PointCloud-C: Benchmarking and Analyzing Point Cloud Perception Robustness under Corruptions

Jiawei Ren⋆, Lingdong Kong⋆, Liang Pan, and Ziwei Liu

Abstract—3D perception, especially point cloud classification and part segmentation, has achieved substantial progress. The advances include new network architectures, data augmentation techniques, as well as new learning paradigms, such as 3D self-supervised learning. However, in real-world deployment, point cloud corruptions are inevitable due to the scene complexity, sensor inaccuracy, and processing imprecision. In this work, we aim to rigorously benchmark and analyze point cloud classification under corruptions. To conduct a systematic investigation, we first provide a taxonomy of common 3D corruptions and identify the atomic corruptions. Then, we perform a comprehensive evaluation of a wide range of representative point cloud models to understand their robustness and generalizability. Our benchmark results show that although point cloud recognition performances improve over time, the state-of-the-art methods are on the verge of being less robust. Based on the obtained observations, we propose several effective techniques to enhance point cloud understanding robustness. We hope our comprehensive benchmark, in-depth analysis, and proposed techniques could spark future research in robust 3D perception. The benchmark suite is available on our project page: https://pointcloud-c.github.io/home.

Index Terms—Point cloud recognition; out-of-distribution robustness; 3D data augmentation; robust architecture design & benchmark

1 Introduction

Robustness to common corruptions is crucial to point cloud understanding. Compared to RGB images, point cloud data suffer more severe corruptions in real-world deployment due to the inaccuracy in 3D sensors and complexity in real-world 3D scenes [1], [2]. Furthermore, the point cloud is widely employed in safety-critical applications such as autonomous driving [3], [4]. Therefore, robustness to the out-of-distribution (OOD) point cloud data caused by corruptions becomes an important part of the test suite since the beginning of learning-based point cloud recognition [5], [6].

Ideally, robustness should be measured in a standard way like how classification/segmentation accuracy and computational cost are measured. However, prior research evaluates point cloud understanding robustness in many different protocols:

Protocol-1. Evaluate the robustness to a selected set of corruptions [5], [8], [9], [13], [19], *e.g.*, random point dropping and random jittering. This evaluation method is popular in point cloud research, as summarized in Table 1. However, the freedom to select corruptions brings both positive and negative effects to the evaluation. On the upside, customized selection allows the evaluation to focus on the most characteristic corruptions. On the downside, a selected set of corruptions cannot provide a comprehensive evaluation of a model's robustness. In addition, different corruption selections and training protocols in implementation also make it difficult to compare across methods.

Protocol-2. Evaluate the robustness to the sim-to-real gap [21], [22], *e.g.*, train on ModelNet40 [7] and test on ScanObjectNN [23]. To exploit the naturally occurring corruptions in real-world point cloud object datasets, robustness is formulated as the generalizability from a synthetic training set to a real test set. However, real-world corruptions always come in a composite way, *e.g.*, self-occlusion and scanner noise, making it hard to analyze each corruption independently. Besides, the sim-to-real performance gap couples with the domain gap within each category, *e.g.*, a *chair* in ModelNet40 [7] and ScanObjectNN [23] may have different styles, which obfuscates the evaluation results.

Protocol-3. Evaluate the robustness to adversarial attack [24], [25], [26], *e.g.*, adversarial point shifting and dropping. Differ-

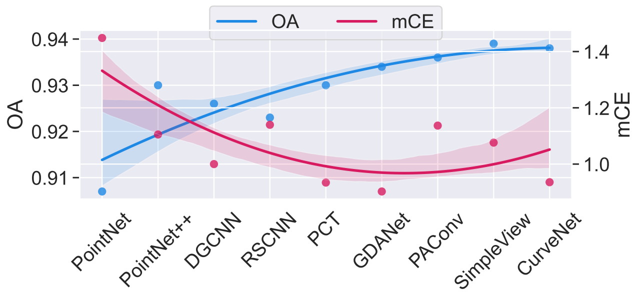


Fig. 1: Blue curve shows overall accuracy (OA) on ModelNet40 [7]. The red curve shows the mean Corruption Error (mCE) on proposed *ModelNet-C*. Methods are sorted in chronological order. OA gradually saturates but mCE is at risk of increasing due to the lack of a standard test suite.

- ⋆ *indicates equal contributions.*
- *J. Ren, L. Pan, and Z. Liu are with the S-Lab, Nanyang Technological University, Singapore.*
- *L. Kong is with the School of Computing, National University of Singapore.*
- *The corresponding author is Ziwei Liu: ziwei.liu@ntu.edu.sg.*

Manuscript received Oct. 24, 2022.

TABLE 1: Corruptions studied in the existing robustness analysis. Prior works evaluate point cloud recognition robustness on different sets of corruptions, and hence their evaluations can be partial and unfair. To standardize the corruption evaluation, our test suite *PointCloud-C* includes all previously studied corruptions, including “Jitter”, “Drop Global/Local”, “Add Global/Local”, “Scale”, and “Rotate”.

Method	Jitter	Drop Global	Drop Local	Add Global	Add Local	Scale	Rotate
PointNet [5]	✓	✓		✓			
ECC [6]	✓	✓					
PointNet++ [8]		✓					
DGCNN [9]		✓					
RSCNN [10]		✓					✓
PointASNL [2]		✓		✓			
Orderly Disorder [11]	✓						
PointAugment [12]	✓	✓				✓	✓
PointMixup [13]	✓	✓				✓	✓
PAConv [14]	✓					✓	✓
OcCo [15]		✓					
Triangle-Net [16]	✓	✓				✓	✓
Curve-Net [17]	✓	✓					
RSMix [18]	✓	✓				✓	✓
PointWolf [19]	✓	✓	✓		✓		
GDANet [20]		✓					✓
PointCloud-C (Ours)	✓	✓	✓	✓	✓	✓	✓

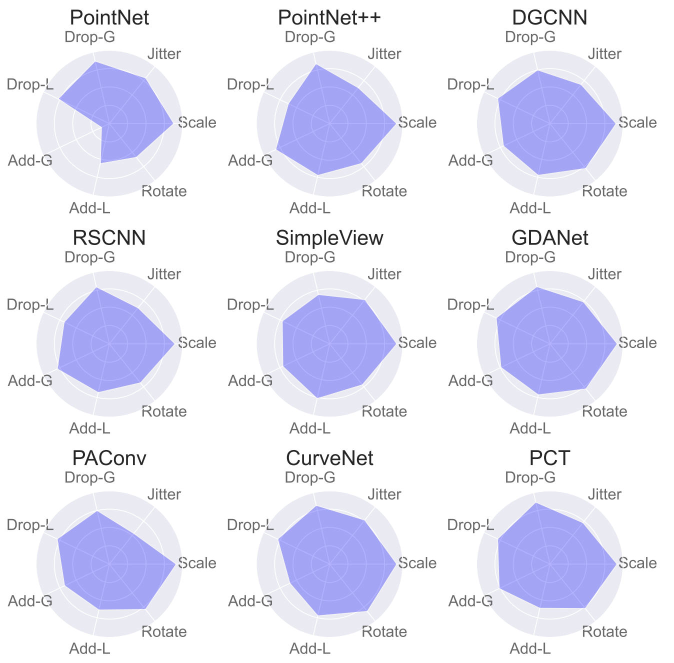


Fig. 2: Point cloud classifier’s robustness to various corruptions in a radar chart. Proposed *PointCloud-C* allows fine-grained corruption analysis. Different architectures have diverse strengths and weaknesses to corruptions. ”-G”: -Global. ”-L”: -Local.

ent from real-world scenarios where corruptions are drawn from natural distributions, adversarial attacks corrupt point clouds for the purpose to deceive a classifier while keeping the attacked point cloud similar to the input. Therefore, adversarial robustness is a good measure of a model’s worst-case performance but can not reflect a point cloud classifier’s robustness to common corruptions in the natural world.

