Appendix of "PointBA: Towards Backdoor Attacks in 3D Point Cloud"

Xinke Li[1*] Zhirui Chen[1*] Yue Zhao[1†]
Zekun Tong[1] Yabang Zhao[1] Andrew Lim[2] Joey Tianyi Zhou[3]
[1]National University of Singapore [2]Southwest Jiaotong University
[3]Institute of High Performance Computing, A*STAR
{xinke.li, zhiruichen, yuezhao, zekuntong, zhaoyabang}@u.nus.edu
i@limandrew.org joey.tianyi.zhou@gmail.com

In this Appendix, we provide further information, visualizations, and qualitative results for our proposed technique. In Sec. A, we discuss the proposed attack pipeline and the performance of clean models. Then, in Sec. B we present further experimental results based on several 3D backdoor trigger variations. Following that, in Sec. C, we go into further depth regarding the clean-label attack method, including the formulation of the feature disentanglement, its optimization, and the transferability experiments. Finally, in Sec. D, we illustrate the exact settings for three defense methods and the associated attack success rates against them.

A. Details of Attack Setting

In this section, we supplement details of the attack settings that have not been stated in the original paper, including the pipeline of the PointPBA, and the performance of the clean models on the test set.

A.1. Pipeline of PointPBA

Our point poison-label backdoor attack is described in detail in the original Sec.4 and is demonstrated in Alg. 1, where the notations are consistent with the original paper.

Algorithm 1: Poison-label Backdoor Attack

Input: A model structure f, training set $\mathbb{P}_N$, injection rate ϵ, a sample vector $\boldsymbol{\delta}$, a global transformation matrix $\boldsymbol{A}$, an interaction shape $\boldsymbol{B}$, a target label t.
Output: Backdoored model $f_{\boldsymbol{\theta}'}$

Random sample ϵ proportion of data in $\mathbb{P}_N$, denote them as $\tilde{\mathbb{P}} = \{z_1, \cdots, z_J\}$ where $J = \lfloor \epsilon N \rfloor, z_j = (\boldsymbol{X}_j, y_j)$
Set $\mathbb{P}'_N \leftarrow \mathbb{P}_N \backslash \tilde{\mathbb{P}}$
for $j = 1$ to J **do**
    Set $\boldsymbol{X}'_j \leftarrow (\boldsymbol{I} - \mathrm{Diag}(\boldsymbol{\delta}))\mathrm{X_jA} + \mathrm{Diag}(\boldsymbol{\delta})\mathrm{B}$
    Set $y'_j \leftarrow t$
    Set $\mathbb{P}'_N \leftarrow \mathbb{P}'_N \cup (\boldsymbol{X}'_j, y'_j)$
end for
Train model of structure f on dataset $\mathbb{P}'_N$ and obtain $f_{\boldsymbol{\theta}'}$

A.2. Baseline of Clean Models

Here we report the test accuracy (ACC) of four clean 3D deep models on different datasets. The split of the dataset and the settings of the training process are consistent with the original paper. Comparing the data in Tab. 1 with the results in the original paper, it can be seen that none of the three proposed attacks (PointPBA-I, PointPBA-O, and PointCBA) results in a drop of more than 3% in the performance of the model.

Table 1. Test accuracy (%) of clean models on different datasets.

Deep Models	ModelNet10	ModelNet40	ShapeNetPart
PointNet	89.65	85.09	98.12
PointNet++	92.07	89.42	98.50
DGCNN	92.62	90.12	98.85
PointCNN	92.18	88.82	98.26

B. Extra Results of 3D Backdoor Attack

In the original paper, we use a sphere as the interaction trigger and a rotation along the z-axis as the orientation trigger to show the power of the proposed triggers empirically. To demonstrate the generalizability of our proposed triggers in backdoor attacks, we investigate additional trigger configurations, such as interaction triggers with different shapes and orientation triggers along with other axes. We emphasize that the settings are identical to those in the original paper except for the differences in the trigger forms. The illustration of different triggers is presented in Fig. 1.

B.1. Interaction Trigger of Other Shapes

As shown in Tab. 2, the ASR results for the three shapes of triggers are almost consistent with the sphere-based interaction trigger. This shows that the shape of the triggers does not have a significant effect on our proposed backdoor attack, and it also demonstrates the flexibility and stealth of this backdoor attack since any physical object can be invoked as a backdoor trigger.

Table 2. ASR (%) of proposed PointPBA-I and PointCBA, and the backdoored model's test accuracy (%) on the clean test set. The experimental setting is consistent with PointNet++ in the original paper except for the interaction object shapes.

