On Adversarial Robustness of 3D Point Cloud Classification under Adaptive Attacks – Appendix

Jiachen Sun[1]
jiachens@umich.edu
Karl Koenig[1]
kamako@umich.edu
Yulong Cao[1]
yulongc@umich.edu
Qi Alfred Chen[2]
alfchen@uci.edu
Z. Morley Mao[1]
zmao@umich.edu

[1] Computer Science & Engineering
University of Michigan
Ann Arbor, MI, USA

[2] Department of Computer Science
UC Irvine
Irvine, CA, USA

A Adaptive Attack Experimental Setup & Visualization

A.1 Experimental Setup

Since **DUP-Net** [22] is open-sourced, we target the publicly released PointNet and PU-Net models. For the L^2 norm-based C&W attack [3], we set the loss function as:

$$\mathcal{L} = (\max_{i \neq t'}(\mathcal{Z}(\boldsymbol{X}')_i) - \mathcal{Z}(\boldsymbol{X}')_{t'})^+ + \lambda \cdot ||\boldsymbol{X} - \boldsymbol{X}'||_2 \tag{1}$$

where $\boldsymbol{X} \in \mathbb{R}^{n\times 3}$ is the matrix version of point cloud $\mathbb{X}$, $\boldsymbol{X}'$ is the optimized adversarial example, $\mathcal{Z}(\boldsymbol{X})_i$ is the i-th element of the output logits, and t' is the target class. We leverage 10-step binary search to find the appropriate hyper-parameter λ from $[10, 80]$. As suggested by [20], we choose 10 distinct classes and pick 25 objects in each class from the ModelNet40 validation set for evaluation. The step size of the adversarial optimization is 0.01 and we allow at most 500 iterations of optimization in each binary search to find the adversarial examples.

For the L^∞ norm-based PGD attack, we adopt the formulation in [9]:

$$\boldsymbol{X}_{t+1} = \Pi_{\boldsymbol{X}+\mathcal{S}}(\boldsymbol{X}_t + \alpha \cdot \text{sign}(\nabla_{\boldsymbol{X}_t}\mathcal{L}(\boldsymbol{X}_t, \boldsymbol{\theta}, \boldsymbol{y}))) \tag{2}$$

where $\boldsymbol{X}_t$ is the adversarial example in the t-th attack iteration, Π is the projection function to project the adversarial example to the pre-defined perturbation space $\mathcal{S}$, which is the L^∞ norm ball in our setup, and α is the step size. The sign() function also normalizes the perturbation into the L^∞ norm ball for each iteration. We select the boundary of allowed perturbations

$\varepsilon = \{0.01, 0.025, 0.05, 0.075\}$ out of the point cloud data range $[-1, 1]$. Since point cloud data is continuous, we set the step size $\alpha = \frac{\varepsilon}{10}$.

For **GvG-PointNet++** [4], we train it based on the single scale grouping (SSG)-PointNet++ backbone. The backbone network has three PointNet set abstraction module to hierarchically aggregate local features, and we enable gather vectors in the last module, which contains 128 local features (*i.e.,* $n' = 128$ in Section 3.2) with 256 dimensions. To learn the gather vectors, we apply three fully connected layers with 640, 640, and 3 hidden neurons respectively, as suggested by Dong *et al.* [4]. Since the data from ModelNet40 is normalized to [-1,1], the global object center is $\boldsymbol{c}_g = [0, 0, 0]$.

For the L^∞ norm-based PGD attack, we leverage the same setup as the attack on DUP-Net. For the L^2 norm-based PGD attack, we follow the settings in [4] to set the L^2 norm threshold $\varepsilon = \delta\sqrt{n \times d_{in}}$, where δ is selected in $\{0.08, 0.16, 0.32\}$, n is the number of points, and d_{in} is the dimension of input point cloud (*i.e.,* 3). The attack iteration is set to 50, and the step size $\alpha = \frac{\varepsilon}{50}$.

A.2 Visualization

We visualize some adversarial examples generated by adaptive attacks on PU-Net and DUP-Net in Figure 1 and Figure 2. It is expected that adversarial examples targeting DUP-Net are noisier than the ones targeting PU-Net as the former needs to break the denoiser layer. However, as mentioned in Section 3.1, they are barely distinguishable from human perception. We also visualize some adversarial examples generated by untargeted adaptive PGD attacks on GvG-PointNet++ in Figure 3 with different perturbation budgets ε.

