# Dynamic Defense for Car-Borne LiDAR Vehicle Detection

Yihan Xu
Nanyang Technological University
Singapore

Dongfang Guo
Nanyang Technological University
Singapore

Qun Song
Singapore University of Technology and Design
Singapore

Yang Lou
City University of Hong Kong
Hong Kong SAR, China

Yi Zhu
Wayne State University
Michigan, USA

Jianping Wang
City University of Hong Kong
Hong Kong SAR, China

Chunming Qiao
SUNY Buffalo
New York, USA

Rui Tan
Nanyang Technological University
Singapore

## Abstract

Adversarial attacks with real objects or lasers on car-borne LiDAR-based object detection are concerning. The existing defense approaches are often designed to address specific attacks and short of considering adaptive attackers who may adapt based on all available information about the deployed defense to maximize attack effect. This paper proposes *Hyper3Def*, a new defense for the function of detecting vehicle objects, which uses a Hypernet to generate an ensemble of multiple new detection models when needed at run time. The detection results of these models are fused to give the final result. As a dynamic defense, Hyper3Def revokes an important basis of the adaptive attack, i.e., the object detection model is needed to plan effective adversarial perturbations. Evaluation based on open data and real-world experiments with embedded system implementation show that, when confronting adaptive attacks, Hyper3Def outperforms various baseline defenses including the adversarial training, which is often cited as the state of the art.

## CCS Concepts

• **Computer systems organization → Embedded and cyber-physical systems**; • **Security and privacy → Systems security**.

**ACM Reference Format:**
Yihan Xu, Dongfang Guo, Qun Song, Yang Lou, Yi Zhu, Jianping Wang, Chunming Qiao, and Rui Tan. 2025. Dynamic Defense for Car-Borne LiDAR Vehicle Detection. In *The 23rd Annual International Conference on Mobile Systems,Applications and Services(MobiSys '25), June 23–27, 2025, Anaheim, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3711875.3729157

## 1 Introduction

Light detection and ranging (LiDAR) is a crucial sensor for autonomous driving. The point cloud yielded by the LiDAR carried by the *ego vehicle* (i.e., the vehicle implementing autonomous driving) captures the three-dimensional (3D) shapes and distances of the objects within its surroundings. The 3D object detection (3D-OD) function, implemented based on LiDAR, has received increasing research attention [30] and has been integrated into autonomous driving products [1, 2, 5]. While deep learning is the state-of-the-art approach to 3D-OD, it is vulnerable to adversarial perturbations [16]. Research has shown that by injecting infrared lasers into the LiDAR sensor [9, 18, 21, 33, 36] or placing real objects at certain locations in the scene [7, 8, 39, 40, 45, 53, 54, 56], LiDAR sensing can be misled. As autonomous driving is safety-critical, it is imperative to understand and enhance the security of car-borne LiDAR sensing against these physically implementable adversarial attacks.

Various defense approaches for LiDAR sensing have been proposed. Earlier studies [32, 48, 52] apply data preprocessing such as down-sampling and statistical outlier removal, hoping to destruct the adversarial perturbations. However, these heuristic techniques, which may be effective against naturally occurring noises and outliers, were not designed under the adversarial setting. Other defense approaches use prior knowledge about the properties of authentic objects' point clouds (e.g., shadows and spatiotemporal consistency across multiple frames) to detect attacks [12, 19, 20, 29, 36, 44, 46]. The adversarial training defense [16] includes adversarial samples with correct labels into the model training dataset. However, both the prior knowledge-based defense and adversarial training lack generalizability in that they are only effective against the specific attack they are designed to counter. Adversarial training may consider multiple attack types, but may suffer degraded defense performance against individual attack types [38].

In this paper, we follow the *dynamic defense* strategy [11, 51] to design a new defense approach for the ego vehicle's function of detecting vehicle objects nearby. Dynamic defense continuously changes the system configurations to increase the attacker's difficulty in obtaining certain information needed about the protected system to launch effective attacks. The prospect of dynamic defense is based on an observation that, the laser-based and object-based attacks require access to the sensing model, in either the black-box or white-box form [8, 21, 54, 56]. The work [35] has shown that, for image classification, adversarial attacks have limited transferability to the models with distinct parameters. In this paper, we aim to utilize a run-time 3D-OD model distinct from that obtained by the attacker for defense. To realize this, we propose *Hyper3Def* based on

*MobiSys '25, June 23–27, 2025, Anaheim, CA, USA*
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1453-5/2025/06
https://doi.org/10.1145/3711875.3729157

the Hypernet technique [17], which is a neural network model that generates the weights of a model accomplishing a certain task. Randomly seeding the generation leads to randomness of the weights, which is undesirable to the attacker. In addition, as the generation is fast, Hyper3Def can generate an ensemble of multiple models for an inference process and yield their fused result to increase robustness. Intuitively, it is harder for the attacker to fool multiple models than a single model.

However, the design of Hyper3Def needs to address the following issues. First, while Hypernet has been used for image classification, training a Hypernet to generate the entire 3D-OD model with a size much larger than image classification models bears significant technical challenges. Second, the multiple models generated by the Hypernet should be *diverse*, because similar models do not present the desired complexity against the attacker. The third issue is engendered when addressing the second issue. That is, when the Hypernet is trained toward the joint goal of optimizing *clean accuracy* (i.e., accuracy in the absence of attacks) and increasing diversity of generated models, it becomes inferior in terms of clean accuracy compared with the original model trained solely for optimizing clean accuracy. Fourth, as attacks are rare events, it is desirable to avoid the unnecessary defense overhead when there is no attack. This requires an effective attack detector to activate the Hypernet-based defense.

To tackle the first issue, we select a subset of model layers as the target of generation. Our extensive sensitivity study shows that selecting the maximum preceding convolutional layers subject to the convergence of Hypernet training is an effective strategy. To address the second issue, we propose to perform *in-batch independent data augmentation* and apply a similarity loss customized for 3D-OD. Both mechanisms are beneficial for the diversity of the generated models. In addition, we propose a clustering-based approach to fuse the inference results of multiple generated models. To address the third issue, we propose to calibrate the fused result with the original model's result. In the absence and presence of attack, the calibration likely yields the original model's result and the fused result, respectively. This restores the clean accuracy effectively. To address the fourth issue, we reuse the Hypernet technique to design a lightweight ensemble-based attack detector and use its positive detection result to activate the defense described above.

We evaluate Hyper3Def based on the KITTI 3D-OD dataset [15] against three attacks, i.e., object-based car hiding (OCH), laser-based car hiding (LCH), and laser-based car creating (LCC). We employ six baselines including adversarial training and a heuristic preprocessing defense called simple random sampling. The evaluation is conducted under the strong adversarial setting of *adaptive attack*. For instance, regarding the adversarial training defense, the attacker can choose the most effective attacks among OCH, LCH, and LCC based on the knowledge of which attack or the mix of all attacks is adopted in adversarial training. A game-theoretic analysis on the obtained results shows that the adversarial training against the mix of all three attacks is the Nash equilibrium choice of the defender, which, however, gives inferior defense performance compared with Hyper3Def. This result shows that the effectiveness of adversarial training needs to be scrutinized based on the threat model. It is superior when the attacker has a sole option, but may be inferior when the attacker has multiple options and is adaptive.

We also conduct real-world experiments with Hyper3Def implemented on an embedded GPU platform. We launch OCH attacks to hide a real vehicle from our victim LiDAR and an ego vehicle operated by a third party in an organized challenge activity. We had no information about the third-party's vehicle, except its LiDAR hardware model. Results show that Hyper3Def maintains good defense performance as in our evaluation based on KITTI dataset.

A succinct summary of our contributions is as follows: (1) We present a first dynamic defense system for LiDAR-based vehicle detection against physically implementable adversarial attacks; (2) Comparative evaluations based on open data and real-world experiments provide comprehensive understanding on the effectiveness of various defense approaches.

*Paper organization:* §2 reviews related work. §3 presents threat model. §4 presents Hyper3Def. §5 and §6 present the results of evaluation based on open data and real-world experiments, respectively. §7 discusses several relevant issues. §8 concludes this paper.

## 2 Background and Related Work

### 2.1 Adversarial Attacks on LiDAR Sensing

*Adversarial attack* has been extensively studied for image classification [10, 16, 22, 28] and 2D object detection [13, 27, 34, 42]. Regarding LiDAR sensing, existing studies [25, 26, 43, 47, 50] consider *point-wise* attacks that shift or detach points or add new points in the point cloud. However, directly tampering with the points requires the attack to have real-time access to the victim vehicle's data processing pipeline, making it questionable in practicability.

To improve the attack realism, recent studies consider physical attacks, including *laser-based* and *object-based* attacks, which are more practical for real-world implementation. The laser-based attacks [9, 18, 21, 33, 36] interfere with the victim LiDAR by injecting infrared laser beams into the LiDAR sensor to add extra points in a strategically positioned locality within the point cloud. The object-based attacks [7, 8, 39, 40, 45, 54, 56] deploy real objects to inject additional points into the point cloud, thereby spoofing LiDAR sensing tasks. The studies [7, 8, 39, 40, 45] design 3D-printable adversarial objects; [54, 56] identify *adversarial locations* and place common objects such as cardboards at such locations.

In terms of the attack effect on 3D-OD, existing physically implementable attacks can be categorized into three types. The *car hiding* attack disables the system to detect a real car in its sensing range [21, 40, 53, 56]. The *car creating* attack misleads the system to detect a "ghost car" that does not exist in reality [9, 21, 36]. The *undetected object* attack is similar to *car hiding* attack, but to hide non-vehicle objects such as traffic cones [8]. In this paper, we focus on car creating and car hiding due to the prevalence of cars on roads and the significant safety risks posed by their incorrect detection.

### 2.2 Defense for LiDAR Sensing

Defenses against adversarial attacks on LiDAR sensing can be categorized into *prior knowledge-based defense*, *heuristic preprocessing*, and *adversarial training*.