Shapes	ACC/ASR (%) of PointPBA-I			ACC/ASR (%) of PointCBA		
	MN10	MN40	SNPart	MN10	MN40	SNPart
Cube	91.9/99.7	89.2/98.2	98.3/98.4	91.0 /53.0	88.8/64.3	98.0/45.9
Cylinder	91.3/99.2	89.0/99.0	98.2/97.9	90.7/50.6	88.9/62.0	98.1/47.4
One-point	91.5/97.0	89.3/96.8	98.4/96.7	91.7/45.9	88.9/58.2	98.1/43.8

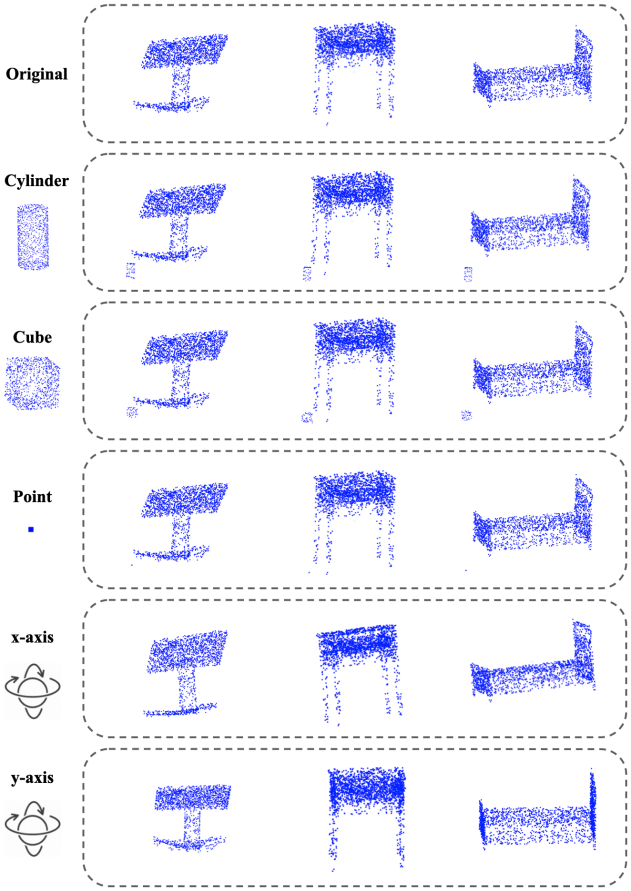

Figure 1. Illustration of 3D backdoor attack via different trigger settings. From top to bottom are, the origin point cloud, point clouds with different shape interaction triggers (cylinder, square, one point), and point clouds with the different rotation axis of orientation triggers (x-axis, y-axis).

Being seen in Fig. 1, as the smallest shape, the one-point-based trigger can hardly be noticed by human inspection. Even if the ASR of the PointCBA is just 45%, due to the clean label setting for bypassing the label inspection, this one-point trigger for backdoor attacks is more stealthy, and therefore more threatening in the real world scenario.

B.2. Orientation Trigger on Other Axes

Tab. 3 provides the results of the backdoor attack based on the orientation trigger with different rotation axes while the magnitude of rotation angle remains the same. It is evident that this is not significantly different from the results based on the z-axis in the original paper, which indicates that the effectiveness of the orientation trigger does not depend on the rotation axis.

Table 3. ASR (%) of our proposed PointPBA-O, and the backdoored model's accuracy (%) on the clean test set. The experiments are conducted on the orientation trigger along different axes. Other settings remain the same as the original paper.

Orientation Axes	ACC/ASR(%) of PointPBA-O Attack		
	MN10	MN40	SNPart
x-axis	91.9/99.2	89.2/98.1	98.7/97.4
y-axis	91.7/99.6	89.1/98.6	98.5/97.7

C. Details of PointCBA

Here, we detail how we optimize the objective of feature disentanglement, including the specific steps for problem reformulation and Bayesian Optimization. The comprehensive result of PointCBA transferability is then provided.

