DETECTING BACKDOOR ATTACKS AGAINST 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

Backdoor attacks (BA) are an emerging threat to deep neural network classifiers. A classifier being attacked will predict to the attacker's target class when a test sample from a source class is embedded with the backdoor pattern (BP). Recently, the first BA against point cloud (PC) classifiers was proposed, creating new threats to many important applications including autonomous driving. Such PC BAs are not detectable by existing BA defenses due to their special BP embedding mechanism. In this paper, we propose a reverse-engineering defense that infers whether a PC classifier is backdoor attacked, without access to its training set or to any clean classifiers for reference. The effectiveness of our defense is demonstrated on the benchmark ModeNet40 dataset for PCs.

Index Terms— backdoor, trojan, point cloud, DNN

1. INTRODUCTION

Deep neural network classifiers have achieved good performance in many point cloud (PC) classification tasks [1, 2]. However, they are shown to be vulnerable to adversarial attacks [3, 4, 5], including a recently proposed PC backdoor attack (BA) [6]. BAs were initially proposed for the image domain. A classifier being attacked will likely predict to the attacker's *target class* whenever a test sample from a *source class* of the attack is embedded with a *backdoor pattern* (BP) [7, 8, 9]. BAs are also not easily detectable since they negligibly affect the classifier's accuracy on clean test samples.

Defenses against BAs have been extensively studied for images. Existing state-of-the-art BA defenses mostly belong to a category of *reverse-engineering defenses* (REDs). REDs detect whether a classifier is backdoor attacked without access to its training set and without reference to any clean classifiers trained for the same domain [10, 11, 12, 13, 14, 15, 16]. These advantages make REDs suitable for the most popular, practical scenario nowadays, where training of the classifier is outsourced due to high computational cost and the need for "big data" for training [17]. In such a scenario, the defender is merely the consumer of the classifier (e.g. a mobile app user) without access to the training process and with no capability to train clean classifiers for the same domain [18].

Despite the success of existing REDs against image BAs, they are not applicable to the recently proposed PC BA. To facilitate practical implementation using physical objects (e.g. a ball carried by a pedestrian), the BP for this PC BA is designed as a small set of points inserted at a *common* spatial location *close* to the original points of all source class PCs [6]. However, typical BPs for image BAs are either tiny additive perturbations [12] or small patch triggers [10]. This difference in BP type prevents existing BP reverse-engineering formulated for image REDs from being applied to PCs.

In this paper, we propose a RED to detect whether a PC classifier is backdoor attacked – this is the *first* defense against PC BAs *without* access to the classifier's training set. We propose a novel BP reverse-engineering problem specific to PCs. Different from image REDs, which trial-estimate a BP either for each putative target class [10, 19] or for each putative (source, target) class pair [12, 15] we perform BP reverse-engineering for each putative *source class* and simultaneously estimate a target class. This different design (compared with image REDs) addresses the generally strong robustness of PC classifiers [20, 4], for which BP estimation is a hard problem for a non-negligible number of putative target classes. Moreover, for some putative source classes, there exists a spatial location *close to* most source class PCs, such that a single inserted point can cause most of these PCs to be misclassified to a *common* target class, *irrespective of the existence of a BA*. Such an "*intrinsic backdoor*" can easily cause false detections when there is actually no attack. While distinguishing intrinsic backdoors from those caused by attack is still an open problem even for image BA defense [12], we propose a novel combined detection statistic to address this challenging problem. Finally, we evaluate our detector on the benchmark ModeNet40 dataset for PCs to show its effectiveness.

2. RELATED WORK

2.1. BA against PC Classifiers

Consider a PC domain with sample space $\mathcal{X}$ and label space $\mathcal{C}$. A PC BA aims to have the victim classifier $f(\cdot) : \mathcal{X} \to \mathcal{C}$ predict to some attacker's target class $t^* \in \mathcal{C}$ whenever a test sample $\mathbf{X} = \{\mathbf{x}_i \in \mathbb{R}^3 | i = 1, \cdots, n\} \in \mathcal{X}$ from some source class $s^* \in \mathcal{C}$ is embedded with a BP $\mathbf{V}^*$ [6]. Here, $\mathbf{V}^*$ is a set of *inserted points* jointly specified by a spatial location $\mathbf{c}^* \in \mathbb{R}^3$ and a local geometry $\mathbf{U}^* = \{\mathbf{u}_j^* \in \mathbb{R}^3 | j = 1, \cdots, n'\}$:

$$\mathbf{V}^* = \{\mathbf{u}_j^* + \mathbf{c}^* | \mathbf{u}_j^* \in \mathbb{R}^3, \mathbf{c}^* \in \mathbb{R}^3, j = 1, \cdots, n'\}. \quad (1)$$

