A Backdoor Attack against 3D Point Cloud Classifiers

Zhen Xiang[1], David J. Miller[1], Siheng Chen[2], Xi Li[1], and George Kesidis[1*]
[1]Pennsylvania State University [2]Shanghai Jiao Tong University

Abstract

Vulnerability of 3D point cloud (PC) classifiers has become a grave concern due to the popularity of 3D sensors in safety-critical applications. Existing adversarial attacks against 3D PC classifiers are all test-time evasion (TTE) attacks that aim to induce test-time misclassifications using knowledge of the classifier. But since the victim classifier is usually not accessible to the attacker, the threat is largely diminished in practice, as PC TTEs typically have poor transferability. Here, we propose the first backdoor attack (BA) against PC classifiers. Originally proposed for images, BAs poison the victim classifier's training set so that the classifier learns to decide to the attacker's target class whenever the attacker's backdoor pattern is present in a given input sample. Significantly, BAs do not require knowledge of the victim classifier. Different from image BAs, we propose to insert a cluster of points into a PC as a robust backdoor pattern customized for 3D PCs. Such clusters are also consistent with a physical attack (i.e., with a captured object in a scene). We optimize the cluster's location using an independently trained surrogate classifier and choose the cluster's local geometry to evade possible PC preprocessing and PC anomaly detectors (ADs). Experimentally, our BA achieves a uniformly high success rate ($\geq 87\%$) and shows evasiveness against state-of-the-art PC ADs. Code is available at `https://github.com/zhenxianglance/PCBA`.

1. Introduction

Tools for 3D point cloud (PC) classification have been developing rapidly due to the increasing popularity of 3D applications in industry such as autonomous driving, industrial robotics, and augmented reality [6, 13]. Recently, deep neural network (DNN) models, e.g. PointNet [4], have demonstrated tremendous performance in 3D PC classification; hence they are widely used as the backbone of many 3D PC processing modules. However, these models are vulnerable to adversarial attacks, which typically aim to induce misclassifications during the classifier's operation [28, 57]. In safety-sensitive domains such as autonomous driving, such misclassifications, e.g. incorrectly recognizing a pedestrian as a car (Fig. 1), can be catastrophic.

*Supported in part by an AFOSR DDDAS grant.

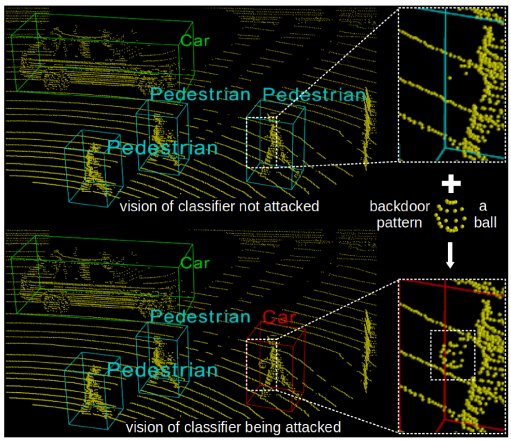


Figure 1: Illustration of a BA during the operation of a 3D PC classifier as part of an autonomous car. Top: If the classifier is not attacked, it functions normally. Bottom: The attacker embeds a backdoor pattern to a PC associated with a pedestrian (e.g. by having the pedestrian carry a ball). The backdoor-attacked classifier incorrectly recognizes the pedestrian as a car, which may be catastrophic.

Existing adversarial attacks against 3D PC classifiers are all *test-time evasion* (TTE) attacks [49]. These attacks aim to "fool" a classifier (i.e. inducing misclassifications) during testing/operation by introducing a customized modification of each test sample – this may involve adding points [49, 43], perturbing points [49, 47, 22], and/or deleting points [64]. These sample-specific modifications are optimized using full knowledge (the architecture and parameters) of the victim classifier to be "fooled". However, in many practical cases the victim classifier is not accessible to the attacker. Moreover, the *transferability* of existing PC TTEs is poor – adversarial test samples created using a surrogate classifier *independently* trained by the attacker do not reliably fool the victim classifier [49, 22]. Thus, the threat of PC TTE attacks in practice is largely diminished.

In this paper, we expose the vulnerability of 3D PC classifiers to a different attack by proposing the first PC backdoor attack (BA). Similar to the BAs proposed against image classifiers, our BA aims to have a 3D PC classifier learn to classify to the attacker's target class during its operation, whenever a test sample from a source class (of the attack) contains a backdoor pattern [12, 7, 65] (see illustration in

Fig. 1). To achieve this goal, the attacker poisons the training set of the victim classifier with a small set of *backdoor training samples*. These samples are originally from the source class, are embedded with the same backdoor pattern that will be embedded in test samples to "fool" the victim classifier, and are labeled to the target class [12]. Similar to traditional data poisoning (DP) attacks [2, 18, 56, 32, 1] and image BAs, our PC BA is based on the assumption that the attacker is able to poison the training set of the victim classifier [12, 7]. Such poisoning capability is facilitated by the need in practice to obtain "big data" suitable for accurately training a DNN classifier for a given domain – to do so, one may need to seek data from as many sources as possible (some of which could be attackers) [62].

Although BAs and their defenses have been extensively studied for images, devising a BA against 3D PC classifiers is challenging in several respects. *Challenge 1*: Existing backdoor patterns for image BAs are either a human-imperceptible, additive perturbation [7, 42, 65, 45, 53], or a pixel patch replacement representing an object physically inserted in a scene [12, 7, 44, 51]. But none of these patterns are applicable to 3D PCs, for which "pixels" are undefined. *Challenge 2*: Designing a backdoor pattern learnable by 3D PC classifiers is difficult since they extract different features than image classifiers, especially convolutional neural networks like [20, 17]. *Challenge 3*: The backdoor pattern should be robust to test-time preprocessing of 3D PCs like random sampling, should be evasive of anomaly detectors (ADs), and should be scene-plausible.

In this paper, we propose to insert a small cluster of points as the backdoor pattern (for Challenge 1), dubbed "backdoor points", which can be implemented either digitally (to mimic, e.g., spurious points caused by vehicle exhaust), or physically using an object (e.g. a ball) captured along with the scene by the 3D sensor. The spatial location of the backdoor cluster is optimized by making use of a surrogate classifier that is *independently* trained by the attacker, using its own (separate) data set (for Challenge 2). Such optimization is necessary to ensure that the victim classifier learns the backdoor pattern during its training. The local geometry of the actual backdoor points embedded in each PC sample is also optimized, such that these points have similar local density as the original points in the PC (for Challenge 3). Our contributions are summarized as follows:

- We propose the first BA against 3D PC classifiers. Unlike PC TTE attacks, we *do not* use any knowledge of the victim classifier or of the clean data possessed by the trainer.
- We propose "backdoor points" customized for 3D PCs, along with approaches for optimizing their spatial location and local geometry.
- We show the effectiveness of our BA for four different types of backdoor point local geometries, three different architectures for the victim classifier, and on two datasets.
- We show through experiments that the effectiveness of our BA mostly depends on the spatial location of the backdoor points, while careful design of their local geometry helps the BA evade the state-of-the-art PC ADs.

