That Person Moves Like A Car: Misclassification Attack Detection for Autonomous Systems Using Spatiotemporal Consistency

Yanmao Man[1], Raymond Muller[2], Ming Li[1], Z. Berkay Celik[2], and Ryan Gerdes[3]

[1]University of Arizona, *{yman, lim}@arizona.edu*
[2]Purdue University, *{mullerr, zcelik}@purdue.edu*
[3]Virginia Tech, *rgerdes@vt.edu*

Abstract

Autonomous systems commonly rely on object detection and tracking (ODT) to perceive the environment and predict the trajectory of surrounding objects for planning purposes. An ODT's output contains object classes and tracks that are traditionally predicted independently. Recent studies have shown that ODT's output can be falsified by various perception attacks with well-crafted noise, but existing defenses are limited to specific noise injection methods and thus fail to generalize. In this work we propose PercepGuard for the detection of misclassification attacks against perception modules regardless of attack methodologies. PercepGuard exploits the spatiotemporal properties of a detected object (inherent in the tracks), and cross-checks the consistency between the track and class predictions. To improve adversarial robustness against defense-aware (adaptive) attacks, we additionally consider context data (such as ego-vehicle velocity) for contextual consistency verification, which dramatically increases the attack difficulty. Evaluations with both real-world and simulated datasets produce a FPR of 5% and a TPR of 99% against adaptive attacks. A baseline comparison confirms the advantage of leveraging temporal features. Real-world experiments with displayed and projected adversarial patches show that PercepGuard detects 96% of the attacks on average.

1 Introduction

Object detection and tracking (ODT) is an essential component in the perception module of an autonomous system, such as self-driving cars [1–3], robots [4], drones [5], etc., as they need to perceive the environment and understand the dynamics of surrounding objects (what/where they are and where they are going) so that actions can be taken accordingly. In Baidu Apollo [1], for example, cameras are used to capture video frames that are fed to machine learning (ML) algorithms for ODT, the results of which are then used for decision-making, e.g. to stop the car for a pedestrian who is, or is about to, cross the street.

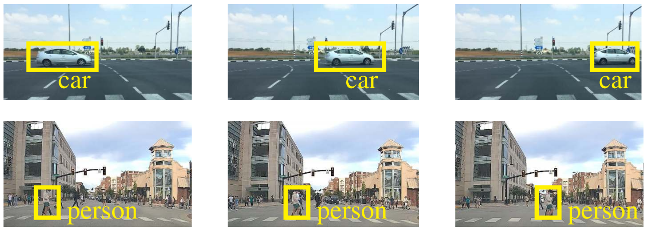


Figure 1: Typically, a person moves slower than a car. Also, their bounding boxes are more vertically thin than a car's.

Recent work has explored attacks against perception approaches/pipelines by physically modifying objects [6–9] or by manipulating the sensing mechanisms (e.g., noise produced by (in)visible light [10–12], acoustic signals [13], radio-frequency electromagnetic waves [14, 15], etc.) such that the subsequent ML algorithms using the output of the sensor produces results that are intended by the attacker. The attack goal of these *perception attacks* against object detection algorithms can be generally categorized into three types: object disappearance [7], object creation [9], and misclassification of detected objects [6]. As a consequence of a particular false perception result, inappropriate decisions/actions may be taken by the planning module of an autonomous system that could impact safety; e.g., a car stops on a highway because it recognizes the car in front of it as a person [16].

Among these aforementioned attacks, this work is focused on misclassification attacks against the vision-based object detectors and trackers of autonomous systems. In particular, we consider an attacker who has white-box knowledge of the system, and aims to change the classification result of the target object by inducing physically-realizable adversarial noise to video frames. E.g., an attacker may place an LCD monitor on the back of a car and display the noise so that the following car recognizes it (the predecessor) as a person [17].

Most existing defenses against these attacks are either specific to the sensing modality (e.g., LiDAR [11], GPS [18], IMU [19]) or, for those that do consider vision, often assume threat models that are specific to one particular attack methodology, e.g., norm-bounded attacks [20] or adversarial patch attacks [21]. The root cause of such specificity is that these defenses focus on the sensing *input* to the ML models, therefore once a new type of attack appears, a new defense is required.

Instead of focusing on the input, in this work we aim to develop a more general defense approach by validating the *output* of the ODT algorithms in order to detect misclassification attacks. For each detected object the ODT system outputs predictions for both the class and trajectory (represented by a *track* in the form of a sequence of bounding boxes over time). Our main idea is to verify the spatiotemporal components of the track by cross-checking with the class prediction; i.e., the perceived spatial and temporal attributes of different object types are unique. For example, for an object to be classified as a person, it has to not only look like a person but also move like a person. In other words, there is an inherent connection between classes and tracks but existing ODT systems predict them separately. Our approach, on the other hand, fuses both together in order to verify the consistency between them; when the consistency is lacking, an attack is indicated.

As a motivating example, we consider a self-driving car that uses a vision-based perception system (Fig. 1). When considering a single frame, we note the shape and location (spatial property) of an object class can be fingerprinted. For instance, the bounding box of a person is usually more vertically thin than of a car, and when the ego-vehicle is driving the box typically appears on the edges of the frame (on the sidewalks) rather than at the center. Furthermore, when considering multiple frames over time, the temporal property of bounding box dynamics is distinctive among different object classes. For example, a car driving across an intersection typically moves faster than a pedestrian does.

Although such empirical knowledge seems intuitive to humans, in order to build an effective defense, several key challenges/research questions need to be resolved. First, we must determine which spatiotemporal attributes/features of an object are statistically-sufficient to distinguish between object classes. Second, we need to effectively and efficiently learn the spatiotemporal properties of these features in order to produce low false positive and negative rates for attack detection. Lastly, the defense needs to be evaluated against practically realizable attacks, under real-world scenarios, assuming a strong threat model with adaptive attacks.

We propose PercepGuard to detect perception-based misclassification attacks. In PercepGuard a recurrent neural network (RNN) is used as a sequence classifier to classify a track into an object class. During training the RNN learns the spatiotemporal property of bounding box sequences for each object class. Our detection criterion is that, if the classification result of PercepGuard does not match that of the ODT system, an alarm will be raised (Fig. 2). Our evaluation with BDD100K [22], a real-world driving dataset, shows a maximum false positive rate (FPR) of only 5%, and true positive rates (TPR) as high as 99% on adversarial patch attacks [23], a prevalent attack where the patches are generated by solving an optimization problem and can be realized via printed stickers [6], TV monitors [17], or projectors [9].

However, for adaptive attackers who also aim to evade PercepGuard, our TPR drops to 85.6% as it is possible to construct patches that accomplish not only misclassification but also detection evasion. For example, to alter a car into a person, an attacker would need to additionally change the bounding boxes from horizontal to vertical for the RNN to classify them as a person. To improve the adversarial robustness, we consider contextual information such as the perceiving (ego) vehicle's velocity, its relative velocity with objects, etc., which are generally available from on-board sensors, such as speed sensors and LiDARs.

Specifically, we augment the feature space of the RNN so that it learns the relation among these features during training. As a result, for the example above, the RNN still classifies those altered bounding boxes as a car even though they are vertical, because a person would not move as fast as the ego vehicle (i.e., with a low relative velocity) when the latter moves at a high velocity. Evaluation using a Carla-simulated dataset [24] shows that with only these two contexts, the TPR increases to 99% from 85.6%. A baseline comparison with NIC [25] and SCEME [26] shows that PercepGuard does carry additional value for considering the consistency in the temporal domain. In real-world experiments, we successfully demonstrate adversarial patch attacks (using an LCD monitor or a projector on a moving vehicle), but fortunately PercepGuard is able to detect 43 out of 45 attack instances, including adaptive attacks. Our contributions are:

- We propose PercepGuard to detect misclassification attacks against vision-based object detection and tracking (ODT) systems. PercepGuard is agnostic to attack methodology because it relies on ODT output (e.g., object classes and tracks) instead of the input (image pixels).
- An RNN is used to classify the spatiotemporal fingerprint, inherent in a detected object's track, into an object class. We compare the result with ODT's classification result for attack detection. To improve the adversarial robustness, we adopt contextual information as additional features.
- We evaluate PercepGuard using a real-world dataset and a simulation dataset. It produces FPRs as low as 5% and TPRs as high as 99% against adaptive attackers. Real-world experiments where adversarial patches are realized using a TV and a projector yield a TPR of 96% on average.
- PercepGuard outperforms two baselines, NIC [25] and SCEME [26], in terms of the TP and FP trade-off. We demonstrate its extensibility by incorporating SCEME's context features. Sensitivity analysis also shows our RNN is not vulnerable to small, direct perturbation to its input.

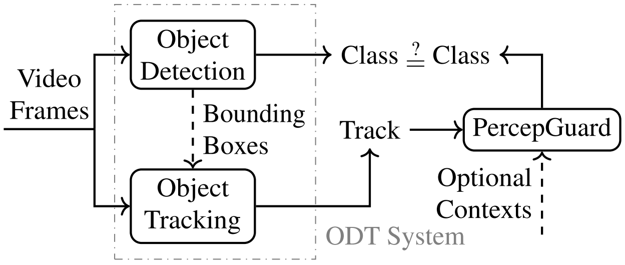


Figure 2: High level idea of PercepGuard

2 Background and Related Work

In this section, we introduce the background of object detection and tracking systems and their vulnerabilities.

2.1 Object Detection and Object Tracking

To perceive the environment, an autonomous system needs to locate the surrounding objects, classify them, and finally track their trajectories. Modern object detection is typically achieved via a two-stage NN (e.g., Faster-RCNN [27]) that is comprised of a region proposal network (RPN) and a detection network, or one-stage (e.g., YOLOv3 [28]) that achieves both at the same time. For object tracking, two-stage algorithms rely on an object detector for bounding box predictions of each frame, then they associate identical objects across frames based on their momentum [29], while a one-stage algorithm utilizes one single NN [30] to achieve both tasks. Both types of methods eventually output a *track* for each object's trajectory, i.e., a sequence of bounding boxes. PercepGuard aims to analyze the track (optionally with contexts) to verify the object's class prediction (Fig. 2) that may have been tampered by an adversary. As a proof-of-concept, we evaluate PercepGuard using vision-based, two-stage tracking systems as they are the most common and developed systems [1, 31, 32].

2.2 Perception Attacks

Although the state-of-the-art object detection and tracking algorithms have shown promising results in non-adversarial scenarios, they are however vulnerable to various perception attacks. There are object creation [10, 13, 33], removal [7, 8, 12, 13, 33], and tracker hijacking attacks [32, 34], but in this paper we focus on misclassification attacks that are equally severe, where the attacker alters the video frames so that the victim object is classified as an attacker-specified class.

The alteration may occur in either of the three domains: (1) Physical domain, where an attacker can place a sticker on a stop sign which will then be misclassified as a yield sign [35], or mount a TV on the back of the preceding vehicle so that it is recognized as a person [17], or simply project a person on the ground or wall [9], etc. (2) Perception domain, for example, against camera-based vision, an attacker may shine light into the camera to create adversarial ghosts [10, 36], use acoustic waves to disrupt the stabilizer of the camera [13], or exploit the rolling shutter effect [12], etc. (3) Digital domain, which is the strongest threat as the attacker has access to the sensing output (e.g., image) and can arbitrarily modify an image/video at the pixel level. Traditional digital domain adversarial examples [37] assume that the attacker induced noise is norm-bounded, to avoid being detected by human users. However, in the context of autonomous system applications, this constraint is no longer relevant since the images/videos are not examined by human users. We do not consider digital domain attacks in this paper, because if the attacker can gain direct access to the sensing output, they can also have the ability to directly modify, for instance, the outcome of perception module/decisions, and detecting such attacks falls into the goal of intrusion detection systems [38].

