IF-DEFENSE: 3D ADVERSARIAL POINT CLOUD DEFENSE VIA IMPLICIT FUNCTION BASED RESTORATION

Ziyi Wu1,∗, Yueqi Duan2,∗,†, He Wang, Qingnan Fan2, Leonidas J. Guibas2
1Tsinghua University, 2Stanford University

ABSTRACT

Point cloud is an important 3D data representation widely used in many essential applications. Leveraging deep neural networks, recent works have shown great success in processing 3D point clouds. However, those deep neural networks are vulnerable to various 3D adversarial attacks, which can be summarized as two primary types: point perturbation that affects local point distribution, and surface distortion that causes dramatic changes in geometry. In this paper, we propose a novel 3D adversarial point cloud defense method leveraging implicit function based restoration (IF-Defense) to address both the aforementioned attacks. It is composed of two steps: 1) it predicts an implicit function that captures the clean shape through a surface recovery module, and 2) restores a clean and complete point cloud via minimizing the difference between the attacked point cloud and the predicted implicit function under geometry- and distribution-aware constraints. Our experimental results show that IF-Defense achieves the state-of-the-art defense performance against all existing adversarial attacks on PointNet, PointNet++, DGCNN and PointConv. Comparing with previous methods, IF-Defense presents 20.02% improvement in classification accuracy against salient point dropping attack and 16.29% against LG-GAN attack on PointNet. Our code is available at https://github.com/Wuziyi616/IF-Defense.

1 INTRODUCTION

Recent years have witnessed a growing popularity of various 3D sensors such as LiDAR and Kinect in self-driving cars, robotics and AR/VR applications. As the direct outputs of these sensors, point cloud has drawn increasing attention. Point cloud is a compact and expressive 3D representation, which represents a shape using a set of unordered points and can capture arbitrary complex geometry. However, the irregular data format makes point clouds hard to be directly processed by deep neural networks (DNNs). To address this, PointNet (Qi et al., 2017a) first uses multi-layer perceptrons (MLPs) to extract point-wise features and aggregate them with max-pooling. Since then, a number of studies have been conducted to design 3D deep neural networks for point clouds, such as PointNet++ (Qi et al., 2017b), DGCNN (Wang et al., 2019) and PointConv (Wu et al., 2019).

One limitation to DNNs is that they are vulnerable to adversarial attacks. By adding imperceptible perturbations to clean data, the generated adversarial examples can mislead victim models with high confidence. While numerous algorithms have been proposed in 2D attack and defense (Goodfellow et al., 2014b; Carlini & Wagner, 2017; Xie et al., 2017; Papernot et al., 2016; Moosavi-Dezfooli et al., 2016; Athalye et al., 2018; Moosavi-Dezfooli et al., 2017), only little attention is paid to its 3D counterparts (Xiang et al., 2019; Zhou et al., 2019; Zheng et al., 2019). They show that point cloud networks such as PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b) are also sensitive to adversarial examples, bringing security threat to those safety-critical applications. By carefully examine the attack methods, we summarize the effects of 3D adversarial attacks on point cloud from existing works into two aspects as shown in Figure 1:

1) Point perturbation changes the local geometry and point-wise sampling pattern, which moves the points either out of the surface to become noises or along the surface to change

* Equal contribution  † Corresponding author: duanyq19@stanford.edu
point distributions. This effect performs similarly to 2D adversarial attack, which adds noise over each pixel within a given budget to fool the classifier.

2) Surface distortion aims to modify the geometry of the point cloud more dramatically by either removing local parts or deforming the shape of the point cloud. In general, surface distortion is difficult to defend due to the significant change of the geometry, yet is also more perceptible by humans.

While some methods have been proposed in recent years for 3D adversarial defense (Zhou et al., 2019; Dong et al., 2020), they fail to simultaneously address both the two aspects. For example, DUP-Net (Zhou et al., 2019) uses a statistical outlier removal (SOR) pre-processor to address out-of-surface point perturbations, followed by an up-sampling network to generate denser point clouds. However, it cannot well recover the point distribution and restore the distorted surface. Gather-vector guidance (GvG) method (Dong et al., 2020) learns to ignore noisy local features, which fails to defend the attacks by local part removal. As a result, these methods fail to protect the victim models from all the attack methods, especially the latest ones, such as saliency point dropping (Zheng et al., 2019), LG-GAN (Zhou et al., 2020) and AdvPC (Hamdi et al., 2020).