Despite various ways to evaluate a point cloud classifier’s robustness, there lacks a standard, comprehensive benchmark for point cloud classification under corruptions. In this work, we present a full corruption test suite to close this gap. First, we break down real-world corruptions in *Protocol-2* into 7 fundamental atomic corruptions as shown in Figure 3, which also forms a superset of the ad-hoc corruption selections in *Protocol-1*. As we aim to measure real-world robustness, adversarial attacks in *Protocol-3* are excluded. Then, we apply the corruptions to the validation sets of ModelNet40 [7] and ShapeNet [27] as our corruption test suite dubbed ***PointCloud-C***, which includes *ModelNet-C* and *ShapeNet-C*. Inspired by the 2D image classification robustness benchmark [28], we further create 5 severity levels for each atomic corruption and use mean the mean Corruption Error (mCE) metric for evaluation. Finally, based on the test suite, we benchmark 17 point cloud classification methods, including 11 architectures, 3 augmentations, and 3 pretrains. As shown in Figure 2, our benchmark results show that although point cloud classification performance on the clean ModelNet40 [7] improves by time, *state-of-the-art (SoTA) methods are on the verge of being less robust*.

To remedy the issue, we conduct an in-depth analysis of the benchmark results and summarize two effective techniques to enhance point cloud classifier robustness. Strictly following the best design choice summarized from the benchmark results, we present Robust Point cloud Classifier (RPC), a robust network architecture for point cloud classification, which achieves the least mCE on *ModelNet-C* benchmark, and comparable overall accuracy on the clean ModelNet40 [7] with the SoTAs. In particular, we present WOLFMix, a strong augmentation baseline that exploits both deformation-based augmentation and mix-based augmentation to provide a stronger regularization. Empirically, WOLFMix achieves the best robustness results compared to existing augmentation techniques. According to our experimental results, the performance gain by augmentations does not equally transfer to all model architectures. We identify the best combination from existing methods and call for a model design that fully exploits the augmentation power.

Our contributions are summarized as follows:

- We present the first systematically-designed test-suite *PointCloud-C* for point cloud classification and part segmentation under corruptions.
- We comprehensively benchmark various existing methods on their robustness to corruptions.

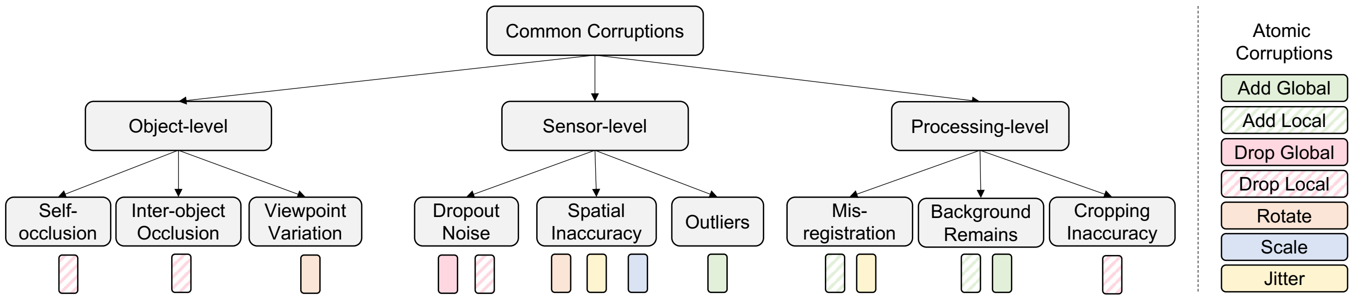


Fig. 3: Corruption taxonomy. We break down common corruptions into detailed corruption sources on object-, senor- and processing levels, which are further simplified into a combination of seven atomic corruptions for a more controllable empirical analysis.

- We summarize several effective techniques, such as RPC and WOLFMix, to enhance the point cloud classifier's robustness and identify that the synergy between architecture and augmentation should be considered in future research.

2 Related Works

Point Cloud Classification and Part Segmentation. Point cloud classification and part segmentation serve as fundamental tasks for 3D understanding from raw hardware inputs. Learning-based point cloud processors have diverse architectural designs. There are MLP-based models [5], [8], convolution-based models [10], [14], graph-based models [6], [9] and recently proposed transformer-based models [29], [30], [31]. Besides, there is a rising discussion on point cloud augmentation, including mix-based augmentations [13], [18], deformation-based augmentations [19] and auto-augmentations [12]. Moreover, self-supervised pre-train has drawn much research attention recently. Pre-trains obtained from pre-text tasks like occlusion reconstruction [15] and mask inpainting [32] provide better classification performance than random initialization.

Robustness in Point Cloud. Several attempts are made to improve the point cloud classifier's robustness. Triangle-Net [16] designs feature extraction that is invariant to positional, rotational and scaling disturbances. Although Triangle-Net [16] achieves exceptional robustness under extreme corruptions, its performance on clean data is not on par with SoTA. PointASNL [2] introduces adaptive sampling and local-nonlocal modules to improve robustness. However, PointASNL [2] takes a fixed number of points in implementation. Other works improve a model's adversarial robustness by denoising and upsampling [24], voting on subsampled point clouds [33], exploiting local feature's relative position [25] and self-supervision [26].

Robustness Benchmarks in Image Classification. A comprehensive robustness benchmark has been built for 2D image classification recently. ImageNet-C [28] corrupts the ImageNet [34]'s test set with simulated corruptions like compression loss and motion blur. ObjectNet [35] collects a test set with rich variations in rotation, background, and viewpoint. ImageNetV2 [36] re-collects a test set following ImageNet's protocol and evaluates the performance gap due to the natural distribution shift. Moreover, ImageNet-A [37] and ImageNet-R [38] benchmark the classifier's robustness to natural adversarial examples and abstract visual renditions.

