SlowLiDAR: Increasing the Latency of LiDAR-Based Detection Using Adversarial Examples

Han Liu, Yuhao Wu, Zhiyuan Yu, Yevgeniy Vorobeychik, Ning Zhang
Washington University in St. Louis
{h.liu1,yuhao.wu,yu.zhiyuan,yvorobeychik,zhang.ning}@wustl.edu

Abstract

LiDAR-based perception is a central component of autonomous driving, playing a key role in tasks such as vehicle localization and obstacle detection. Since the safety of LiDAR-based perceptual pipelines is critical to safe autonomous driving, a number of past efforts have investigated its vulnerability under adversarial perturbations of raw point cloud inputs. However, most such efforts have focused on investigating the impact of such perturbations on predictions (integrity), and little has been done to understand the impact on latency (availability), a critical concern for real-time cyber-physical systems. We present the first systematic investigation of the availability of LiDAR detection pipelines, and SlowLiDAR, an adversarial perturbation attack that maximizes LiDAR detection runtime. The attack overcomes the technical challenges posed by the non-differentiable parts of the LiDAR detection pipelines by using differentiable proxies and uses a novel loss function that effectively captures the impact of adversarial perturbations on the execution time of the pipeline. Extensive experimental results show that SlowLiDAR can significantly increase the latency of the six most popular LiDAR detection pipelines while maintaining imperceptibility [1].

1. Introduction

The promise of autonomous transit has stimulated extensive efforts towards the development of self-driving platforms [1–3, 5]. A central feature of most such platforms is a LiDAR-based perceptual pipeline (usually integrated with other sensors, such as camera and radar) for critical control tasks, such as localization and object detection [1, 3, 51]. Since errors in localization or obstacle detection can cause the vehicle to crash, these tasks are crucial in ensuring the safety of an autonomous vehicle. As a result, extensive prior research has been devoted to understanding and assuring the robustness of the LiDAR-based perceptual pipelines against adversarial perturbations on its raw point cloud inputs [22, 51, 53, 54]. A key focus in prior work has been on perceptual prediction accuracy, for example, the ability of the adversary to hide real obstacles or create phantom obstacles [15,32,43,44]. Another significant aspect of the robustness analysis of LiDAR-based perception that has received little attention is its *real-time performance (availability)*. In particular, a delay in perceptual processing caused solely by computational quirks of the LiDAR processing pipeline can be just as damaging as a mistaken prediction. For example, a delay in obstacle recognition can cause the vehicle to react to an obstacle too late, failing to avoid a crash.

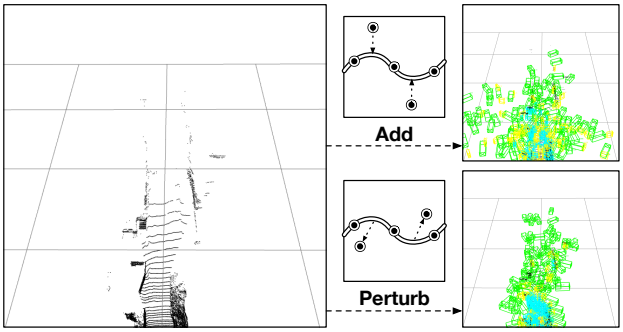


Figure 1. **SlowLiDAR attack.** The attack goal is to maximize the runtime latency of the state-of-the-art LiDAR detection models with either an adding-based attack or a perturbation-based attack.

We present the first systematic analysis of the impact of adversarial point cloud perturbations on the runtime latency of common LiDAR perceptual pipelines. To this end, we propose SlowLiDAR, an algorithmic framework for adding adversarial perturbations to point cloud data aiming to increase the execution time of LiDAR-based object detection. Specifically, we consider two attack mechanisms (see Figure 1): 1) perturbation of the 3D coordinates of existing points in the raw point cloud (*point perturbation attacks*), and 2) adding points to the raw point cloud (*point addition attacks*). Due to the unique representation and processing procedure of LiDAR detec-

[1] Code is available at: https://github.com/WUSTL-CSPL/SlowLiDAR

tion pipelines, existing availability attacks on other AI pipelines, such as camera-based detection [39], cannot be directly applied. Specifically, there are three new challenges: C1. Non-differentiable aggregation. LiDAR point clouds are generally sparsely distributed in a large 3D space without an organized pattern. To process such unstructured data, the state-of-the-art LiDAR-based detection models extract aggregated features of the points at the level of either 2D or 3D cells [30, 52, 55]. However, the aggregation operation is by nature non-differentiable [14], which presents new challenges to end-to-end optimization. C2. New objective function. Different than existing attacks targeting accuracy, a novel loss function needs to be carefully designed to effectively capture the impact of adversarial perturbations on runtime latency. C3. Large search space. Because points are distributed in a large space, the perturbation search space is quite large, and conventional gradient-based optimization approaches for crafting attacks may yield poor local optima.

We address these technical challenges as follows. First, in order to enable end-to-end training of adversarial input perturbations, we develop a differentiable proxy to approximate the non-differentiable pre-processing pipelines. Second, we conduct a response time analysis on the detection pipeline to identify the most vulnerable components, and design a novel loss function to best capture the impact of input perturbations on the runtime latency of the identified components. Third, to tackle the large search space, we propose a probing algorithm to identify high-quality initialization for gradient-based attack optimization.

We evaluate SlowLiDAR on six popular LiDAR-based detection frameworks that are adopted in modern commercial autonomous driving systems. Our experiments show that SlowLiDAR is effective in slowing down the models while retaining comparable imperceptibility. Moreover, the performance of our attacks remains consistent across different hardware and implementations. Our contributions can be summarized as follows:

- We systematically dissect the LiDAR detection models to analyze the attack surface of their runtime latency to adversarial inputs.
- We propose the first adversarial perturbation attacks against LiDAR detection with the goal of maximizing runtime latency. To this end, we overcome the non-differentiability of the pre-processing pipelines in order to perform end-to-end gradient-based attack optimization, and design a novel loss function to capture the impact of input perturbations on execution time.
- We evaluate our attacks on six popular LiDAR detection models in modern commercial autonomous driving systems in different hardware and implementations to demonstrate the ability to significantly increase latency with comparable imperceptibility.