For $\mathbf{V}^*$, the spatial location $\mathbf{c}^*$ is optimized by the attacker such that its distance to these class s^* PCs, measured by $\mathbb{E}_{\mathbf{X} \sim P_{s^*}}[d(\mathbf{c}^*, \mathbf{X})]$, is sufficiently small, where $d(\mathbf{c}, \mathbf{X}) = \min_{\mathbf{x} \in \mathbf{X}} ||\mathbf{c} - \mathbf{x}||_2$ is the distance between $\mathbf{c} \in \mathbb{R}^3$ and $\mathbf{X} \in \mathcal{X}$. P_k is the sample distribution for class $k \in \mathcal{C}$. Then, a *perfectly* successful *backdoor mapping* would achieve

$f(\mathbf{X} \cup \mathbf{V}^*) = t^*, \forall \mathbf{X} \sim P_{s^*}$; and with no misclassifications induced for samples not from a source class of the attack. Like image BAs, a PC BA is typically launched by poisoning the classifier's training set with a small set of PCs originally from class s^*, embedded with the same BP $\mathbf{V}^*$, and labeled to class t^*. PC BAs are also hard to detect since they have negligible effect on classifier's predictions for PCs with BP.

2.2. BA Defense

BA defenses have been extensively studied for images; but no defenses have been proposed for PC BAs yet. Some BA defenses aim to find and remove poisoned samples (with BP embedded) from the training set [21, 22, 23]. Irrespective of their deployment being infeasible for scenarios where the defender is the user of the classifier without access to the training set, these defenses are anyway not effective for PC BAs [6].

A family of *reverse-engineering defenses* does not require access to the classifier's training set. They trial-estimate a BP for each putative (source, target) class pair (or target class only [10, 19]) using a small, clean dataset independently collected by the defender [12]. When there is an attack, the pattern estimated for the true BA class pair should be related to the BP used by the attacker and exhibit some atypicality compared with patterns estimated for non-BA class pairs. For example, to detect image BAs with an additive perturbation BP, [12] builds a purely unsupervised anomaly detector based on the fact that the norm of such BP is much smaller than the minimum perturbation norm required to induce high group misclassification for non-BA class pairs. Thus, a BA is detected when there exists a class pair with an abnormally small estimated perturbation norm. However, REDs are tailored to the BP embedding mechanism; thus, existing REDs designed for images BAs are not applicable to PC BAs.

3. METHOD

3.1. Overview

Goals and assumptions. The defender aims to infer whether a classifier is backdoor attacked and to determine the target class if an attack is detected. The defender has no access to the classifier's training set and is not capable of training any classifiers. The defender does possess an independently collected small, clean dataset for detection. These goals and assumptions are the same as for existing image REDs [12, 10, 11].

Key ideas. Our detector is based upon the following intuitions. These intuitions not only guide our detector design, but are also verified experimentally by the success of our detector. **I1**: For *most* non-BA class pairs, a *common* set of inserted points that induces high group misclassification from source class to target class will be spatially *far* from the points of source class PCs; but for BA class pair (s^*, t^*), the existence of the *backdoor mapping* guarantees the existence of a *common* spatial location close to the source class PCs (likely near $\mathbf{c}^*$), where a set of inserted points can induce most source class PCs to be misclassified to the target class. **I2**: A few non-BA class pairs may be associated with an *"intrinsic backdoor"*. For these class pairs, like a true BA class pair, there exists a spatial location close to most PCs from the source class, such that a common set of inserted points there will induce most of these PCs to be misclassified to the target class. However, such a spatial location, different from $\mathbf{c}^*$ (specified by the attacker) for the true backdoor, will likely be *close to* the points of most *target class* PCs. **I3**: Unlike backdoor mappings caused by attack with a *single common* spatial location $\mathbf{c}^*$, an intrinsic backdoor is likely due to the source and target classes being *"semantically"* similar, such that there may exist *several* intrinsic backdoor points for a given non-BA class pair, with each one close to source class PCs (Fig. 1). In this case (for a source class with an intrinsic backdoor), the *closest* (*sample-wise*) spatial location for a set of inserted points to induce a (*sample-wise*) misclassification to the target class can be different for different PCs from the same source class.