2. Related Work

2.1. 3D Point Cloud Classification

A 3D point cloud (PC) is a set of 3D points commonly captured by 3D sensors including radio detection and ranging (RADAR) [38], light detection and ranging (LiDAR) [61], and ultrasonic sensors [19]. Techniques for 3D PC classification have rapidly developed due to the increasing popularity of 3D sensors in many applications like autonomous driving [6]. Early approaches include 3D convolutional neural networks, e.g. VoxNet [26], which represents 3D PCs using a series of voxels for classification. Multi-view based methods combine features associated with different views of an object into a global descriptor [39, 40]. PointNet [4] is the pioneering method directly taking a 3D PC as input and achieving permutation invariance of points by using a symmetric function – max pooling. Due to the simplicity and strong representation capability of PointNet, it is used as the backbone of many 3D learning modules [6], and is also the basis for many subsequent methods, e.g. [37, 46, 63, 60]. Like existing PC TTE attacks, we focus on PointNet and its variants in this paper.

2.2. Adversarial Attacks against 3D PC Classifiers

Typical adversarial attacks against classifiers include test-time evasion (TTE) attacks, general data poisoning (DP) attacks [2, 18], and BAs, which are the focus of this paper. Existing adversarial attacks against 3D PC classifiers are all TTE attacks, which were originally proposed against image classifiers. Image TTE attacks aim to "fool" a victim classifier (i.e. have it classify incorrectly) by introducing a human-imperceptible perturbation to a test image [41, 11, 35, 33, 25, 3, 24]. Such perturbations can be learned using knowledge of the victim classifier, including its architecture and parameters[1], or *transferred* from an independently trained surrogate classifier, i.e. learned using knowledge of the surrogate classifier [34, 36]. Existing PC TTE attacks "fool" a victim classifier by adding points to a test PC, perturbing its points, or removing some of its points [49, 47, 22, 43] – these operations are the analogue, for PCs, of TTE perturbations applied to 2D images. However, PC TTE attacks do not transfer nearly as well as image TTE attacks. Even for two classifiers trained on the same dataset, with the same architecture but different parameter initializations, test PCs generated using one classifier do not reliably "fool" the other [49, 22]. Such poor transferability

[1] TTE perturbations can also be created by querying the victim classifier [33], though this method has not been extended to PCs yet. Also, frequent queries may be denied due to the security protocol of the victim classifier.

may be due to the larger discrepancy of the decision boundaries between two PC classifiers. Especially for PointNet and its variants, the "critical points" selected from a PC by max pooling may be very different for two classifiers; thus perturbing a critical point selected by classifier A cannot "fool" classifier B if it is "dropped out" (i.e. not selected by max pooling) by classifier B. Hamdi et al. [15] improve the transferability of PC TTE attacks; but the success rate to "fool" the victim classifier using their transferred attack is still less than 65%, even with the victim classifier's training set exploited by the attacker. In summary, the effectiveness of existing PC TTE attacks largely relies on knowledge of the victim classifier, which is usually not available in practice. By contrast, our backdoor attack does not require such knowledge, nor does it need access to the clean training set.

2.3. Backdoor Attacks against Image Classifiers

A backdoor attack (BA) is a type of adversarial attack initially proposed against DNN image classifiers [7, 12, 21]. A source class, a target class, and a backdoor pattern (a.k.a. a backdoor trigger) are the three elements of a BA, which are all specified by the attacker. An image BA aims to: 1) have the victim classifier learn to classify to the target class, whenever any test image from the source class is embedded with the backdoor pattern; 2) not degrade the accuracy of the victim classifier on clean, backdoor-free test images. An image BA can be launched by poisoning the training set of the victim classifier with a small set of backdoor training images that are originally from the source class, embedded with the same backdoor pattern, and labeled to the target class. For image BAs, existing backdoor patterns include a human-imperceptible, additive perturbation [7, 65, 45, 53], and a pixel patch replacement representing an object physically inserted in a scene [12, 7, 44, 51]. However, none of these backdoor patterns or embedding mechanisms are applicable to 3D PCs. Designing a suitable backdoor pattern for 3D PCs is thus the main challenge for devising a PC BA.

3. Backdoor Attacks against 3D PC Classifiers: Scenario, Goals and Assumptions

We consider a common practical scenario, wherein a training authority learns a 3D PC classifier using a dataset collected from the public. Unfortunately, among the data donors, there is an attacker who aims to embed a backdoor mapping in the classifier. Thus, the training set of the victim classifier is $\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{clean}} \cup \mathcal{B}$, where $\mathcal{D}_{\text{clean}}$ contains clean, labeled training samples and $\mathcal{B}$ denotes the set of backdoor training samples contributed by the attacker. Unaware of the attack, the trainer performs regular learning on $\mathcal{D}_{\text{train}}$, usually by solving:

$$\underset{\Theta}{\text{minimize}} \sum_{(\mathbf{X},y)\in\mathcal{D}_{\text{train}}} L(f_{\text{v}}(\mathbf{X};\Theta),y) \qquad (1)$$

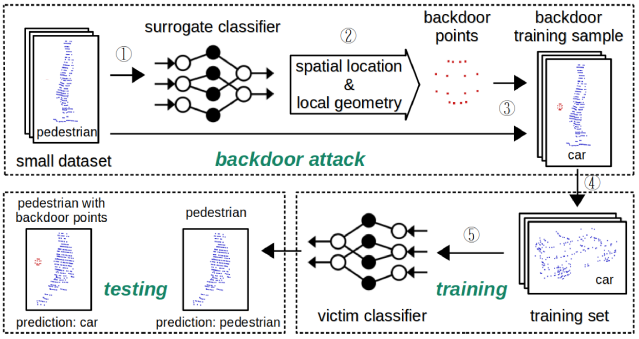


Figure 2: Outline of our BA. The attacker collects a small dataset to train a surrogate classifier (①). The backdoor points is generated using the surrogate classifier with optimized spatial location (Sec. 4.2) and local geometry (Sec. 4.1) (②). The backdoor points is embedded in clean PCs from a source class, e.g. "pedestrian" (③), to generate backdoor training samples labeled to a target class, e.g. "car". These samples are used to poison the training set possessed by the trainer (④), on which the victim classifier is trained (⑤). During testing, the victim classifier is supposed to classify source class PCs embedded with the backdoor points to the target class (Eq. (2)), while correctly classifying backdoor free test PCs (Eq. (3)).

e.g. via stochastic (mini-batch) gradient descent (SGD) [10]. Here, $\mathbf{X} = \{\mathbf{x}_i \in \mathbb{R}^3 | i = 1, \cdots, n\} \in \mathcal{X}$ denotes a PC, where $\mathbf{x}_i$ is a point with (x, y, z) coordinates[2]. $\mathcal{X}$ and $\mathcal{Y}$ denote the 3D PC space and the label space, respectively. The loss function $L(\cdot,\cdot) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, the architecture of the victim classifier $f_{\text{v}}(\cdot;\Theta) : \mathcal{X} \to \mathcal{Y}$ (with Θ the classifier's parameters), and other training settings are all specified by the trainer independent of the presence of the BA.