3 Corruptions Taxonomy and Test Suite

3.1 Corruptions Taxonomy

Real-world corruptions come from a wide range of sources, based on which we provide a taxonomy of the corruptions in Figure 3. Common corruptions are categorized into three levels: object-level, sensor-level, and processing-level corruptions. Object-level corruptions come inherently in complex 3D scenes, where an object can be occluded by other objects or parts of itself. Different viewpoints also introduce variations to the point cloud data in terms of rotation. Note that viewpoint variation also leads to a change in self-occlusion. Sensor-level corruptions happen when perceiving with 3D sensors like LiDAR. As discussed in prior works [1], [39], sensor-level corruptions can be summarized as 1) dropout noise, where points are missing due to sensor limitations; 2) spatial inaccuracy, where point positions, object scale, and angle can be wrongly measured; 3) outliers, which are caused by the structural artifacts in the acquisition process. More corruptions could be introduced during postprocessing. For example, inaccurate point cloud registration leads to misalignment. Background remain and imperfect bounding box are two common corruptions during 3D object scanning.

However, it is challenging to directly simulate real-world corruptions for the following reasons. 1) Real-world corruptions have a rich variation, *e.g.*, different hardware may have different sensor-level corruptions. 2) The combination of inter-object occlusion or background remains can be inexhaustive. 3) Moreover, a few corruptions lead to the same kind of operations to point clouds, *e.g.*, self-occlusion, inter-object occlusion, and cropping error all lead to the missing a local part of the object. To this end, we simplify the corruption taxonomy into seven fundamental atomic corruptions: "Add Global", "Add Local", "Drop Global", "Drop Local", "Rotate", "Scale", and "Jitter". Consequently, each real-world corruption is broken down into a combination of the atomic corruptions, *e.g.*, background remain can be viewed as a combination of "Add Local" and "Add Global".

Although atomic corruptions cannot seamlessly simulate real-world corruptions, they provide a practical solution to achieve controllable empirical study on fundamentally analyzing point cloud classification robustness. Note that

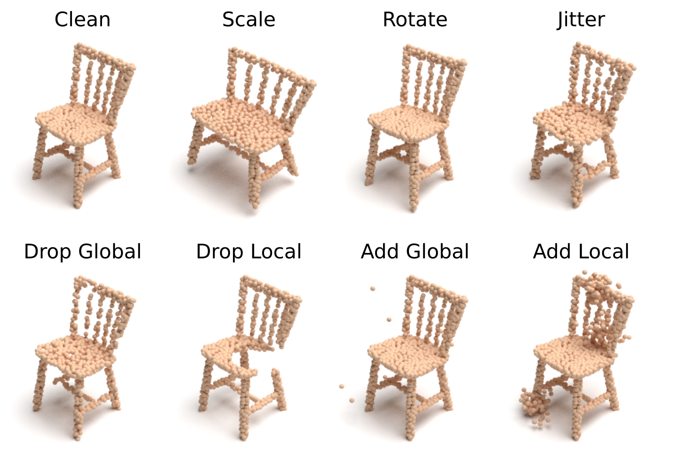


Fig. 4: Examples of *ModelNet-C*. We corrupt the clean test set of *ModelNet40* [7] using seven types of corruptions with five levels of severity to provide a comprehensive robustness evaluation. The listed examples are from severity level 2.

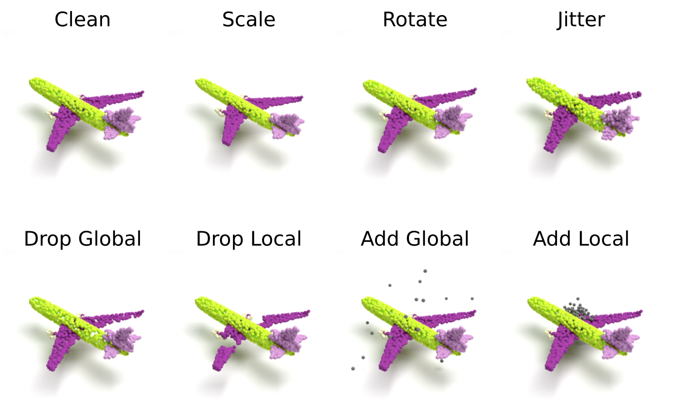


Fig. 5: Examples of our proposed *ShapeNet-C*. Similar to *ModelNet-C*, we corrupt the clean test set of *ShapeNet* [27] using seven types of corruptions with five levels of severity. The listed examples are from severity level 2.

noisy translation and random permutation are not considered in this work, because point cloud normalization and permutation-invariance are two basic properties among recent point cloud recognition approaches.

3.2 PointCloud-C: A Robustness Test Suite

3.2.1 ModelNet-C for Point Cloud Classification

ModelNet40 [7] is one of the most commonly used benchmarks in point cloud classification, and it collects 12,311 CAD models in 40 categories (9,843 for training and 2,468 for testing). Most recent point cloud classification methods follow the settings of PointNet [5], which samples 1024 points from each aligned CAD model and then normalizes them into a unit sphere. Based on ModelNet40 [7] and the settings by [5], we further corrupt the ModelNet40 [7] test set with the aforementioned seven atomic corruptions to establish a comprehensive test-suite *ModelNet-C*. To achieve fair comparisons and meanwhile following the OOD evaluation principle, we use the same training set with ModelNet40 [7]. Similar corruption operations are strictly *not allowed* during the model training phase.

The seven atomic corruptions are implemented as follows: "Scale" applies a random anisotropic scaling to the point cloud; "Rotate" rotates the point cloud by a small angle; "Jitter" adds a Gaussian noise to point coordinates; "Drop Global" randomly drops points from the point cloud; "Drop Local" randomly drops several k-NN clusters from the point cloud; "Add Global" adds random points sampled inside a unit sphere; "Add Local" expand random points on the point cloud into normally distributed clusters. The examples of corrupted point clouds from *ModelNet-C* are shown in Figure 4. In addition, we set different five severity levels for each corruption, based on which we randomly sample from the atomic operations to form a composite corruption test set. Note that we restrict the rotation to small angle variations, as in real-world applications we mostly observe objects from common viewpoints with small variations. Robustness to arbitrary SO(3) rotations is a specific challenging research topic [40], [41], which is out of the scope of this work.

3.2.2 ShapeNet-C for Part Segmentation

In a similar manner to *ModelNet-C*, we curate a corrupted part segmentation dataset dubbed as *ShapeNet-C*. *ShapeNet-C* is built upon the standard part segmentation dataset, the ShapeNet part dataset [42]. The ShapeNet part dataset includes 16,881 objects from 16 object-level categories and 50 part-level categories, with 2,048 points in each object.

