A Taxonomy of System-Level Attacks on Deep Learning Models in Autonomous Vehicles

MASOUD JAMSHIDIYAN TEHRANI, Università della Svizzera italiana, Switzerland
JINHAN KIM, Università della Svizzera italiana, Switzerland
ROSMAEL ZIDANE LEKEUFACK FOULEFACK, University of Trento, Italy
ALESSANDRO MARCHETTO, University of Trento, Italy
PAOLO TONELLA, Università della Svizzera italiana, Switzerland

The advent of deep learning and its astonishing performance has enabled its usage in complex systems, including autonomous vehicles. On the other hand, deep learning models are susceptible to mis-predictions when small, adversarial changes are introduced into their input. Such mis-predictions can be triggered in the real world and can result in a failure of the entire system. In recent years, a growing number of research works have investigated ways to mount attacks against autonomous vehicles that exploit deep learning components. Such attacks are directed toward elements of the environment where these systems operate and their effectiveness is assessed in terms of system-level failures triggered by them. There has been however no systematic attempt to analyze and categorize such attacks. In this paper, we present the first taxonomy of system-level attacks against autonomous vehicles. We constructed our taxonomy by selecting 21 highly relevant papers, then we tagged them with 12 top-level taxonomy categories and several sub-categories. The taxonomy allowed us to investigate the attack features, the most attacked components and systems, the underlying threat models, and the failure chains from input perturbation to system-level failure. We distilled several lessons for practitioners and identified possible directions for future work for researchers.

CCS Concepts: • **Security and privacy** → **Software security engineering**.

Additional Key Words and Phrases: Security Testing, Deep Learning Security, Autonomous Vehicle Security

ACM Reference Format:
Masoud Jamshidiyan Tehrani, Jinhan Kim, Rosmael Zidane Lekeufack Foulefack, Alessandro Marchetto, and Paolo Tonella. 2025. A Taxonomy of System-Level Attacks on Deep Learning Models in Autonomous Vehicles. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 31 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The advent of Autonomous Vehicles (AVs) has brought about a paradigm shift in transportation [31]. They leverage an array of sensors, cameras, and Deep Learning (DL) models to perceive and navigate their surroundings for safer and more efficient travel. However, as these vehicles have become increasingly reliant on complex DL components, they also expose new attack surfaces.

Authors' Contact Information: Masoud Jamshidiyan Tehrani, masoud.jamshidiyantehrani@usi.ch, Università della Svizzera italiana, Lugano, Switzerland; Jinhan Kim, jinhan.kim@usi.ch, Università della Svizzera italiana, Lugano, Switzerland; Rosmael Zidane Lekeufack Foulefack, rz.lekeufack@unitn.it, University of Trento, Trento, Italy; Alessandro Marchetto, alessandro.marchetto@unitn.it, University of Trento, Trento, Italy; Paolo Tonella, paolo.tonella@usi.ch, Università della Svizzera italiana, Lugano, Switzerland.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

While highly effective in most cases, these DL components possess inherent vulnerabilities, such as being sensitive to adversarial examples [26], that can lead to catastrophic failures. A mis-prediction at the DL model level can quickly escalate into a system-level failure, resulting in vehicle crashes or collisions with pedestrians, thus endangering lives. Some studies discuss model- and system-level failures under critical conditions, such as bad weather[63]. These are considered part of functional testing for AVs. However, they do not evaluate model- or system-level failures under adversarial attacks. Salay et al. [65] pointed out that there is a missing link between system-level safety requirements and component-level performance guarantees, particularly for perception tasks. These tasks are both critical and challenging because they heavily rely on machine learning, which introduces uncertainty and variability in performance. Burton [66] concluded that safety assurance for machine learning requires probabilistic reasoning and a causal understanding of how model insufficiencies can propagate into system-level failures. He also emphasized that relying on a single form of evidence is insufficient to ensure safety. A large body of research has focused on the model-level security of DL components, tested by introducing various attack (resp. defense) strategies to induce (resp. prevent) mis-predictions in these models [27]. As evidenced by Wang et al. [7], however, not all attacks on the DL model propagate to the system of an AV, meaning that the success of the model-level attack does not always translate to system failures. In their study, this was largely due to a lack of consideration for the physical properties of the underlying vehicle system, such as maximum and minimum acceleration/deceleration, steering rates, and sensor mounting positions. There are safety standards for autonomous systems, such as ANSI/UL 4600 and ISO/PAS 8800, that highlight the connections between model-level errors and system-level failures. These standards also emphasize the importance of understanding safety-related subsets of the input space and how certain types of errors can contribute to failures at the system level. A few recent studies have extended the evaluation of attack methods beyond the model level, considering their system-level impacts in real-world settings, such as the frequency of vehicle crashes [30]. To the best of our knowledge, no research has comprehensively taxonomized the attacks on AVs' DL models that cause system-level failures. This paper presents a taxonomy of system-level attacks on AVs, building upon existing research that has explored the results of DL model-level failures under adversarial attacks in the context of AVs. The contribution of this work is to provide an understanding of the vulnerabilities and potential attacks on AV systems, which can help developers identify areas where defenses are most needed.

We adopted a bottom-up approach to the creation of a taxonomy for classifying the existing literature. We allowed the incremental identification and creation of new categories, as we analyzed the literature in depth. We collected papers by searching scientific databases using a carefully designed query. This initial search yielded 9,748 papers. Then, we manually analyzed and filtered these papers based on their venues, reducing our list to 1,125 potential candidates. Lastly, through snowballing and a thorough assessment by three reviewers, we identified 21 papers that formed the basis of our taxonomy. These papers were categorized into 12 distinct groups, including system-level failure-specific categories such as 'System Under Attack' or 'System-level Results' to differentiate our taxonomy from previous work that focused only on the characteristics of model-level attacks [26]. We further provide statistics to highlight the distribution of properties in the taxonomy, as well as a tree-like visualization to enhance the readability and practical usage of the taxonomy. Finally, our discussion elaborates on the implications and potential future directions arising from our taxonomy and findings, opening up new avenues for future exploration in this emerging field.

The contributions of the paper are as follows:

- The first taxonomy of security vulnerabilities that affect autonomous vehicles whose perception components are based on DL. The taxonomy consists of 12 top-level categories and several subcategories.
- A detailed analysis of 21 papers that are highly relevant for system-level attacks against autonomous vehicles. We provide a thorough mapping between papers and taxonomy categories.
- A comprehensive overview of the AV domain to identify categories covered or missing in the taxonomy papers, highlighting gaps and directions for future research.
- A discussion of the main findings obtained while constructing the taxonomy and analysing the papers. The discussion includes implications for practitioners and possible avenues for future research.

The rest of the paper is structured as follows. Section 2 introduces some important notions about AVs and about DL security. Section 3 comments on the research questions that triggered and motivated the construction of our taxonomy. Section 4 analyzes the existing domains of AVs, DL models and simulators. Section 5 describes the methodology adopted for paper selection and taxonomy construction. Section 6 presents our findings, including a description of the taxonomy and the mapping of the considered papers. Section 7 answers the initial research questions and discusses the implications of this work for practitioners and researchers. Section 8 comments on the related work. Section 9 enumerates the main threats to the validity of our findings and the ways in which we tried to mitigate them. Section 10 contains our conclusion and anticipates some possible future works.

2 Background

2.1 Autonomous Vehicles

Vehicles equipped with automated components can achieve different degrees of autonomy. For self-driving cars, five levels have been defined [24], ranging from *driving assistance* to *full driving automation*. However, according to Faisal et al. [32], an automated vehicle system can only be termed *autonomous* when the vehicle's automated system can seamlessly perform all dynamic driving tasks across a diverse range of environments (i.e., only at level 5, *full driving automation* [24]). In this paper, we are not interested in differentiating among different degrees of automation vs. full autonomy, as we consider in the scope of our taxonomy all works that perform system-level security attacks against the deep learning components used to provide *some* automation to a vehicle, up to full autonomy.

The motion control of a vehicle can be divided into two primary tasks: *lateral* motion control and *longitudinal* motion control. Lateral control, managed by the vehicle's steering, aims to maintain the vehicle's lane position and perform maneuvers such as lane changes and collision avoidance. This is typically achieved through DL components that process images from onboard cameras. Longitudinal control, managed by the gas and brake pedals, focuses on reaching/maintaining the desired speed, ensuring safe distances from other vehicles, and preventing rear-end collisions. This control relies on sensors like radar and lidar, as well as cameras, to measure relative velocity and distance [31]. If an AV combines data from different types of sensors to improve accuracy and robustness, it's using a Multi-Sensor Fusion (MSF) system [68]. If it merges multiple or single inputs from the same sensor type to enhance perception, it's using a Single-Sensor Fusion (SSF) system [69]. Other relevant tasks that are assisted by DL components include parking, driver's attention checking, pedestrian and obstacle avoidance, traffic sign/light detection and recognition. Overall, AVs may contain the following modules to achieve autonomy [67]:

- **Perception:** Extracts and interprets sensor data to detect objects and understand the environment.
- **Localization:** Determines the vehicle's precise position and orientation in the world.
- **Planning:** Creates safe and efficient paths based on goals (global planning) and surroundings (local planning).
- **Control:** Executes planned paths by commanding the vehicle's actuators.
- **End-to-End:** Uses a single model to map sensor inputs directly to control actions.

2.2 Adversarial Attacks

DL models have achieved impressive performance across various application domains. However, these technologies can be easily deceived by adversarial examples, such as carefully perturbed input samples designed to mislead detection systems during testing. Researchers have demonstrated that deep neural networks for object recognition or classification can be fooled by input images altered in ways that are imperceptible to humans [26]. A *model-level attack* is an attack that exploits weaknesses of the DL model in isolation to obtain a mis-prediction for specific, possibly manipulated, inputs. Specifically, there are two primary *model-level attack* strategies. Firstly, *poisoning attacks* occur when an attacker, with access to the training data, inserts malicious data into the training set. Such data acts as a trigger and when the trigger is present in the input at inference time, it forces the model to make incorrect predictions. Secondly, *evasion attacks* occur when an attacker manipulates input data during testing without altering the model itself [26, 27]. These attacks highlight the vulnerabilities of DL models in isolation, but they might be leveraged to trigger potential consequences at the system-level, in safety-critical applications such as AVs.

To establish the scope of our taxonomy, we first clarify the key concepts used in this work. We define a *model-level attack* as an attack that exploits vulnerabilities and limitations in machine learning models, leading to model-level failures such as misclassification, increased latency, and model uncertainty. We define a *system-level attack* as an attack that manipulates the environment where the AV operates, in order to cause a system-level failure of the AV. In turn, a *system-level failure* is a deviation of the autonomous system's behavior from its functional/safety requirements. When mounting a system-level attack, one or more model-level attacks can be exploited by first forcing some malicious inputs into the models under attack, and then ensuring the mis-predictions propagate to a system-level failure. This paper aims to build a taxonomy based on the literature that has explored system-level attacks and their impact in terms of system-level AV failures.

3 Research Questions

Research on system-level attacks against AVs is an emerging, rapidly growing field. The *goal* of our work is to acquire a better understanding of the research landscape by categorizing the existing works and defining the main dimensions that characterize them. More specifically, we build a system-level attack taxonomy to answer the following research questions:

RQ1 [Attack Features]: *What are the prevalent features of the attacks, based on the paper distribution across taxonomy categories?*
With this RQ, we investigate how extensively the research landscape has been explored, to highlight regions that are still relatively unexplored and to identify the features that characterize the most populated regions. This might indicate possible directions for future work, but also the reasons why some research directions are more feasible than others.

RQ2 [Attacked Components]: *Which components of the system are the targets of the attack?*

DL models are used for several tasks in AVs, including lane keeping, obstacle recognition and avoidance, and traffic sign/light recognition, which are components of different AV modules, such as the perception, planning, localization, and control modules. With this RQ, we investigate if there are preferred attack targets among the DL components commonly integrated into AVs. We aim to identify the features of those components that make them suitable for system-level attacks, possibly operated by manipulating the environment where the AV operates.