The attacker's goals are two-fold. a) Having the classifier learn the *"backdoor mapping"*. I.e., for any test PC from a prescribed source class $s \in \mathcal{Y}$, the trained classifier should classify to the attacker's target class $t \in \mathcal{Y}$ ($t \neq s$) whenever the test PC is embedded with the attacker's backdoor pattern $\mathbf{V}$. Formally, the attacker aims to maximize:

$$\mathbb{E}_{\mathbf{X}\sim P_s}[\mathbb{1}(f_{\text{v}}(m(\mathbf{X};\mathbf{V});\Theta) = t)], \qquad (2)$$

where P_s is the distribution of PCs from class s, $\mathbb{1}(\cdot)$ is a logical indicator function. $m(\cdot;\mathbf{V}) : \mathcal{X} \to \mathcal{X}$ is the embedding function associated with the backdoor pattern $\mathbf{V}$ – its design requires a surrogate classifier independently trained by the attacker on a small dataset (details in Sec. 4). b) *Not* degrading the accuracy of the trained classifier on clean test PCs. Formally, the attacker aims to maximize:

$$\mathbb{E}_{\mathbf{X}\sim P_y}[\mathbb{1}(f_{\text{v}}(\mathbf{X};\Theta) = y)], \quad \forall y \in \mathcal{Y}, \qquad (3)$$

where P_y is the sample distribution for class y. This is different from the goal of traditional DP attacks, which aim to

[2]General PCs may involve higher-dimensional point representations with additional features beyond 3D coordinates for each point [63].

degrade the accuracy of the classifier. The motivation for b) is so that validation set accuracy degradation, e.g. [30], *cannot* be reliably used to detect BAs.

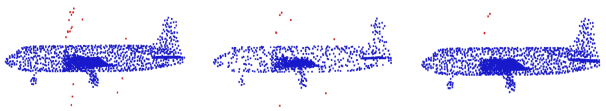
To achieve these two goals, similar to image BAs, a set of backdoor training samples $\mathcal{B} = \{(m(\mathbf{X}; \mathbf{V}), t) | \mathbf{X} \sim P_s\}$ is used to poison the training set. Then, (2) and (3) are jointly and automatically maximized when the trainer solves (1), where the loss function is a differentiable surrogate of the indicator function in (2) and (3). The outline of our BA is shown in Fig. 2. The scenario for our BA is as follows:
1) The attacker has *no access* to the training process, including knowledge of the victim classifier's architecture, the loss function, and other training configurations.
2) The attacker has *no access* to $\mathcal{D}_{\text{clean}}$, the clean training data collected by the trainer from other (benign) sources.
3) The attacker is able to collect data independently (to train a surrogate classifier and create backdoor training samples). This collected data is assumed i.i.d. with $\mathcal{D}_{\text{clean}}$.
4) The attacker has the capability to contribute its data to the training set of the victim classifier.
The first two assumptions are consistent with the role of a backdoor attacker, who is merely one of the data donors. These two assumptions make our BA more practical than existing PC TTE attacks, which rely on knowledge of the victim classifier. The third and the fourth assumptions are the basis of image BAs and traditional DP attacks – the classifier can be more adequately trained by collecting data from as many sources as possible, among which there may be an attacker. Note that *these assumptions are strictly followed during experimental evaluation of our BA*.

4. Backdoor Points

The key to our PC BA is the design of the backdoor pattern and the associated embedding function. Due to the irregularity of 3D PCs, and inspired by PC TTE attacks [49, 64], candidate backdoor embedding mechanisms include adding points, dropping points, and perturbing points. Here, we choose to add/insert a small cluster of points as the backdoor pattern, for two reasons. First, in practice, a set of inserted points can potentially be implemented physically by placing an object, e.g. a ball, in the scene, captured by a 3D sensor; or, these points can be digitally inserted into a PC to *mimic* an object or a cluster of spurious points (which are usually caused by vehicle exhaust in the context of autonomous driving). Second, an ideal backdoor pattern is a *common* pattern; but point dropping and point perturbations are a function of the original points – it is thus difficult to create a common backdoor pattern using these mechanisms. Formally, the *backdoor embedding function* is defined as:

$$m(\mathbf{X}; \mathbf{V}) = \mathbf{X} \cup \mathbf{V}, \tag{4}$$

where the backdoor pattern $\mathbf{V}$, dubbed *"backdoor points"*, is defined as:



(a) prior-processing (b) random sampling (c) outlier removal

Figure 3: Preprocessing and anomaly detection of test PCs. (a) A PC with randomly inserted points (in red). (b) PC undergoes random sampling (with half the points removed). (c) PC undergoes the point AD in [66] which removes outlier points – most of the inserted points are removed.

$$\mathbf{V} = \{\mathbf{u}_i + \mathbf{c} | \mathbf{u}_i \in \mathbb{R}^3, \mathbf{c} \in \mathbb{R}^3, i = 1, \cdots, n'\}. \tag{5}$$

Note that $\mathbf{V}$ is jointly determined by its *local geometry* $\mathbf{U} = \{\mathbf{u}_i \in \mathbb{R}^3 | i = 1, \cdots, n'\}$ and its *spatial location* $\mathbf{c}$ – how to specify these two elements is discussed next.

4.1. Local Geometry of Backdoor Points

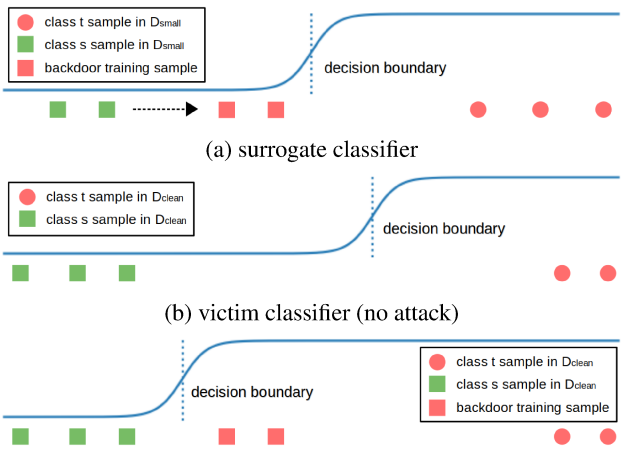
Ideally, the embedded backdoor points should have the same local geometry for all backdoor training/test PCs. However, this is not feasible for BAs physically implemented using even the same object – the actual points associated with the object captured by a 3D sensor are likely different from PC to PC. Fortunately, local geometry of backdoor points is less critical than its spatial location for the victim classifier to learn the backdoor mapping (Eq. (2)), as will be empirically shown by our substantial experiments in Sec. 5.4. Here, we allow backdoor points embedded in each PC to have slightly *different* local geometry.