2.3 Existing Defenses

Traditional defenses against adversarial examples (e.g., [39]) are not applicable to perception attacks as they have a general assumption of norm-bounded perturbation which cannot apply to physical/perceptional attacks where typically the noise are not bounded by a small norm, otherwise they would be nearly impossible to realize. Defenses that do not have such an assumption limit themselves to a particular attack methodology, e.g., adversarial patch attacks [21, 40] or physically-realizable attacks [41]. Because PercepGuard verifies only the output of ODT systems (Fig. 2), it is agnostic to different attack methodologies and object detection algorithms.

Here, we focus more on consistency-based defenses for vision systems; they can be categorized into two types: sensor fusion-based and single sensor-based. Consistency checks can be done across multiple sensors, be they heterogeneous [19] or homogeneous [42]. For example, model-based methods such as Savior [19] verify physical invariants across multiple, heterogeneous sensors which requires sophisticated physical models, while we take a data-driven approach that is easier to deal with uncertainty and consider additional contexts. Zhang et al. [42] propose to fuse different views from multiple cameras to detect DoS attacks on cameras [43] but they are vulnerable to non-DoS camera attacks [10, 36].

On the other hand, one can use a single sensor's output, but with empirical/semantic knowledge that verifies the integrity of the sensory data. For example, AdvIT [20] detects adversarial noise from videos based on optical flow consistency, however it is limited to norm-bounded perturbation, thus vulnerable to universal adversarial patches [23] which are able to maintain the flow consistency across multiple frames. Co-existence consistency among multiple objects can also be utilized (e.g., traffic signs more likely co-exist with cars than a sink), thus incoherent objects can be detected [26, 44], although these methods do require multi-object scenes and may be subject to the richness of context and adaptive attacks [45].

For a single object, recently Gurel et al. proposed a knowledge-enhanced ML pipeline that verifies the consistency of a static object's (e.g., a traffic sign) attributes, such as color, text, and shape [46], while Wang et al. applied a similar idea to person re-identification [47]. However, their methodology is inapplicable to modeling the spatiotemporal consistency of moving objects, because the semantic attributes of static objects are discrete and finite, which can be fully specified by domain experts. However, spatiotemporal features are much more complex as they are continuous and high-dimensional, and involve more uncertainty. For example, having an octagon shape is a necessary condition for a STOP sign [46], but the movement of a vehicle cannot be described in such a deterministic manner.

Moving target defenses were proposed against traditional adversarial examples [37] where slightly different ML models are randomly selected from a model pool for deployment which makes the attack optimization problems more difficult to solve [48], however these methods do assume norm-bounded noise. Occluding relations among objects can be used to detect LiDAR attacks [11, 49], but such approaches are only applicable to LiDAR-based systems.

The idea of using temporal consistency and multimodal classifiers has been applied to other domains, such as biometrics-based user authentication/identification [50–54] and human activity/emotion recognition [55]. For example, to authenticate users, one can learn the time-series data from multimodal sensors available on a mobile phone, including the pressure sensor when they are entering their PIN [53], or the accelerometer and the gyroscope [51] whose data can also be classified using an RNN model [50]. Wearables such as smartwatches equipped with similar sensors can also help authenticate users [56], e.g., by their keystroke dynamics [52]. Driver stress detection can be performed by fusing eye movement, ego-vehicle dynamics, and the environmental context by utilizing the CNN-LSTM models [55]. Blood flow from retina scanning can be used for liveness detection [54]. However, it is infeasible to directly apply these methods, since our goal is to detect misclassification attacks in autonomous system's perception considering adaptive attacks, which involves unique technical challenges in the system design.

3 System and Threat Model

3.1 System Model

We consider an object detection and tracking system (Fig. 3) for autonomous vehicle applications, that are equipped with vision-based perception modules. The vision input is mainly provided by images/videos captured by an on-board camera as the main sensor. We assume that the on-vehicle camera is placed at a fixed location (e.g., dashboard) with a fixed orientation as is typical for existing systems [2, 3, 31]. The images are fed to a NN for object detection [28]. The object detection result of each frame will be integrated for object tracking. Other *optional* sensors may also be included to provide auxiliary input, such as lidar/radar, and vehicle speed sensors (VSS). Their corresponding sensory data processing algorithms can also be part of the system. The planning and actuating modules are out of the scope of this work, as are sensor fusion algorithms [18, 19].

3.2 Threat Model

We consider perception attacks where the adversary's objective is to alter an existing object's classification result into a target class, thus jeopardizing the behavior of the autonomous vehicle (e.g., forcing it to stop on a highway because it recognizes the front car as a person). We assume the attacker is able to cause misclassification for multiple frames consecutively in order to affect the decision-making of the autonomous system [32]. The attacks can be realized via physical modifications to the objects [9, 17] e.g., adversarial patches [23] which we will use for demonstration, or tampering sensors themselves [10, 13]. Therefore, the attacker must consider the *physical constraints* such as the limited patch size, location, and magnitude to make it realizable in the real world. Instead of bounding the noise norm, they minimize the patch magnitude to reduce attack cost. We do not consider digital domain attacks that alter the sensory data digitally[1], since it requires access to the internal network (e.g., the CAN bus) which can potentially take over the whole system [38].

We assume the adversary possesses white-box knowledge of the sensory data processing algorithms, e.g., (hyper) parameters of the NNs, as well as the raw video frames taken by the camera. We consider two types of adversaries:

- *Detection-unaware* attackers, who are not aware of our defense and only tries to cause misclassification. To do that, they can either inject adversarial noise opportunistically, or solve an optimization problem for the optimal perturbation (to the benign video frames), in which they aim at minimizing the magnitude of the perturbation subject to succeeding in targeted misclassification.

- *Detection-aware* attackers, a.k.a. adaptive attackers, who know the existence of our defense, possess the parameters of it, and try to evade the detection. To do so, they additionally add a constraint for detection evasion to the formulation.

When more than one sensor is used, we assume the adversary can obtain and modify the data from any combination of sensors. Finally, we assume the autonomous system (including PercepGuard) runs on trusted/tamper-proof hardware, firmware, and software (i.e., a trusted computing base [58]); that is, we do not consider internal threats that directly hack the CAN bus [38] or the operating system of a vehicle.

4 The Design of PercepGuard

In this section, we first state the problem of attack detection, and then give an overview of our approach with main

[1] Although PercepGuard can also detect norm-bounded adversarial examples that span multiple video frames in the digital domain [37, 57].

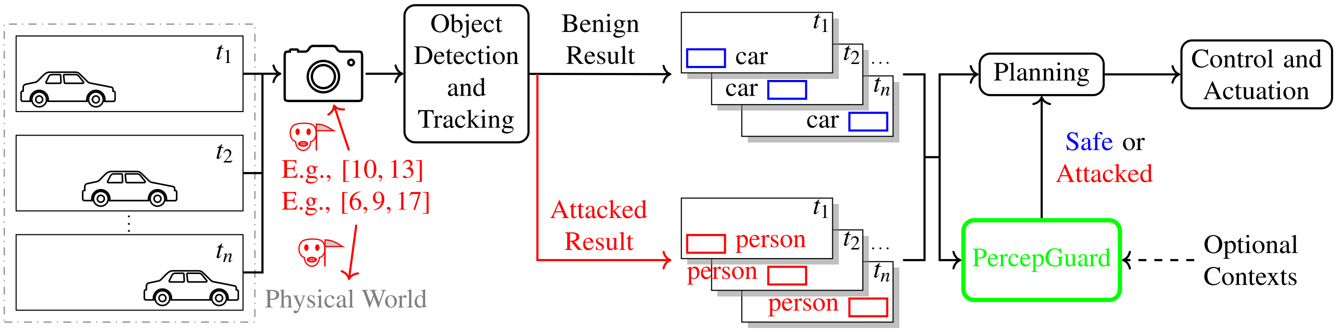


Figure 3: System and Threat Model. In the perception module of an autonomous system, vision sensors (e.g., cameras) perceive the environment and convert it into video frames for object detection and tracking, which labels an object with class and bounding boxes. The planning module takes these labels (among other inputs) for decision-making, and finally the control module executes these decisions. When the perception module is compromised, either by physical attacks (e.g., using an LCD monitor [17], or a sticker [6]), or camera sensor attacks (e.g., [10, 12, 13, 36]), the perception module outputs falsified labels which may mislead the planning and control modules. PercepGuard verifies these labels and alarms the system when an attack is detected.

challenges identified. Later, we introduce the basic method using bounding boxes only, and then the context-augmented detection which enhances the adversarial robustness.

4.1 Problem Statement and Challenges

Given a series of consecutive frames captured by a vision sensor (e.g., camera), for each detected object, the ODT system returns the class prediction of the object c, as well as a track, which is a sequence of bounding boxes $B = \{b_i : 1 \leq i \leq N\}$ where b_i is the bounding box at time i, and N is the number of frames in which this object is detected. A 2-D bounding box b is described by a vector $(x, y, h, w)^\top$, representing its center coordinates (x, y), height h and width w. Given (B, c), our goal is to verify whether $c = c_o$ where c_o is the ground-truth class. To achieve the detection goal, we propose to classify B into a category c' that shares the same domain of c (i.e., c' can take any value that c can). If $c \neq c'$, we regard the video frames as adversarial input (Sec. 4.2). Moreover, we consider strong attackers with white-box knowledge of our detection algorithm. They manipulate the frames for both c and B to make $c = c'$. We incorporate additional context information to enhance the adversarial robustness (Sec. 4.3).

There are several challenges involved. First, the existence of such a classifier is unknown. In other words, does B provide sufficient information to distinguish one object class from another? Second, if such a classifier exists, it is non-trivial to design it in a way that it achieves high-accuracy, and can be trained, tested, and extended efficiently. Third, it is equally important for an attack detector to ensure low FPRs as to ensure high TPRs. Finally, it is challenging to defend against adaptive attackers who are aware of the defense and try to evade it, and are able to compromise multiple, even heterogeneous sensors simultaneously.

4.2 Basic Attack Detection Approach

Intuitively, spatiotemporal characteristics of different object types are distinctive. By visually examining the empirical distributions of several objects from a real-world traffic dataset BDD100K [22] (details of them are plotted in Appendix A.1), we verify that their bounding boxes indeed have statistically distinctive behaviors. We propose to use long short-term memory (LSTM)-based [59] recurrent neural networks (RNNs) to learn the spatiotemporal property of bounding boxes, as they are well-known in sequence classification [60] (e.g., to see whether an English sentence is positive or negative). Note that, our RNN model is not intended to be used as a standalone object classifier but as a cross validation for an object detector's classification result.

An RNN-based sequence classifier takes a sequence of features as input and outputs the class of the sequence. It is composed of two steps: feature extraction and feature classification. For feature extraction, a typical recurrent operation is a feedback loop (Fig. 4), which can be described as

$$g_i = R_\theta(f_i, g_{i-1}), \tag{1}$$

where f_i is the i-th raw feature vector from the sequence, g_{i-1} is the hidden feature vector calculated from all the previous vectors. Parameter θ is identical for all recurrent operations.