2. Related Work

Point Cloud Recognition. Deep learning has achieved remarkable progress in 2D image-based recognition tasks in computer vision [24, 26, 38], and has been recently applied to 3D point cloud-based recognition for its exceptional performance. Differing on the pre-processing steps, two of the most popular LiDAR detection models are bird's eye view (BEV)-based detection, and voxel-based detection. BEV-based methods generally project point clouds into a top-down view and learn features through 2D convolutional neural networks (CNNs) [18, 33, 52]. Notably, the widely-used autonomous driving platform *Apollo* [1] adopts BEV-based methods in its LiDAR perception module. In contrast, voxel-based methods slice the point clouds into 3D voxel grids and extract features through a voxel-based detection network [29, 30, 55]. *Autoware* [3], another popular industry-level self-driving platform, adopts PointPillars [30]. Furthermore, there is another line of work that directly learns the feature from the raw point cloud, often referred to as point-based methods [36, 37, 40].

Adversarial Point Cloud Attack. Recognizing the importance of security, recent works have investigated the feasibility of adversarial attacks on point cloud recognition models. Zheng et al. proposed an untargeted attack by utilizing a saliency map to drop out critical points [53], while Xiang et al. proposed a targeted attack by perturbing or generating the points while using several metrics (e.g., Chamfer Measurement) to improve the imperceptibility [48]. Subsequent studies focused on improving imperceptibility by considering various shapes and positions of the point cloud [27, 34, 47]. In contrast to existing adversarial attacks that target integrity, our attack focuses on availability, requiring a new formulation to capture the temporal property.

Availability-based Slowdown Attack. Recently, availability has emerged as a new attack surface [31, 45]. Shumailov et al. were the first to present an availability-based attack [41], exploiting the vulnerability of the embedding process of NLP models. Following [41], there have been studies that target adaptive neural networks [23, 25] and token-output process of neural image caption generation models [17]. Closely related to our work, Shapira et al. proposed a slowdown attack on camera-based object detection algorithms by exploiting the vulnerability of the post-processing stage, overloading the models with predicted candidate bounding boxes [39]. However, availability attack on LiDAR detection models require new formulations due to the differences in representations and processing procedures.

3. Availability Attack Surface Analysis

LiDAR Processing Pipelines. LiDAR-based detection modules are extensively utilized in autonomous driving ve-

Algorithm 1: Non-maximum Suppression

Input: Bounding box proposals $\mathcal{C}$, confidence scores of proposals $\mathcal{S}$, confidence score threshold T, maximum number of boxes threshold K, IoU threshold I.
Output: Selected bounding box proposals $\mathcal{C}^*$, confidence scores of selected proposals $\mathcal{S}^*$.
$\mathcal{S}' \leftarrow \{\}, \mathcal{C}' \leftarrow \{\}, \mathcal{S}^* \leftarrow \{\}, \mathcal{C}^* \leftarrow \{\}$
$\downarrow \mathcal{O}(N), \Omega(N)$
for s, c *in* $Zip(\mathcal{S}, \mathcal{C})$ **do**
  if $s > T$ **then**
    $\mathcal{S}' \leftarrow \mathcal{S}' \cup \{s\}, \mathcal{C}' \leftarrow \mathcal{C}' \cup \{c\}$
  end
end
$\downarrow \mathcal{O}(NlogN), \Omega(N)$
$\mathcal{S}'', \mathcal{C}'' \leftarrow Sort(\mathcal{S}', \mathcal{C}')[:K]$
$\downarrow \mathcal{O}(K^2), \Omega(K)$
while $|S''| > 0$ **do**
  $\mathbf{u} \leftarrow IoU(S''(0), S''(1:))$ **if** $\mathbf{u} > I$ **then**
    $\mathcal{S}'' \leftarrow \{\mathcal{S}''\} \backslash \{s_u\}, \mathcal{C}'' \leftarrow \{\mathcal{C}''\} \backslash \{c_u\}$
  end
  $\mathcal{S}'' \leftarrow \{\mathcal{S}''\} \backslash \{s_0\}, \mathcal{C}'' \leftarrow \{\mathcal{C}''\} \backslash \{c_0\}$
  $\mathcal{S}^* \leftarrow \mathcal{S}^* \cup \{s_0\}, \mathcal{C}^* \leftarrow \mathcal{C}^* \cup \{c_0\}$
end
return $\mathcal{S}^*, \mathcal{C}^*$

hicles for environment perception. LiDAR first uses laser scanning to collect point cloud data, which contains the 3D coordinates and intensities of the reflected points. A detection model then processes the point clouds and identifies obstacles. A common pipeline of LiDAR-based detection models comprises three modules: a pre-processing module, a backbone network, and a post-processing module.

Pre-processing often falls into either BEV-based that projects point cloud onto feature maps [18, 33, 52] or voxel-based that transforms point cloud into 3D voxel grids [29, 30, 55]. The backbone network processes the pre-processed feature maps and generates a set of bounding boxes that contain information on potential obstacles, such as confidence scores, location coordinates, orientation, etc. The post-processing module filters proposals with low confidence scores and performs a non-maximum suppression (NMS) to remove proposals with large overlapping areas.

Response Time Analysis. There are three key steps in the processing pipeline. First, the pre-processing step typically employs a rule-based point aggregation manipulation. For example, in voxel-based detection models, the 3D space is divided into voxel units, and each raw point is transposed into the corresponding voxel based on its coordinates. The time complexity of this process is $\mathcal{O}(n)$, where n represents the number of points in the point cloud. Due to the linear time complexity, a considerable amount of points needs to be added to effectively slow this process down, resulting in increased cost and perceptibility of the attack.

Second, the time consumption of the backbone network in LiDAR detection models depends on the computation dimension [41] and the number of computations [23, 25]. However, the input dimension and number of computations for the backbone network are generally fixed.