We apply the same corruption taxonomize in subsection 3.1 to the ShapeNet test set to curate *ShapeNet-C*. For each corruption type, we set five increasing severity levels. We show the corrupted examples in Figure 5

3.3 Evaluation Metrics

To normalize the severity of different corruptions, we choose DGCNN, a classic point cloud classification method, as the baseline. Inspired by the 2D robustness evaluation metrics [28], we use mean CE (mCE), as the primary metric. To compute mCE, we first compute CE. For classification:

$$\mathrm{CE}_i = \frac{\sum_{l=1}^{5}(1-\mathrm{OA}_{i,l})}{\sum_{l=1}^{5}(1-\mathrm{OA}_{i,l}^{\mathrm{DGCNN}})}, \tag{1}$$

where $\mathrm{OA}_{i,l}$ is the overall accuracy on a corrupted test set i at corruption level l, $\mathrm{OA}_{i,l}^{\mathrm{DGCNN}}$ is baseline's overall accuracy. For part segmentation:

$$\mathrm{CE}_i = \frac{\sum_{l=1}^{5}(1-\mathrm{mIoU}_{i,l})}{\sum_{l=1}^{5}(1-\mathrm{mIoU}_{i,l}^{\mathrm{DGCNN}})}, \tag{2}$$

where $\mathrm{mIoU}_{i,l}$ is the mean Intersection of Union on a corrupted test set i at corruption level l, $\mathrm{mIoU}_{i,l}^{\mathrm{DGCNN}}$ is baseline's mIoU.

mCE is the average of CE over all seven corruptions:

$$\mathrm{mCE} = \frac{1}{N}\sum_{i=1}^{N}\mathrm{CE}_i, \tag{3}$$

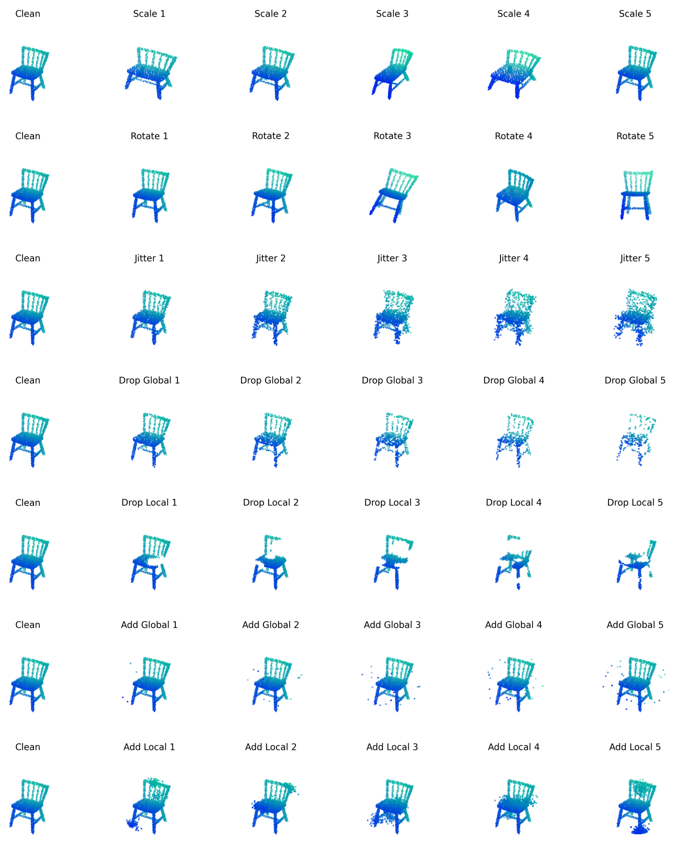


Fig. 6: Visualizations of *ModelNet-C* on all severity levels.

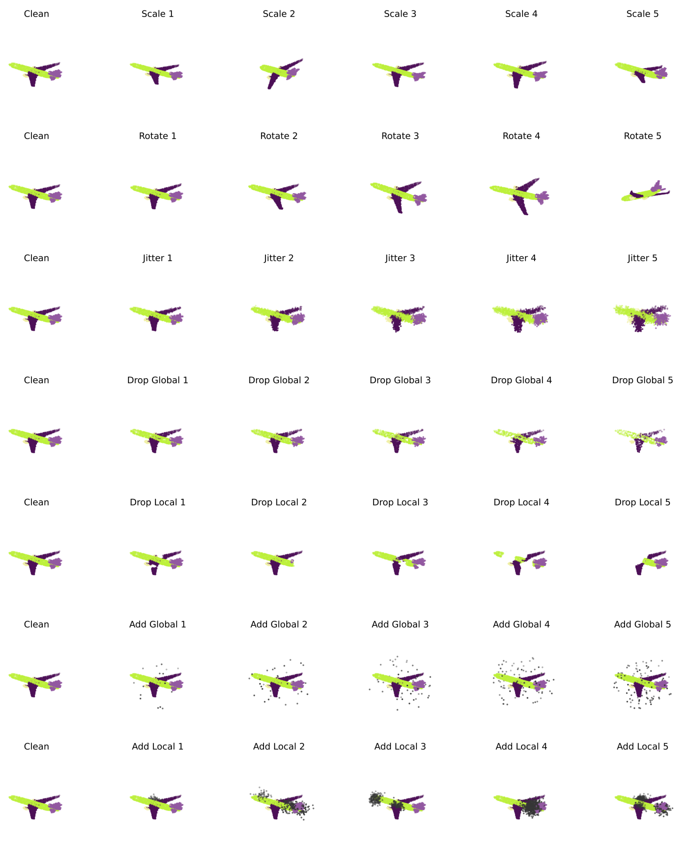


Fig. 7: Visualizations of *ShapeNet-C* on all severity levels.

where $N = 7$ is the number of corruptions. We also compute Relative mCE (RmCE), which measures performance drop compared to a clean test set as:

$$\mathrm{RCE}_i = \frac{\sum_{l=1}^{5}(\mathrm{OA}_{\mathrm{Clean}} - \mathrm{OA}_{i,l})}{\sum_{l=1}^{5}(\mathrm{OA}_{\mathrm{Clean}}^{\mathrm{DGCNN}} - \mathrm{OA}_{i,l}^{\mathrm{DGCNN}})}, \tag{4}$$

$$\mathrm{RmCE} = \frac{1}{N}\sum_{i=1}^{N}\mathrm{RCE}_i, \tag{5}$$

where $\mathrm{OA}_{\mathrm{Clean}}$ is the overall accuracy on the clean test set.

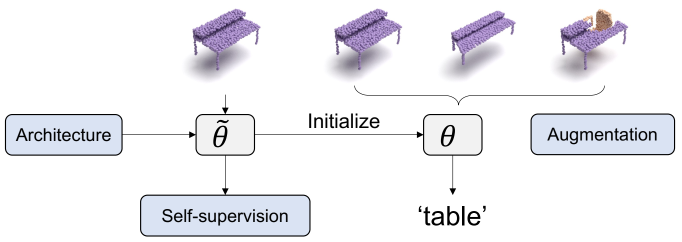


Fig. 8: Robust point cloud understanding paradigm. Point cloud recognition robustness to various corruptions largely depends on three main components: architecture design, self-supervised pretraining, and augmentation techniques.