**Prior knowledge-based defense** detects the adversarial attacks by checking whether the point cloud data conforms to certain prior knowledge. The defense approaches in [12, 20, 36, 44, 46] identify ghost objects by checking whether the point cloud parts of the objects conform to known physical properties of the authentic objects [20, 36, 44] or demonstrate spatiotemporal consistency across frames [12, 46]. The defense in [19] detects car hiding by identifying 3D shadows projected by the hidden cars. However, prior knowledge-based defenses have two drawbacks. First, their effectiveness is often limited to specific attack types. For instance, while point density check, as in [44], may detect ghost vehicles, it is ineffective against car hiding. This allows attackers to circumvent the defense by switching to a different attack method. Second, the complexity of designing and implementing these defenses varies significantly depending on the attack being countered. Defenses against car creating typically inspect the points within each bounding box given by the unhardened 3D-OD model and/or the accompanying shadow to detect ghost cars. Differently, defenses against hiding attacks, where the object is missed in the detection, necessitate a computationally expensive search across the entire point cloud frame to locate the hidden vehicles. This process is slow (e.g., 36.5 seconds per scene [19]) and in general does not meet the real-time processing requirements of autonomous driving.

**Heuristic preprocessing** purifies or perturbs the input data to nullify the adversarial perturbations. The simple random sampling (SRS) and statistical outlier removal (SOR) are two basic approaches based on random sampling and clustering, respectively, to preprocess point clouds [52]. The work [53] smoothens irregular lines in LiDAR data to deal with a specific car hiding attack. The study in [52] combines SOR with a point up-sampler to form a deep learning-based denoiser to purify the input data. In [37, 48], a diffusion network is used to noisify and then denoisify the point cloud, aiming at destructing the adversarial perturbations. The denoiser- and diffusion-based purification techniques [48, 52] are designed to process the point cloud containing a single object. Differently, a point cloud captured by a car-borne LiDAR is sparse and includes multiple objects. This leads to a chicken-or-the-egg situation. That is, to apply these purification techniques, we need to detect the objects first and then isolate the point cloud parts containing objects as inputs to the purification, which, however, should be applied before the object detection. SRS, which preprocesses the data by randomly sampling a portion of the points, can process the entire frame containing multiple objects. It is not subject to the above dilemma. The defense proposed in [49], referred to as RL-Remove, addresses this dilemma by introducing a mechanism to first extract suspicious point clusters in front of the ego vehicle and then eliminate adversarial points through a reinforcement learning-based search. However, it is specifically designed to counter hiding attacks and does not address creating attacks. We adopt SRS and RL-Remove as baseline approaches when evaluating Hyper3Def.

**Adversarial training** enhances model robustness by incorporating adversarial examples with their correct labels into the training dataset. It achieves good defense performance against adversarial perturbations that follow the same distributions of those contained in the adversarially augmented training dataset [16]. However, when confronting out-of-distribution adversarial perturbations, its defense performance drops. In this paper, adversarial training is employed as a baseline approach.

In summary, while existing defense methods demonstrate effectiveness against specific attack types, most of them are vulnerable when confronting adversaries capable of launching diverse attacks or possessing knowledge of the defense mechanisms. This paper aims to achieve better defense performance under a strong adversarial setting where the attacker has all static information about the deployed defense.

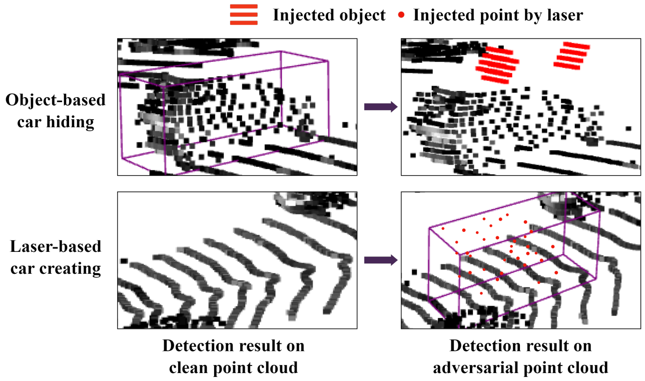


**Figure 1: Examples of attacks considered in this paper. Point clouds are perceived by the victim vehicle.**

Under the recent *certified robustness* paradigm [24], a robustly trained model defeats all adversarial perturbations with intensities within a specified bound. However, robust training cannot scale with model size. So far, it can only handle up to 100k neurons [24], but LiDAR sensing models often have at least millions of neurons.

## 3 Threat Model

In this paper, we consider *adaptive attacks*, which are explained in terms of attack types and attack capabilities.

**Attack types.** We consider physically implementable attacks on LiDAR-based 3D-OD, including laser-based and object-based attacks. In terms of attack effect, we focus on car hiding and car creating. Figure 1 shows examples of the two attack types. The vehicle carrying the victim LiDAR is regarded as the *victim vehicle*. In this paper, ego vehicle and victim vehicle refer to the same vehicle. For car hiding, the real vehicle to be hidden using adversarial objects or laser injections is regarded as the *target vehicle*.

**Attack capability.** In this paper, the *original model* refers to the 3D-OD model that is optimized in terms of clean accuracy and is not hardened against adversarial attacks. An undefended system uses the original model to perform 3D-OD. We consider a strong adversarial setting in which the attacker is *adaptive*, i.e., the attacker attempts to optimize the attack effectiveness based on all the obtained information including those related to defense. Taking an undefended system for instance, the attacker may obtain the original model in either white-box or black-box form. This can be achieved by memory extraction or compromising the employees of the vehicle manufacturer via social engineering. With the obtained model, the attacker plans the attacks. Under the white-box setting, the attacker can access the gradients of the model to optimize the efficiency of the attack planning. Under the black-box setting, the attacker queries the model when planning the attack. When a defense is deployed, the attacker may obtain no or different amounts of information about the defense. With the obtained information, the attacker tries to adapt the attack to maximize attack effectiveness.

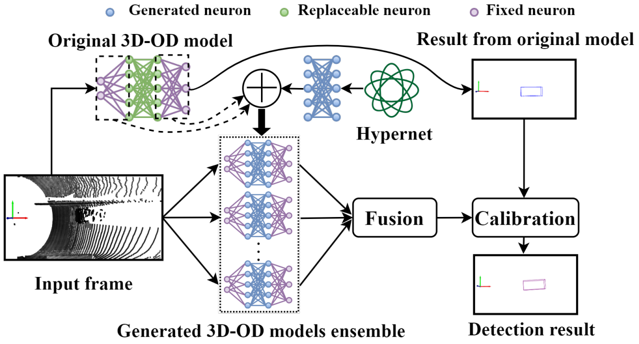


**Figure 2: Hyper3Def inference workflow.**

Since this attack adaptation depends on the details of the defense, we will defer its description to when a defense is introduced.

## 4 Proposed Defense: Hyper3Def

This section presents our proposed defense called Hyper3Def. §4.1 overviews Hyper3Def. §4.2 presents the detailed design of the Hypernet used by Hyper3Def. §4.3 presents the fusion and calibration of the 3D-OD results given by the Hypernet-generated ensemble. §4.4 presents an improved version of Hyper3Def that further integrates a dynamic defense-based attack detector to avoid unnecessary execution of the ensemble in the absence of attacks.

### 4.1 Overview of Hyper3Def Inference

Figure 2 illustrates Hyper3Def's inference workflow. Let $\Psi(\mathbf{z}|\Omega)$ denote a Hypernet that generates the weights of the 3D-OD model based on a random vector $\mathbf{z} \in \mathbb{R}^{256}$ sampled from a normal distribution, where $\Omega$ is the weights of the Hypernet obtained in offline training. For each dynamic defense process, the ego vehicle uses the Hypernet to generate $M$ distinct 3D-OD models to form an ensemble of $\{\Theta_1, \Theta_2, \ldots, \Theta_M\}$, where $\Theta_m = \Psi(\mathbf{z}_m|\Omega)$. In addition, we let $\Theta_0$ denote the weights of the original model. Given a potentially adversarial point cloud input $\mathbf{x}$, each of the original model and the $M$ models is executed to give a 3D-OD result: $\mathbf{y}_m = \Phi(\mathbf{x}|\Theta_m)$, where $m \in [0, M]$, $\Phi$ represents the architecture of the 3D-OD model, $\mathbf{y}_m$ is a collection of the 3D bounding boxes. Each bounding box is described by a 7-tuple including a 3D coordinate, size (i.e., length, width, height), and an orientation. Lastly, the results of the $M$ models are fused and calibrated to yield the final vehicle detection result denoted by $\mathbf{y}^*$. Specifically, $\mathbf{y}^* = \xi_c(\mathbf{y}_0, \xi_f(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_M))$, where $\xi_f$ and $\xi_c$ are the fusion and calibration functions. The $M$ should be set to balance robustness and computation cost. The calibration restores the clean accuracy in the absence of attacks. Note that as the current design of Hyper3Def focuses on vehicle detection, the $\xi_f$ only fuses vehicle detection results and skips all non-vehicle objects detected. In §7, we will discuss the extension of Hyper3Def to address non-vehicle objects.

We assume (require) that the model ensemble can be updated at a speed faster than that at which the attacker can obtain the latest model ensemble, plan and deploy the corresponding attack, which takes at least several minutes. From our results obtained on a car-borne class GPU, a single update can be completed within 8.5 ms. Thus, even if a low update frequency (e.g., every tens of frames) is applied to reduce compute overhead, the update period is only a few seconds, shorter than the time needed by the attacker.

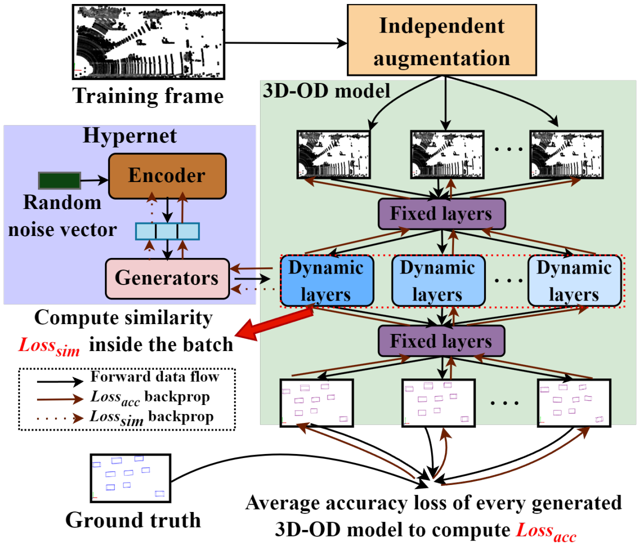


**Figure 3: Training of Hypernet used by Hyper3Def.**

Hyper3Def addresses adaptive attacks. If the attackers can only obtain the original model, they plan the attack based on the original model. If they obtain an outdated model generated by the Hypernet, they optimize the perturbations against the obtained model. If they obtain the Hypernet $\Psi(\cdot|\Omega)$, they generate an ensemble offline and optimize the perturbations against the generated ensemble.