RQ3 [Threat Models]: *What are the threat models?*
Different system-level attacks are mounted under very different assumptions about the knowledge available to attackers and their capability to manipulate the environment. Hence, some assumptions may be more reasonable or realistic than others, substantially changing the probability that an attack could be performed in practice. With this RQ, we aim to understand whether the research effort was directed mostly toward realistic attacks that are reasonably possible in the real world, or toward scenarios based on unrealistic assumptions that make them very unlikely.

RQ4 [Consequences]: *What are the consequences of each attack type at the system level?*
The main difference between a model-level and a system-level attack is that in the latter case, an attack is successful only if model-level mis-predictions influence downstream components and ultimately lead to a system-level failure. With this RQ, we investigate the attack paths exploited in existing works, considering the attack vector (e.g., the environment element manipulated by the attack), its effects on a model input, a model mis-prediction, and the outcomes of such mis-prediction in external observable actions that eventually cause a system failure.

4 Analysis of the AV Domain

AVs range widely, from cars and trucks to spacecraft. They rely on various DL models, modular architectures, and simulation environments for development and testing. Before searching for scientific papers that deal with system level attacks against AVs, we compile a comprehensive list of *AV types*, *DL models*, and *simulation tools* used in this domain. The goal is to get a thorough understanding of the AV domain that is not biased by the AV types, models and simulators considered in practice in the papers selected for taxonomy construction. We have utilized a Large Language Model (LLM), specifically the GPT-4-turbo variant [61], for this task, using a carefully crafted prompt that requires a structured output along with relevant reference links. All outputs and links were manually reviewed to ensure correctness and relevance. We chose to leverage an LLM over traditional web search due to its ability to synthesize domain-specific information from broad, unstructured sources. While LLM-generated content may be prone to hallucinations, our manual verification revealed no instances of hallucinations, misleading, or incomplete information. Based on the responses, we created Table 1, which we overview below. The exact prompt and corresponding outputs are included in the replication package on Zenodo[1] and GitHub[2].

4.1 Typology of AVs

AVs encompass a wide array of systems capable of operating independently across various environments, including land, air, sea, and space, by leveraging integrated sensors, artificial intelligence (supervised and reinforcement learning), control systems, and general software. According to the LLM's output, these vehicles can be classified based on their operational domains. Land-based autonomous systems include self-driving cars, trucks, buses, agricultural machinery, and ground

[1]https://zenodo.org/records/16407709
[2]https://github.com/MasoudJTehrani/sys-tax-artifact

Table 1. An overview of autonomous vehicles, models and simulations

Autonomous Vehicles	
Surface	**Types**
Land	UGV (including military robots), Autonomous Cars/Trucks/Buses
Air	UAVs/Drones, eVTOL Air-Taxis, Autonomous Combat/Security Drones
Sea	USVs (boats, cargo ships, saildrones, survey vessels)
Underwater	AUVs (cylinder, glider, hybrid), Large UUVs, ROVs (remote)
Space	Spacecraft, planetary rovers, autonomous satellites/explorers

Autonomous Models & Modules	
Modules	**Common Models & Techniques**
Perception	CNNs (YOLO, Mask R-CNN, SSD), Panoptic NN (YOLOP), LiDAR networks (VoxelNet, PointRCNN)
Localization & Mapping	SLAM (EKF, GraphSLAM), Visual odometry with DL-enhanced features
Prediction & Planning	LSTM/RNN for trajectories, GNNs (VectorNet), DRL, inverse RL
Control	DRL/Q-networks, inverse RL for policy generation
End-to-End	CNN "black box" (Openpilot, DAVE-2), encoder-decoder + depth, MLLM (EMMA), vision-first foundation models

Simulation Environment	
Surface Domain	**Simulators**
Land	CARLA, AirSim (cars), Gazebo/Ignition, Webots, CoppeliaSim, Unreal/Unity car sims, SUMO, Drake
Air	AirSim, Gazebo (PX4 SITL, RotorS), Webots, CoppeliaSim, FlightGear/X-Plane, Pegasus/Isaac Sim, Flightmare, ROSUnitySim, USARSim, MORSE, GrADyS-SIM
Sea	Gazebo + usv_sim_lsa, Marine AUV/USV Simulator (commercial)
Underwater	Gazebo + uuv_simulator, MATLAB & Simulink AUV Toolbox
Space	Trick (NASA), Gazebo (NASA rover sim)

robots [41, 42]. Airborne vehicles encompass fixed-wing drones, rotary-wing UAVs, vertical take-off and landing (VTOL) aircrafts, and high-altitude pseudo-satellites (HAPS) [43]. Surface-sea vehicles (USVs) operate above water for tasks such as marine surveillance and cargo transport [44], while underwater vehicles (AUVs/UUVs) conduct deep-sea exploration and inspection missions [45]. Space-based AVs range from planetary rovers to autonomous spacecraft performing orbit control, docking, and planetary landing [46].

4.2 DL Models in AV Systems

AVs rely on a suite of DL models. According to the LLM's output, such models are modularly deployed across perception, localization, prediction, planning, and control systems. In the perception module, convolutional neural networks (CNNs) such as YOLO [40], Mask R-CNN [49], Single Shot MultiBox Detector (SSD), and panoptic models like YOLOP are commonly used for object detection, segmentation, and road understanding. LiDAR-based models like VoxelNet [47] and PointRCNN [48] process 3D spatial data. Localization and mapping leverage SLAM [51] techniques augmented with visual odometry and deep feature extractors. For trajectory forecasting and environmental modeling, models such as LSTMs, RNNs, and graph neural networks (e.g., VectorNet [64]) are employed. Decision-making and motion control are informed by deep reinforcement learning (DRL), inverse reinforcement learning, and policy gradient methods. Finally, end-to-end autonomous driving stacks may use CNN-based architectures, encoder-decoder frameworks, or multimodal large models integrating vision, LiDAR, and navigation data.

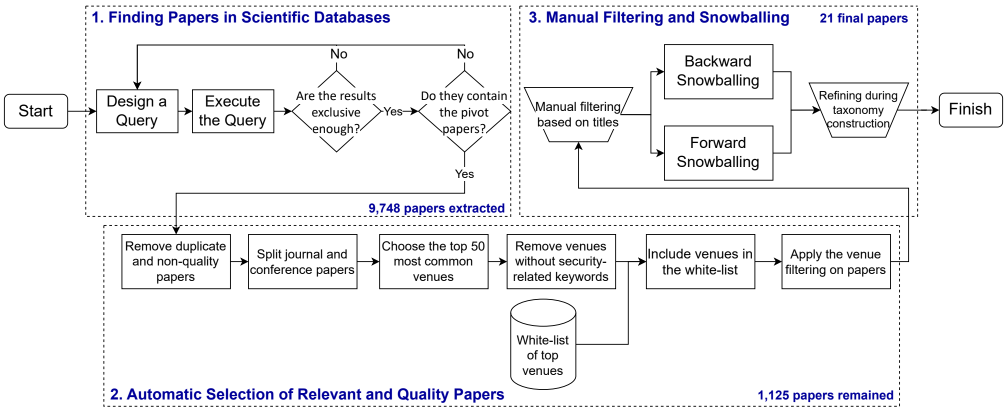
1. Finding Papers in Scientific Databases
Start
Design a Query
Execute the Query
Are the results exclusive enough?
No
Yes
Do they contain the pivot papers?
No
Yes
9,748 papers extracted
3. Manual Filtering and Snowballing
21 final papers
Manual filtering based on titles
Backward Snowballing
Forward Snowballing
Refining during taxonomy construction
Finish
Remove duplicate and non-quality papers
Split journal and conference papers
Choose the top 50 most common venues
Remove venues without security-related keywords
Include venues in the white-list
Apply the venue filtering on papers
White-list of top venues
2. Automatic Selection of Relevant and Quality Papers
1,125 papers remained

Fig. 1. Overview of the paper collection process

4.3 Simulation Environments for Autonomous Vehicle Testing

Simulation environments are critical for developing, testing, and validating AV systems across diverse domains. According to the LLM's output, land-based simulation platforms such as CARLA [33], AirSim (ground mode) [52], Webots [53], and CoppeliaSim [54] enable realistic testing of self-driving cars and wheeled robots with customizable sensors and traffic agents. Airborne systems benefit from tools like AirSim (drone mode), Gazebo (with PX4/RotorS) [55], and FlightGear [56], supporting UAV simulation and flight dynamics. Surface-sea and underwater vehicles are modeled using Gazebo plugins (usv_sim_lsa and uuv_simulator), MATLAB's AUV toolbox, and commercial marine platforms. For space applications, NASA's Trick [57] and Gazebo are the most widely used simulators, as reported by the LLM.

5 Methodology

Figure 1 presents an overview of our paper collection process. We adopted a systematic bottom-up approach, to curate a comprehensive collection of papers from diverse sources. We designed a search query to gather the papers from scientific databases such as Scopus (see below), spanning a 7-year period from 2017 to 2024[3]. This was followed by automatic selection and manual filtering. Automatic selection refines the results based on venue reputation, so as to include only well-established sources, while manual filtering involves the examination of each paper by multiple assessors. Additionally, we included relevant papers discovered through snowballing, ensuring a detailed collection by leveraging references to/from other works.

5.1 Initial Paper Collection Through Scientific Databases

We employed `Findpapers` [25] to collect our initial set of papers. This tool allows us to design a customized search query, focusing on paper titles and abstracts, and supporting multiple databases including IEEE Explore[4], Scopus[5], and ACM Digital Library[6]. To ensure both comprehensive coverage and relevance, we iteratively refined our query based on two key criteria: the number of

[3]We selected 2017 as our starting year since research in the security of deep learning in AVs began to gain significant momentum after that year.
[4]https://ieeexplore.ieee.org/Xplore/home.jsp
[5]https://www.elsevier.com/products/scopus
[6]https://dl.acm.org/

papers collected and the inclusion of three manually selected pivotal papers. Our pivotal papers contain two survey papers [26, 27] and DeepBillboard [29], which deal with adversarial attacks on AVs[7]. Through this iterative process, we balanced the query output, encompassing key papers while maintaining a broad search scope.

Given our focus on system-level attacks on AVs, we designed a search query that requires the papers' metadata (abstract, title, and keywords) to contain three key terms: 'autonomous', 'vehicle', and 'attack'. Considering that not all papers may use these terms explicitly, we expanded our query to include synonyms. For 'autonomous', we include variations such as 'automated', 'driverless', 'self-driving', and 'unmanned', connected with the `OR` operator to account for any of these terms appearing in the metadata. Similarly, for 'vehicle*', we incorporate terms like 'car*', 'drone*', 'truck*', 'marine', etc, also connected with `OR`. Note that the use of asterisks ensures coverage of plural forms.

To ensure that each metadata contains at least one key term, all the term groups and their connected synonyms are linked with the `AND` operator. Consequently, our final query is as follows:

```
(([autonomous] OR [self-driving] OR [driverless] OR [unmanned] OR [automated])
AND
([vehicle*] OR [car*] OR [drone*] OR [truck*] OR [tractor*] OR [submarine*] OR [
    boat*] OR [maritime] OR [robotaxi*] OR [marine] OR [space] OR [spacecraft] OR
    [agriculture] OR [agricultural] OR [delivery] OR [construction])
AND
([attack*] OR [adversarial]))
```

Executing this query across three databases from January 2017 to January 2024, we retrieved a total of 9,748 papers. Note that terms like 'machine learning', 'deep learning' and 'system-level' are not explicitly mentioned in this query as they might exclude too many relevant papers. However, the subsequent manual filtering process ensured that only papers in the scope of our study (i.e., system level attacks against deep learning models) were taken into consideration. Moreover, the term 'robots' is included to incorporate dynamic and mobile autonomous robots, such as delivery robots.

