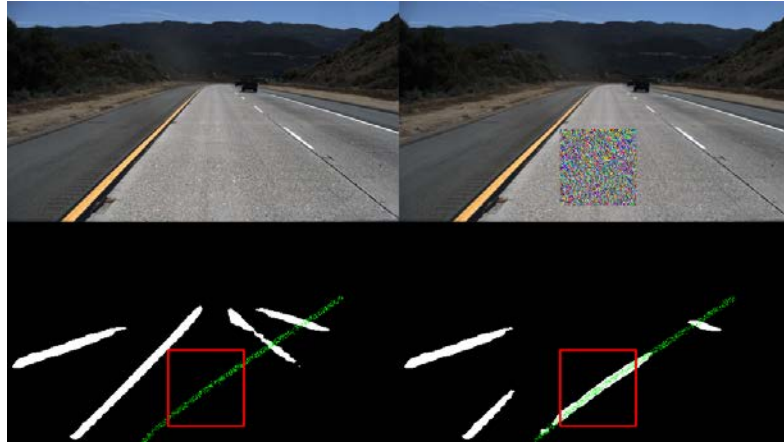


FIGURE 6.2: Lane detection attack. First row: the original input image (left) and the adversarial image with a fixed patch. Second row: the corresponding lane segmentation results from the ADS. Red boxes show the patch localization; induced lanes are marked with green.

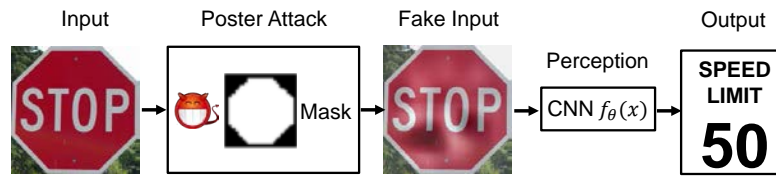


FIGURE 6.3: Poster attacks on the traffic sign.

adopt generative adversarial networks to craft malicious patches to compromise the traffic sign recognition model [222].

6.3.3 Problem Statement

We aim to address the following problem: *How to develop and design a unified and efficient method to detect anomalies of the ADS caused by different kinds of attacks at real time?* We want to have an attack-agnostic approach, i.e., the detector is built from normal data and conditions, but general and effective for various known and unknown threats.

Without loss of generality, we consider the following six attacks from three categories when designing our approach. They represent state-of-the-art security threats against the lane-following mechanism in modern ADSs.

Localization Attacks. We consider the GPS spoofing attack [173] in our method design and evaluation. We assume a malicious vehicle follows the victim AV and

interfere with its GPS signals. The faked signals can fool the MSF algorithms based on the take-over vulnerability. We further assume there are no other obstacles on the road, so the motion change of the victim AV only depends on the localization. We focus on two specific attack goals: (1) an off-road attack tries to lead the victim to hit the curb; (2) a wrong-way attack tries to deviate the victim AV to the opposite pavement.

Traffic Sign Recognition Attacks. We consider two types of attacks in this category: (3) a boundary attack is a decision-based adversarial attack [200]. The adversary does not need any information about the target model in the ADS. He generates the adversarial perturbations on the traffic sign only from the prediction results of the model corresponding to given input images. (4) In a poster attack, the adversary generates malicious posters for traffic signs using a novel Robust Physical Perturbations algorithm [62]. In these two attacks, the adversary is able to physically alter the traffic signs (e.g., adding posters or patches) without changing their visual semantics.

Lane Detection Attacks. We assume the adversary is able to add carefully-crafted patches on the road to deceive the lane detection model in the target ADS. We adopt the Projected Gradient Descent to generate two types of adversarial patches [82]: (5) a fixed-size patch with the size of 100×100 is injected to the images of 512×288 ; (6) a varied-size patch has the size scaled based on the distance from the camera to the destination lane segments.

6.4 ADS-Lead

In this section, we describe **ADS-Lead**, our anomaly detection system for the lane-following scenario.

6.4.1 System Overview

Figure 6.4 shows the overview of our **ADS-Lead** system. The essential component is a powerful anomaly detector deployed in an ADS for attack detection. The workflow contains two stages, as described below.

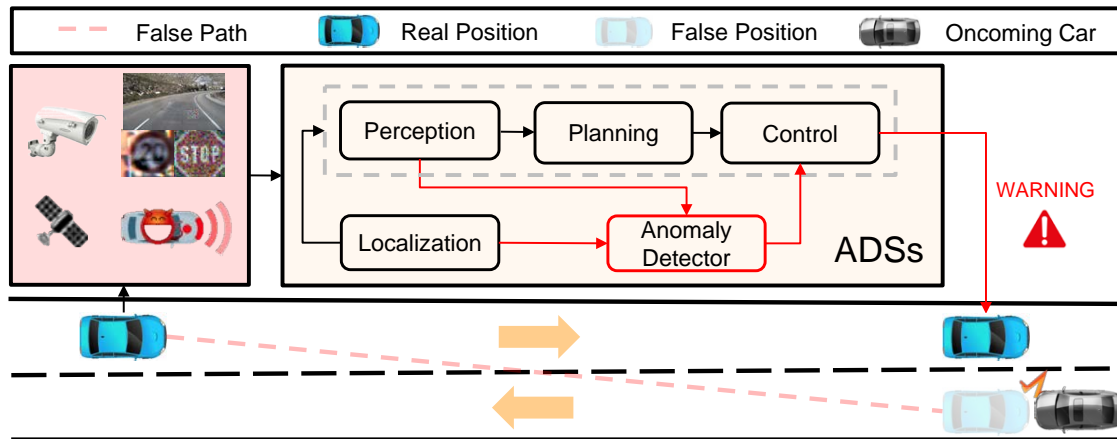


FIGURE 6.4: Overview of our anomaly detection methodology.

The first stage is *offline training*. We train a one-class model to describe the normal behaviors of the ADS. Since we aim to have an attack-agnostic approach, we cannot include any attack-specific data samples when training the detection model, which are hard to obtain and not general for other unknown attacks. Instead, we just collect normal data during the vehicle operations. Note that the normal data can be collected by making the AV run automatically without launching any attack. The collection is not related to any specific road or road condition. Then this model is able to predict whether the incoming data samples belong to the same distribution as the training data (labeled as benign), or deviate a lot from the normal ones (labeled as malicious).

Since our model is designed to be general for different attacks, it should be able to handle different formats of sensory data in the ADS. Specifically, we consider two types of sensory data that are vulnerable to be manipulated by the adversary to compromise the vehicle operation. The first one is IMU messages, which are time series data. The second one is images captured from the cameras. These are used to mitigate attacks against the lane detection and traffic sign recognition. We introduce approaches to preprocess those types of data before feeding them into the model for training and inference.

The second stage is *online prediction*. The model is implemented as a module in the ADS to monitor the outputs of the perception and localization modules during the AV operation. When the AV receives malicious sensory data crafted by the adversary (e.g., traffic sign with the adversarial patch, spoofed GPS signals), the anomaly detector is able to identify such suspicious events from these two monitor

modules, and then send notifications to the control module. The control module will perform some mitigation actions, e.g., stopping the vehicle, warning and asking the driver in the vehicle to take control of it.

6.4.2 T-GP: One-class Model for Anomaly Detection

We design T-GP, a novel one-class classification model based on the transformer structure, for each vehicle to achieve anomaly detection. A transformer [223] is a deep neural network structure using the self-attention mechanism. It replaces the Recurrent Neural Network (RNN) structure with an encoder and decoder. It can significantly improve the model accuracy for Natural Language Processing (NLP) tasks. Besides, it is also highly interpretable and supports fully parallel computing. Recently, researchers extended the transformer structure to the domain of Computer Vision (CV) [224], which also demonstrates remarkable performance for image classification.

Inspired by the successful applications of the transformer in the NLP and CV domains, we aim to apply it to build a one-class model for anomaly detection. Figure 6.5 shows the network structure of our proposed model, T-GP. It adopts an input embedding component (i.e., Input Embedding Matrix) to map each original input data into a vector with a fixed length and an encoder (i.e., Transformer Encoder) as the feature extractor to learn the hidden patterns of normal data and detect abnormal data (i.e., malicious sensory input in ADSs).

Specifically, the input $X = (x_1^T, \dots, x_t^T)^T \in \mathbb{R}^{t \times P}$ of the model is a two-dimensional matrix, where t is the length of the input sequence, P is the dimension of each input data x_i , i.e., $x_i \in \mathbb{R}^{1 \times P}$, for $i = 1, 2, \dots, t$, and $(\cdot)^T$ denotes the transpose operator. Note that our model is unified and can accept both the image data and IMU time series data. Each image is reshaped into a sequence of flattened 2D patches by dividing the original image into t patches [224]. For the IMU data, each single sample x_i is recorded at a time instant. Since the transformer encoder requires a constant latent vector size, each input sequence is first mapped to a fixed-length sequence of patch embeddings using a learnable embedding vector x_{class} , a trainable linear projection E , and a standard learnable 1D position embeddings \mathbf{E}_{pos} [224],

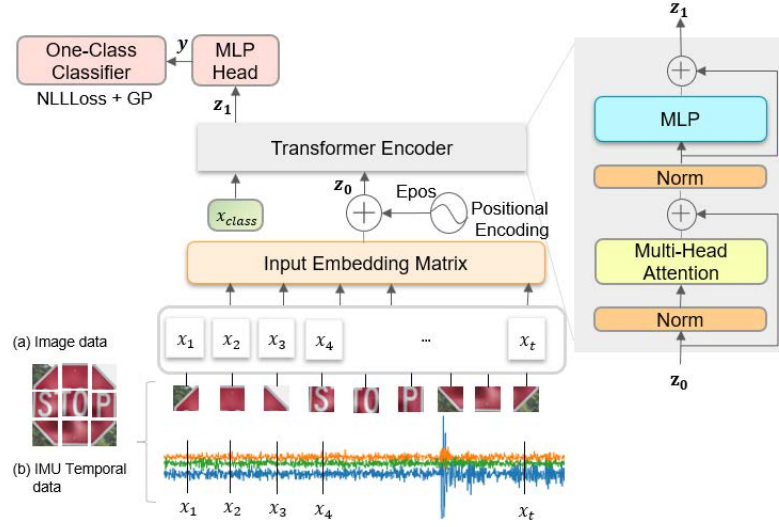


FIGURE 6.5: T-GP model structure.

as given in Equation 6.1:

$$z_0 = (x_{class}^T, \mathbf{E}^T X^T)^T + \mathbf{E}_{pos} \quad (6.1)$$

where $x_{class} \in \mathbb{R}^{1 \times D}$ and its output can be used for classification, $\mathbf{E} \in \mathbb{R}^{P \times D}$ is a fully connected layer, and $\mathbf{E}_{pos} \in \mathbb{R}^{(t+1) \times D}$ is introduced to add the positional information of the input sequence to the patch embeddings.

The patch embeddings z_0 are sent to the transformer encoder, which is used to extract the feature representation of the input data and consists of a Multi-headed Self-Attention (MSA) network and a two-layer Perceptron (MLP) with GELU. Note that the inputs of MSA and MLP are first normalized via layer normalization (LN) [225]. Hence, the operation of the transformer encoder can be formulated as:

$$z'_1 = MSA(LN(z_0)) + z_0 \quad (6.2)$$

$$z_1 = MLP(LN(z'_1)) + z'_1 \quad (6.3)$$

We design a novel loss function in T-GP to achieve one-class classification. Negative Log Likelihood Loss (NLLLoss) is widely used in multi-class classification tasks. However, in our one-class model, the output has only one class, so we use the sigmoid function in NLLLoss to calculate the probability that an input x belongs to the class. It generally requires regularization due to the sigmoid saturation and feature bias in NLLLoss [80]. It means that an unimportant feature with a

larger value may have larger effects on the computation of the probability. Hence, inspired by [226], which adds 1-Lipschitz constraints to the discriminator of WGAN by gradient penalty (GP), we apply the gradient penalty in T-GP to mitigate such biases and obtain the following loss function:

$$\begin{aligned} \text{loss} = & E_{x \sim P_x}[-\log(\text{Sigmoid}(f(x)))] \\ & + \lambda E_{x \sim P_x}[(\|\nabla_x f(x)\|_2 - 1)^2] \end{aligned} \quad (6.4)$$

The first term is NLLoss and the second one is gradient penalty. P_x denotes the data distribution of the given positive class, and λ is a hyper-parameter to balance the penalty. $\text{Sigmoid}(f(x)) \in (0, 1)$ is the probability that x belongs to the positive class. The advantage of the gradient penalty will be demonstrated in our evaluations by comparing with the H-regularization [80].

6.5 Model Evolution

It is possible that the AV behaviors can drift over a long period of time, possibly caused by the varied environment and road conditions. Hence, frequent model update is necessary to maintain the high anomaly detection performance. The vehicle can periodically collect the runtime data when it is at the normal state. Then it fine-tunes the detection model based on such data. This process is lightweight compared with model training from scratch, and thus computationally feasible using the on-board computer. We further leverage two technologies to enhance the efficiency of model update.

6.5.1 Lifelong Learning

ADS-Lead applies lifelong learning for model evolution with runtime data. Lifelong learning is defined as an adaptive machine learning algorithm, which is able to progressively learn from a continuous stream of information over a long time span [227]. A good lifelong learning algorithm can produce a machine learning model with great accommodation of the new information. Lifelong learning has become more important in autonomous agents and systems, which need to interact with the dynamic real world.

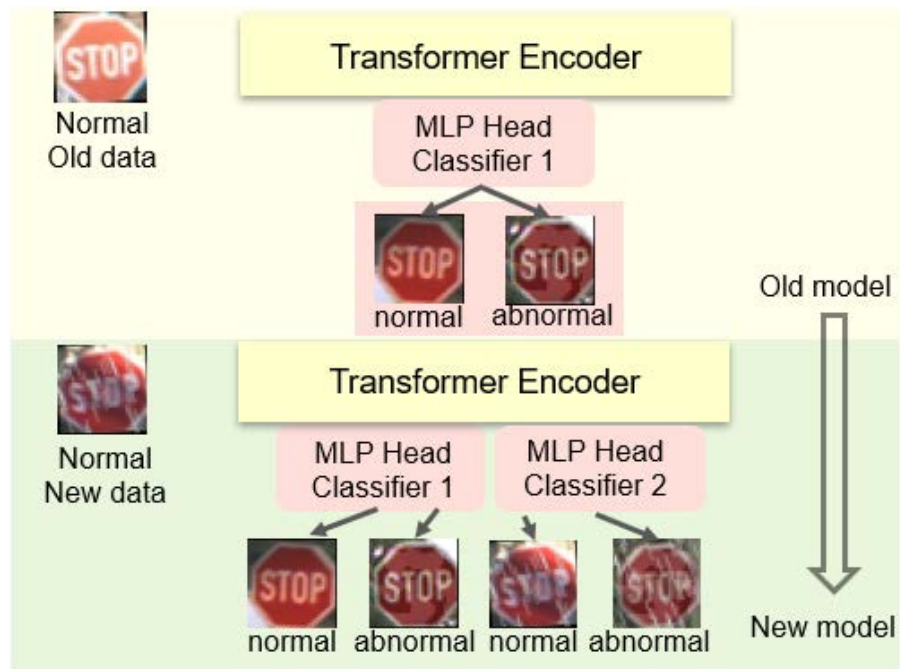


FIGURE 6.6: Lifelong learning for one-class model update.