In this paper, we propose a novel 3D adversarial point cloud defense algorithm named IF-Defense through implicit function based restoration, which is more universal and can simultaneously address both the attack effects. Figure 2 shows the pipeline of IF-Defense. We first employ SOR to pre-process the input point cloud following the existing work (Zhou et al., 2019). Inspired by the recent success in deep implicit functions which reconstruct accurate surfaces even under partial observations (Park et al., 2019; Duan et al., 2020; Mescheder et al., 2019; Peng et al., 2020; Chen & Zhang, 2019), we predict an implicit function that captures the clean shape using a learned deep implicit function network. Then, the defended point cloud is restored by optimizing the coordinates of the input points under the geometry-aware and distribution-aware constraints. The geometry-aware loss is enforced by the predicted implicit surface, which aims to remove out-of-surface geometric changes, such as Figure 1(b)(d)(e), while the distribution-aware loss aims to distribute points evenly and get rid of the on-surface point perturbation, as illustrated in Figure 1(c). Experimental results show that our IF-Defense consistently outperforms existing defense methods against various 3D adversarial attack methods for PointNet, PointNet++, DGCNN and PointConv.
Figure 2: The pipeline of our IF-Defense method. We first pre-process the input point cloud by SOR, and then we learn point cloud restoration via implicit function based optimization. Finally, we send the restored point cloud to the classifier. \textit{Pert.} and \textit{Distort.} indicate Perturbation and Distortion, while \textit{Geo.} and \textit{Dist.} mean Geometry and Distribution.

Table 1: Correspondence between existing 3D attacks and the attack effects. In the table, ✓ indicates the main effects of an attack while △ shows the less significant ones.

| Attack Type | Out-of-surface Pert. | On-surface Pert. | Part Removal | Geo. Deform. |
|-------------|----------------------|------------------|--------------|--------------|
| Perturb     | ✓                    | ✓                | ✓            | ✓            |
| Add         | ✓                    | ✓                | ✓            | ✓            |
| kNN         | ✓                    | ✓                | ✓            | ✓            |
| AdvPC       | ✓                    | ✓                | ✓            | ✓            |
| Drop        | ✓                    | ✓                | ✓            | ✓            |
| LG-GAN      | ✓                    | ✓                | ✓            | ✓            |

2 RELATED WORKS

Deep learning on point clouds. The pioneering work PointNet (Qi et al., 2017a) is the first deep learning algorithm that operates directly on 3D point clouds. After that, PointNet++ (Qi et al., 2017b) further improves the performance of PointNet by exploiting local information. Another representative work is Dynamic Graph CNN (DGCNN) (Wang et al., 2019), which constructs kNN graphs and applies EdgeConv to capture local geometric structures. In recent years, there are more and more convolution based methods proposed in the literature (Wu et al., 2019; Thomas et al., 2019; Hermosilla et al., 2018), which run convolutions across neighboring points using a predicted kernel weight. Though these point cloud networks have achieved promising results, they are vulnerable to adversarial attacks and require defense methods to improve the robustness.

3D adversarial attack. Existing 3D adversarial attack methods can be roughly divided into three classes: optimization based methods, gradient based methods and generation based methods. For optimization based methods, Xiang et al. (2019) first propose to generate adversarial point clouds using C&W attack framework (Carlini & Wagner, 2017) by point perturbation and adding. In contrast, Tsai et al. (2020) propose to add a kNN distance constraint, a clipping and a projection operation to generate adversarial point clouds that are resistant to defense. Besides, Hamdi et al. (2020) propose AdvPC by utilizing a point cloud auto-encoder (AE) to improve the transferability of adversarial examples. Because of the limited budget, these attacks mainly introduce point perturbations. For gradient based methods, Liu et al. (2019) extend the fast/iterative gradient method to perturb the point coordinates. Additionally, Zheng et al. (2019) develop a point dropping attack by constructing a gradient based saliency map, which would remove important local parts. LG-GAN (Zhou et al., 2020) is a generation based 3D attack method, which leverages GANs (Goodfellow et al., 2014a) to generate adversarial point clouds guided by the input target labels. We summarize the correspondence between existing 3D attacks and the attack effects in Table 1.
Figure 3: Comparison of the re-meshing and optimization based IF-Defense. Given the (a) input point cloud, the (b) reconstructed mesh using Marching Cubes according to the implicit field fails to capture the legs. As a result, the (c) re-sampled point cloud is misclassified as a monitor by PointNet. In contrast, the (d) optimized point cloud successfully retain the legs and is classified correctly.

3D adversarial defense. Liu et al. (2019) employ adversarial training to improve the robustness of models by training on both clean and adversarial point clouds. Yang et al. (2019) propose Gaussian noising and point quantization, which are adopted from 2D defense. They also introduce a Simple Random Sampling (SRS) method which samples a subset of points from the input point cloud. Recently, Zhou et al. (2019) propose a Statistical Outlier Removal (SOR) method that removes points with a large $k$NN distance. They also propose DUP-Net, which is a combination of SOR and a point cloud up-sampling network PU-Net (Yu et al., 2018). The non-differentiability of SOR also improves its robustness. Instead of designing a pre-processing module to recover adversarial examples, Dong et al. (2020) leverage the intrinsic properties of point clouds and develop a variant of PointNet++ (Qi et al., 2017b) that can identify and eliminate adversarial local parts of an input. Although these defenses are effective against simple attacks (Xiang et al., 2019), their performance against more complex methods (Tsai et al., 2020; Zhou et al., 2020) is relatively poor, which is because they fail to simultaneously address the aforementioned two attack effects.