Next, our task was to refine this extensive pool of papers by filtering out those considered irrelevant or lacking in quality. Initially, we removed duplicate papers. Subsequently, we evaluated the publication venues based on established academic standards, such as the presence of a rigorous peer-review process and their focus on topics relevant to security. Papers from venues that did not meet these criteria were excluded. Employing a *venue frequency filter*, we reduced the number of papers from 9,748 to 1,125. This filter was designed by ordering journals and conferences separately by frequency of occurrence in the pool and then selecting the top 50 most prevalent venues. Out of them, we eliminated venues lacking security-related keywords in their titles. For example, communication-related journals and conferences, and multimedia journals and conferences are excluded. Recognizing that this approach may exclude reputable software engineering/security-focused venues, we created a white list comprising prominent venues, ensuring their inclusion after the filtering process. This venue white-list is available online at https://zenodo.org/records/16407709.

We continued by manually filtering the 1,125 papers, focusing solely on their titles, which narrowed them down to 83 papers. From there, we delved deeper, examining the abstracts and further refining our selection to 30 papers.

[7]These papers are excluded from the final taxonomy since they do not satisfy criteria introduced in later phases of the process (e.g., no survey; system-level failure propagation).

5.2 Snowballing

The snowballing [28] phase is aimed at expanding the pool of relevant papers by following their connections with other relevant papers. We employed both backward and forward snowballing techniques on the 30 papers obtained from the previous phase.

Backward snowballing involves examining the references cited within each of the selected 30 papers and including additional papers based on their titles and abstracts. Conversely, forward snowballing involves identifying papers that cite the selected 30 papers. To achieve this, we utilized Google Scholar[8] to locate the papers referencing each selected paper and considered as candidates to be relevant papers only those within the first two pages of the search results. We limited our selection to the first two pages because Google Scholar ranks results based on relevance. To prevent excessive iterations, we did not perform snowballing on the newly added papers. In the end, our collection expanded to 50 papers. However, further filtering was still required, since selecting papers based only on their titles and abstracts is not necessarily accurate. This further selection was performed during the process of paper reading and categorization for taxonomy construction.

5.3 Creation of Taxonomy Categories

In this phase, we have read the 50 papers collected after snowballing, with the goal of creating the taxonomy categories and excluding papers that do not meet the inclusion criteria. In fact, sometimes it is necessary to read a paper fully to realize that it cannot be included in the taxonomy.

The process of category definition for the taxonomization of the papers was bootstrapped with the categories already available from the literature of attacks on DL models and AVs. Initially, we adopted ten categories suggested by Biggio et al. [26] and Cinà et al. [27]. Subsequently, during taxonomy construction, we widened the scope to include concepts explicitly associated with system-level attacks and failures. This involved factors such as the system targeted by the attack and the specific system failures encountered during the attacks. In total, we identified 12 top-level categories, each listed in Table 2 with the corresponding explanations. A detailed description of these categories, as well as the sub-categories, is carried out below, in Section 6.3.

In the phase of paper categorization and taxonomy refinement, we read each paper thoroughly. Specifically, each paper was read and categorized by at least two assessors. Weekly consensus meetings were held to resolve and negotiate any differences in the categorization. We excluded papers that do not specifically target the DL component of the AV under attack. For instance, papers that attack the hardware sensors and cause them to malfunction, resulting in incorrect point cloud generation and subsequent failure to recognize targets, are discarded as they do not directly involve any DL model [40]. Additionally, papers that only evaluate their attack on a dataset without executing the system, or collect their own dataset using cameras mounted on non-autonomous, manually driven vehicles, are excluded [29]. This criterion led to the exclusion of many papers, including the only two studies that specifically targeted Maritime Autonomous Surface Ships (MASS) [58, 59]. Papers that do not report any system-level failures, despite executing their attacks against AVs [38, 39, 60], are also discarded. Although these papers may report mispredictions of the vehicle's DL model, they do not provide information on the system-level vehicle's behavior once the attack has caused a model-level misprediction. As a result of this strict refinement process, the number of relevant papers was reduced from 50 to 21. Other avenues for understanding attacks on AVs include research that targets network-based functionalities or attacks on DL models without system-level evaluation, which were not our goal in this taxonomy.

[8]https://scholar.google.com

Table 2. Examples of Possible Values of Each Taxonomy Category of System-level Attacks

Category	Explanation	Possible Values
Application Domain	The domain or type of vehicle in which the attacker is attempting to cause a system-level failure	Cars, Drones, Trucks, Ships
DL Model Under Attack	The specific deep-learning model and module, targeted by the attack	Yolov5, ResNet-34, Dave2
System Under Attack	The specific autonomous system targeted by the attack	Baidu Apollo in LGSVL, CARLA's agents (MSF or SSF)
Attack Scenario	The particular scenario that must be present in order for the attack to be successful	Cars in traffic jams, Cars in intersections
Attacked Target	The specific object or element that the attacker targeted to cause a failure	Roads, Billboards, Input image
Attacker's Capability	The capabilities the attacker needs to execute the attack	Attach stickers on the road, Install malware
Attack Type	Whether the attack involves poisoning vs evasion tactics	Poisoning, Evasion
Attacker's Knowledge	The level of knowledge or access the attacker must possess about the system or model	White-Box, Black-box, Gray-box
Attack/Error Specificity	Whether attack/model-level error are generic or specific	Generic/Specific Attack/Error
Failure Specificity	Whether the attacker aims to induce a specific system-level failure or not	Generic, Specific
Model-level Result	Outcome of the attack at the DL model-level	Misclassification, Misdetection
System-level Result	Impact of the attack at the system level	Collision with object, Crash, Freeze

6 Results

In this section, we first provide a summary of each paper. Following this, we present the taxonomy mapping — a table that classifies each paper according to the identified categories (multiple categories can be applied to each paper at the same time), offering a comprehensive overview of the existing research in the field. Finally, we comment on the taxonomy tree, which visually links each category and subcategory to the corresponding papers.

6.1 Summary of papers

A summary of the 21 papers that met our criteria is provided here to give an overview before delving into the detailed classification of each paper. The papers are organized in chronological order, starting with the most recent to the oldest.

"Slowtrack: Increasing the latency of camera-based perception in autonomous driving using adversarial examples": Ma et al. [1] introduced SlowTrack, a framework designed to increase the execution time of camera-based autonomous vehicle perception by inserting fake bounding boxes. This attack targets both object detection and tracking systems, resulting in vehicle crashes with a success rate of 95%.

"RPAU: Fooling the Eyes of UAVs via Physical Adversarial Patches": Liu et al. [2] placed adversarial patches in the environment, including on moving vehicles, to disrupt drone perception. These patches caused the drones to either miss real objects and crash into them, change direction and veer off course, or detect fake objects and freeze in response.

"Adversarial Attacks on Adaptive Cruise Control Systems": Guo et al. [3] used leading vehicles with adversarial patches placed on their rear, causing the target vehicle to either fail to detect the vehicle ahead or perceive it as being farther away – potentially leading to a collision. To make the attack stealthy, they disguised the patch as a fake advertisement displayed on the back of a truck.

"Learning when to use adaptive adversarial image perturbations against autonomous vehicles": Yoon et al. [4] worked on injecting malware into the system to manipulate the input image. By adding perturbations, the attacker can create fake bounding boxes in specific locations, deceiving the victim's vehicle. This causes the vehicle to lose track of its intended target, potentially leading to a collision.

"Deepmaneuver: Adversarial test generation for trajectory manipulation of autonomous vehicles": Stein et al. [5] successfully manipulated the steering angle prediction model of a victim's vehicle by installing a perturbation on a billboard on the left-hand side of the road. This caused the model to predict the wrong steering angle for a given input image, leading the vehicle to leave the road or collide.

"Kidnapping deep learning-based multirotors using optimized flying adversarial patches": Hanfeld et al. [6] proposed an attack method capable of hijacking multi-rotor drones using flying adversarial patches, or small images strategically positioned in the surroundings. As a result, the drone deviates from its intended trajectory, losing track of the target human and instead following the adversarial patch.

"Does physical adversarial example really matter to autonomous driving? Towards system-level effect of adversarial object evasion attack": Wang et al. [7] explored the limitations of existing object evasion attacks in autonomous driving systems, noting their failure to achieve system-level effects due to inadequate understanding of the vehicle's physical properties. These include parameters like maximum/minimum acceleration/deceleration, steering rates, and sensor placements. To overcome these limitations, the paper introduced SysAdv, a novel attack design that integrates such system-specific knowledge. SysAdv significantly enhanced the success rate of attacks by approximately 70%.

"On data fabrication in collaborative vehicular perception: Attacks and countermeasures": Zhang et al. [8] proposed an attack method that exploits LiDAR-based collaborative perception in AVs, which can spoof or remove objects at specified locations in the victim's perception results, making all mainstream types of collaborative perception schemes vulnerable.

"Rolling colors: Adversarial laser exploits against traffic light recognition": Yan et al. [9] proposed an attack targeting the color recognition systems of traffic lights in autonomous vehicles (AVs). They exploited the vulnerability of rolling shutters in CMOS cameras by directing a laser beam to create a colored stripe in the input image. This manipulation resulted in the misclassification of the traffic light's color.

"Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems": Wang et al. [10] researched poisoning a DRL-based controller in congested traffic. The attacker's car sets a trigger based on a combination of speed and position, causing the victim's vehicle to perform inappropriate actions like accelerating or decelerating incorrectly. This ultimately results in a collision with the vehicle in front, occurring on both one-lane and two-lane roads.

"Attack and fault injection in self-driving agents on the carla simulator–experience report": Piazzesi et al. [11] modified the trained agent's neurons and weights, as well as the input image, to cause the car to make incorrect steering decisions and mis-detect traffic lights. This resulted in collisions with surroundings, lane departure, going off-road, running through crossroads, and ignoring traffic lights.

"Dirty road can attack: Security of deep learning based automated lane centering under Physical-World attack": To deceive the victim vehicle's automated lane-centering system, Sato et al. [12] printed specially designed patches on the road that resembled dirty road markings. This manipulation caused the system to make incorrect steering predictions, leading the vehicle to engage in hazardous maneuvers such as turning left or right incorrectly, driving off-road, and causing collisions.

"Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks": Cao et al. [13] conducted a physical sensor-based attack aimed at deceiving the camera and Lidar perception systems of the victim's vehicle, specifically while driving in a single road lane. Their strategy involved 3D printing and placing an adversarial traffic cone on the road that was designed to appear broken and thus remain undetected by both sensors. This deception caused the vehicle's systems to fail to recognize the object, resulting in collisions.

"Too good to be safe: Tricking lane detection in autonomous driving with crafted perturbations": Jing et al. [14] implemented a technique where three white dots are painted on the road either upon completing the road lines or when there is space between them. These dots are designed to trick cars into mistaking them for real lane markers, leading the vehicles to follow the dots and veer into the opposite lane.

"Robust roadside physical adversarial attack against deep learning in lidar perception modules": Yang et al. [15] utilized 3D printing to create a small adversarial object, which they placed on the road. The purpose was to deceive the victim's car into mistaking the object for another vehicle, prompting it to either change lanes or come to a complete stop upon detecting the object.

"ML-driven malware that targets AV safety": Jha et al. [16] installed malware to introduce noise to the outputs of object detection, tracking, and sensor fusion systems. The malware modifies the outputs under specific conditions by adding perturbative data as sensor noise. This noise makes the victim's car believe that a car in another lane is moving into its lane or that a car in the same lane in front of it has disappeared. Additionally, it can make the car think that pedestrians are moving into the street. This results in emergency braking or unwanted acceleration, leading to a car crash.

"Attacking vision-based perception in end-to-end autonomous driving models": In the research of Boloor et al. [17], when the victim's car is at an intersection or on a turning road, the attacker paints adversarial road lines pointing in the opposite direction of the lane turn, causing the car to turn the wrong way.