B Adversarial Training Setup

B.1 PGD Attack in Adversarial Training

We also follow the formulation in Equation 8 to find the worst adversarial examples in adversarial training. Specifically, we empirically select $\varepsilon = 0.05$ into the training recipe as there is no quantitative study on how much humans can bear the point cloud perturbations. Figure 3 shows that adversarial examples with $\varepsilon = 0.05$ are still recognizable by human perception. Moreover, because point cloud data is continuous, we set the step size of PGD attacks as:

$$\alpha = \begin{cases} \dfrac{\varepsilon}{\text{step}}, & \text{step} < 10 \\ \dfrac{\varepsilon}{10}, & \text{step} \geq 10 \end{cases} \tag{3}$$

in both training and evaluation phases to make sure PGD attacks reach the allowed maximum perturbations.

B.2 Point Cloud Classification Model Architecture Detail

PointNet [11], DeepSets [21], and DSS [10] are the fundamental architectures in point cloud classification. Other models, such as PointNet++ [12] and DGCNN [18], are built upon PointNet and DeepSets. Moreover, complex models oftentimes apply non-differentiable layers like $knn(\cdot)$ into end-to-end learning, which will make the adversarial training ineffective. In this work, we aim at exploring how the symmetric (permutation-invariant) function can benefit adversarial training. To this end, we choose PointNet, DeepSets, and DSS as the backbone networks. For the ModelNet40 dataset, we follow the default setting to split into

Figure 1: Visualizations of adversarial examples (2048 points) generated by L^2 norm-based C&W adaptive attacks on PU-Net.

9,843 objects for training and 2,468 objects for validation [19]. We randomly sample 1024 points from each object to form its point cloud, if not otherwise stated.

PointNet. We leverage the default architecture in PointNet codebase[1] and exclude the transformation nets (*i.e.,* T-Net) and dropout layers for simplicity and reproducibility. PointNet leverages shared fully connected (FC) layers as the permutation-equivariant layer ϕ_l : $\mathrm{FC}_l(\boldsymbol{F}_{l:,i}) \rightarrow \boldsymbol{F}_{l+1:,i}$ and `MAX` pooling as the symmetric function $\rho(\cdot)$.

DeepSets. We leverage the default architecture in DeepSets codebase[2]. Different from PointNet, DeepSets first applies a symmetric function to each feature map and aggregate it with the original feature map. Afterwards, DeepSets also leverages FC layers to further process the features: $\phi_l : \mathrm{FC}_l(\boldsymbol{F}_{l:,i} - \zeta(\boldsymbol{F}_l)) \rightarrow \boldsymbol{F}_{l+1:,i}$, where $\zeta(\cdot)$ is column-wise `MAX` pooling in the original implementation. Similarly, `MAX` pooling is still used as $\rho(\cdot)$ in DeepSets.

DSS. DSS generalizes DeepSets architecture and applies another FC layer to $\zeta(\boldsymbol{F}_l)$ in DeepSets

[1] https://github.com/charlesq34/pointnet
[2] https://github.com/manzilzaheer/DeepSets

Figure 2: Visualizations of adversarial examples (2048 points) generated by L^2 norm-based C&W adaptive attacks on DUP-Net.

so that $\phi_l : \mathrm{FC}_{l1}(\boldsymbol{F}_{l:,i}) + \mathrm{FC}_{l2}(\zeta(\boldsymbol{F}_l)) \to \boldsymbol{F}_{l+1:,i}$. Different from other two achitectures, DSS utilizes `SUM` pooling as $\rho(\cdot)$. Since there is no available codebase at the time of writing, we implement DSS by ourselves.

We visualize the $\phi(\cdot)$ of different backbones in Figure 4 and summarize the layer information in Table 1.

B.3 Parametric Pooling Design & Implementation

We have introduced `ATT` in Section 4.3.1. In our implementation, we choose $L = 512$ so that $\boldsymbol{V} \in \mathbb{R}^{512\times 1024}$ to train the backbone models.