For practical consideration, the design of backdoor points' local geometry mainly addresses its *robustness* to possible PC preprocessing, e.g. point sub-sampling, and PC anomaly detectors (ADs) deployed during testing. As shown in Fig. 3, point sub-sampling keeps a subset of points for classification; thus, part of the inserted backdoor points will be inevitably removed. A PC AD, e.g. [66], removes outlier points with abnormal local density. Accordingly, the backdoor points should: a) contain a sufficient number of points; b) have a similar local point density as the PC into which they are embedded. For BAs implemented physically using an object, criterion a) can be achieved if the object is sufficiently large. Criteria b) is automatically achieved due to the usually stable scanning frequency of 3D sensors.

For digitally implemented BAs, backdoor points' local geometry $\mathbf{U}$ can be specified by the attacker by defining a suitable stochastic point generator. For example, in one of our experiments, to mimic a physically implemented BA using a ball, we generate backdoor points randomly located on a sphere with some radius r using the random generator:

$$\mathbf{g}(r, \boldsymbol{\theta}, \boldsymbol{\phi}) = [r \sin \boldsymbol{\theta} \cos \boldsymbol{\phi}, r \sin \boldsymbol{\theta} \sin \boldsymbol{\phi}, r \cos \boldsymbol{\theta}]^T, \tag{6}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are random variables uniformly distributed in $[0, \pi]$ and $[0, 2\pi]$ respectively and r is a parameter to be specified. Regardless of the generator's form, criterion a) can be achieved by generating a sufficient number of points.



(a) surrogate classifier

(b) victim classifier (no attack)

(c) victim classifier (attacked)

Figure 4: An intuitive illustration of class t posterior probability for: (a) The surrogate classifier trained on $\mathcal{D}_{\text{small}}$. Clean samples from class s are "pushed" towards class t with backdoor points embedding, and are labeled to class t. (b) The victim classifier without backdoor poisoning (trained on $\mathcal{D}_{\text{clean}}$). (c) The victim classifier trained on the backdoor poisoned training set $\mathcal{D}_{\text{train}}$ – the backdoor training samples influence the learned decision boundary.

For criterion b), we propose to optimize (over the parameters of the generator) the distribution of the local density of all points in $\mathbf{U}$ by a novel approach based on median absolute deviation (MAD), a robust measure of variability ([16]). Inspired by [66], we measure the local density of a point using its kNN distance. Then, the median kNN distance of a PC $\mathbf{X} \in \mathcal{X}$ for backdoor point embedding is:

$$D_{\text{knn}}(\mathbf{X}) = \underset{i\in\{1,\cdots,n\}}{\text{median}} \frac{1}{k} \sum_{\mathbf{x}_j \in \mathcal{S}(\mathbf{x}_i,k)} ||\mathbf{x}_i - \mathbf{x}_j||_2, \quad (7)$$

where $\mathcal{S}(\mathbf{x}_i, k)$ contains k nearest neighbors of $\mathbf{x}_i$. For the same example of generating random points on a sphere with radius r, for each PC $\mathbf{X}$ for embedding, we find the optimal radius r by solving:

$$\begin{aligned} \min_{r>0} \quad & \mathbb{E}_{\boldsymbol{\theta},\boldsymbol{\phi}}\Big[\underset{i\in\{1,\cdots,n'\}}{\text{median}} \Big|D_{\text{knn}}(\mathbf{X}) - \frac{1}{k} \sum_{\mathbf{u}_j \in \mathcal{S}(\mathbf{u}_i,k)} ||\mathbf{u}_i - \mathbf{u}_j||_2\Big|\Big] \\ \text{s. t.} \quad & \mathbf{u}_i = \mathbf{g}(r, \boldsymbol{\theta}, \boldsymbol{\phi}), \quad \forall i \in \{1, \cdots, n'\}, \end{aligned} \quad (8)$$

We practically solve (8) via a grid search. Note that for other geometries e.g. a cube, a cluster of points mimicking spurious points, etc., a different generator function would be chosen, possibly with different parameters to be optimized.

4.2. Spatial Location of Backdoor Points

Given the local geometry $\mathbf{U}$ fixed, the spatial location $\mathbf{c}$ should be specified following two criteria. **C1**: The backdoor mapping (Eq. (2)) should be well learned by the victim classifier. **C2**: The backdoor points should be spatially *close* to the PC into which they are embedded, so that, in practice, the inserted backdoor object can be captured along with the object associated with the PC (i.e., in the same bounding box) by a 3D sensor.

Empirically, a BA with randomly located backdoor points is not guaranteed to be successful, as will be shown in Sec. 5.5. Thus, our attacker *optimizes* the spatial location $\mathbf{c}$ using a surrogate classifier $f(\cdot;\Phi) : \mathcal{X} \to \mathcal{Y}$ independently trained on a small dataset $\mathcal{D}_{\text{small}}$, with Φ the classifier's parameters. If there is no BA, the landscape of the posterior probability function associated with the target class will likely be similar between the victim classifier (trained on $\mathcal{D}_{\text{clean}}$ only) and the surrogate classifier. This is due to the fact that $\mathcal{D}_{\text{clean}}$ and $\mathcal{D}_{\text{small}}$ are generated i.i.d. according to the same distribution. In other words, for both classifiers, the target class posterior probability will likely be large for a *typical* target class PC, and be small for a *typical* source class PC. However, there is no guarantee for the two classifiers to have the same (or a very similar) decision boundary between the source class and the target class. This intuition is jointly illustrated in Fig. 4a and Fig. 4b.

The purpose of backdoor poisoning is to have the victim classifier learn to classify backdoor training samples to the target class (i.e. **C1**) – target class posterior probability for these PCs should also be large after the victim classifier's training on the poisoned training set. Thus, we optimize the spatial location $\mathbf{c}$ such that the embedding of backdoor points "pushes" the backdoor training PCs toward typical target class PCs. A simple illustration of the *expected* landscape of the target class posterior probability function (and the learned decision boundary) for the victim classifier being attacked is shown in Fig. 4c. For this classifier, a typical source class test PC embedded with similar backdoor points will also have a large target class posterior probability.

Formally, we denote the *surrogate classifier's* posterior probability function for the target class t as $p(t|\cdot,\Phi) : \mathcal{X} \to [0,1]$, with Φ the classifier's parameters. For a non-target class PC, $p(t|\cdot,\Phi)$ is supposed to increase when it is "pushed" towards the target class t. Thus, considering also **C2**, we find the minimum average distance from $\mathbf{c}$ to the source class PCs, such that any point[3] inserted at spatial location $\mathbf{c}$ induces these source class PCs to have at least a certain level of average posterior probability for class t, i.e.:

$$\begin{aligned} \min_{\mathbf{c}\in\mathbb{R}^3} \quad & \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{X},y)\in\mathcal{D}_s} d(\mathbf{c}, \mathbf{X}) \\ \text{s. t.} \quad & \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{X},y)\in\mathcal{D}_s} p(t|m(\mathbf{X};\mathbf{V}),\Phi) \geq \epsilon_0 + \epsilon, \end{aligned} \quad (9)$$

where $\mathcal{D}_s \subset \mathcal{D}_{\text{small}}$ is the subset of samples from class s possessed by the attacker. $d(\mathbf{c}, \mathbf{X})$ measures the distance from point $\mathbf{c}$ to PC $\mathbf{X} \in \mathcal{X}$. In our experiments, we use

[3] One can append $\mathbf{V}$ with arbitrary local geometry or even a single point at $\mathbf{c}$ to $\mathbf{X}$ when solving (9).