"Feasibility and suppression of adversarial patch attacks on end-to-end vehicle control": Pavlitskaya et al. [18] proposed manipulating an autonomous vehicle's trajectory by placing a printed adversarial patch on the roadside. This action causes the vehicle to steer towards the patch, resulting in a collision.

"Phantom of the ADAS: Securing advanced driver-assistance systems from split-second phantom attacks": Nassi et al. [19] developed a method to project images of humans or traffic signs onto roads or flat surfaces. These projections, referred to as phantoms, are intended to deceive cars into perceiving them as real signs or pedestrians, prompting the vehicles to respond accordingly.

"Adversarial sensor attack on lidar-based perception in autonomous driving": Cao et al. [20] directed lasers from an adversarial vehicle toward the target vehicle's Lidar system. This intentional interference aims to create false objects that do not exist, leading the car to freeze or suddenly brake.
"Trojaning attack on neural networks": In the work of Liu et al. [21], the attacker deceived the victim's vehicle by poisoning its driving model with trojan data to cause a wheel angle misprediction. When the vehicle encounters the trojan attached to a billboard, it is instructed to turn right, leading to the vehicle driving off the road.

6.2 Mapping the papers

For the taxonomy mapping, we employed a rigorous process: each paper was independently reviewed and categorized by at least two assessors. To ensure consistency and reliability, the assessors regularly met to discuss and resolve disagreements in their mappings. The results of this process are presented in the Appendix, in Tables 3, 4, and 5, where each column represents a distinct category, and each row corresponds to an individual paper and its associated categorization.

We conducted weekly meetings to ensure that the assessors' perspectives aligned with each other while reading and categorizing the papers. Over time, the meetings led to a decrease in disagreements, which typically involved the category to choose (major disagreement) or how to precisely formulate a (sub-)category (minor disagreement). For paper mapping, we used a shared spreadsheet containing 228 cells. Among the 228 cells that were filled, around 20 cells (9%) faced major disagreements and approximately 80 cells (35%) faced minor disagreements. On average, 8 disagreements were resolved per meeting, resulting in a total of about 96 resolved disagreements over 12 meetings. Initially, the categories with the most disagreement were "Attacker's System/-Model Knowledge," followed by "Attack/Error Specificity" and "Attacked Target." Possible reasons for the higher disagreement in these categories were the difficulty of giving a unique interpretation and purpose to these categories, and the difficulty of extracting the needed information from some papers, as it is not always directly provided by the authors. The consensus meetings turned out to be a fundamental instrument to address such initial disagreements and to come out with a common understanding and interpretation, which was reflected in a much clearer definition of the categories and choice of the labels for the categories after such meetings.

6.3 Taxonomy tree

Figure 2 provides a visual representation of the resulting taxonomy as a tree. The root node branches into 12 top-level categories, each further subdivided into relevant subcategories. The leaf nodes of the tree contain lists of papers associated with their respective (sub)categories. Each category and subcategory is detailed in the following.

(1) **Application Domain:** Domain or vehicle type that is the target of the attack, such as "Cars" and "Drones. If we compare the application domains covered by the papers included in the taxonomy with the AV types reported by the LLM in Table 1, we can notice that most AV surfaces, including sea, underwater and space, are uncovered. On the other hand, all these uncovered AV types represent critical domains where security testing would be quite important. We conclude that research is missing on the system level security of AVs operating on surfaces other than land and air.
(2) **DL Model Under Attack:** The DL model under attack, which is responsible for the system-level failure, such as YOLOv5, ResNet-34, or DAVE-2. The "DL Model Under Attack" category has four subcategories. The first, "Object Detection & Tracking", includes DL models (e.g., YOLO) that handle object tracking or detection. The "Steering Wheel Angle Prediction"

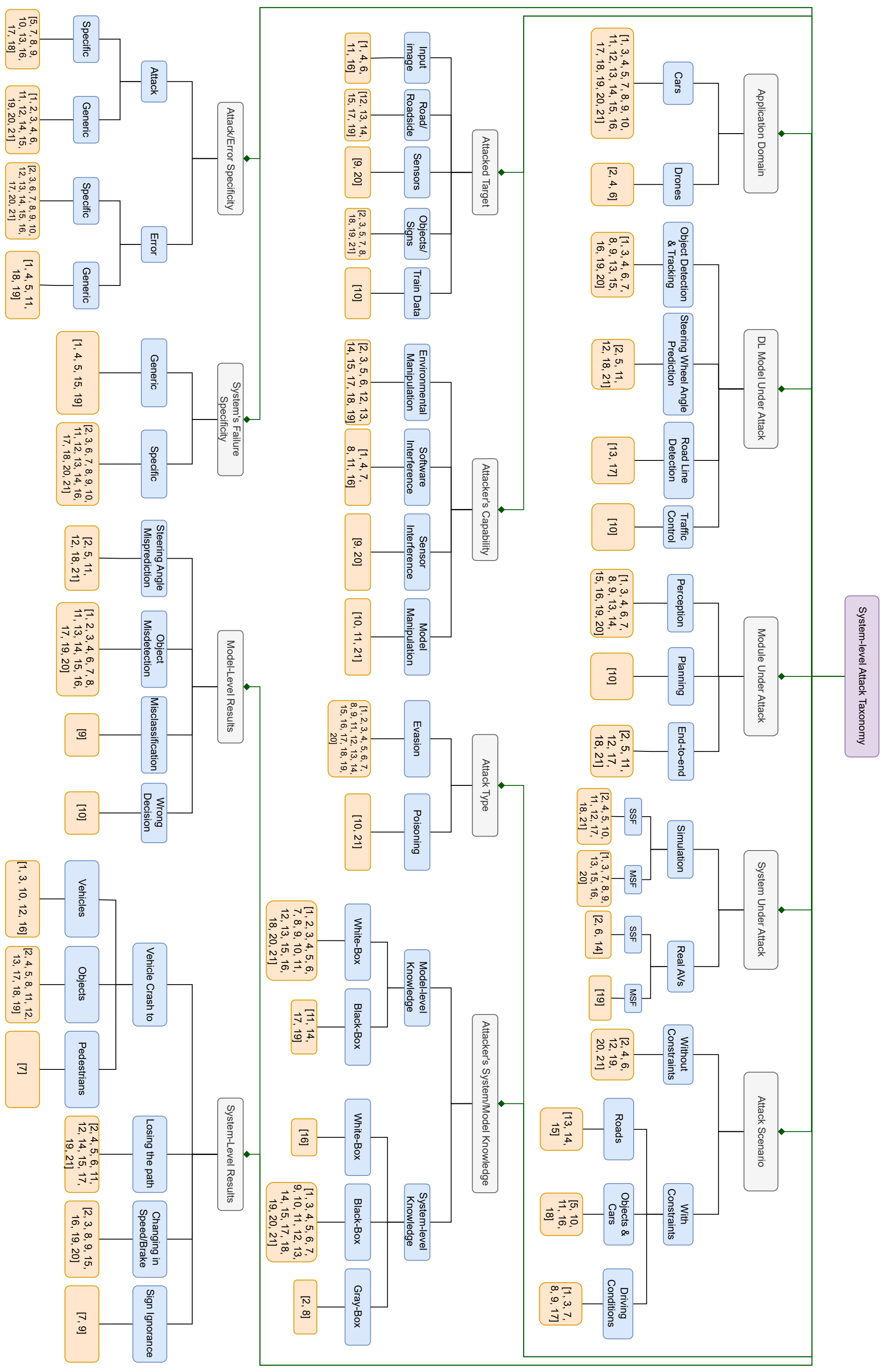

Fig. 2. Taxonomy Tree

covers models that control the car's steering angle. "Road Line Detection" helps the system detect and follow road lines, while "Traffic Control" assists in navigating complex traffic situations. All of these DL models contribute to the autonomous capabilities of vehicles and are part of a modular AV architecture, such as perception, planning, and end-to-end driving. If we compare the models under attack covered by the papers included in the taxonomy with the AV models and modules reported by the LLM in Table 1, we can notice that most types of modules are covered. However, modules for localization and mapping have not been subjected to system-level security testing in any of the included papers. This might be due to the difficulty associated with faithful simulation of such modules. Another missing module type is associated with the low level "control" functionality, often implemented via reinforcement learning. This indicates a general preference of security testing papers toward supervised learning, as opposed to reinforcement learning. A similar preference seems to hold for model-level adversarial attacks, which are typically crafted against neural networks trained in a supervised setting.

(3) **System Under Attack:** The autonomous system under attack, such as a driving agent like Baidu Apollo in the LGSVL simulator or the driving agent in the CARLA simulator. "System Under Attack" has two subcategories, "Simulation" and "Real AVs", respectively indicating that the system under attack is simulated or is a real-world autonomous vehicle. "Baidu Apollo" [34] and "CARLA's Agent" [33] represent examples of autonomous systems implemented in the LGSVL [35] and CARLA simulators — the most popular simulation environments in our list of papers. Other less popular simulators include BeamNG [36] and Udacity [37]. "Tesla" is instead an example referred to by papers that used a real Tesla vehicle to test their attacks. As systems in AVs may use either single or multi-sensor fusion to make autonomous decisions, in this category, we distinguish between them by using *MSF* for multi-sensor fusion systems and *SSF* for single-sensor fusion systems. If we compare the systems under attack covered by the papers included in the taxonomy with the simulation environments reported by the LLM in Table 1, we can notice that simulators for several AV surfaces, including sea, underwater and space, are not used in the included papers, or are used for another surface (e.g., Gazebo can be used to simulate air, sea, underwater and space, but in the included papers it was used only to simulate AVs operating in the air – specifically, drones). Similarly to the remark made above for the application domain, the missing simulation environments point to the need for further research to bring system-level attacks to simulators operating in the sea, underwater, or in space.

(4) **Attack Scenario:** The scenario in which the attack is designed to be effective, such as when a car is at an intersection or stuck in a traffic jam. Hence, the "Attack Scenario" category provides details on the *necessary conditions for an attack*, i.e., the conditions that must be satisfied in order for an attack to be successful. Some papers did not specify any constraints, so they fall under the "Without Constraints" subcategory. For those requiring specific conditions, we created the "With Constraints" subcategory, which includes three further distinctions. The "Roads" subcategory covers attacks that need modifications to the road itself. The "Objects & Cars" subcategory includes attacks that require objects, such as billboards or non-autonomous cars, to be positioned in specific places. Lastly, the "Driving Condition" subcategory relates to papers that do not require physical changes but need the AV to be in a specific scenario, such as approaching a traffic light, being at an intersection, or navigating a turn.

(5) **Attacked Target:** The specific target of the attack, such as a billboard where to place adversarial noise or a road where to add adversarial lines. If an attack operates as malicious software that directly alters the input (e.g., image) received by the DL model, then the attacked target would be e.g. the input image. The subcategories of "Attacked Target" concern the

specific elements that the attacker needs to change, manipulate, or perturb. This can include the "Input Image" fed into the DL model, the "Road" or "Roadside", the AV's "Sensors", "Objects" like billboards or traffic "Signs" like stop signs, and the "Training Data" used to train the model before deployment.

(6) **Attacker's Capability:** The capabilities required by the attacker to execute the attack, such as the ability to alter the physical environment or to install malware into the system. The "Attacker's Capability" category has four subcategories. Attacks requiring changes to the environment, such as placing objects on the road, fall under the "Environmental Manipulation" subcategory. Those needing alterations at the software level, like acting as a man in the middle, are categorized under "Software Interference". Attacks that interfere with the sensors' regular functionality, such as directing lasers at them, are included in the "Sensor Interference" subcategory. Lastly, those requiring manipulation of the model, such as adding backdoors to the training data, fall under the "Model Manipulation" subcategory.

(7) **Attack Type:** The attacker's general approach, such as altering training data ("Poisoning") or producing mis-prediction at inference time ("Evasion").