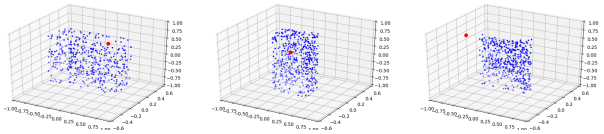

Fig. 1: Example of intrinsic backdoor from one of our experiments (P_6-PN in Tab 1): for PCs from the same source class, spatial locations estimated *sample-wise* (in red) are all close to these PCs (in blue), but are different from PC to PC.

Detection procedure. Our detection method consists of a BP estimation step followed by a detection inference step.

3.2. Step 1: BP Estimation

To find BA class pairs if there are any, based on I1, we need to first find, for each class pair, the *common* spatial location *closest* to the source class PCs such that a set of points inserted there induces most of these PCs to be misclassified to the target class, i.e., we need to reverse-engineer the BP. According to [6], the backdoor mapping mostly relies on the spatial location $\mathbf{c}^*$ but not the local geometry $\mathbf{U}^*$ of the inserted points. Thus, we can focus on reverse-engineering the spatial location of BP with arbitrary local geometry – for simplicity, we insert a *single* point at the spatial location. Formally, we aim to solve the following problem for each class pair $(s, t) \in \mathcal{C} \times \mathcal{C}$:

$$\begin{aligned} \min_{\mathbf{c} \in \mathbb{R}^3} \quad & \sum_{\mathbf{X} \in \mathcal{D}_s} d(\mathbf{c}, \mathbf{X}) \\ \text{s. t.} \quad & \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{X} \in \mathcal{D}_s} \mathbb{1}\big[f(\mathbf{X} \cup \{\mathbf{c}\}) = t\big] \geq \pi, \end{aligned} \tag{2}$$

where $\mathcal{D}_s$ is the subset of clean samples from class s possessed by the defender; $\mathbb{1}[\cdot]$ is the indicator function; and $\pi \in [0, 1]$ is a group misclassification fraction which is typically set large (e.g. $\pi = 0.9$ was chosen in [12]).

However, problem (2) is difficult to solve in practice. First, the indicator function in the constraint is not differentiable. Second, unlike image BPs (e.g. an additive perturbation) typically constrained by some range of valid pixel

Algorithm 1 Spatial location estimation for class $s \in \mathcal{C}$.

1: **Inputs**: data subset $\mathcal{D}_s$, classifier $f(\cdot)$, fraction π, step size δ, maximum iteration count $\tau_{\max}$, scaling factor α.
2: **Initialization**: $\mathbf{c}^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\lambda^{(0)}$ set to a small positive number (e.g. 10^{-5}), $\hat{\mathbf{c}}(s) = \boldsymbol{\infty}$, $\rho^{(0)} = 0$.
3: **for** $\tau = 0 : \tau_{\max} - 1$ **do**
4: $\quad \mathbf{c}^{(\tau+1)} = \mathbf{c}^{(\tau)} - \delta \nabla_{\mathbf{c}} l(\mathbf{c}; \mathcal{D}_s, \lambda^{(\tau)})|_{\mathbf{c}=\mathbf{c}^{(\tau)}}$
5: $\quad \rho^{(\tau+1)} = \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{X} \in \mathcal{D}_s} \mathbb{1}\big[f(\mathbf{X} \cup \{\mathbf{c}^{(\tau+1)}\}) \neq s\big]$
6: $\quad$ **if** $\rho^{(\tau+1)} \geq \pi$ **then**
7: $\quad\quad \lambda^{(\tau+1)} = \lambda^{(\tau)} \cdot \alpha$
8: $\quad\quad$ **if** $\sum_{\mathbf{X} \in \mathcal{D}_s} \big[d(\mathbf{c}^{(\tau+1)}, \mathbf{X}) - d(\hat{\mathbf{c}}(s), \mathbf{X})\big] < 0$ **then**
9: $\quad\quad\quad \hat{\mathbf{c}}(s) = \mathbf{c}^{(\tau+1)}$
10: $\quad\quad$ **else** $\lambda^{(\tau+1)} = \lambda^{(\tau)} / \alpha$
11: **Outputs**: $\hat{\mathbf{c}}(s)$