$d(\mathbf{c}, \mathbf{X}) = \min_{\mathbf{x} \in \mathbf{X}} ||\mathbf{c} - \mathbf{x}||_2$ for its simplicity and piecewise differentiability in $\mathbf{c}$. $\epsilon_0 = \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{X}, y) \in \mathcal{D}_s} p(t|\mathbf{X}, \Phi)$ is the initial "soft" class confusion from class s to class t, which is usually close to zero due to the inevitable overfitting on $\mathcal{D}_{\text{small}}$ during the surrogate classifier's training. Finally, ϵ is a small positive number describing how close the source class PCs should be "pushed" toward class t by inserting a point at $\mathbf{c}$. Unlike the image domain, where a small, common perturbation can induce a group of images from one class to be misclassified to another class [29], the feasible set of (9) for even a moderately large ϵ may contain only spatial locations far apart from the original PCs in $\mathcal{D}_s$, which violates **C2**. Thus, in practice, ϵ is chosen to ensure that there is at least one solution with sufficiently small objective value for (9) (e.g. $\epsilon = 0.02$ in our experiments).

We solve (9) using Alg. 1, where

$$J(\mathbf{c}, \lambda) = \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{X}, y) \in \mathcal{D}_s} \big[\lambda \cdot d(\mathbf{c}, \mathbf{X}) - \log p(t|m(\mathbf{X}; \{\mathbf{c}\}), \Phi)\big] \tag{10}$$

is the Lagrangian of (9), with the logarithm used for better smoothness. λ is updated automatically (using a scaling factor $\alpha > 1$) to constrain the optimization variables in the feasible set (as an alternative to projection which is hard to realize here). $\mathcal{N}(\mathbf{0}, \mathbf{I})$ is a standard normal distribution used to initialize $\mathbf{c}$ – the PCs are usually aligned to the origin for classification [4]. To avoid poor local optima, one can perform Alg. 1 multiple times, with different initialization, and pick the best solution to (9).

Algorithm 1 Optimal spatial location for backdoor points.

Inputs: source class s, target class t, data subset $\mathcal{D}_s$, surrogate classifier $f(\cdot; \Phi)$, ϵ and ϵ_0, step size δ, maximum iteration count $\tau_{\max}$, scaling factor α.
Initialization: $\mathbf{c}^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\lambda^{(0)}$ set to a small positive number (e.g. 10^{-5}), $\mathbf{c}^* = \boldsymbol{\infty}$, $\rho^{(0)} = 0$.
for $\tau = 0 : \tau_{\max} - 1$:
  $\mathbf{c}^{(\tau+1)} = \mathbf{c}^{(\tau)} - \delta \nabla_{\mathbf{c}} J(\mathbf{c}^{(\tau)}, \lambda^{(\tau)})$
  $\rho^{(\tau+1)} = \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{X}, y) \in \mathcal{D}_s} p(t|m(\mathbf{X}; \{\mathbf{c}^{(\tau+1)}\}), \Phi)$
  if $\rho^{(\tau+1)} \geq \epsilon_0 + \epsilon$:
    $\lambda^{(\tau+1)} = \lambda^{(\tau)} \cdot \alpha$
    if $\sum_{(\mathbf{X}, y) \in \mathcal{D}_s} \big[d(\mathbf{c}^{(\tau+1)}, \mathbf{X}) - d(\mathbf{c}^*, \mathbf{X})\big] < 0$:
      $\mathbf{c}^* = \mathbf{c}^{(\tau+1)}$
  else:
    $\lambda^{(\tau+1)} = \lambda^{(\tau)} / \alpha$
Outputs: $\mathbf{c}^*$

5. Experiments

5.1. Datasets

Like existing PC TTE attacks [49, 15, 22], we use the aligned benchmark dataset ModelNet40 [48] for our experiment. ModelNet40 contains 12311 CAD models (2048 points for each PC) from 40 common object categories. Following the original train-test split of ModelNet40, 2468 PCs are used for testing. From the remaining 9843 PCs, we randomly choose 1000 PCs as the "small dataset" ($\mathcal{D}_{\text{small}}$) possessed by the attacker. The remaining 8843 PCs are possessed by the trainer ($\mathcal{D}_{\text{clean}}$) and are not accessible to the attacker. Additionally, we consider a practical street view LiDAR dataset KITTI [27]. From each scene, we extract PCs corresponding to labeled objects inside their bounding boxes provided with the dataset and align them. Due to high class imbalance of the original KITTI dataset, we construct two (super) classes: a "vehicle" class consists of "car", "van", and "truck" from the original dataset; a "human" class consists of "pedestrian" and "cyclist" from the original dataset. We consider PCs with no less than 256 points and randomly keep 256 points for each PC. Also, we keep a subset of PCs for the "vehicle" class such that the two classes have equal number of samples. Consequently, we obtain 2662 PCs evenly distributed in the two classes – 200 are possessed by the attacker, 1800 are possessed by the trainer, and 662 are used for testing.

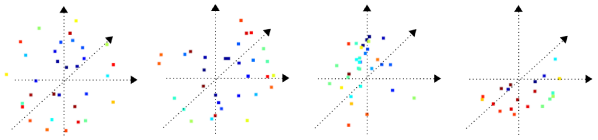
5.2. Attack Implementation

We implemented 36 attacks involving 9 (source, target) class pairs in total for the two datasets – for each class pair, we create 4 attacks with different types of local geometry for the embedded backdoor points.

Specify source and target classes: For ModelNet40, we arbitrarily chose 7 (source, target) class pairs, which are: (chair, toilet), (vase, curtain), (laptop, chair), (nigh stand, table), (sofa, monitor), (cone, lamp), (airplane, wardrobe). For KITTI, we consider the only two ordered class pairs: (human, vehicle) and (vehicle, human). We name these 9 class pairs as $\mathbf{P_1}$, $\mathbf{P_2}$, ..., $\mathbf{P_9}$ respectively for brevity.

Train a surrogate classifier: For each dataset, we trained a PointNet with the same architecture in [4] on the PCs possessed by the attacker. Training was performed for 250 epochs with batch size 32 and learning rate 10^{-3} (with 0.5 decay per 20 epochs). 2048 points and 256 points per PC are used for ModelNet40 and KITTI, respectively.