(8) **Attacker's Knowledge:** The attacker's knowledge about the system or model, split into white-, black-, or grey-box knowledge:
- "Model-level White-Box": Full access to the code used to create the model and to the dataset used to train/test the model.
- "Model-level Black-Box": The attacker can only know the input and output vectors accepted/produced by the model.
- "System-level White-Box": The attacker has unrestricted access to the entire system, including the source code of the system, the hardware specs, DL models, simulator (if any) and datasets.
- "System-level Black-Box": The attacker's knowledge is limited to input (e.g., sensor) and output (e.g. actuator) data, with no visibility into the internal system components.
- "System-level Grey-Box": The attacker has some insights into the system but not to the full extent of a white-box setting (e.g., the code of some components is available, but not the code of the entire system).

(9) **Attack/Error Specificity:** This category includes Attack-Specific or Attack-Generic, as well as Error-Specific or Error-Generic. Some attacks are highly specific, targeting only one particular object or aspect (e.g., only traffic signs are attacked), which makes them "Attack-Specific". On the other hand, the "Attack-Generic" subcategory represents a more general type of attack, which can be applied to a wider range of targets (e.g., any object on the side of the road). Regarding the model mis-predictions resulting from the attack, "Error-Specific" attacks aim to induce a particular output or trigger unique misbehavior in the model (e.g., confusing a 60 speed limit with a 90), while "Error-Generic" attacks allow any kind of misbehavior of the model (e.g., mis-classifying a traffic sign in any possible way).

(10) **System's Failure Specificity:** Similar to the previous category, this one indicates whether the attacker expects a particular system-level failure. For instance, if the attacker aims to make the vehicle suddenly brake, the attack is considered "Failure-Specific". Alternatively, a "Failure-Generic" attack aims at a general system failure, without specifying the exact expected outcome.

(11) **Model-level Results:** The outcome of the DL model resulting from the attack, such as a mis-classification or mis-detection by the object classifier or object detection component of the autonomous vehicle. In this category, papers causing a wrong steering angle prediction fall under the "Steering Angle Misprediction" subcategory. Those causing the object detection module to fail at detecting objects correctly are categorized under "Object Misdetection". The

"Misclassification" subcategory includes e.g. a paper where traffic light colors are incorrectly classified. Finally, the "Wrong Decision" subcategory is for instances where models make incorrect decisions, such as in traffic jams.

(12) **System-level Results:** This category indicates the outcomes of the attack at the system level, such as collisions with other cars, collisions with the environment, or sudden braking, resulting from model-level misbehavior propagated into system-level faults. For the "System-Level Results", we have four subcategories. Some papers cause AV cars to crash into various things. Therefore, under the "Vehicle Crash To" subcategory, we have three more subcategories to specify what the AV crashed into. If the AV crashes into another vehicle (often a non-autonomous car), it falls under the "Vehicles" subcategory. The "Objects" subcategory is for instances when the AV hits objects such as road curbs, billboards, traffic cones, etc. There's also a subcategory for cases where the AV crashes into "Pedestrians". For results that don't fall under "Vehicle Crash To" we have additional subcategories. The "Losing the Path" subcategory covers cases where the AV loses its path and goes into another lane or off-road. The "Changing in Speed/Brake" subcategory includes attacks that cause the AV to accelerate, decelerate, or suddenly brake. Lastly, the "Sign Ignorance" subcategory is for instances where the AV ignores stop signs or traffic lights.

7 Discussion

In this section, we first answer the research questions and then point out to the takeaways of this work and the future research directions.

7.1 Answers to Research Questions

7.1.1 RQ1 [Attack features]. Figure 3 illustrates the distribution of papers across the taxonomy tree. It reveals substantial differences in the number of papers associated with some categories.

Figure 3a illustrates that the perception module is the most frequently targeted in attacks [1, 3, 4, 6–9, 13–16, 19, 20]. End-to-end models follow as the second most common target [2, 5, 11, 12, 17, 18, 21]. However, only a single attack was recorded on the prediction and planning module, while no attacks were reported on the DL components of the localization or control modules. Since the perception module is the first to process sensor data in an AV, any compromise at this stage can lead downstream modules to make incorrect decisions, ultimately resulting in faulty control outputs. End-to-end models are also highly vulnerable, as they bypass preprocessing and act directly on raw sensor inputs, increasing their susceptibility to adversarial manipulation.

Figure 3b shows that attacks are fairly evenly distributed between MSF and SSF systems. Although MSF systems are often considered more secure due to their integration of multiple sensor types [62], this study reveals that they remain vulnerable to well-crafted adversarial attacks.

In Figure 3c, we observe that only three papers focus on the application domain of autonomous drones, compared to 18 papers on autonomous cars. As autonomous drones continue to gain traction across various sectors, we expect such proportion to change in the future. The notable lack of research on system-level adversarial attacks in other application domains, such as autonomous trucks, buses, ships, spacecraft, and more, highlights a clear gap in the current literature, pointing to a whole research area that could be investigated more deeply in the future.

Figure 3d shows that most attacks target the object detection and tracking models of AVs, indicating that this component is the most vulnerable part of an AV system. Steering wheel angle prediction was also a notable target for attacks, as it is influenced by numerous factors on the road that can be the target of an attack [5, 11, 12, 18, 21].

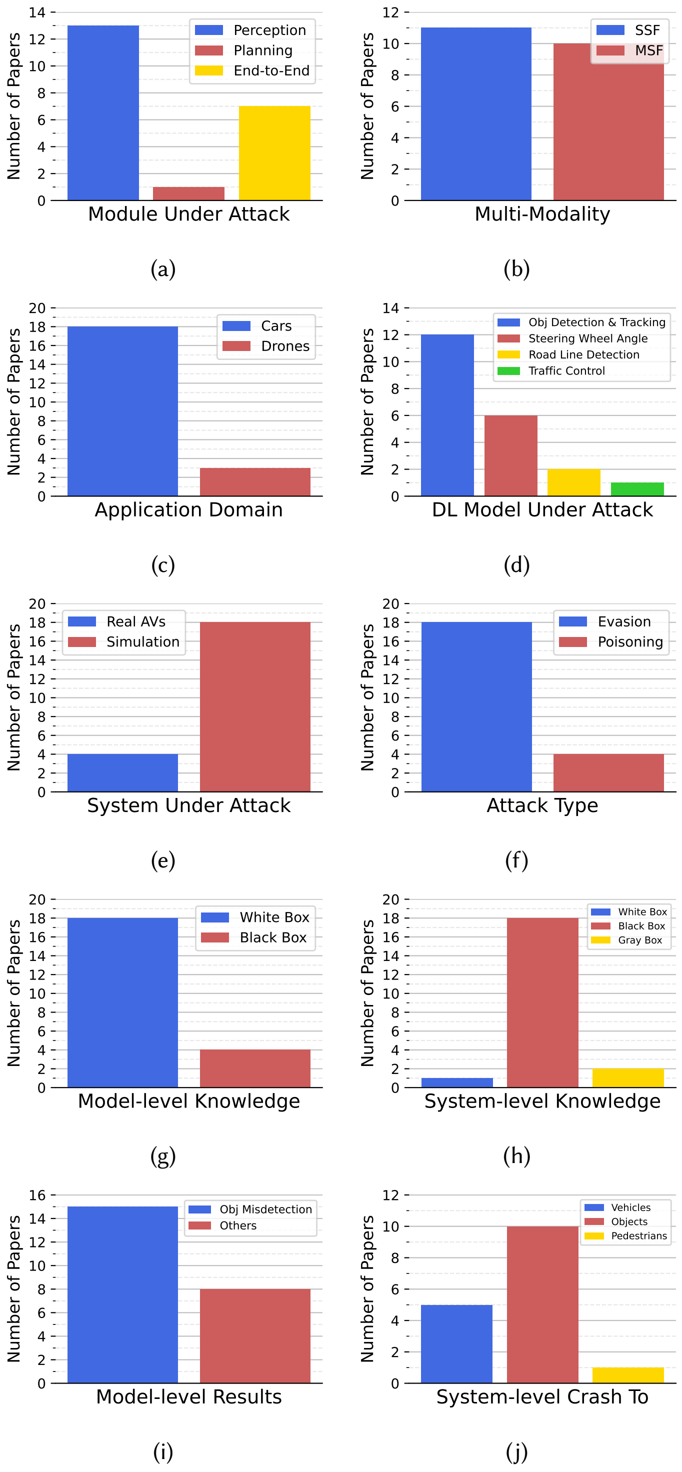

Fig. 3. Paper distribution in the taxonomy tree

Figure 3e clearly shows that the majority of papers focused on deploying attacks in a simulation environment. This trend may be due to the inherent dangers of the attacks, or legal constraints in various countries, making simulation environments more feasible for conducting such research.

Figure 3f illustrates the approach chosen by attackers to deploy their attacks. Evasion attacks are prevalent, likely because poisoning attacks require access to the model's dataset before training. This scenario is hard to achieve in the real world, making evasion attacks more realistic and feasible.

In Figure 3g, the model-level knowledge is predominantly considered to be white-box. While this may not be very practical in real-world scenarios, it is often necessary for attackers to make their attacks effective. In this respect, the four papers [11, 14, 17, 19] that utilized a black-box knowledge approach would represent more feasible options for real-world attacks.

Figure 3h addresses the system-level knowledge required for attacks. It is noteworthy that almost every attack does not require in-depth knowledge or access to the source code of the system. On one hand, this might be due to the practical difficulties that prevent the acquisition of white-box or grey-box knowledge of complex, possibly proprietary, systems. On the other hand, while at the model level, white-box knowledge is often a prerequisite that makes the attack otherwise impossible, at the system level, plain knowledge of the environment elements and system input is often enough to introduce manipulations that cause a model mis-prediction and possibly a system failure.

In line with Figure 3d, Figure 3i also shows that the outcome of attacks on the DL component is predominantly the misdetection of targets or objects the AV is trying to recognize. These objects can include traffic signs, cars, pedestrians, etc. This highlights a significant vulnerability in the object detection and recognition capabilities of autonomous vehicles.

Figure 3j suggests that most crashes involving AVs were with objects such as obstacles on the road. The figure indicates that only one paper reported crashes involving pedestrians. This could be related to the difficulty in effectively targeting pedestrians as part of an attack.

RQ1 [Attack Features]: The distribution of papers across the taxonomy categories is notably unbalanced. System-level attacks on AVs are primarily focused on cars and, to a lesser extent, drones. Other types of AVs, such as tractors, spacecraft, and maritime surface ships, are largely absent from the current research landscape. Most attacks target the *perception* module of AVs, with significantly less attention given to other critical components like *planning*, *localization*, and *control*. While one might assume that AVs employing MSF are more resilient to adversarial attacks, this taxonomy reveals that MSF systems are still vulnerable. The majority of studies focus on simulated car environments, targeting components such as object detection/tracking and steering angle prediction, with the primary goal being mispredictions (evasion). These attacks are often conducted with white-box access to the model but only black-box knowledge of the system. At the model level, attacks are typically designed to cause object misdetection, whereas at the system level, the aim is often to cause the AV to crash into other vehicles or obstacles.

7.1.2 RQ2 [Attacked Components]. By examining the taxonomy tree, we observe that the DL models under attack consist of object detection & tracking, steering angle prediction, lane detection, and traffic control models. Most of them support the perception tasks carried out by the AV, by identifying relevant features in the perceived input and manipulating them to cause a mis-prediction. Often, the perceived input is a camera image [1, 4, 6, 11, 16], making it possible to reuse the wide range of attacks investigated in previous works on model-level evasion against images, achieved via adversarial noise patches [26]. This is confirmed by the knowledge required by these attacks, which is white-box at the model level and black-box at the system level (see Figures 3g, 3h). Hence, the problem becomes how to manipulate the simulation environment in which the AV operates in order to introduce the attack patches that are known to trigger a mis-prediction at the model level, assuming that the induced error will propagate inside the system, which is known only as a black box, to eventually cause a system failure.