values, the search space for a point spacial location is unlimited. Also, due to the generally strong adversarial robustness of recent PC classifiers [20, 4], for many class pairs, there may not even exist a spatial location reasonably close to the source class PCs, where an inserted point can induce high (e.g. at least π) group misclassification to the target class. For these class pairs, even finding a solution just to satisfy the constraint of (2) may be infeasible in practice. To address the two challenges above, we perform BP estimation for each putative *source class* by minimizing a *differentiable surrogate* objective. In particular, for each source class $s \in \mathcal{C}$, we search for the closest spatial location to PCs from class s, such that a point inserted there causes at lease π fraction of these PCs to be misclassified to any class *other than* s (untargeted misclassifications). Formally, we minimize loss:

$$l(\mathbf{c}; \mathcal{D}_s, \lambda) = \sum_{\mathbf{X} \in \mathcal{D}_s} \big[h(s|\mathbf{X} \cup \{\mathbf{c}\}) - \max_{k \neq s} h(k|\mathbf{X} \cup \{\mathbf{c}\})\big] + \lambda \sum_{\mathbf{X} \in \mathcal{D}_s} d(\mathbf{c}, \mathbf{X}) \tag{3}$$

over $\mathbf{c}$ using Alg. 1. The first sum in Eq. (3) is inspired by the *untargeted* CW loss in [24], where $h(k|\mathbf{X})$ is the output of $\mathbf{X} \in \mathcal{X}$ for class $k \in \mathcal{C}$ directly prior to the softmax activation, which is smoother for optimization than the class posterior constrained in interval $[0, 1]$. Using such untargeted loss, we let the source class PCs "vote" for a target class by:

$$\hat{t}(s) = \operatorname*{argmax}_{k \neq s} \sum_{\mathbf{X} \in \mathcal{D}_s} \mathbb{1}\big[f(\mathbf{X} \cup \{\hat{\mathbf{c}}(s)\}) = k\big] \tag{4}$$

where $\hat{\mathbf{c}}(s)$ is the spatial location estimated for class s. Also, compared with some image REDs that estimate BPs for each class pair [12], our detector performs $\mathcal{O}(K)$ BP estimations instead of $\mathcal{O}(K^2)$ (where $K = |\mathcal{C}|$ is the number of classes); thus it is more *efficient* for large K. The second sum in Eq. (3) is a regularizer which constrains the distance of $\mathbf{c}$ to the points of the source class PCs. The coefficient λ is automatically adjusted according to line 6-10 of Alg. 1.

In addition to the *group* BP estimation above, based on intuition I3, for each putative source class $s \in \mathcal{C}$, we also need to estimate a *sample-wise* spatial location for each $\mathbf{X} \in \mathcal{D}_s$, given the estimated target class $\hat{t}(s)$. Formally, we minimize the following loss using the same Alg. 1:

$$\tilde{l}(\mathbf{c}; \mathbf{X}, \lambda) = h(s|\mathbf{X} \cup \{\mathbf{c}\}) - h(\hat{t}(s)|\mathbf{X} \cup \{\mathbf{c}\}) + \lambda d(\mathbf{c}, \mathbf{X}) \tag{5}$$

with $\mathcal{D}_s$ replaced by a set of a single PC $\{\mathbf{X}\}$, and with the loss in line 4 replaced by Eq. (5). We denote the estimated sample-wise (SW) spatial location for $\mathbf{X} \in \mathcal{D}_s$ as $\hat{\mathbf{c}}_{\text{sw}}(s, \mathbf{X})$.