### 4.2 Hypernet Design and Training

*4.2.1 Design Overview.* A hypernet $\Psi(\mathbf{z}|\Omega)$ [17, 31] consists of an encoder and multiple independent generators. The encoder maps an input random vector $\mathbf{z}$ sampled from a multivariate normal distribution to a latent code. The latent code is then divided into multiple parts as the input for each generator. A generator outputs the parameters of a layer of the target model. In literature, the largest models generated by Hypernet are in the scale of about one million parameters [31, 35]. However, 3D-OD models are often in the scale of more than five million parameters. To enable Hypernet training, we employ Hypernet to generate a subset of layers of the 3D-OD model, while the remaining layers are from the original model and fixed during both the training and inference of Hyper3Def. The selection of the generation target layers (including quantity and positions) is a design factor to be evaluated in §5.

Figure 3 illustrates Hypernet training. In the $t$th forward pass, for a given clean training frame, *independent data augmentation* is performed to generate $M$ augmented frames as the inputs to the $M$ generated models. The $m$th model, parameterized by $\Theta^t_m$, is formed by the fixed layers from the original model and the dynamic layers generated by the Hypernet with an independent random vector $\mathbf{z}^t_m$. The *accuracy loss*, denoted by $Loss^t_{acc}$, is computed based on these models' 3D-OD results and the ground truth. In addition, the *similarity loss*, denoted by $Loss^t_{sim}$, is computed based on the generated models' parameters to reflect the diversity of the models in favor of robustness against attacks. The final loss, $Loss^t_{final} = Loss^t_{acc} + Loss^t_{sim}$, is used to guide the backpropagation for optimizing the Hypernet parameters $\mathbf{\Omega}$. The independent data augmentation and the similarity loss drive the Hypernet training to generate diverse models.

*4.2.2 Independent Data Augmentation and Loss Design.* **Independent data augmentation**: In the existing applications of Hypernet [17, 31, 35], during the training phase, the Hypernet-generated models are fed with the same inputs. In this paper, we propose to leverage the existing point cloud augmentation approaches to diversify the inputs to the generated models. Specifically, in the $t$th forward pass of the training, the input to the $m$th generated model, denoted by $\mathbf{x}^t_m$, is obtained by independently performing a series of data augmentation operations on the given training frame $\mathbf{x}^t$. The operations include adding object noise, random global rotation, random global flipping, and etc. As a result, the generated models are fed with different inputs derived from the same training frame.

**Similarity loss**: Diversity of the $M$ generated models in each forward pass is in favor of dynamic defense. The reason is as follows. Because each model is generated from an independent noise sample $\mathbf{z}$, if the diversity of the $M$ models is low, the diversity among the models generated in different forward passes is also low, resulting in less dynamism against adaptive attacks. Thus, we use cosine similarity among the vectors flattened from the $M$ models' parameters (denoted by $\widetilde{\mathbf{\Theta}}^t_m$, $m = 1, \ldots, M$) as a loss term: $Loss^t_{sim} = \frac{2}{M(M-1)} \left( \sum_{m=1}^{M-1} \sum_{n=m+1}^{M} \exp \left( S_{\cos}(\widetilde{\mathbf{\Theta}}^t_m, \widetilde{\mathbf{\Theta}}^t_n) \right) \right)$, where $S_{\cos}(\cdot, \cdot)$ represents the cosine similarity.

**Accuracy loss**: Let $\mathbf{y}^t_{gt}$ denote the ground truth bounding boxes of all objects in the training frame $\mathbf{x}^t$. The accuracy loss is $Loss^t_{acc} = \frac{1}{M} \sum_{m=1}^{M} F_{loss}(\mathbf{y}^t_{gt}, \Phi(\mathbf{x}^t_m | \mathbf{\Theta}^t_m))$, where $F_{loss}$ is the loss function used to train the original model.

*4.2.3 Model Diversity in Making Vehicle Detection Errors.* It is undesirable if the generated models produce the same error, i.e., false negative (FN) or false positive (FP), for the same input, because the fusion of their identically erroneous results will also be wrong. We adopt the following definitions of FN and FP for 3D-OD based on the Intersection over Union (IoU) in the bird's-eye view (BEV), denoted by $\text{IoU}_{BEV}$. Specifically, given two bounding boxes $b_1$ and $b_2$, $\text{IoU}_{BEV}(b_1, b_2) = \frac{\hat{b}_1 \cap \hat{b}_2}{\hat{b}_1 \cup \hat{b}_2}$, where $\hat{b}$ represents $b$'s area projection on the ground plane. A model makes an FN error regarding a real vehicle with ground truth bounding box of $b_{gt}$ if $\text{IoU}_{BEV}(b, b_{gt}) < 0.5$ for every bounding box $b$ predicted by this model. Differently, a vehicle bounding box $b$ predicted by the model is an FP error if $\text{IoU}_{BEV}(b, b_{gt}) < 0.5$ for every vehicle's ground truth bounding box $b_{gt}$. The 0.5 threshold is a standard setting in 3D-OD benchmarks such as the KITTI 3D-OD dataset [15].

We use two metrics called $\text{IoU}_{FN}$ and $\text{IoU}_{FP}$ to measure the diversity between two models in making vehicle detection errors [41]. We denote the FN sets of the $m$th and $n$th models in processing multiple point cloud frames by $FN_m$ and $FN_n$. Then, $\text{IoU}_{FN} = \frac{FN_m \cap FN_n}{FN_m \cup FN_n}$, where $b \in FN_m \cap FN_n$ if $b$ is an FN of both models regarding the same real vehicle in the same frame. Similarly, we denote the FP sets of the $m$th and $n$th models by $FP_m$ and $FP_n$. Then, $\text{IoU}_{FP} = \frac{FP_m \cap FP_n}{FP_m \cup FP_n}$, where $b_m \in FP_m$ and $b_n \in FP_n$ form an element of $FP_m \cap FP_n$ if $\text{IoU}_{BEV}(b_m, b_n) \geq 0.5$. When $\text{IoU}_{FN}$ and $\text{IoU}_{FP}$ reach one, the two models exhibit the same error pattern in detecting vehicles, which makes no contribution to the performance of the fusion. Thus, lower $\text{IoU}_{FN}$ and $\text{IoU}_{FP}$ values are desired.

**Table 1: Model diversity in making vehicle detection errors.**

| Between | $\overline{\text{IoU}_{FN}}$ | $\overline{\text{IoU}_{FP}}$ |
|---|---|---|
| Original model & 20 generated models | 0.375 | 0.688 |
| Original model & 20 retrained models | 0.364 | 0.675 |
| 20 generated models | 0.432 | 0.705 |
| 20 retrained models | 0.364 | 0.675 |

Table 1 presents the average $\text{IoU}_{\text{FN}}$ and $\text{IoU}_{\text{FP}}$ values regarding vehicle detection under various settings: between the original model and any of 20 generated models, between the original model and any of 20 models trained from scratch, between any two of the 20 generated models, and between any two of the 20 models trained from scratch. These values are calculated while processing 154 frames in the absence of attacks. Note that the training from scratch incorporates the independent data augmentation. From Table 1, the generated models yield error patterns different from the original model. Intuitively, this error pattern diversity is in favor of counteracting non-adaptive attacks on the original model, because FN and FP share some common nature with the effects of car hiding and car creating attacks, respectively. From Table 1, the models generated and trained from scratch achieve similar low $\text{IoU}_{FN}$ and $\text{IoU}_{FP}$ values (i.e., about 0.4 in FN sets and 0.7 in FP sets). This is in favor of counteracting adaptive attacks. An advantage of Hypernet generation over training from scratch is the generation's lower compute overhead. From our measurement, generating $M = 4$ models takes about 1.2 milliseconds on a GPU-equipped computer, while the training from scratch of only one model takes about 10 hours on the same computer.

## 4.3 Vehicle Detection Results Fusion and Calibration

*4.3.1 Fusion of Results.* As a generated 3D-OD model shares some layers with the original model, an adversarial example effective against the original model will maintain a reduced level of effectiveness against the generated model. However, the attack effect varies across the generated models. Considering car hiding as an example, some models may still miss the target vehicle, while others give bounding boxes with distinct offsets. The fusion aims to aggregate these noisy results to produce an attack-resistant and accurate result. As illustrated in Figure 4, the fusion function $\xi_f$ has the following three steps:

① **Result filtering and stacking**: As the current design of Hyper3Def focuses on the perception of vehicle objects, we only keep the detected vehicle objects in the generated models' detection results. If the attack causes misclassification, this filtering may miss a true vehicle or wrongly include a non-vehicle object as a vehicle. Such errors are to be rectified by the steps ② and ③ presented shortly. The filtered results (denoted by $\hat{\mathbf{y}}_m$, where $m = 1, \ldots, M$) are stacked into a single map: $\hat{\mathbf{y}} = \hat{\mathbf{y}}_1 \cup \hat{\mathbf{y}}_2 \cup \ldots \cup \hat{\mathbf{y}}_M$.

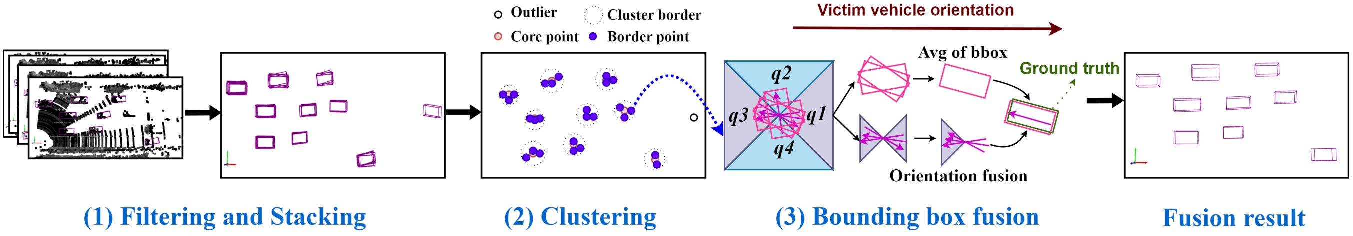


**Figure 4: Fusion of the ensemble's vehicle detection results.**

**Table 2: Clean precision of fusion & original model.**

| Average Precision (%) | Fused | Original model |
|---|---|---|
| Bounding Box (2D) | 89.95 | 90.52 |
| Angle of Arrival Score | 90.12 | 90.32 |
| Bounding Box (BEV) | 88.91 | 89.94 |
| Bounding Box (3D) | 73.31 | 83.37 |

② **Clustering**: We apply DBSCAN [14] to group the center positions of the bounding boxes in $\hat{\mathbf{y}}$. We set the `epsilon` parameter of DBSCAN to be 0.9 m, which is about half of the typical vehicle width, ensuring that two center positions separated by 0.9 m meters or more are not considered neighbors. As the maximum number of center positions for a detected vehicle is $M$, we set another parameter of DBSCAN to eliminate any cluster with less than $M \times$ `min_samples_rate` center positions. This helps remove the sparse ghost cars. §5 will evaluate the setting of `min_samples_rate`.