Implicit representation. Different from the voxel-based, mesh-based and point-based methods that explicitly represent shape surface, implicit functions learn a continuous field and represent surface as the zeroth level-set. More recently, deep learning based methods use DNNs to approximate the occupancy field (Mescheder et al., 2019; Chen & Zhang, 2019) or signed distance function (Park et al., 2019; Michalkiewicz et al., 2019; Duan et al., 2020), which capture more complex geometries. Apart from their strong representation power, previous works show that implicit models encode shape priors in the decoder space, which are able to reconstruct complete shapes from partial observations (Park et al., 2019; Duan et al., 2020). Inspired by this, we propose an implicit function based restoration method to recover clean points from the attacked ones.

3 IF-DEFENSE

IF-Defense consists of two modules, namely surface recovery and point cloud restoration. For surface recovery, we train a deep implicit function network to represent shape surface implicitly. We adopt Occupancy Networks (ONet) (Mescheder et al., 2019) and Convolutional Occupancy Networks (ConvONet) (Peng et al., 2020) in our implementation as they are widely used in the literature. These networks are composed of an encoder, which takes as input a point cloud and outputs a latent code, and a decoder, which outputs implicit fields. Using the trained implicit function network, we obtain the implicit function of the point cloud pre-processed by SOR. As the implicit model is trained purely on clean data, the output space of the decoder lies in the complete and accurate shape manifold, which is beneficial to defend the attack of any out-of-surface geometric changes.

Given an implicit representation of the recovered surface, the next step is to restore the original clean point cloud, which can reverse the attack effects. An intuitive way is to explicitly reconstruct the shape as a mesh using Marching Cubes (Lorensen & Cline, 1987), and then sample from the
mesh using the same point sampling method as in training data to get a point cloud. However, the Marching Cubes algorithm completely relies on the predicted implicit field, which may contain certain errors. Previous studies show that some geometry such as slender parts of an object are difficult to be captured by implicit functions (Duan et al., 2020). Also, the noise in the attacked point cloud may lead to imprecise shape latent codes, which further enlarge the reconstruction errors. For example, ONet (Mescheder et al., 2019) fails to reconstruct the legs of a chair in Figure 3(b). As a result, the re-sampled point cloud in Figure 3(c) is misclassified by PointNet as a monitor.

Although ONet fails to reconstruct the chair legs as shown in Figure 3(b), this information is actually provided by the input point cloud in Figure 3(a). Inspired by this, we further propose an optimization based method which simultaneously exploits the information from both the input point cloud and implicit surface. More specifically, we first initialize the defense point cloud $\hat{X}$ as the input. Since the number of the input points may differ from the clean point clouds, we randomly duplicate points in $\hat{X}$ to maintain the same number of points as the training data. Then, instead of reconstructing meshes from the implicit field, we directly optimize the coordinates of $\hat{X}$ based on the predicted implicit surface with two losses: geometry-aware loss and distribution-aware loss.

**Geometry-aware loss** aims to encourage the optimized points to lie on the shape surface. At each time, we concatenate the latent code and the coordinate of point as the input of implicit function, where the output shows the predicted occupancy probability. Then, we employ the binary cross-entropy loss to force the optimized points to approach the surface as follows:

$$\mathcal{L}_S = \sum_{i=1}^{N} \mathcal{L}_{ce}(f_{\theta}(z, x_i), \tau),$$

where $z$ is the shape latent code extracted from the input point cloud, $x_i$ is the point coordinate to be optimized, and $N$ is the number of points. $f_{\theta}(z, x_i)$ is the implicit function that outputs the occupancy probability at location $x_i$. $\tau$ is a hyper-parameter controlling the object boundary, which is used as the ground-truth occupancy probability of surface. By minimizing the geometry-aware loss, we can drive the points in $\hat{X}$ towards the object surface.

**Distribution-aware loss** is used to maximize the distance from a point to its $k$-nearest neighbors ($k$NN), which encourages a more uniform point distribution:

$$\mathcal{L}_D = \sum_{i=1}^{N} \sum_{x_j \in knn(x_i, k)} -||x_i - x_j|| \cdot e^{-||x_i - x_j||^2/h^2},$$

where $knn(x_i, k)$ denotes the $k$NN of a point $x_i$. The exponential term especially punishes the points that are too close to each other, and $h$ is a hyper-parameter controlling the decay rate with respect to the distance. Similar penalization has also been proposed in the previous point up-sampling work (Yu et al., 2018), known as the repulsion loss. We optimize the point coordinates $x_i$ by minimizing the following objective function with a hyper-parameter $\lambda$ balancing the weights of two terms:

$$\mathcal{L}(\hat{X}) = \mathcal{L}_S + \lambda \mathcal{L}_D.$$  

**Implementation details.** We implemented the implicit function network with the widely-used ONet (Mescheder et al., 2019) and ConvONet (Peng et al., 2020) in IF-Defense, which are trained on multiple categories without class labels. We first pre-trained them on the ShapeNet dataset (Chang et al., 2015) and then finetuned them on the ModelNet40 dataset (Wu et al., 2015). For the optimization based IF-Defense, we used $\tau = 0.2$ as suggested by Mescheder et al. (2019). Parameters $h$ and $k$ were set to be 0.03 and 5 following Xu et al. (2018), and $\lambda$ was set as 500. We optimized the coordinates of points for 200 iterations using the Adam (Kingma & Ba, 2014) optimizer with a learning rate equals to 0.01. More implementation details are provided in the appendix.