3.3. Step2: Detection Inference

Our detection statistic is composed of three basic statistics (obtained from BP estimation for each putative source class) corresponding to the three intuitions in Sec. 3.1 respectively. For each $s \in \mathcal{C}$, we get: 1) the average distance from the estimated spatial location to points of source class PCs: $r_{\text{s}}(s) = \frac{1}{|\mathcal{D}_s|} \sum_{\mathbf{X} \in \mathcal{D}_s} d(\hat{\mathbf{c}}(s), \mathbf{X})$; 2) the average distance from the estimated spatial location to points of PCs from the estimated target class $\hat{t}(s)$: $r_{\text{t}}(s) = \frac{1}{|\mathcal{D}_{\hat{t}(s)}|} \sum_{\mathbf{X} \in \mathcal{D}_{\hat{t}(s)}} d(\hat{\mathbf{c}}(s), \mathbf{X})$; and 3) a normalized similarity score: $w(s) = (z(s) - \min_{k \in \mathcal{C}} z(k)) / (\max_{k \in \mathcal{C}} z(k) - \min_{k \in \mathcal{C}} z(k))$, where $z(k) = \frac{1}{|\mathcal{D}_k|} \sum_{\mathbf{X} \in \mathcal{D}_k} \frac{\hat{\mathbf{c}}(k) \cdot \hat{\mathbf{c}}_{\text{sw}}(k, \mathbf{X})}{|\hat{\mathbf{c}}(k)| \, |\hat{\mathbf{c}}_{\text{sw}}(k, \mathbf{X})|}$ is the average cosine similarity[1] between the estimated sample-wise spatial location for each $\mathbf{X} \in \mathcal{D}_k$ and the estimated group spatial location for class $k \in \mathcal{C}$. The normalization limits the similarity score in interval $[0, 1]$ for generalization to different domains.

According to intuition I1, $r_{\text{s}}(s)$ will likely be large if $(s, \hat{t}(s))$ is a non-BA class pair; otherwise, $r_{\text{s}}(s)$ will likely be small. If for some class s, $(s, \hat{t}(s))$ is associated with an intrinsic backdoor mapping, such that $r_{\text{s}}(s)$ is abnormally small, based on I2 and I3, $r_{\text{t}}(s)$ or $w(s)$ (or both) will likely be abnormally small. Thus, for each putative source class $s \in \mathcal{C}$, we compute the combined detection statistic:

$$r(s) = w(s) \frac{r_{\text{t}}(s)}{r_{\text{s}}(s)}, \tag{6}$$

which will be abnormally large only if $(s, \hat{t}(s))$ is a BA pair.

Our inference is based on an *unsupervised* anomaly detection. We check among the statistics for all $s \in \mathcal{C}$ if there exists an abnormally large one. Ideally, like the image RED in [12], we can fit a null distribution using all statistics excluding the *largest one* and evaluate its *atypicality* under the null using the maximum *order statistic* p-value, which is generally insensitive to the number of statistics used for detection. However, like image BAs, a PC BA may cause *collateral damage*: a backdoor mapping with the same BP may be learned for some class pair (s, t^*) with $s \in \mathcal{C} \setminus \{s^*, t^*\}$ [12]. In such case, there may be *multiple* abnormally large statistics associated with *the same target class* t^* but different source classes; thus, the null distribution estimated with only the largest statistic being excluded may be biased. To solve this problem, we exclude statistics for all $s \in \mathcal{C}$ such that $\hat{t}(s) = \hat{t}(s_{\max})$ when estimating the null, where $s_{\max} = \arg\max_{k \in \mathcal{C}} r(k)$. Given all

[1] PCs are usually aligned to the origin for classification.

	P_1-PN	P_2-PN	P_3-PN	P_4-PN	P_5-PN	P_6-PN	P_7-PN	P_1-PN++	P_1-DGCNN
$1/r_s$	(**6.2e^{-3}**, **0.36**)	(**3.8^{-3}**, **0.16**)	(**4.3e^{-15}**, **0.33**)	(**2.2e^{-7}**, 2.6e^{-2})	(0.24, **0.11**)	(0.24, 1.6e^{-2})	(**4.3e^{-3}**, 9.7e^{-2})	(**u.f.**, 8.2e^{-6})	(**4.4e^{-5}**, 4.3e^{-2})
r_t/r_s	(**4.5e^{-2}**, 9.2e^{-6})	(**u.f.**, **0.32**)	(**6.1e^{-6}**, 9.8e^{-2})	(**2.8e^{-3}**, **0.58**)	(0.12, **0.19**)	(0.21, **0.60**)	(**6.7e^{-5}**, 9.0e^{-3})	(**u.f.**, **0.99**)	(0.10, **0.59**)
w/r_s	(**1,7e^{-7}**, **0.19**)	(**3.5e^{-3}**, **0.26**)	(**u.f.**, **0.27**)	(**5.6e^{-9}**, 9.2e^{-3})	(**1.4e^{-2}**, 6.1e^{-2})	(**1.4e^{-2}**, 2.6e^{-2})	(**5.5e^{-9}**, 7.0e^{-3})	(**u.f.**, **0.94**)	(0.22, 2.9e^{-2})
$r = w \cdot r_t/r_s$	(**3.3e^{-3}**, **0.38**)	(**u.f.**, **0.19**)	(**u.f.**, **0.20**)	(**u.f.**, **0.22**)	(**5.4e^{-2}**, **0.27**)	(**7.6e^{-4}**, **0.33**)	(**u.f.**, 9.3e^{-2})	(**5.5e^{-13}**, **0.99**)	(**1.9e^{-3}**, **0.18**)