A variety of strategies have been designed to achieve lifelong learning. In ADS-Lead, we use the method proposed in [80], which is a continuous learning process and shows better performance than other strategies. Its idea is to train a new sub-classifier for each new task, and the prediction is done by selecting the result of one of the existing sub-classifiers. The detailed process of lifelong learning in ADS-Lead is shown in Figure 6.6. As described in Section 6.4.2, the T-GP model consists of two main parts: a transformer encoder, which extracts feature vectors of different formats of sensor data, and an MLP head, which classifies the extracted features to make decisions. During lifelong learning, the transformer encoder is fixed. Every time a new dataset is collected, ADS-Lead trains a new MLP head to memorize the new data distributions. Then new MLP heads will be integrated with the old ones in the updated model for decision making. Note that to guarantee the scale of the model, we may set the maximal number of MLP head classifiers and forget the very old classifiers. In this way, we can guarantee to not only remember the historic knowledge but also learn new information from the newly acquired data.

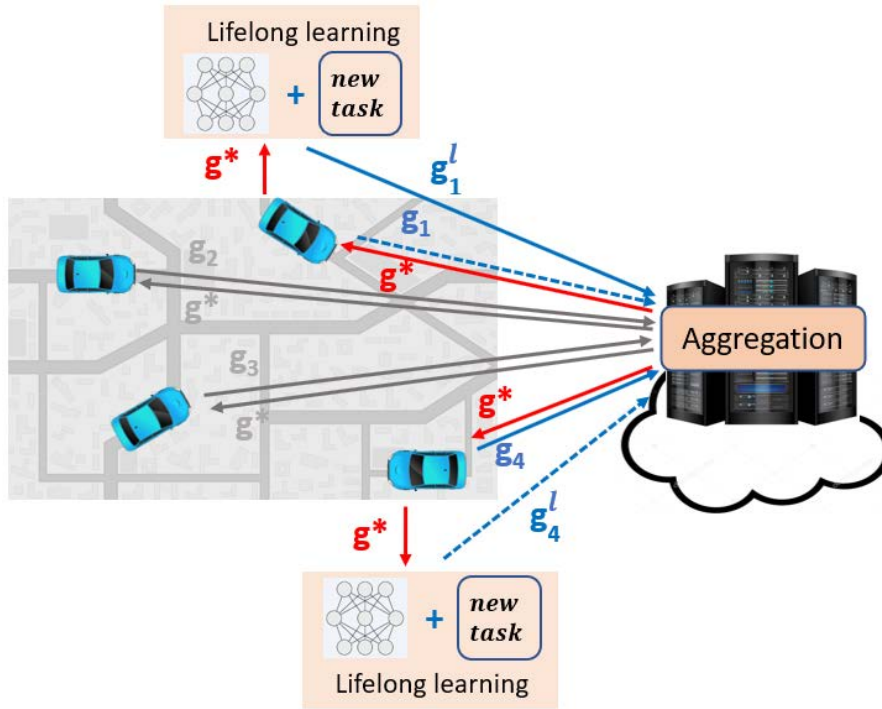


FIGURE 6.7: Model training and update with federated learning in ADS-Lead.

6.5.2 Model Update with Federated Learning

Since there may be multiple AVs on the roads, and different vehicles have different private data, they can collaborate to train and update a more robust model. Hence, we further propose to use federated learning [228, 229] to optimize the model training process, which enables different vehicles to collaborate on the model training without releasing their data. Hence, the data privacy of the vehicles (e.g., location) is preserved compared to the case where the data are offloaded to the cloud for model training. Figure 6.7 shows the process of the detection model update with federated learning. A centralized Parameter Server (PS) is introduced in the remote cloud. Each AV in the C-ITS is able to talk with the PS via the V2C communication technology [230]. During the training process, each vehicle trains the model gradient g_i from its local collected data, and uploads the results to the PS. Then the PS will receive multiple gradients from different vehicles. It aggregates them into one gradient vector g^* by calculating the average value, and releases the new model to each vehicle in the C-ITS. So each vehicle can use the latest model for online anomaly detection with better generalization and performance.

It is worth noting that we adopt the asynchronous training instead of synchronous

training. The PS does not need to wait for the gradients from all the vehicles in the network, since some vehicles may not participate in the model update process. It performs the gradient aggregation and model release at a fixed frequency, to guarantee the model update service is always available. It is possible that some vehicles are malicious or compromised, trying to send the PS wrong updates to compromise the detection model. We can adopt some sophisticated aggregation rules [231] to filter out such malicious gradients. Besides, we can also follow the works [232] to further mitigate the indirect leakage from the gradients. Implementation of these advanced solutions into ADS-Lead is our future work.

6.6 Evaluations

In this section, we evaluate the effectiveness and robustness of the proposed ADS-Lead system and T-GP model against the three kinds of attacks described in Section 6.3.3.

6.6.1 Evaluation of T-GP

We first evaluate the performance of T-GP on GPS attacks, traffic sign attacks, and lane detection attacks.

6.6.1.1 Defeating Localization Attacks

Data Sets. Since there are no public datasets for GPS spoofing attacks, we deploy the attacks in Baidu *Apollo* 5.0 running with the *LGSVL* simulator on the San Francisco map, and collect data for normal and malicious cases. We consider the attack scenario where an adversarial vehicle tailgates the victim AV while launching GPS spoofing. Following the attack settings in [173], we consider two concrete adversarial goals as shown in Figure 6.8: off-road attack aims to deviate the AV to hit the curb; wrong-way attack aims to deviate the AV to the opposite lane and hit the oncoming vehicle.

GPS spoofing will cause a sudden change of the AV’s localization computation, resulting in the change of AV’s motion. Hence, we monitor the IMU messages, whose

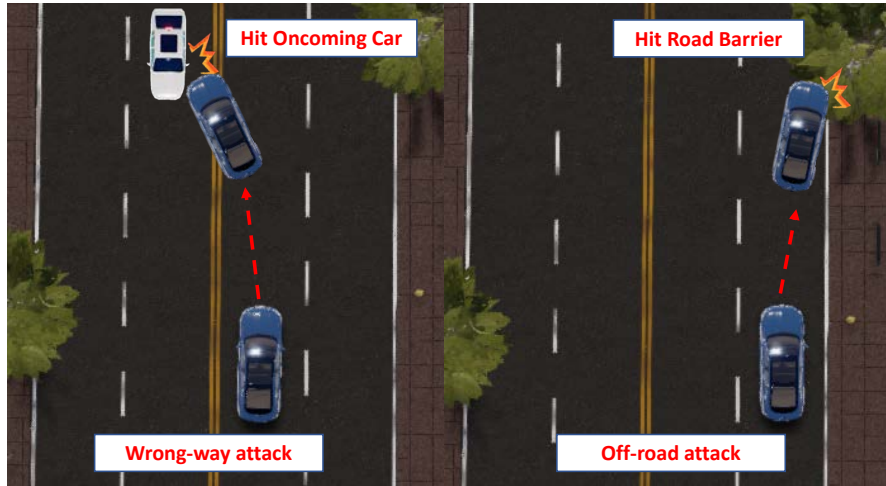


FIGURE 6.8: GPS spoofing attacks in LGSVL simulator.

channel name is `/apollo/sensor/gnss/corrected_imu` in the *Apollo* ADS. There are three kinds of motion data in the IMU messages and each one is a 3D vector: linear acceleration (ax, ay, az) , angular velocity (avx, avy, avz) , and Euler angles (α, β, γ) . Since the current HD map for *Apollo* does not contain the altitude information, only the linear accelerations ax and ay , angular velocity avz , and Euler angle γ are affected by the motion of the AV. Moreover, based on our observation of the real-time IMU data, these four values exhibit distinct behaviors when the AV deviates from the predetermined path, compared to the scenarios of lane change or turn. Hence, at each time instant, we collect these four types of data as the model features. Figure 6.9 shows two data sequences of the four selected data types during the AV motion under GPS spoofing attacks, where the message sampling frequency is around 85 FPS (Frame-Per-Second) in our experiments.

Since our task is one-class anomaly detection, only benign data are available for model training. The road in the map of LGSVL simulator is flat and we set random NPCs (vehicles and passengers) in the map. We randomly set the destination for the vehicle and collect the four types of IMU data from *Apollo* when the vehicle is in normal and secure states. A total of 32,115 raw data samples are generated for model training. The testing set should contain both the normal and attack samples. We run *Apollo* ten times under the two types of GPS spoofing attacks, and collect the related IMU data. We label the data before the attack occurrence as "normal". We also assign the "abnormal" label to the data collected in a short period right after the GPS spoofing is launched (around 20 new IMU messages). Table 6.1 summarizes the ten testing data sequences.

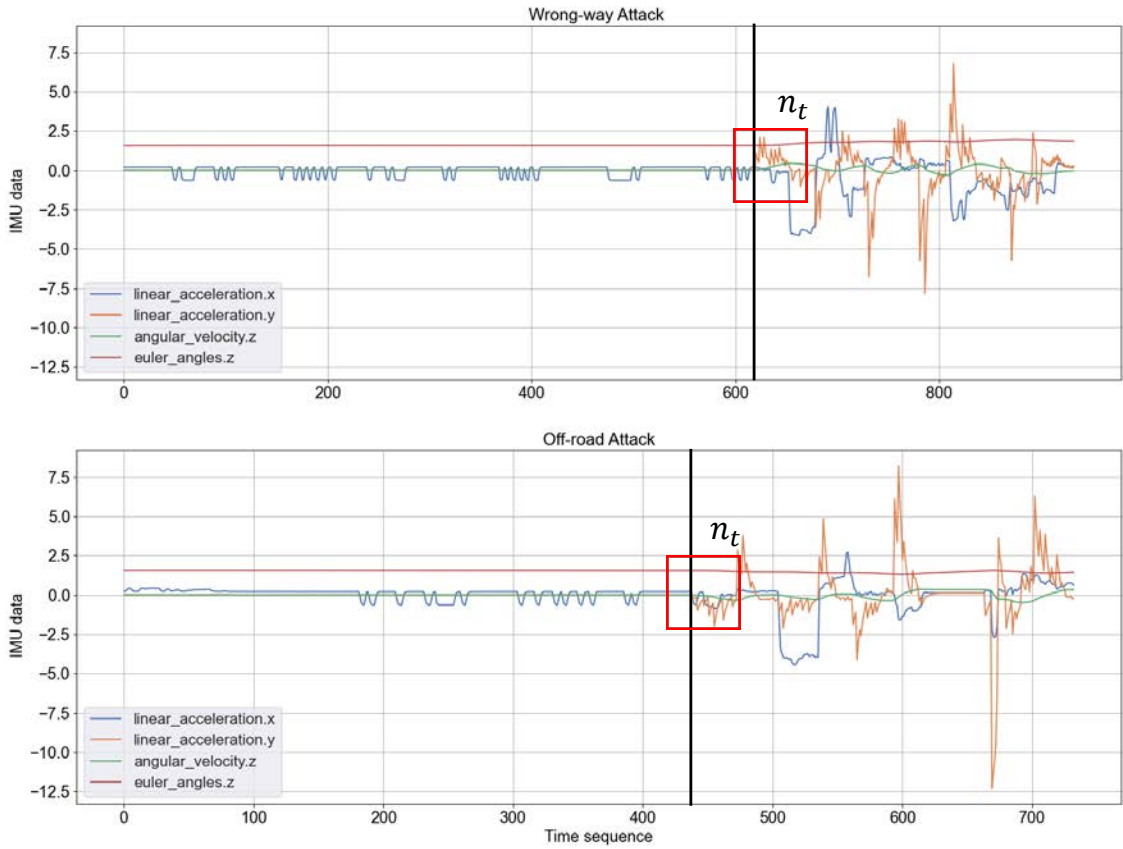


FIGURE 6.9: Data sequences of ax , ay , avz , and γ when the AV is under the off-road and wrong-way attacks, respectively. The black line represents the moment the spoofing attack starts. The red box is the sliding window with the length of $n = 10$ data samples. n_t represents that the attack is detected after n_{th} samples of the attack occurrence.

Once we obtain the training and testing data sequences, we generate the corresponding training and testing datasets by dividing each data sequence into a set of sub-sequences with the length of 10. We use the sliding window method with a stride of 1 to generate the sub-sequences. Hence, a sequence with n samples can generate $(n - 9)$ sub-sequences. Note that we employ the same data preprocessing method to all the models for fair comparison.

Model Configurations. According to the format of the generated data samples, the input dimension of T-GP is set as 10×4 , i.e., each input sequence has 10 consecutive data samples and each sample is a 4D vector. In terms of the model hyper-parameters, we use an embedding dimension of 4 units, 4 transformer heads, and 128 units in the hidden layer of the output MLP head. We use the AdamW optimizer with a learning rate of $1e-4$. λ is set as 0.1.

TABLE 6.1: Number of data samples in each testing sequence

Sequence		#0	#1	#2	#3	#4	#5	#6	#7	#8	#9
off-road	normal	420	423	237	294	571	494	210	461	363	535
attack	abnormal	20	17	23	16	19	16	20	19	17	25
wrong-way	normal	245	616	418	325	550	274	271	338	204	396
attack	abnormal	25	14	22	26	20	16	19	22	16	24

Baseline Methods We compare our T-GP model with the following baselines.

- *OC-SVM* [233]: this is a traditional one-class classifier based on kernel SVM. In our implementation, the RBF kernel is applied and the hyper-parameter is selected from a set of discretized values in the interval $[0, 1]$.
- *iForest* [234]: this is another popular one-class classifier. It isolates anomaly points by building decision trees. We use the default values of the hyper-parameters.
- *Deep-SVDD* [78]: this is a deep one-class model. It classifies anomaly data by penalizing the distance between the extracted feature vector, from the network and the center of the initial hypersphere. Since it only supports non-trivial high-dimensional images, we use the transformer encoder in T-GP to extract features for Deep-SVDD.
- *HRN* [80]: this is a state-of-the-art one-class models based on holistic regularization. We use the default structure with a three-layer perception, whose input, hidden and output dimensions are 40, 100, and 1, respectively.
- *T-L2*: this is a variant of our T-GP model. We replace the gradient penalty-based regularization with L2-regularization.