③ **Bounding box fusion**: We fuse the bounding boxes in each cluster by averaging their 3D coordinates and sizes. Fusing orientation is more challenging, because orientation estimations are susceptible to noises. We observe that most detected vehicles tend to have orientations that are either aligned with, opposite to, or perpendicular to the victim vehicle. Thus, based on the victim vehicle's orientation, we partition its surrounding plane into four quadrants q1, q2, q3, and q4, where q1 encompasses orientations deviating no more than 45° to the left or right from the victim vehicle's orientation. The q1 to q4, each spanning 90°, are arranged counter-clockwise. We employ a two-step majority voting to refine the orientation estimation. First, an initial vote identifies the dominant quadrant group, e.g., the q1 and q3 in Figure 4(3), based on the detected orientations of all bounding boxes. Next, a second majority vote within the selected group confirms the predominant direction. Any orientations contrary to this dominant direction are adjusted by 180°.

*4.3.2 Calibrating Fused Result.* The generated ensemble may be inferior in terms of 3D-OD accuracy compared with the original model optimized solely for 3D-OD. Table 2 shows the average precision of the fused result and the original result in the absence of attack. The average precision is based on four criteria widely employed for characterizing 3D-OD accuracy [15]. While the fusion performs comparably to the original model in the first three metrics, it underperforms in the last metric, which is the most stringent.

Given above, we propose a calibration algorithm shown in Algorithm 1 to refine the output of the ensemble using the original model's result. It makes a match between every detected vehicle bounding box in fusion result $\mathbf{y}$ and a vehicle bounding box in original model output $\mathbf{y}_0$. It replaces the 7-tuple box coordinates of a bounding box in $\mathbf{y}$ with the coordinates of its matched bounding box in $\mathbf{y}_0$ if they have a $\text{IoU}_{BEV}$ score higher than a threshold which is 0.2 in our implementation. Table 3 presents an example of the calibration effect under various attacks and ground truths. The calibration tends to yield the original model's result in the absence of attacks and the ensemble's result in the presence of attacks. Detailed evaluation will be presented in §5.2.

**Algorithm 1:** Ensemble output calibration (i.e., $\xi_c$)

```
Data: Original model output $\mathbf{y}_0$, ensemble's fused output $\mathbf{y}$
Result: Calibrated vehicle detection result
Configuration: Threshold $\eta$ to regard two bounding boxes
    belonging to the same vehicle
for each bounding box $b$ in $\mathbf{y}$ do
    if there exists a bounding box $b_0$ in $\mathbf{y}_0$ such that
     $\text{IoU\_BEV}(b, b_0) > \eta$ then
        use $b_0$ to replace $b$ in $\mathbf{y}$
return calibrated vehicle detection result $\mathbf{y}$
```

**Table 3: Example of the calibration effect. The calibration tends to select the results with "$*$" as the final outputs. (Null: no vehicle in a specific bounding box)**

| Output type | Attack type | | | |
|---|---|---|---|---|
| | No attack | | Hiding | Creating |
| Ground truth existence | Vehicle | null | Vehicle | null |
| Original model's result | $b_0$* | null* | null | $b_0$ |
| Ensemble's result | $b$ | null | $b$* | null* |
| Calibrated result | $\mathbf{b_0}$ | **null** | $\mathbf{b}$ | **null** |

## 4.4 Hyper3Def+ with Attack Detection

We design two attack detectors that are sensitive to the car hiding and car creating attacks, respectively. At run time, if either or both of them claim(s) detection of attack, the Hyper3Def inference described in §4.1 is activated; otherwise, only the original model is executed. In the rest of this paper, this improved version of Hyper3Def that integrates attack detection is called Hyper3Def+. Figure 5 illustrates the inference process of Hyper3Def+. The rest of this section presents the design of the attack detector sensitive to attack $A$, where $A$ represents car hiding or car creating. In Figure 5, we use subscripts $H$ and $C$ to denote the symbols related to car hiding and car creating, respectively.

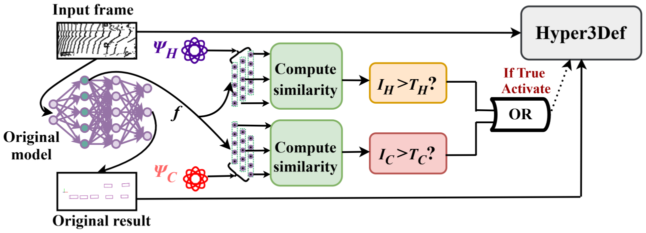


**Figure 5: Inference process of Hyper3Def+**

The attack detector takes an intermediate feature map of the original model as input, which is denoted by $f$. For instance, in our implementation, we use the feature map generated by a convolution layer of the original PointPillars model [23] as the $f$. To address attack $A$, where $A = H$ and $A = C$ refer to car hiding and car creating, respectively, a Hypernet denoted by $\Psi_A$ is used to generate $M_A$ layers denoted by $\{\delta^A_1, \delta^A_2, \ldots, \delta^A_{M_A}\}$. The $\Psi_A$ is different from the Hypernet $\Psi$ presented in §4.2. The Hypernet $\Psi$ uses multiple generators to generate multiple layers at once for a single model, whereas $\Psi_A$ uses the same generator to generate each of the $M_A$ layers with the same structure. Each of the $M_A$ layers processes the input to produce a new feature map $\delta^A_i(f)$, $i \in 1, \ldots, M_A$. We define the *attack indicator* denoted by $I_A(f)$ based on the average pair-wise cosine similarity between the new feature maps, i.e., $I_A(f) = 1 - \frac{2}{M_A(M_A-1)} \sum_{i=1}^{M_A-1} \sum_{j=i+1}^{M_A} S_{\cos}(\delta^A_i(f), \delta^A_j(f))$. If $I_A(f)$ is greater than a threshold $T_A$, the detector claims presence of attack. By using the Hypernet $\Psi_A$ here, the attack detector follows the dynamic defense strategy for hardened robustness against the adaptive attackers who aim to bypass the detection. We apply contrastive learning to train the Hypernet $\Psi_A$ such that $I_A(f)$ is high and low in the presence and absence of attack, respectively. Specifically, for a set of $N_A$ contrastive training data pairs $\{\langle f^n_c, f^n_A \rangle | n = 1, \ldots, N_A\}$, where $f^n_c$ and $f^n_A$ represent clean feature map and adversarial feature map containing attack $A$, the contrastive loss is $L_A = \frac{1}{N_A} \sum_{n=1}^{N_A} \left[ I_A(f^n_c) - \epsilon_A \cdot I_A(f^n_A) \right] + \gamma_A \cdot Loss^n_{sim}$, where $Loss^n_{sim}$ is the similarity loss computed based on the weights of the $M_A$ generated layers in the $n$th iteration, $\epsilon_A$ and $\gamma_A$ are two positive coefficients controlling the relative importances of the three loss components. Inclusion of the similarity loss is beneficial to dynamic defense, as discussed in §4.2.

Compared with the ensemble generated by Hyper3Def for 3D-OD, the $M_A$ layers used for attack detection is lightweight. Thus, with attack detection, the execution of the ensemble can be avoided mostly in the absence of attack, unless the attack detection produces a false positive. A concerning case is that the attacker induces a false negative detection. Our evaluation in next section considers such a strong attacker who constructs the perturbations to induce false negative detection and also mislead the original model.

# 5 Evaluation with Open Data

## 5.1 Settings and Methodology

**Dataset:** We use the KITTI 3D-OD dataset [15] to drive evaluation. The ego vehicle in the dataset is the victim vehicle. For the car-hiding attack, we randomly select 154 frames containing a vehicle approaching from the opposite direction as the target vehicle. For car creating, we randomly set a fake bounding box in the space of each frame to launch the attack.

**3D-OD model:** We choose PointPillars [23] as the original model, owing to its balanced performance and speed. We run PointPillars and the associated defenses, implemented in Python 3.8 with PyTorch 2.01. The default confidence threshold for a candidate bounding box to be yielded is 0.2.

**Attacks:** We launch the following three attacks. (1) The *object-based car hiding* (OCH) [56], a black-box attack, places objects to inject adversarial points around the target vehicle to hide it from detection. We use 3 adversarial cardboards with 0.5 m side length within a $1.6 \times 1.6 \times 0.6\,\mathrm{m}^3$ space above the target vehicle's geometric center. Larger cardboards do not further improve attack effectiveness much and become too obvious. (2) The *laser-based car hiding* (LCH) [21], a white-box attack, injects 200 points within a $1 \times 1 \times 1\,\mathrm{m}^3$ space above the target vehicle's geometric center to hide it. Note that infrared lasers are stealthy to human vision and injecting up to 200 points is achievable according to [21]. (3) The *laser-based car creating* (LCC) [21], a white-box attack, injects 100 points within a $4.4 \times 1.8 \times 1.7\,\mathrm{m}^3$ space to create a ghost car detected by the victim vehicle.