Lastly, in the post-processing module, the number of proposals before the NMS module can vary greatly for different inputs. The NMS algorithm, which is provided in Algorithm 1, first filters out proposals by a pre-defined confidence score threshold. The best-case and worst-case time complexity for this step is $\Omega(N)$ and $\mathcal{O}(N)$, respectively, where N is the number of bounding box proposals. Next, the proposals are sorted by their confidence scores and filtered by a pre-defined threshold. Since the default sorting algorithm in Python is TimSort [12], this process has a best-case time complexity of $\Omega(N)$ and a worst-case time complexity of $\mathcal{O}(NlogN)$, where N represents the number of proposals after filtering. Finally, the proposal with the highest score is selected, and other proposals with a significant overlap are filtered out. This step can have significant variance in terms of time complexity. In the best-case scenario, where all proposals are highly overlapping, the loop will run only once with a time complexity of $\Omega(K)$. In contrast, in the worst-case scenario, every proposal is highly dispersed, and the loop will continue until only one proposal is left, resulting in a time complexity of $\mathcal{O}(K^2)$.

4. Methodology

4.1. Threat Model, Formulation, and Overview

Threat Model and Formulation. For a LiDAR detection model $\mathcal{D}$, given the pristine 3D point cloud x, the adversary aims to craft an adversarial example x^* to maximize the latency of the processing pipeline while remaining imperceptible. We formulate this as the following optimization problem:

$$\underset{x^*}{\operatorname{argmax}} \mathcal{T}(\mathcal{M}(x^*)) \text{ s.t. } \mathcal{D}(x, x^*) < \eta, \tag{1}$$

where $\mathcal{T}$ is the latency measurement function, $\mathcal{M}$ is the LiDAR detection model, and $\mathcal{D}$ is the distance between the original and adversarial point clouds. Following prior work on adversarial perturbations to LiDAR point clouds [34, 47, 51, 56], we make the assumption of over-the-line adversary capable of modifying sensed LiDAR data, and consider two types of attacks to analyze the algorithmic latency robustness: 1) *point perturbation* attacks and 2) *point addition* attacks. Point perturbation attacks modify the 3D coordinates of existing points, subject to a constraint that limits the magnitude of coordinate perturbations. Addition attacks generate a new set of adversarial points and combine them with the original points to form the final adversarial point clouds, while restricting both the number of inserted points and their distance from the original points.

Overview. In this paper we propose SlowLiDAR, the first adversarial attack maximizing the latency of LiDAR detection models. The overview of the attack pipeline is given in

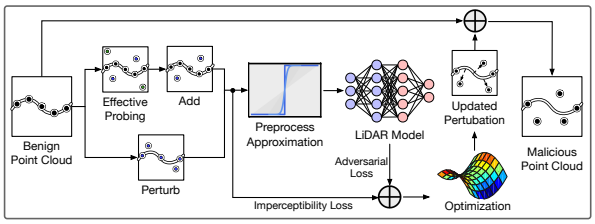


Figure 2. Overview of proposed SlowLiDAR attacks.

Figure 2. There are two types of attacks: in perturbation attacks, the positions of existing points are optimized; in addition attacks, extra points are first initialized and then get optimized. As discussed previously (§1), there are three challenges. To address C1, the pre-processing procedure is approximated with a differentiable operation to perform end-to-end training (§4.2). To address C2, we design both an adversarial loss function and an imperceptibility loss function for tackling the optimization in a new scenario (§4.3). To address C3, we propose an effective probing algorithm to identify a suitable initialization point in the large search space (§4.4).

4.2. Pre-processing Pipeline Approximation

To perform end-to-end attacks on raw point clouds, the pre-processing pipelines of two widely-used LiDAR detection models, BEV-based models [33, 52] and voxel-based models [29, 55], are dissected. The details of these pre-processing pipelines are described in the following.

BEV-based Detection Model. The BEV-based detection methods first slice the point cloud into evenly-spaced cubes in the 3D space, with each cube having the same size $L \times W \times H$. Then, the occupancy tensor is determined by an occupancy encoding procedure. Specifically, for each cube, if there exist points within the grid, the value is 1 otherwise it is 0. The reflectance data is aggregated along the height dimension and normalized to be within $[0, 1]$ to get a 2D reflectance tensor. The final BEV representation is a combination of the 3D occupancy tensor and the 2D reflectance tensor. Despite the increase in processing efficiency, the occupancy encoding is by nature non-differentiable since it produces discontinuous binary values. To render the occupancy encoding differentiable, we apply a differentiable and accurate approximation of the encoding process. Specifically, we formulate the encoding values following trilinear interpolation [13, 14] with the occupancy value for a point p_i in cube c_j computed as

$$O(p_i, c_j) = \left(1 - \frac{d(l_i, l_j)}{L}\right) \cdot \left(1 - \frac{d(w_i, w_j)}{W}\right) \cdot \left(1 - \frac{d(h_i, h_j)}{H}\right), \tag{2}$$

where l_i is the x-coordinate, w_i is the y-coordinate, and h_i is the z-coordinate of the point p_i. Similarly, l_j, w_j, h_j represents the (x, y, z) coordinates of the central point in cube c_j. The function $d(\cdot)$ is defined as

$$d(a, b) = \frac{X}{2} + \frac{X}{2} \cdot \text{sign}\left(\|a - b\| - \frac{X}{2}\right). \tag{3}$$

The intuition behind this approximation is that when the distance between point p_i and the central coordinate of cube c_j for all coordinates is smaller than the dimension of the cube, the occupancy value is 1; otherwise, the occupancy value is 0. However, the formulated encoding procedure is still non-differentiable due to the existence of the sign function. To address this, we apply principal Padé approximation [28] to approximate the sign function as

$$\text{sign}(z) \approx \tanh(k \cdot \text{arctanh}(z)), \tag{4}$$

where k controls the steepness of the approximation: a larger k yields a better approximation, but also less stable gradients, making the optimization problem more challenging. After substituting 4 into 3, we get a full approximation rendering of the BEV pre-processing pipelines.