3.4 Evaluation Protocol

Because most SoTA methods adopt the *DGCNN protocal* [43], we also use it as the consistent protocol for the benchmark. Two conventional augmentations are used during training: 1) random anisotropic scaling in the range [2/3, 3/2]; 2) random translation in the range [-0.2, +0.2]. Note that the random scaling ranges for training and testing are not overlapped. Point cloud sampling is fixed during training, and no voting is used in the inference stage. For each method, we select the model that performs the best on the clean ModelNet40 [7] and ShapeNet [27] test sets during evaluation. We highlight that the same corruptions are not allowed during training to reflect model OOD generalizability. The following works are recommended to specify augmentations in training when reporting results on our *PointCloud-C*.

4 Systematic Benchmarking

4.1 Benchmarked Methods

We benchmark **17** methods in total, covering three key components for robust point cloud classification and part segmentation, as shown in Figure 8. **Architectures**: PointNet [5], PointNet++ [8], DGCNN [9], RSCNN [10], SimpleView [43], GDANet [20], CurveNet [17], PAConv [14], PCT [29], Point-Transformers [30], and Point-MLP [44]. **Pretrains**: OcCo [15], Point-BERT [32], and Point-MAE [45]. **Augmentation**: Point-Mixup [13], RSMix [18], and PointWOLF [19]. For PointNet, PointNet++, DGCNN, RSCNN, and SimpleView, we use the pretrained models provided by [43]. For CurveNet, GDANet, and PAConv, we use their official pretrained models. The rest of the models are trained using their official code.

4.2 Corruptions and Severity Level Settings

In this section, we elaborate on the implementation of corruptions and severity level settings. A visualization is shown in Figure 6.

Jitter. We add a Gaussian noise $\epsilon \in \mathcal{N}(0, \sigma^2)$ to each of a point's X, Y, and Z coordinates, where $\sigma \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$ for the five levels.

Scale. We apply random scaling to the X, Y, and Z axis respectively. The scaling coefficient for each axis are independently sampled as $s \sim \mathcal{U}(1/S, S)$, where $S \in \{1.6, 1.7, 1.8, 1.9, 2.0\}$ for the five levels. Point clouds are re-normalized to a unit sphere after scaling.

Rotate. We randomly apply a rotation described by an X-Y-Z Euler angle (α, β, γ), where $\alpha, \beta, \gamma \sim \mathcal{U}(-\theta, \theta)$ and $\theta \in \{\pi/30, \pi/15, \pi/10, \pi/7.5, \pi/6\}$ for the five levels. Note that the sampling method does not guarantee a uniform SO(3) rotation sampling, but is sufficient to cover a range of rotation variations.

Drop Global. We randomly shuffle all points and drop the last $N * \rho$ points, where $N = 1024$ is the number of points in the point cloud and $\rho \in \{0.25, 0.375, 0.5, 0.675, 0.75\}$ for all five levels.

Drop Local. We drop K points in total, where $K \in \{100, 200, 300, 400, 500\}$ for the five levels. We randomly choose C, the number of local parts to drop, by $C \in \mathcal{U}\{1, 8\}$. We further randomly assign i-th local part a cluster size N_i so that $K = \sum_{i=1}^{C} N_i$. Then we repeat the following steps for C times: we randomly select a point as the i-th local center and drop its N_i-nearest neighbor points (including itself) from the point cloud.

Add Global. We uniformly sample K points inside a unit sphere and add them to the point cloud, where $K \in \{10, 20, 30, 40, 50\}$ for the five levels.

Add Local. We add K points in total, where $K \in \{100, 200, 300, 400, 500\}$ for the five levels. We randomly shuffle the points and select the first $C \in \mathcal{U}\{1, 8\}$ points as the local centers. We further randomly assign i-th local part a cluster size N_i so that $K = \sum_{i=1}^{C} N_i$. Neighbouring point's X-Y-Z coordinates are generated from a Normal distribution $\mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2 \mathrm{I})$, where $\boldsymbol{\mu}_i$ is the i-th local center's X-Y-Z coordinate and $\sigma_i \in \mathcal{U}(0.075, 0.125)$. We then add each local part to the point cloud one by one.

4.3 Main Results

Benchmark results (mCE) are reported in Table 3, Table 4 and Table 5 for architectures, pretrains, and augmentations, respectively. RmCE and Overall Accuracy are reported Table 7. In Figure 2, we sort benchmarked architectures in chronological order and visualize second-order polynomial fitting results with a 50% confidence interval. We observe that although the new architecture's performance is constantly progressing and saturates around 0.94, their mCE performance shows a large variance. We also observe that self-supervised pretraining is able to transfer the pretrain signal to the downstream model, but has a mixed effect on the overall performance. Moreover, recent point cloud augmentations can substantially improve robustness.

TABLE 2: Systematic study for architecture design.

	Representation	Local Operation	Advanced Grouping	Featurizer	mCE ↓
PointNet	3D	No	No	Conventional	1.422
PointNet++	3D	Ball-query	No	Conventional	1.072
DGCNN	3D	k-NN	No	Conventional	1.000
RSCNN	3D	Ball-query	No	Adaptive	1.130
PAConv	3D	k-NN	No	Adaptive	1.104
CurveNet	3D	k-NN	Curve	Conventional	0.927
GDANet	3D	k-NN	Frequency	Conventional	0.892
PCT	3D	k-NN	No	Self-attention	0.925
SimpleView	2D	-	-	-	1.047
RPC (Ours)	3D	k-NN	Frequency	Self-attention	**0.863**

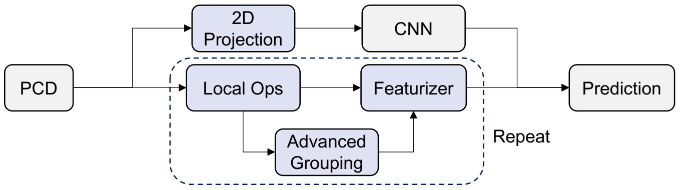


Fig. 9: Key components in architecture design. Point cloud data (PCD) repeatedly goes through local operations, advanced grouping, and featurization before being classified. Alternatively, PCD may be projected into multi-view images and processed by traditional CNN backbones. This figure means to show how key components are usually connected, but not to faithfully show every detailed architecture design.

5 Comprehensive Analysis

5.1 Architecture Design

We analyze four key components of point cloud classifier architectures: local operations, advanced grouping, featurizer, and representation dimension, as illustrated in Figure 9. The design choices of recent classifier architectures are summarized in Table 2. When analyzing a specific component, we group all methods that utilize the component. Since design choices are not rigorously controlled variables in the analysis, we visualize the 95% confidence interval together with the mean value in the bar charts, and only low variance results are considered in our conclusion. Furthermore, to empirically verify our conclusion, we build a new architecture, RPC, strictly following the conclusions.