4 Experiments

We conducted all the experiments on the ModelNet40 (MN40) dataset (Wu et al., 2015). ModelNet40 is a commonly used shape classification benchmark that contains 12,311 CAD models from 40 man-made object classes. We used the official split with 9,843 shapes for training and 2,468 for
testing. Following [Qi et al. 2017a], we uniformly sampled 1,024 points from the surface of each object and normalize them into a unit sphere. We applied PointNet [Qi et al. 2017a], PointNet++ [Qi et al. 2017b], DGCNN [Wang et al. 2019] and PointConv [Wu et al. 2019] as the victim models, with the single scale grouping (SSG) strategy for PointNet++ and PointConv.

For the attack methods, we employed the point perturbation and individual point adding attack (Xiang et al. 2019), kNN attack (I sai et al. 2020), point dropping attack (Zheng et al. 2019) as well as two recently proposed attacks LG-GAN (Zhou et al. 2020) and AdvPC (Hamdi et al. 2020). For the defense baselines, we employed SRS (Yang et al. 2019), SOR (Zhou et al. 2019) and DUP-Net (Zhou et al. 2019). We also trained a point cloud AE with a SOR pre-processor method as a baseline called SOR-AE. Following previous works, we tested on targeted attack and reported the classification accuracy after defense, where higher accuracy indicates better defense.

### 4.1 Comparison with the State-of-the-Art Methods

Table 2 and Table 3 illustrate the classification accuracy under various attack and defense methods on PointNet and PointNet++. In the Tables, Ours-Mesh and Ours-Opt represent the methods based on re-meshing and optimization respectively. We use † and ‡ to show the results of two implicit function networks ONet and ConvONet. We observe that the optimization based method consistently outperforms the re-meshing based method, which shows the effectiveness of leveraging the input points in generating defense point clouds. Also, employing ConvONet usually leads to better or comparable accuracy compared with ONet because of the stronger representation capacity of ConvONet. For perturbation and point adding attacks, IF-Defense achieves relatively small improvements compared with existing methods, because these attacks mainly lead to local out-of-surface perturbation and can be alleviated by the simple SOR. However, our method boosts the performance significantly for kNN, point dropping, LG-GAN and AdvPC attack. The reason is that these attacks mainly introduce on-surface perturbation or significant surface distortion, while IF-Defense can recover natural shape surface via implicit function network and restore point clouds with desired point distribution.

As shown in Table 4 and Table 5 we draw similar observations for DGCNN and PointConv. The optimization based IF-Defense still outperforms its re-meshing based counterpart, and ConvONet demonstrates competitive or superior performance compared with ONet. It is worth noticing that DUP-Net performs poorly on these two models. DGCNN and PointConv are sensitive to local point distributions as they extract and propagate features through kNN graphs. However, DUP-Net up-samples points to a much higher density using PU-Net, which largely affects the learned local kNN graphs due to the difference in point distributions. Instead, the proposed IF-Defense
Table 4: Classification accuracy of MN40 under various attack and defense methods on DGCNN.

| Defenses   | Clean   | Perturb | Add-CD | Add-HD | kNN | Drop-100 | Drop-200 | LG-GAN | AdvPC |
|------------|---------|---------|--------|--------|-----|----------|----------|--------|-------|
| No defense | 91.49%  | 0.00%   | 1.46%  | 1.42%  | 20.02% | 75.16%   | 55.06%   | 15.41% | 9.23% |
| SRS        | 81.32%  | 50.20%  | 63.82% | 43.35% | 41.25% | 49.23%   | 23.82%   | 20.07% | 41.62%|
| SOR        | 88.61%  | 76.50%  | 72.53% | 63.74% | 55.92% | 64.68%   | 59.36%   | 30.82% | 56.49%|
| SOR-AE     | 89.20%  | 79.03%  | 76.38% | 66.25% | 56.78% | 66.78%   | 63.70%   | 32.96% | 58.67%|
| DUP-Net    | 53.54%  | 42.67%  | 44.94% | 33.02% | 35.45% | 44.45%   | 36.02%   | 21.38% | 29.38%|
| Ours-Mesh  | 83.91%  | 81.56%  | 81.73% | 67.50% | 79.38% | 78.97%   | 70.34%   | 46.09% | 65.54%|
| Ours-Opt†  | 88.25%  | 82.25%  | 81.77% | 67.75% | 82.29% | 79.25%   | 73.30%   | 53.08% | 76.01%|
| Ours-Opt‡  | 89.22%  | 85.53%  | 84.20% | 72.93% | 82.33% | 83.43%   | 73.22%   | 50.70% | 79.14%|

