

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

Anonymous authors

Paper under double-blind review

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 implementation¹ 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 implementation² 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 μ_d and σ_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 α as 2

¹<https://github.com/RyanHangZhou/LG-GAN>

²<https://github.com/ajhamdi/AdvPC>

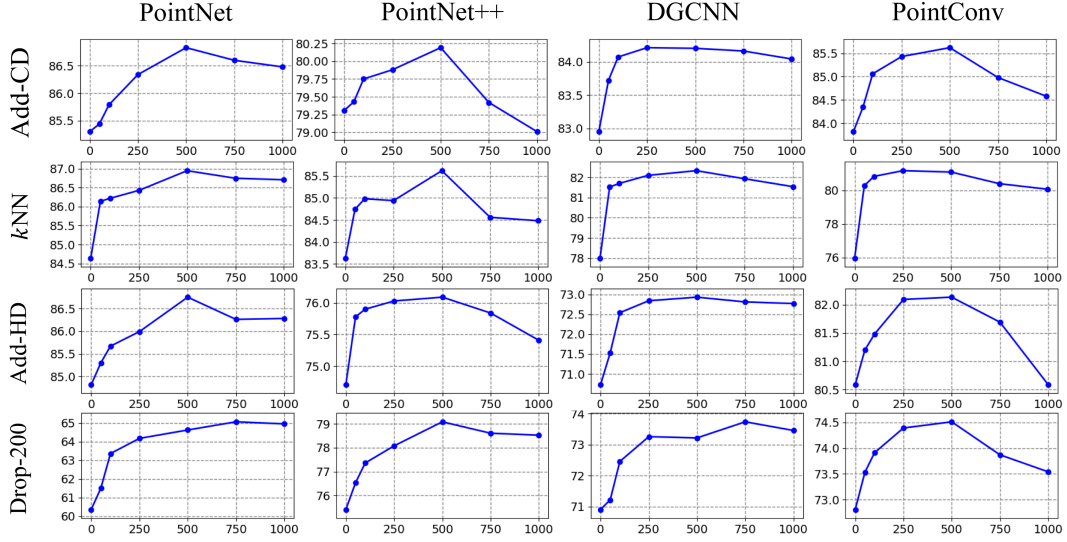


Figure 1: Ablation study w.r.t. hyper-parameter λ , where we show the defense accuracy of four victim models against point adding attack with Chamfer or Hausdorff measurement, k NN attack and point dropping attack. The defense evaluated here is our optimization based restoration with ConvONet.

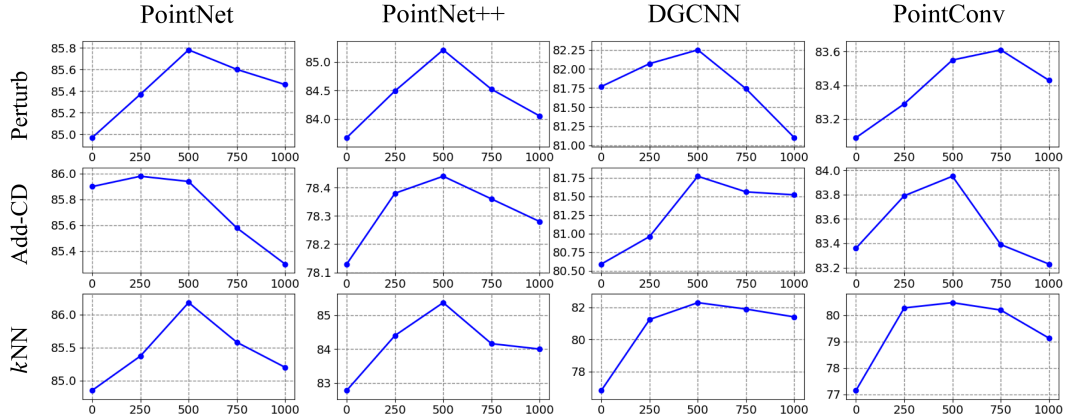


Figure 2: Ablation study w.r.t. hyper-parameter λ , where we show the defense accuracy of four victim models against point perturbation attack, point adding attack with Chamfer measurement and k NN attack. The defense evaluated here is our optimization based restoration with ONet.