Local Operations. We compare the robustness of different local aggregations, including no local operations, k-NN, and ball-query. As shown in Figure 10a, the exploitation of the point cloud locality is a key component to robustness. Without local aggregations, PointNet [5] (shown as "No Local Ops.") has the highest mCE. Considering each corruption individually, PointNet is on the two extremes: it shows the best robustness to "Jitter" and "Drop-G", meanwhile being one of the worst methods for the rest corruptions. Local operations target to encode informative representations by exploiting local geometric features. Ball-query randomly samples neighboring points in a predefined radius, while k-NN focuses on the nearest neighboring points. Generally, *k-NN performs better than ball-query in the benchmark*, especially for "Drop-L". The reason is that points surrounding the dropped local part will lose their neighbors in ball-query due to its fixed searching radius, but k-NN will choose neighbors from the remaining points. However, ball-query shows the advantage over k-NN in "Add-G", since, for a point on the

TABLE 3: Architectures. **Bold**: best in column. Underline: second best in column. Blue: best in row. Red: worst in row.

Method	OA ↑	**mCE** ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN [9]	0.926	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PointNet [5]	0.907	1.422	1.266	**0.642**	0.500	1.072	2.980	1.593	1.902
PointNet++ [8]	0.930	1.072	0.872	1.177	0.641	1.802	**0.614**	0.993	1.405
RSCNN [10]	0.923	1.130	1.074	1.171	0.806	1.517	0.712	1.153	1.479
SimpleView [43]	**0.939**	1.047	0.872	0.715	1.242	1.357	0.983	**0.844**	1.316
GDANet [20]	0.934	0.892	**0.830**	0.839	0.794	0.894	0.871	1.036	0.981
CurveNet [17]	0.938	0.927	0.872	0.725	0.710	1.024	1.346	1.000	**0.809**
PAConv [14]	0.936	1.104	0.904	1.465	1.000	1.005	1.085	1.298	0.967
PCT [29]	0.930	0.925	0.872	0.870	0.528	1.000	0.780	1.385	1.042
RPC (Ours)	0.930	**0.863**	0.840	0.892	**0.492**	**0.797**	0.929	1.011	1.079

TABLE 4: Pretrain. **Bold**: best in column. Underline: second best in column. Blue: best in row. Red: worst in row. †: Randomly initialized.

Method	OA ↑	**mCE** ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN [9]	**0.926**	**1.000**	1.000	1.000	1.000	**1.000**	**1.000**	1.000	1.000
+OcCo [15]	0.922	1.047	1.606	**0.652**	0.903	1.039	1.444	**0.847**	**0.837**
Point-BERT†	0.919	1.317	**0.936**	0.987	0.899	1.295	2.336	1.360	1.409
+Point-BERT [32]	0.922	1.248	**0.936**	1.259	**0.690**	1.150	1.932	1.440	1.326

TABLE 5: Augmentation. Bold: best in column. Underline: second best in column. Blue: best in row. Red: worst in row.

Method	OA ↑	**mCE** ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN [9]	0.926	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
+PointWOLF [19]	0.926	0.814	0.926	0.864	0.988	0.874	0.807	0.764	0.479
+RSMix [18]	0.930	0.745	1.319	0.873	0.653	0.589	**0.281**	0.629	0.870
+WOLFMix (Ours)	**0.932**	**0.590**	0.989	0.715	0.698	**0.575**	0.285	**0.415**	**0.451**
PointNet++ [8]	0.930	1.072	**0.872**	1.177	**0.641**	1.802	0.614	0.993	1.405
+PointMixUp [13]	0.915	1.028	1.670	**0.712**	0.802	1.812	0.458	0.615	1.130

TABLE 6: Results of combining WOLFMix with different architectures. **Bold**: best in column. Underline: second best in column. Blue: best in row. Red: worst in row. †: evaluated on the final epoch.

Method	OA ↑	**mCE** ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN [9]	0.926	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
+WOLFMix (Ours)	0.932	0.590	0.989	0.715	0.698	0.575	**0.285**	**0.415**	0.451
PointNet [5]	0.907	1.422	1.266	0.642	0.500	1.072	2.980	1.593	1.902
+WOLFMix (Ours)	0.884	1.180	2.117	**0.475**	0.577	1.082	2.227	0.702	1.079
PCT [29]	0.930	0.925	0.872	0.870	0.528	1.000	0.780	1.385	1.042
+WOLFMix (Ours)	0.927	0.786	0.894	0.991	**0.464**	1.000	0.610	1.091	0.451
GDANet [20]	**0.934**	0.892	**0.830**	0.839	0.794	0.894	0.871	1.036	0.981
+WOLFMix (Ours)	**0.934**	**0.571**	0.904	0.883	0.532	**0.551**	0.305	**0.415**	**0.409**
RPC (Ours)	0.930	0.863	0.840	0.892	0.492	0.797	0.929	1.011	1.079
+WOLFMix (Ours)	0.933	0.917	0.979	1.203	0.556	0.908	1.319	0.993	0.465
+WOLFMix† (Ours)	0.923	0.714	0.947	0.829	0.516	0.908	0.678	0.669	0.451

object, outliers are less likely to fall in the query ball than to be its nearest neighbors.

Advanced Grouping. Recent methods design advanced grouping techniques, such as Frequency Grouping [20] and Curve Grouping [17], to introduce structural prior into architecture design. Frequency grouping uses a graph high-pass filter [46], [47] to group point features in the frequency domain. Curve grouping forms a curve-like point set $\{P_1, P_2, ...P_N\}$ by walking from P_i to P_{i+1} following a learnable policy π. As shown in Figure 10b, we observe that *both grouping techniques improve model robustness by a clear margin.* The idea of frequency grouping aligns with the observations in [48]: there is a trade-off between model robustness to low-frequency corruptions and high-frequency corruptions. By viewing local-grouped features as low-frequency features and curve-grouped features as high-frequency features, the robustness gain can be again interpreted from a frequency perspective. Nonetheless, it is noteworthy that advanced grouping is more time-consuming during both training and testing.

Featurizer. We refer to conventional operators as shared MLPs and convolutional layers, which are common building blocks for point cloud models. Recent works explore various advanced feature processing methods, such as adaptive kernels and self-attention operations. RSCNN [10] and PAConv [14] design adaptive kernels whose weights change with low-level features like spatial coordinates and surface normals. Based on self-attention, PCT [29] proposes the offset-attention operation, which achieves impressive performance for point cloud analysis. Despite the success of RSCNN [10]

TABLE 7: Full results for Relative mCE. **Bold**: best in column. Underline: second best in column. Blue: best in row. Red: worst in row. †: random initialized. ‡: evaluate on the final epoch.

