CausalPC: Improving the Robustness of Point Cloud Classification by Causal Effect Identification

Supplementary Material

A. Proof of the Causal Effect Identification

We detail the proof of Theorem 1 as follows.

Theorem 1. *Under the causal graph in Fig.2 (a), suppose the real-world object of a point cloud data x is o. The causal effect is transformed into the following equation:*

$$P(Y|do(X=x)) = P(Y|do(X=x,O=o)) = \int P(Z|x)\left[\int P(Y|Z,X)P(X|o)dX\right]dZ. \tag{5}$$

Proof. Firstly, we start with the following transformation:

$$\begin{aligned} P(Y|do(X=x,O=o)) &= P(Y|do(X=x)) \\ &= \int P(Y|Z,U)P(Z|x)P(U)dUdZ \\ &= \int P(Z|x)\big[\int P(Y|Z,U)P(U)dU\big]dZ \end{aligned} \tag{6}$$

According to the data generation process defined by the causal graph, when we observe a specific value of x, the information of o is inherently captured in the causal effect with $do(X=x)$. Therefore, we have the causal effect $P(Y|do(X=x,O=o))$ be equivalent to $P(Y|do(X=x))$. Formally, the intervention $do(X=x,O=o)$ sets $P(x|o)=1$ and $P(o)=1$, which eliminates the terms related to o in the equation.

However, there remains a critical challenge in identifying the causal effect in Eq.6, i.e., the estimation of the term $\int P(Y|Z,U)P(U)dU$. Note that the aforementioned estimation involves the explicit modeling of the unobservable U. Moreover, we have:

$$P(Y|Z) \neq \int P(Y|Z,U)P(U)dU, \tag{7}$$

because the variable Z also contains the information of U through the path $U \rightarrow X \rightarrow Z$. This prevents us from estimating the marginal probability $P(Y|Z)$ directly for $\int P(Y|Z,U)P(U)dU$ [29, 38, 53, 54, 65].

To address this term, we first have $P(U|Z,X) = P(U|X)$ because variable X blocks the influence between Z and U [29]. Similarly, we have $P(Y|X,Z,U) = P(Y|Z,U)$ for any X, and $P(U|X,O) = P(U|X)$ for any O. Based on these, we derive the following equation under the intervention $do(O)=o$,

$$\begin{aligned} &\int P(Y|Z,U)P(U)dU \\ &= \int P(Y|Z,U)P(U|X)P(X|do(O)=o)dXdU \\ &= \int P(Y|Z,U)P(U|X)P(X|o)dXdU \\ &= \int P(Y|X,Z,U)P(U|X,Z)P(X|o)dXdU \\ &= \int P(X|o)\big[\int P(Y|X,Z,U)P(U|X,Z)dU\big]dX \\ &= \int P(X|o)\big[\int P(Y,U|X,Z)dU\big]dX \\ &= \int P(X|o)P(Y|X,Z)dX \end{aligned} \tag{8}$$

By replacing the term $\int P(Y|Z,U)P(U)dU$ in Eq.6 by $\int P(X|o)P(Y|X,Z)dX$, we have:

$$P(Y|do(X=x)) = \int P(Z|x)\big[\int P(X|o)P(Y|X,Z)dX\big]dZ \tag{9}$$

which proves the validity of Theorem 1. □

B. Implementation Details

B.1. The Proposed CausalPC

We further present the implementation details of our proposed CausalPC.

CausalPC can be roughly divided into the structure extraction module, the point cloud reconstruction module, and the joint attention classification module, which are dedicated to modeling $P(Z|x)$, $P(X|o)$, and $P(Y|Z,X)$, respectively. For the *structure extraction module*, we sample 512 points for each point cloud as its structural information. For the *point cloud reconstruction module*, we leverage the generative network PU-Net g_ϕ [56] used in $P(X|o)$ is trained to up-sample a point cloud from 2048 points to 4096 points. We down-sample each dense point cloud to 2048 points as x $P(X|o)$. We further draw noise from a uniform distribution $U(-1,1)$ to simulate sampling noise. For the *joint attention classification module*, we adapt the single-head attention implementation used in [41], where learnable parameter matrices are first multiplied with the query, key, and value features before the attention score computation. The dimension of the transformed features

for attention computation is set to be 256. After attention-based aggregation, the global features h_z^{att} and h_x^{att} are first concatenated and fused by a linear layer. Then, the resulting feature is further fused with h_{zx} to achieve the overall point cloud feature h for classification, whose dimension aligns with the one used for the vanilla classifiers.