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 λ 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 λ

We examine the effects of hyper-parameter λ more extensively by testing against more attacks and employing another implicit function network ONet. Figure 1 shows the results for ConvONet based restoration. For most of the attacks, we observe that the accuracy improves as we increase λ until

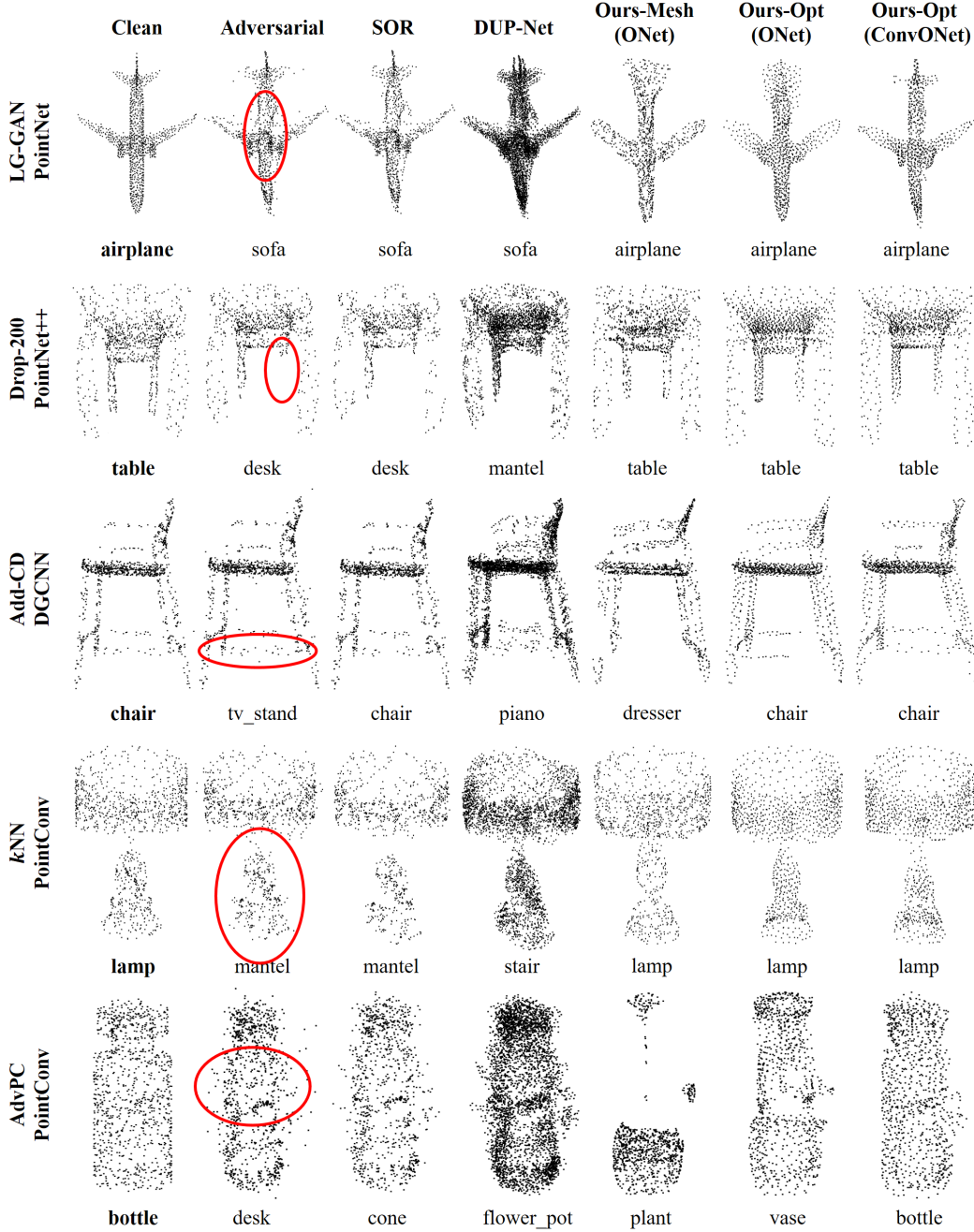


Figure 3: Visualizations of different defense results. The labels under each point cloud are the prediction outputs of the victim models.

reaching its optimum at $\lambda = 500$. For larger λ , the accuracy becomes worse again. We draw similar observations when adopting ONet as the implicit function network from Figure 2, 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 λ 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 3 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 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.

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.
- 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.
- Abdullah Hamdi, Sara Rojas, Ali Thabet, and Bernard Ghanem. AdvPC: Transferable Adversarial Perturbations on 3D Point Clouds. In *ECCV*, September 2020.
- Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Daniel Liu, Ronald Yu, and Hao Su. Extending Adversarial Attacks and Defenses to Deep 3D Point Cloud Classifiers. In *ICIP*, pp. 2279–2283, 2019.
- 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.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia 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.
- 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.
- 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.