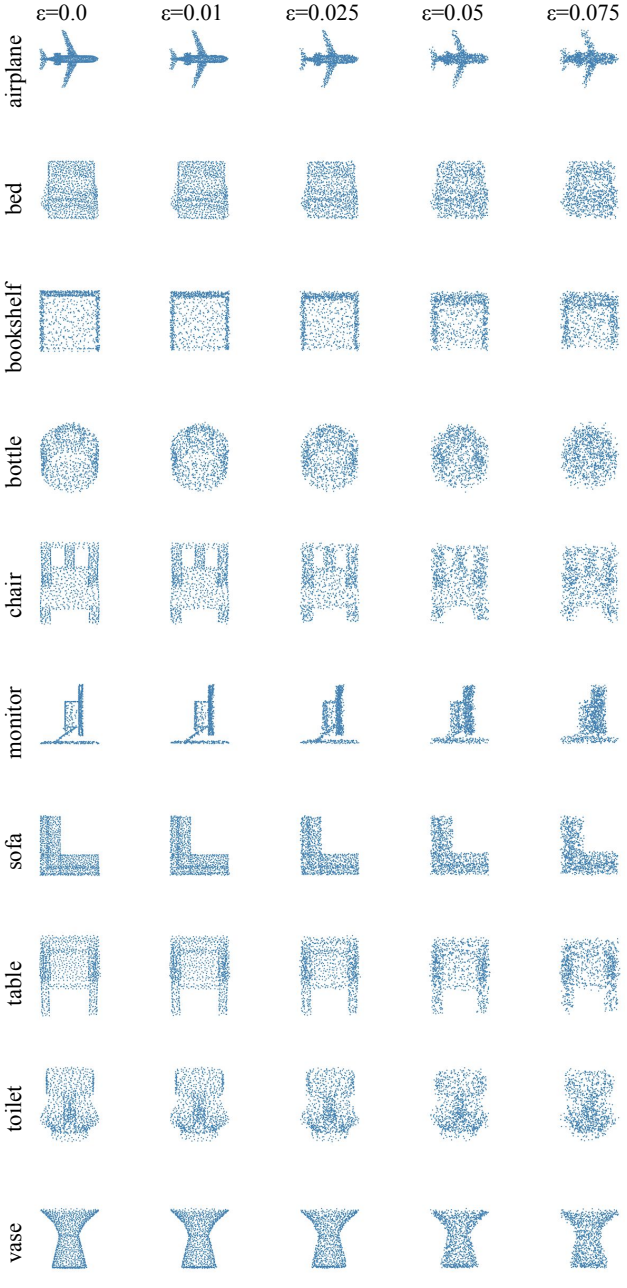
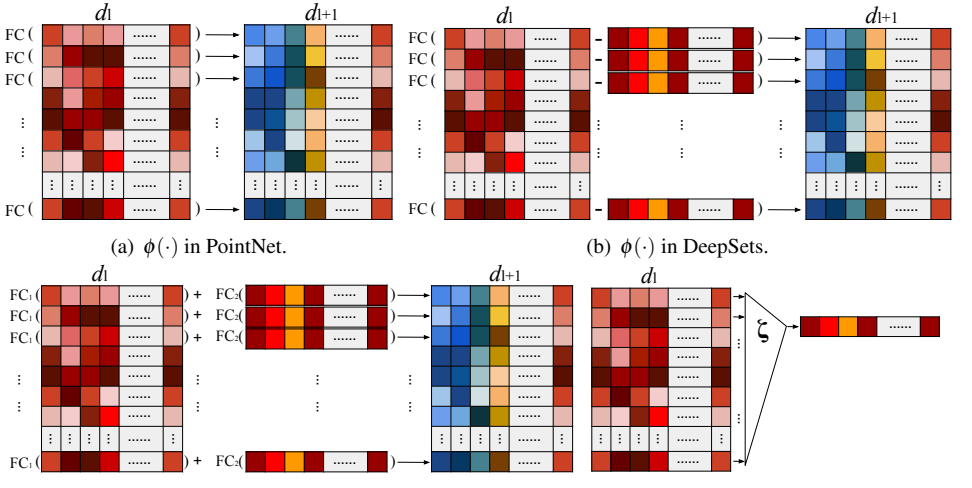


Figure 3: Visualizations of adversarial examples (1024 points) generated by L^∞ norm-based PGD adaptive attacks on GvG-PointNet++.

Table 1: Layer information of PointNet, DeepSets, and DSS. BN represents a batch normalization layer.