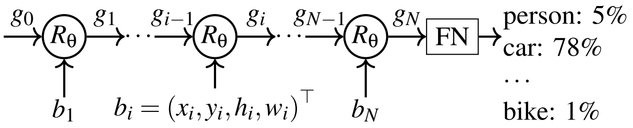


Figure 4: RNN-based Sequence Classifier. Bounding boxes are sequentially, and recurrently fed to the RNN layer, R_θ. A fully-connected layer (FN) is then used for classification, with a softmax layer (not shown) for normalization.

Algorithm 1: PercepGuard Online Attack Detection

```
Input: $b$'s: Bounding boxes (optionally with contexts)
       $g_0$: Feature initialization vector
       $\theta$: RNN parameters
       $c$: Object detector's classification result
       $\tau_F$: At least $\tau_F$ frames are required
Output: Alarm if attack detected
$t \leftarrow 0$
$g \leftarrow g_0$ ;                              // Feature initialization
while a new bounding box $b$ comes do
    $t \leftarrow t+1$ ;                          // Counting frames
    $g \leftarrow R_\theta(b,g)$ ;                // Feature extraction
    $c' \leftarrow \arg\max_i C(g)[i]$ ;          // Feature classification
    if $t \geq \tau_F$ and $c' \neq c$ then
        Raise an alarm
```

Once we have g_N after the entire sequence has been processed, we apply a fully connected layer to g_N for the class confidence vector, $y = C(g_N)$. Further details of the RNN are omitted as they are not required to understand the rest of the paper.

For our application, we abstract the entire sequence classification process as a function L,

$$y \leftarrow L(B), \quad \text{where} \quad B \in [0,1]^{N\times M}. \tag{2}$$

Matrix $B = \{b_i\}_{i=1}^N$ denotes the sequence of bounding boxes thus $M = 4$ for now and $f_i = b_i$. Here, N is the length of the sequence. The features are normalized into $[0,1]$ for better classification accuracy. Vector $y = \{y_i : 1 \leq i \leq r\}$ is softmax'ed therefore $\sum_i y_i = 1$, and the classification result is $c' = \arg\max_i y_i$, and r is the number of object classes the RNN model can handle. The class domains of our RNN model and the object detector are identical in principle.

Our actual detection algorithm runs in an online manner (Alg. 1). Instead of computing (2) after obtaining all the bounding boxes for multiple frames, for each new frame, once the object detector outputs a new bounding box for the object we have been tracking, we only need to execute (1) once to update the hidden feature g. In this way, we take advantage of the recurrent property of the model to reduce computational time with a minimal space cost of saving the previous g. Finally, we call $C(g)$ to check if its result matches the object detector's classification result. If not, an alarm will be raised. Our evaluation shows out that as few as five frames (which means 0.17-second assuming 30 fps) are sufficient for classification though we do not put an upper bound on it.

4.3 Context-based Enhancement

As we will show in Sec. 5.4, when we assume an adaptive attacker who also aims to evade PercepGuard, our attack detection rate drops to 86% (from over 99% against non-adaptive attacks). This is because the attacker is able to alter the bounding box features such that our RNN model classifies them into an attacker-intended class. Because the basic model is trained only based on *perceived* object movement which is *relative* to the measurement platform (ego-vehicle)'s movement, the model does not possess high differentiability between cases involving distinct objects that exhibit similar perceived temporal behavior. We show such an example in Fig. 11 in Appendix, where two vehicles are following each other at 30 mph (low relative velocity). In order to alter the preceding car into a person, an attacker can merely change its bounding boxes' shapes (from horizontal to vertical) without shifting their locations (keeping the same speed), such that our RNN model recognizes them as a person, because there are plausible cases where the ego-vehicle is moving at a low speed (or stopped) which leads to a low relative speed with a pedestrian.

Therefore, in order to defend against defense-aware attacks, we need more reference/context to validate camera's perception data. There are mainly two sources of reference. First, it can be from other perception sensors such as lidars and radars. Their data can be combined with camera's data via sensor fusion. But similar to a camera, these sensors provide measurement relative to the reference frame (i.e., the sensing platform, the ego-vehicle). Second, it can be the states of the ego-vehicle itself, such as its own speed from VSS', velocity and location from GPS receivers, pitch-roll-yaw positions from IMU, or even environmental factors such as road layout/speed limits and surrounding buildings/terrain from maps, or real-time traffic and weather conditions from live maps (e.g., Apple Maps [61]), etc. They provide side information which leads to more accurate measures of an object's absolute spatiotemporal behavior (as a result of calibrating the relative perception by considering the reference frame), unlike relative measurements output by perception sensors. Interestingly, we will show in our evaluation that relative contexts do not contribute as significantly as those absolute contexts in terms of enhancing adversarial robustness.

To incorporate additional contexts, we extend the feature space of the RNN model to implicitly learn the relations among all features. In particular, we concatenate all contextual features, denoted as a single vector o, to the original bounding box features, $u = b \,\|\, o$. We use U to denote the sequence of all these features over time, i.e., $U = \{u_i : 1 \leq i \leq N\}$. Then, the sequence classification is

$$y \leftarrow L(U), \quad \text{where} \quad U \in [0,1]^{N\times M}. \tag{3}$$

which is similar to (2), though M is now larger than four. For example, if we use the ego-velocity $v_e = (x_e, y_e, z_e)^\top$ as context, then $u = (x, y, h, w, x_e, y_e, z_e)^\top$ and $M = 7$.

Incorporating additional context information gathered from other sensors (that provides a more accurate view of the intrinsic spatiotemporal behavior of an object) significantly increases the attack difficulty (in searching for the feasible and optimal perturbation) and the attack cost (in realizing them). This is because to bypass PercepGuard, the attacker is forced to either manipulate the absolute spatiotemporal attributes of the target object (infeasible due to the constraint of overlapping with the perceived location of object), or modify not only the bounding boxes but also the contexts (by compromising additional sensors of different modalities).

5 Evaluation

We evaluate PercepGuard under both adversarial and non-adversarial scenarios with two datasets. We also compare with two baselines, and conduct real-world experiments.

5.1 Evaluation Methodology and Setup

We report the true negative (TNR), true positive (TPR), attack misclassification (AMR), and attack success (ASR) rates. The AMR is defined as the number of misclassified objects divided by the total number of objects, while the ASR is the number of misclassified objects that are not detected divided by the total number of objects, i.e., $\text{ASR} = \text{AMR} \times (1 - \text{TPR})$.

We implement the LSTM-based RNN model as our sequence classifier using Keras [62] with TensorFlow 2.4 [63] as the backend. We use the default architecture of LSTM provided by Keras with 50 memory units[2]. We use Adam optimizer [64] for both training the RNN models and solving the attack optimization problems. We adopt the variable-length training approach as it outperforms the fix-length setting. We combine YOLOv3 [28] with SORT [29] as the object detection and tracking system [32], in which we follow SORT's idea for associating objects across frames by matching ground-truth bounding boxes with YOLO-predicted bounding boxes based on their Intersection over Union (IoU).

Datasets: Two datasets are used for both scenarios: The BDD100K dataset [22], specifically its multi-object tracking (MOT) subset, contains 1,400 training videos and 200 test videos. All videos are 40-second long and recorded in five frames per second, containing a total of 130K objects with 3.3M of bounding boxes. From the original ten categories, we select bikes, buses, cars, pedestrians, and trucks, with a total of 5K instances due to the imbalance of the original dataset.

For contextual data, we create a dataset collected from the Carla simulator [24], referred as the Carla dataset [65]. In the simulation, a car is driving around in realistic residential areas and highways meanwhile recording video frames of the front scene. There are 1,000 videos, each of which is 200-second long recorded in five frames per second. For each object in each frame, in addition to its bounding box and class, we also recorded contextual data such as the relative velocity and the ego-vehicle velocity. Three categories of objects were labeled: vehicles, pedestrian and traffic signs.

5.2 Attack Methodology

In general there are two ways to realize misclassification attacks against the perception component of an autonomous system. The simplest way is to opportunistically place (e.g., via sticker, projection or display) real images of objects that obscure the victim object [9]. Alternatively, the attacker can adopt adversarial ML techniques to generate physically-realizable adversarial patches [23].

The attacker may use an LCD monitor [17] or a projector [9] to add adversarial patches onto the victim object. Either way, they first need to digitally synthesize those patches that can achieve their attack goals, while making sure that the patches can be implemented in the physical world, therefore some physical constraints need to be considered. For example, if they choose to mount an LCD monitor on the back of a car, the physical constraints are the location of the patches (e.g., they cannot be far away from the car), the size (limited by the screen size and attack distance), and the magnitude (limited by the monitor's maximum brightness), etc. We show that despite these challenges, they are still relatively easy to launch, thus bringing severe threats to autonomous systems.

Depending on the attacker's goal, knowledge and capability, there are two types of attackers: defense-unaware and defense-aware attackers (Sec. 3.2). Namely, the latter is aware of our detection and tries to evade it while achieving their misclassification goal. From a high level, the adversary aims to find a perturbation sequence Δ to the videos that

$$\underset{\Delta}{\text{minimize}} \quad \|\Delta\| \quad \text{such that} \quad \bar{c} = c'', \quad \bar{c} = c', \tag{4}$$

where $\Delta = \{\delta_i\}_{i=1}^N$ with N being the number of video frames. $X = \{x_i\}_{i=1}^N$ is the benign video, thus the perturbed video is $X'' = \{x''\}_{i=1}^N = \{x_i + \delta_i\}_{i=1}^N$. When an object detector takes X'' as input, it outputs B'' and c'' which are the sequence of bounding boxes and the object category, respectively.

The attacker minimizes the ℓ_2 norm of Δ to reduce attack cost. The first constraint means that the classification result of the object c'' (by the object detector) needs to be the same as the targeted class $\bar{c}$, i.e., a successful misclassification attack. The second constraint means that the attacker also aims to fool our attack detector to classify the object as $\bar{c}$ as well. Our actual implementation of (4) considers the detail of YOLOv3 and transfers the constraint into the objective function [37]:

$$\min_{\Delta} \|\Delta\| + \mu_{\text{MC}} \cdot \mathcal{L}_{\text{MC}}(X'', \bar{c}) + \mu_{\text{ST}} \cdot \mathcal{L}_{\text{ST}}(X'', \bar{c}), \tag{5}$$

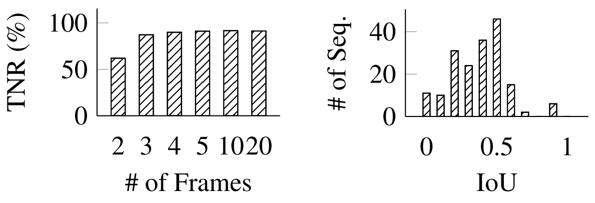
where the misclassification (MC) loss measuring how unsuccessful the perturbation Δ is thus far [37], i.e.,

$$\mathcal{L}_{\text{MC}}(X'', \bar{c}) = \max_{x'' \in X''} \left[\max_{\hat{c} \neq \bar{c}} P_j(x'', \hat{c}) - P_j(x'', \bar{c}) \right]. \tag{6}$$

$P_j(x, c)$ is the confidence of the class c for the frame x reported by the j-th cell which is the detection cell responsible for the victim object (similar to [6]). It already considers $P_j(x)$, the confidence that the j-th cell contains an object, but we omit it for brevity. The stealthiness loss $\mathcal{L}_{\text{ST}}$ is defined as

$$\mathcal{L}_{\text{ST}}(X'', \bar{c}) = \max_{\hat{c} \neq \bar{c}} L_{\hat{c}}(B'') - L_{\bar{c}}(B''). \tag{7}$$

[2] Source code: `https://github.com/harry1993/percepguard`



(a)

(b)

Figure 5: (a) Our RNN needs only 5 frames for classification. (b) IoU for direct perturbation to bounding box sequences.