C.1. Optimization of Feature Disentanglement

Problem reformulation. The objective of feature disentanglement is demonstrated in Eq. (10) of the original paper. Given a sample $\boldsymbol{X}_j$, we have the following optimization problem to find such ω_j

$$\begin{aligned} \max_{\boldsymbol{\omega}_j} \quad & \sum_{\boldsymbol{X}_i \in \mathbb{P}_t \setminus \{\boldsymbol{X}_j\}} \mathcal{D}(\phi_{\boldsymbol{\theta}}(\boldsymbol{X}_j \boldsymbol{A}_{\boldsymbol{\omega}_j}), \phi_{\boldsymbol{\theta}}(\boldsymbol{X}_i)) \\ \text{s.t.} \quad & \boldsymbol{\omega}_j \in \mathcal{R} \end{aligned}, \tag{1}$$

where $\mathcal{D}$ is a distance metric in feature space $\mathbb{R}^d$ and $\mathcal{R} \subseteq \mathbb{R}^3$ is a range to restrict the rotation magnitude. In practice we use the normalized Euclidean distance for the distance metric $\mathcal{D}$. Instead of representation in Euler angles, we adopt the axis-angle representation of rotation matrix A, which is defined by a unit vector $\mathbf{e}$ indicating an axis and an angle ω_e describing the magnitude of the rotation on the axis. The advantage of such representation is that the magnitude of perturbation can be simply controlled by a single variable ω_e. With this representation, the feature disentanglement function $c(\cdot)$ will be,

$$\begin{aligned} c(\boldsymbol{X_j}) &= \boldsymbol{X_j} \boldsymbol{A}_{\mathbf{e},\boldsymbol{\omega_e}} \\ \text{s.t.} \quad \|\mathbf{e}\| &= 1, \omega_e \in [0, \omega_{max}] \end{aligned}. \tag{2}$$

Therefore, the formulated objective of rotation-based feature disentanglement is,

$$\max_{\mathbf{e},\omega_e} \quad \sum_{\boldsymbol{X}_i \in \mathbb{P}_t \setminus \{\boldsymbol{X}_j\}} \frac{\|\phi_{\boldsymbol{\theta}}(\boldsymbol{X}_j \boldsymbol{A}_{\mathbf{e},\omega_e}) - \phi_{\boldsymbol{\theta}}(\boldsymbol{X}_i)\|}{\|\phi_{\boldsymbol{\theta}}(\boldsymbol{X}_j \boldsymbol{A}_{\mathbf{e},\omega_e})\|} \quad . \tag{3}$$
$$\text{s.t.} \quad \|\mathbf{e}\| = 1, \omega_e \in [0, \omega_{max}]$$

Bayesian optimization. Generally speaking, Bayesian Optimization (BO) is a powerful tool to help find the global optimum of computationally expensive or black-box functions [1]. Let $g : \mathcal{U} \to \mathbb{R}$, the problem is to

$$\max_{\boldsymbol{u} \in \mathcal{U}} \; g(\boldsymbol{u}) \tag{4}$$

where $\mathcal{U}$ is the domain of decision variable u. The BO method mainly consists of two steps: 1) first, it uses the Gaussian Process Regression to set up a prior distribution of g by observing the function value at several data points initially, where a mean function and kernel need to be selected (in our experiment, we use Gaussian Process model to construct the surrogate distribution and Matérn 5/2 kernel for the covariance matrix); 2) it further optimizes the function by if more evaluation points of g are allowed. In this procedure, the so-called acquisition function is utilized to select the next point for evaluation with consideration of exploration and exploitation. Common acquisition functions include Upper Confidence Bound (UCB), Expected Improvement (EI), and Probability of Improvement (PI), and we used EI in our design because of its fewer tuning parameters and decent performance. EI is defined as the expected value of improvement over the current best

$$\mathrm{EI}_n(\boldsymbol{u}) = \mathbb{E}_n \left[\max \left(g(\boldsymbol{u}) - g_n^*, \, 0 \right) \right], \tag{5}$$

where n is the number of total evaluated points $\boldsymbol{u}_1, \cdots, \boldsymbol{u}_n$, $\mathbb{E}_n$ is expectation taken under the posterior distribution given the evaluation of g at these points, and $g_n^* = \max_{k \le n} g(\boldsymbol{u}_n)$ is the current best function value.

We utilize the BO method to help find the best parameter $\boldsymbol{\theta}$ in our attack, the experiment process is shown in Alg. 2. In addition, we are not directly optimizing a rotation axis $\mathbf{e}$, but convert it to an equivalent representation (φ_e, ϑ_e) in continuous domain by

$$\begin{aligned} \varphi_e &= \arccos(e_z) \in [0, \; 180^\circ] \\ \vartheta_e &= \arctan\left(\frac{e_y}{e_x}\right) \in [0, \; 180^\circ], \end{aligned} \tag{6}$$

where $\mathbf{e} = [e_x, e_y, e_z]^\top$.