PointNet	DeepSets	DSS
$\phi_1 : n \times 3 \to n \times 64$	$\phi_1 : n \times 3 \to n \times 256$	$\phi_1 : n \times 3 \to n \times 64$
BN + ReLU	BN + ELU	BN + ReLU
$\phi_2 : n \times 64 \to n \times 64$	$\phi_2 : n \times 256 \to n \times 256$	$\phi_2 : n \times 64 \to n \times 256$
BN + ReLU	BN + ELU	BN + ReLU
$\phi_3 : n \times 64 \to n \times 128$	$\rho : n \times 256 \to 256$	$\phi_3 : n \times 256 \to n \times 256$
BN + ReLU	$\sigma_1 : 256 \to 256$	BN + ReLU
$\phi_4 : n \times 128 \to n \times 1024$	BN + Tanh	$\rho : n \times 256 \to 256$
BN + ReLU	$\sigma_2 : 256 \to 40$	$\sigma_1 : 256 \to 256$
$\rho : n \times 1024 \to 1024$		BN + ReLU
$\sigma_1 : 1024 \to 512$		$\sigma_2 : 256 \to 40$
BN + ReLU		
$\sigma_2 : 512 \to 256$		
BN + ReLU		
$\sigma_3 : 256 \to 40$		



(a) $\phi(\cdot)$ in PointNet. (b) $\phi(\cdot)$ in DeepSets.

(c) $\phi(\cdot)$ in DSS. (d) The aggregated feature in (b) and (c).

Figure 4: Different architectures of $\phi(\cdot)$ in PointNet, DeepSets, and DSS.

ATT−GATE is a variant of ATT with more learnable parameters:

$$\begin{aligned} \boldsymbol{g} &= \sum_{i=1}^{n} a_i \cdot \boldsymbol{f}_i \\ a_i &= \frac{\exp(\boldsymbol{w}^\top \cdot (\tanh(\boldsymbol{V} \cdot \boldsymbol{f}_i^\top) \odot \mathrm{sigm}(\boldsymbol{U} \cdot \boldsymbol{f}_i^\top)))}{\sum_{j=1}^{n} \exp(\boldsymbol{w}^\top \cdot (\tanh(\boldsymbol{V} \cdot \boldsymbol{f}_j^\top) \odot \mathrm{sigm}(\boldsymbol{U} \cdot \boldsymbol{f}_j^\top)))} \end{aligned} \tag{4}$$

where $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{L \times M}$, $\mathrm{sigm}(\cdot)$ is the sigmoid activation function, and $\odot$ is an element-wise multiplication. We also choose $L = 512$ in ATT−GATE to train the backbones.

PMA [8] adopts multi-head attention into pooling on a learnable set of k seed vectors

$\boldsymbol{S} \in \mathbb{R}^{k \times d_m}$ Let $\boldsymbol{F} \in \mathbb{R}^{n \times d_m}$ be the matrix version of the set of features.

$$\begin{aligned} &\mathrm{PMA}_k(\boldsymbol{F}) = \mathrm{MAB}(\boldsymbol{S}, \mathrm{FC}(\boldsymbol{F})) \\ &\mathrm{MAB}(\boldsymbol{X}, \boldsymbol{Y}) = \boldsymbol{H} + \mathrm{FC}(\boldsymbol{H}) \\ &\text{where} \quad \boldsymbol{H} = \boldsymbol{X} + \mathrm{Multihead}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Y}; \boldsymbol{w}) \end{aligned} \tag{5}$$

where $\mathrm{FC}(\cdot)$ is the fully connected layer and $\mathrm{Multihead}(\cdot)$ is the multi-head attention module [16]. We follow the implementation in the released codebase[3] to choose $k = 1$, the number of head = 4, and the hidden neurons in $\mathrm{FC}(\cdot)$ = 128 to train the backbone models.

`SoftPool` [17] re-organizes $\boldsymbol{F}$ so that its j-th dimension is sorted in a descending order, and picks the top k point-level embeddings $\boldsymbol{F'}_j \in \mathbb{R}^{k \times d_m}$ to further form $\widetilde{\boldsymbol{F}} = [\boldsymbol{F'}_1, \boldsymbol{F'}_2, \ldots, \boldsymbol{F'}_{d_m}]$. Then, `SoftPool` applies CNN to each $\widetilde{\boldsymbol{F}}_j \to \boldsymbol{g}_j$ so that the pooled representation is $\boldsymbol{g} = [\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_{d_m}]$. Since `SoftPool` [17] sorts the feature set in each dimension, it requires the number of dimensions d_m to be relatively small. We follow the description in their paper to choose $d_m = 8$ and $k = 32$ so that each $\boldsymbol{F}_j{}' \in \mathbb{R}^{32 \times 8}$. We apply one convolutional layer to aggregate each $\boldsymbol{F'}_j$ into $g_j \in \mathbb{R}^{1 \times 32}$ so that the final $\boldsymbol{g} \in \mathbb{R}^{1 \times 256}$. Therefore, for all backbone networks with `SoftPool`, we apply the last equivariant layer as $\phi : n \times d_{m-1} \to n \times 8$ and $\rho : n \times 8 \to 256$.