Voxel-based Detection Model. The voxel-based models first slice the point cloud into the evenly spaced grid in the 3D space and then put each point into the grid according to its coordinates. The number of points in each voxel is generally constrained, random sampling is applied when a voxel holds too much data, and zero padding is applied when the voxel contains too little data. After removing the empty voxels, the final dense voxel representation is obtained. This process is differentiable and the gradient can be directly propagated to each individual raw point. Typically, to accelerate the training and inference process, the pre-processing pipelines are implemented in C++[2] [3]; we re-implemented this code using Pytorch [4] to ensure a differentiable end-to-end pipeline.

4.3. Loss Function Design

Adversarial Loss. As shown in the response time analysis given in Section 3, the main computational complexity is from the sorting procedure and the box selection procedure. Therefore, in order to maximize the processing latency, we need to increase the number of bounding box proposals before the confidence-based filtering process and also increase the number of iterations in the box-selection procedure. In LiDAR detection models, for each bounding box proposal it predicts, it would also predict a confidence score. In order to increase the number of boxes after filtering, we need to maximize the number of boxes whose confidence score is above the filtering threshold. To achieve this, we formulate the loss function as

[2] PIXOR: https://github.com/philip-huang/PIXOR
[3] VoxelNet: https://github.com/skyhehe123/VoxelNet-pytorch
[4] Pytorch: https://pytorch.org

$$L_{conf} = \frac{1}{N}\sum_{n}\max(T_{conf} - C_n, \lambda), \tag{5}$$

where N represents the total number of predicted proposals, C_n represents the confidence scores of proposal n, T_{conf} is the filtering threshold set by the model, and λ is a control parameter to prevent our algorithm from increasing the confidence scores of several candidate proposals.

To increase the number of iterations in the proposal-selection procedure, it is crucial to ensure that proposals are sparsely distributed and are not overlapping with one another, since overlapping proposals tend to get filtered out within a few iterations. Generally, the larger the area of proposals and the closer two proposals are, the easier it is for two proposals to overlap. Therefore, we formulate the loss function as

$$L_{prop} = \frac{1}{N(N-1)}\sum_{i}\sum_{j\neq i}\max\left(\frac{\sqrt{S_i}\cdot\sqrt{S_j}}{D_{i,j}}, \kappa\right), \tag{6}$$

where S_i represents the area of proposal i, and $D_{i,j}$ represents the distance between the central coordinates of proposals i and j. Similarly, we use a parameter κ to control the magnitude. Then the adversarial loss is represented by

$$L_{adv} = L_{conf} + \alpha L_{prop}, \tag{7}$$

Imperceptibility Loss. Following the existing methods [22, 34, 51], we adopt the Chamfer Distance (CD) [19] as the perturbation metric, defined as the average distance of all nearest point pairs. Formally, the loss function is

$$L_{cd} = \frac{1}{\|P^*\|}\sum_{p_1\in P^*}\min_{p_2\in P}\|p_1 - p_2\|_2^2, \tag{8}$$

where P^* is the adversarial and P the original point cloud. Then, the total loss function is defined as

$$L_{total} = L_{adv} + \beta L_{cd}. \tag{9}$$

Ensemble Training. Since the adversary cannot always have white-box access to the targeted models, attack transferability is important. Ensemble training over multiple models can help generalize the adversarial examples and improve transferability [39, 46]. We formulate the ensemble training process as a min-max optimization problem to find an ideal performance balance between different models following [49]. Specifically, we define our problems as

$$\min_{p^*\in\mathcal{P}}\max_{\boldsymbol{w}\in\mathcal{W}}\sum_{i} w_i L_{adv}^i(p^*) - \frac{\sigma}{2}\left\|\boldsymbol{w} - \frac{1}{N}\right\|_2^2 + \beta L_{cd}(p, p^*), \tag{10}$$

where p is the original cloud, p^* is the adversarial point cloud, $\mathcal{W}$ is the a probabilistic simplex given by $\mathcal{W} = \{\boldsymbol{w}|1^T\boldsymbol{w} = 1 \geq 0\}$, L_{adv}^i is the adversarial loss function for detection model i, N is the total number of models, and σ is a regularization parameter.

Algorithm 2: Effective Probing

Input: Probing steps K, number of added points N_{add}, number of probing point N_{prob}, original point cloud P_{ori}, Gaussian distribution mean μ_1, μ_2, μ_3 and variance $\sigma_1^2, \sigma_2^2, \sigma_3^2$.
Output: Initial coordinate of added points P^*.
$S^* \leftarrow \{\}, P^* \leftarrow \{\}$
for $i = 1$ *to* K **do**
  $P \leftarrow \{\}$
  while $i < N_{prob}$ **do**
    $x \leftarrow \mathcal{N}(\mu_1, \sigma_1^2), y \leftarrow \mathcal{N}(\mu_2, \sigma_2^2), z \leftarrow \mathcal{N}(\mu_3, \sigma_3^2)$
    $P \leftarrow P \cup \{(x, y, z)\}$
  end
  $S \leftarrow \|\frac{\partial L_{total}(\mathcal{M}(P\cup P_{ori}))}{\partial P}\|_2^2$
  $S^* \leftarrow S^* \cup S, P^* \leftarrow P^* \cup P$
  $\mathcal{S}^*, P^* \leftarrow Sort(\mathcal{S}^*, P^*)[: N_{add}]$
end
return $\mathcal{P}^*$

4.4. Effective Probing for Initialization

Point addition attacks are non-trivial: given that LiDAR point clouds are sparsely distributed in a large space, the optimization on such a large scale can easily converge to a local optimal. To address this issue, we need to find a good initialization position. To this end, we propose an effective probing (EP) algorithm for point initialization (see Algorithm 2 for details). Specifically, we generate a large set of random probing points. We then calculate the importance of each point using the gradient with respect to the loss function as

$$S = \left\|\frac{\partial L_{total}(\mathcal{M}(P\cup P_{ori}))}{\partial P}\right\|_2^2, \tag{11}$$

where P is the added point set, P_{ori} is the original point set, and S is the importance scores corresponding to each point in P. This step is repeated to keep the points with the highest importance scores.