Table 5: Classification accuracy of MN40 under various attack and defense methods on PointConv.

| Defenses   | Clean   | Perturb | Add-CD | Add-HD | kNN | Drop-100 | Drop-200 | LG-GAN | AdvPC |
|------------|---------|---------|--------|--------|-----|----------|----------|--------|-------|
| No defense | 88.49%  | 0.00%   | 0.54%  | 0.68%  | 3.12% | 77.96%   | 64.02%   | 4.42%  | 6.45% |
| SRS        | 85.23%  | 76.22%  | 71.31% | 61.98% | 55.75% | 69.45%   | 48.87%   | 5.10%  | 37.62%|
| SOR        | 87.28%  | 79.25%  | 82.41% | 72.73% | 26.13% | 77.63%   | 63.78%   | 5.48%  | 51.75%|
| SOR-AE     | 87.40%  | 78.08%  | 77.27% | 74.55% | 56.50% | 72.45%   | 60.37%   | 8.64%  | 50.96%|
| DUP-Net    | 78.73%  | 68.84%  | 72.61% | 61.14% | 43.76% | 70.75%   | 58.23%   | 5.02%  | 49.35%|
| Ours-Mesh  | 82.78%  | 81.74%  | 81.85% | 75.61% | 77.15% | 75.97%   | 68.44%   | 15.46% | 53.81%|
| Ours-Opt†  | 86.10%  | 83.55%  | 83.95% | 76.86% | 80.47% | 78.85%   | 70.34%   | 18.78% | 61.77%|
| Ours-Opt‡  | 88.21%  | 86.67%  | 85.62% | 82.13% | 81.08% | 81.20%   | 74.51%   | 16.55% | 59.82%|

optimizes towards uniform point distribution, which leads to better kNN graphs. Therefore, we achieve significantly better results than DUP-Net against all the attacks on DGCNN and PointConv.

4.2 Ablation Study

In this subsection, we study the effect of the hyper-parameter \( \lambda \) of the optimization based method (Ours-Opt), where ConvONet is adopted as it achieves the best performance against most of the attacks. We varied \( \lambda \) between 0 and 1,000 and recorded the accuracy of the victim models after defense. As shown in Figure 4, with the increase of \( \lambda \), the accuracy first improves and then begins to decrease. In most cases, we observe that the best accuracy is achieved at \( \lambda = 500 \). The distribution-aware loss enforces the points to distribute uniformly over the surface. The points are not able to cover the entire object uniformly with a small \( \lambda \), while a large \( \lambda \) fails to capture the surface precisely due to the ignorance of the geometry-aware loss. To this end, we select a proper \( \lambda \) to balance the importance between accurate surfaces and uniform point distributions.

4.3 Black-Box Attacks and Defenses

We explore the transferability of attacks and the performance of various defense methods in this black-box setting. We first generated adversarial examples against PointNet, and then transferred them to the other three victim models. We adopted the optimization based IF-Defense with ConvONet for comparison. The results are summarized in Table 6. As the attacked point clouds are generated against PointNet, they are less effective for other network architectures due to the limited transferability. We observe that our method consistently outperforms other defense methods. For SOR and DUP-Net, the classification accuracy even drops in some situations compared with directly

Figure 4: The ablation study of the hyper-parameter \( \lambda \). We show the defense accuracy of four victim models against point perturbation attack.
Table 6: Classification accuracy of black-box attacks and defenses.

| Network  | Defense      | Add-CD | kNN  | Drop | LG-GAN | AdvPC |
|----------|--------------|--------|------|------|--------|-------|
| PointNet | No defense   | 0.00%  | 8.51 | 40.24% | 4.40%  | 0.00% |
|          | SOR          | 87.66% | 80.47% | 79.90% | 23.18% | 70.07% |
|          | DUP-Net      | 87.12% | 84.04% | 73.06% | 50.90% | 72.94% |
|          | Ours-Opt‡    | 88.17% | 85.98% | 79.38% | 54.85% | 80.59% |
| PointNet++ | No defense   | 78.24% | 80.19% | 73.14% | 35.12% | 74.54% |
|          | SOR          | 85.58% | 87.16% | 66.57% | 40.23% | 78.49% |
|          | DUP-Net      | 53.20% | 49.47% | 35.01% | 20.35% | 38.77% |
|          | Ours-Opt‡    | 88.09% | 88.01% | 76.90% | 62.13% | 85.61% |
| DGCNN    | No defense   | 84.81% | 77.11% | 76.26% | 22.41% | 64.19% |
|          | SOR          | 84.57% | 82.43% | 72.41% | 47.52% | 70.89% |
|          | DUP-Net      | 79.74% | 75.20% | 57.37% | 32.15% | 66.78% |
|          | Ours-Opt‡    | 87.76% | 86.55% | 77.19% | 56.25% | 76.69% |
| PointConv | No defense   | 85.58% | 87.16% | 66.57% | 40.23% | 78.49% |
|          | SOR          | 85.57% | 82.43% | 72.41% | 47.52% | 70.89% |
|          | DUP-Net      | 79.74% | 75.20% | 57.37% | 32.15% | 66.78% |
|          | Ours-Opt‡    | 87.76% | 86.55% | 77.19% | 56.25% | 76.69% |

Figure 5: Visualization results of different defense results. The labels under each point cloud are the prediction outputs of the victim models.

using the noisy point cloud. Instead, our IF-Defense continuously boosts the performance, which demonstrates its effectiveness and robustness.