To mitigate the impact of extreme outliers within an observed point cloud, we incorporate the SOR algorithm as a pre-processing step before conducting causal effect identification. For the sampling times used in the structure extraction module and the point cloud reconstruction module, we set the hyperparameters $M_x = 10$ and $M_z = 10$. We finetune the attention module and the linear layers in the vanilla classifier for 10 epochs on the normal train set of each dataset with the learning rate set as 0.001. For other hyperparameters related to the network structure and training process of PU-Net and existing point cloud classifiers [19, 32, 45], we follow the specifications provided in the respective original works.

B.2. Experimental Settings

We further detail the background and the hyper-parameter settings of the baseline attack and defense strategies we used in our paper.

Attack Methods. We briefly summarize the attack methods involved in our work. Ten attack methods are considered in this paper, including Minimal [15], Smooth [25], IFGM [21], Gen3D-Add [51], Gen3D-Pert [51], AdvPC [11], Drop [62], KNN [40], GeoA3 [46], and ShapeInvariant (SI) [12]. Specifically, Minimal proposes to exploit the L_1 constraint to approximate the L_0 constraint, which tends to reduce the number of perturbed points. IFGM exploits normalized gradients to generate perturbations instead of the product of the sign of gradients and a perturbation clip threshold originally used in FGSM [9]. Smooth attempts to obtain more robust gradients by adding random noises to a point cloud before generating perturbations based on IFGM and averaging them. Gen3D-Pert and Gen3D-Add stand for the perturbation and adding attacks proposed in the original work, where Gen3D-Pert imposes small perturbations on all points of a point cloud under the constraint of L_2 distance, while Gen3D-Add imposes the perturbations on a copy of a separate subset of points from the input point cloud, considered as the added points, under the constraint of Chamfer distance. AdvPC generates an adversarial point cloud given the benign one with an autoencoder trained for a dataset with the L_2 distance constraint. Drop attack selects the most important set of points that influences the final extracted feature for a point cloud and deletes them. KNN is another perturbation attack with an additional k-NN distance constraint other than Chamfer distance and additional clip operations based on L_∞ norms and normal vectors. GeoA3 takes Chamfer distance, Hausdorff distance and local curvature loss into account when performing a perturbation attack. SI uses specific designs to restrict the direction of perturbations on points. The optimization of the perturbations concerning all the attacks above is based on the C&W attack [3] except for IFGM and Smooth.

We provide the hyper-parameter settings for the attacks as follows for potential reproduction needs. For all attacks, we first apply the settings from the original works if provided and then tune them for better attack performance on a small validation set. For Minimal, we set the attack step size as 0.02, the number of iterations as 20 for ModelNet40 and 45 for ShapeNet, and the perturbation clip threshold ϵ as 0.5. For Smooth, we set the attack step size as 0.02, the number of iterations as 100, the clip threshold as 0.5, and the random noise are sampled from a Uniform distribution $U_{[0,0.1]}$, the number of iterations for searching robust gradients is 15. For IFGM, we set the attack step size as 0.02, the number of iterations as 100, and the clip threshold ϵ as 0.5. For Gen3D-Pert, we set the attack step size as 0.001, the number of iterations as 200, and the distance loss weight as 0.1. For Gen3D-Add, we set the number of points added as 256, the attack step size as 0.001, the number of iterations as 400, and the distance loss weight as 400.0. For AdvPC, we set the number of iterations as 100, and the distance constraint weight as 0.1. For Drop, we set the total number of points to delete as 200, and the number of iterations as 20. For KNN, we set the attack step size as 0.001, the number of iterations as 400, the κ used in the C&W attack as 5.0, the Chamfer distance weight as 10.0, the k-NN distance weight as 0.5, and the clip threshold is 0.8. For GeoA3, we set the max binary search steps of attack step size as 10, the number of iterations as 300, and the weights of Chamfer distance, Hausdorff distance, and curvature loss are 10.0, 0.1, 1.0, respectively. For SI, we set the attack step size as 0.07, the number of iterations as 50, and the clip threshold as 0.16. All the attacks are performed with the settings above unless otherwise specified.