C `DeepSym` Ablation

It is worth noting that `DeepSym` does not require the final layer to have only one neuron. However, to have a fair comparison with other pooling operations that aggregate into one feature from each dimension, our implementation of `DeepSym` also aggregates into one feature from each dimension.

C.1 Evaluation Detail

We also perform extensive evaluations using different PGD attack steps and budgets ε on PGD-20 trained PointNet. Figure 5 shows that PointNet with `DeepSym` consistently achieves the best adversarial accuracy. We also validate `MEDIAN` pooling indeed hinders the gradient backward propagation. The adversarial accuracy of PointNet with `MEDIAN` pooling consistently drops even after PGD-1000. However, the adversarial accuracy of PointNet with other pooling operations usually converges after PGD-200. Figure 6 shows that `DeepSym` also outperforms other pooling operations under different adversarial budgets ε.

We leverage the default setup in FGSM, BIM, and MIM in our evaluation. FGSM is a single-step attack method, which can be represented as:

$$\boldsymbol{X}_{adv} = \boldsymbol{X} + \varepsilon \cdot \mathrm{sign}(\nabla_{\boldsymbol{X}} \mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}, \boldsymbol{y})) \tag{6}$$

The BIM attack is similar to PGD attacks described in Appendix A.1. The differences are 1) the attack starts from the original point cloud $\boldsymbol{X}$ and 2) the step size $\alpha = \varepsilon / T$, where T is the number of attack steps. The MIM attack introduces momentum terms into the adversarial optimization:

$$\boldsymbol{g}_{t+1} = \mu \cdot \boldsymbol{g}_t + \frac{\nabla_{\boldsymbol{X}_t} \mathcal{L}(\boldsymbol{X}_t, \boldsymbol{\theta}, \boldsymbol{y})}{||\nabla_{\boldsymbol{X}_t} \mathcal{L}(\boldsymbol{X}_t, \boldsymbol{\theta}, \boldsymbol{y}))||_1} \tag{7}$$

$$\boldsymbol{X}_{t+1} = \boldsymbol{X}_t + \alpha \cdot \mathrm{sign}(\boldsymbol{g}_{t+1}) \tag{8}$$

[3] https://github.com/juho-lee/set_transformer

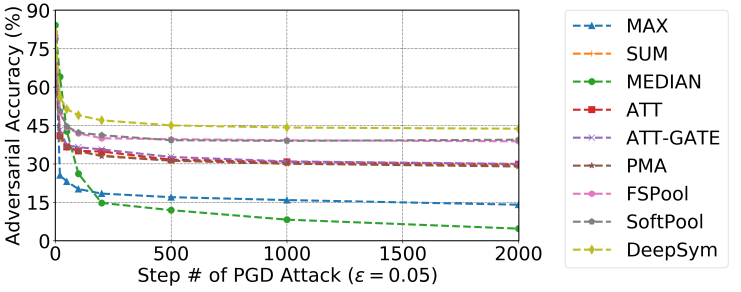


Figure 5: Adversarial accuracy of PGD-20 trained PointNet with different pooling operations. We leverage the PGD attack with different steps to evaluate the model's robustness.

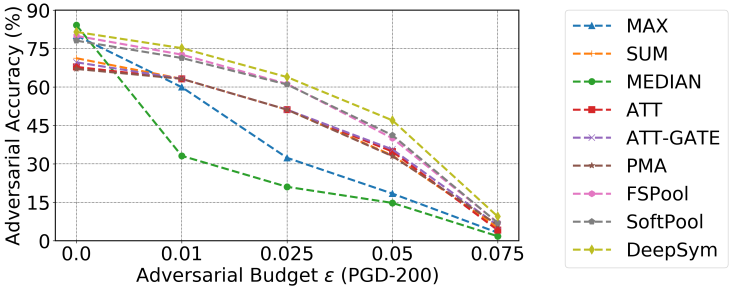


Figure 6: Adversarial accuracy of PGD-20 trained PointNet with different pooling operations. We leverage the PGD attack with different budgets to evaluate the model's robustness.

Similar to BIM, the attack starts from the original point cloud $\boldsymbol{X}$ and the step size $\alpha = \varepsilon/T$. We set $\mu = 1$ following the original setup [6].