The systems under attack include Baidu Apollo, CARLA's driving agent, Tesla, and several other AV systems. However, most attacked systems are executed within a simulator (see Figure 3e). There are several practical reasons to prefer simulated AVs over real ones when experimenting with new attacks against AVs, including: (1) simulations are cheaper, faster and not affected by safety constraints; (2) simulations are less constrained and offer higher flexibility in the manipulation of the environment, which could be very expensive or infeasible in the real world; (3) simulations provide more information about the execution, e.g., obtained by monitoring and logging different types of execution/simulation data, which is useful to fine-tune an attack. Such information might be unavailable in the real world.

RQ2 [Attacked Components]: The most attacked components are image-processing neural networks in the perception module, for which several model-level attacks are readily available in the literature. The problem is then how to manipulate the environment such that the model-level input resembles an adversarial input and how to ensure the propagation of a model-level error to a system-level failure. This seems easier to achieve in simulation than in the real world because a simulator offers higher flexibility, more information, and is generally cheaper, faster, and not dangerous for human beings. Finally, almost every attack on MSF systems required a white-box knowledge of the DL model.

7.1.3 RQ3 [Threat Models]. The most common assumption is that the attacker has white-box access to a DL model used in an AV system, while the system itself is known only as a black-box. Under the assumption of white-box model knowledge and black-box system knowledge, the papers included in our taxonomy explored various threat models:

(1) **Outside Attack**: The attacker cannot manipulate the DL model input. Hence, the attacker can only manipulate the environment [2, 5, 6, 12–15, 17–19] or can interfere with the sensors [9, 20] to supply an adversarial input to the model.
(2) **Inside Attack at Inference Time**: The attacker can manipulate directly the DL model input at inference time [1, 4, 7, 8, 11, 16]. This can be achieved e.g. by installing some malware that can intercept and occasionally replace the actual model input with an adversarial one.
(3) **Inside Attack at Training Time**: The attacker can manipulate directly the dataset used to train the DL model [10, 11, 21]. This can be achieved e.g. by obtaining unauthorized access to the victim's computers or with the complicity of an employer working at the victim's company.

Under the assumption of black-box model knowledge, nearly all attacks were **Outside Attacks**, manipulating the environment, such as the road, with adversarial input [14, 17, 19]. The only exception is an attack that employs a weight injection tool to generate perturbations directly on images [11]

The above-listed threat models require an increasing amount of capabilities and permissions on the attacker's side, with the Outside Attack threats being the easiest to mount and the Inside Attack threats the most difficult and challenging. It should be noticed that not all papers are explicit about the underlying threat model they assume. As a result, we often had to infer it from the algorithmic description of the attack. As the risk associated with an attack depends not only on the damage it can cause, but also on the probability that an attacker can successfully mount the attack, explicit knowledge of the threat model is important to understand and possibly estimate the likelihood of an attack.

RQ3 [Threat Models]: Threat models span from limited capabilities of manipulation of the environment where an AV operates, to the possibility of directly corrupting or replacing input values at inference or even at training time. Such a wide range corresponds to different degrees of attack likelihood and realism. Papers often lack explicit information on the assumed threat model, which makes it difficult for the reader to assess the feasibility of an attack in a given context. Although modifying the environment might appear easy, most attacks rely on white-box access to the DL model. This makes it less realistic to believe that such attacks could occur in real-world scenarios, unless the model is leaked or the attacker's surrogate model closely resembles the target model.

7.1.4 RQ4 [Consequences]. Most attacks that cause a misprediction of the steering angle result in the vehicle losing its path and sometimes hitting road objects, such as side walls [5, 11, 12, 18, 21]. On the other hand, object misdetection often causes the vehicle to fail to see an object or to perceive a fake object as real, leading to unexpected changes in speed or crashes [1, 3, 8, 11, 13, 15–17, 19, 20]. Misclassifications by the model and incorrect decisions can cause the car to ignore traffic signs or collide with other vehicles [9, 10].

There are two main chains of attack consequences from the input to a system failure:

(1) **Targeted Attacks**: one or more adversarial inputs are fed into the DL model and cause a misprediction, which goes in a specific direction, aimed at producing a specific system-level failure [2, 8–14, 16–20, 20, 21]. For instance, a patch is added to the environment in order to revert the numerical sign of the steering angle prediction, which is supposed to make the car go off its lane [17].

(2) **Untargeted Attacks**: one or more adversarial inputs are fed into the DL model and cause an arbitrary misprediction. Propagation to an arbitrary system failure is conjectured and often possible, but it is not targeted explicitly [1, 4, 5, 15, 19]. For instance, perturbations are added to the input image to cause arbitrary, incorrect bounding boxes. As a result of the model-level misclassification, the self-driving car misbehaves [1].

Clearly, the targeted attack chain is more effective as it causes a model-level mis-prediction that is specifically functional to an overall misbehavior of the whole system. However, such targeted chains require deeper knowledge of the system, which is often unavailable or incomplete. Hence, often papers adopt the untargeted attack chain and then just observe and evaluate its effectiveness empirically, with no warranty of success.

RQ4 [Consequences]: There are two main approaches to achieve the system-level consequences of an attack: (1) specific model-level mis-predictions are targeted in specific execution contexts, to ensure the desired consequence at the system level; (2) arbitrary model-level mis-predictions are triggered, and system-level consequences are just observed, but not controlled.

7.2 Implications for Practitioners and Researchers

Based on our analysis of the papers and based on the taxonomy extracted from them, we can distill a few lessons and implications that are potentially useful to practitioners and represent interesting opportunities for researchers.

(1) **Vulnerability of Image Processing Components**: Based on the categories of "DL Model Under Attack" and "Attacked Target", most system-level attacks manipulate the environment

in order to affect some image processing components, which are vulnerable to adversarial attacks. It should be noted that incorporating a DL component for image processing requires viewing environmental objects as potential threats and designing defenses accordingly.

(2) **Vulnerability of Modules and Systems:** The perception module is the primary target of most attacks, while other modules, such as planning, localization, and control, are rarely targeted directly. While end-to-end AV systems were expected to be vulnerable to adversarial attacks, as confirmed in the taxonomy, the surprising finding is that MSF systems are also susceptible. However, almost every attack on MSF systems requires white-box knowledge of the DL model and was not designed in a black-box way. We can conclude that, when designing defenses, it is important to consider that the perception module in MSF autonomous vehicles is the most frequently targeted module.

(3) **Attack Propagation**: Wang et al. [7] highlighted that some existing model-level object evasion attacks in AV fail to produce system-level effects. Therefore, it is crucial to analyze the propagation of these attacks, as the effectiveness of a model-level attack on a system depends on the propagation chain it exploits. During our review of related papers, we identified a lack of research addressing why certain attacks on DL models do not escalate to system-level failures. This gap represents a promising area for further investigation in the field. Filling this gap to understand why an attack leads to system-level failures can help in identifying both attacks and defenses more efficiently and accurately, potentially leading to more secure AVs.

(4) **System-level knowledge**: Nearly every attack in the reviewed papers was conducted without any prior knowledge of the system, classifying them as black-box attacks. This demonstrates that effective attacks can still be carried out without access to the source code or detailed information about the autonomous vehicle. Consequently, it should not be assumed that concealing the AV's system information would inherently enhance its security.

(5) **Additional Attack Targets**: Existing attacks commonly involve creating adversarial objects or fake road lines to mislead the perception components of an AV. These objects are static and do not move, hence, defense techniques should take into account environmental objects and elements more carefully, as they can play a significant role in enabling or triggering attacks. However, Hanfeld et al. [6] demonstrated an attack using a moving drone, showing that dynamic objects, such as autonomous or human-controlled units, can also be employed by attackers to carry out their attacks. This points to a potential direction for future work, exploring the use of dynamic objects (e.g., moving vehicles or people) as carriers of adversarial patches.

8 Related Work

Biggio & Roli [26] proposed a comprehensive survey investigating the vulnerability of ML and the development of suitable countermeasures within the research field of adversarial ML. This work provides an in-depth overview of the evolution of this research area over the past decade and beyond, starting from early investigations into the security of non-DL algorithms to more recent studies aimed at understanding the security properties of DL algorithms in computer vision and cybersecurity. The authors reviewed the main threat models and attacks defined in this context and discussed the significant limitations of current research, as well as the future challenges in designing more secure learning algorithms.

In their subsequent work, Cinà et al. [27] provided a comprehensive survey that systematizes poisoning attacks and defenses in ML over the past 15 years. They begin by categorizing the current threat models and attacks, and then organize the existing defenses accordingly. While their focus is primarily on computer vision applications, they argue that their systematization also encompasses state-of-the-art attacks and defenses for other data modalities. The main difference between these two papers and our work is that their surveys cover model-level attacks, while we are interested in system-level attacks. Our paper aims to provide categorized information for readers interested in understanding how to mount an attack against a whole system, not just one of its DL components in isolation, and how this can lead to system-level failures.

Shen et al. [30] systematized the knowledge within the growing area of semantic autonomous driving AI security. They analyzed a total of 53 papers, categorizing them based on research aspects critical to the security field such as the attack/defense targeted AI component, attack/defense goal, attack vector, attack knowledge, defense deployability, defense robustness, and evaluation methodologies. While this paper is the closest to our work, there are remarkable differences. First of all the focus, which is *semantic* AI security. As a consequence, although this paper mentions attacks that can cause system-level failures, its approach to collecting and analyzing papers was not systematically focused on all possible system-level attacks, rather just on those exhibiting a *semantic* gap. Hence, they cover the existing literature on system-level attacks to a quite limited extent. Correspondingly, our taxonomy provides more detailed information about the relationship between each attack and the whole system under test, including how the attacks are performed in the environment, which system components they target, the resulting system-level failures, and the propagation chain that allows an attack to manifest itself.

9 Threats to Validity

9.1 Internal Validity

Internal validity threats are caused by the tools and methodology employed to conduct the study. We have identified the following internal validity threats: (1) the query string; (2) the filtering phase; (3) subjectivity of categorization; (4) exclusion criteria.

The *query string* was crafted with a focus on attacks on autonomous vehicles and refined iteratively based on the number of retrieved papers and the presence of pivotal papers in our results. Running the query string from 2017 to 2024 delivered an initial list of approximately 9,700 papers. Although changing the query and pivot papers would alter this initial number, our subsequent steps — filtering and snowballing — compensate for this effect. So, we expect that any replications would yield results similar to ours.

In the *filtering phase*, we excluded venues that did not meet our desired quality standards or fell outside the scope of our taxonomy. We also white-listed some high-reputation, related venues to ensure their inclusion, as determining the scope based solely on titles was not feasible. While changing the chosen venues might affect the results, manual inspection of the retrieved papers, along with those identified in the snowballing phase, ensured that all included papers were high-value and fully in the scope of our work.

The selection of categories was done by starting from previous model-level surveys and creating new categories when needed. The resulting categories were defined through consensus in weekly meetings held among the authors. The primary task involved comprehensively reading the papers and categorizing them. Consequently, each paper was reviewed by at least two assessors, and weekly meetings were held to resolve any differences in categorization. These measures were taken to ensure accurate categorization and mitigate any *subjectivity threat* possibly affecting the categorization.

We had two main criteria for excluding papers. The first criterion was to exclude attacks not targeting any DL model. The second criterion was to exclude attacks evaluated only at the model level. While reviewing the papers, we encountered some that targeted the sensors rather than the DL component, such as those that blinded the sensors. In these cases, the attack could not be attributed to any vulnerable DL model. Thus, these papers were discarded. Some papers claimed real-world evaluation but only involved mounting a camera on a car and taking pictures of traffic signs and surroundings. Since no autonomous vehicle was involved, this merely resulted in creating their own dataset, leading to their exclusion from our taxonomy. This manual exclusion of papers was carried out through meetings and discussions, and we tried to stick to the two criteria explained above as strictly as possible. However, we acknowledge that other researchers might have different perspectives.