Defense Methods. We consider three input-oriented methods, i.e., SOR [36], DUP-Net [63], and GvG [7], as our defense baselines. SOR computes the k-NN distance for each point in a point cloud and removes those points with distances larger than $\mu + \alpha \cdot \sigma$, where μ and σ denote the mean and standard deviation of the distances. DUP-Net further utilizes an up-sampling network [55] to enhance the visual quality of a point cloud after SOR. For the hyper-parameters, we set k as 2 and α as 1.1 as described in the original work. GvG leverages the predicted gather vector for each point to recover the perturbed prediction. We apply GvG to PointNet only due to its model-specific design.

We also consider two adversarial training-based methods: Adversarial training (AT) [25] and PAGN [20]. For AT, we combined adversarial examples generated by the IFGM attack with benign samples for fine-tuning the vanilla clas-

Table 4. Mean test accuracy on ModelNet40 and ShapeNet.

		PointNet	DGCNN	PointCNN
ModelNet40	Vanilla	86.2%	88.9%	89.6%
	Ours	86.9%	87.3%	87.1%
ShapeNet	Vanilla	78.6%	80.5%	77.9%
	Ours	73.4%	77.1%	71.7%

sification model.

While we take these methods to compare the defense performance of our proposed framework, we would like to point out that our framework integrates a sampling strategy, a reconstruction module, and an attention module on existing classifiers, which is compatible with all the existing recovery-based defense methods, e.g., one can first utilize recovery methods to restore an adversarial example before inputting it into our model to further boost the robustness.

More Details. We first describe the design of the random targeted attack in the main body in detail. For each sample, we randomly assign a label other than the ground truth label as the target label and maximize the model's prediction of this label. For attack methods that are originally designed for performing untargeted attacks, e.g., IFGM and Minimal, we alter their classification loss to a targeted one, i.e., from maximizing the loss against the ground truth label to minimizing the loss against the target label. Further, we may follow the design in the C&W-based attack [3] and use the margin logit loss.

Under such a design of targeted attacks, the computation of the attack success rate (ASR) is defined as,

$$\text{ASR} = \frac{1}{N_{\text{test}}} \sum_{\tilde{X}} I(f_\theta(\tilde{X}), \tilde{y}) \tag{10}$$

where $\tilde{y}$ is the target label for each $\tilde{X}$ in the test set, N_{test} is the size of the testing set and $I(\cdot, \cdot)$ is the indicator function, where $I = 1$ if the model predicts the target label $\tilde{y}$ and otherwise 0. A lower ASR indicates better adversarial robustness.

As for the untargeted attack, the attack is set to aim at misleading the prediction of the model without any designated label, i.e., causing the model to predict any label other than the ground truth one. Therefore, we utilize classification accuracy to measure the effectiveness of an attack, where a higher accuracy indicates better adversarial robustness. Specifically, we take IFGM, Minimal, Gen3D-Pert, and Drop as representative attacks for the corresponding experiments.

For the implementation of the classification models and baseline methods, we directly apply those with released PyTorch implementations. For those without source code released or with TensorFlow implementations only, we implement them according to their work. We trained all the models ourselves including those used for classification and those for defense baselines.

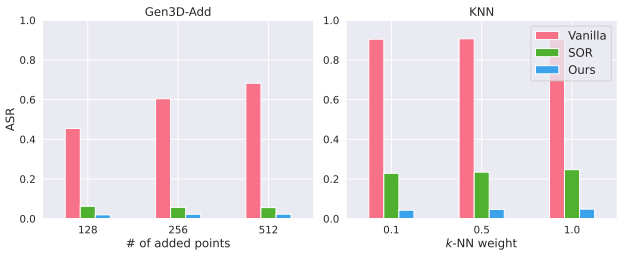

Figure 6. The influence of the attack hyper-parameters on the adversarial robustness. (Left: Gen3D-Add, Right: KNN)

All the attacks and defenses are performed on the test set of both datasets. Specifically, for time efficiency, we perform GeoA3 [46] attack only on a subset with 800 samples consisting of 20 samples from each of the 40 classes from ModelNet40. Note that in the original work, the authors sampled a subset of 250 pairs of samples and target labels in their experiments as well. We believe that our sampling strategy can provide us with a subset of samples representative enough.

All the experiments are conducted on a machine with a 32-core CPU, 128 GBs of memory, and 2 NVIDIA 2080Ti GPUs.

C. More Empirical Results

C.1. Normal Utility and Robustness of CausalPC

We first evaluate the classification accuracy of vanilla classifiers combined with CausalPC and summarize the results on two datasets in Table 4. The results state that the proposed CausalPC could achieve robustness against adversarial attacks without much loss of normal utility. Note that existing defenses, whether adversarial training-based or input-oriented, typically incur some degree of damage to the normal utility of the model. Certain randomized defense strategies exhibit a nearly 10% decrease in test set accuracy [23].