Specify the spatial location of backdoor points: For the four attacks associated with each (source, target) class pair, we specified one *common* spatial location for backdoor point embedding using Alg. 1 and the surrogate classifier trained on its associated dataset. The parameters for the attacker's optimization were set to $\epsilon = 0.02$, $\delta = 0.01$, $\tau_{\max} = 3000$, $\alpha = 1.5$. In particular, although ϵ is numerically small, there is already a moderate distance between the optimal spatial location (solution to (9)) and the PCs used for backdoor embedding, as shown in Apdx. A. Larger ϵ may cause the embedded backdoor points to be too far from the PC to be captured in the same bounding box by a 3D



(a) GS (b) RS (c) RP (d) HS

Figure 5: Illustration of the four types of local geometry. GS is a non-optimized geometry; while RS, RP, and HS are optimized geometries with stochastic generators.

sensor. The choices of the other three parameters are not critical to the performance of our BA.

Specifying the local geometry of backdoor points: For each class pair, we created four attacks with the following four different types of local geometry respectively. We set $k = 4$ in Eq. (7) and (8) for local geometry optimization. Examples of these local geometries are shown in Fig. 5.

1) **GS**: 32 points uniformly spaced on a sphere, generated by Eq. (6) with deterministic $\theta \in \{\frac{1}{8}\pi, \frac{3}{8}\pi, \frac{5}{8}\pi, \frac{7}{8}\pi\}$, and $\phi \in \{\frac{1}{8}\pi, \frac{3}{8}\pi, \cdots, \frac{15}{8}\pi\}$. Radius is manually set to $r = 0.04$ for scene-plausibility, but *without* optimizing (8).

2) **RS**: 32 points randomly distributed on a sphere generated by Eq. (6), with $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ uniformly sampled from $[0, \pi]$ and $[0, 2\pi]$ respectively. Radius r is obtained by solving (8).

3) **RP**: 32 points randomly distributed in a ball generated in the same way as RS, except that r is now a random variable uniformly distributed in $[0, r_{\max}]$, where $r_{\max}$ is optimized instead of r in (8).

4) **HS**: Points randomly distributed on a half sphere with random orientation (to mimic a surface of a ball facing a 3D scanner) – generated from RS by keeping points having positive inner product with a random vector.

Create backdoor training samples: For each attack, with the specified spatial location and local geometry, we generated backdoor training samples using a subset of clean PCs possessed by the attacker from the source class, following Eq. (5) and (4). Example backdoor training samples are shown in Apdx. A. For ModelNet40 and KITTI, 15 and 30 backdoor training samples are generated for poisoning the training set, respectively.

5.3. Training

Learning the victim classifier is performed by the trainer, on the poisoned training set $\mathcal{D}_{\text{train}}$. Based on the assumptions in Sec. 3, the entire training process is not accessible to the attacker. Like PC TTE attacks ([49, 47]), we consider three DNN architectures for the victim classifier – PointNet [4], PointNet++ [37], and DGCNN [46]. We use the same DNN architecture and training protocol for these models as described in their original papers. Notably, for ModelNet40, each PC is *preprocessed* by randomly sampling 1024 points before feeding to the classifier *(both during training and test)*. Similarly, 128 points are randomly chosen to remain for each PC for KITTI. As a benchmark, without poisoning, the test accuracy of the trained PointNet, PointNet++, and DGCNN are 88.5%, 91.5%, and 91.4% for ModeNet40; 99.5%, 99.7%, and 99.7% for KITTI.

		ModeNet40				KITTI			
		ASR (avg)	ASR (min)	ACC (avg)	ACC (min)	ASR (avg)	ASR (min)	ACC (avg)	ACC (min)
PointNet [4]	GS	94.0	91.9	88.7	88.2	92.8	89.1	99.3	99.2
	RS	96.0	93.0	88.7	88.2	93.4	87.3	99.4	99.4
	RP	94.9	90.0	88.6	87.8	94.0	90.9	99.4	99.1
	HS	96.0	93.0	88.6	88.2	91.2	91.2	99.5	99.5
PointNet++ [37]	GS	94.6	89.5	91.4	91.0	95.9	92.7	99.5	99.5
	RS	96.9	92.0	91.0	90.2	93.1	87.6	99.4	99.4
	RP	96.9	95.0	91.0	90.2	93.5	89.7	99.7	99.5
	HS	93.7	88.0	91.4	91.1	88.6	87.6	99.5	99.5
DGCNN [46]	GS	93.2	90.0	92.9	90.8	96.7	95.5	99.5	99.5
	RS	93.9	87.0	91.1	90.7	95.0	91.5	99.8	99.7
	RP	96.1	90.0	91.0	90.6	96.4	93.1	99.6	99.4
	HS	93.7	87.0	91.0	90.8	92.8	90.6	99.5	99.4

Table 1: Average and minimum ASR and ACC (in %), respectively, over the 9 attacks (for class pairs P1, P2, ..., P9), for the 4 local geometries (GS, RS, RP, and HS), the 2 datasets (ModelNet40 and KITTI), and the three victim classifier architectures (PointNet, PointNet++, and DGCNN). All attacks are successful with ASR $\geq 87\%$.

5.4. Performance Evaluation (Main Results)

The performance of our BA is evaluated using the test set and the following two metrics for each attack we created:

1) Attack success rate (**ASR**): For each test PC from the source class, we embed backdoor points with the same type of local geometry and spatial location as used to create the backdoor training samples. ASR is defined as the percentage of misclassifications to the target class.

2) Clean test accuracy (**ACC**): The accuracy of victim classifier on the clean test PCs from all classes.

Based on the attacker's goals in Sec. 3, a successful BA should have a high ASR and negligible degradation in ACC compared with the clean benchmarks in Sec. 5.3. Thus, **all 36 attacks are successful** (with all ASRs $\geq 87\%$) regardless of the victim classifier's architecture, as shown in Tab. 1 (ASR and ACC for each attack are shown in Apdx. B). Apart from that, we observe that for each class pair, **with the same optimal spatial location, the choice of the local geometry does not significantly affect the learning of the backdoor mapping**. Especially for attacks with geometry RP, the backdoor points inserted to each PC have high randomness; but the ASR for these attacks are still uniformly high. For physically implemented BAs, this property allows more freedom in choosing the geometry of the inserted object to achieve scene-plausibility. Also, since high ASRs are achieved when each test PC is sub-sampled to 1024 points – nearly half of the points are removed – **our BA is robust to test-time sub-sampling.** Moreover, in Fig. 6, we show ASR curves for the three attacks with local geometry RP for class pairs P1, P2, and P3, over a range of number of backdoor training samples used for poisoning the victim classifier's training set. Our BA is effective, with only a few backdoor training samples inserted in the training set containing 8843 clean PCs; thus it is also very stealthy.

Additionally, we compare our BA with PC TTE attacks

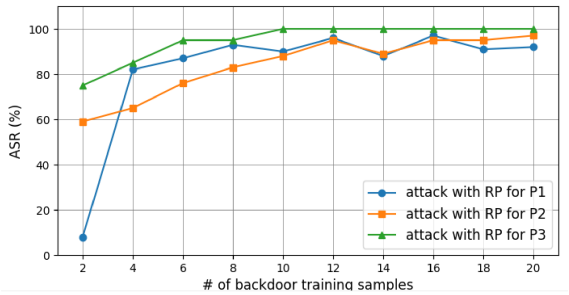


Figure 6: ASR versus number of backdoor training samples for attacks with local geometry RP associated with class pairs P1, P2, and P3 (for example). With merely 8 backdoor training samples, all three attacks achieve ASR $> 80\%$.