Table 1: Order statistic p-value (pv), in form of (attack pv, clean pv), for nine pairs of classifiers being attacked with the associated clean classifier, for the statistic r used by our detector, and for three alternative statistics ($1/r_s$, r_t/r_s, and w/r_s). Attacks are associated with class pairs $P_1, ..., P_7$ in [6]; classifier architectures include PointNet (PN), PointNet++ (PN++), and DGCNN. "u.f." represents "underflow" for numbers in range $(0, 10^{-323})$. Successful detections (with $\phi = 0.05$) are in bold.

statistics in $[0, \infty)$, similar to [12], we choose a single-tailed parametric density form (e.g. Gamma distribution in both our experiments and [12]) for our null distribution, with cdf $G(\cdot)$, such that any abnormally large statistics (likely corresponding to BA class pairs) will likely appear in the tail (e.g. Fig. 2). Then the *maximum order statistic p-value* is:

$$\text{pv} = 1 - G(r(s_{\max}))^{K-J}, \tag{7}$$

where J is the number of statistics being excluded when estimating the null distribution. A detection threshold ϕ is chosen (e.g. the classical $\phi = 0.05$), such that a BA is detected with confidence $1 - \phi$ if pv $< \phi$. When a BA is detected, $\hat{t}(s_{\max})$ is inferred as the target class.

4. EXPERIMENTS

4.1. Experiment Settings

Dataset: The benchmark ModelNet40 dataset contains 12311 aligned CAD models from 40 object categories [25]. 9843 and 2468 PCs are in the training set and test set, respectively.

Attack configurations: We consider the seven attacks created in [6] for ModelNet40, where the BP for each attack is a set of points inserted at an optimal spatial location and with a random local geometry (i.e. the "RS" geometry in [6]). The source and target class pair for these seven attacks are also specified in [6] and named as "$P_1, ..., P_7$". For each attack, 15 source class PCs embedded with the BP and labeled to the target class are used for poisoning the training set.

Classifier, training, and attack effectiveness: We consider the same state-of-the-art PC classifiers as in [6]. In particular, we consider the same PointNet classifiers [20] trained for all seven attacks, the same PointNet++ classifier [26] and the same DGCNN classifier [27] trained for the attack associated with class pair P_1, with the training configurations detailed in [6]. All the classifiers being attacked exhibit high attack success rate and almost no degradation in clean test accuracy as reported in [6]. To evaluate false detections of our detector, for each classifier being attacked, we also train a classifier with no BA, using the same training configurations.

4.2. Detection Performance Evaluation and Results

Detector configurations: Following the assumptions in Sec. 3.1, for each class, we randomly select 10 clean PCs that are correctly classified from the ModelNet40 data set, to form the clean set used for detection – these PCs are not used for training. For BP estimation (both group-wise and sample-wise) using Alg. 1, we set $\pi = 0.9$, $\delta = 0.1$, $\tau_{\max} = 3$k, and $\alpha = 1.5$. These choices are not critical to the performance of our detector, and can be easily chosen to minimize the loss value at convergence. We also perform 10 random initializations and pick the best solution, which is a common practice for solving highly non-convex problems.