Compared with (2) where $L(B)$ returns a confidence vector of all classes, here $L_c(B)$ returns the confidence of Class c for a given bounding box sequence B. Of course, B can be replaced with a feature sequence U when context is considered (Eq. 3). Two losses are balanced by two constants, μ_{MC} and μ_{ST}. For defense-unaware attacks, $\mu_{\text{ST}} = 0$ and $\mu_{\text{MC}} = 100$ because lower values yield low AMRs while higher values do not make a significant improvement. For defense-aware attacks, we set $\mu_{\text{ST}} = 100$ due to the same reason. A similar formulation of adaptive attack has been adopted by [45].

5.3 Non-adversarial Scenarios

In non-adversarial scenarios (no attack), we evaluate PercepGuard's performance in terms of the TNR as too many false alarms may cause the autonomous system to malfunction.

BDD100K Dataset Results: To test the model, we use YOLO's bounding box predictions instead of the manual labels that we used to train the model as there will not be manual labels in actual deployments. For reference, YOLOv3's mean Average Precision is 55.3. Results show that the test accuracy is 95%, which means the FPR rate is only 5%. We also evaluate how many frames PercepGuard requires to make accurate classification. Fig. 5a shows that the TNR saturates at around four or five frames, which means it needs as few as five frames (with 30 FPS, it is 0.17-second) to make a decision.

Carla Dataset Results: The 5-fold cross-validation accuracy is 95%. When testing the model, we run YOLO on Carla's simulated images for the bounding box predictions. The test accuracy of our RNN model on these predictions is 92%, indicating a FPR of 8% which is higher than the BDD100K; this is because the official, pre-trained YOLO model was trained on the COCO dataset [66], a real-world dataset whose pixel distribution is different than the Carla-simulated pixels.

Since our Carla dataset also provides rich contextual data, we select two contexts, relative velocity and ego velocity, along with bounding boxes to train and test our RNN model to see whether they can help improve the TNRs. We added random noise drawn from the standard Gaussian distribution, $\mathcal{N}(0,1)$, to those contextual data in order to simulate natural sensor errors (compared with 0-60 MPH, the range of those velocities). Results show that the improvement is minor, with a test accuracy of also roughly 92% regardless of which velocity/velocities we append to the bounding box features. This is because those additional features do not provide extra information for sequence classification; however, we will show that they become useful in adversarial scenarios where the consistency among features is more critical.

Table 1: Adversarial patch attacks with BDD100K

Attack Type	Patch Size	AMR	TPR	ASR
Defense-unaware	20×20	83.47%	99.63%	0.3%
	40×40	89.41%	100%	0%
	60×60	92.94%	100%	0%
Defense-aware	20×20	73.25%	98.74%	0.92%
	40×40	80.49%	90.33%	7.78%
	60×60	87.6%	85.67%	12.55%

5.4 Adversarial Scenarios

Defense-unaware attacks and defense-aware attacks (a.k.a. adaptive attacks) are evaluated here. For demonstration purposes, we assume the attacker aims to alter a "car" into a "person" as a representative, severe case which could result in a hard brake executed by the victim vehicle on a highway.

We use the Adam optimizer [64] to solve attack optimization problems with $\mu_{\text{MC}} = 100$ and $\mu_{\text{ST}} = 100$, which indicates that the optimizer is to prioritize misclassification and stealthiness over minimizing perturbation magnitude[3]. The optimization process terminates upon convergence or when it reaches the maximum number of iterations (1,000 steps). The learning rate is 0.01. After its termination, we check both losses: If $\mathcal{L}_{\text{MC}} \leq \tau_{\text{MC}}$, we mark the object as misclassified; if $\mathcal{L}_{\text{ST}} \leq \tau_{\text{ST}}$, we mark this attack as stealthy as it has evaded detection. We let $\tau_{\text{MC}} = 0$ and $\tau_{\text{ST}} = 0$ for the digital domain (simulated) evaluation as it is easier for the attacker to achieve. However, we raise them for real-world evaluation (Sec. 5.8) for higher attack confidence due to the random environmental factors that make the attack not as stable as in simulation.

BDD100K Results: We first present the results on the BDD100K dataset (Table 1). We vary the size of adversarial patches from 20×20 to 60×60 pixels to emulate various attack distances and patch display sizes, as compared to the input image size for YOLOv3 which is 416×416 pixels. As the patch size increases, the attack performance generally gets better because there are more pixels upon which the attacker can manipulate. For defense-unaware attacks (Table 1) specifically, the AMR starts from 83% when the patch size is only 20×20, and approaches 93% at 60×60, which means the attack is considerably effective, indicating a severe threat to autonomous vehicles. However, when we feed to PercepGuard the bounding boxes of those compromised objects that are misclassified as "person" by YOLOv3, nearly all of them

[3] Ideally, both constants should be adjusted via binary search in order for them to be as small as possible [37]; however, that is only for norm-bounded perturbation, which we do not consider as it is too weak to test against PercepGuard.

are (correctly) classified into their benign class “car”, i.e., the TPR is almost 100%. As a result, almost all the attacks are unsuccessful, i.e., the ASR is nearly zero.

On the other hand, defense-aware attacks (Table 1) yield lower AMRs, because in this case the attacker aims for two objectives: misclassification and stealthiness for which the noise to achieve each objective does not always intersect and can be mutually exclusive. The TPRs of PercepGuard are slightly lower because such an attacker aims to evade the detection by optimizing the adversarial noise to modify the object class and bounding box behavior (e.g., shape and locations) at the same time. Without any additional context, although the RNN classifier can distinguish different object classes when the attacker does not intentionally modify the bounding box behavior, there can be boundary cases where two objects of different classes appear to have similar behavior (which is also why there are false positives). For example, a vehicle that is turning at an intersection may see a person on a sidewalk (with the bounding box shape being modified by an attacker) as similar to a car crossing the street, which can be confused with the case of seeing a real vehicle crossing the intersection when the ego vehicle is waiting for a red light. The defense-aware attack may exploit such corner cases in the confusion matrix, which may only require changes in bounding box shapes but not their locations. Note that the latter is more difficult to achieve as the attacker only apply patches within the region of the victim object. Even with a slightly lower TPR, we note that for a 416×416 image, a 60×60 patch is large enough to entirely occlude a car from the back for most instances in BDD100K.

In summary, the defense-unaware attacks are highly effective, but PercepGuard is able to detect most of them with a detection rate as high as 100%. Adaptive attackers are able to bypass our basic detection approach for at most 12.55% of their attacks but note that such attacks are fairly impractical to realize. Nevertheless, next we show that contextual data can help us strengthen the robustness of PercepGuard.

Carla with Context: We use our Carla dataset with contextual information to evaluate the same adversaries to show how context can help enhance PercepGuard. We only consider defense-aware attacks here as they are stronger.

For demonstration, we select two contexts, ego velocity and relative velocity, which can be obtained from vehicle speed sensors (VSS) and LiDAR/radar, respectively. We train the RNN model following the form of (3), where $M = 10$ because each velocity is represented in 3-D. Here, we assume a strong attack model where the attacker can compromise different sensors simultaneously, and more importantly they can synchronize each sensor attack precisely.

Table 2 summarizes the attack and defense performance with different attack capabilities, namely which sensors are compromised. When comparing the first row of Table 2 with the last row of Table 1 where only the camera is compromised, the AMR is similar but TPR is increased from 86% to 99% which makes the ASR approach zero; this is because the RNN model has learned the consistency among those ten features during training thus being able to detect the inconsistency of the tampered bounding boxes, therefore still classifies the feature sequence as a car for 99.4% of the cases.

Table 2: Differing attack capabilities at 60×60 patch size against RNN trained with contexts

Compromised Sensors	AMR	TPR	ASR
Camera Only	88.76%	99.35%	0.6%
Camera + LiDAR	89.90%	97.88%	1.9%
Camera + VSS	90.47%	85.26%	13.33%
Camera + VSS + LiDAR	90.85%	72.74%	24.76%

Comparing the second and the third rows of Table 2, where the attacker can additionally compromise LiDAR or VSS sensors, we observe that our detection rate decreases more when the VSS is compromised: 85% vs 98% (for compromising LiDAR), although both velocities have an identical number of features (three scalars). This suggests that the ego velocity plays a more important role in PercepGuard’s consistency verification, because relative velocities can be implicitly inferred from bounding box size changes (e.g., when the front car gets closer, its bounding box gets larger). In fact, such size changes have been used to estimate the following distance [67]. In other words, existing LiDAR attacks [7, 49] would not have a large impact on PercepGuard’s performance because LiDAR’s outputs are not as critical as VSS’, and yet the VSS attack is more difficult to conduct as it requires close proximity (a few centimeters) to the victim vehicle [68].

Finally, if the attacker can compromise three sensors at the same time, PercepGuard can still detect 73% of the attacks. The reason why it yields a lower detection rate is that when the capacity of a machine learning model (e.g., a neural network in our case) increases, it becomes more susceptible to adversarial attacks [57]. At a glance, this is worse than not using contextual data (Table 1). However, we argue that such an attack is extremely difficult to launch physically as it requires the attacker to gain the white-box knowledge of three different systems and meanwhile synchronize three separate attacks precisely. Another way to achieve such a strong attack would be to remotely hack into the CAN bus of a vehicle, where an attacker digitally alters the sensor data before or after the processing algorithms [69]. However, such an attacker is not considered in our threat model.

Model Transferability: Since manual labeling is expensive but simulated datasets can be scaled easier, we evaluate the transferability of our RNN model here. In particular, we train the RNN model using the Carla dataset and test it on BDD100K. Results show that the TNR is 83%, and the TPRs against defense-unaware attacks and defense-aware attacks are 92%, 87% respectively for 20×20 patches in simulation. Although these rates are not significant compared with Table 1, they still suggest that one could train the RNN model

Table 3: Attacks with larger perturbation areas.

Dataset	AMR	TPR	ASR
BDD100K	88.08%	42.35%	62.7%
Carla	90.67%	96.91%	1.9%

on a large-scale, simulated dataset and then fine-tune it with a small-scale, real-world dataset.

5.5 Sensitivity Analysis

We evaluate PercepGuard's sensitivity to direct, subtle perturbation to its input, i.e., the bounding box sequence, and to attacks that can perturb much larger areas than the patches.