	P1	P2	P3	P4	P5	P6	P7	P8	P9
PointNet	11.4	32.0	10.5	25.0	44.7	45.5	39.4	18.2	28.5
PointNet++	0	45.0	0	16.7	1.1	18.2	0	1.3	0
DGCNN	12.4	40.5	38.9	20.8	7.2	27.3	11.6	0	0.3

Table 2: Success rate of targeted PC TTE attacks (for class pairs P1-P9) transferred from the surrogate classifier, for victim classifier architectures PointNet, PointNet++, and DGCNN – PC TTE attacks transfer poorly.

implemented by point addition in the same scenario described in Sec. 3. Following [49], for each of the 9 class pairs, we created adversarial PCs by inserting 32 points to test PCs from the source class. The locations for the inserted points are optimized using the *surrogate classifier* such that the adversarial PCs are classified to the target class by the surrogate classifier. As shown in Tab. 2, these adversarial PCs cannot reliably "fool" the victim classifier trained on clean PCs possessed by the trainer – PC TTEs transfer poorly; thus, they are less threatening than our BA in cases where the victim classifier is not accessible to the attacker.

5.5. Backdoor Points with Random Spatial Location

Here, we show the **necessity of spatial location optimization** for our BA. For class pair P1 and local geometry GS, we created 50 attacks in the same way as described in Sec. 5.2, but *without* spatial location optimization. In particular, for each attack, we pick a random spatial location $\mathbf{c} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and scale it such that the average distance from the scaled $\mathbf{c}$ to the source class PCs (i.e. objective of (9)) is the same as for the optimized spatial location obtained for the attack associated with P1 and GS. As shown in Fig. 7, all 50 attacks (with maximum ASR 91.0%) have smaller ASR than the attack with the optimized spatial location (with ASR 94.0%, shown in Tab. 4 Apdx. B). Moreover, some of the 50 attacks are not reliable, with low ASR.

5.6. BA against PC Anomaly Detectors (ADs)

Existing state-of-the-art detectors for image BAs, e.g. [44, 14], highly depend on the format of the backdoor pattern; hence they are not applicable to our PC BA (more details are in Apdx. D). Still, we consider the state-of-the-art

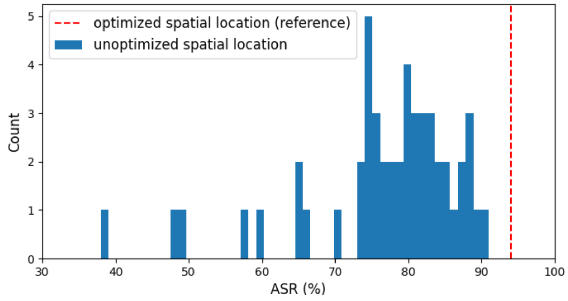


Figure 7: Histogram of ASR for 50 attacks created without spatial location optimization – most of them are clearly outperformed by our BA with optimized spatial location.

	GS	RS	RP	HS
P1	49.0 (94.0)	90.0 (93.0)	88.0 (94.0)	81.0 (93.0)
P2	9.0 (93.0)	97.0 (98.0)	96.0 (96.0)	87.0 (97.0)
P3	51.0 (95.0)	100 (100)	100 (100)	90.0 (100)
P4	8.1 (91.9)	94.0 (95.0)	95.3 (96.5)	81.4 (95.3)
P5	2.0 (95.0)	90.0 (95.0)	87.0 (90.0)	84.0 (93.0)
P6	35.0 (95.0)	90.0 (95.0)	90.0 (90.0)	95.0 (100)
P7	63.0 (94.0)	94.0 (96.0)	98.0 (98.0)	88.0 (94.0)
P8	97.0 (96.4)	99.7 (99.4)	98.8 (97.0)	92.4 (91.2)
P9	87.9 (89.1)	85.2 (87.3)	90.9 (90.9)	90.6 (91.2)

Table 3: Attack success rate (ASR) (in %) for the 36 attacks for victim classifier architecture PointNet, when the PC AD in [66] is deployed during test. ASRs (in %) without AD deployed are shown in parenthesis for reference.

defense against PC TTE attacks – a PC AD in [66], which aims to remove points inserted/perturbed by a TTE attacker. It measures the kNN distance (with $k = 2$) for each point in a PC and removes points with abnormally high or low kNN distance (falling outside of ± 1.1 standard deviation interval around the average). In Tab. 3, we show ASR of the 36 attacks for victim classifier being a PointNet, when the above PC AD is deployed during testing. For brevity, results associated with PointNet++ and DGCNN are deferred to Apdx. C. For the non-optimized geometry GS, most attacks are no longer reliable because the backdoor points embedded in many test PCs are entirely removed. For the three optimized geometries (RS, RP, and HS), the PC AD only causes limited degradation in ASR compared with the no detector case. There is still a 81.0% minimum ASR for the 27 attacks for these three local geometries.

6. Conclusions

We propose the first BA against 3D PC classifiers. Our BA is devised by inserting a small cluster of points with optimized spatial location and local geometry. Spatial location optimization helps the backdoor mapping to be learned; while local geometry optimization makes the inserted points robust to possible point preprocessing and helps our BA evade possible defenses like a PC AD. One future research direction is to extend sample-specific BAs ([31, 59]) to PCs.

References

[1] A. Shafahi and W. R. Huang and M. Najibi and O. Suciu and C. Studer and T. Dumitras and T. Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6106–6116, 2018.

[2] B. Biggio and F. Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[3] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, May 2017.

[4] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[5] B. Chen, W. Carvalho, N. Baracaldo, H. Ludwig, B. Edwards, T. Lee, I. Molloy, and B. Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. http://arxiv.org/abs/1811.03728, Nov 2018.

[6] S. Chen, B. Liu, C. Feng, C. Vallespi-Gonzalez, and C. Wellington. 3d point cloud processing and learning for autonomous driving: Impacting map creation, localization, and perception. *IEEE Signal Processing Magazine*, 38(1):68–86, 2021.

[7] X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. https://arxiv.org/abs/1712.05526v1, 2017.

[8] E. Chou, F. Tramèr, G. Pellegrino, and D. Boneh. Sentinet: Detecting physical attacks against deep learning systems, 2018.

[9] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal. STRIP: A defence against trojan attacks on deep neural networks. In *Proc. ACSAC*, 2019.

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[11] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proc. ICLR*, 2015.

[12] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access*, 7:47230–47244, 2019.

[13] J. Guo, P. V. K. Borges, C. Park, and A. Gawel. Local descriptor for robust place recognition using LiDAR intensity. *IEEE Robotics and Automation Letters*, 4(2):1470–1477, 2019.