5. Experiments

5.1. Experimental Setup

Datasets and Target Models. We use the KITTI dataset [20] for training and evaluating the proposed attacks. The KITTI dataset contains LiDAR point clouds collected in the wild and 3D bounding box labels, serving as a widely-used vision benchmark for autonomous driving with 7481 point clouds for training, and 7518 point clouds for testing. For target models, we cover two popular types of LiDAR detection models in commercial autonomous driving systems: BEV-based models - PIXOR [52], Complex-YOLO [42], and YOLO3D [11] and voxel-based models - PointPillar [30], VoxelNet [55], and Second [50]. We choose popular open-sourced repositories as targets with two different implementations of NMS in Python [4, 6, 8, 10], and C++ [7, 9] for each model. We train them on KITTI dataset following

their original works to obtain the victim models.

Evaluation Metrics. To measure the increase in response latency of the model, we measure the rate of increase (ROI) in the number of candidate proposals before NMS and processing time. Additionally, we define two metrics regarding processing time, ROI-P and ROI-Latency, in terms of the rate of increase in proposals and latency as

$$\begin{aligned} \text{ROI-Proposal} &= \frac{\text{Proposal}(x^*) - \text{Proposal}(x)}{\text{Proposal}(x)}, \\ \text{ROI-Latency} &= \frac{\text{Latency}(x^*) - \text{Latency}(x)}{\text{Latency}(x)}. \end{aligned} \tag{12}$$

We compute the ROI-Latency for both NMS stage (ROI-NMS) and end-to-end processing time (ROI-TT).

Testing Hardware. We evaluated the effectiveness of our attack in various hardware to simulate real-world deployed platforms of LiDAR detection models since processing latency is hardware-dependent. We tested the latency on 3 hardware platforms: AMD Ryzen 9 3900X CPU, GeForce RTX 3070 Ti GPU, and GeForce RTX 3090 Ti GPU.

Comparison Baselines. To the best of our knowledge, we are the first to propose slowdown attacks against LiDAR detection models while existing adversarial attacks focus on corrupting the integrity of the models. To evaluate the performance of SlowLiDAR and show that existing attacks do not effectively slow down the models, we compare SlowLiDAR with three LiDAR integrity-based attack algorithms: PGD [35], I-FGM [21], and CW [16]. For each algorithm, we implement both the point perturbation and addition attacks, training the samples with randomly chosen target labels under the Chamfer distance constraint. For addition attacks, in order to successfully change the labels, we follow [51] to add 10% of the total points.

Implementation Details. For all the LiDAR detection models, we adopt the threshold setting of the NMS stage in the official implementation. In our ablation study, we further alter these thresholds to observe their effect on attacks. For addition attacks, we set the maximum number of added points as 3% of the number of original points. We use Adam optimizer with learning rate 10^{-2} for perturbation attacks, and 10^{-3} for addition attacks. For effective probing, we set the number of probing points in each step to 3000 and the probing step to 30. Each adversarial sample is trained with 2000 iterations.

5.2. Evaluation Results

A comparison between our proposed method and baseline methods is shown in Table 1. SlowLiDAR can increase the total number of proposals in PointPillar up to 165.92 times for perturbation attacks and 156.16 times for addition attacks. However, a larger ROI-P does not necessarily result in higher latency, as it may depend on the implementation and the size of base candidate proposals. Therefore, we further measured the processing latency.

Table 1. Comparison of rate of increase in proposals (ROI-P) before NMS stage with baseline in LiDAR detection models.

Model	Attack Type	PGD	CW	I-FGM	Slow-LiDAR
PIXOR [52]	Perturb	0.05	0.00	0.01	35.21
	Add	0.55	0.16	0.22	45.15
YOLO3D [11]	Perturb	-0.50	0.00	-0.50	499.00
	Add	2.50	1.50	2.00	479.00
Complex-YOLO [42]	Perturb	1.00	0.00	0.00	539.00
	Add	5.00	2.00	2.00	552.00
VoxelNet [55]	Perturb	0.25	0.00	0.03	48.29
	Add	0.83	0.14	0.28	33.38
PointPillar [30]	Perturb	0.16	0.03	0.05	165.92
	Add	0.62	0.16	0.29	156.16
Second [50]	Perturb	0.22	0.00	0.11	127.70
	Add	0.44	0.22	0.22	187.20

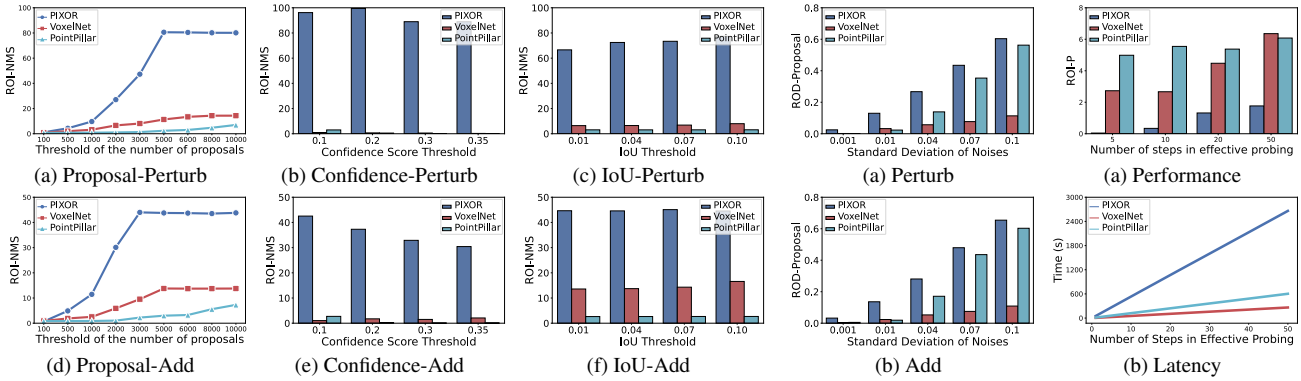
Table 2 presents the results of our processing latency tests on various hardware configurations. Our proposed system, SlowLiDAR, can increase the processing latency in both the NMS-stage and end-to-end. Specifically, we observed an average 116 times increase in the processing time of the NMS-stage for perturbation attacks, and an average 65.75 times increase for addition attacks in the PIXOR model. For the end-to-end processing time, we observed that it can be slowed down up to 85.08 times in perturbation attacks and up to 48.22 times in addition attacks on the 3090 Ti GPU. The difference in the effectiveness of slowdown between the CPU and the two GPUs is due to the slower model forward time on the CPU, which limits the effectiveness of increasing the NMS-stage latency. Additionally, models with NMS implemented in Python are more vulnerable compared to those implemented in C++, with the exception of Second, which is implemented in Python but equipped with GPU acceleration. Furthermore, our SlowLiDAR system achieved a slightly larger Chamfer distance for perturbation attacks, as compared to the accuracy-based baseline methods. This is because the baseline methods perturb the points around certain proposals to flip their predictions, while our SlowLiDAR system aims to perturb all points to create as many proposals as possible.