Evaluation Results We use the standard metrics (precision, recall and F1-measure) to quantify and compare the performance of our model with others baselines. Figure 6.16 shows the results on the testing datasets of off-road and wrong-way attacks. Note that in anomaly detection tasks, anomaly data are considered as positive. From Figures 6.10 and 6.11, we can find that for both kinds of attacks, the transformer-based models (i.e., T-L2 and T-GP) have higher average precision and lower variance than other models. Hence, the adoption of the transformer exhibits better robustness. They can detect anomalies more precisely with fewer false alarms. As shown in Figures 6.12 and 6.13, the two transformer-based models also have higher average recall than others, indicating that they have smaller false negative rates, i.e., missing fewer anomaly data. Moreover, compared to T-L2, T-GP

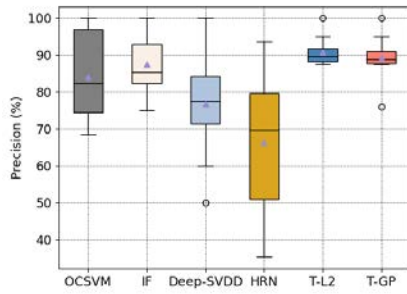


FIGURE 6.10: Precision (off-road)

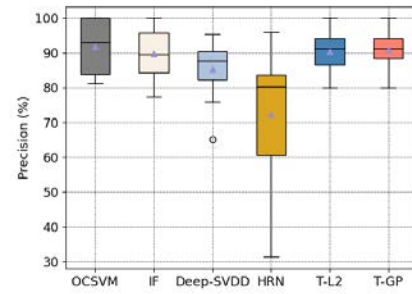


FIGURE 6.11: Precision (wrong-way)

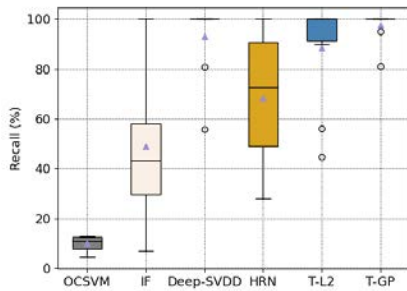


FIGURE 6.12: Recall (off-road)

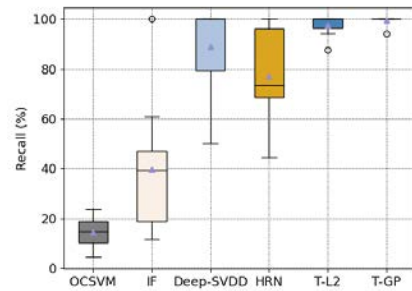


FIGURE 6.13: Recall (wrong-way)

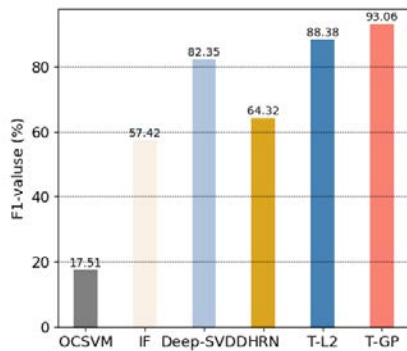


FIGURE 6.14: F1-measure (off-road)

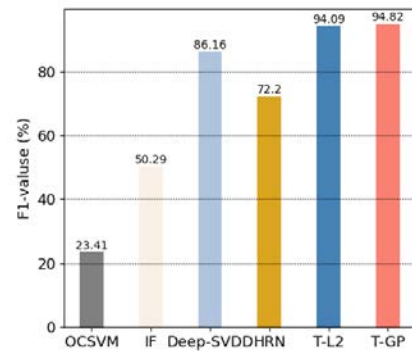


FIGURE 6.15: F1-measure (wrong-way)

FIGURE 6.16: Results of Precision, Recall and F1-measure on the two GPS spoofing attack datasets.

can provide more fine-grained control over the penalty function and a higher recall with smaller fluctuations. The F1-measure results are shown in Figures 6.14 and 6.15. We also find that T-GP has the highest F1-measure. It means T-GP not only has high precision and recall values, but also can balance these two measures. Hence, we conclude that the proposed T-GP outperforms other one-class models on the 20 testing sequences.

TABLE 6.2: Levene’s test and t-test on F1-value between our T-GP and each of other models. A higher value indicates the model is more similar as T-GP in detection performance.

Baselines		OC-SVM	IF	DSVDD	HRN	T-GP
Off-road attack	Levene’s test	0.3908	0.0025	0.1346	0.0060	0.2606
	T-test	4e-11	0.0012	0.0026	0.0007	0.1337
Wrong-way attack	Levene’s test	0.0180	0.0023	0.0482	0.0003	0.5477
	T-test	2e-10	0.0003	0.0489	0.0017	0.2549

To analyze the statistical significance of these models, we perform Levene’s test and two-sample t-test for equal variance testing and equal mean testing in terms of the F1-measure. The results are shown in Table 6.2. We can observe that given the 95% confidence interval, our T-GP has significant differences for the mean of F1-measure, from other non-transformer models. Hence, T-GP demonstrates higher performance statistically. Moreover, we can find that there are no significant differences between T-GP and T-L2, indicating the two loss functions in T-GP and T-L2 have similar performance in balancing the precision and recall.

During the online anomaly detection, another important requirement is to detect attacks promptly so that we can prevent accidents as soon as possible. Hence, we also compute the detection time of different models in *Apollo*. We find that T-GP can detect an attack within 6 data samples after launching the attack ($\sim 0.07s$), while other models need more time to identify anomalous events, which is relatively less practical in reality.

In conclusion, our transformer-based model can accurately disclose the underlying dependency in the time series data during the AV’s motion, whilst other models cannot describe such temporal relations, even using the sliding window technique. Moreover, the results also show that the transformer with GP is better than with L2 regularization.

6.6.1.2 Defeating Traffic Sign Recognition Attacks

We examine the effectiveness of our model on detecting adversarial traffic signs.

Datasets. We conduct our experiments on the *GTSRB* (German Traffic Sign Recognition Benchmark) dataset, which only contains clean traffic sign images. We select four representative categories of traffic signs, i.e., stop, speed limit, keep



FIGURE 6.17: Clean (first row) and adversarial (second row) traffic signs. (a) Boundary attack (b) Poster attack.

TABLE 6.3: Number of images in each dataset

Attack	Traffic Sign	Training	Test	
		Normal	Normal	Abnormal
Boundary	Stop	780	270	20
Poster	Stop	780	270	270
Poster	Speed limit 30	2220	720	720
Poster	Keep right	2070	690	690
Poster	Traffic signals	600	180	180

right and traffic signals, from this dataset for training. The numbers of these categories are 780, 2220, 2070, and 600, respectively. For testing, we adopt the boundary attack [200] and poster attack [62] to generate adversarial example from the normal testing images. Specifically, we perform the boundary attack on the stop sign category to generate 20 adversarial samples, and the poster attack on the four categories to generate the same numbers of adversarial images as the testing samples. Figure 6.17 visualizes the adversarial samples of different attacks and traffic signs.

Table 6.3 gives the details of the training and testing datasets. We remove 10% border of each category and resize the images to 32×32 as presented in [78]. In addition, global contrast normalization using L1-norm is applied.

Model Configurations. For T-GP, we use the same structure described in Section 6.4.2, where each input image is divided into 64 patches with an equal size of 4×4 . According to the scale of the datasets, λ is set as around 1.5 (similar results for $[0.1, 3]$) and the initial learning rate is $3e-4$.

Baseline Methods. We compare our model with Deep-SVDD and HRN in detecting adversarial traffic signs. Specifically, for Deep-SVDD, we apply a CNN

TABLE 6.4: Average AUCs for different models in detecting different attacks

Attack	Traffic Sign	Deep-SVDD	HRN	T-GP
Boundary	Stop	80.78%	95.5%	98.2%
Poster	Stop	72.46%	72.83%	93.26%
Poster	Speed limit 30	51.96%	64.18 %	65.56%
Poster	Keep right	62.64%	83.54%	84.03%
Poster	Traffic signals	76.46%	85.68%	77.83%

structure with three filters of sizes $32 \times (5 \times 5 \times 3)$, $64 \times (5 \times 5 \times 3)$ and $128 \times (5 \times 5 \times 3)$, followed by a fully connected layer with 128 units. We get the maximum accuracy with the AdamW optimizer whose learning rate is set as $1e - 3$. For HRN, a three-layer MLP is adopted with the size of $3 \times [1024-300]-[900-300]-[300-1]$. The first layer contains three sub-modules (each one has a size of $[1024-300]$) to deal with 3 channels, and the outputs are concatenated as the input of the second layer; the second and third layers have the sizes of $[900-300]$ and $[300-1]$, respectively. The optimizer is set as SGD with momentum and the learning rate is $5e - 4$.

Evaluation Results. Table 6.4 shows the AUC (Area Under the ROC) values of different models for detecting the boundary and poster attacks on different traffic signs. The results show that our model outperforms Deep-SVDD and HRN for both kinds of attacks.

We also compare the performance of the transformer-based one-class model with three kinds of loss functions: NLLoss, L2 penalty and GP (gradient penalty). Table 6.5 shows the detection results of the loss functions on the poster attack. We can observe that the model with gradient penalty introduced from WGAN has higher AUC values than the other two loss functions. A possible reason is that, since the output has just one class, we use $\text{sigmoid}(\cdot)$ function in NLLoss to calculate the probability of input x labeling $y(x)$. When minimizing the NLLoss function, we need to include a penalty function to reduce the possibility of feature bias. Such feature bias exists as we do not have any other classes to compare, and do not know which feature is essential for class differentiation. Some features and their related parameters with high values may not be important, thus leading a low accuracy.

TABLE 6.5: Average AUCs for different transformers and loss functions in detecting poster attacks

Solution	Stop	Speed limit 30	Keep right	Traffic signals
T-NLL	88.81%	59.07%	82.01%	77.59%
T-L2	60.25%	63.88%	81.85%	86.98%
T-GP	93.26%	65.56%	84.03%	77.83%



FIGURE 6.18: Samples of fixed-size patch and varied-size patch.

6.6.1.3 Defeating Lane Detection Attacks

Datasets To evaluate the effectiveness of our method on detecting lane attacks, we adopt the widely-used *Tusimple* traffic lane dataset. This dataset consists of 6,408 annotated images, which are the latest frames from video clips recorded by a high-resolution (720×1280) forward-view camera under various traffic and weather conditions on highways of United States in the daytime. It is split into a training set (3268), a validation set (358), and a testing set (2782). We generate two types of adversarial examples from the validation set following the Patch Attack [82], including fixed-size patch and varied-size patch (Figure 6.2). The size of the former patch is 100×100 , and the later patch is scaled according to the lane width and lane marker height. After adding the adversarial patches, all the images are scaled to the size of 320×320 . For each type of patches, we obtain 3268 normal images used for training, 358 normal images and 358 abnormal images for testing. Figure 6.18 shows two adversarial samples under the fixed- and varied-size patch attacks, respectively.

Model Configurations Different with the configurations in adversarial traffic sign detection, we add a split layer before the model input, thus the images are split into fixed-size patches first in order to capture the anomalies more carefully. Specifically, we split each image of $320 \times 320 \times 3$ to 100 patches of $32 \times 32 \times 3$. This gives us 3268×100 training samples, 358×100 normal testing samples and 358×100 abnormal testing samples. During testing, if any one of the 100 patches is flagged

TABLE 6.6: Average AUCs of different models in detecting the patch attacks

Patch Attack	Deep-SVDD	HRN	T-GP
Fixed-size	68.19%	52.79%	92.25%
Varied-size	60.60%	51.54%	67.86%

as abnormal, then the entire image is regarded as anomaly. We use the same preprocessing method for all the models to achieve fair comparison.

Baseline Methods. We compare our transformer-based method with Deep-SVDD and HRN. The two models follow the same settings in Section 6.6.1.2.

Evaluation Results. Table 6.6 shows the average AUC values for different models. We observe that T-GP shows better performance than the other two baseline models. Particularly, all these models have relatively low accuracy in detecting the varied patch attacks. One possible reason is that some patches are too small to be recognized as adversarial samples, causing higher false negative rates. But T-GP still outperforms prior solutions. We will explore new models to further enhance the detection accuracy as future work.

6.6.2 Evaluation of ADS-Lead

We evaluate the effectiveness of ADS-Lead with lifelong and federated learning on the attack detection. As we discovered in the GPS spoofing detection experiments, the pattern of IMU data shows no divergence in different scenarios when the AV is running in normal and secure states. Hence, the redundant IMU samples from different vehicles cannot further improve the performance of the proposed detector. Therefore, we mainly focus on the detection of traffic sign attacks and lane attacks in this section.

6.6.2.1 Datasets

In federated learning, each vehicle participates in gradient update during the training process. Therefore, assigning sufficient training data for each vehicle is crucial for the convergence of the model. To amend this, data argumentation is performed over the training datasets on each vehicle.

TABLE 6.7: Number of images in each traffic sign datasets. Note the abnormal data are generated by the poster attack

Datasets	Traffic Sign	Training	Test	
		Normal	Normal	Abnormal
Task 1 (Non-Rainy)	Stop	3855	270	50
	Speed limit 30	10970	720	100
	Keep right	10230	690	100
	Traffic signals	2965	180	20
Task 2 (Rainy)	Stop	1605	270	50
	Speed limit 30	4570	720	100
	Keep right	4260	690	100
	Traffic signals	2965	180	20

TABLE 6.8: Number of images in lane detection datasets. Note the abnormal data in Task 1 and Task 2 include both varied and fixed patch attacks

Fixed_Varied	Training	Test	
	Normal	Normal	Abnormal
Task 1 (Non-Rainy)	1634	358	358
Task 2 (Rainy)	1634	358	358

For the traffic sign data sets, we first rotate the images clockwise and counter-clockwise by 5, 10 and 15 degrees, respectively; second, we divide the data into two subsets to represent tasks at two different time instants. Considering the impact of environmental factors (e.g., light, whether and camera resolution), we randomly synthesize the latter subset with the effects of rain by adding controlled random noise. The statistics of the traffic sign datasets are reported in Table 6.7. Note that for in each testing set, the abnormal samples are generated by the poster attack.

For lane detection attacks, we expand the data set by adding rain effects to the original images. Specifically, we first divide the original training data set equally into two subsets: Task 1 for the first phase of training and Task 2 for model update; moreover, we synthesize the images in Task 2 with the same rainy effects as in the traffic sign data set. Second, for either testing data set of Task 1 or Task 2, we apply both the fixed- and varied-size patch attacks to generate adversarial samples, and the testing data set in Task 2 is also added with the rain effects. Table 6.8 shows the statistics of the two data sets. Figure 6.19 shows some samples with rainy effects in traffic sign and lane detection data sets.

6.6.2.2 Baseline Models and Model Configurations

We compare the following algorithms for model update:



FIGURE 6.19: The synthesized images with rain.