Bounding Box Perturbation: We directly perturb the input to the RNN, i.e., bounding box sequences, to see how much perturbation is needed to change the classification result. The optimization formulation is

$$B^* = \arg\min_{B'} \ \mathcal{L}_{\text{ST}}(B', \bar{c}) + c_{\text{IoU}} \cdot \mathcal{L}_{\text{IoU}}(B, B'), \quad (8)$$

where B is the benign bounding box sequence, $\bar{c}$ is the target class, and $\mathcal{L}_{\text{IoU}}$ is the IoU loss that measures how close B is to B'. The stealthiness loss $\mathcal{L}_{\text{ST}}$ indicates how likely the RNN classifies B' as $\bar{c}$, defined as

$$\mathcal{L}_{\text{ST}}(B, \bar{c}) = \max_{\hat{c} \neq \bar{c}} L_{\hat{c}}(B) - L_{\bar{c}}(B).$$

Recall that $L_c(B)$ is the confidence that the RNN classifies B as c. To find the minimal perturbation to B, we follow the strategy from [37]: We start $c_{\text{IoU}} = 1$ and see if $\mathcal{L}_{\text{ST}}(B', \bar{c}) < 0$ when the optimization process converges. If not, we increase c_{IoU} and try again, until $\mathcal{L}_{\text{ST}}(B', \bar{c})$ falls below zero.

We compute (8) for 181 sequences of bounding box of length ten, and the IoU histogram (Fig. 5b) of the maximum IoU between perturbed bounding boxes and the benign ones in terms of bounding box sequence, shows that over 87% of the bounding box sequences need to be resized and/or shifted by an IoU less or equal to 50%, which means in order to change the classification result of the RNN model, the attacker needs to alter the bounding boxes significantly.

Larger Perturbation Area: Previously, we evaluated adversarial patch attacks with sizes up to 60 × 60 which is limited by the size of the monitor. Here, we relax such a limitation by simulating an attacker that uses a projector to project adversarial noise onto the front scene of the victim vehicle, similar to [9]. We assume that the projector is powerful enough to perturb any area other than the sky. In this case, the perturbation area occupies over 50% of the pixels (See Fig. 9 in Appendix for an example). We use image segmentation to identify the feasible perturbation areas for BDD100K and use LiDAR points for our Carla dataset based on distance.

Results from two datasets are shown in Table 3. When comparing with the last row of Table 1, the AMR for BDD100K is slightly higher even the perturbation area is now much larger. This is because perturbing the pixels that are not on the vehicle has little impact on altering its classification result. However, the TPR drops since now the attacker has more room to shift the bounding boxes in order to bypass PercepGuard. This result also verifies that our attack is strong. When considering the ego vehicle velocity and relative velocity provided by the Carla dataset, our TPR increases to 96.91% because the additional contexts that are not compromised can provide more evidence to detect the spatio-temporal inconsistency of the shifted bounding boxes.

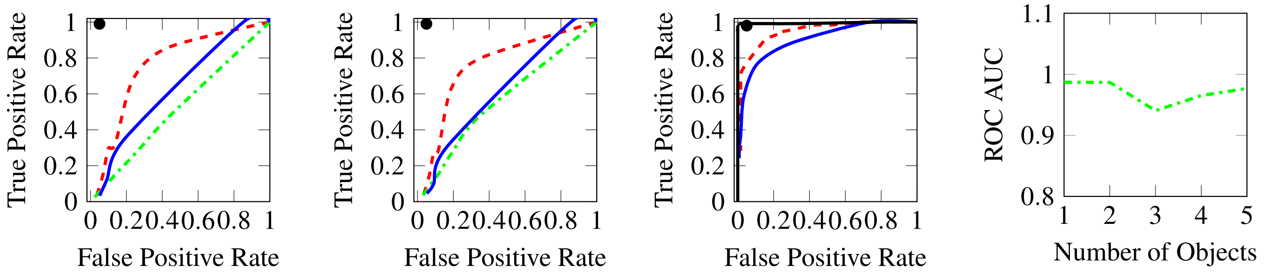
5.6 Baseline Comparison

We experimentally compare PercepGuard with NIC [25] and SCEME [26] as baselines, since they are both also input-agnostic, and SCEME also uses contextual consistency.

NIC detects adversarial examples against image classifiers by checking the network invariant. It produces a binary decision for each individual video frame. SCEME also works on individual frames, but it produces a scalar for each object by verifying its contextual consistency with other objects, backgrounds, and scenes within the frame. Note that we define contexts as the data from other sensors which is different from their definition. See Appendix A.3 for how they work in detail. A common drawback of both is their limited applicability: NIC is designed for image classification only, which is not as commonly used in autonomous driving as object detection. Though its principle may be applied to object detection models but it needs to be significantly redesigned. SCEME relies on RPNs thus being fundamentally incompatible with one-stage algorithms such as YOLO which is more commonly used in autonomous driving [1, 32].

Most importantly, both methods do not consider temporal features across multiple frames. Let us elaborate conceptually on why temporal features are useful in object misclassification detection. Consider this scenario where an object labeled "person" is located at the image center, which means it's in front of the ego vehicle. SCEME is unable to distinguish whether the object is a real person walking across the street with the ego-vehicle waiting for them, or a preceding vehicle that is misclassified as a person, since it only focuses on individual frames. On the other hand, PercepGuard can identify the difference since it additionally considers temporal features and contexts. If the object is moving from side to side slowly while the ego speed is nearly zero, it's probably a person; If it stays at the image center while the ego vehicle is driving (e.g., at 40 MPH), it is very likely a misclassified vehicle.

Methodology: Since both of them output detection scores for a single frame, we need to first integrate inter-frame scores into a single score for a sequenc of frames. To address this issue, given a sequence of detection decisions, $\{s_i; 1 \leq i \leq N\}$ with s_i being one (adversarial) or zero (benign), we define two integration criteria: (1) By *portion*: The percentage of frames marked adversarial, i.e., $\sum_i s_i/n$, and (2) By the length of the



(a) NIC with [37]

(b) NIC with DenseNet

(c) SCEME

(d) Varying Object Numbers

Figure 6: Baseline comparison results. Real-image attacks are tested with the portion criterion (-·-·-). Adversarial patch attacks are tested with both the portion criterion (- - -) and the consecutive criterion (——). PercepGuard's performance is marked with ● (binary) and —— (soft). (a)(b) NIC's ROC performance with two different NN architectures. (c) SCEME's ROC. (d) SCEME's performance (-·-·-) depends on the number of objects in the scene, which is consistent with the original paper [26].

longest *consecutive* frames marked adversarial. For both criteria, if the integrated score s exceeds some threshold, we mark the whole sequence as adversarial. We present receiver operating characteristic (ROC) curves with varying thresholds for NIC and SCEME, to compare with PercepGuard's performance which is a single point in the ROC plots (there is no threshold for PercepGuard because we use binary comparisons as the operating point by default except in Fig. 6c where we also test the soft comparison which results in a curve). We use the official implementations of NIC and SCEME.

Considering the differences in design and applicability between NIC and PercepGuard, to compare NIC with PercepGuard we crop out the portion of the object of interest from the original frame in the BDD100K videos and use the crop-out images to form a benign dataset that has ten object classes. We use the dataset to train two image classifiers in two architectures, DenseNet [70] and the one from [37] (same as [25]). Both classifiers achieve test accuracy higher than 98%. Then, we train NIC on each of them. We use defense-unaware attacks (Section 5.1) to perturb 9K objects using adversarial patches and 7K objects using real images. At test time, for each perturbed object, we obtain a sequence of crop-out images $\{x'_i : 1 \leq i \leq N\}$ from its video frames and feed each x'_i to the classifier and its corresponding NIC for which we obtain a sequence of binary decision $S' = \{s'_i : 1 \leq i \leq N\}$ for TPR. Similarly, for each benign test object we obtain a benign score sequence $S = \{s_i : 1 \leq i \leq N\}$ to compute FPR.

Because SCEME is incompatible with YOLOv3, in order to compare it with PercepGuard, we conduct new experiments to evaluate PercepGuard under Faster-RCNN [27]. We use the MOT dataset for the evaluation. From each video, for each "car" or "person" object across the N frames we obtain a sequence of its context profiles $U = \{u_i : 1 \leq i \leq N\}$, as well as a sequence of its bounding boxes $B = \{b_i : 1 \leq i \leq N\}$. Then, we use the IFGSM algorithm [26,71] to perturb the "car" objects in each frame so that classifies them as a "person". Similarly, we collect the context profile sequence U' and the bounding box sequence B' of each successfully misclassified object. To train the autoencoders, we gather 10K context profiles u per class of benign objects and divide them into a training dataset (80%) and a test dataset (20%). We use the same hyper-parameters as in [26] for the training. We put together the benign test profiles and the adversarial profiles u' to get the threshold of the reconstruction loss, τ_{RL}, at the equal error rate. With this threshold, we can convert a sequence of context profiles into a sequence of binary detection.

For PercepGuard, we feed B to our RNN for the FPR and B' for the TPR, which produces a point. We also test soft operating points. When the RNN's confidence on the class of interest is lower than a threshold, we mark the sequence as adversarial. We vary the threshold to obtain an ROC curve.

Results: Following our two inter-frame integration criteria, we calculate the ROC curves (Fig. 6a) based on benign score sequences and adversarial score sequences. The threshold for the portion criterion varies from 0% to 100% and the threshold for the consecutive criterion is from two to six. Overall, PercepGuard performs better than NIC in all the cases since NIC is designed for image classification while the video frames used at the test time are perturbed against object detection. Furthermore, NIC is unable to detect real-image attacks because the noise is not generated via optimization-based methods, which does not cause the network invariants to be out-of-distribution for NIC to detect.

From the ROC curves in Fig. 6c, we can see that SCEME works significantly better than NIC (Fig. 6a). This is because SCEME is designed for object detection, but NIC is for image classification. Also, the proportion criterion outperforms the consecutive one similar to NIC's results. PercepGuard achieves 98% TPR and 5% FPR, which is close to the result with YOLOv3. This also confirms PercepGuard is agnostic to the underlying object detection and attack methods (i.e., IFGSM [71] and C&W [37]). Soft combination (20 thresholds within $[0,1]$) yields a better TP-FP trade-off than SCEME.

Because SCEME relies on the context of the scene, we anticipate that the richness of the context (e.g., object-object spatial/coexistence relationship) affects its performance; meanwhile PercepGuard's performance is not subject to such variation since it mainly relies on the temporal features which exists for every single object. To verify this, we divide the profile sequences into groups, and in each group, the average number of objects across the frames in every sequence is identical. For each group, we calculate the area under the ROC curve (AUC), and we plot the AUCs in Fig. 6d. We can see that SCEME's performance first drops and then arises as the number of objects increases, which is consistent with the result in Fig. 6 from the SCEME paper due to the balance between object-object context and the rest of contexts [26]. However, PercepGuard performs more stably with varying numbers of objects, with true positive rates above 98% and false positive rates around 5%.

5.7 Model Extensibility

Our framework is general such that it can be easily extended to work with more input modalities and/or additional contextual features as long as they contain spatiotemporal attributes, e.g., 3-D bounding boxes, point clouds from LiDAR, shapes from image segmentation, etc.

We demonstrate PercepGuard's extensibility by integrating SCEME. We extend the context profiles output by SCEME to the temporal domain, and verify the spatial-temporal consistency of each object. Recall that for each object in each frame, SCEME outputs a context profile $u = [r, \gamma_{u1}, \gamma_{u2}, \gamma_{r1}, \gamma_{r2}]$. We run SCEME on the MOT subset of the BDD100K dataset, and for each object we obtain a sequence of its context profile $U = \{r_i : 1 \leq i \leq N\}$ where $N = 5$ and i refers to the index of frame. We select four object classes: bike, bus, car, and pedestrian (missing truck because the context Faster-RCNN used by SCEME is trained with the VOC dataset [26] that do not include trucks). We collect a total of 10K of such sequences and use 80% of them to train the LSTM with 1000 memory units (the size of a context profile u is 5×4096), and 20% for testing. The 5-fold cross validation shows the test accuracy is 97% (false positive rate is 3%).