Setting and result. In the experiment, we implement the Matérn kernel and length scale of gamma prior $\Gamma(3.0, 6.0)$ for the covariance matrix. We also utilize the Latin sampler for initial sampling with a number of 10. The optimization of acquisition function EI is conducted via L-BFGS with 16 multistarts and the max iteration for the BO process is 30. We select a sample from 'Table' in ModelNet10 and demonstrate the optimization process of the algorithm on it with a fixed angle $\omega_e = 25^\circ$. The maximization of feature disentanglement loss is actually a nonconvex problem with respect to the angle representation of the rotation axis, thus we utilize the global optimization method. Finally, we may achieve a decent solution close to the global optimal point by using the BO method.

Algorithm 2: BO of Feature Disentanglement

Input: Loss $L(\cdot)$; A point cloud $\boldsymbol{X}$; Acquisition function EI$(\cdot)$;Threshold ω_{max}; Iteration Numbers $N_{\max}$; Initial Numbers n_0
Output: Best parameters $\boldsymbol{\theta}_{\text{best}} = (\mathbf{e}_{\text{best}}, \omega_{\text{best}})$;

Random Initial Points: Compute $L(\boldsymbol{\theta}_i|\boldsymbol{X})$ for $i = 1, \ldots, n_0$;
$\boldsymbol{\theta}_{best} \leftarrow \arg\min_{\boldsymbol{\theta}_i} L(\boldsymbol{\theta}_i|\boldsymbol{X})$;
$l_{best} \leftarrow L(\boldsymbol{\theta}_{best}|\boldsymbol{X})$;
while $n \leqslant N_{max}$ **do**
  Fit/Update Gaussian Process by data points: $\{(\boldsymbol{\theta}_i, l_i) : \; i = 1, \ldots, n\}$;
  $\boldsymbol{\theta}_{n+1} \leftarrow \arg\max_{\boldsymbol{\theta}} \mathrm{EI}_n(\boldsymbol{\theta})$;
  if $L(\boldsymbol{\theta}_{n+1}|x) < l_{\text{best}}$ **then**
    $\boldsymbol{\theta}_{best} \leftarrow \boldsymbol{\theta}_{n+1}$;
    $l_{\text{best}} \leftarrow L(\boldsymbol{\theta}_{n+1}|\boldsymbol{X})$;
  end
  $n \leftarrow n + 1$;
end
Return $\boldsymbol{\theta}_{\text{best}}$;

In the original paper, we mentioned the need to obtain a decent rotation angle to strike a balance between the perturbation on the original data and feature disentanglement. Here we provide a visualization of the poisoned samples based on different angles in Fig. 2. As it shows, the original data combined with a rotation with a small ω_e of and an interaction trigger with a small size is actually difficult to notice.

C.2. Transferability

We report the comprehensive transferability results of PointCBA in Tab. 4. The results show that the proposed PointCBA has comparatively strong transferability between different models.

Table 4. Attack success rates (%) of PointCBA by transferring the poisoned dataset across different models.

Clean Model	Backdoored Model			
	PN	PN++	PCNN	DGCNN
PN	82.5	40.9	52.1	46.6
PN++	68.1	53.8	60.6	38.3
PCNN	69.2	48.7	63.4	43.2
DGCNN	65.1	47.1	55.4	46.8

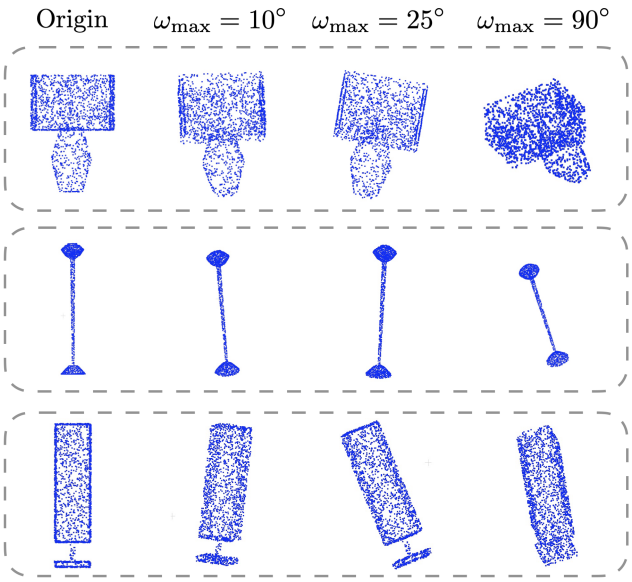

Figure 2. Visualization of point cloud after feature disentanglement. The points are from the 'lamp' category of ModelNet40. Different ω_{max} means different upper bounds for rotation angle searching in feature disentanglement.