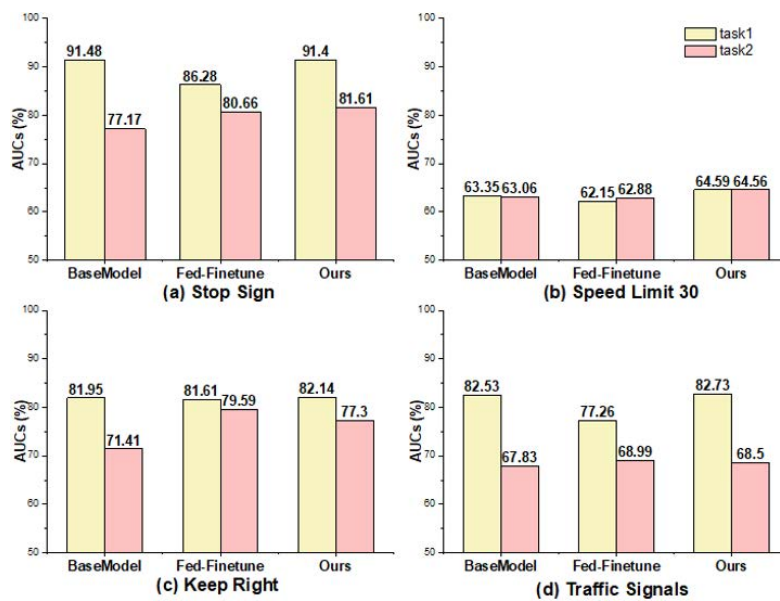


FIGURE 6.20: Evaluation results on traffic sign dataset. **BaseModel**: the federated learning model is trained on Task 1, and tested on Task 1 and Task 2. **Fed-Finetune**: the federated learning model trained on Task 1, and finetuned on Task 2. **Our ADS-Lead**: the model is trained on Task 1 and lifelong learned on Task 2.

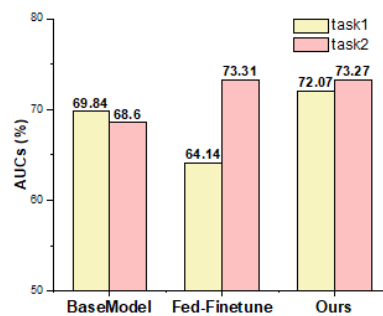


FIGURE 6.21: Evaluation results on the lane detection dataset.

BaseModel: This is for federated learning only. In our experiments, we consider a system of 5 vehicles, partition the training data sets equally into 5 sets, and assign each to one vehicle. For each round, we randomly select 4 vehicles to update the gradients for aggregation, to simulate the asynchronous mechanism. We set the batch size as 32 and run 50 epochs with the same hyper-parameters of T-GP as shown in Section 6.4.

Fed-Finetune: In addition to training the model with federated learning, we further finetune the aggregated model using the dataset from Task 2.

ADS-Lead: This is our solution in ADS-Lead. In addition to training the model with federated learning, we further perform lifelong learning on Task 2 to obtain the updated model. We adopt the same federated learning settings, i.e., batch size of 32 and 50 running epochs.

6.6.2.3 Evaluation Results

Figure 6.20 presents the AUC values of the three algorithms for the two tasks of traffic sign attack detection, respectively. We can find that **BaseModel** performs well on Task 1 (e.g., 91.48% for stop sign) but not well on Task 2 (e.g., 77.17% for stop sign), as the model is trained only from Task 1. **Fed-Finetune** improves the performance over **BaseModel** on Task 2 (e.g., from 77.17% to 80.66% for stop sign) due to the fine-tuning operation with the dataset of Task 2. However, its performance on Task 1 is degraded (e.g., from 91.48% to 86.28% for stop sign). This indicates that simply finetuning the model can make it learn new knowledge but forget some prior knowledge. Our ADS-Lead model can balance the performance on both Task 1 and Task 2. In detail, we observe that the model performance on Task 1 and Task 2 is similar as **BaseModel** and **Fed-Finetune**, respectively. Hence, with lifelong learning, our model can not only learn new knowledge of new tasks (e.g., Task 2) but also remember the learned knowledge from previous tasks (e.g., Task 1).

Similarly, Figure 6.21 demonstrates the effectiveness of ADS-Lead on lane attack detection. For **Fed-Finetune**, after model fine-tuning on Task 2, the prediction accuracy of Task 2 rises from 68.60% to 73.32%, whereas the accuracy of Task 1 drops from 68.60% to 64.14%. Fortunately, with lifelong learning, our ADS-Lead balance the model performance on Task 1 and Task 2 significantly.

Even though the results are encouraging, the improvement brought by lifelong learning for Speed Limit 30 and Keep Right signs is limited. This is because the data in Task 2 are synthesized by only adding normal noise to simulate the rainy effects, and the pattern difference between Task 1 and Task 2 is not very significant. Despite that, the experimental results still show that our proposed ADS-Lead is practical for anomaly detection in ADSs, achieved by the globally-trained high-quality model with lifelong learning.

6.6.3 Discussion on the Robustness of ADS-Lead

Finally, we discuss the robustness of our system against possible adaptive attacks. Even though the adversary knows the defense mechanism, it is hard for him to attack our detector. On one hand, federated learning can mitigate the attacks on a single vehicle, as the server will aggregate the local models to generate a global one. On the other hand, with lifelong learning, the server will update the model over time, so each vehicle will update its model such that the adversary cannot use the previous knowledge on the model to launch attacks. We also point out that it is possible for the adversary to launch attacks during two successive update time instants. However, these attacks can be mitigated by setting specific update frequency such that there is no enough time for the adversary to retrieve the model information and then launch proper attacks. How to design more advanced attacks as well as enhancing the system will be our future work.

6.7 Conclusion

In this chapter, we propose ADS-Lead, a novel system based on federated learning and lifelong learning to detect anomalies in the lane following scenario of ADSs. We introduce T-GP, a novel one-class classification model with a transformer encoder for feature extraction and new loss function with gradient penalty. It is able to detect GPS spoofing, traffic sign recognition and lane detection attacks with high accuracy. We extensively evaluate our model on the mainstream Baidu *Apollo* ADS with the LGSVL simulator, and two public traffic datasets: GTSRB and Tusimple. The results show that T-GP significantly outperforms existing state-of-the-art one-class models. We also show the practicality and effectiveness of attack detection

with advanced model evolution solutions. In the future, we aim to incorporate our system into real-world AVs and study the anomaly detection of other sensor attacks (e.g., Lidar attacks) and scenarios (e.g., lane changing and overtaking).

Chapter 7

Conclusion and Future Work

In this chapter, we first give a summary of the work conducted in this thesis and then discuss some future research directions based on our current results.

7.1 Conclusion

The security of ADS is of paramount importance. Although ADS has the potential to revolutionize transportation by increasing safety, efficiency, and convenience, it also carries risks in the context of adversarial attacks. Unfortunately, in the real-world scenario, existing research falls short of providing a comprehensive evaluation, leaving us with an incomplete understanding of the threat posed by physical adversarial attacks on AD. Consequently, there is an urgent imperative for the development of evaluation tools tailored to these specific threats. Nevertheless, the nature of rich semantics in the real world, coupled with the diverse attack methods proposed in recent years, makes it extremely challenging to test these attacks against ADSs, which also makes it difficult to design unified defense methods. In this thesis, we concentrate on addressing these issues by conducting comprehensive physical adversarial evaluations, and vulnerability testings and designing unified defense methods based on deep insights into ADS.

This research starts with building the first comprehensive benchmarking platform `BatAV` to evaluate the physical backdoor vulnerability of modern ADS. Backdoor

attacks have evolved to be one of the most severe threats to DL models. Investigating backdoor attacks against ADS perception allows us for the first time to understand how much the backdoor attacks can affect the end-to-end ADS. Specifically, **BatAV** can automatically synthesize backdoor attacks targeting different vision-based perception functions with the customized attack budget and trigger design. It can also deploy these attacks to three levels of testbeds (dataset, ADS simulator, physical vehicle) for thorough analysis. **BatAV** includes 7 backdoor attacks with 4 representative triggers to attack 3 perception functions and 11 real-world models. Leveraging **BatAV**, we summarise several novel observations about the vulnerability of perception models in ADSs.

After that, we develop a vulnerability testing platform **STFA** for the decision-making module of ADS, this gives us the chance to find new attack surfaces by leveraging the vulnerabilities. The decision-making module serves as a crucial intermediary between the perception and control modules. It utilizes real-time perception information to generate obstacle trajectories and the vehicle’s motion trajectory, enabling the system to make correct actions, e.g., braking or steering. In practice, this module comprises not only machine learning algorithms but also traditional rule-based algorithms. Its complexity makes it exceedingly challenging to design such a platform. To overcome the challenge, we introduce the fuzzing techniques to discover the system-level vulnerability in end-to-end ADS. Inspired by the vulnerabilities, we propose two novel types of attacks, i.e., *Direct-attack* and *Indirect-attack*, based on the interaction between the ego vehicle and non-playable characters (NPCs), and design a total of seven novel attack methods. **STFA** leverages the Large Language Model (LLM) to extract accident information from real-world accident reports and then employs a genetic algorithm to guide the generation of adversarial scenarios that violate given specifications provided by the LLM. It is a fully automated attack generation platform that requires no manual operations. Currently, **STFA** surpasses 2.5 million lines of code.

Various studies have investigated the adversarial attacks to the perception module in the ADS. Therefore, we then focus on developing novel defense methods against adversarial attacks on ADS. Specifically, we introduce two novel defense methods: (1) **ADS-Lead**, an efficient collaborative anomaly detection methodology to protect the lane-following mechanism of ADSs. **ADS-Lead** is equipped with a novel transformer-based one-class classification model to identify time series anomalies

(GPS spoofing) and adversarial image examples (traffic sign and lane recognition attacks). Besides, AVs inside the Cooperative Intelligent Transport Systems (C-ITS) form a cooperative network, enabling us to apply federated learning to our anomaly detection method, where the vehicles in the C-ITS jointly update the detection model with higher model generalization and data privacy. (2) **VisionGuard**, a unified defense framework that can detect and mitigate various physical adversarial attacks to vision-based perception functions. Before that, we first comprehensively evaluate 9 state-of-the-art physical adversarial attack methods in real-world scenarios. We observe that none of the existing adversarial attacks can achieve perfect attack effectiveness and keep consistency over time. Based on such observations, the key insight of **VisionGuard** is to leverage the spatiotemporal inconsistency to comprehensively identify different adversarial attacks. **VisionGuard** consists of three modules: State Correction Module (SCM) for obtaining the current driving states by raw data calibration and integration, State Prediction Module (SPM) for predicting the motion states by tracking historical states, and Attack Detection Module (ADM) for checking the motion state inconsistency. It is agnostic to attack goals, target objects, models, sensors, adversarial objectives, and contextual information.

7.2 Future Work

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

- **Novel backdoor defense platform.** In our work, we integrated various backdoor attack techniques to different perception functions in ADS, to comprehensively understand how much the backdoor attacks can affect ADS. In this way, it would be better if we could include physical backdoor defense strategies into **BatAV**. Indeed, there are only a few libraries or benchmarks for backdoor learning that have included defense methods, e.g., TrojAI [235], TrojanZoo [94], and BackdoorBench [95]. However, all these benchmarks mainly focused on standard image classification tasks, while ADS contains diverse models with different tasks, making them not applicable to AD scenarios. In the future, we will investigate all the state-of-the-art defense methods

to comprehensively evaluate them in the AD context and to understand how much these defense methods can secure ADS.

- **Novel backdoor defense solution for ADS.** It is extremely hard to design a unified defense method to defeat backdoor attacks targeting diverse functions and model types in ADS. In the future, we will combine the Monitor-Analyze-Plan-Execute over a shared Knowledge (MAPE-K) [236] techniques to design the real-time unified defense method, thus improving the robustness of ADS.
- **Novel decision-making defense method.** In STFA, we proposed a novel fuzzing-based vulnerability testing platform to find new attack surfaces of decision-making modules in ADS. However, we must devise sophisticated decision-making defense strategies that align with the intricacies of the attacks. This entails an in-depth exploration of techniques such as anomaly detection, dynamic policy adaptation, and reinforcement learning-based approaches to fortify decision-making modules against adversarial manipulation. This will be my future work.
- **Conditional physical adversarial attacks.** We have evaluated various state-of-the-art physical adversarial attacks, and we find that existing efforts primarily focus on one adversarial example to achieve a specific attack goal, e.g., adding perturbations to a stop sign, resulting in a semantic transformation (e.g., detected as a speed limit 100). While some methods use real-time projection to dynamically change the attack target, such approaches are costly and easily detectable, e.g., leveraging a drone to project an adversarial example onto an electronic billboard. In contrast, we hope to utilize an affordable and inconspicuous paper sticker, which can dynamically alter the attack target in real-time by leveraging various lighting techniques.
- **Backdoor attack and defense of MSF-based perception in ADS.** Production-level ADS, e.g., Baidu Apollo, and Google Waymo, generally adopt camera-LiDAR based sensor fusion techniques to enhance the robustness of ADS perception. However, the DNN-based MSF functions are also vulnerable to backdoor attacks as we discussed in Chapter 3. Due to its complex structure, e.g., inherent modality interactions, multiple attack surfaces, and unbalanced modality contributions, it is important to investigate

backdoor attacks towards MSF techniques. In the future, we hope to design novel backdoor attack methods to effectively and efficiently destroy the MSF functions. In addition, to effectively defend against such attacks, we hope to develop a novel method to specifically address joint backdoor features from the victim models, since generally, these two poisoned modalities have demonstrated joint features that are never discussed in previous works. This also will be my future work.

List of Publications

- **Xingshuo Han**, Yutong Wu, Qingjie Zhang, Yuan Zhou, Yuan Xu, Han Qiu, Guowen Xu, Tianwei Zhang. Backdooring Multimodal Learning. in *Oakland S&P 2024*.
- **Xingshuo Han**, Guowen Xu, Yuan Zhou, Xuehuan Yang, Jiwei Li, Tianwei Zhang. Physical Backdoor Attacks to Lane Detection Systems in Autonomous Driving. In *Proceedings of the 30th ACM International Conference on Multimedia, 2022*.
- **Xingshuo Han**, Yuan Zhou, Kangjie Chen, Han Qiu, Meikang Qiu, Yang Liu, Tianwei Zhang. ADS-lead: Lifelong anomaly detection in autonomous driving systems. in *IEEE Transactions on Intelligent Transportation Systems, 2022*.
- **Xingshuo Han**, Kangjie Chen, Yuan Zhou, Meikang Qiu, Chun Fan, Yang Liu, Tianwei Zhang. A Unified Anomaly Detection Methodology for Lane-Following of Autonomous Driving Systems. in *IEEE International Symposium on Parallel and Distributed Processing with Applications, 2021, (Most Innovative Paper Award)*.
- **Xingshuo Han**, Qingjie Zhang, Guowen Xu, Yuan Xu, Han Qiu, Qi Li, Ke Lu, Tianwei Zhang. BatAV: A Physical Benchmarking Platform for Backdoor Attacks to Autonomous Vehicles.
- **Xingshuo Han***, Kangqiao Zhao*, Gelei Deng, Yuan Xu, Tianwei Zhang. VisionGuard: A Unified Defense Framework against Physical Adversarial Attacks in Autonomous Driving Perception.
- **Xingshuo Han**, Yuan Zhou, Gelei Deng, Yuan Xu, Han Qiu, Tianwei Zhang. DMAFuzz: Dynamic Physical-world Attacks to Decision-making Module in Autonomous Driving Systems.
- Shengmin Xu, **Xingshuo Han**, Guowen Xu, Jianting Ning, Xinyi Huang, Robert H. Deng. *IEEE Transactions on Services Computing, 2023*.