To test the attack detection performance, we feed U', the sequences of context profiles of attacked objects, to the RNN classifier and the true positive rate is 98% based on 200 sequences. This means that the framework of PercepGuard is compatible with features that are extracted automatically by neural networks instead of being selected manually by human. We underline that we only use the r part of the entire context profile u; we test the cases where we either use only one of the remaining feature vectors (γ, the GRU gates), or concatenate all of them together with r: the test accuracies are all below 40%. This is because r is the final fused result which contains much richer contextual information than the gates.

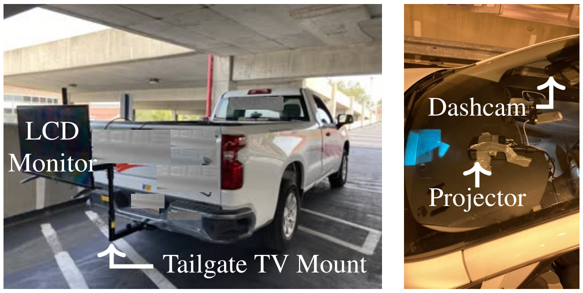


Figure 7: Real-world experiment setup. (Left) An LCD monitor is mounted on the back of the front vehicle. (Right) A dash-cam is mounted on the windshield of the ego vehicle. A portable projector is taped on the dashboard.

5.8 Real-world Experiments

We conducted a series of real-world experiments to demonstrate the threat of adversarial patch attacks, and more importantly, to evaluate PercepGuard's performance in real-world environments and under realistic attacks.

Methodology and Setup: To implement the adversarial patches, we extend the experiments done by Hoory et al. [17], where we mount a 40-inch LG LCD monitor, on the back of a vehicle (Fig. 7) using a tailgate TV hitch mount. Hence, we are able to drive around with the LCD while Hoory et al. only performed stationary tests, thus being more practical. We additionally use a portable projector, Optoma LV130, mounted on the dashboard of the ego vehicle to replace the LCD monitor as a different noise injection vector. We collected 45 videos in total. The experiments were conducted in a large campus parking garage during both daytime and night, and on a campus street only at night because during daytime both the monitor and the projector would be completely overwhelmed by the sunlight. We reduced the monitor brightness at night because otherwise the content would be saturated.

We regard safety as the highest priority. The location and time were carefully chosen for minimal traffic. We confirmed the setup was secured before driving with it. Our team is experienced with outdoor vehicle experiments. See Appendix A.2 for more oversight that we took to minimize the risk.

The victim vehicle is driving behind with a dash cam (VIOFO A129 Pro) installed on its windshield acting as the vision camera of an autonomous vehicle [3], that records videos of the front scene. The dash-cam is equipped with a GPS receiver so it stamps vehicle speeds on the frames. We manually took those speeds out and transfered them into 3-D velocities with two other dimentions being zeros. We trained the RNN classifier with our Carla dataset to utilize the speed context. If the attacker makes the object misclassified for three consecutive frames, we count it as an effective attack.

Real Images: We first implement the opportunistic attacks (without any optimization), by using both devices to display some real images of people and stop signs on the back of the

Table 4: Real image attacks in the real-world

Real Images of	Device	ARR	TPR	ASR
People	Monitor	63.2%	83.3%	10.6%
	Projector	58.8%	100%	0%
Stop Signs	Monitor	40.0%	100%	0%
	Projector	20.0%	100%	0%

Table 5: Adversarial patch attacks in the real-world

Attack Type	Device	AMR	TPR	ASR
Defense-unaware	Monitor	52.2%	100%	0%
	Projector	27.3%	100%	0%
Defense-aware	Monitor	45.5%	100%	0%
	Projector	20.0%	100%	0%

front vehicle (Fig. 12 in Appendix) to see if YOLO can recognize them, and more importantly, to see if PercepGuard can detect them with the speed context. In general, the monitor yields higher attack recognition rates (ARR, for which YOLO classifies them as intended by the attacker) than the projector does (Table 4); this is because when the following distance changes, it was challenging for the driver, who in our experiments also operated the vehicle, to simultaneously adjust the projector's focal length in real time. Another reason is that the projection screen was unstable while the monitor was fixed on the truck. More interestingly, the ARR of stop signs are lower than that of a person. Because we only displayed the top of a stop sign without the pole, one explanation is that YOLOv3 also takes objects' surroundings into consideration [33, 72] which in this case is the pole. PercepGuard is able to detect most of the attack instances (except two pedestrian cases). We manually checked those failing cases; the reason is that both vehicles were driving at only 2 MPH, which PercepGuard accepts because it is also a reasonable speed for pedestrians. In fact, because such failures occur mainly at low-speed scenarios, they could be addressed in time by the human driver.

Adversarial Patches: Next, we implement the (optimized) adversarial patch attacks in the physical/perception domain and conduct experiments with the following procedure. We first drove around the area with one car following another at various distances from one to ten meters during daytime and at night, meanwhile capturing benign frames without the LCD. Then, we fed those benign frames into the optimization processes (Sec. 5.2) to solve for 20×20 (because it is a 40-inch monitor) adversarial patches that alter a "truck" into a "person". We specify the pixel sizes and coordinates of the patches so that they can be realized by both attack devices. Because the color accuracy of the dashcam and both devices is imperfect, to compensate for channel effects, we manually calibrated the color by making the red and green channels twice as stronger and the blue channel weaker by half. Next, we installed the monitor/projector and displayed those optimal adversarial patches (with calibrated color), at the same time we drove around the same area while recording adversarial frames. Finally, we analyze these adversarial frames for AMRs, and PercepGuard's TPRs.

Results in Table 5 show that defense-unaware attacks yield a higher AMR but PercepGuard is able to detect all the attack instances. This is partly because the physical noise makes the bounding box predictions more unstable, or bouncier, than the instances from simulations (Secs. 5.3 and 5.4). Especially, in the defense-aware attacks, the attacker tries to control the bounding boxes to fool PercepGuard, but to the contrary, they became even more unstable in the real world.

Attack Cost and Difficulty: For the display-based attack, we conducted the LCD monitor experiments with a 20×20 patch size because it is a 40-inch monitor at a distance of five meters. In order to achieve 60×60 as we simulated where the attacker has non-zero success rate (Table 1), they need at least a 120-inch display for outdoor. 50, 55, 75-inch outdoor displays are roughly \$3.2K, \$3.8K and \$10K [73]. Based on such an exponential trend, a 120-inch monitor could be \$40K.

Similarly, projectors with higher brightness are more expensive. Empirically, the luminance required for a projector is proportional to the noise magnitude ℓ_∞, ambient illuminance, distance square, i.e., Lumen $\propto \ell_\infty(\Delta) \cdot \text{Illum} \cdot d^2$. From our real-world experiments at night with an illuminance of 30 lx, our \$180, 300-lumen projector was able to induce $\ell_\infty = 0.1$ at a distance of two meters. In order to bypass PercepGuard where typically an ℓ_∞ norm of one is required, the attacker needs a 9K-lumen one priced at \$13K [74] at night at a distance of three meters. During daytime when the illuminance is 40 klx, one may need a 75K-lumen projector for which a used one is priced at \$375K [75].

Note that, these attacks can easily be thwarted using additional contexts (Table 2). In terms of evading our context-enhanced defense, to attack multiple, heterogeneous sensors simultaneously, the attacker needs not only the knowledge of those different sensor systems, but also the equipment to spoof those sensors; let alone synchronizing each individual attack precisely. In summary, PercepGuard significantly increases the cost and difficulty for an attacker to evade the detection.

6 Discussion and Limitations

6.1 Other Attack Types

Although PercepGuard focuses on detecting misclassification attacks, it can potentially detect object creation attacks [9, 10, 13] and tracker hijacking attacks [32, 34], because these attack can also cause spatiotemporal inconsistency. For example, in displaying images of objects on an LCD monitor mounted on the back of the truck, we were creating objects so far as YOLOv3 could determine (even though the monitor was neither large enough nor bright enough to obscure the truck). As the objects were not consistent with their inherent context—i.e., the spatiotemporal attributes of the objects did not match their classes (viz., a stop sign should not be moving)—PercepGuard detected most of the attack instances. For attacks against (multi-) object tracking, where detected bounding boxes' locations are shifted in order to deceive object trajectory predictions [32], existing work used simulated patches to demonstrate feasibility. Because of the static nature of the patch (it must move according to the object it is affixed to), it would be difficult for an attacker to maintain the spatiotemporal consistency of the spoofed object.

Existing single sensing modality, consistency-based defenses, including PercepGuard, are not applicable to object removal attacks, since the features needed for verification do not exist for undetected objects. Leveraging the fusion of inputs from multiple, different sensors can be a potential defense; e.g., using a LiDAR to cross-check the distance to potential objects perceived by a camera sensor and assuming that both sensors are not compromised simultaneously.

6.2 Robustness Enhancement

Operating Points: A simple output comparison rule is used for attack detection but combination rules can be adopted. For example, the confidence score of the class of interest from the RNN can be used to make soft decisions (see Fig. 6c for our preliminary results). PercepGuard's result can be combined with other detectors' results by weights. Finally, an LSTM autoencoder can be used to calculate the reconstruction loss for each class and set a threshold for detection.

Moving Target Defense: We assumed white-box knowledge about the context on the part of the adversary: it is thus possible to bypass our defense with high enough cost or power. To prevent such attacks, the configuration of the measurement platform could be randomized (e.g., changing the camera's orientation or focal length, ego-vehicle speed, etc.), so that the context becomes a secret/difficult for an adversary to predict in real time (similar to [48]).

Post-Detection and More Robust Perception: Currently PercepGuard is intended as a standalone attack detector; however, the need exists for the perception engine of autonomous systems to accurately classify even spurious objects (i.e., to indicate that the spoofed object's true class). In principle PercepGuard could be extended to provide this functionality. For example, we may integrate PercepGuard into the perception pipeline using Gurel et al.'s weighted majority-voting framework [46], by regarding our classification result as a necessary condition (among others such as the shape and color etc.) to better reason the true class, with the RNN model's confidence scores as a part of the weight.

6.3 Limitations

False Positives: A data-driven approach has two inherent limitations. First, they are limited by training datasets. For example, BDD100K was collected in the US, making it unwise to train PercepGuard on this data set and deploy it in the UK. We argue that fine-tuning may be sufficient such that we do not have to retrain the model from scratch (see results on model transferability in Sec. 5.4). Second, there are corner cases for which FP may arise, which is why we recommend that an anomaly be issued to the planning module instead of an instruction to ignore the object (Fig. 3). To reduce FP and FN, a deeper network trained with a larger dataset can help. Also, we can train different networks for different scenarios, such as residential versus highway because objects with the same class move differently in different scenarios.

Limited Number of Classes: Our evaluation of PercepGuard involves small numbers of object categories, e.g., five categories from BDD100K for simulation and three from Carla for both simulation and experiments. We argue that PercepGuard is designed for attack detection instead of object classification, thus from the perspective of safety and security, only coarse-grained categorization is necessary. For example, cars, buses and trucks all belong to "vehicles" because they behave similarly, e.g., they can travel at high speeds which can be dangerous to pedestrians/bikes, another category that is less protected. In any case, we may potentially scale PercepGuard using commercial datasets that are likely to be much larger and contain a multitude of classes.