5.3. Ablation Study

Attack Effectiveness Under Different Thresholds. As discussed in Section 3, the effectiveness of attacks on LiDAR models is most impacted by three factors: (i) proposal limit thresholds, (ii) confidence score thresholds, and (iii) IoU thresholds. To evaluate their impact on latency, we varied the thresholds with different values and measured processing latency in the NMS stage on the 3070 Ti GPU. Our results show that (i) and (ii) visibly impact the defense capability, as illustrated in Figure 3. However, setting better thresholds cannot fully defend against our attacks. For in-

Table 2. Results of NMS-stage and end-to-end processing latency with CD in different LiDAR detection models and hardware.

Model	Attack Type	Hardware	PGD			CW			I-FGM			Slow-LiDAR		
			ROI-NMS↑	ROI-TT↑	CD↓	ROI-NMS↑	ROI-TT↑	CD↓	ROI-NMS↑	ROI-TT↑	CD↓	ROI-NMS↑	ROI-TT↑	CD↓
PIXOR [52]	Perturb	CPU	0.04	0.01		0.00	0.00		0.01	0.01		168.10	4.91	
		3070 Ti	0.04	0.01	3.69	0.00	0.00	0.00	0.01	0.01	0.01	79.10	65.07	0.60
		3090 Ti	0.04	0.01		0.00	0.00		0.01	0.01		116.00	85.08	
	Add	CPU	0.04	0.01		0.02	0.00		0.02	0.05		94.24	2.80	
		3070 Ti	0.04	0.01	8.87	0.02	0.00	1.10	0.02	0.00	2.56	42.48	35.53	0.78
		3090 Ti	0.04	0.01		0.02	0.00		0.02	0.00		65.75	48.22	
YOLO3D [11]	Perturb	CPU	0.00	0.00		0.00	0.00		0.00	0.00		2019.11	229.11	
		3070 Ti	0.00	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.00	2031.35	230.68	0.07
		3090 Ti	0.00	0.00		0.00	0.00		0.00	0.00		3136.99	327.33	
	Add	CPU	0.00	0.00		0.00	0.00		0.00	0.00		815.62	92.30	
		3070 Ti	0.00	0.00	1.55	0.00	0.00	0.88	0.00	0.00	0.12	825.24	94.30	0.78
		3090 Ti	0.00	0.00		0.00	0.00		0.00	0.00		907.80	94.73	
Complex-YOLO [11]	Perturb	CPU	0.01	0.00		0.01	0.00		0.01	0.00		398.98	7.79	
		3070 Ti	0.01	0.00	0.88	0.01	0.00	0.01	0.01	0.00	0.00	400.08	134.94	0.29
		3090 Ti	0.01	0.00		0.01	0.00		0.01	0.00		444.72	136.76	
	Add	CPU	0.03	0.01		0.02	0.01		0.02	0.01		103.76	2.02	
		3070 Ti	0.03	0.01	1.89	0.02	0.01	0.67	0.02	0.01	1.42	104.25	36.95	0.44
		3090 Ti	0.03	0.01		0.02	0.01		0.02	0.01		118.70	36.41	
VoxelNet [55]	Perturb	CPU	0.05	0.01		0.01	0.00		0.02	0.01		14.32	2.41	
		3070 Ti	0.05	0.01	8.32	0.01	0.00	0.01	0.02	0.01	0.04	14.13	2.37	0.85
		3090 Ti	0.05	0.01		0.01	0.00		0.02	0.01		14.36	2.88	
	Add	CPU	0.05	0.01		0.02	0.01		0.02	0.01		13.65	1.80	
		3070 Ti	0.05	0.01	18.87	0.02	0.01	3.03	0.02	0.01	5.90	13.58	1.75	2.48
		3090 Ti	0.05	0.01		0.02	0.01		0.02	0.01		13.88	1.92	
PointPillar [30]	Perturb	3070 Ti	0.05	0.01	3.86	0.02	0.01	0.00	0.02	0.01	0.02	7.86	2.72	0.41
		3090 Ti	0.05	0.01		0.02	0.01		0.02	0.01		8.65	3.12	
	Add	3070 Ti	0.05	0.01	11.88	0.02	0.01	1.73	0.02	0.01	4.73	6.77	1.91	1.10
		3090 Ti	0.05	0.01		0.02	0.01		0.02	0.01		9.23	2.30	
Second [50]	Perturb	3070 Ti	0.01	0.00	2.34	0.01	0.00	0.01	0.01	0.00	0.02	40.55	3.56	0.40
		3090 Ti	0.02	0.00		0.01	0.00		0.01	0.00		32.33	4.22	
	Add	3070 Ti	0.03	0.01	3.23	0.02	0.00	1.11	0.02	0.00	2.88	22.67	2.39	0.59
		3090 Ti	0.03	0.01		0.02	0.01		0.02	0.01		19.05	2.68	



(a) Proposal-Perturb (b) Confidence-Perturb (c) IoU-Perturb (a) Perturb (a) Performance

(d) Proposal-Add (e) Confidence-Add (f) IoU-Add (b) Add (b) Latency

Figure 3. Attack effectiveness under different thresholds in NMS stage.

Figure 4. Robustness tests.