**Defenses:** For *Hyper3Def*, by default, we set ensemble size $M = 4$, `min_samples_rate` = 50%. For *Hyper3Def+*, we set the attack detection ensemble size $M_A = 2$ and the attack detection thresholds $T_H = 0.5$, $T_C = 0.8$, which strikes a balance between the false positive and false negative rates in attack detection. We employ six baselines. (1) The *Defenseless* applies no defense mechanisms. (2) The adversarial training on a specific attack, denoted by *AdvTrain-specific*, adversarially trains the 3D-OD model against a specific attack described earlier. Depending on the employed attack, it has three variants denoted by *AdvTrain-specific (OCH)*, *AdvTrain-specific (LCH)*, and *AdvTrain-specific (LCC)*. (3) The adversarial training on multiple attacks, denoted by *AdvTrain-multiple*, adversarially trains the 3D-OD model against all the three attacks described earlier. (4) The *SRS* [52] reviewed in §2.2 randomly samples a portion (50% in our implementation) of the points to form the input to the 3D-OD model. (5) The *Static Ensemble* uses $M$ models that are trained from scratch offline and remain fixed at run time. (6) *RL-Remove* [49] uses a reinforcement learning-based search to identify potential injected objects and remove the relevant points until the vehicle is correctly detected. The result calibration technique presented in §4.3.2 can be also integrated with the baseline defense approaches.

**Evaluation metrics:** We determine whether an attack is successful by checking the IoU value. A car hiding is successful if no detected bounding box has an $\mathrm{IoU}_X$ value greater than a threshold $\eta$ with the target vehicle's ground truth bounding box; a car creating is successful if at least one detected bounding box has an $\mathrm{IoU}_X$ value greater than $\eta$ with the aimed fake bounding box. The criterion $X$ can be one of those listed in Table 2. Based on the above, we use the following two metrics to characterize the attack and system performances. (1) *Attack success rate (ASR)* is the ratio of frames where the attack is successful to the total number of frames with attack on, where we adopt the $\mathrm{IoU}_{BEV}$ criterion with $\eta = 0.3$. In the absence of defense, attack success rate suggests the effectiveness of the attack. (2) *Average Precision (AP)* is the area under the precision-recall curve regarding vehicle detection, where different points on the curve are associated with different confidence

**Table 4: Vehicle detection performance in the absence of attacks.**

| Defense | Attack-free AP (%) | |
|---|---|---|
| | w/o calibration | w/ calibration |
| Defenseless | 83.37 | n/a |
| SRS [52] | 75.15 | 82.95 |
| AdvTrain-specific (OCH) | 81.78 | 83.19 |
| AdvTrain-specific (LCH) | 77.45 | 82.97 |
| AdvTrain-specific (LCC) | 76.20 | 81.16 |
| AdvTrain-multiple | 76.10 | 82.95 |
| Static Ensemble ($M$=4) | 87.38 | n/a |
| RL-Remove [49] | 83.98 | n/a |
| Hyper3Def | 73.31 | 83.01 |
| Hyper3Def+ | 77.31 | 83.18 |

OCH: Object-based Car Hiding LCH: Laser-based Car Hiding
LCC: Laser-based Car Creating SRS: Simple Random Sampling

thresholds used by the 3D-OD model. A higher average precision indicates better 3D-OD performance. The precision and recall are based on the $\mathrm{IoU}_{3D}$ criterion with $\eta = 0.7$, which has been widely adopted in 3D-OD benchmarks.

## 5.2 Performance in the Absence of Attacks

Table 4 shows the performance of the vehicle detection with a certain defense in the absence of attacks. Without result calibration, most defense approaches lead to decreases in average precision except *Static Ensemble* and *RL-Remove.* Note that each member model of the *Static Ensemble* achieves similar average precision of *Defenseless.* With our proposed results fusion method, the *Static Ensemble* achieves even higher average precision of 87.38%. *RL-Remove* only processes the point cloud clusters detected malicious. As a result, its overall accuracy on attack-free data remains largely unaffected compared with the defenseless system. Therefore, using the original model's result to calibrate *Static Ensemble* or *RL-Remove* brings no additional benefit. We do not apply the result calibration to either method. The Hypernet-generated 3D-OD models are inferior compared with the original model in terms of average precision. Hyper3Def+ achieves higher average precision than Hyper3Def, as it usually uses the original model for clean inputs unless the attack detector is wrongly triggered. In other words, the average precision of Hyper3Def+ without calibration is a mix of the higher attack-free average precision of the original model and the lower attack-free average precision of the wrongly triggered Hyper3Def.

## 5.3 Defense Performance of Hyper3Def

For Hyper3Def, we consider three cases regarding the adaptive attacker's knowledge about the system, as illustrated by the left part of Table 5. **Case-①**: The attacker obtains the original model $\Theta_0$ and constructs the attack against it. **Case-②**: The attacker obtains a 3D-OD model $\Theta_1$ (i.e., a member of the ensemble) generated by Hyper3Def in the past and then constructs the attack against it. **Case-③**: The attacker obtains the $\Theta_0$ and the Hypernet $\Psi$. By following Hyper3Def's workflow, the attacker uses $\Psi$ to generate an ensemble of $M'$ models. This $M'$ may be different from the $M$ used by Hyper3Def. The attacker optimizes the attacks against the $M'$-model ensemble. Specifically, for white-box attacks (LCC and LCH), the attacker tries to compromise the $M'$-model ensemble by averaging the gradients computed by the attack. For the black-box attack OCH, the attacker uses the majority strategy to search for the optimal positions that induce most models in a generated ensemble to make mistakes. Compared with the perturbations constructed against the original model, the perturbations constructed against the $M'$-model ensemble are more transferable across the models generated by Hyper3Def.

**Table 5: ASR when the adaptive attacker obtains different sets of knowledge about Hyper3Def.**

| Case | Knowledge | | | ASR (%) | | |
|---|---|---|---|---|---|---|
| | $\Theta_0$ | $\Theta_1$ | $\Psi$ | OCH | LCH | LCC |
| ① | ✓ | | | 5.19 | 3.90 | 4.24 |
| ② | | ✓ | | 9.24 | 4.75 | 17.25 |
| ③ | ✓ | | ✓ | **14.29** | **6.69** | **24.19** |

$\Theta_0$: Original model; $\Theta_1$: One generated model; $\Psi$: Hypernet

**Table 6: ASR of Hyper3Def+ when attacker obtains different sets of info about Hyper3Def+.**

| Case | Knowledge | | | | | ASR (%) | | |
|---|---|---|---|---|---|---|---|---|
| | $\Theta_0$ | $\Theta_1$ | $\Psi$ | $\{\delta_1^A, .., \delta_{M_A}^A\}$ | $\Psi_A$ | OCH | LCH | LCC |
| ① | ✓ | | | | | 5.84 | 3.25 | 3.25 |
| ② | | ✓ | | | | 9.08 | 4.55 | 16.85 |
| ③ | ✓ | | ✓ | | | **11.69** | 5.19 | 24.32 |
| ④ | ✓ | | | ✓ | | 3.60 | 16.25 | 4.42 |
| ⑤ | ✓ | | | | ✓ | 3.04 | **21.43** | 5.84 |
| ⑥ | ✓ | | ✓ | | ✓ | 5.19 | 10.39 | **26.18** |

$\Theta_0$: Original model; $\Theta_1$: One generated model; $\Psi$: Hypernet
$\Psi_A$, $\{\delta_1^A, \ldots, \delta_{M_A}^A\}$: Hypernet and generated layers for attack detection

For Hyper3Def+, we consider the following extra cases regarding the adaptive attacker's knowledge about the attack detector, which are summarized in the left part of Table 6. **Case-④**: The attacker obtains a batch of generated layers used for attack detection. **Case-⑤**: The attacker obtains the Hypernet used to generate the attack detection layers. **Case-⑥**: The attacker has full knowledge about the static information of Hyper3Def (i.e., $\Theta_0$ and $\Psi$) and the Hypernet for attack detection (i.e., $\Psi_A$). In these extra cases, the attacker constructs perturbations that can simultaneously bypass the attack detection and mislead vehicle detection.

For the defenseless system, the OCH, LCH, and LCC attacks achieve attack success rate of 64.9%, 69.5%, and 65.6%, respectively. Table 5 shows attack success rates of OCH, LCH, and LCC under the three cases of the adaptive attacker's knowledge about Hyper3Def. Compared with the attack optimized against a single model in Case-① or Case-②, the attack that obtains the Hypernet in Case-③ achieves higher attack success rate. Nevertheless, Hyper3Def reduces attack success rate compared with the defenseless system.

Table 6 shows the results when Hyper3Def+ is deployed. For the car hiding attacks (OCH and LCH), the joint attacks on the attack detection and Hyper3Def are generally more threatening than those merely on Hyper3Def. The most threatening OCH and LCH, which achieve attack success rates of 11.69% (Case-③) and

**Table 7: ASR (%) of attacks under different defenses.**

| **Defense** | **Attack** | | | |
|---|---|---|---|---|
| | OCH | LCH | LCC | Worst |
| Defenseless | 64.90 | 69.48 | 65.58 | **69.48** |
| SRS [52] | 59.48 | 49.43 | 20.25 | **59.48** |
| AdvTrain-specific (OCH) | 7.14 | 27.27 | 71.43 | 71.43 |
| AdvTrain-specific (LCH) | 22.08 | 11.69 | 61.04 | 61.04 |
| AdvTrain-specific (LCC) | 88.50 | 77.20 | 5.60 | 88.50 |
| AdvTrain-multiple | 38.31 | 58.10 | 51.95 | **58.10** |
| Static Ensemble ($M = 4$) | 55.84 | 45.40 | 46.53 | **55.84** |
| RL-Remove [49] | 3.15 | 1.19 | 65.58 | **65.58** |
| Hyper3Def | 14.29 | 6.69 | 24.19 | **24.19** |
| Hyper3Def+ | 11.69 | 21.43 | 26.18 | **26.18** |

**Bold number**: Nash equilibrium of defense versus attacks.

21.43% (Case-⑤), are based on the original model $\Theta_0$. When the attacker additionally obtains $\Psi$ (Case-⑥), the attack success rates drop. This is because that, the joint car hiding attack optimization that tries to deal with two kinds of system dynamics from vehicle detection and attack detection is too sophisticated. Differently, the most threatening LCC uses all static information of Hyper3Def and attack detection (Case-⑥). This is because, different from the car hiding attack which needs to carefully plan the adversarial perturbations according to the existing point clouds, the car creating attack only needs to inject points to form a car contour. This makes the joint optimization against the two dynamics tractable.

In the rest of this paper, we use the *highest* attack success rates from Table 5 for Hyper3Def and Table 6 for Hyper3Def+ to characterize their performance against each attack. This is conservative from the defense perspective, which ensures no unfairness to other compared defense approaches.