9.2 External Validity

External validity threats affect the generalizability of our findings beyond the study conducted for this paper. A replication of this study following our methodology is not ensured to produce exactly the same list of papers, taxonomy categories and taxonomy mapping. However, we have tried to document all the key decisions made in the process, describing the adopted methodology in detail. We have adopted the best practices available for the execution of a systematic literature review [23]. Hence, we think that our main findings (see our answers to the research questions in Section 6) are to a major extent valid beyond our specific instance of the study.

10 Conclusion

While model-level adversarial attacks have received huge attention from researchers in the last decades, it is only more recently that researchers considered the possibility of injecting attack vectors into elements of the environment where the system under attack operates, to observe a system-level failure, not just a model-level mis-prediction. The literature on system-level attacks has grown quite rapidly in recent years, going in different directions. Hence, this is the time to organize the existing works into a taxonomy. For this purpose, we retrieved 21 more relevant papers from the literature, and created a taxonomy to categorize them. Our key findings indicate an unbalanced distribution of papers across taxonomy categories; a prevalence of attacks against image processing components and the perception module in simulation environments; a variety of threat models, which include black-box attacks and white-box attacks, the latter either at inference or at training time; two main attack chains, consisting of targeted attacks, aimed at specific system-level failures, and untargeted attacks.

We also discussed a number of implications for practitioners and researchers. In the former case, they include guidelines and suggestions for security testing (e.g., on risk assessment and attack effectiveness). In the latter case, they indicate possible directions for future work, such as: performing attacks on DL components beyond image processing ones; designing attacks targeting modules outside perception; addressing the vulnerabilities of both multi and single-sensor fusion systems; investigating the transferability of attacks from a simulator to the real world; propagation of an attack from model to system; system level security testing for the creation of more secure vehicles, including dynamic entities among the environment elements that contribute to a system-level attack.

Conflict of Interest and Data Availability

The implementations, source code, data, and experimental results are publicly available at https://zenodo.org/records/16407709.

Acknowledgments

This work is funded by the European Union’s Horizon Europe research and innovation programme under the project Sec4AI4Sec, grant agreement No 101120393.

References

[1] C. Ma, N. Wang, Q. A. Chen, and C. Shen, "Slowtrack: Increasing the latency of camera-based perception in autonomous driving using adversarial examples," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 5, 2024, pp. 4062–4070.

[2] T. Liu, C. Yang, X. Liu, R. Han, and J. Ma, "Rpau: Fooling the eyes of uavs via physical adversarial patches," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 3, pp. 2586–2598, 2024.

[3] Y. Guo, T. Sato, Y. Cao, Q. A. Chen, and Y. Cheng, "Adversarial attacks on adaptive cruise control systems," in *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*. New York, NY, USA: Association for Computing Machinery, 2023, p. 49–54. [Online]. Available: https://doi.org/10.1145/3576914.3587493

[4] H.-J. Yoon, H. Jafarnejadsani, and P. Voulgaris, "Learning when to use adaptive adversarial image perturbations against autonomous vehicles," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4179–4186, 2023.

[5] M. von Stein, D. Shriver, and S. Elbaum, "Deepmaneuver: Adversarial test generation for trajectory manipulation of autonomous vehicles," *IEEE Transactions on Software Engineering*, 2023.

[6] P. Hanfeld, K. Wahba, M. M.-C. Höhne, M. Bussmann, and W. Hönig, "Kidnapping deep learning-based multirotors using optimized flying adversarial patches," in *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2023, pp. 78–84.

[7] N. Wang, Y. Luo, T. Sato, K. Xu, and Q. A. 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*, 2023, pp. 4412–4423.

[8] Q. Zhang, S. Jin, J. Sun, X. Zhang, R. Zhu, Q. A. Chen, and Z. M. Mao, "On data fabrication in collaborative vehicular perception: Attacks and countermeasures," *arXiv preprint arXiv:2309.12955*, 2023.

[9] C. Yan, Z. Xu, Z. Yin, S. Mangard, X. Ji, W. Xu, K. Zhao, Y. Zhou, T. Wang, G. Gu *et al.*, "Rolling colors: Adversarial laser exploits against traffic light recognition," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 1957–1974.

[10] Y. Wang, E. Sarkar, W. Li, M. Maniatakos, and S. E. Jabari, "Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4772–4787, 2021.

[11] N. Piazzesi, M. Hong, and A. Ceccarelli, "Attack and fault injection in self-driving agents on the carla simulator–experience report," in *Computer Safety, Reliability, and Security: 40th International Conference, SAFECOMP 2021, York, UK, September 8–10, 2021, Proceedings 40*. Springer, 2021, pp. 210–225.

[12] T. Sato, J. Shen, N. Wang, Y. Jia, X. Lin, and Q. A. Chen, "Dirty road can attack: Security of deep learning based automated lane centering under {Physical-World} attack," in *30th USENIX security symposium (USENIX Security 21)*, 2021, pp. 3309–3326.

[13] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. A. Chen, M. Liu, and B. 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)*. IEEE, 2021, pp. 176–194.

[14] P. Jing, Q. Tang, Y. Du, L. Xue, X. Luo, T. Wang, S. Nie, and S. Wu, "Too good to be safe: Tricking lane detection in autonomous driving with crafted perturbations," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3237–3254.

[15] K. Yang, T. Tsai, H. Yu, M. Panoff, T.-Y. Ho, and Y. Jin, "Robust roadside physical adversarial attack against deep learning in lidar perception modules," in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 2021, pp. 349–362.

[16] S. Jha, S. Cui, S. Banerjee, J. Cyriac, T. Tsai, Z. Kalbarczyk, and R. K. Iyer, "Ml-driven malware that targets av safety," in *2020 50th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. IEEE, 2020, pp. 113–124.

[17] A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Attacking vision-based perception in end-to-end autonomous driving models," *Journal of Systems Architecture*, vol. 110, p. 101766, 2020.

[18] S. Pavlitskaya, S. Ünver, and J. M. Zöllner, "Feasibility and suppression of adversarial patch attacks on end-to-end vehicle control," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–8.

[19] B. Nassi, Y. Mirsky, D. Nassi, R. Ben-Netanel, O. Drokin, and Y. Elovici, "Phantom of the adas: Securing advanced driver-assistance systems from split-second phantom attacks," in *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, 2020, pp. 293–308.

[20] Y. Cao, C. Xiao, B. Cyr, Y. Zhou, W. Park, S. Rampazzi, Q. A. Chen, K. Fu, and Z. M. Mao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2267–2281.

[21] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," 2017.

[22] A. Stocco, B. Pulfer, and P. Tonella, “Mind the gap! a study on the transferability of virtual versus physical-world testing of autonomous driving systems,” *IEEE Transactions on Software Engineering*, vol. 49, no. 4, pp. 1928–1940, 2022.

[23] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” 2007.

[24] “Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles,” SAE International, Tech. Rep. J3016_202104, April 2021.

[25] J. Grosman, “Findpapers: A tool for helping researchers who are looking for related works,” https://github.com/jonatasgrosman/findpapers, 2020.

[26] B. Biggio and F. Roli, “Wild patterns: Ten years after the rise of adversarial machine learning,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 2154–2156.

[27] A. E. Cinà, K. Grosse, A. Demontis, S. Vascon, W. Zellinger, B. A. Moser, A. Oprea, B. Biggio, M. Pelillo, and F. Roli, “Wild patterns reloaded: A survey of machine learning security against training data poisoning,” *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–39, 2023.

[28] C. Wohlin, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

[29] H. Zhou, W. Li, Z. Kong, J. Guo, Y. Zhang, B. Yu, L. Zhang, and C. Liu, “Deepbillboard: Systematic physical-world testing of autonomous driving systems,” in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 347–358.

[30] J. Shen, N. Wang, Z. Wan, Y. Luo, T. Sato, Z. Hu, X. Zhang, S. Guo, Z. Zhong, K. Li, Z. Zhao, C. Qiao, and Q. A. Chen, “Sok: On the semantic ai security in autonomous driving,” 2024. [Online]. Available: https://arxiv.org/abs/2203.05314

[31] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, “A survey of deep learning applications to autonomous vehicle control,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 2, pp. 712–733, 2020.

[32] A. Faisal, M. Kamruzzaman, T. Yigitcanlar, and G. Currie, “Understanding autonomous vehicles,” *Journal of transport and land use*, vol. 12, no. 1, pp. 45–72, 2019.

[33] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An open urban driving simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[34] B. A. team, “Apollo: Open source autonomous driving.” [Online]. Available: https://github.com/ApolloAuto/apollo

[35] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Možeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta *et al.*, “Lgsvl simulator: A high fidelity simulator for autonomous driving,” *arXiv preprint arXiv:2005.03778*, 2020.

[36] BeamNG GmbH, “BeamNG.tech.” [Online]. Available: https://www.beamng.tech/

[37] Udacity, “A self-driving car simulator built with unity.” [Online]. Available: https://github.com/udacity/self-driving-car-sim

[38] R. Muller, Y. Man, Z. B. Celik, M. Li, and R. Gerdes, “Physical hijacking attacks against object trackers,” in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2309–2322.

[39] C. Fu, S. Li, X. Yuan, J. Ye, Z. Cao, and F. Ding, “Ad 2 attack: Adaptive adversarial attack on real-time uav tracking,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5893–5899.

[40] Y. Cao, S. H. Bhupathiraju, P. Naghavi, T. Sugawara, Z. M. Mao, and S. Rampazzi, “You can’t see me: Physical removal attacks on {lidar-based} autonomous vehicles driving frameworks,” in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 2993–3010.

[41] Wikipedia contributors, “Unmanned ground vehicle,” https://en.wikipedia.org/wiki/Unmanned_ground_vehicle, 2025, accessed: 2025-07-11.

[42] ——, “Self-driving car,” https://en.wikipedia.org/wiki/Self-driving_car, 2025, accessed: 2025-07-11.

[43] ——, “Unmanned aerial vehicle,” https://en.wikipedia.org/wiki/Unmanned_aerial_vehicle, 2025, accessed: 2025-07-11.

[44] ——, “Unmanned surface vehicle,” https://en.wikipedia.org/wiki/Unmanned_surface_vehicle, 2025, accessed: 2025-07-11.

[45] ——, “Autonomous underwater vehicle,” https://en.wikipedia.org/wiki/Autonomous_underwater_vehicle, 2025, accessed: 2025-07-11.

[46] ——, “Vehicular automation,” https://en.wikipedia.org/wiki/Vehicular_automation, 2025, accessed: 2025-07-11.

[47] Y. Zhou and O. Tuzel, “Voxelnet: End-to-end learning for point cloud based 3d object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[48] S. Shi, X. Wang, and H. Li, “Pointrcnn: 3d object proposal generation and detection from point cloud,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

[49] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[50] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[51] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, “Simultaneous localization, mapping and moving object tracking,” *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, 2007.

[52] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics: Results of the 11th international conference.* Springer, 2017, pp. 621–635.

[53] Webots, "http://www.cyberbotics.com," open-source Mobile Robot Simulation Software. [Online]. Available: http://www.cyberbotics.com

[54] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[55] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. Ieee, 2004, pp. 2149–2154.

[56] A. R. Perry, "The flightgear flight simulator," in *Proceedings of the USENIX annual technical conference*, vol. 686, 2004, pp. 1–12.

[57] J. Penn and A. Lin, "The trick simulation toolkit: a nasa/opensource framework for running time based physics models," in *AIAA modeling and simulation technologies conference*, 2016, p. 1187.

[58] C. Lee and S. Lee, "Vulnerability of clean-label poisoning attack for object detection in maritime autonomous surface ships," *Journal of Marine Science and Engineering*, vol. 11, no. 6, p. 1179, 2023.