Figure 5. EP Evaluation.

stance, when the proposal limit is set to 3000 in PIXOR, the ROI-NMS is only 47.28. Similarly, with a confidence threshold of 0.35, the ROI-NMS is 88.9. Furthermore, fine-tuning (i) and (ii) as defenses are not feasible since they significantly reduce performance on benign samples.

Feasibility of Physical Attacks. While our main emphasis is on the algorithmic aspects of attacks, we also investigated their feasibility under physical constraints. To account for the potential impact of noise, we conducted a robustness analysis by adding Gaussian noise with different standard deviations to the modified point clouds. Figure 4 shows the rate of decrease (ROD) of prediction proposals. For common noise at the level of 0.01m, the ROD is less than 3% for PointPillar and VoxelNet, and less than 13% for PIXOR. Additionally, we conducted an experiment with a revised loss function to account for the predefined grid of directions in which a LiDAR device shoots laser rays. The point cloud was first converted to spherical coordinates, and we used the sign of the gradient for variable updates to ensure a discrete resolution of optimization steps. Our attacks achieved an

ROI-P of 90.016 for PointPillar, 23.209 for VoxelNet, and 12.979 for PIXOR with a perturbation resolution of 0.02.

Variance of Running Time. To account for the potential variability in running time due to multiple factors, we repeated the experiments and measured the variances of ROI-NMS and ROI-TT, as presented in Table 3. The variance was found to be negligible compared to the original values.

Transferability. To evaluate the transferability of our attack, we train the samples on each targeted model, then test their effectiveness on them using the pre-trained samples. We also evaluate the effectiveness of the ensemble training strategy. The results are given in Table 4. As we can see, the transferability is limited compared to directly training on the target models due to the unique pre-processing pipelines in different LiDAR detection models. On the other hand, ensemble learning boosts transferability and increases the number of proposals by 12.994 times on PIXOR models.

Table 3. Variances of ROI-NMS and ROI-TT in repeated experiments with different hardware.

Model	Attack Type	Variance (CPU, 3070 Ti, 3090 Ti)					
		ROI-NMS			ROI-TT		
PIXOR	Perturb	0.124	0.329	0.318	0.024	0.086	0.084
	Add	0.302	0.171	0.229	0.061	0.137	0.008
VoxelNet	Perturb	0.060	0.034	0.042	0.009	0.020	0.012
	Add	0.037	0.016	0.183	0.024	0.012	0.014
PointPillar	Perturb	-	0.064	0.066	-	0.024	0.024
	Add	-	0.051	0.054	-	0.014	0.017

Effectiveness of Efficient Probing. To assess the effectiveness of our efficient probing (EP) algorithm for point initialization, we conducted experiments with different probing steps and evaluated the resulting ROI-P post-initialization. Figure 5 (a) shows that as the number of probing iterations increases, we obtain more initialized proposals, providing evidence of the effectiveness of our probing algorithm. However, the probing step introduces additional latency, as shown in Figure 5 (b). We found that processing time is linearly proportional to the number of probing steps.

6. Discussion

Real-world Impacts. Our study investigates the algorithmic runtime robustness of LiDAR detection models by manipulating digital data. We have further investigated the physical realizability by further considering the noise and working principles of lasers in LiDAR. Although our investigation is on digital domain manipulation, it expands the attack surface beyond accuracy-based attacks and sheds light on the vulnerability of autonomous driving systems from the availability perspective. Our tool can also aid the future investigation of runtime latency robustness of LiDAR processing pipelines in autonomous systems.

Table 4. Transferability of SlowLidar in different models.

Attack	Source \ Target	PIXOR	VoxelNet	PointPillar
Perturb	PIXOR	35.213	1.722	1.344
	VoxelNet	1.115	48.286	3.359
	PointPillar	3.318	1.087	165.922
	Min-max Train	8.259	6.239	7.321
Add	Source \ Target	PIXOR	VoxelNet	PointPillar
	PIXOR	45.149	0.831	3.016
	VoxelNet	1.213	33.381	2.183
	PointPillar	8.851	1.087	156.156
	Min-max Train	12.994	4.889	7.972

Limitations and Future Work. Firstly, our attack performance is limited in point-based detection models (e.g., PointRCNN) because the run-time variance of these models primarily depends on the process of candidate proposals in the first stage of the detection algorithm. Even though an adversary may manipulate runtime by creating more proposal candidates, we found that most existing implementations of point-based methods offer little room for adversarial manipulation due to the combination of aggressive proposal generation strategies with thousands of proposals generated simultaneously and a relatively low pre-defined proposal limit threshold. Second, even though we've modeled physical-world realizability, the feasibility and engineering challenges of tampering with physical-world signals remain unexplored. Third, we design SlowLiDAR attacks in a white-box setting. Although we evaluated the transferability of our attacks in a black-box setting through ensemble training, the performance of black-box attacks is limited compared to white-box attacks.

7. Conclusion

We present SlowLiDAR, the first adversarial perturbation attack that maximizes LiDAR detection runtime. To achieve the goal, we propose differentiable proxies to approximate the non-differentiable parts of the LiDAR detection pipelines, and design a novel loss function that effectively captures the impact of adversarial perturbations on runtime efficiency. Extensive experimental results show that SlowLiDAR can significantly increase the latency of the six most popular LiDAR detection pipelines while maintaining imperceptibility.

Acknowledgments

We thank Shixuan Zhai for his valuable assistance in revising this paper. This work was partially supported by the NSF (CNS-1837519, CNS-1916926, CNS-2038995, CNS-2229427, CNS-2238635, IIS-1905558, ECCS-2020289), ARO (W911NF2010141, W911NF1910241), and Intel.

References

[1] Apollo. https://www.apollo.auto. Accessed: 2022-08-28.

[2] Autopilot. https://www.tesla.com/autopilot. Accessed: 2022-08-28.

[3] Autoware. https://www.autoware.org. Accessed: 2022-08-28.

[4] Complex-yolo implementation. https://github.com/maudzung/Complex-YOLOv4-Pytorch. Accessed: 2022-10-01.

[5] Nio autonomous driving. https://www.nio.cn/nad. Accessed: 2022-08-28.

[6] Pixor implementation. https://github.com/philip-huang/PIXOR. Accessed: 2022-10-01.