- Yutong Wu, **Xingshuo Han**, Han Qiu, Tianwei Zhang. Computation and Data Efficient Backdoor Attacks. in *International Conference of Computer Vision, 2023*.
- Guowen Xu, **Xingshuo Han**, Shengmin Xu, Tianwei Zhang, Hongwei Li, Xinyi Huang, Robert H. Deng. Hercules: Boosting the Performance of Privacy-preserving Federated Learning. in *IEEE Transactions on Dependable and Secure Computing, 2022*.
- Yuan Xu, **Xingshuo Han**, Gelei Deng, Yang Liu, Jiwei Li, Tianwei Zhang. SoK: Rethinking Sensor Spoofing Attacks against Robotic Vehicles from a Systematic View. in *IEEE European Symposium on Security and Privacy, 2023*.
- Guowen Xu, **Xingshuo Han**, Tianwei Zhang, Shengmin Xu, Jianting Ning, Xinyi Huang, Hongwei Li, Robert H. Deng. SIMC 2.0: Improved Secure ML Inference Against Malicious Clients. in *IEEE Transaction Dependable Secure Computing, 2023*.
- Guowen Xu, **Xingshuo Han**, Gelei Deng, Tianwei Zhang, Shengmin Xu, Anjia Yang, Hongwei Li. VerifyML: Obviously Checking Model Fairness Resilient to Malicious Model Holder. in *IEEE Transaction Dependable Secure Computing, 2023*.
- Jianfei Sun, Guowen Xu, Tianwei Zhang, Xiaochun Cheng, **Xingshuo Han**, Mingjian Tang. Secure data sharing with flexible cross-domain authorization in autonomous vehicle systems. in *IEEE Transactions on Intelligent Transportation Systems, 2022*.

Bibliography

- [1] Baidu apollo project. <https://github.com/ApolloAuto/apollo/>, 2020. 3, 12, 14, 24, 25, 31, 35, 55, 62
- [2] Google waymo. <https://waymo.com/intl/zh-cn/waymo-driver/>, 2017. 3, 14, 92
- [3] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z. Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *ACM Conference on Computer and Communications Security*, 2019. 3, 13, 59, 64
- [4] Jonathan Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. In *Black Hat Europe*, 2015.
- [5] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications. In *Cryptographic Hardware and Embedded Systems*, 2017.
- [6] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z. Morley Mao. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *USENIX Security Symposium*, 2020.
- [7] Yi Zhu, Chenglin Miao, Tianhang Zheng, Foad Hajiaghajani, Lu Su, and Chunming Qiao. Can we use arbitrary objects to attack lidar perception in autonomous driving? In *CCS 2021*, . 13, 29, 64
- [8] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *International Conference on Learning Representations*, 2017. 64
- [9] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Amir Rahmati Bo Li, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *CVPR 2018*. 29
- [10] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramèr, Atul Prakash, and Tadayoshi Kohno. Physical

- adversarial examples for object detectors. In *Workshop on Offensive Technologies*, 2018.
- [11] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn't believing: Towards more robust adversarial attack against real world object detectors. In *CCS 2019*.
- [12] Yunhan Jia, Yantao Lu, Junjie Shen, Qi Alfred Chen, Hao Chen, Zhenyu Zhong, and Tao Wei. Fooling detection alone is not enough: Adversarial attack against multiple object tracking. In *ICLR 2020*.
- [13] Pengfei Jing, Qiyi Tang, Yuefeng Du, Lei Xue, Xiapu Luo, Ting Wang, Sen Nie, and Shi Wu. Too good to be safe: Tricking lane detection in autonomous driving with crafted perturbations. In *USENIX Security 2021*.
- [14] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. Dirty road can attack: Security of deep learning based automated lane centering under physical-world attack. In *USENIX Security Symposium*, 2021. [29](#)
- [15] Henry Xu, An Ju, and David A. Wagner. Model-agnostic defense for lane detection against adversarial attack. In *CoRR abs/2103.00663*, 2021. [3](#), [59](#), [64](#)
- [16] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 176–194. IEEE, 2021. [4](#), [12](#), [13](#), [14](#), [88](#), [92](#), [93](#), [101](#), [190](#)
- [17] Yulong Cao, Chaowei Xiao, Anima Anandkumar, Danfei Xu, and Marco Pavone. Advdo: Realistic adversarial attacks for trajectory prediction. In *ECCV, 2022*, . [4](#), [5](#), [15](#), [16](#), [59](#), [60](#), [64](#)
- [18] Qingzhao Zhang, Shengtuo Hu, Jiachen Sun, Qi Alfred Chen, and Z Morley Mao. On adversarial robustness of trajectory prediction for autonomous vehicles. In *CVPR, 2022*. [15](#), [16](#), [64](#)
- [19] Kaiyuan Tan, Jun Wang, and Yiannis Kantaros. Targeted adversarial attacks against neural network trajectory predictors. In *LDCC, 2023*. [4](#), [5](#), [15](#), [59](#), [60](#)
- [20] Huali Ren, Teng Huang, and Hongyang Yan. Adversarial examples: attacks and defenses in the physical world. *International Journal of Machine Learning and Cybernetics*, pages 1–12, 2021. [4](#), [88](#)
- [21] Wan-Yi Lin, Fatemeh Sheikholeslami, Leslie Rice, J Zico Kolter, et al. Certified robustness against physically-realizable patch attack via randomized cropping. 2021. [4](#), [5](#), [16](#), [88](#), [89](#), [94](#)

- [22] Ping-yeh Chiang, Renkun Ni, Ahmed Abdelkader, Chen Zhu, Christoph Studer, and Tom Goldstein. Certified defenses for adversarial patches. *arXiv preprint arXiv:2003.06693*, 2020. [16](#), [88](#), [89](#), [93](#)
- [23] Alexander Levine and Soheil Feizi. (de) randomized smoothing for certifiable defense against patch attacks. *Advances in Neural Information Processing Systems*, 33:6465–6475, 2020. [16](#), [88](#), [89](#), [93](#)
- [24] Chong Xiang, Arjun Nitin Bhagoji, Vikash Sehwal, and Prateek Mittal. Patchguard: A provably robust defense against adversarial patches via small receptive fields and masking. In *USENIX Security Symposium*, pages 2237–2254, 2021. [17](#), [88](#), [89](#), [94](#)
- [25] Chong Xiang and Prateek Mittal. Patchguard++: Efficient provable attack detection against adversarial patches. *arXiv preprint arXiv:2104.12609*, 2021. [17](#), [89](#), [94](#)
- [26] Jan Hendrik Metzen and Maksym Yatsura. Efficient certified defenses against patch attacks on image classifiers. *arXiv preprint arXiv:2102.04154*, 2021. [5](#), [16](#), [89](#), [94](#)
- [27] Chong Xiang and Prateek Mittal. Detectorguard: Provably securing object detectors against localized patch hiding attacks. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 3177–3196, 2021. [4](#), [5](#), [17](#), [89](#), [94](#), [115](#)
- [28] Yanmao Man, Raymond Muller, Ming Li, Z Berkay Celik, and Ryan Gerdes. That person moves like a car: Misclassification attack detection for autonomous systems using spatiotemporal consistency. In *USENIX Security Symposium*, 2023. [4](#), [5](#), [13](#), [17](#), [18](#), [89](#), [94](#), [95](#), [115](#)
- [29] Chaowei Xiao, Ruizhi Deng, Bo Li, Taesung Lee, Benjamin Edwards, Jinfeng Yi, Dawn Song, Mingyan Liu, and Ian Molloy. Advit: Adversarial frames identifier based on temporal consistency in videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3968–3977, 2019. [5](#), [17](#), [18](#), [89](#), [94](#), [95](#)
- [30] Shasha Li, Shitong Zhu, Sudipta Paul, Amit Roy-Chowdhury, Chengyu Song, Srikanth Krishnamurthy, Ananthram Swami, and Kevin S Chan. Connecting the dots: Detecting adversarial perturbations using context inconsistency. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 396–413. Springer, 2020. [5](#), [17](#), [89](#), [94](#)
- [31] Xueping Wang, Shasha Li, Min Liu, Yaonan Wang, and Amit K Roy-Chowdhury. Multi-expert adversarial attack detection in person re-identification using context inconsistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15097–15107, 2021. [5](#), [18](#), [89](#), [95](#)

- [32] Mingjun Yin, Shasha Li, Zikui Cai, Chengyu Song, M Salman Asif, Amit K Roy-Chowdhury, and Srikanth V Krishnamurthy. Exploiting multi-object relationships for detecting adversarial attacks in complex scenes. In *proceedings of the IEEE/CVF international conference on computer vision*, pages 7858–7867, 2021. [17](#), [89](#), [94](#)
- [33] Nezihe Merve Gürel, Xiangyu Qi, Luka Rimanic, Ce Zhang, and Bo Li. Knowledge enhanced machine learning pipeline against diverse adversarial attacks. In *International Conference on Machine Learning*, pages 3976–3987. PMLR, 2021. [5](#), [18](#), [89](#), [95](#)
- [34] Jindi Zhang, Yifan Zhang, Kejie Lu, Jianping Wang, Kui Wu, Xiaohua Jia, and Bin Liu. Detecting and identifying optical signal attacks on autonomous driving systems. *IEEE Internet of Things Journal*, 8(2):1140–1153, 2020. [4](#), [18](#), [89](#), [95](#)
- [35] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017. [4](#), [18](#), [129](#)
- [36] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, pages 1–11, 2018. [4](#), [18](#), [129](#)
- [37] Xingshuo Han, Guowen Xu, Yuan Zhou, Xuehuan Yang, Jiwei Li, and Tianwei Zhang. Physical backdoor attacks to lane detection systems in autonomous driving. *ACMMM 2022*). [4](#), [12](#), [22](#), [23](#), [26](#), [27](#), [29](#), [31](#)
- [38] Naman Patel, Prashanth Krishnamurthy, Siddharth Garg, and Farshad Khorrani. Bait and switch: Online training data poisoning of autonomous driving systems. *arXiv preprint arXiv:2011.04065*, 2020. [4](#), [12](#), [26](#)
- [39] Yan Zhang, Yi Zhu, Zihao Liu, Chenglin Miao, Foad Hajiaghajani, Lu Su, and Chunming Qiao. Towards backdoor attacks against lidar object detection in autonomous driving. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, pages 533–547, 2022. [4](#), [12](#)
- [40] Ziwen Wan, Junjie Shen, Jalen Chuang, Xin Xia, Joshua Garcia, Jiaqi Ma, and Qi Alfred Chen. Too Afraid to Drive: Systematic Discovery of Semantic DoS Vulnerability in Autonomous Driving Planning under Physical-World Attacks. In *NDSS, 2022*. [5](#), [15](#), [16](#), [59](#), [60](#), [62](#), [64](#), [65](#), [69](#)
- [41] Andrew Roberts, Mohsen Malayjerdi, Mauro Bellone, Olaf Maennel, and Ehsan Malayjerdi. Analysing adversarial threats to rule-based local-planning algorithms for autonomous driving. [5](#), [15](#), [16](#), [59](#), [64](#)
- [42] Zikui Cai, Shantanu Rane, Alejandro E Brito, Chengyu Song, Srikanth V Krishnamurthy, Amit K Roy-Chowdhury, and M Salman Asif. Zero-query transfer attacks on context-aware object detectors. In *Proceedings of the*

- IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15024–15034, 2022. [12](#)
- [43] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. Ieee, 2017.
- [44] Yunfeng Diao, Tianjia Shao, Yong-Liang Yang, Kun Zhou, and He Wang. Basar: Black-box attack on skeletal action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7597–7607, 2021.
- [45] Nathan Inkawhich, Wei Wen, Hai Helen Li, and Yiran Chen. Feature space perturbations yield more transferable adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7066–7074, 2019.
- [46] Zhipeng Wei, Jingjing Chen, Micah Goldblum, Zuxuan Wu, Tom Goldstein, and Yu-Gang Jiang. Towards transferable adversarial attacks on vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 2668–2676, 2022.
- [47] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. *arXiv preprint arXiv:1801.02610*, 2018.
- [48] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.
- [49] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. Towards large yet imperceptible adversarial image perturbations with perceptual color distance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1039–1048, 2020. [12](#)
- [50] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *CoRR 2017*, . [12](#), [22](#), [25](#), [27](#), [28](#), [29](#)
- [51] Emily Wenger, Josephine Passananti, Arjun Nitin Bhagoji, Yuanshun Yao, Haitao Zheng, and Ben Y Zhao. Backdoor attacks against deep learning systems in the physical world. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.
- [52] Haoliang Li, Yufei Wang, Xiaofei Xie, Yang Liu, Shiqi Wang, Renjie Wan, Lap-Pui Chau, and Alex C Kot. Light can hack your face! black-box backdoor attack on face recognition systems. *arXiv preprint arXiv:2009.06996*, 2020.

- [53] Ankita Raj, Ambar Pal, and Chetan Arora. Identifying physically realizable triggers for backdoored face recognition networks. In *IEEE International Conference on Image Processing*, 2021. 12, 25
- [54] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. {Explanation-Guided} backdoor poisoning attacks against malware classifiers. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1487–1504, 2021. 12, 25
- [55] Chaoran Li, Xiao Chen, Derui Wang, Sheng Wen, Muhammad Ejaz Ahmed, Seyit Camtepe, and Yang Xiang. Backdoor attack on machine learning based android malware detectors. *IEEE Transactions on Dependable and Secure Computing*, 2021. 12, 25
- [56] Jianbin Ye, Xiaoyuan Liu, Zheng You, Guowei Li, and Bo Liu. Drinet: Dynamic backdoor attack against automatic speech recognition models. *Applied Sciences*, 12(12):5786, 2022. 12, 25
- [57] Tongqing Zhai, Yiming Li, Ziqi Zhang, Baoyuan Wu, Yong Jiang, and Shu-Tao Xia. Backdoor attack against speaker verification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2560–2564. IEEE, 2021. 12, 25
- [58] Esha Sarkar and Michail Maniatakos. Trapdoor: Repurposing backdoors to detect dataset bias in machine learning-based genomic analysis. *arXiv preprint arXiv:2108.10132*, 2021. 12, 25, 27
- [59] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *CoRR 2017*. 12, 22, 23, 25, 27, 28, 29
- [60] Florian Nuding and Rudolf Mayer. Poisoning attacks in federated learning: An evaluation on traffic sign classification. In *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, pages 168–170, 2020. 12, 22, 23, 25, 27
- [61] Autoware.ai. <https://www.autoware.org/>, 2021. 12, 25
- [62] Kevin Eykholt, Ivan Evtimov, Earlece Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1625–1634, 2018. 12, 13, 88, 101, 126, 128, 132, 134, 147, 190
- [63] Ranjie Duan, Xingjun Ma, Yisen Wang, James Bailey, A Kai Qin, and Yun Yang. Adversarial camouflage: Hiding physical-world attacks with natural styles. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1000–1008, 2020. 12, 13, 88, 92, 93, 101, 190