We further present the performance comparison of CausalPC with baseline defenses on ShapeNet in Table 5 and 6. The results show consistent performance with Table 1 and 2 in Section 5, where the proposed CausalPC substantially outperforms other defenses in various settings against different types of attacks.

C.2. Influences of Attack Intensities

To evaluate the adversarial robustness of point cloud classifiers with our CausalPC under various attack intensities, we adjust the hyper-parameters of specific attack methods to explore the resulting ASR. Specifically, we pick Gen3D-Add and KNN as the representatives of normal and shape-invariant attacks and perform them on PointNet with Mod-

Table 5. Classification accuracy (%) of untargeted attack strategies against different defense methods on ShapeNet dataset. The best among all defenses is in bold.

	PointNet							DGCNN						PointCNN					
	Vanilla	SOR	DUP-Net	GvG	AT	PAGN	Ours	Vanilla	SOR	DUP-Net	AT	PAGN	Ours	Vanilla	SOR	DUP-Net	AT	PAGN	Ours
IFGM	0.3	37.1	23.9	0.3	23.7	32.0	**57.5**	0.0	31.3	46.9	25.5	14.8	**65.4**	43.2	**66.1**	51.3	22.4	55.5	**66.1**
Minimal	26.6	51.2	51.1	26.8	42.9	59.3	**60.1**	23.2	48.9	53.9	43.6	49.3	**67.7**	62.8	**63.6**	54.8	22.8	60.9	61.8
Gen3D-Pert	40.8	40.3	40.3	40.8	36.9	41.4	**64.6**	18.5	20.2	**37.1**	11.7	24.7	32.6	43.0	44.1	42.7	16.1	39.5	**44.9**
Drop	57.4	**64.2**	10.0	52.9	24.5	63.6	59.7	68.3	**76.7**	4.1	37.9	70.4	74.1	**75.1**	71.3	9.5	43.8	66.1	70.5
Avg.	31.3	48.2	31.3	30.2	32.0	49.1	**60.5**	27.5	44.3	35.5	29.7	39.8	**59.9**	56.0	**61.3**	39.6	26.3	55.5	60.8

Table 6. Attack success rates (%) of targeted attack strategies against different defense methods on ShapeNet dataset. The best among all defenses is in bold.

	PointNet							DGCNN						PointCNN					
	Vanilla	SOR	DUP-Net	GvG	AT	PAGN	Ours	Vanilla	SOR	DUP-Net	AT	PAGN	Ours	Vanilla	SOR	DUP-Net	AT	PAGN	Ours
Minimal	29.6	6.0	5.6	12.0	6.7	2.8	**2.2**	8.6	1.8	1.0	2.4	2.1	**0.7**	**0.5**	0.6	0.7	0.9	0.9	0.8
Smooth	15.6	5.0	5.1	40.3	30.1	3.8	**1.7**	6.8	2.1	1.7	62.3	11.5	**0.9**	1.1	**0.7**	1.2	2.6	1.1	0.8
IFGM	61.2	4.1	3.2	54.1	57.6	2.2	**0.9**	85.4	1.4	0.6	95.0	1.0	**0.4**	2.8	**1.4**	1.7	3.7	1.5	**1.4**
Gen3D-Add	41.8	4.5	4.3	32.8	34.5	2.8	**1.2**	13.1	1.5	1.1	10.5	2.5	**0.6**	0.7	**0.6**	1.1	1.3	0.7	0.9
Gen3D-Pert	97.2	5.3	4.0	50.9	62.2	2.5	**0.9**	87.4	3.8	1.3	84.0	2.2	**0.4**	13.9	6.9	3.0	3.4	**2.2**	3.0
AdvPC	99.4	4.0	**1.5**	96.6	99.4	3.5	1.6	61.8	2.9	1.7	58.7	2.4	**0.7**	4.0	2.6	**1.6**	**1.6**	2.2	2.6
KNN	80.9	29.2	24.7	87.8	92.0	23.0	**3.7**	94.1	19.6	**1.3**	94.1	12.4	1.6	23.9	12.6	**4.8**	14.9	9.2	7.7
ShapeInvariant	48.6	5.2	2.2	47.8	45.0	5.0	**1.7**	8.3	2.1	1.6	4.2	2.6	**1.1**	3.3	3.0	1.9	1.9	1.9	**1.2**
Avg.	59.3	7.9	6.3	52.8	53.4	5.7	**1.7**	45.7	4.4	1.3	51.4	4.6	**0.8**	6.3	3.6	**2.0**	3.8	2.5	2.3

elNet40. For Gen3D-Add, we adjust the number of added points to be $128, 256, 512$, while for KNN, we adjust the weight of the k-NN distance loss to be $0.1, 0.5, 1.0$. The ASR of the vanilla PointNet, PointNet with SOR, and PointNet with CausalPC is demonstrated in Fig.6.