[59] C. Lee and S. Lee, "Evaluating the Vulnerability of YOLOv5 to Adversarial Attacks for Enhanced Cybersecurity in MASS," *Journal of Marine Science and Engineering*, vol. 11, 2023.

[60] Q. Huang, C. Gu, Y. Wang, and D. Hu, "Spotattack: Covering spots on surface to attack lidar-based autonomous driving systems," *IEEE Internet of Things Journal*, vol. 11, no. 24, pp. 40 634–40 644, 2024.

[61] OpenAI, "Chatgpt (gpt-4)," https://chat.openai.com/, 2023, accessed: 2025-06-25.

[62] G. Lou, Y. Deng, X. Zheng, M. Zhang, and T. Zhang, "Testing of autonomous driving systems: where are we and where should we go?" in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 31–43.

[63] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 1016–1026.

[64] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 525–11 533.

[65] R. Salay, K. Czarnecki, H. Kuwajima, H. Yasuoka, T. Nakae, V. Abdelzad, C. Huang, M. Kahn, and V. D. Nguyen, "The missing link: Developing a safety case for perception components in automated driving," *arXiv preprint arXiv:2108.13294*, 2021.

[66] S. Burton, "A causal model of safety assurance for machine learning," *arXiv preprint arXiv:2201.05451*, 2022.

[67] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE intelligent vehicles symposium (IV).* IEEE, 2011, pp. 163–168.

[68] W. Li, Z. Wang, G. Wei, L. Ma, J. Hu, and D. Ding, "A survey on multisensor fusion and consensus filtering for sensor networks," *Discrete Dynamics in Nature and Society*, vol. 2015, no. 1, p. 683701, 2015.

[69] L. Heng, B. Choi, Z. Cui, M. Geppert, S. Hu, B. Kuan, P. Liu, R. Nguyen, Y. C. Yeo, A. Geiger *et al.*, "Project autovision: Localization and 3d scene perception for an autonomous vehicle with a multi-camera system," in *2019 International Conference on Robotics and Automation (ICRA).* IEEE, 2019, pp. 4695–4702.

A Tables with Paper Mapping

Below, we report the full tables with the paper mapping produced by the assessors and agreed upon in the consensus meetings.

Table 3. Paper Mapping (Part 1)

Ref	App Domain	DL Model Under Attack	System Under attack	Attack Scenario
[1]	Cars	SORT(Y5), FairMOT, ByteTrack, BoT-SORT (Perception)	Baidu Apollo in LGSVL simulator (MSF)	AVs changing lanes when another car is in the adjacent lane or approaching an intersection with a stop sign where another car is present
[2]	Drones	A variation of DroNet (E2E)	Parrot Bebop 2 Drone and Matlab simulation (SSF)	Drones moving to its destination
[3]	Cars	OpenPilot's Adaptive Cruise Control ACC system (Perception)	CARLA and Baidu Apollo (MSF)	AV driving behind the attacker's vehicle
[4]	Cars, Drones	YOLO v5 (Perception)	Vision-based guidance system in CARLA or AirSim (SSF)	AVs following their target
[5]	Cars	Dave2 (E2E)	An AV vehicle (hopper vehicle) in BeamNG (SSF)	AVs driving past a roadside billboard.
[6]	Drones	PULP Frontnet (Perception)	A human-following drone control system known as the nano multirotor, the Crazyflie by Bitcraze. (SSF)	Autonomous drones following a human target
[7]	Cars	YOLO v2, v3, v5 (Perception)	Baidu Apollo in LGSVL simulator (MSF)	AVs approaching a critical physical road object on a sunny day, such as a stop sign or pedestrian
[8]	Cars	PointPillars, VoxelNet, V2VNet, CoBEVT, FPV-RCNN (Perception)	AVs utilizing LiDAR or GPS information from other cars in Baidu Apollo (MSF)	Multiple AVs driving and jointly performing collaborative perception tasks
[9]	Cars	Nexar &amp; YOLO v4 (Perception)	Baidu Apollo in LGSVL simulator (MSF)	AVs approaching a traffic light
[10]	Cars	DRL in decision-making module of SUMO (Planning)	Microscopic traffic simulator SUMO And intelligent driver model (SSF)	AVs driving in one or two lanes, following the malicious leading vehicle in traffic
[11]	Cars	ResNet-34 (E2E)	Learning by cheating agent in CARLA (SSF)	AVs on a road with pedestrians and other vehicles
[12]	Cars	OpenPilot's Automated Lane Centering (ALC) (E2E)	Automated Lane Centering (ALC) system in LGSVL (SSF)	AVs driving
[13]	Cars	(A5-L, A5-C) In-road obstacle detection (Perception)	Baidu Apollo in LGSVL simulator v5 (MSF)	Av on a single lane road
[14]	Cars	Tesla's APE lane recognition (AutoPilot ECU, Electronic Control Unit) (Perception)	Tesla model S (SSF)	AVs on roads driving into empty spaces in road line markings
[15]	Cars	PointRCNN and PointPillar and PV-RCNN (Perception)	Baidu Apollo in LGSVL simulator (Lincoln MKZ car) (MSF)	AVs driving on a single lane road
[16]	Cars	Kalman Filter in YOLO v3 (Perception)	Baidu Apollo in LGSVL simulator (MSF)	AVs on a road with pedestrians and vehicles in front of the car
[17]	Cars	Conditional Imitation and Reinforcement learning (E2E) in CARLA	End to end driving system in CARLA (SSF)	AVs in turns and intersections
[18]	Cars	DriveNet (an extension of Dave2) (E2E)	Autonomous Driving agent in CARLA (SSF)	AVs driving past a roadside billboard
[19]	Cars	Faster_rcnn_inception_v2 and Tesla's (Perception) model	Tesla Model X HW 2.5/3 &amp;amp; Renault Captur (equipped with Mobileye 630) (MSF)	AVs driving
[20]	Cars	DNN of the Lidar-based (Perception) module in Baidu Apollo	Driving agent in Baidu Apollo (MSF)	AVs driving
[21]	Cars	NN model of Udacity Simulator (E2E)	Udacity Simulator (SSF)	AVs driving

*MSF stands for Multi-Sensor Fusion and SSF stands for Single-Sensor Fusion

Table 4. Paper Mapping (Part 2)

Ref	Attacked Target	Attacker's Capability	Attack Strategy	Attacker's System/Model knowledge	Attack/Error Specificity	Failure Specificity
[1]	Input image	Purturbing the image input to insert fake bounding boxes into it	Evasion	WM / BS *	AG / EG *	Generic
[2]	Environment (anything that can hold a patch horizontally)	Attaching the patch to anything that can hold it horizontally, facing the drone.	Evasion	WM / GS	AG / ES	Specific
[3]	Vehicles and Trucks	Placing the patch on the back of a vehicle and drive it in front of the AV	Evasion	WM / BS	AS / ES	Specific
[4]	Input image	Inserting the malware into the system and training the perturbations on the real training set	Evasion	WM / BS	AG / EG	Generic
[5]	Billboards (placing their own malicious billboard)	Installing a perturbed billboard or altering an existing one	Evasion	WM / BS	AS / EG	Generic
[6]	Input image	Moving a printed patch in front of a drone	Evasion	WM / BS	AG / ES	Specific
[7]	Stop signs and pedestrians	Attaching adversarial patches on stop signs or pedestrians' shirts.	Evasion	WM / BS	AS / ES	Specific
[8]	Objects in the shared feature map	Manipulating shared LiDAR vision data by spoofing or removing objects from it	Evasion	WM &amp;amp; BM / GS	AS / ES	Specific
[9]	Camera	Directing a laser beam towards the camera of the victim	Evasion	WM / BS	AS / ES	Specific
[10]	Training data	Injecting a backdoor into the model by retraining it with poisoned data.	Poisoning	WM / BS	AS / ES	Specific
[11]	Input image	Manipulating the neurons and weights of the NN, or manipulating input images	Evasion	WM &amp;amp; BM / BS	AG / EG	Specific
[12]	Roads	Placing patches on the road surface	Evasion	WM / BS	AG / ES	Specific
[13]	Roads	Placing a 3D traffic cone that resembles a broken cone on the road	Evasion	WM / BS	AS / ES	Specific
[14]	Roads	Painting or attaching perturbations on the road surface	Evasion	BM / BS	AG / ES	Specific
[15]	Roadsides	Put adversarial objects on the roadside	Evasion	WM &amp;amp; BM / BS	AG / ES	Generic
[16]	Input image	Installing malware to gain access to the live camera feed and modify its ourput	Evasion	WM / WS	AS / ES	Specific
[17]	Roads	Attaching stickers or painting adversarial lane markings on the road.	Evasion	BM / BS	AS / ES	Specific
[18]	Billboard	Placing adversarial patches on billboards located on the right side of the roadside	Evasion	WM / BS	AS / EG	Specific
[19]	The environment (anywhere with a smooth surface)	Projecting a phantom using a projector or embedding the phantom on a digital advertising billboard	Evasion	BM / BS	AG / EG	Generic
[20]	LIDAR Sensor	Injecting spoofed LiDAR data points by shooting lasers at the AV	Evasion	WM / BS	AG / ES	Specific
[21]	Billboards	Retraining the model with Trojan data and attaching it to billboards.	Poisoning	WM / BS	AG / ES	Specific

*WM and BM stand for white-box model-level knowledge and black-box model-level knowledge, respectively, whereas WS, BS, and GS stand for white-box, black-box, and gray-box system-level knowledge. In some cases, multiple attacks were introduced, which is why their model-level knowledge has two values.
*AG and AS stand for attack generic and attack specific, while EG and ES stand for error generic and error specific.

Table 5. Paper Mapping (Part 3)

Ref	Model-level results	System-level results (Failure Propagation)
[1]	Losing detection due to increased latency and subsequently tracking of the target object	Crashing into another car
[2]	Misdetection of objects and Misprediction of the steering wheel angle	Crashing to objects, freezing or going off-route
[3]	Misdetection of the vehicle in front	Acceleration and crash into the car in front
[4]	Misdetection of the correct coordinates of the target bounding box	Losing path, stability and collision to surrounding obstacles
[5]	Misprediction of the steering wheel angle	Going off-road
[6]	Misclassification of the adversarial patch as a real human	Following the adversarially patched image and failing to detect the real human
[7]	Failure to detect stop signs and pedestrians	Running a stop sign and colliding with pedestrians
[8]	Detecting fake objects as real while failing to detect a real object	Stopping unexpectedly or causing collisions with objects
[9]	Misclassification of traffic light colors	Running a red light, causing a car crash or emergency stop, which leads to the vehicle freezing
[10]	Failing to predict the correct reaction to traffic.	Causing a collision with the car in front
[11]	Misprediction of the steering wheel angle and failure to detect traffic lights	Losing lane, going offroad, crashing into buildings, ignoring traffic lights, going to crossroads
[12]	Misprediction of the steering wheel angle	Driving off-road, colliding with road curbs, crashing into obstacles, and having a car crash with oncoming traffic
[13]	Failing to detect an adversarial object	Crashing into the adversarial object
[14]	Detecting the perturbation as a real line	Following the fake line into oncoming traffic
[15]	Detecting adversarial objects as real cars	Stopping completely or suddenly changing lane
[16]	Misdetection or failure to detect cars or pedestrians	Emergency braking or unsafe acceleration leading to a car crash
[17]	Incorrectly detecting lines (wrong control decisions)	Moving into another lane or veering out of bounds, resulting in collisions with road walls
[18]	Misprediction of the steering wheel angle	Collision with the poster
[19]	Detecting phantoms as real road signs or real-world objects	Acting according to the detected object (Brake or decelerate or follow a phantom lane into collision or into the upcoming lane and, etc...)
[20]	Detecting nonexistent obstacles	Emergency braking or the vehicle freezing
[21]	Misprediction of the steering wheel angle	Moving to the right and then going off the road