## 5.4 Comparison with Other Defenses

Table 7 shows the attack success rates achieved by the OCH, LCH, and LCC attacks when different defenses are applied. Although *SRS* reduces the effectiveness of all attacks, car hiding attacks' attack success rates of around 50% to 60% are concerning. If *SRS* is more aggressive by dropping more points, its defense effectiveness will increase but further worsen its low clean accuracy.

*AdvTrain-specific* demonstrates good defense effectiveness only against the attack it is adversarially trained for, but limited effectiveness against other attacks. For instance, when adversarially trained for OCH, it reduces the attack success rate of OCH from 64.90% to 7.14%. However, it has comparatively reduced effectiveness against LCH and concerning effectiveness against LCC. This is because, when the 3D-OD model fits car hiding after adversarial training, it less captures car creating on the opposite. Similar observations can be found for *AdvTrain-specific (LCH)* and *AdvTrain-specific (LCC)*.

The above results suggest a key limitation of *AdvTrain-specific*, i.e., its defense performance is tied to the specific attack used during the adversarial training. When the actual attack deviates from the assumed attack, the defense performance may drop drastically. We conduct an extra experiment to show this limitation. Specifically, we control how much *AdvTrain-specific (LCC)* learns about the LCC attack by adjusting the learning rate. Then, with the enhanced 3D-OD models, we measure the attack success rate of OCH and LCH. Figure 6 shows the results. It also shows the average precision without fused result calibration in the absence of attack. The variation in LCC's attack success rate is due to the variation of learning rate in the adversarial training. The declines of the AP-ASR curves suggest that when the *AdvTrain-specific* overfits to an attack more, it captures other attacks less. The increase of the AP-ASR curve is consistent with the result in §5.2.

*AdvTrain-multiple* attempts to address the above problem by including all the three attacks in the adversarial training. Specifically, we collect equal amounts of adversarial training samples from the three attacks, resulting in 3x total adversarial training samples compared with that used by *AdvTrain-specific*. Because car hiding and car creating pursue different adversarial goals, the adversarial training against the two types of attacks alternate. For the training against car hiding, OCH and LCH samples are mixed. Table 7 includes the attack success rate of the three attacks on the system enhanced by *AdvTrain-multiple*, which are mild. Although *AdvTrain-multiple* is not particularly vulnerable to a certain attack, its defense effectiveness to each attack is also not impressive. In particular, *AdvTrain-multiple* exhibits lower defense performance against an attack compared with the *AdvTrain-specific* trained against the same attack. This is consistent with the result in [38], which considers image classification and views perturbations constructed based on different norms as different attacks. Our scenario of addressing attacks with opposite goals is more challenging than addressing common-goal attacks based on different norms.

Since the attacker has three options (OCH, LCH, LCC) and the adversarial training defender has four adversarial training options, we analyze the Nash equilibrium between them. After the defender adopts one of the four options, the adaptive attacker can obtain the enhanced model, test it with the three attack options, and choose that with maximum attack success rate. We include the highest attack success rate among the three attack options for each defense option in the last column of Table 7. The rational defender should choose *AdvTrain-multiple* with the minimum of the four maximum attack success rates (i.e., 58.10%), which is the Nash equilibrium. However, this equilibrium attack success rate is not significantly better than that of *SRS*.

*Static Ensemble* does not achieve good defense performance, because the adaptive attacker can still craft effective perturbations after obtaining the static ensemble. Similar to adversarial training, *RL-Remove* demonstrates strong defense effectiveness only against the specific attack it is designed to counter, i.e., hiding attacks, but offers no resistance against creating attacks.

For comparison, Table 7 also includes the results of Hyper3Def and Hyper3Def+ presented in Table 5 and Table 6. When Hyper3Def or Hyper3Def+ is deployed, the adaptive attacker should choose the most effective attack, which is LCC, achieving attack success rate of 24.19% or 26.18%, respectively. Compared with the attack success rates of 59.48% under *SRS*, 58.10% under *AdvTrain*, and 65.58% of under *RL-Remove*, Hyper3Def and Hyper3Def+ achieve the best defense performance. Results in this section are obtained based on the default cardboard size and laser injection points amount stated in §5.1. Hyper3Def and Hyper3Def+ remain the best under a wide range of settings. If we compare Hyper3Def and Hyper3Def+, the defense performance of Hyper3Def+ is lower under some attacks.

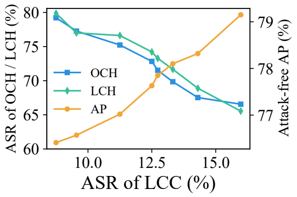


Figure 6: Overfit effect of AdvTrain-specific.

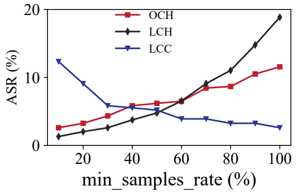


Figure 7: Impact of `min_samples_rate`.

This is because Hyper3Def+ has a larger attack surface due to the introduction of attack detector. However, the gap between the worst attack success rates under Hyper3Def and Hyper3Def+ is only 1.99%, much smaller than the gaps from the baselines. Note that Hyper3Def+ brings substantial computational overhead reduction compared with Hyper3Def, as shown shortly.

## 5.5 Computational Overhead

We measure the average per-frame inference times of all compared defense approaches on two computing platforms: (1) a workstation equipped with a 5.20GHz CPU (Ryzen 7900X) and a 24GB GPU (NVIDIA RTX 4090), (2) an NVIDIA Jetson AGX Orin 64GB (referred to as "Orin" hereafter), which operates at a power setting of 50 Watts. Note that we implement result calibration (cf. §4.3.2) for all defense approaches to ensure good and comparable attack-free average precision (cf. Table 4). Table 8 shows the results. As Hyper3Def+ and *RL-Remove* have two phases, i.e., attack detection and the subsequent reactive defense, their per-frame inference times are bi-polar depending on the existence of the attack.

First, we analyze the results obtained on the workstation. The defenseless system's inference time, 10.52 ms, can be used as the baseline to understand the additional overhead introduced by the defense. *SRS* and *AdvTrain* with result calibration introduce extra inference times of 8.83 ms and 3.92 ms, respectively. However, their defense performances are unsatisfactory. The *Static Ensemble* and Hyper3Def introduce similar extra inference times of about 18 ms and 20 ms. In the absence of attack, Hyper3Def+ introduces an extra inference time of 6.15 ms, similar to that of *AdvTrain*. This extra overhead is caused by the attack detection and the subsequent defense triggered by false positives. As *RL-Remove* uses a heavy attack detector, it incurs long inference times of 0.4 to 2.6 seconds. Overall, in the absence of attack, Hyper3Def+ introduces similar compute overhead as *SRS* and *AdvTrain*. In the presence of attack, Hyper3Def+ exhibits a doubled compute overhead. As return, Hyper3Def+ reduces attack success rate by more than two folds as shown in Table 7. The *RL-Remove* incurs tens of times compute overhead compared with other defense approaches.

On a single Orin, the per-frame inference time of each system increases by multiple folds, compared with that on the workstation. However, the latency of our proposed Hyper3Def and Hyper3Def+ still have good potential to meet the real-time requirement of carborne object detection that is related to the data sampling interval. For instance, in the absence of attacks, the projected processing throughput of Hyper3Def+ on two units of Orin (a typical configuration of mainstream commercial vehicles with L2+ autonomy [3, 6]) is 15.2 frames per second (FPS). This is higher than the typical LiDAR sampling rate of 10 FPS. In real adoption, the specification of the computation hardware should be chosen to ensure that Hyper3Def+ and other concurrent compute tasks fully meet their respective real-time requirements.

Table 8: Average per-frame inference time (ms).

| Defense | | Workstation | Orin |
|---|---|---|---|
| Defenseless | | 10.52 | 80.00 |
| SRS [52] | | 19.35 | 151.25 |
| AdvTrain | | 14.44 | 126.32 |
| Static Ensemble ($M = 4$) | | 28.57 | 231.33 |
| RL-Remove [49] | No attack | 405.23 | 912.53 |
| | Attack | 2643.12 | 8131.89 |
| Hyper3Def ($M = 4$) | | 31.27 | 247.95 |
| Hyper3Def+<br>$M = 4$, $M_A = 2$ | No attack | 16.69 | 132.28 |
| | Attack | 33.45 | 264.50 |

## 5.6 Sensitivity Analysis for Hyper3Def

Hyper3Def has two key parameters, i.e., `min_samples_rate` used by DBSCAN and the selection of dynamic layers. We conduct sensitivity studies on them.

**`Min_samples_rate`.** We set $M = 10$ and evaluate the impact of `min_samples_rate` on defense performance. As shown in Figure 7, the attack success rate of car hiding and creating attacks exhibit opposite trends as `min_samples_rate` changes, due to the opposite nature of the two attacks. To balance defense performance without prior knowledge of attack likelihoods, `min_samples_rate` can be set to a value between 40% and 60%. If there is such prior knowledge, the `min_samples_rate` can be set according to the overall risk. We also find that when `Min_samples_rate` is fixed, the increase in $M$ provides almost no improvement in defense performance.

**Selection of dynamic layers.** We adopt *defense success rate* (DSR) as the metric of defense performance. It is the ratio of the frames where the attack turns from successful to unsuccessful because of the defense, to the total number of frames where the attack is successful before the defense is applied. For each dynamic layers selection, we employ all three attacks and report the average attack success rate and defense success rate. The original model PointPillars includes an *encoder*, a convolutional feature extraction module, and a *head* output module. The average attack success rate and defense success rate when various combinations of layer(s) are selected for Hypernet generation are reported in Table 9. When the encoder or head is selected, low defense success rates (13.2% and 7.7%) are achieved, because they only contain around 1% and 6% neurons of the entire model and updating them leads to limited dynamism. Then, we examine the layers in the feature extraction module, consisting of *Group-1*, *Group-2*, and *Group-3*, along the direction of inference. Selecting three layers in a group gives higher defense success rate than doing so in a subsequent group. Selecting one layer per group yields a defense success rate of 41.18%, whiles selecting two increases it to 93.33%. Selecting three or more layers in each group makes the convergence of the Hypernet training challenging, as three layers in all three groups involve more than

**Table 9: Defense performance vs. dynamic layers.**

| Dynamic layer(s) | Avg ASR (%) | Avg DSR (%) |
|---|---|---|
| None | 66.65 | - |
| Encoder | 57.84 | 13.22 |
| Head | 61.53 | 7.68 |
| 3 layers of Group-1 | 40.83 | 38.73 |
| 3 layers of Group-2 | 47.51 | 28.72 |
| 3 layers of Group-3 | 55.65 | 16.50 |
| 1 layer per group | 39.20 | 41.18 |
| 2 layers per group | 4.44 | 93.33 |

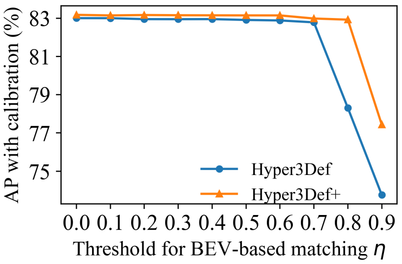


**Figure 8: AP of calibrated result vs. threshold for BEV-based bounding box matching.**

half of the entire model's parameters. A general observation is that, selecting more preceding feature extraction layers is beneficial to defense performance. As selecting two layers per group gives the highest defense performance, Hyper3Def adopts this setting.