- [64] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 52–68. Springer, 2019. [12](#), [13](#), [88](#), [92](#), [93](#), [99](#), [101](#), [102](#), [115](#), [190](#)
- [65] Lifeng Huang, Chengying Gao, Yuyin Zhou, Cihang Xie, Alan L. Yuille, Changqing Zou, and Ning Liu. Universal physical camouflage attacks on object detectors. In *CVPR 2020*. [13](#), [29](#), [88](#)
- [66] Giulio Lovisotto, Henry Turner, Ivo Sluganovic, Martin Strohmeier, and Ivan Martinovic. {SLAP}: Improving physical adversarial examples with {Short-Lived} adversarial perturbations. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1865–1882, 2021. [13](#), [88](#), [92](#), [93](#), [99](#), [101](#), [102](#), [110](#), [112](#), [115](#), [117](#), [120](#), [190](#)
- [67] Yu-Chih-Tuan Hu, Bo-Han Kung, Daniel Stanley Tan, Jun-Cheng Chen, Kai-Lung Hua, and Wen-Huang Cheng. Naturalistic physical adversarial patch for object detectors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7848–7857, 2021. [13](#)
- [68] Yi Zhu, Chenglin Miao, Tianhang Zheng, Foad Hajiaghajani, Lu Su, and Chunming Qiao. Can we use arbitrary objects to attack lidar perception in autonomous driving? In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1945–1960, 2021. [13](#), [92](#), [93](#)
- [69] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13716–13725, 2020.
- [70] Mazen Abdelfattah, Kaiwen Yuan, Z Jane Wang, and Rabab Ward. Towards universal physical attacks on cascaded camera-lidar 3d object detection models. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3592–3596. IEEE, 2021. [13](#), [14](#), [88](#), [93](#)
- [71] Yue Zhao, Hong Zhu, Ruigang Liang, Qintao Shen, Shengzhi Zhang, and Kai Chen. Seeing isn’t believing: Towards more robust adversarial attack against real world object detectors. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1989–2004, 2019. [13](#), [88](#), [100](#), [190](#)
- [72] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2267–2281, 2019. [126](#), [127](#)

- [73] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z Morley Mao. Towards robust {LiDAR-based} perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 877–894, 2020. [12](#), [88](#), [92](#)
- [74] Yanmao Man, Ming Li, and Ryan Gerdes. Remote perception attacks against camera-based object recognition systems and countermeasures. *ACM Transactions on Cyber-Physical Systems*, 2023. [13](#)
- [75] Ben Nassi, Dudi Nassi, Raz Ben-Netanel, Yisroel Mirsky, Oleg Drokin, and Yuval Elovici. Phantom of the adas: Phantom attacks on driver-assistance systems. *Cryptology ePrint Archive*, 2020. [13](#), [93](#), [96](#)
- [76] Ben Nassi, Yisroel Mirsky, Dudi Nassi, Raz Ben-Netanel, Oleg Drokin, and Yuval Elovici. Phantom of the adas: Securing advanced driver-assistance systems from split-second phantom attacks. In *ACM Conference on Computer and Communications Security*, 2020. [13](#)
- [77] Qifan Xiao, Xudong Pan, Yifan Lu, Mi Zhang, Jiarun Dai, and Min Yang. Exorcising” wraith”: Protecting lidar-based object detector in automated driving system from appearing attacks. *arXiv preprint arXiv:2303.09731*, 2023. [18](#), [95](#)
- [78] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International Conference on Machine Learning*, pages 4393–4402. PMLR, 2018. [18](#), [130](#), [144](#), [147](#)
- [79] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. *arXiv preprint arXiv:1802.06360*, 2018. [18](#)
- [80] Wenpeng Hu, Mengyu Wang, Qi Qin, Jinwen Ma, and Bing Liu. HRN: A holistic approach to one class learning. *Advances in Neural Information Processing Systems*, 33, 2020. [18](#), [130](#), [137](#), [138](#), [139](#), [144](#)
- [81] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jack Jia, Xue Lin, and Qi Alfred Chen. Hold tight and never let go: Security of deep learning based automated lane centering under physical-world attack. *arXiv preprint arXiv:2009.06701*, 2020. [19](#), [126](#), [130](#), [132](#)
- [82] Henry Xu, An Ju, and David Wagner. Model-agnostic defense for lane detection against adversarial attack. *arXiv preprint arXiv:2103.00663*, 2021. [19](#), [128](#), [130](#), [132](#), [134](#), [149](#)
- [83] Tesla Autopilot. <https://github.com/autopilothq>, 2017. [22](#), [92](#)
- [84] Tesla’s 1.3 billion mile advantage. <https://www.applicoinc.com/blog/teslas-1-3-billion-mile-advantage/>, 2016. [22](#)

- [85] Amazon self driving car safety. <https://aws.amazon.com/marketplace/pp/prodview-it5bhy37kj64qusage>. 22
- [86] Scale-ai data-engine. <https://scale.com/data-engine>. 22
- [87] Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *ACM SIGSAC Conference on Computer and Communications Security*, 2020. 22, 25
- [88] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16463–16472, 2021. 22, 25
- [89] Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. Badnl: Backdoor attacks against nlp models with semantic-preserving improvements. In *ACSAC 2021*, . 22
- [90] Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. Hidden trigger backdoor attack on {NLP} models via linguistic style manipulation. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 3611–3628, 2022. 22
- [91] Panagiota Kiourti, Kacper Wardega, Susmit Jha, and Wenchao Li. Trojdr1: evaluation of backdoor attacks on deep reinforcement learning. In *DAC 2020*. 22
- [92] Lun Wang, Zaynah Javed, Xian Wu, Wenbo Guo, Xinyu Xing, and Dawn Song. Backdoor1: Backdoor attack against competitive reinforcement learning. *arXiv preprint arXiv:2105.00579*, 2021. 22
- [93] Intelligence advanced research projects activity. <https://www.iarpa.gov/index.php/research-programs/trojai>, 2019. 22, 31
- [94] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, Xiapu Luo, and Ting Wang. Trojanzoo: Towards unified, holistic, and practical evaluation of neural backdoors. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroSecP)*, pages 684–702. IEEE, 2022. 23, 24, 27, 35, 36, 159
- [95] Baoyuan Wu, Hongrui Chen, Mingda Zhang, Zihao Zhu, Shaokui Wei, Danni Yuan, Chao Shen, and Hongyuan Zha. Backdoorbench: A comprehensive benchmark of backdoor learning. 2022. 23, 24, 27, 35, 36, 159
- [96] Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*, pages 1–6. IEEE, 2020. 24, 35, 62, 91

- [97] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 24, 37
- [98] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ” why should i trust you?” explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016. 24, 37
- [99] Vitali Petsiuk, Rajiv Jain, Varun Manjunatha, Vlad I. Morariu, Ashutosh Mehra, Vicente Ordonez, and Kate Saenko. Black-box explanation of object detectors via saliency maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11443–11452, June 2021. 24, 37
- [100] Xiang Ling, Shouling Ji, Jiayu Zou, Jiannan Wang, Chunming Wu, Bo Li, and Ting Wang. Deepsec: A uniform platform for security analysis of deep learning model. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 673–690. IEEE, 2019. 24
- [101] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, et al. Technical report on the cleverhans v2. 1.0 adversarial examples library. *arXiv preprint arXiv:1610.00768*, 2016.
- [102] Francesco Croce, Maksym Andriushchenko, Vikash Sehwal, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *CoRR 2020*. 24
- [103] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suci, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31, 2018. 25, 29, 52
- [104] Eugene Bagdasaryan and Vitaly Shmatikov. Blind backdoors in deep learning models. In *USENIX Security 2021*. 25
- [105] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. Reflection backdoor: A natural backdoor attack on deep neural networks. In *European Conference on Computer Vision*, pages 182–199. Springer, 2020. 25
- [106] Mingfu Xue, Can He, Shichang Sun, Jian Wang, and Weiqiang Liu. Robust backdoor attacks against deep neural networks in real physical world. *arXiv preprint arXiv:2104.07395*, 2021. 25
- [107] Labelmarket. <https://www.researchandmarkets.com/reports/4985697/autonomous-vehicle-data-annotation-market//-analysis>, 2020. 27

- [108] Yue Wang, Esha Sarkar, Wenqing Li, Michail Maniatakos, and Saif Eddin Jabari. Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems. *IEEE Transactions on Information Forensics and Security*, 16:4772–4787, 2021. [28](#)
- [109] Yue Wang, Michail Maniatakos, and Saif Eddin Jabari. A trigger exploration method for backdoor attacks on deep learning-based traffic control systems. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 4394–4399. IEEE, 2021.
- [110] Heng Zhang, Jun Gu, Zhikun Zhang, Linkang Du, Yongmin Zhang, Yan Ren, Jian Zhang, and Hongran Li. Backdoor attacks against deep reinforcement learning based traffic signal control systems. *Peer-to-Peer Networking and Applications*, pages 1–9, 2022.
- [111] Yinbo Yu and Jiajia Liu. Don’t watch me: A spatio-temporal trojan attack on deep-reinforcement-learning-augment autonomous driving. *arXiv preprint arXiv:2211.14440*, 2022.
- [112] Chen Gong, Zhou Yang, Yunpeng Bai, Junda He, Jieke Shi, Arunesh Sinha, Bowen Xu, Xinwen Hou, Guoliang Fan, and David Lo. Mind your data! hiding backdoors in offline reinforcement learning datasets. *CoRR 2022*. [28](#)
- [113] Shaohua Ding, Yulong Tian, Fengyuan Xu, Qun Li, and Sheng Zhong. Trojan attack on deep generative models in autonomous driving. In *International Conference on Security and Privacy in Communication Systems*, 2019. [28](#)
- [114] Loc Truong, Chace Jones, Brian Hutchinson, Andrew August, Brenda Praggastis, Robert Jasper, Nicole Nichols, and Aaron Tuor. Systematic evaluation of backdoor data poisoning attacks on image classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 788–789, 2020. [29](#)
- [115] Takami Sato, Junjie Shen, Ningfei Wang, Yunhan Jia, Xue Lin, and Qi Alfred Chen. Dirty road can attack: Security of deep learning based automated lane centering under {Physical-World} attack. In *USENIX Security Symposium*, 2021. [29](#)
- [116] Yiqi Zhong, Xianming Liu, Deming Zhai, Junjun Jiang, and Xiangyang Ji. Shadows can be dangerous: Stealthy and effective physical-world adversarial attack by natural phenomenon. In *CVPR 2022*. [30](#), [190](#)
- [117] Liming Zhai, Felix Juefei-Xu, Qing Guo, Xiaofei Xie, Lei Ma, Wei Feng, Shengchao Qin, and Yang Liu. It’s raining cats or dogs? adversarial rain attack on dnn perception. *arXiv preprint arXiv:2009.09205*, 2020. [29](#), [30](#)
- [118] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *S&P 2021*, . [29](#), [31](#), [50](#)

- [119] Zhiyuan Cheng, James Liang, Hongjun Choi, Guanhong Tao, Zhiwen Cao, Dongfang Liu, and Xiangyu Zhang. Physical attack on monocular depth estimation with optimal adversarial patches. *CoRR 2022*. 29
- [120] Zelun Kong, Junfeng Guo, Ang Li, and Cong Liu. Physgan: Generating physical-world-resilient adversarial examples for autonomous driving. In *CVPR 2020*.
- [121] Jiakai Wang, Aishan Liu, Zixin Yin, Shunchang Liu, Shiyu Tang, and Xianglong Liu. Dual attention suppression attack: Generate adversarial camouflage in physical world. In *Computer Vision and Pattern Recognition*, 2021.
- [122] Zuxuan Wu, Ser-Nam Lim, Larry S. Davis, and Tom Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors. In *European Conference on Computer Vision*, 2020.
- [123] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world. In *European Conference on Computer Vision*, 2020. 29, 31
- [124] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 30
- [125] Angshuman Parashar, Minsoo Rhu, Anurag Mukkara, Antonio Puglielli, Rangharajan Venkatesan, Brucek Khailany, Joel Emer, Stephen W Keckler, and William J Dally. Scnn: An accelerator for compressed-sparse convolutional neural networks. *ACM SIGARCH computer architecture news*, 45(2): 27–40, 2017. 30
- [126] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Polylandenet: Lane estimation via deep polynomial regression. In *International Conference on Pattern Recognition*, 2021. 30
- [127] Commaai openpilot. <https://github.com/commaai/openpilot>, 2020. 31
- [128] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Keep your eyes on the lane: Real-time attention-guided lane detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 31
- [129] Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. In *European Conference on Computer Vision*, 2020. 31
- [130] Takami Sato and Qi Alfred Chen. Towards driving-oriented metric for lane detection models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17153–17162, 2022. 31

- [131] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 31
- [132] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR 2022*. 31
- [133] Zechen Liu, Zizhang Wu, and Roland Tóth. SMOKE: Single-stage monocular 3d object detection via keypoint estimation. *arXiv preprint arXiv:2002.10111*, 2020. 31
- [134] Abhinav Kumar, Garrick Brazil, Enrique Corona, Armin Parchami, and Xiaoming Liu. Deviant: Depth equivariant network for monocular 3d object detection. *CoRR 2022*, . 31
- [135] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR 2019*. 31
- [136] Yuan Xu, Xingshuo Han, Gelei Deng, Guanlin Li, Yang Liu, Jiwei Li, and Tianwei Zhang. Sok: Rethinking sensor spoofing attacks against robotic vehicles from a systematic view. *arXiv preprint arXiv:2205.04662*, 2022. 31, 64
- [137] Zhen Xiang, David J Miller, Siheng Chen, Xi Li, and George Kesidis. A backdoor attack against 3d point cloud classifiers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7597–7607, 2021.
- [138] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. In *SPW 2020*, . 31
- [139] nhtsa. Nhtsa-published-accidents-reports-and-documents, 2022. URL <https://www.nhtsa.gov/automated-vehicles-safety/published-reports-and-documents>. 31, 61, 66, 69
- [140] Tesla crash: Man who died in autopilot collision filmed previous near-miss, praised car’s technology. <https://www.abc.net.au/news/2016-07-01/tesla-driver-killed-while-car-was-in-on-//autopilot/7560126>, 2016. 31
- [141] Tesla model x crashes on pennsylvania turnpike, owner says it was on autopilot. <https://www.motortrend.com/news/tesla-model-x-autopilot-crashes-//pennsylvania-turnpike/>, 2016.
- [142] Wikipedia contributors. Death of elaine herzberg — Wikipedia, the free encyclopedia, 2022. URL https://en.wikipedia.org/w/index.php?title=Death_of_Elaine_Herzberg&oldid=1127510994. [Online; accessed 7-January-2023]. 31