As is shown in the left subplot, for Gen3D-Add, the ASR of vanilla PointNet rises as the number of points added increases, while the one of PointNet with SOR defense slightly decreases, which indicates that for normal attacks, a stronger attack setting may reduce the stealthiness of the attack, resulting in the success of input-oriented defenses. As for KNN, the ASR of the vanilla classifier plateaus while decreasing the distance loss weight (, which in turn brings a stronger attack). In the meantime, the ASR of SOR defense slightly increases, which means that a tighter restriction of shape-invariant attacks produces more stealthy adversarial examples when the ASR saturates. For both types of attacks, the classifier with our CausalPC keeps the lowest ASR consistently, which further validates the adversarial robustness brought by the causal inference of the point cloud classification under various attack strengths.

C.3. Visualization of the Overall Framework

To gain an intuitive understanding of the source of adversarial robustness in our CausalPC, we visualize the generated z_* and x'_* in Fig.7. We randomly select an adversarial example produced by the GeoA3 attack on PointNet for the ModelNet40 dataset. The ground truth label of the example is *chair*, and the target label is *plant*. For simplicity, we only demonstrate the most representative 4 of the 10 generated z_* and x'_*. In the visualization, the first row/column represents x'_* / z_* individually, while the other positions display the concatenated sample for prediction. The title of each point cloud summarizes the top three predicted classes and their corresponding confidence.

This visualization offers a comprehensive understanding of the intuition behind our CausalPC. Firstly, it provides additional examples showcasing how the structural information extraction module and point cloud reconstruction module effectively capture the relevant information within a point cloud. Specifically, the column of z_* preserves the key structure of the chair, while the row of x'_* offers more detailed information. Secondly, the prediction results for each combined sample demonstrate that even if some of the generated x'_* or z_* still contain local adversarial patterns that could mislead the model's prediction, e.g., the X_3 column, the overall adversarial robustness is ensured through the aggregation of predictions.

D. More Discussion

D.1. Certified Defenses

Another line of work that seems similar to our work is certified defenses [6, 18], which share the motivation of defending against various potential noises with our work. Certified defenses for point cloud classification focus on defending against natural noises like 3D transformations [5, 28, 30] and adversarial noises [23, 60]. Though providing a theoretical robustness certification, the randomized smoothing strategy [6] they use has also brought their limitations. To satisfy the requirements of certified robustness, these approaches not only need to sample numerous point clouds for inference ($10,000$ times of inference for one point cloud), but also need to retrain the original classification model,

e.g., for classifying point clouds of only 16 points each [23]. Such a mechanism limits the availability of these methods to various kinds of classification models. In comparison, our CausalPC is a model-agnostic solution free of retraining the whole model. Moreover, our framework based on the causal framework requires significantly fewer samples during inference (100 times in total).

D.2. Adaptive Attacks

As a defense method, it is suggested to consider potential attacks when an attacker knows about the defense procedure. To perform an adaptive attack against CausalPC, the main challenge is to acquire the accurate gradient. Since CausalPC involves multiple randomization operations with up- and sub-sampling strategies, it becomes hard for an attacker to estimate the actual gradient that contributes to the final classification for each point within a sample. Moreover, both point-wise samplings (FPS, random) and sample-wise samplings (Eq.4 in the paper) themselves are the source of robustness. Only drastic changes in an attack can survive the samplings, which violates the imperceptibility requirement.

However, it is still possible for anyone to put the whole inference process of CausalPC together and perform backpropagation. Due to the 100 times of classification involved in one inference, such a backpropagation has a huge computational overhead for our experimental resource. Therefore, we briefly discuss this potential instead of evaluating it.

Figure 7. Visualization of the extracted z_* and the reconstructed x'_* and the concatenated samples for an observed x. Each row/column of samples shares the same z_* / x'_*. Samples with red titles are misclassified.