**Threshold of BEV matching for result calibration.** From the result in Table 2, both the ensemble's prediction and the original model's prediction have high average precisions in terms of the BEV criterion. Therefore, as shown in Figure 8, when the threshold $\eta$ used in Algorithm 1 for matching two bounding boxes in terms of BEV varies in a wide range from 0 to 0.6, the average precision of the calibrated result is nearly unaffected.

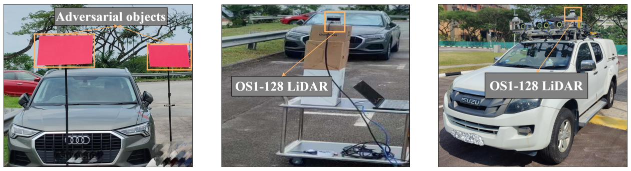
## 6 Real-World Experiments

We conduct experiments on a real-world testbed with OCH attack. In addition, we participate in an organized challenge activity, in which a third-party vehicle is engaged as a data-recording victim and the LiDAR hardware model is the only information made available to us before the experiment.

### 6.1 Experiments Setup

**Attack setup.** We use a crossover sport utility vehicle (SUV) as OCH's target vehicle. We employ two attack devices, each with a cardboard (0.64 m × 0.49 m) mounted on a tripod. We follow [56] to search cardboard positions for effective OCH. Different initial conditions of the search can yield different cardboard positions. Figure 9(a) shows the deployed OCH.

**Victim LiDAR.** We employ an OS1-128 unit which is a car-borne 128-channel LiDAR. We use a trolley cart to carry it and raise it to a typical vehicle altitude via boxes, as shown in Figure 9(a), to mimic the victim vehicle.



(a) Target vehicle (b) Our victim LiDAR (c) 3rd-party victim

**Figure 9: Real-world experiment setup.**

**Third-party victim vehicle.** It is a SUV with an OS1-128 mounted on the top. Figure 9(c) shows a picture of the setup. The vehicle was driven by a human driver during the experiment. We have no other information about this vehicle, but have access to its LiDAR data after the experiments.

### 6.2 Experiments with Our Victim LiDAR

First, we evaluate the attack and defense methods when our victim LiDAR is positioned at a fixed point. We achieve 7 successful attack trials with varying cardboard positions against the original model. Then, we apply *SRS* and Hyper3Def+. *SRS* achieves a quite low defense success rate of 14.28%, which is similar to the evaluation results obtained on open data. With a high-resolution victim LiDAR, *SRS* is less effective as the randomly down-sampled adversarial point clouds remain dense enough to spoof the original 3D-OD model. Hyper3Def+ successfully detects all the effective attacks, and achieves a 71.43% defense success rate in recovering the system from effective attacks. This defense success rate is lower than the 85.7% defense success rate achieved in §5, likely due to different environments and attack conditions.

Second, we evaluate the attack and defense when the victim LiDAR is at different positions, as depicted in Figure 10. The attack is optimized for the case where the victim LiDAR is 10.37 m from the center of the target vehicle. When the victim LiDAR moves to 11 positions, represented by the blue dots in Figure 10, the attack remains effective. Hyper3Def+ detects all attack attempts at 14 positions, including 3 ineffective. Among the 11 effective attacks, our defense restores the system against 8 attacks, achieving a defense success rate of 72.73%.

### 6.3 Challenge Activity with 3rd-Party Victim

We deploy the OCH attack designed in §6.2. Note that the optimal positions for the adversarial objects depend on the victim LiDAR's altitude and pitch angle. However, these parameters of the third-party victim vehicle are unknown and may be different from the settings we use to optimize the attack. The third party drives the vehicle toward the target vehicle. The attack is effective when the victim vehicle is at 6 out of 17 positions and thus relatively less effective compared with that in §6.2. This is probably due to the mismatch of LiDAR altitude and pitch angle. Note that the third-party victim vehicle is customized. In reality, the specifications and operational parameters of the off-the-shelf vehicles can be obtained and even public. When attackers launch attacks, they may optimize the attacks based on these data. Nevertheless, in our experiment, Hyper3Def+ defeats the attack when the victim vehicle is at all the

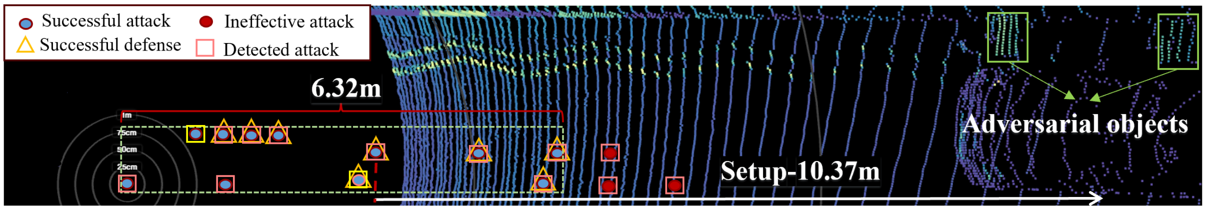


**Figure 10: Results of real-world experiments with our victim LiDAR.**

6 positions (i.e., 100% defense success rate, empirically) where the attack is successful against the defenseless system.

## 7 Limitations

This section discusses several limitations of this study.

**Defense for multi-modal fusion.** Multi-modal fusion has received increasing attention. However, LiDAR modality is still a primary sensing channel for many vehicle products due to its precise 3D sensing capability. Certain open-source (e.g., Autoware.Universe [2]) and commercial (e.g., VueOne [4]) platforms default to LiDAR-only perception pipelines, rendering this study essential. The multi-modal perception may mitigate the effectiveness of the single-modality attacks. However, recent studies [8, 55] have shown that attacks jointly designed against multiple modalities can also spoof the fusion system. The defense for each modality can be a basis for forming the joint defense for the multi-modal fusion system. Depending on the fusion strategy, our proposed defense can still be used with different degrees of redesign. For instance, if a late fusion strategy (i.e., decision fusion) [55] is employed for a fusion system, the defense described in this paper can be applied directly to the LiDAR-based detector and a new but similar defense for camera-based object detector is needed. If an early or mid-stage fusion strategy is employed, a new dynamic defense design is needed. These required new defenses are meaningful future work items.

**Defense for other target objects.** In this paper, we focus on vehicles as the target objects for implementing attacks, given their prevalence on roads and the severe safety hazards associated with their incorrect detection. However, other objects including pedestrians, bicycles [21], and traffic cones [8] can also be targeted in attacks against LiDAR 3D-OD. Hyper3Def can be adapted to counteract such attacks on these other target objects by adjusting the result filtering and tuning the `epsilon` parameter in the result clustering. We have implemented these changes and conducted a set of preliminary experiments to understand the performance of Hyper3Def for two other object classes, i.e., pedestrians and cyclists. The results are shown in Table 10. Hyper3Def reduces attack success rate by 2.4X to 3X for cyclists and by 1.3X to 1.7X for pedestrians. Compared with the 2.7X to 8X attack success rate reduction for the vehicle objects as shown in Table 7, the defense performance reduces with the object volume. This is probably correlated with the lower clean accuracy of the original model and the Hypernet-generated ensembles in detecting smaller objects. A pathway for mitigating the defense performance reduction is to introduce more augmented data of small objects for Hypernet training.

**Table 10: ASRs of LCH and LCC attacks on small objects under no defense or Hyper3Def defense.**

| Defense | Cyclist | | Pedestrian | |
|---|---|---|---|---|
| | LCH | LCC | LCH | LCC |
| Defenseless | 50.0 | 60.0 | 40.0 | 85.0 |
| Hyper3Def | 16.6 | 25.0 | 30.0 | 50.0 |

**Model-agnostic attacks.** If the attacker has the ability to inject a pre-recorded point cloud about an object into the victim LiDAR sensor's data output [21], the attack is agnostic to the 3D-OD model and may render Hyper3Def ineffective. However, this strong attack imposes high logistics requirements. To deal with this attack, we can check whether the object's point clouds across multiple consecutive frames conform to the perspective changes of the LiDAR when the ego vehicle moves. This prior knowledge-based defense further complicates the logistics of the successful attack.

## 8 Conclusion

This paper proposes Hyper3Def, a dynamic defense against physical adversarial attacks on car-borne LiDAR-based vehicle detection. Hyper3Def updates the protected target, i.e., the object detection neural network model, by generating part of its weights at run time. This defensive update addresses the adaptive attackers who try to optimize attacks based on the information about the system. The advanced version of Hyper3Def further integrates an attack detector to avoid unnecessary defense overhead in clean scenes. Evaluation shows that Hyper3Def reduces the success rates of attacks by more than two folds compared with heuristic defense and adversarial training, and maintains clean accuracy as the original system.

## Acknowledgments

The work by Xu, Guo, and Tan is supported by the National Research Foundation, Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-GC-2023-006). The work by Song is supported by the Ministry of Education, Singapore, under its MOE AcRF Tier 1, and funded through the SUTD Kickstarter Initiative (SKI) administered by SUTD (SKI 2021_06_11). The work by Lou and Wang is supported in part by a project from Hong Kong Research Grant Council under GRF 11210622. The work by Qiao is supported in part by NSF CNS 2413876. However, any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors, and do not necessarily reflect the views of the sponsors such as NSF.

# References

[1] 2025. *Apollo Auto: perception-lidar-detection.* https://github.com/ApolloAuto/apollo/tree/master/modules/perception/lidar_detection.

[2] 2025. Autoware Universe. https://github.com/autowarefoundation/autoware.universe.