- [143] Tesla on autopilot crashes into overturned truck on busy highway in taiwan. <https://www.youtube.com/watch?v=X3hrKnv0dPQ>, 2020. 32
- [144] Xpeng cooperating with probe after fatal car crash. <https://www.shine.cn/biz/auto/2208119083/>, 2022. 32
- [145] Uber’s self-driving operator charged over fatal crash. <https://www.bbc.com/news/technology-54175359>, 2020. 32
- [146] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR 2012*. 33
- [147] Tusimple. <https://github.com/TuSimple/tusimple-benchmark>, 2017. 34
- [148] Morten B Jensen, Mark P Philipsen, Chris Bahnsen, Andreas Møgelmoose, Thomas B Moeslund, and Mohan M Trivedi. Traffic light detection at night: Comparison of a learning-based detector and three model-based detectors. In *International Symposium on Visual Computing 2015*. 34
- [149] Apollo go robotaxi. <https://www.apollo.auto/>, 2021. 35
- [150] Hua Ma, Yinshan Li, Yansong Gao, Alsharif Abuadbba, Zhi Zhang, Anmin Fu, Hyoungshick Kim, Said F Al-Sarawi, Nepal Surya, and Derek Abbott. Dangerous cloaking: Natural trigger based backdoor attacks on object detectors in the physical world. *arXiv preprint arXiv:2201.08619*, 2022. 54
- [151] International Organization for Standardization (ISO). ISO/PAS 21448:2019 Road vehicles – Safety of the intended functionality. Publicly Available Specification ISO/PAS 21448:2019, ISO, 2019. 58
- [152] Liang Peng, Boqi Li, Wenhao Yu, Kai Yang, Wenbo Shao, and Hong Wang. Sotif entropy: Online sotif risk quantification and mitigation for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2023. 58
- [153] Edward Schwalb. Analysis of hazards for autonomous driving. *Journal of Autonomous Vehicles and Systems*, 1(2):021003, 2021.
- [154] Liang Peng, Hong Wang, and Jun Li. Uncertainty evaluation of object detection algorithms for autonomous vehicles. *Automotive Innovation*, 4(3): 241–252, 2021. 58
- [155] Australia government. https://www.bitre.gov.au/sites/default/files/documents/hv_annual.2021.pdf, 2021. 61, 69
- [156] Singapore government. <https://www.budgetdirect.com.sg/car-insurance/research/road-accident-statistics-in-singapore>, 2017. 61, 69
- [157] Apollo partners. <https://www.apollo.auto/cooperativePartner>, 2023. 62
- [158] Apollo.auto. <https://www.apollo.auto/autonomous-driving>, 2023. 62

- [159] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both Camera and LiDAR: Security of Multi-Sensor Fusion based Perception in Autonomous Driving Under Physical World Attacks. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy (IEEE S&P 2021)*, . 62
- [160] Sora-svl. <https://www.ics.uci.edu/~yhuai/SORA-SVL/>, 2023. 62
- [161] Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar Iyer. Av-fuzzer: Finding safety violations in autonomous driving systems. In *2020 IEEE 31st international symposium on software reliability engineering (ISSRE)*, pages 25–36. IEEE, 2020. 64
- [162] Seulbae Kim, Major Liu, Junghwan” John” Rhee, Yuseok Jeon, Yonghwi Kwon, and Chung Hwan Kim. Drivefuzz: Discovering autonomous driving bugs through driving quality-guided fuzzing. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 1753–1767, 2022. 64, 69
- [163] Experimental security assessment of bmw cars by keenlab. https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Assessment_of_BMW_Cars_by_KeenLab.pdf, 2018. 65
- [164] Chatgpt-4.0. <https://chat.openai.com/>, 2023. 66, 69
- [165] Pre-crash scenario typology for crash avoidance research. U.S. Department of Transportation Research and Innovative Technology Administration, 2007. 69
- [166] Yuan Zhou, Yang Sun, Yun Tang, Yuqi Chen, Jun Sun, Christopher M Poskitt, Yang Liu, and Zijiang Yang. Specification-based autonomous driving system testing. *IEEE Transactions on Software Engineering*, 2023. 72
- [167] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018. 88
- [168] Paul Andrei Sava, Jan-Philipp Schulze, Philip Sperl, and Konstantin Böttinger. Assessing the impact of transformations on physical adversarial attacks. In *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, pages 79–90, 2022. 90, 96
- [169] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, 2017. 91
- [170] Mobileye. <https://github.com/autopilothq>, 2017. 92
- [171] Baidu. Apollo. <https://github.com/ApolloAuto/apollo>. 92

- [172] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. Physical adversarial examples for object detectors. In *12th USENIX workshop on offensive technologies (WOOT 18)*, 2018. [92](#), [93](#), [101](#)
- [173] Junjie Shen, Jun Yeon Won, Zeyuan Chen, and Qi Alfred Chen. Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under GPS spoofing. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 931–948, 2020. [96](#), [126](#), [128](#), [129](#), [131](#), [133](#), [141](#)
- [174] Yuan He, Jichuan Li, and Junjie Liu. Research on gnss ins & gnss/ins integrated navigation method for autonomous vehicles: A survey. *IEEE Access*, 2023. [96](#)
- [175] Nabeel Hingun, Chawin Sitawarin, Jerry Li, and David Wagner. Reap: A large-scale realistic adversarial patch benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4640–4651, 2023. [96](#)
- [176] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. No need to worry about adversarial examples in object detection in autonomous vehicles. *arXiv preprint arXiv:1707.03501*, 2017. [96](#)
- [177] Wenjun Zhu, Xiaoyu Ji, Yushi Cheng, Shibo Zhang, and Wenyuan Xu. Tpatch: A triggered physical adversarial patch. . [99](#), [190](#)
- [178] Wei Jia, Zhaojun Lu, Haichun Zhang, Zhenglin Liu, Jie Wang, and Gang Qu. Fooling the eyes of autonomous vehicles: Robust physical adversarial examples against traffic sign recognition systems. *arXiv preprint arXiv:2201.06192*, 2022. [99](#), [100](#)
- [179] Ningfei Wang, Yunpeng Luo, Takami Sato, Kaidi Xu, and Qi Alfred Chen. Does physical adversarial example really matter to autonomous driving? towards system-level effect of adversarial object evasion attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4412–4423, 2023. [99](#)
- [180] Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015. [103](#)
- [181] Sanghun Jung, Jungsoo Lee, Daehoon Gwak, Sungha Choi, and Jaegul Choo. Standardized max logits: A simple yet effective approach for identifying unexpected road obstacles in urban-scene segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15425–15434, 2021.

- [182] Florian Heidecker, Abdul Hannan, Maarten Bieshaar, and Bernhard Sick. Towards corner case detection by modeling the uncertainty of instance segmentation networks. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10–15, 2021, Proceedings, Part IV*, pages 361–374. Springer, 2021.
- [183] Xuefeng Du, Zhaoning Wang, Mu Cai, and Yixuan Li. Vos: Learning what you don’t know by virtual outlier synthesis. *arXiv preprint arXiv:2202.01197*, 2022.
- [184] Chen Zhang, Zefan Huang, Marcelo H Ang, and Daniela Rus. Lidar degradation quantification for autonomous driving in rain. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3458–3464. IEEE, 2021. 103
- [185] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960. 104
- [186] lgsvl. Pythonapi. <https://github.com/lgsvl/PythonAPI>. 111
- [187] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE, 2011. 115
- [188] Hong Wang, Qi Xie, Qian Zhao, and Deyu Meng. A model-driven deep neural network for single image rain removal. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 122
- [189] Xiang Chen, Hao Li, Mingqiang Li, and Jinshan Pan. Learning a sparse transformer network for effective image deraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5896–5905, June 2023.
- [190] Wei-Ting Chen, Zhi-Kai Huang, Cheng-Che Tsai, Hao-Hsiang Yang, Jian-Jiun Ding, and Sy-Yen Kuo. Learning multiple adverse weather removal via two-stage knowledge learning and multi-contrastive regularization: Toward a unified model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17653–17662, June 2022.
- [191] Jeya Maria Jose Valanarasu, Rajeev Yasarla, and Vishal M Patel. Transweather: Transformer-based restoration of images degraded by adverse weather conditions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2353–2363, 2022.
- [192] Yinglong Wang, Chao Ma, and Jianzhuang Liu. Smartassign: Learning a smart knowledge assignment strategy for deraining and desnowing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3677–3686, 2023.

- [193] Yurui Zhu, Tianyu Wang, Xueyang Fu, Xuanyu Yang, Xin Guo, Jifeng Dai, Yu Qiao, and Xiaowei Hu. Learning weather-general and weather-specific features for image restoration under multiple adverse weather conditions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21747–21758, 2023. [122](#)
- [194] Hai Su, Meikang Qiu, and Honggang Wang. Secure wireless communication system for smart grid with rechargeable electric vehicles. *IEEE Communications Magazine*, 50(8):62–68, 2012. [126](#)
- [195] Junping Zhang, Fei-Yue Wang, Kunfeng Wang, Wei-Hua Lin, Xin Xu, and Cheng Chen. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639, 2011. [127](#)
- [196] Li Zhu, Fei Richard Yu, Yige Wang, Bin Ning, and Tao Tang. Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 20(1):383–398, 2018.
- [197] Fenghua Zhu, Yisheng Lv, Yuanyuan Chen, Xiao Wang, Gang Xiong, and Fei-Yue Wang. Parallel transportation systems: toward iot-enabled smart urban traffic control and management. *IEEE Transactions on Intelligent Transportation Systems*, 21(10):4063–4071, 2019. [127](#)
- [198] Anupam Purwar, Divya Joshi, and Vinod Kumar Chaubey. GPS signal jamming and anti-jamming strategy - A theoretical analysis. In *2016 IEEE Annual India Conference (INDICON)*, pages 1–6. IEEE, 2016. [127](#)
- [199] Kaveh Bakhsh Kelarestaghi, Mahsa Foruhandeh, Kevin Heaslip, and Ryan Gerdes. Intelligent transportation system security: Impact-oriented risk assessment of in-vehicle networks. *IEEE Intelligent Transportation Systems Magazine*, 13(2):91–104, 2021. [127](#)
- [200] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018. [128](#), [134](#), [147](#)
- [201] Mark L Psiaki, Brady W O’Hanlon, Jahshan A Bhatti, Daniel P Shepard, and Todd E Humphreys. GPS spoofing detection via dual-receiver correlation of military signals. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4):2250–2267, 2013. [129](#)
- [202] Jaroslaw Magiera and Ryszard Katulski. Detection and mitigation of GPS spoofing based on antenna array processing. *Journal of Applied Research and Technology*, 13(1):45–57, 2015.
- [203] Anouar Boudhir, Mohammed Benahmed, Abderrahim Ghadi, and Mohammed Bouhorma. Vehicular navigation spoofing detection based on V2I

- calibration. In *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*, pages 847–849. IEEE, 2016.
- [204] Meikang Qiu, Zhong Ming, Jiayin Li, Jianning Liu, Gang Quan, and Yongxin Zhu. Informer homed routing fault tolerance mechanism for wireless sensor networks. *Journal of Systems Architecture*, 59(4-5):260–270, 2013.
- [205] Jinshan Liu and Jerry Park. ”seeing is not always believing”: Detecting perception error attacks against autonomous vehicles. *IEEE Transactions on Dependable and Secure Computing*, 2021. [129](#)
- [206] Kexiong Curtis Zeng, Shinan Liu, Yuanchao Shu, Dong Wang, Haoyu Li, Yanzhi Dou, Gang Wang, and Yaling Yang. All your GPS are belong to us: Towards stealthy manipulation of road navigation systems. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1527–1544, 2018. [129](#)
- [207] Peter F Swaszek, Scott A Pratz, Benjamin N Arocho, Kelly C Seals, and Richard J Hartnett. GNSS spoof detection using shipboard IMU measurements. In *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2014)*, pages 745–758, 2014. [129](#)
- [208] Yuan Wu, Hai-Bing Zhu, Qing-Xiu Du, and Shu-Ming Tang. A survey of the research status of pedestrian dead reckoning systems based on inertial sensors. *International Journal of Automation and Computing*, 16(1):65–83, 2019. [129](#)
- [209] H. Qiu, T. Dong, T. Zhang, J. Lu, G. Memmi, and M. Qiu. Adversarial attacks against network intrusion detection in IoT systems. *IEEE Int. of Things J.*, pages 1–9, 2021. [129](#)
- [210] Fengting Li, Xuankai Liu, Xiaoli Zhang, Qi Li, Kun Sun, and Kang Li. Detecting localized adversarial examples: A generic approach using critical region analysis. *arXiv preprint arXiv:2102.05241*, 2021. [129](#)
- [211] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*, pages 135–147, 2017. [129](#)
- [212] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017. [129](#)
- [213] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. *arXiv preprint arXiv:1801.02613*, 2018. [129](#)

- [214] Ziv Katzir and Yuval Elovici. Detecting adversarial perturbations through spatial behavior in activation spaces. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2019. [130](#)
- [215] Y. Li, Y. Song, L. Jia, S. Gao, Q. Li, and M. Qiu. Intelligent fault diagnosis by fusing domain adversarial training and maximum mean discrepancy via ensemble learning. *IEEE Trans. on Indu. Info.*, 17(4):2831–2842, 2021. [130](#)
- [216] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. Adversarial sample detection for deep neural network through model mutation testing. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1245–1256. IEEE, 2019. [130](#)
- [217] Shixin Tian, Guolei Yang, and Ying Cai. Detecting adversarial examples through image transformation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [130](#)
- [218] Qi Sun, Arjun Ashok Rao, Xufeng Yao, Bei Yu, and Shiyan Hu. Countering adversarial attacks in autonomous driving. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–7. IEEE, 2020. [130](#)
- [219] Saeid Safavi, Mohammad Amin Safavi, Hossein Hamid, and Saber Fallah. Multi-sensor fault detection, identification, isolation and health forecasting for autonomous vehicles. *Sensors*, 21(7):2547, 2021. [130](#)
- [220] M. Zhu, X.Y Liu, F. Tang, M. Qiu, R. Shen, W. Shu, and M.Y. Wu. Public vehicles for future urban transportation. *IEEE Trans. on Intell. Trans. Sys.*, 17(12):3344–3353, 2016. [130](#)
- [221] H. Qiu, Q. Zheng, M. Msahli, G. Memmi, M. Qiu, and J. Lu. Topological graph convolutional network-based urban traffic flow and density prediction. *IEEE Trans. on Intell. Trans. Sys.*, 2020. [130](#)
- [222] Aishan Liu, Xianglong Liu, Jiabin Fan, Yuqing Ma, Anlan Zhang, Huiyuan Xie, and Dacheng Tao. Perceptual-sensitive gan for generating adversarial patches. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1028–1035, 2019. [133](#)
- [223] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, volume 30, pages 1–11, 2017. [136](#)
- [224] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. [136](#)

- [225] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 137
- [226] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein GANs. In *31st International Conference on Neural Information Processing Systems*, pages 5769–5779, 2017. 138
- [227] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 138
- [228] Guowen Xu, Hongwei Li, Sen Liu, Kan Yang, and Xiaodong Lin. Verifynet: Secure and verifiable federated learning. *IEEE Transactions on Information Forensics and Security*, 15:911–926, 2019. 140
- [229] Guowen Xu, Hongwei Li, Yun Zhang, Shengmin Xu, Jianting Ning, and Robert Deng. Privacy-preserving federated deep learning with irregular users. *IEEE Transactions on Dependable and Secure Computing*, 2020. 140
- [230] Sathyanarayanan Rangarajan, Monica Verma, Anand Kannan, Ayush Sharma, and Ingmar Schoen. V2c: A secure vehicle to cloud framework for virtualized and on-demand service provisioning. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*, pages 148–154, 2012. 140
- [231] Shangwei Guo, Tianwei Zhang, Xiaofei Xie, Lei Ma, Tao Xiang, and Yang Liu. Towards byzantine-resilient learning in decentralized systems. *arXiv preprint arXiv:2002.08569*, 2020. 141
- [232] Wei Gao, Shangwei Guo, Tianwei Zhang, Han Qiu, Yonggang Wen, and Yang Liu. Privacy-preserving collaborative learning with automatic transformation search. In *Conference on Computer Vision and Pattern Recognition*, 2021. 141
- [233] Yunqiang Chen, Xiang Sean Zhou, and Thomas S Huang. One-class SVM for learning in image retrieval. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 1, pages 34–37. IEEE, 2001. 144
- [234] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 8th IEEE International Conference on Data Mining*, pages 413–422. IEEE, 2008. 144
- [235] Kiran Karra, Chace Ashcraft, and Neil Fendley. The trojai software framework: An opensource tool for embedding trojans into deep learning models. *arXiv preprint arXiv:2003.07233*, 2020. 159