D. Resistance to Defense Methods

D.1. Resistance to Data Augmentation

In this Appendix, we provide more details about the resistance of data augmentation to our attacks. Our data augmentation methods and parameter settings are listed in following: (1) for random jitter, we use a truncated Gaussian distribution with mean $\mu = 0$, variance $\sigma = 0.01$, and clip at 0.05; (2) for random scaling, we also use a truncated Gaussian distribution with mean $\mu = 1$, variance $\sigma = 0.1$ and clip at 1.3 and 0.7; (3) for random rotation, we imply rotation along the y-axis, same as the typical implementations[4, 2]. The random angle for rotation is set under $30°$. The experimental model and are PointNet++ and ModelNet10, respectively.

The experimental results in Tab. 5 illustrate that the commonly used data augmentation approaches are barely resistant to the proposed backdoor attacks. Using PointPBA-O and PointCBA which contain the rotation transform, we conduct additional experiments and come to the following conclusions: 1) When the rotation employed to augmentation is parallel to the rotation axis of the orientation trigger, the augmentation effectively resists PointPBA-O. 2) The ASR of PointCBA remains steady as the angle of rotation data augmentation rises as shown in Tab. 6. The first conclusion is self-evident, and the second conclusion is because the trigger enhancement will be stronger as the random rotation angle grows, which compensates for the decreased effectiveness of the feature disentanglement in PointCBA.

Table 5. The ASR(%) towards varying data augmentations for PointNet++ on ModelNet10.

Augmentation	PointPBA-I	PointPBA-O	PointCBA
None	97.5	95.2	53.8
Jitter	97.3	94.4	47.6
Scaling	96.4	93.7	54.7
Rotation (y-axis)	91.1	89.8	48.2

Table 6. ASR (%) and test accuracy(%) of PointCBA towards rotation-based data augmentation.

Dataset	ACC/ASR(%) under Rotation Aug		
	$5°$	$10°$	$30°$
SNPart	97.8/46.8	97.5/48.3	97.0/44.3
MN10	92.1/52.2	93.2/50.7	92.3/48.2

Table 7. The ASR(%) towards SOR for PointNet++ on ModelNet10.

k	PointPBA-I	PointCBA
2	98.4	51.9
10	96.9	49.7
20	91.2	41.1

D.2. Resistance to SOR

According to [7], SOR operation comprises two influential factors, k the number of neighbor points and α the percentage of points that are regarded as outliers. We first follow the settings in [7] which are $k = 2$ and $\alpha = 1.1$ and conduct the SOR in training the backdoored model. Observing that SOR does not decrease ASR for PointPBA-I and PointCBA, we increased k further and present the results for various k values in Tab. 7. The results indicate that ASR of the interaction trigger is obviously reduced only when k exceeds 20. This is because, despite its diminutive size, the object comprises approximately 30 points, and SOR with a tiny k does not successfully remove such outliers.

D.3. Resistance to Data Filtering

The authors of [6] propose a data filter method to successfully reject poison-label attack samples in 2D images. Inspired by their approach, we propose a simple method based on the conventional point cloud descriptors to conduct data filtering. Our filtering method is divided into three steps: 1) Construct a small and clean dataset, where the construction method can be manually filtered from the potentially poisoned dataset. 2) Extract the point cloud descriptors of the data and train a classification model on them. 3) Use the model to filter the data with large classification loss in the poisoned dataset.

In the experiment, we construct the clean dataset by randomly sampling 20% data of ModelNet10. The descriptors of the point cloud are the combination of SHOT[5] and GRSD[3]. We use a Multi-layer Perception (MLP) with one 128-units hidden layer as a classification model and cross-

entropy loss for training. Instead of directly threshing by loss, we set a prediction confidence threshold of 0.4 for data filtering. The results show that the PointCBA can resist such a data filtering method while the PointPBA poisoned data would be filtered effectively. Specifically, the poisoned data in PointPBA can be removed with up to 82% filtering rates, while the PointCBA can resist this defense with a rate of only 14%. However, we have to suffer from the loss of benign data with data filtering, which precisely in our experiments can reach up to 24% of the whole dataset.

References

[1] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018. 3

[2] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems*, pages 820–830, 2018. 4

[3] Zoltan-Csaba Marton, Dejan Pangercic, Radu Bogdan Rusu, Andreas Holzbach, and Michael Beetz. Hierarchical object geometric categorization and appearance classification for mobile manipulation. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 365–370. IEEE, 2010. 4

[4] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 4

[5] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010. 4

[6] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. Clean-label backdoor attacks. 2018. 4

[7] Hang Zhou, Kejiang Chen, Weiming Zhang, Han Fang, Wenbo Zhou, and Nenghai 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. 4