[3] 2025. Lixiang L9. https://www.lixiang.com/configuration.html?carModel=L9.

[4] 2025. VueOne. https://www.vueron.com/vueone/.

[5] 2025. Waymo Driver. https://waymo.com/waymo-driver/.

[6] 2025. *Xiaomi Unveils Five Core Automotive Technologies and Debuts Xiaomi SU7, Completing the Human x Car x Home Smart Ecosystem.* https://www.mi.com/global/discover/article?id=3095

[7] Mazen Abdelfattah, Kaiwen Yuan, Z Jane Wang, and Rabab Ward. 2021. Towards universal physical attacks on cascaded camera-lidar 3d object detection models. In *IEEE International Conference on Image Processing (ICIP).*

[8] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. 2021. 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 (SP).*

[9] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. 2019. Adversarial sensor attack on lidar-based perception in autonomous driving. In *ACM SIGSAC Conference on Computer and Communications Security (CCS).*

[10] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP).* Ieee, 39–57.

[11] Jin-Hee Cho, Dilli P Sharma, Hooman Alavizadeh, Seunghyun Yoon, Noam Ben-Asher, Terrence J Moore, Dong Seong Kim, Hyuk Lim, and Frederica F Nelson. 2020. Toward proactive, adaptive defense: A survey on moving target defense. *IEEE Communications Surveys & Tutorials* 22, 1 (2020), 709–745.

[12] Minkyoung Cho, Yulong Cao, Zixiang Zhou, and Z.Morley Mao. 2023. ADoPT: LiDAR Spoofing Attack Detection Based on Point-Level Temporal Consistency. In *British Machine Vision Conference (BMVC).*

[13] Andrew Du, Bo Chen, Tat-Jun Chin, Yee Wei Law, Michele Sasdelli, Ramesh Rajasegaran, and Dillon Campbell. 2022. Physical adversarial attacks on an aerial imagery object detector. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV).* 1796–1806.

[14] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, Vol. 96. 226–231.

[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. 2015. The kitti vision benchmark suite. *URL http://www. cvlibs. net/datasets/kitti* 2, 5 (2015), 1–13.

[16] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR).*

[17] David Ha, Andrew M. Dai, and Quoc V. Le. 2017. HyperNetworks. In *International Conference on Learning Representations (ICLR).*

[18] Zhongyuan Hau, Kenneth T Co, Soteris Demetriou, and Emil C Lupu. 2021. Object removal attacks on lidar-based 3d object detectors. In *Automotive and Autonomous Vehicle Security Workshop (AutoSec).*

[19] Zhongyuan Hau, Soteris Demetriou, and Emil C Lupu. 2022. Using 3d shadows to detect object hiding attacks on autonomous vehicle perception. In *2022 IEEE Security and Privacy Workshops (SPW).* IEEE, 229–235.

[20] Zhongyuan Hau, Soteris Demetriou, Luis Muñoz-González, and Emil C Lupu. 2021. Shadow-catcher: Looking into shadows to detect ghost objects in autonomous vehicle 3d sensing. In *European Symposium on Research in Computer Security (ESORICS).*

[21] Zizhi Jin, Xiaoyu Ji, Yushi Cheng, Bo Yang, Chen Yan, and Wenyuan Xu. 2023. Pla-lidar: Physical laser attacks against lidar-based 3d object detection in autonomous vehicle. In *IEEE Symposium on Security and Privacy (SP).*

[22] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. 2018. Adversarial examples in the physical world. In *Artificial intelligence safety and security.* Chapman and Hall/CRC, Chapter 8, 99–112. https://doi.org/10.1201/9781351251389

[23] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[24] Linyi Li, Tao Xie, and Bo Li. 2023. Sok: Certified robustness for deep neural networks. In *IEEE symposium on security and privacy (SP).*

[25] Daizong Liu and Wei Hu. 2022. Imperceptible transfer attack and defense on 3d point cloud classification. *IEEE transactions on pattern analysis and machine intelligence* 45, 4 (2022), 4727–4746.

[26] Daniel Liu, Ronald Yu, and Hao Su. 2019. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. In *IEEE International Conference on Image Processing (ICIP).*

[27] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. 2018. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299* (2018).

[28] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR).*

[29] Yanmao Man, Raymond Muller, Ming Li, Z Berkay Celik, and Ryan Gerdes. 2023. That person moves like a car: Misclassification attack detection for autonomous systems using spatiotemporal consistency. In *USENIX Security Symposium.*

[30] Rui Qian, Xin Lai, and Xirong Li. 2022. 3D object detection for autonomous driving: A survey. *Pattern Recognition* 130 (2022), 108796.

[31] Neale Ratzlaff and Fuxin Li. 2019. HyperGAN: A Generative Model for Diverse, Performant Neural Networks. In *International Conference on Machine Learning (ICML).*

[32] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. 2008. Towards 3D Point cloud based object maps for household environments. *Robotics and Autonomous Systems* (Nov 2008), 927–941.

[33] Hocheol Shin, Dohyun Kim, Yujin Kwon, and Yongdae Kim. 2017. Illusion and dazzle: Adversarial optical channel exploits against lidars for automotive applications. In *Cryptographic Hardware and Embedded Systems (CHES).*

[34] Dawn Song, Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Florian Tramer, Atul Prakash, and Tadayoshi Kohno. 2018. Physical adversarial examples for object detectors. In *12th USENIX workshop on offensive technologies (WOOT 18).*

[35] Qun Song, Zhenyu Yan, Wenjie Luo, and Rui Tan. 2022. Sardino: Ultra-Fast Dynamic Ensemble for Secure Visual Sensing at Mobile Edge. In *International Conference on Embedded Wireless Systems and Networks (EWSN).*

[36] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z Morley Mao. 2020. Towards robust LiDAR-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *USENIX Security Symposium.*

[37] Jiachen Sun, Jiongxiao Wang, Weili Nie, Zhiding Yu, Zhuoqing Mao, and Chaowei Xiao. 2023. A critical revisit of adversarial robustness in 3D point cloud recognition with diffusion-driven purification. In *International Conference on Machine Learning.* PMLR, 33100–33114.

[38] Florian Tramer and Dan Boneh. 2019. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems* 32 (2019).

[39] James Tu, Huichen Li, Xinchen Yan, Mengye Ren, Yun Chen, Ming Liang, Eilyan Bitar, Ersin Yumer, and Raquel Urtasun. 2022. Exploring adversarial robustness of multi-sensor perception systems in self driving. In *Conference on Robot Learning (CoRL).*

[40] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. 2020. Physically realizable adversarial examples for lidar object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[41] Kai Wang, Zhaopan Xu, Yukun Zhou, Zelin Zang, Trevor Darrell, Zhuang Liu, and Yang You. 2024. Neural network diffusion. *arXiv preprint arXiv:2402.13144* (2024).

[42] Xingxing Wei, Siyuan Liang, Ning Chen, and Xiaochun Cao. 2018. Transferable adversarial attacks for image and video object detection. *arXiv preprint arXiv:1811.12641* (2018).

[43] Chong Xiang, Charles R Qi, and Bo Li. 2019. Generating 3d adversarial point clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[44] Qifan Xiao, Xudong Pan, Yifan Lu, Mi Zhang, Jiarun Dai, and Min Yang. 2023. Exorcising "Wraith": Protecting LiDAR-based Object Detector in Automated Driving System from Appearing Attacks. In *USENIX Security Symposium.*

[45] Kaichen Yang, Tzungyu Tsai, Honggang Yu, Max Panoff, Tsung-Yi Ho, and Yier Jin. 2021. Robust roadside physical adversarial attack against deep learning in lidar perception modules. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS).*

[46] Chengzeng You, Zhongyuan Hau, and Soteris Demetriou. 2021. Temporal consistency checks to detect LiDAR spoofing attacks on autonomous vehicle perception. In *Workshop on Security and Privacy for Mobile AI (MAISP).*

[47] Jinlai Zhang, Lyujie Chen, Binbin Liu, Bo Ouyang, Qizhi Xie, Jihong Zhu, Weiming Li, and Yanmei Meng. 2023. 3d adversarial attacks beyond point cloud. *Information Sciences* 633 (2023), 491–503.

[48] Kui Zhang, Hang Zhou, Jie Zhang, Qidong Huang, Weiming Zhang, and Nenghai Yu. 2023. Ada3diff: Defending against 3d adversarial point clouds via adaptive diffusion. In *Proceedings of the 31st ACM International Conference on Multimedia.* 8849–8859.

[49] Yan Zhang, Zihao Liu, Chongliu Jia, Yi Zhu, and Chenglin Miao. 2024. An Online Defense against Object-based LiDAR Attacks in Autonomous Driving. In *Proceedings of the 22nd ACM Conference on Embedded Networked Sensor Systems.* 380–393.

[50] Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. 2019. Pointcloud saliency maps. In *IEEE/CVF International Conference on Computer Vision (ICCV).*

[51] Yu Zheng, Zheng Li, Xiaolong Xu, and Qingzhan Zhao. 2022. Dynamic defenses in cyber security: Techniques, methods and challenges. *Digital Communications and Networks* 8, 4 (2022), 422–435.

[52] Hang Zhou, Kejiang Chen, Weiming Zhang, Han Fang, Wenbo Zhou, and Nenghai Yu. 2019. DUP-Net: Denoiser and Upsampler Network for 3D Adversarial Point Clouds Defense. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*.

[53] Shenchen Zhu, Yue Zhao, Kai Chen, Bo Wang, Hualong Ma, et al. 2024. AE-Morpher: Improve Physical Robustness of Adversarial Objects against {LiDAR-based} Detectors via Object Reconstruction. In *33rd USENIX Security Symposium (USENIX Security 24)*. 7339–7356.

[54] Yi Zhu, Chenglin Miao, Foad Hajiaghajani, Mengdi Huai, Lu Su, and Chunming Qiao. 2021. Adversarial attacks against lidar semantic segmentation in autonomous driving. In *ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

[55] Yi Zhu, Chenglin Miao, Hongfei Xue, Yunnan Yu, Lu Su, and Chunming Qiao. 2024. Malicious attacks against multi-sensor fusion in autonomous driving. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (MobiCom)*. 436–451.

[56] Yi Zhu, Chenglin Miao, Tianhang Zheng, Foad Hajiaghajani, Lu Su, and Chunming Qiao. 2021. Can we use arbitrary objects to attack lidar perception in autonomous driving?. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*.