4.4 Qualitative Results

Figure 5 illustrates two groups of defense results using SOR, DUP-Net and all the three variants of IF-Defense. The first row shows the results under point dropping attack on PointNet, where the head of the airplane is discarded in the adversarial example. SOR fails to defend the point dropping attack because it just removes more points from the point cloud. Although DUP-Net further up-samples the point cloud with PU-Net, it only depends on its input point cloud so that the missing part cannot be recovered. Instead, all three IF-Defense methods successfully restore the shape by extending the front end and trying to generate a head, which demonstrates its effectiveness in reconstructing the whole shapes under partial observations. The second row is the kNN attack on PointConv. Most of the points are perturbed along the surface because of the kNN constraint, resulting in significant changes in point distribution. DUP-Net fails to recover the original point distribution as it outputs a much denser point cloud. Ours-Mesh re-samples points from the reconstructed mesh, which is able to maintain the similar point distribution as the clean one, and Ours-Opt outputs uniformly distributed points. Consequently, PointConv correctly classifies the airplane in both cases.
5 Conclusion

In this paper, we have proposed a general defense framework called IF-Defense for adversarial defense in 3D point cloud, which simultaneously addresses both key attack effects including point perturbation and surface distortion. Our IF-Defense restores the attacked point cloud by predicting an implicit representation of the clean shape and then optimizing the point coordinates according to the geometry-aware loss and the distribution-aware loss, so that the distorted surfaces are recovered by the implicit functions and the perturbed points are denoised through optimization. Extensive experiments show that IF-Defense consistently outperforms existing adversarial defense methods against various attacks on PointNet, PointNet++, DGCNN and PointConv.

Acknowledgements

This research was supported by a Vannevar Bush Faculty Fellowship, a grant from the Stanford-Ford Alliance, NSF grant IIS-1763268, and gifts from Adobe, Amazon AWS, and Snap, Inc.

References

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning Representations and Generative Models for 3D Point Clouds. In ICML, pp. 40–49, 2018.

Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing Robust Adversarial Examples. In ICML, pp. 284–293, 2018.

Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. In 2017 IEEE Symposium on Security and Privacy (SP), pp. 39–57, 2017.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An Information-Rich 3D Model Repository. arXiv preprint arXiv:1512.03012, 2015.

Zhiqin Chen and Hao Zhang. Learning Implicit Fields for Generative Shape Modeling. In CVPR, pp. 5939–5948, 2019.

Xiaoyi Dong, Dongdong Chen, Hang Zhou, Gang Hua, Weiming Zhang, and Nenghai Yu. Self-Robust 3D Point Recognition via Gather-Vector Guidance. In CVPR, pp. 11513–11521, 2020.

Yueqi Duan, Haidong Zhu, He Wang, Li Yi, Ram Nevatia, and Leonidas J Guibas. Curriculum DeepSDF. 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In NeurIPS, pp. 2672–2680, 2014a.

Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Álvar Vinacua, and Timo Ropinski. Monte Carlo Convolution for Learning on Non-Uniformly Sampled Point Clouds. In ECCV, September 2020.

William E Lorensen and Harvey E Cline. Marching Cubes: A high resolution 3D surface construction algorithm. SIGGRAPH, 21(4):163–169, 1987.
Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. In CVPR, pp. 4460–4470, 2019.

Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Implicit Surface Representations As Layers in Neural Networks. In ICCV, pp. 4743–4752, 2019.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In CVPR, pp. 2574–2582, 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal Adversarial Perturbations. In CVPR, pp. 1765–1773, 2017.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. In 2016 IEEE Symposium on Security and Privacy (SP), pp. 582–597, 2016.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In CVPR, pp. 165–174, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalie Gimelshein, Luca Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In NeurIPS, pp. 8026–8037, 2019.

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional Occupancy Networks. In ECCV, 2020.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In CVPR, pp. 652–660, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In NeurIPS, pp. 5099–5108, 2017b.

Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In ICCV, pp. 6411–6420, 2019.

Tzungyu Tsai, Kaichen Yang, Tsung-Yi Ho, and Yier Jin. Robust Adversarial Objects against Deep Learning Models. In AAAI, volume 34, pp. 954–962, 2020.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic Graph CNN for Learning on Point Clouds. TOG, 38(5):1–12, 2019.

Wenxuan Wu, Zhongang Qi, and Li Fuxin. PointConv: Deep Convolutional Networks on 3D Point Clouds. In CVPR, pp. 9621–9630, 2019.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In CVPR, pp. 1912–1920, 2015.