Due to the computational resource constraints, we set the sample size = 32 and allow 2000 quires to find each adversarial example in the score-based black-box attack [7, 14]. For the evolution attack, we use the default loss $\mathcal{L}$ as the fitness score, and initialize 32 sets of perturbations from a Gaussian distribution $\mathcal{N}(0,1)$. 4 sets of perturbations with top fitness scores will remain for the next iteration, while others will be discarded. We also allow 2000 generations of evolution to find the adversarial example.

We also find the PointNet implementation[4] default leverages random rotation for data augmentation to improve its isometric stability. As we do not take isometric robustness into consideration, we further remove such augmentation to simplify the learning task. We then leverage the default setting (PGD-7) to adversarially train the model and test its robustness. Figure 7 shows that the baseline AT results improve due to the simplicity without rotation, as expected. Nevertheless, `DeepSym` still outperforms other pooling operations by a significant margin (13.6%).

Since `DeepSym` brings deep trainable layers into the original backbones, it is necessary to report its overhead. We leverage TensorFlow [2] and NVIDIA profiler [1] to measure the inference time, the number of trainable parameters, and GPU memory usage on PointNet. Specifically, the inference time is averaged from 2468 objects in the validation set, and the GPU memory is measured on an RTX 2080 with batch size = 8. As shown in Table 2,

[4] https://github.com/charlesq34/pointnet

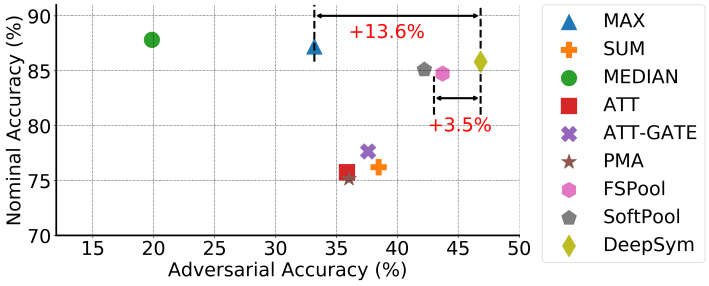


Figure 7: Robustness of PointNet on ModelNet40 without rotation augmentation under PGD-200 at $\varepsilon = 0.05$.

Table 2: Overhead measurement of PointNet with different pooling operations.

Pooling Operation	Inference Time (ms)	# Trainable Parameters	GPU Memory (MB)
MAX	2.21	815,336	989
MEDIAN	2.44	815,336	989
SUM	2.23	815,336	989
ATT	2.71	1,340,649	1980
ATT-GATE	3.07	1,865,962	2013
PMA	**2.10**	652,136	981
FSPool	2.89	1,863,912	1005
SoftPool	2.85	**355,328**	**725**
DeepSym (ours)	3.10	1,411,563	2013

DeepSym indeed introduces more computation overhead by leveraging the shared MLP. However, we believe the overhead is relatively small and acceptable, compared to its massive improvements on the adversarial robustness. To further have a lateral comparison, point cloud classification backbones are much more light-weight than image classification models. For example, ResNet-50 [6] and VGG-16 [13] have 23 and 138 million trainable parameters, respectively, and take much longer time to do the inference. The reason that models with SoftPool and PMA have fewer trainable parameters is that they limit the number of dimensions in the global feature by design.

C.2 Evaluation on ScanObjectNN

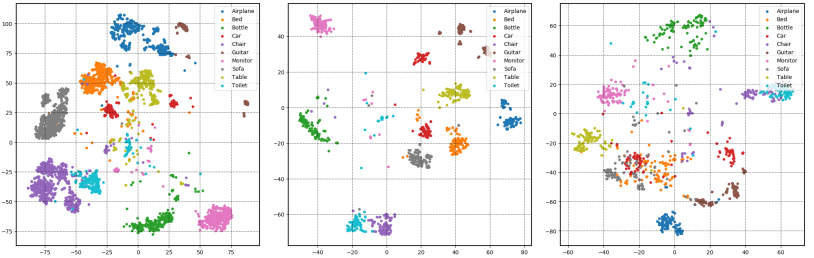
We also evaluate the adversarial robustness of different pooling operations on a new point cloud dataset, ScanObjectNN [15], which contains 2902 objects belonging to 15 categories. We leverage the same adversarial training setup as ModelNet10 (*i.e.,* PGD-1). Table 3 shows the results. We find that PointNet with DeepSym still achieves the best adversarial robustness. Since the point clouds from ScanObjectNN are collected from real-world scenes, which suffers from occlusion and imperfection, both nominal and adversarial accuracy drops compared to the results ModelNet40. We find that even some clean point clouds cannot be correctly recognized by human perception. Therefore, the performance degradation is also expected and we believe the results are not as representative as ones on ModelNet40.