Method	RmCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN [9]	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PointNet [5]	1.488	1.300	0.455	0.178	0.970	3.557	1.716	2.241
PointNet++ [8]	1.114	0.600	1.248	0.511	2.278	0.502	1.010	1.645
RSCNN [10]	1.201	1.200	1.211	0.707	1.782	0.602	1.194	1.709
SimpleView [43]	1.181	1.050	0.682	1.420	1.654	1.036	0.851	1.574
GDANet [20]	0.865	0.600	0.822	0.753	0.895	0.864	1.090	1.028
CurveNet [17]	0.978	1.000	0.690	0.655	1.128	1.516	1.060	0.794
PAConv [14]	1.211	1.050	1.649	1.057	1.083	1.158	1.458	1.021
PCT [29]	0.884	0.600	0.847	0.351	1.030	0.724	1.547	1.092
RPC (Ours)	0.778	0.450	0.876	0.299	0.714	0.923	1.035	1.149
DGCNN+OcCo [15]	1.302	3.650	0.529	0.839	1.030	1.575	0.771	0.723
Point-BERT†	1.330	0.350	0.955	0.816	1.406	2.751	1.458	1.574
Point-BERT [32]	1.262	0.500	1.322	0.534	1.203	2.226	1.582	1.468
PN2+PointMixUp [13]	1.254	3.600	0.579	0.655	2.180	0.226	0.418	1.121
DGCNN+PointWOLF [19]	0.698	0.650	0.822	0.983	0.805	0.742	0.677	0.206
DGCNN+RSMix [18]	0.839	2.700	0.851	0.529	0.391	0.059	0.512	0.830
DGCNN+WOLFMix (Ours)	0.485	1.250	0.653	0.603	0.383	0.072	0.229	0.206
PCT+WOLFMix (Ours)	0.653	0.550	0.992	0.241	1.008	0.484	1.129	0.170
GDANet+WOLFMix (Ours)	0.439	0.950	0.880	0.379	0.361	0.109	0.239	0.156
RPC+WOLFMix (Ours)	0.940	1.250	1.293	0.408	0.910	1.457	1.025	0.234
RPC+WOLFMix‡ (Ours)	0.532	0.600	0.764	0.293	0.835	0.557	0.532	0.142

and PAConv [14] on clean point cloud classifications, they tend to be more sensitive to corruptions than conventional operators in our experiments shown in Figure 10c. Data corruption exacerbates through data-dependent kernels. Compared to conventional operators, *self-attention operations improve classifier robustness in several aspects*, particularly in “Drop-G”. We speculate that its robustness gains to “Drop-G” come from its ability to understand non-local relations from the global perspective. Note that Point-BERT [32] also introduces a self-attention-based architecture. However, it includes a fixed tokenizer that is trained on pretext tasks, which could be the bottleneck for its robustness performance. Therefore, we do not include the randomly initialized Point-BERT [32] result in the architecture analysis.

2D vs. 3D Representation. A few methods [43], [49] first project 3D shapes to 2D frames from different viewpoints, and then use 2D classifiers for recognizing 3D points. The recently proposed projection-based method, SimpleView [43] performs surprisingly well on clean 3D point clouds. In our experiments shown in Figure 10d, *projecting 3D points to 2D images brought mixed effects to classification*. The projection significantly reduces the effect of “Jitter” and “Add-L”, but suffers a lot from point scarcity, particularly “Drop-G”. This is consistent with human visual perception, as it is challenging for human vision to recognize the shape from point projections, especially for sparse and noisy points without texture information. Adding more observations from different perspectives might improve 2D perception accuracy, while extra efforts are required. In a nutshell, we think 3D cues are more straightforward and preferable for building a robust point cloud classifier.

5.2 Self-Supervised Pretraining

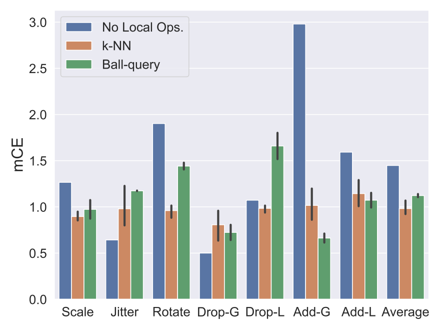
Recently, various self-supervised pretrain methods have been proposed for point cloud classification models, such as Point-BERT [32] and OcCo [15]. We study their robustness against corruptions in Figure 10e, which reveals that *pretrain signals can be transferred*, and hence benefiting classification under specific corruptions. During self-supervised pretrain, Point-BERT [32] first drops points using the block-wise masking strategy and then reconstructs the missing points based on the rest points. Interestingly, models finetuned on Point-BERT [32] pretrain show better classification robustness when the local part is missing. OcCo [15] employs a similar reconstruction pretrain task, but with a different masking strategy. By observing from different camera views, OcCo [15] masks the points that are self-occluded. Meanwhile, point clouds are also rotated with different camera angles. Consequently, the OcCo [15] pretrained models are significantly more robust to rotation perturbations. Moreover, OcCo [15] also improves the robustness to “Jitter” and “Add-L”.

5.3 Augmentation Method

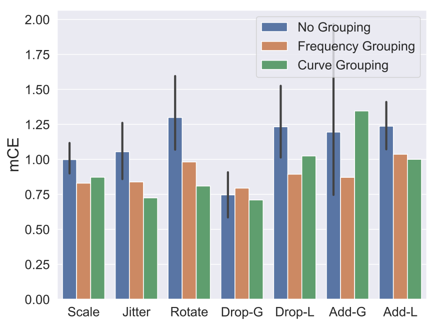
Following the principle of OOD evaluation, the corruptions should not be used as augmentations during training, and therefore we choose mixing and deformation augmentations. As shown in Figure 10f, *mixing and deformation augmentations can bring significant improvements to model robustness.* Point-MixUp [13] and RSMix [18] are two mix strategies. Similar to MixUp [50] in 2D augmentation, PointMixup mixes two point clouds using shortest-path interpolation. Similar to CutMix [51] in 2D augmentation, RSMix [18] mixes two point clouds using rigid transformation. Both mix strategies substantially reduce CE on corruptions including “Add-G”, “Add-L”, “Rotate” and “Jitter”. However, an unexpected side effect of the mix strategies is that classifiers become more vulnerable to scaling effects. By non-rigidly deforming local parts of an object, PointWOLF [19] enriches the data variation, which constantly improves recognition robustness on all evaluated corruptions.

6 Boosting Corruption Robustness

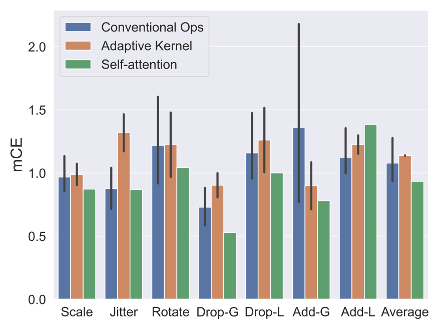
Based on the above observations, we propose to improve point cloud recognition robustness in the following ways.