7 Conclusion

We introduced the first approach to use spatiotemporal consistency of an object to detect misclassification attacks against the perception module of an autonomous system. We utilized a data-driven approach, which extracts spatiotemporal features from the output of the object tracker and cross-checks the consistency with the class label outputed by the object detector. To enhance the adversarial robustness, we incorporate additional context information, such as ego-vehicle velocity. We evaluate PercepGuard against both defense-unaware and defense-aware attacks, using a real-world and simulated datasets, and experiments. The FPR is as low as 5%, while the average TPR is as high as 99% from simulation and 96% from real-world experiments. PercepGuard is agnostic to attack methods and object detectors and is extensible. Our comparison with two SOTA defenses confirms the advantages of incorporating temporal features.

Acknowledgments

The work was partly supported by Army Research Office (ARO) grant W911NF-21-1-0320, National Science Foundation (NSF) grant CNS-2144645, and through a gift by Qualcomm Technologies, Inc.

References

[1] Baidu, "Apollo," https://github.com/ApolloAuto/apollo.

[2] Waymo, "Waymo," https://waymo.com, 2020.

[3] commaai, "openpilot," https://github.com/commaai/openpilot, 2021.

[4] Starship, "The self-driving delivery robot," https:/www.starship.xyz.

[5] Amazon, "Prime air."

[6] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, F. Tramèr, A. Prakash, T. Kohno, and D. Song, "Physical adversarial examples for object detectors," in *USENIX Conf. on Offensive Technologies*, 2018.

[7] Y. Cao, N. Wang, C. Xiao, D. Yang, J. Fang, R. Yang, Q. 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 *IEEE Symposium on Security and Privacy*, 2021.

[8] G. Lovisotto, H. Turner, I. Sluganovic, M. Strohmeier, and I. Martinovic, "SLAP: Improving physical adversarial examples with short-lived adversarial perturbations," in *USENIX Security Symposium*, 2021.

[9] 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 *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020.

[10] Y. Man, M. Li, and R. Gerdes, "Ghostimage: Remote perception attacks against camera-based image classification systems," in *International Symposium on Research in Attacks, Intrusions and Defenses*, 2020.

[11] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao, "Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures," in *USENIX Security Symp.*, 2020.

[12] S. Köhler, G. Lovisotto, S. Birnbach, R. Baker, and I. Martinovic, "They see me rollin': Inherent vulnerability of the rolling shutter in cmos image sensors," *arXiv preprint arXiv:2101.10011*, 2021.

[13] X. Ji, Y. Cheng, Y. Zhang, K. Wang, C. Yan, W. Xu, and K. Fu, "Poltergeist: Acoustic adversarial machine learning against cameras and computer vision," in *IEEE Symp. on Security and Privacy*, 2021.

[14] J. Selvaraj, G. Y. Dayanıklı, N. P. Gaunkar, D. Ware, R. M. Gerdes, and M. Mina, "Electromagnetic induction attacks against embedded systems," in *Asia Conference on Computer and Comm. Security*, 2018.

[15] S. Köhler, R. Baker, and I. Martinovic, "Signal injection attacks against ccd image sensors," *arXiv preprint arXiv:2108.08881*, 2021.

[16] Y. Man, R. Muller, M. Li, Z. B. Celik, and R. Gerdes, "Evaluating perception attacks on prediction and planning of autonomous vehicles," in *USENIX Security Symposium, Poster Session*, 2022.

[17] S. Hoory, T. Shapira, A. Shabtai, and Y. Elovici, "Dynamic adversarial patch for evading object detection models," *arXiv:2010.13070*, 2020.

[18] J. Shen, J. Y. Won, Z. Chen, and Q. A. Chen, "Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under gps spoofing," in *USENIX Security Symposium*, 2020.

[19] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, "Savior: Securing autonomous vehicles with robust physical invariants," in *USENIX Security Symposium*, 2020.

[20] C. Xiao, R. Deng, B. Li, T. Lee, B. Edwards, J. Yi, D. Song, M. Liu, and I. Molloy, "Advit: Adversarial frames identifier based on temporal consistency in videos," in *IEEE Intl. Conf. on Computer Vision*, 2019.

[21] C. Xiang, A. N. Bhagoji, V. Sehwag, and P. Mittal, "Patchguard: A provably robust defense against adversarial patches via small receptive fields and masking," in *USENIX Security Symposium*, 2021.

[22] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[23] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," in *31st Conf. on Neural Information Processing Systems*, 2017.

[24] Carla, "Carla simulator," https://github.com/carla-simulator/carla.

[25] S. Ma, Y. Liu, G. Tao, W.-C. Lee, and X. Zhang, "Nic: Detecting adversarial samples with neural network invariant checking." in *Network and Distributed System Security Symposium*, 2019.

[26] S. Li, S. Zhu, S. Paul, A. Roy-Chowdhury, C. Song, S. Krishnamurthy, A. Swami, and K. S. Chan, "Connecting the dots: Detecting adversarial perturbations using context inconsistency," in *European Conference on Computer Vision*. Springer, 2020.

[27] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015.

[28] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement."

[29] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *IEEE Intl. Conf. on Image Processing*, 2016.

[30] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*. Springer, 2016.

[31] Tesla, "Autopilot," https://www.tesla.com/autopilot, 2020.

[32] Y. Jia, Y. Lu, J. Shen, Q. A. Chen, H. Chen, Z. Zhong, and T. Wei, "Fooling detection alone is not enough: Adversarial attack against multiple object tracking," in *Intl. Conf. on Learning Representations*, 2020.

[33] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, and K. Chen, "Seeing isn't believing: Towards more robust adversarial attack against real world object detectors," in *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[34] R. Muller, Y. Man, Z. B. Celik, M. Li, and R. Gerdes, "Physical hijacking attacks against object trackers," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.

[35] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *IEEE conference on computer vision and pattern recognition*, 2018.

[36] C. Zhou, Q. Yan, Y. Shi, and L. Sun, "Doublestar: Long-range attack towards depth estimation based obstacle avoidance in autonomous systems," in *USENIX Security Symposium*, 2022.

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

[38] M. Foruhandeh, Y. Man, R. Gerdes, M. Li, and T. Chantem, "Simple: Single-frame based physical layer identification for intrusion detection and prevention on in-vehicle networks," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019.

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

[40] A. P. Anand, H. Gokul, H. Srinivasan, P. Vijay, and V. Vijayaraghavan, "Adversarial patch defense for optical flow networks in video action recognition," in *IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020.

[41] T. Wu, L. Tong, and Y. Vorobeychik, "Defending against physically realizable attacks on image classification," in *International Conference on Learning Representations*, 2020.

[42] J. Zhang, Y. Zhang, K. Lu, J. Wang, K. Wu, X. Jia, and B. Liu, "Detecting and identifying optical signal attacks on autonomous driving systems," *IEEE Internet of Things Journal*, 2021.

[43] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote attacks on automated vehicles sensors: Experiments on camera and lidar," *Black Hat Europe*, vol. 11, 2015.

[44] M. Yin, S. Li, Z. Cai, C. Song, M. S. Asif, A. K. Roy-Chowdhury, and S. V. Krishnamurthy, "Exploiting multi-object relationships for detecting adversarial attacks in complex scenes," in *International Conference on Computer Vision*, 2021.

[45] M. Yin, S. Li, C. Song, M. S. Asif, A. K. Roy-Chowdhury, and S. V. Krishnamurthy, "Adc: Adversarial attacks against object detection that evade context consistency checks," in *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022.

[46] N. M. Gürel, X. Qi, L. Rimanic, C. Zhang, and B. Li, "Knowledge enhanced machine learning pipeline against diverse adversarial attacks," in *International Conference on Machine Learning*, 2021.

[47] X. Wang, S. Li, M. Liu, Y. Wang, and A. K. Roy-Chowdhury, “Multi-expert adversarial attack detection in person re-identification using context inconsistency,” in *Intl. Conf. on Computer Vision*, 2021.

[48] A. Amich and B. Eshete, “Morphence: Moving target defense against adversarial examples,” in *Annual Comp. Sec. Applications Conf.*, 2021.

[49] 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 *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.

[50] N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, and G. Taylor, “Learning human identity from motion patterns,” *IEEE Access*, vol. 4, 2016.

[51] C. Wu, K. He, J. Chen, Z. Zhao, and R. Du, “Liveness is not enough: Enhancing fingerprint authentication with behavioral biometrics to defeat puppet attacks,” in *USENIX Security Symposium*, 2020.

[52] A. Acar, H. Aksu, A. S. Uluagac, and K. Akkaya, “A usable and robust continuous authentication framework using wearables,” *IEEE Transactions on Mobile Computing*, vol. 20, no. 6, pp. 2140–2153, 2020.

[53] M. Zhou, Q. Wang, X. Lin, Y. Zhao, P. Jiang, Q. Li, C. Shen, and C. Wang, “Presspin: Enabling secure pin authentication on mobile devices via structure-borne sounds,” *IEEE Transactions on Dependable and Secure Computing*, 2022.

[54] N. K. Shaydyuk and T. Cleland, “Biometric identification via retina scanning with liveness detection using speckle contrast imaging,” in *IEEE International Carnahan Conf. on Security Technology*, 2016.

[55] L. Mou, C. Zhou, P. Zhao, B. Nakisa, M. N. Rastgoo, R. Jain, and W. Gao, “Driver stress detection via multimodal fusion using attention-based cnn-lstm,” *Expert Systems with Applications*, vol. 173, 2021.

[56] S. Vhaduri and C. Poellabauer, “Multi-modal biometric-based implicit authentication of wearable device users,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, 2019.

[57] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv preprint arXiv:1412.6572*, 2014.

[58] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, “Authentication in distributed systems: Theory and practice,” *ACM Transactions on Computer Systems*, 1992.

[59] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, 1997.

[60] A. Graves, “Supervised sequence labelling,” in *Supervised sequence labelling with recurrent neural networks*. Springer, 2012.

[61] Apple, “Maps,” https://www.apple.com/maps/.

[62] F. Chollet *et al.* Keras. https://github.com/fchollet/keras.

[63] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: A system for large-scale machine learning,” in *USENIX symposium on operating systems design and implementation (OSDI)*, 2016.

[64] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations*, 2015.

[65] R. Muller, Y. Man, Z. B. Celik, M. Li, and R. Gerdes, “Drivetruth: Automated autonomous driving dataset generation for security applications,” in *International Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2022.

[66] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014.

[67] Y. Ma, J. Sharp, R. Wang, E. Fernandes, and X. Zhu, “Sequential attacks on kalman filter-based forward collision warning systems,” in *AAAI Conference on Artificial Intelligence*, 2021.

[68] Y. Shoukry, P. Martin, P. Tabuada, and M. Srivastava, “Non-invasive spoofing attacks for anti-lock braking systems,” in *International Conference on Cryptographic Hardware and Embedded Systems*, 2013.

[69] A. Z. Mohammed, Y. Man, R. Gerdes, M. Li, and Z. B. Celik, “Physical layer data manipulation attacks on the can bus,” in *Intl. Workshop on Automotive and Autonomous Vehicle Security (AutoSec)*, 2022.

[70] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *IEEE conference on computer vision and pattern recognition*, 2017.

[71] A. Kurakin, I. Goodfellow, and S. Bengio, “Adversarial machine learning at scale,” in *Intl. Conf. on Learning Representations*, 2016.