Table 3: Adversarial robustness of PointNet with different pooling operations under PGD-200 at $\varepsilon = 0.05$.

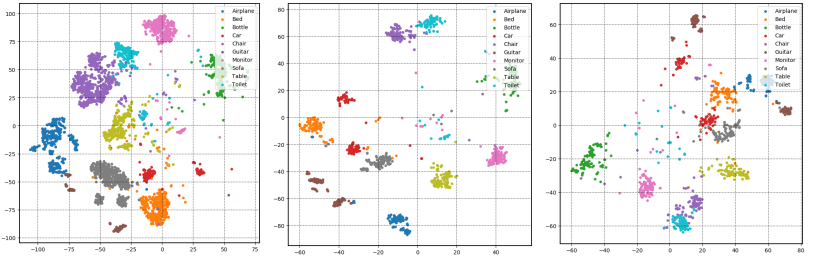
Pooling Operation	Nominal Accuracy	Adversarial Accuracy
MAX	75.2%	16.8%
MEDIAN	68.4%	8.2%
SUM	63.5%	18.3%
ATT	62.7%	17.9%
ATT-GATE	59.8%	17.1%
PMA	61.2%	16.2%
FSPool	**76.8%**	20.1%
SoftPool	73.2%	17.2%
DeepSym (ours)	76.7%	**22.8%**

C.3 T-SNE Visualization

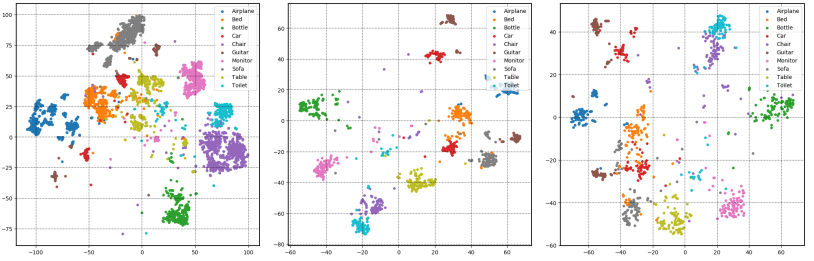
We visualize the global feature embeddings of adversarially trained PointNet under PGD-20 with different pooling operations in Figure 8 and their logits in Figure 9. Since it is hard to pick 40 distinct colors, though we put all data from 40 classes into the T-SNE process, we only choose 10 categories from ModelNet40 to realize the visualizations.



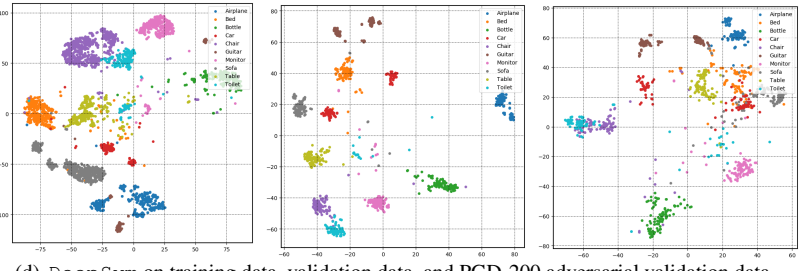
(a) MAX pooling on training data, validation data, and PGD-200 adversarial validation data.



(b) FSPool on training data, validation data, and PGD-200 adversarial validation data.

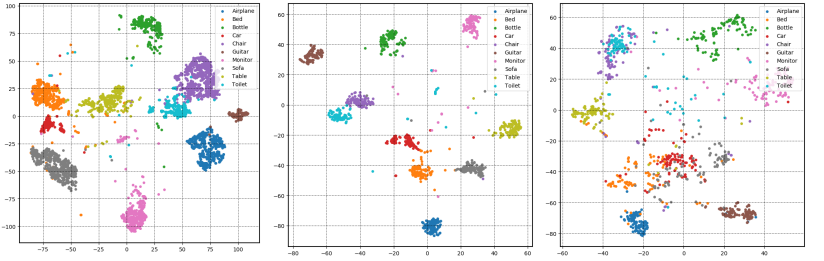


(c) SoftPool on training data, validation data, and PGD-200 adversarial validation data.