[14] W. Guo, L. Wang, X. Xing, M. Du, and D. Song. TABOR: A highly accurate approach to inspecting and restoring Trojan backdoors in AI systems. https://arxiv.org/abs/1908.01763, 2019.

[15] A. Hamdi, S. Rojas, A. Thabet, and B. Ghanem. AdvPC: Transferable Adversarial Perturbations on 3D Point Clouds". In *ECCV 2020*, pages 241–257, 2020.

[16] F. R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69, 1974.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

[18] L. Huang, A.D. Joseph, B. Nelson, B.I.P. Rubinstein, and J.D. Tygar. Adversarial machine learning. In *Proc. 4th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2011.

[19] G. Korres and M. Eid. Haptogram: Ultrasonic point-cloud tactile stimulation. *IEEE Access*, 4:7758–7769, 2016.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[21] Y. Li, B. Wu, Y. Jiang, Z. Li, and S.-T. Xia. Backdoor learning: A survey, 2020.

[22] D. Liu, R. Yu, and H. Su. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2279–2283, 2019.

[23] K. Liu, B. Doan-Gavitt, and S. Garg. Fine-pruning: Defending against backdoor attacks on deep neural networks. In *Proc. RAID*, 2018.

[24] S.-M. M.-Dezfooli, A. Fawzi, and P. Frossard. DeepFool: a simple and accurate method to fool deep neural networks. In *Proc. CVPR*, 2016.

[25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *Proc. ICLR*, 2018.

[26] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[27] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[28] D. J. Miller, Z. Xiang, and G. Kesidis. Adversarial learning in statistical classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE*, 108:402–433, March 2020.

[29] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Proc. CVPR*, 2017.

[30] B. Nelson and B. Barreno et al. Misleading learners: Co-opting your spam filter. In *Machine Learning in Cyber Trust: Security, Privacy, and Reliability*, 2009.

[31] A. Nguyen and A. Tran. Input-aware dynamic backdoor attack. In *Proc. NIPS*, 2020.

[32] P. Liang P. Koh. Understanding black-box predictions via influence functions. In *ICML*, 2017.

[33] P.-Y. Chen and H. Zhang and Y. Sharma and J. Yi and C.-J. Hsieh, C.-J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, page 15–26, 2017.

[34] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, B. Z. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, page 506–519, 2017.

[35] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Proc. 1st IEEE European Symp. on Security and Privacy*, 2016.

[36] N. Papernot, P. D. McDaniel, and I. J. Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. 2016.

[37] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, volume 30, pages 5099–5108, 2017.

[38] O. Schumann, M. Hahn, J. Dickmann, and C. Wöhler. Semantic segmentation on radar point clouds. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2179–2186, 2018.

[39] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015.

[40] J.-C. Su, M. Gadelha, R. Wang, and S. Maji. A deeper look at 3d shape classifiers. In Laura Leal-Taixé and Stefan Roth, editors, *Computer Vision – ECCV 2018 Workshops*, 2019.

[41] C. Szegedy, W. Zaremba, I Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *Proc. ICLR*, 2014.

[42] B. Tran, J. Li, and A. Madry. Spectral signatures in backdoor attacks. In *Proc. NIPS*, 2018.

[43] J. Tu, M. Ren, S. Manivasagam, M. Liang, B. Yang, R. Du, F. Cheng, and R. Urtasun. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[44] B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B.Y. Zhao. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *Proc. IEEE Symposium on Security and Privacy*, 2019.

[45] R. Wang, G. Zhang, S. Liu, P.-Y. Chen, J. Xiong, and M. Wang. Practical detection of trojan neural networks: Data-limited and data-free cases. In *Proc. ECCV*, 2020.

[46] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. Graph.*, 38(5), 2019.

[47] Y. Wen, J. Lin, K. Chen, C. L. P. Chen, and K. Jia. Geometry-aware generation of adversarial point clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[48] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[49] C. Xiang, C. R. Qi, and B. Li. Generating 3d adversarial point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[50] Z. Xiang, D.J. Miller, and G. Kesidis. A benchmark study of backdoor data poisoning defenses for deep neural network classifiers and a novel defense. In *Proc. IEEE MLSP*, Pittsburgh, 2019.

[51] Z. Xiang, D.J. Miller, Hang Wang, and G. Kesidis. Revealing Perceptible Backdoors in DNNs, Without the Training Set, via the Maximum Achievable Misclassification Fraction Statistic. In *Proc. IEEE MLSP*, Oct. 2020.

[52] Z. Xiang, D. J. Miller, and G. Kesidis. Detection of backdoors in trained classifiers without access to the training set. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2020.

[53] Z. Xiang, D. J. Miller, and G. Kesidis. Revealing Backdoors, Post-Training, in DNN Classifiers via Novel Inference on Optimized Perturbations Inducing Group Misclassification. In *Proc. IEEE ICASSP*, pages 3827–3831, 2020.

[54] Z. Xiang, D. J. Miller, and G. Kesidis. L-red: Efficient post-training detection of imperceptible backdoor attacks without access to the training set. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3745–3749, 2021.

[55] Z. Xiang, D. J. Miller, and G. Kesidis. Reverse engineering imperceptible backdoor attacks on deep neural networks for detection and training set cleansing. *Computers and Security*, 106:102280, 2021.

[56] H. Xiao, B. Biggio, B. Nelson, H. Xiao, C. Eckert, and F. Roli. Support vector machines under adversarial label contamination. *Neurocomputing*, 160(C):53–62, July 2015.

[57] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17:151–178, 2020.

[58] X. Xu, Q. Wang, H. Li, N. Borisov, C. A. Gunter, and B. Li. Detecting AI trojans using meta neural analysis. https://arxiv.org/abs/1910.03137, 2019.

[59] B. Wu L. Li R. He S. Lyu Y. Li, Y. Li. Backdoor Attack with Sample-Specific Triggers. https://arxiv.org/abs/2012.03816, 2020.

[60] J. Yang, Q. Zhang, B. Ni, L. Li, J. Liu, M. Zhou, and Q. Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3318–3327, 2019.

[61] X. Yue, B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli. A LiDAR Point Cloud Generator: From a Virtual World to Autonomous Driving. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, ICMR '18, page 458–464, 2018.

[62] D. Zhang. Big data security and privacy protection. In *Proceedings of the 8th International Conference on Management and Computer Science (ICMCS 2018)*, pages 275–278, 2018/10.

[63] H. Zhao, L. Jiang, C. Fu, and J. Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[64] T. Zheng, C. Chen, J. Yuan, B. Li, and K. Ren. Pointcloud saliency maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[65] H. Zhong, H. Zhong, A. Squicciarini, S. Zhu, and D.J. Miller. Backdoor embedding in convolutional neural network models via invisible perturbation. In *Proc. CODASPY*, March 2020.

[66] H. Zhou, K. Chen, W. Zhang, H. Fang, W. Zhou, and N. Yu. DUP-Net: Denoiser and Upsampler Network for 3D Adversarial Point Clouds Defense. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1961–1970, 2019.