[72] A. Saha, A. Subramanya, K. Patil, and H. Pirsiavash, “Role of spatial context in adversarial robustness for object detection,” in *CVPR Workshop on Adversarial Machine Learning in Computer Vision*, 2020.

[73] LG, “75-inch xs4g-b series uhd window facing high brightness display.”

[74] Epson, “Pro l1490u.”

[75] Barco, “Xdl-4k75.”

A Appendix

A.1 Spatiotemporal Statistics

Intuitively, spatiotemporal characteristics of different object types are distinctive. From the spatial perspective, a car is more likely to appear at the center of the dashcam image while the pedestrians appear more on either sides. When we consider a dynamic scene where a car is driving at a high speed, a pedestrian on the sidewalk would likely appear closer to the image center as they are far away, and then gradually moves to the left/right side as they get closer. Meanwhile, a car may stay at the image center if it is the preceding car, or it may first appear at the center top but rapidly disappears on the lower left corner if it is an oncoming car.

We verify these empirical observations using a real-world driving dataset, BDD100K [22]. We first extract 2-D histograms of bounding box center coordinates from the BDD100K dataset where videos were recorded from the dashcam on moving vehicles (See Sec. 5.1 for more details of BDD100K). The results are shown in Fig. 10. The center coordinates (x,y) of bounding boxes are normalized and recorded as frequencies represented by different gray scales.

Comparing Fig. 10a with Fig. 10b for single-frame histgorams, we indeed observe that cars incline to be located at the center (of a vehicle camera's field of view) and people are often distributed on either side of the road. Moreover, since BDD100K was collected in the U.S., pedestrians appear on the right side more frequently than on the left, as they are closer to the ego vehicle.

To present the movement patterns of objects over time, we select those objects whose initial coordinates are at the center, and we show their coordinates in the next fifth frame after the initial one in Figs. 10c and 10d, for cars and pedestrians respectively. We can observe that cars tend to stay at the center, meanwhile pedestrians (on the side walks) tend to move rightwards because when they are far away they appear at the center but as the ego vehicle drives forward they get closer thus reappear on the right side, and pedestrians crossing the street appears on either sides of the center.

(a) Residential (b) Highway

Figure 8: Carla Simulated World

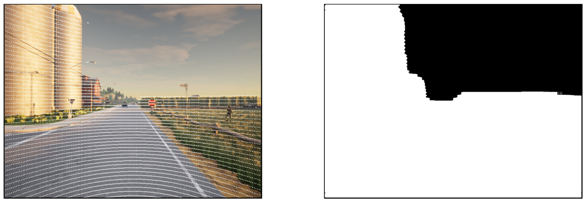
In summary, statistics suggest that there are visual differences of the bounding box behavior of different object categories, which implies that one may use them to distinguish one object category from another.

A.2 Safety in Real-World Experiments

We regard safety as the highest priority in real-world experiments. We carefully chose the location and time to conduct the experiments so that there was minimal traffic. For example, we chose to conduct experiments on holidays and weekends when there were not many pedestrians and vehicles in garages or on the street. The experiment was set up and tested on the highest level of a parking garage with no moving vehicles or pedestrians. Once we confirmed the attack setup was secured (e.g., by rocking the vehicle to ensure the setup was firmly in place), we moved on to the lower levels and then to the streets, where the route was carefully planned in advance such that the roads were not crowded, the speed limit was 25 MPH or below, and there was only one lane for each traffic direction. We did not stay in one street for more than three minutes to avoid blocking normal traffic, and we maintained speeds at most 5 MPH below the speed limit.

To make sure the LCD monitor was securely mounted on the vehicle, we used a tailgate TV hitch mount and a pickup truck with a hitch for the mount to attach. We made sure each joint of the mount was fastened tightly before each round of experiments. We additionally used bungee cords and duct tape to prevent any relative movement of the monitor to the truck (such as panning or tilting). They were also used to secure the cables, preventing them from snagging on objects.

Both vehicles were entirely operated by experienced human drivers with no autonomous driving module installed (since the experiments were conducted only to collect data for offline testing). We avoided aggressive acceleration and deceleration, and minimized turns on the road to further reduce the chance of the monitor falling. All traffic rules were obeyed strictly. The driver in the following vehicle paid extra attention to the monitor for any sign of instability. At any time, the following vehicle was directly behind the preceding vehicle without any other vehicle intercepting, and the phone communication between the two drivers was maintained so that in case of any emergency, such as the monitor falling off, both vehicles could stop immediately without any vehicle rolling over it. The experiment executors are experienced with similar outdoor experiments with vehicles in motion. The experiments were supervised by professors with many years of experience in autonomous vehicle research.



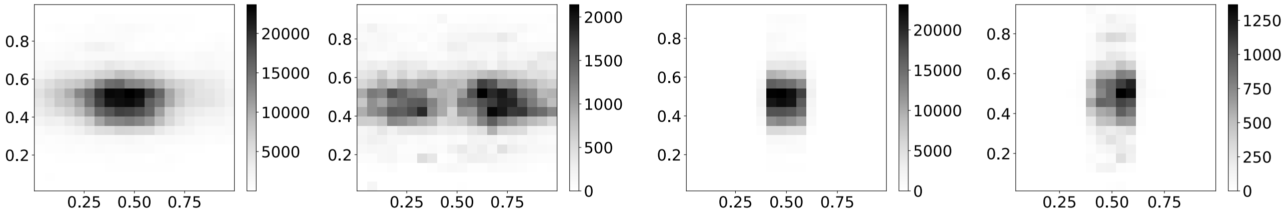
(a) RGB Pixels and Lidar Points (b) Perturbation Mask

Figure 9: From the Carla dataset, the perturbation mask is based on the LiDAR points. The white part of the mask indicates the perturbable area. For BDD100K, masks are based on image segmentation which are similar.

A.3 Baseline Comparison Details

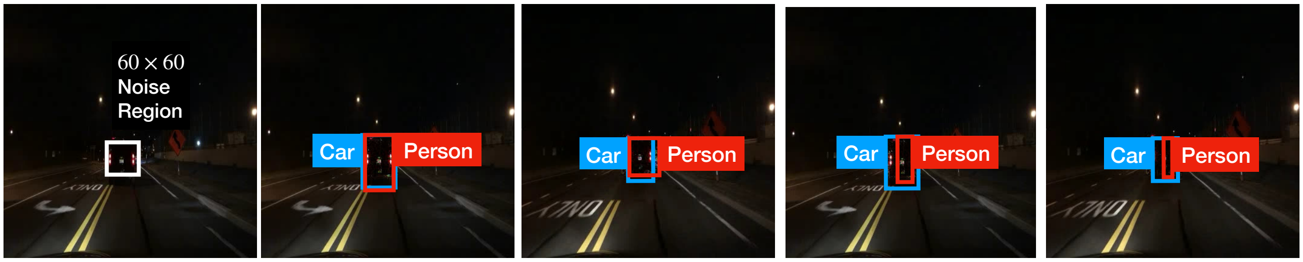
NIC NIC [25] is proposed to detect adversarial examples against image classifiers by checking the network invariant. It produces a binary detection for each individual video frame. Specifically, when an image is fed to an image classifier, NIC obtains the activation values of each individual layer (value invariants, V_i), and every two consecutive layers (provenance invariants, $P_{i,j}$), where i and j are the layer indices. They first use PCA to reduce the dimension of these invariants and obtain V_i' and $P_{i,j}'$. For each V_i', an SVM is used to produce a score v_i. For each $P_{i,j}'$, they first feed it to a fully connected layer that has M neurons where M is the number of classes, and then feed the M activation values to an SVM for a score $p_{i,j}$. Finally, an SVM is used to classify all v_i and $p_{i,j}$ into a binary decision (on whether the input image is adversarial).

SCEME Similar to NIC, SCEME also works on individual frames but it produces a soft decision score for each object in the frame by verifying the contextual consistency among objects, backgrounds and scenes. For each region proposal in a frame, SCEME employs context-aware Faster-RCNN [26] to extract a context profile $u = [r, \gamma_{u1}, \gamma_{u2}, \gamma_{r1}, \gamma_{r2}]$, and then feed it to an autoencoder for reconstruction losses. If the loss is above a certain threshold, SCEME claims the object (if the region contains an object) as adversarial. There is one autoencoder for each object class. To train them, SCEME uses context profiles collected from benign images to learn the distribution of a benign object's profile for a given class; thus at test time, it will be able to identify misclassified objects.



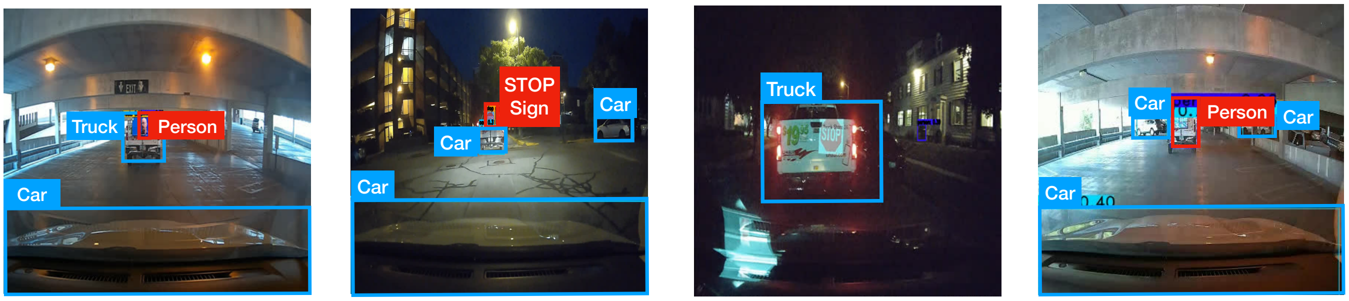
(a) Car at one frame (b) Person at one frame (c) Car between two frames (d) Person between two frames

Figure 10: Spatial and temporal histograms of bounding box center coordinates, i.e., x and y, of cars and people from BDD100K. The gray scale of each square represents the frequency of that coordinate. (a)(b) The spatial distributions of all time. (c)(d) The conditional distributions given that they first appear around the image center in the preceding frame, i.e., temporal distributions.



(a) Noise Region (b) $t = 1$ (c) $t = 2$ (d) $t = 1$ (e) $t = 2$

Figure 11: A video from BDD100K where the defense-unaware attack fails to evade PercepGuard's detection but the defense-aware attack succeeds. These bounding boxes are recreated based on actual predictions for better presentation. (a) The vehicle at the image center is overlaid with an adversarial patch. (b)(c) The defense-unaware attackers effectively alters the classification result of a car into a person but the bounding boxes are left similar to a car's bounding boxes, therefore the attack is detected by PercepGuard as the track and the class predictions are inconsistent. (d)(e) On the other hand, since the defense-aware attackers considers PercepGuard when they are solving for the optimal adversarial patches, the optimizer finds those patches that not only alters the classification results, but also "shrinks" the bounding boxes to be vertical (as if a person's bounding box would be) so that PercepGuard regards the track to be consistent with the object class.



(a) Person (b) Stop sign on monitor (c) Projected stop sign (d) Adversarial patch

Figure 12: Real image attack examples. Bounding boxes recreated for better presentation. (a)(b) Both the person/stop-sign images and the truck are recognized by YOLOv3. (c) Projected stop sign obscured by the truck's paint thus not detected by YOLOv3. (d) The adversarial patch alters the truck into a person.