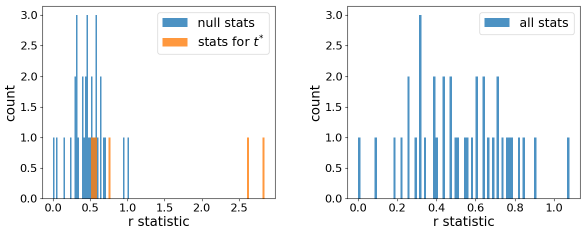

Fig. 2: Histogram of r statistics for the classifier with BA (left) and its associated clean classifier without BA (right). For the attack case, statistics for two source classes "voting" to the BA target class t^* are abnormally large.

Detection performance: For each classifier being attacked and its associated clean classifier, we report the order statistic p-values obtained using our statistic $r = w \cdot r_t/r_s$ (Eq. (6)) in Tab. 1 (last row). In general, the p-values are large for most clean classifiers and are small when there is a BA, as we expect. Applying the classical $\phi = 0.05$ threshold to these p-values, *we only missed one attack* (P_5-PN, and barely, since the p-value is 0.054) *with zero false detections*. For each detected attack, the BA target class is also correctly inferred.

Ablation study: As the *first* defense designed for PC BAs, we do not have a competitor to compare with. Still, we show detection performance using simplified statistics instead of the combined one used by our detector. In Tab. 1, for these simplified statistics, the p-values for some clean classifiers (e.g. P_4-PN, P_6-PN) are small due to the existence of *intrinsic backdoors*, which easily causes false detections. Even for some classifiers being attacked (e.g. P_5-PN, P_6-PN), using these simplified statistics, the null estimation will be affected by intrinsic backdoors, such that the resulting p-value will not be small enough to trigger a detection. These results highlight the importance of all three components (each strongly motivated by the intuition in Sec. 3.1) of our detection statistic.

Visualization of detection: In Fig. 2, we show the distribution of our statistic for both the classifier being attacked and the clean classifier for P_4-PN, for example. When there is a BA, two statistics associated with the true BA target class clearly appear in the "tail" of the null distribution, triggering a detection with correct inference of the BA target class.

5. CONCLUSIONS

We proposed the first RED to detect a recent PC BA, without access to the training set. We are the first to reverse-engineer BPs for PCs. We also proposed a novel detection statistic that conquers the intrinsic backdoor problem.

6. REFERENCES

[1] 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*, vol. 38, no. 1, pp. 68–86, 2021.

[2] 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*, vol. 4, no. 2, pp. 1470–1477, 2019.

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

[4] C. Xiang, C. R. Qi, and B. Li, "Generating 3d adversarial point clouds," in *CVPR*, June 2019.

[5] D. Liu, R. Yu, and H. Su, "Extending adversarial attacks and defenses to deep 3d point cloud classifiers," in *ICIP*, 2019, pp. 2279–2283.

[6] Z. Xiang, D. J. Miller, S. Chen, X. Li, and G. Kesidis, "A backdoor attack against 3d point cloud classifiers," in *ICCV*, 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] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, and J Zhai, "Trojaning attack on neural networks," in *Proc. NDSS*, San Diego, CA, Feb. 2018.

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

[10] 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.

[11] 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.

[12] 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*, pp. 1–15, 2020.

[13] Y. Dong, X. Yang, Z. Deng, T. Pang, Z. Xiao, H. Su, and J. Zhu, "Black-box detection of backdoor attacks with limited information and data," in *ICCV*, 2021.

[14] H. Chen, C. Fu, J. Zhao, and F. Koushanfar, "Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks," in *IJCAI*, 2019.

[15] Z. Xiang, D. J. Miller, H. Wang, and G. Kesidis, "Detecting Scene-Plausible Perceptible Backdoors in Trained DNNs Without Access to the Training Set," *Neural Computation*, pp. 1329–1371, 2021.

[16] Z. Xiang, D.J. Miller, H. Wang, and G. Kesidis, "Revealing Perceptible Backdoors, without the Training Set, via the Maximum Achievable Misclassification Fraction Statistic," in *Proc. IEEE MLSP*, 2020.

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

[18] 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.

[19] 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.

[20] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *CVPR*, 2017.

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

[22] 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.

[23] 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*, 2019.

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

[25] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A Deep Representation for Volumetric Shapes," in *CVPR*, June 2015.

[26] 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*, 2017, vol. 30, pp. 5099–5108.

[27] 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.*, 2019.