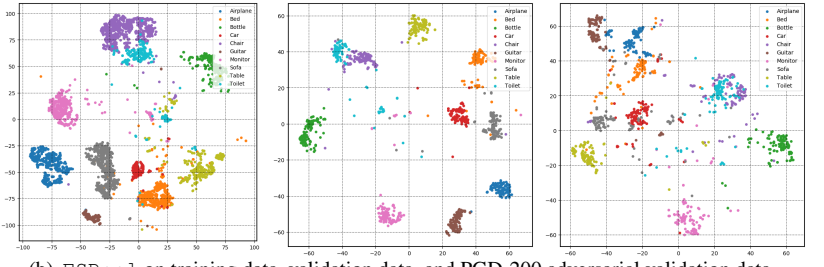


(d) DeepSym on training data, validation data, and PGD-200 adversarial validation data.

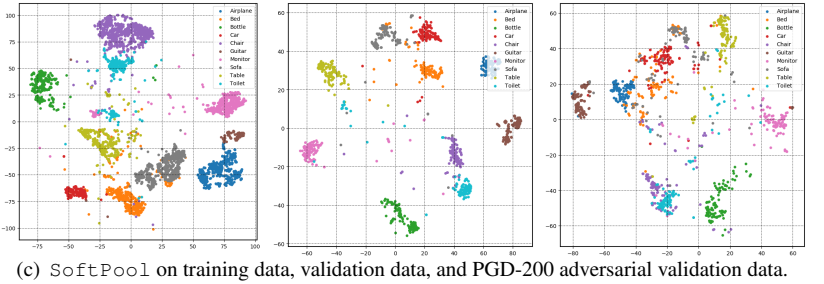
Figure 8: T-SNE visualizations of PointNet feature embeddings with MAX, FSPool, SoftPool, and DeepSym pooling operations. Three columns correspond to training data, validation data, and PGD-200 adversarial validation data, from left to right.



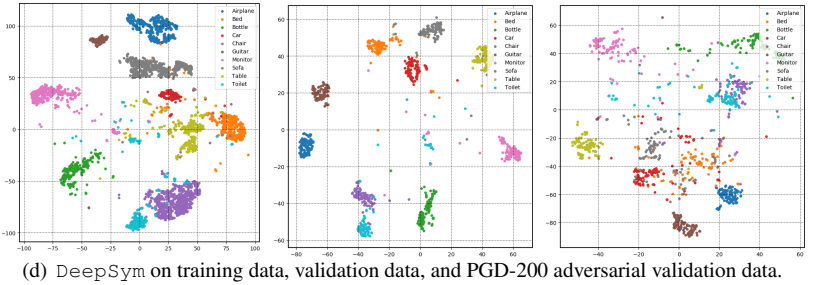
(a) MAX pooling on training data, validation data, and PGD-200 adversarial validation data.



(b) FSPool on training data, validation data, and PGD-200 adversarial validation data.



(c) SoftPool on training data, validation data, and PGD-200 adversarial validation data.



(d) DeepSym on training data, validation data, and PGD-200 adversarial validation data.

Figure 9: T-SNE visualizations of PointNet logits with MAX, FSPool, SoftPool, and DeepSym pooling operations. Three columns correspond to training data, validation data, and PGD-200 adversarial validation data, from left to right.

References

[1] NVIDIA System Management Interface. https://developer.nvidia.com/nvidia-system-management-interface, 2021.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[3] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017. doi: 10.1109/sp.2017.49. URL http://dx.doi.org/10.1109/SP.2017.49.

[4] Xiaoyi Dong, Dongdong Chen, Hang Zhou, Gang Hua, Weiming Zhang, and Nenghai Yu. Self-robust 3d point recognition via gather-vector guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9185–9193, 2018.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. *arXiv preprint arXiv:1804.08598*, 2018.

[8] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753, 2019.

[9] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

[10] Haggai Maron, Or Litany, Gal Chechik, and Ethan Fetaya. On learning sets of symmetric elements. *arXiv preprint arXiv:2002.08599*, 2020.

[11] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[12] 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.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] Jonathan Uesato, Brendan O'Donoghue, Aaron van den Oord, and Pushmeet Kohli. Adversarial risk and the dangers of evaluating against weak attacks. *arXiv preprint arXiv:1802.05666*, 2018.

[15] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[17] Yida Wang, David Joseph Tan, Nassir Navab, and Federico Tombari. Softpoolnet: Shape descriptor for point cloud completion and classification. *arXiv preprint arXiv:2008.07358*, 2020.

[18] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[19] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[20] Chong Xiang, Charles R Qi, and Bo Li. Generating 3d adversarial point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9136–9144, 2019.

[21] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3391–3401. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6931-deep-sets.pdf.

[22] 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/CVF International Conference on Computer Vision (ICCV)*, October 2019.