Chong Xiang, Charles R Qi, and Bo Li. Generating 3D Adversarial Point Clouds. In CVPR, pp. 9136–9144, 2019.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating Adversarial Effects Through Randomization. arXiv preprint arXiv:1711.01991, 2017.

Jiancheng Yang, Qiang Zhang, Rongyao Fang, Bingbing Ni, Jinxian Liu, and Qi Tian. Adversarial Attack and Defense on Point Sets. arXiv preprint arXiv:1902.10899, 2019.

Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-Net: Point Cloud Upsampling Network. In CVPR, pp. 2790–2799, 2018.
Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. PointCloud Saliency Maps. In *ICCV*, pp. 1598–1606, 2019.

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 *ICCV*, pp. 1961–1970, 2019.

Hang Zhou, Dongdong Chen, Jing Liao, Kejiang Chen, Xiaoyi Dong, Kunlin Liu, Weiming Zhang, Gang Hua, and Nenghai Yu. LG-GAN: Label Guided Adversarial Network for Flexible Targeted Attack of Point Cloud Based Deep Networks. In *CVPR*, pp. 10356–10365, 2020.
A  APPENDIX

A.1  MORE IMPLEMENTATION DETAILS ABOUT IF-DEFENSE

A.1.1  TRAINING OF IMPLICIT FUNCTION NETWORKS

We trained the implicit function networks in the reconstruction task, and then employed them in our IF-Defense. Occupancy Network (ONet) (Mescheder et al., 2019) and Convolutional Occupancy Network (ConvONet) (Peng et al., 2020) are adopted in IF-Defense. In this section, we describe the datasets and training schedules used in our experiments.

Datasets. We used two datasets to train the implicit function networks. For the ShapeNet dataset (Chang et al., 2015), we directly used the processed data provided by Mescheder et al. (2019), which contains point clouds sampled from each 3D object and their corresponding ground-truth occupancy values. For the ModelNet40 dataset (Wu et al., 2015), we utilized the pre-processing pipeline provided by Mescheder et al. (2019) to generate training data. We also subdivided the training set of ModelNet40 into a training and a validation subset on which we tracked the loss of the models to determine when to stop training. It is worth noticing that, we only trained the implicit function networks on the training set of ModelNet40, so neither the clean point clouds in the test set nor the adversarial point clouds generated by the attacks are leveraged during training.

Training schedules. We adopted the same training schedules as their original implementations (e.g. the Adam (Kingma & Ba, 2014) optimizer with a constant learning rate equals to $1e^{-4}$) except for two aspects. The first one is that we performed random rotation along the z-axis as data augmentation because the object orientations are not aligned in the ModelNet40 dataset. The second aspect is that we set the number of input points as 600 in ConvONet (Peng et al., 2020) compared with the original 3000 since the adversarial point clouds often consist of only 1024 points. The training of ONet took for around 2500k iterations on ShapeNet and 350k iterations on ModelNet40. For ConvONet, we trained for around 1000k and 200k iterations on the two datasets respectively.

A.1.2  HYBRID TRAINING OF VICTIM MODELS

We applied PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), DGCNN (Wang et al., 2019) and PointConv (Wu et al., 2019) as the victim models in our experiments. However, we discovered that the victim models trained purely using clean data degraded the accuracy by up to 5% when tested on clean point clouds with the IF-Defense module, which is unacceptable. In order to eliminate the domain gap between clean point clouds and their defense counterparts, we trained our victim models on both clean and defense point clouds (we call this hybrid training) and used them to evaluate the performance of IF-Defense.

We argue that this setting is valid in our task. On one hand, the attacks are still conducted against the hybrid trained models under the white-box setting, and the post-attack model accuracy only differs less than 2% for each attack. On the other hand, the hybrid training can be regarded as a simple data augmentation. It is completely different from adversarial training (Liu et al., 2019) because the victim models never utilize the adversarial examples during hybrid training.

A.2  BASELINES

In order to conduct fair comparisons between IF-Defense and baseline methods, we re-implemented all the attacks (Xiang et al., 2019; Zheng et al., 2019; Tsai et al., 2020) and defenses (Yang et al., 2019; Zhou et al., 2019) in PyTorch (Paszke et al., 2019) (except for LG-GAN (Zhou et al., 2020) and AdvPC (Hamdi et al., 2020) that we modified their official implementations to fit in with our experimental settings). Here we detail some of the hyper-parameters and settings in our experiments.

A.2.1  ATTACKS

Except for point dropping attack (Zheng et al., 2019) that cannot targetedly attack the victim models, we conducted targeted attack for all the other methods. We randomly assigned a target label unequal to the ground-truth class for every example in the ModelNet40 (Wu et al., 2015) test set, and this target label assignment was kept unchanged in all the attacks to eliminate the effect of randomness.
Perturb. We followed Xiang et al. (2019) and used the C&W attack framework with per-point $L_2$ norm as the perturbation metric. We performed 10-step binary search with 500 iterations in each step.

Add. We followed Xiang et al. (2019) and used the C&W attack framework with Chamfer or Hausdorff measurements as the perturbation metrics (dubbed Add-CD and Add-HD respectively). 512 points were added and perturbed during optimization while the original points in the point cloud remain unchanged. We performed 10-step binary search with 500 iterations in each step.