[7] Pointpillar official implementation. https://github.com/open-mmlab/OpenPCDet. Accessed: 2022-10-30.

[8] Second implementation. https://github.com/traveller59/second.pytorch. Accessed: 2022-10-01.

[9] Voxelnet implementation. https://github.com/skyhehe123/VoxelNet-pytorch. Accessed: 2022-10-01.

[10] Yolo3d implementation. https://github.com/RichardMinsooGo-ML/Bible_3_42_Pytorch_Yolo_3d_Yolov4. Accessed: 2022-10-01.

[11] Waleed Ali, Sherif Abdelkarim, Mahmoud Zidan, Mohamed Zahran, and Ahmad El Sallab. Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018.

[12] Nicolas Auger, Vincent Jugé, Cyril Nicaud, and Carine Pivoteau. On the worst-case complexity of timsort. In *26th Annual European Symposium on Algorithms (ESA 2018)*, volume 112, pages 4–1, 2018.

[13] Paul Bourke. Interpolation methods. *Miscellaneous: projection, modelling, rendering*, 1(10), 1999.

[14] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 176–194. IEEE, 2021.

[15] Yulong Cao, Chaowei Xiao, Benjamin Cyr, Yimeng Zhou, Won Park, Sara Rampazzi, Qi Alfred Chen, Kevin Fu, and Z Morley Mao. Adversarial sensor attack on lidar-based perception in autonomous driving. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2267–2281, 2019.

[16] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[17] Simin Chen, Zihe Song, Mirazul Haque, Cong Liu, and Wei Yang. Nicgslowdown: Evaluating the efficiency robustness of neural image caption generation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15365–15374, 2022.

[18] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[19] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[21] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[22] Abdullah Hamdi, Sara Rojas, Ali Thabet, and Bernard Ghanem. Advpc: Transferable adversarial perturbations on 3d point clouds. In *European Conference on Computer Vision*, pages 241–257. Springer, 2020.

[23] Mirazul Haque, Anki Chauhan, Cong Liu, and Wei Yang. Ilfo: Adversarial attack on adaptive neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14264–14273, 2020.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] Sanghyun Hong, Yigitcan Kaya, Ionuț-Vlad Modoranu, and Tudor Dumitras. A panda? no, it's a sloth: Slowdown attacks on adaptive multi-exit neural network inference. In *International Conference on Learning Representations*, 2021.

[26] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[27] Qidong Huang, Xiaoyi Dong, Dongdong Chen, Hang Zhou, Weiming Zhang, and Nenghai Yu. Shape-invariant 3d adversarial point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15335–15344, 2022.

[28] Charles S Kenney and Alan J Laub. A hyperbolic tangent identity and the geometry of padé sign function iterations. *Numerical Algorithms*, 7(2):111–128, 1994.

[29] Hongwu Kuang, Bei Wang, Jianping An, Ming Zhang, and Zehan Zhang. Voxel-fpn: Multi-scale voxel feature aggregation for 3d object detection from lidar point clouds. *Sensors*, 20(3):704, 2020.

[30] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[31] Ao Li, Marion Sudvarg, Han Liu, Zhiyuan Yu, Chris Gill, and Ning Zhang. Polyrhythm: Adaptive tuning of a multi-channel attack template for timing interference. In *2022 IEEE Real-Time Systems Symposium (RTSS)*, pages 225–239. IEEE, 2022.

[32] Yiming Li, Congcong Wen, Felix Juefei-Xu, and Chen Feng. Fooling lidar perception via adversarial trajectory perturbation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7898–7907, 2021.

[33] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 641–656, 2018.

[34] Daniel Liu, Ronald Yu, and Hao Su. Adversarial shape perturbations on 3d point clouds. In *European Conference on Computer Vision*, pages 88–104. Springer, 2020.

[35] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[36] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[37] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[38] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[39] Avishag Shapira, Alon Zolfi, Luca Demetrio, Battista Biggio, and Asaf Shabtai. Denial-of-service attack on object detection model using universal adversarial perturbation. *arXiv preprint arXiv:2205.13618*, 2022.

[40] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[41] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 212–231. IEEE, 2021.

[42] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[43] Jiachen Sun, Yulong Cao, Qi Alfred Chen, and Z Morley Mao. Towards robust {LiDAR-based} perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th USENIX Security Symposium (USENIX Security 20)*, pages 877–894, 2020.

[44] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13716–13725, 2020.

[45] Jinwen Wang, Ao Li, Haoran Li, Chenyang Lu, and Ning Zhang. Rt-tee: Real-time system availability for cyber-physical systems using arm trustzone. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 352–369. IEEE, 2022.

[46] Jingkang Wang, Tianyun Zhang, Sijia Liu, Pin Yu Chen, Jiacen Xu, Makan Fardad, and Bo Li. Beyond adversarial training: Min-max optimization in adversarial attack and defense. *Nuclear Physics, Section A*, 2019.

[47] Yuxin Wen, Jiehong Lin, Ke Chen, CL Philip Chen, and Kui Jia. Geometry-aware generation of adversarial point clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[48] Chong Xiang, Charles R Qi, and Bo Li. Generating 3d adversarial point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9136–9144, 2019.

[49] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world. In *European conference on computer vision*, pages 665–681. Springer, 2020.

[50] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[51] Bo Yang, Yushi Cheng, Zizhi Jin, Xiaoyu Ji, and Wenyuan Xu. Generating 3d adversarial point clouds under the principle of lidars.

[52] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[53] Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. Pointcloud saliency maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1598–1606, 2019.

[54] Hang Zhou, Dongdong Chen, Jing Liao, Kejiang Chen, Xiaoyi Dong, Kunlin Liu, Weiming Zhang, Gang Hua, and Nenghai Yu. Lg-gan: Label guided adversarial network for flexible targeted attack of point cloud based deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10356–10365, 2020.

[55] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.

[56] Yi Zhu, Chenglin Miao, Tianhang Zheng, Foad Hajiaghajani, Lu Su, and Chunming Qiao. Can we use arbitrary objects to attack lidar perception in autonomous driving? In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1945–1960, 2021.