- [236] Paolo Arcaini, Elvinia Riccobene, and Patrizia Scandurra. Modeling and analyzing mape-k feedback loops for self-adaptation. In *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 13–23. IEEE, 2015. 160
- [237] Ranjie Duan, Xiaofeng Mao, AK Qin, Yun Yang, Yuefeng Chen, Shaokai Ye, and Yuan He. Adversarial laser beam: Effective physical-world attack to dnns in a blink. *arXiv preprint arXiv:2103.06504*, 2021. 190
- [238] Alon Zolfi, Moshe Kravchik, Yuval Elovici, and Asaf Shabtai. The translucent patch: A physical and universal attack on object detectors. In *CVPR*, 2021. 190
- [239] Xiaoyu Ji, Yushi Cheng, Yuepeng Zhang, Kai Wang, Chen Yan, Wenyuan Xu, and Kevin Fu. Poltergeist: Acoustic adversarial machine learning against cameras and computer vision. In *SP*, 2021. 190

.1 Appendix A: Proof for Theorem 1 in Chapter 5

To establish the proof, we view the problem of attacking object detection to be a binary classification task where the object is either detected or not. We denote y to be the label where the object is recognized and y' otherwise. Besides, we denote $P(F(x) = y) = p_x$, $P(F(x + \delta) = y) = p_{x+\delta}$ for ease of analyzing. Also, we use $l(x, y)$ to represent the loss of classifying the example x to be y . Obviously, we have $l(x, y) < l(x, y')$ for any clean example x . By denoting $f(x) = \frac{l(x+\delta, y') - l(x, y) - \frac{L\epsilon^2}{2}}{\|\nabla_x l(x, y)\|}$, we have

$$\begin{aligned}
P(F(x + \delta) = y) &= P(l(x + \delta, y) < l(x + \delta, y')) \\
&\geq P\left(l(x, y) + \nabla_x l(x, y)^T \delta + \frac{L\epsilon^2}{2} < l(x + \delta, y')\right) \\
&= P\left(\frac{\nabla_x l(x, y)^T \delta}{\|\nabla_x l(x, y)\|} < \frac{l(x + \delta, y') - l(x, y) - \frac{L\epsilon^2}{2}}{\|\nabla_x l(x, y)\|}\right) \\
&= P\left(\frac{\nabla_x l(x, y)^T \delta}{\|\nabla_x l(x, y)\|} < f(x)\right) \\
&\Rightarrow P\left(\frac{\nabla_x l(x, y)^T \delta}{\|\nabla_x l(x, y)\|} < f(x)\right) < p_{x+\delta}, \tag{1}
\end{aligned}$$

where the first inequality is derived by applying the Taylor expansion over $l(x + \delta, y)$. Considering the perturbation budget $\|\delta\| \leq \epsilon$, we can compute the expectation of

the $\frac{\nabla_x l(x, y)^T \delta}{\|\nabla_x l(x, y)\|}$ as

$$\mathbb{E} \frac{\nabla_x l(x, y)^T \delta}{\|\nabla_x l(x, y)\|} \leq p_{x+\delta} f(x) + (1 - p_{x+\delta}) \epsilon. \quad (2)$$

Instead of considering the vehicle coming closer to the object, we consider driving the vehicle back from the position where it is parallel to the object for ease of analysis. We use $g(x + \delta) = x + \gamma \delta$ to denote the transformed example of $x + \delta$ where $\gamma < 1$ is the scaled size. Based on the L-smoothness assumption, we have

$$\begin{aligned} l(x, y) + \gamma \nabla_x l(x, y)^T \delta - \frac{L\gamma^2 \epsilon^2}{2} \\ \leq l(g(x + \delta), y) \leq l(x, y) + \gamma \nabla_x l(x, y)^T \delta + \frac{L\gamma^2 \epsilon^2}{2}. \end{aligned} \quad (3)$$

Based on (3), we have

$$\begin{aligned} & P(F(g(x + \delta)) = y) \\ &= P(l(g(x + \delta), y) < l(g(x + \delta), y')) \\ &\geq P\left(l(x, y) + \gamma \nabla_x l(x, y)^T \delta + \frac{L\gamma^2 \epsilon^2}{2} < l(g(x + \delta), y')\right) \\ &= P\left(\gamma \nabla_x l(x, y)^T \delta < l(g(x + \delta), y') - l(x, y) - \frac{L\gamma^2 \epsilon^2}{2}\right) \\ &= P\left(\frac{\nabla_x l(x, y)^T \delta}{\|\nabla_x l(x, y)\|} < \frac{l(g(x + \delta), y') - l(x, y) - \frac{L\gamma^2 \epsilon^2}{2}}{\gamma \|\nabla_x l(x, y)\|}\right) \\ &\geq 1 - \frac{\mathbb{E} \frac{\nabla_x l(x, y)^T \delta}{\|\nabla_x l(x, y)\|}}{\frac{l(g(x + \delta), y') - l(x, y) - \frac{L\gamma^2 \epsilon^2}{2}}{\gamma \|\nabla_x l(x, y)\|}} \\ &\geq 1 - \frac{\gamma(p_{x+\delta} f(x) + (1 - p_{x+\delta}) \epsilon) \|\nabla_x l(x, y)\|}{l(g(x + \delta), y') - l(x, y) - \frac{L\gamma^2 \epsilon^2}{2}}. \end{aligned} \quad (4)$$

Since $P(F(g(x + \delta)) = y) = 1 - P(F(g(x + \delta)) \neq y)$, we have

$$\begin{aligned} & P(F(g(x + \delta)) \neq y) \\ &< \frac{\gamma \left(p_{x+\delta} f(x) + (1 - p_{x+\delta}) \epsilon \right) \|\nabla_x l(x, y)\|}{l(g(x + \delta), y') - l(x, y) - \frac{L\gamma^2 \epsilon^2}{2}} \\ &= \frac{\gamma \left(p_{x+\delta} \cdot \left(l(x + \delta, y') - l(x, y) - \frac{L\epsilon^2}{2} \right) + (1 - p_{x+\delta}) \epsilon \|\nabla_x l(x, y)\| \right)}{l(g(x + \delta), y') - l(x, y) - \frac{L\gamma^2 \epsilon^2}{2}} \end{aligned} \quad (5)$$

Considering the probability is positive, we can conclude that the denominator and numerator have the same sign. For simplicity, we consider $l(x + \delta, y') \geq l(x, y)$ and both of them to be positive. Note that the following derivation is similar when

they are negative, which derives the same conclusion. Specifically, we aim to find the value of the scale size s when the probability $P(F(g(x + \delta) \neq y)) \leq 1 - p_{x+\delta}$:

$$\begin{aligned} P(F(g(x + \delta) \neq y)) \leq 1 - p_{x+\delta} &\Leftrightarrow \gamma p_{x+\delta} l(x + \delta, y') \\ &\quad - l(g(x + \delta), y')(1 - p_{x+\delta}) + \gamma \epsilon (1 - p_{x+\delta}) \|\nabla_x l(x, y)\| \\ &\quad + l(x, y)(1 - p_{x+\delta} - \gamma p_{x+\delta}) \leq (\gamma p_{x+\delta} + \gamma^2 p_{x+\delta} - \gamma^2) \frac{L\epsilon^2}{2}. \end{aligned} \quad (6)$$

Applying Taylor expansion of $l(g(x + \delta), y')$ over (7) derives

$$\begin{aligned} &\gamma p_{x+\delta} l(x + \delta, y') - (1 - p_{x+\delta}) \left(l(x + \delta, y') \right. \\ &\quad \left. + \nabla_{x+\delta} l(x + \delta, y')^T (g(x + \delta) - (x + \delta)) + \frac{L(g(x + \delta) - (x + \delta))^2}{2} \right) \\ &\quad + l(x, y)(1 - p_{x+\delta} - \gamma p_{x+\delta}) + \gamma \epsilon (1 - p_{x+\delta}) \|\nabla_x l(x, y)\| \\ &\leq (\gamma p_{x+\delta} + \gamma^2 p_{x+\delta} - \gamma^2) \frac{L\epsilon^2}{2} \\ &\stackrel{(a)}{\Leftrightarrow} (\gamma p_{x+\delta} + p_{x+\delta} - 1)(l(x + \delta, y') - l(x, y)) \\ &\leq (\gamma p_{x+\delta} + \gamma^2 p_{x+\delta} - \gamma^2 + (1 - p_{x+\delta})(\gamma - 1)^2) \frac{L\epsilon^2}{2} \end{aligned} \quad (7)$$

where (a) by considering $\nabla_x l(x, y) \approx 0$ and $\nabla_{x+\delta} l(x + \delta, y') \approx 0$. Since $\gamma \leq 1$ and $p_{x+\delta} < \frac{1}{2}$, we have $\gamma p_{x+\delta} + p_{x+\delta} - 1 < 0$. Thus, the formula (7) holds when

$$\begin{aligned} &\gamma p_{x+\delta} + \gamma^2 p_{x+\delta} - \gamma^2 + (1 - p_{x+\delta})(\gamma - 1)^2 \geq 0 \\ &\Leftrightarrow \gamma \leq \frac{1 - p_{x+\delta}}{2 - 3p_{x+\delta}}. \end{aligned} \quad (8)$$

In summarize, when the vehicle stays in the position where the scale size s satisfies $\gamma \leq \frac{1 - p_{x+\delta}}{2 - 3p_{x+\delta}}$, the probability bound of $P(F(g(x + \delta)) \neq y)$ is bounded $P(F(g(x + \delta)) \neq y) \leq 1 - p_{x+\delta}$.

By denoting the number of frames of the pictures as N and the size of each frame as $d \times d$, we can obtain that the picture with the same size has $2N/d$ frames. By further denoting the minimum scale size corresponding to recognizing clean pictures is \bar{d} , we can calculate the maximum scale times r of which the attacking probability is less than $1 - p_{x+\delta}$ as:

$$\arg \max_r d \left(\frac{1 - p_{x+\delta}}{2 - 3p_{x+\delta}} \right)^r \geq \bar{d} \quad (9)$$

to obtain $r = \lfloor \frac{\ln \bar{d} - \ln d}{\ln(1 - p_{x+\delta}) - \ln(2 - 3p_{x+\delta})} \rfloor$. Based on the times, we can then compute the number n of frames of which the attacking probability is less than $1 - p_{x+\delta}$ as

$$n = \frac{2N(r + 1)}{d} = \frac{2N}{d} \left(\lfloor \frac{\ln \bar{d} - \ln d}{\ln(1 - p_{x+\delta}) - \ln(2 - 3p_{x+\delta})} \rfloor + 1 \right). \quad (10)$$

As a consequence, by denoting the probability of successfully attacking one frame as P_a , the maximum probability of successfully attacking all frames is less than the following probability:

$$P_{\max} = P_a^N \leq (1 - p_{x+\delta})^n = (1 - p_{x+\delta})^{\frac{2N}{d} (\lfloor \frac{\ln \bar{d} - \ln d}{\ln(1-p_{x+\delta}) - \ln(2-3p_{x+\delta})} \rfloor + 1)}. \quad (11)$$

TABLE 1: Physical adversarial attacks evaluated on a *real road* from existing works.

Sensor	Task	Method	Transformation	Adversarial Object	Target	Attack Success Rate (%)		Distance (m)			Inconsistent Periods (s)	
						White-box	Black-box	1-5	5-10	10-20	White-box	Black-box
Camera	Classification	AdvCam [63]	$S, \mathcal{R}, \mathcal{B}$	Patch	Traffic Sign	\times	\times	\times	\times	\times	\times	\times
	Classification	AdvLB [237]	$\mathcal{R}, \mathcal{T}, \mathcal{B}lur$	Laser	Traffic Sign	\times	\times	\times	\times	\times	\times	\times
	Classification	ShadowAttack [116]	$S, \mathcal{B}, \mathcal{P}\mathcal{T}, \mathcal{B}lur$	Shadow	Traffic Sign	MA: 95.91	\times	\times	\times	\times	\times	\times
	Classification	RP2 [62]	$S, \mathcal{R}, \mathcal{B}$	poster	Traffic Sign	MA: 100 *	\times	\times	\times	0:00-0:03	\times	\times
	Classification	RP2 [62]	$S, \mathcal{R}, \mathcal{B}$	art	Traffic Sign	MA: 84.8	\times	\times	\times	0:00-0:04	\times	\times
	Detection	ShapeShifter [64]	$S, \mathcal{R}, \mathcal{T}$	Patch	Traffic Sign	MA: 36-47	\times	\times	\times	0:07-0:09 0:09-0:12 0:13-0:15	\times	All attacks, all 27 videos are discontinuously
	Detection	Nested-AE [71]	—	Patch	Traffic Sign	AA: 63-81 HA: 60-75	\times	\times	\times	0:13-0:14, 0:21-0:27 0:24-0:27, 0:37-0:39 0:43-0:45	\times	\times
	Detection	Translucent Patch [238]	S, \mathcal{T}	Patch	Traffic Sign	HA: 21.54-42.27	\times	\times	\times	\times	\times	\times
	Detection	Poltgeist [239]	—	Blurring	Traffic Sign	HA: 98.3 CA: 43.7 AA: 43.1	\times	\times	\times	\times	\times	All videos are discontinuously except HA-TUNNEL**
	Detection	SLAP [66]	$S, \mathcal{R}, \mathcal{B}, \mathcal{P}\mathcal{T}, \mathcal{B}lur$	Optical	Traffic Sign	HA: 77-89.5	\times	1.5	1.5	89.5/77.5	5:45-5:48	-
Detection	TPatch [177]	$S, \mathcal{R}, \mathcal{B}, \mathcal{R}GB$	Patch	Traffic Sign	MA: 86.4-100	\times	100	85	\times	\times	\times	
MSF	Detection	MSF-adv [16]	\mathcal{R}, \mathcal{T}	3D obstacle	Cone & Bench	HA: 84.8	\times	\times	\times	\times	Not given	

S Scaling/Shearing, \mathcal{R} Rotation, \mathcal{T} Translation, \mathcal{B} Brightness, $\mathcal{P}\mathcal{T}$ Perspective Transformation, $\mathcal{B}lur$ Gaussian/Motion Blur/Noise, $\mathcal{R}GB$ Contrast, Saturation, Hue, \times : We did not get the data/details from their paper. *: The attack success rate shown in their video is not 100%. **: The camera shakes badly.