$k$NN. We followed Tsai et al. (2020) and used the C&W attack framework with Chamfer measurement and $k$-nearest neighbors ($k$NN) distance as the perturbation metrics. We applied the same clipping and projection operations as their original paper. The optimization iterations were 2,500 in the attack.

Drop. We dropped 100 or 200 points in this attack (dubbed Drop-100 and Drop-200 respectively). Following Zheng et al. (2019), we used a greedy algorithm that iteratively calculated the saliency map of all the remaining points and discarded 5 points with the highest saliency scores.

LG-GAN. We modified the online official implementation$^1$ to perform targeted attack according to our pre-assigned target labels. The hyper-parameters were the same as Zhou et al. (2020) for all four victim models.

AdvPC. We modified the online official implementation$^2$ to perform targeted attack according to our pre-assigned target labels. Following Hamdi et al. (2020), 2 different initializations for the optimization were used and the number of iterations was 200 in each optimization. We used the default hyper-parameters provided in their official implementation.

A.2.2 DEFENSES

SRS. We randomly dropped 500 points from the input point cloud as done by Zhou et al. (2019).

SOR. We trimmed the points in a point cloud $X$ according to $X' = \{x_i | d_i \leq \mu_d + \alpha \cdot \sigma_d\}$, where $\mu_d$ and $\sigma_d$ are the mean and standard deviation of the $k$NN distance of all points in $X$. We used $k = 2$ and $\alpha = 1.1$ as Zhou et al. (2019).

DUP-Net. For the SOR pre-processor, we used $k = 2, \alpha = 1.1$ as in the individual SOR defense. For the PU-Net (Yu et al., 2018), we trained it on Visionair dataset (Yu et al., 2018) using Chamfer distance loss and repulsion loss as Zhou et al. (2019). The up-sampling rate was 4 and we used the same training schedules. We also tested further finetuning PU-Net on the ModelNet40 training set but observed no performance gain.

SOR-AE. Hamdi et al. (2020) leveraged a point cloud auto-encoder (AE) (Achlioptas et al., 2018) as a baseline defense to evaluate the effectiveness of their attack. They showed that this defense can even outperform DUP-Net against some attacks. Therefore, we also adopted point cloud AE as a baseline in our experiments. Additionally, we discovered that adding a SOR pre-processor can further improve its robustness, which is the SOR-AE defense we utilized. We still set $k$ and $\alpha$ as 2 and 1.1 in the SOR pre-processor. And the point cloud AE is pre-trained on the ShapeNet dataset in the reconstruction task as Hamdi et al. (2020).

A.3 ADDITIONAL EXPERIMENTAL RESULTS

In this section, we present additional quantitative and qualitative experimental results. We first conduct more ablation studies on the hyper-parameter $\lambda$ in the optimization objective. Then, we show more visualizations of the defense point clouds to compare different defense methods.

A.3.1 MORE EFFECTS OF HYPER-PARAMETER $\lambda$

We examine the effects of hyper-parameter $\lambda$ more extensively by testing against more attacks and employing another implicit function network ONet. Figure 6 shows the results for ConvONet based restoration. For most of the attacks, we observe that the accuracy improves as we increase $\lambda$ until

$^1$https://github.com/RyanHangZhou/LG-GAN
$^2$https://github.com/ajhamdi/AdvPC
reaching its optimum at $\lambda = 500$. For larger $\lambda$, the accuracy becomes worse again. We draw similar observations when adopting ONet as the implicit function network from Figure 7, where $\lambda = 500$ is the optimal value for most of the attacks on all four victim models.

In conclusion, the selection of an appropriate balancing weight $\lambda$ is critical for our defense method, and we select an optimal $\lambda = 500$ to balance the importance between accurate shape surfaces and uniform point distributions.

### A.3.2 More Visualization Results

We present more visualization results in Figure 8 to compare the defense outputs of different methods against various attacks. For LG-GAN attack on PointNet, all three variants of IF-Defense successfully recover the straight body of the airplane. For point dropping attack on PointNet++, they re-introduce the lost leg of the table. In contrast, neither SOR nor DUP-Net can achieve such restorations. For point adding attack under Chamfer distance constraint on DGCNN, the re-meshing based
Figure 8: Visualizations of different defense results. The labels under each point cloud are the prediction outputs of the victim models.

IF-Defense fails to reconstruct the entire legs and their connection parts, whereas two optimization based IF-Defense succeed. For $k$NN attack on PointConv, DUP-Net outputs a lamp with a deformed base and denser points comparing to the clean one, thus being classified as a stair by the victim model. On the contrary, our methods restore point clouds with either original or uniform distributions. The last row shows the results of AdvPC attack on PointConv, where both re-meshing and optimization based IF-Defense using ONet fail to represent the correct object surface. In contrast, because of the stronger representation capacity, IF-Defense with ConvONet retains the shape of a bottle, thus being correctly classified by PointConv.