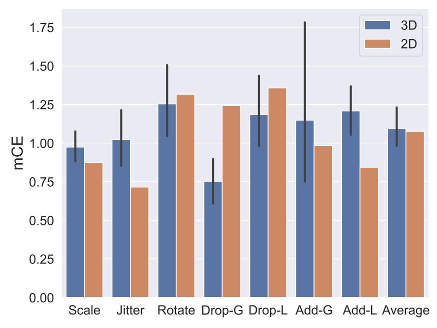
(a) Local Operations



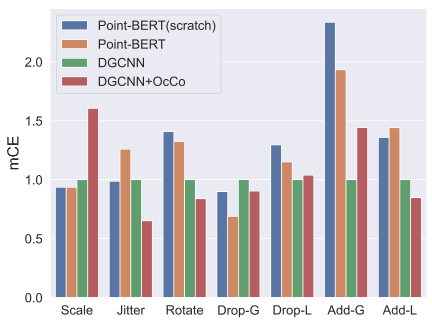
(b) Advanced Grouping



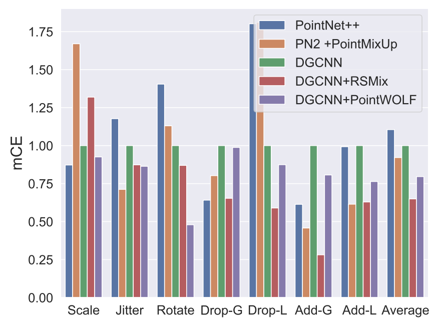
(c) Featurizer



(d) 2D v.s. 3D



(e) Pretrain



(f) Augmentation

Fig. 10: Analysis on different architecture designs, pretrain strategies and augmentation strategies' effect to classifier's performance under different corruptions. "-G": Global. "-L": Local.

6.1 RPC: A Robust Point Cloud Classifier

Following the conclusions in the architecture analysis, we construct RPC using 3D representation, k-NN, frequency grouping, and self-attention. We show a detailed architecture of RPC in Figure 11. As reported in Table 3, RPC achieves the best mCE compared to all SoTA methods. The success of RPC empirically verifies our conclusions on the architecture design choices, and it could serve as a strong baseline for future robustness research.

6.1.1 Hyperparameters

For local operation, we use k=30 for the number of neighbors in k-NN. For, frequency grouping, we follow the default hyper-parameters in GDANet [20]. The number of points in each frequency component is set to 256.

6.1.2 Training

We train the model for 250 epochs with a batch size of 32. We use SGD with a momentum of 0.9 for optimization. We use a cosine annealing scheduler to gradually decay the learning rate from 1e-2 to 1e-4.

6.2 WOLFMix: A Strong Augmentation Strategy

We design WOLFMix upon two powerful augmentation strategies, PointWOLF [19] and RSMix [18].

During training, WOLFMix first deforms the object, and then rigidly mixes the two deformed objects together. Ground-truth labels are mixed accordingly. We show an illustration of WOLFMix in Figure 12.

Concretely, for the deformation step, we use the default hyper-parameters in PointWOLF [19]. We set the number of anchors to 4, the sampling method to farthest point sampling, kernel bandwidth to 0.5, maximum local rotation range to 10 degrees, maximum local scaling to 3, and maximum local translation to 0.25. AugTune proposed along with PointWOLF [19] is not used in training. For the mixing step, we use the default hyper-parameters in RSMix [18]. We set RSMix probability to 0.5, β to 1.0, and a maximum number of point modifications to 512. For training, the number of neighbours in k-NN is reduced to 20 and the number of epochs is increased to 500 for all methods.

By taking advantage of both rigid and non-rigid transformations, WOLFMix brings substantial robustness gain over standalone PointWOLF [19] and RSMix [18] in Table 5.

Synergy between Architecture and Augmentation. We observe that augmentation techniques do not equally transfer to different architectures. Table 6 shows that the improvement by WOLFMix on corruption robustness varies drastically with different models. Although RPC achieves the lowest standalone mCE, its improvements by WOLFMix are much less than WOLFMix for DGCNN [9] or GDANet [20]. We notice that the final epoch model outperforms the best model on the clean test set, but still has a gap from top-performing methods. PointNet [5] and PCT [29] enjoy limited robustness gain as well. Hence, we speculate that there is a capacity upper bound to corruptions for each architecture. Future classification robustness research is suggested to study: 1) standalone robustness for architecture and augmentations independently; and 2) their synergy in between. Furthermore, we identify that training GDANet [20] with WOLFMix achieves the best robustness in all existing methods, with an impressive 0.571 mCE.

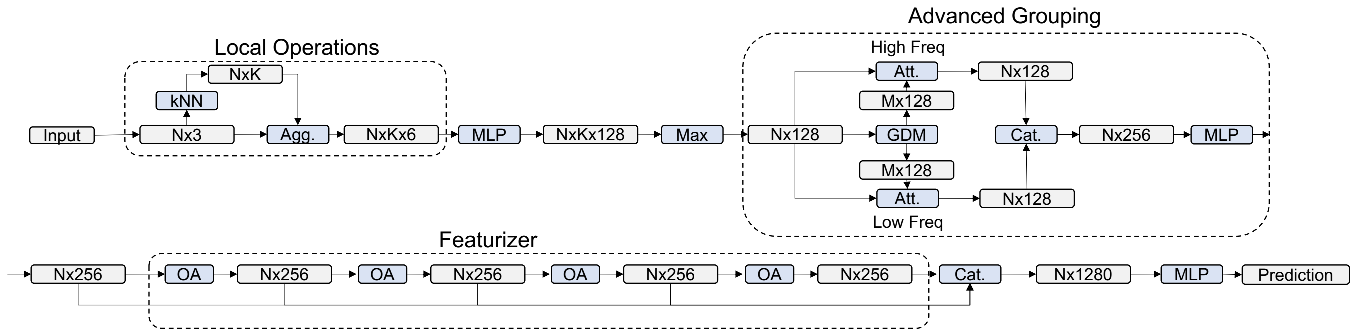


Fig. 11: Detailed architecture of RPC. We design RPC following the conclusions we draw from the benchmark. It optimizes the use of existing building blocks in point cloud classifiers and serves as a strong baseline for corruption robustness.

TABLE 8: Results for mCE on *ShapeNet-C*. **Bold**: best in column. Underline: second best in column. Blue: best in row. Red: worst in row. mIoU: average IoU over all corruptions.

Method	mIoU↑	mCE↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN [9]	0.852	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PointNet [5]	0.833	1.178	1.082	1.050	0.983	1.132	1.386	1.173	1.438
PointNet++ [8]	0.857	1.112	0.950	1.081	0.856	1.983	0.886	1.083	0.947
PAConv [14]	0.859	0.927	0.927	1.072	0.925	0.927	0.743	**0.948**	0.948
GDANet [20]	0.857	**0.923**	0.922	**1.012**	0.942	0.946	**0.712**	0.957	0.969
PointTransformers [30]	0.840	1.049	1.076	1.072	1.032	1.081	1.112	1.066	0.907
Point-MLP [44]	0.853	0.977	0.965	1.132	0.887	0.991	0.929	1.061	**0.876**
OcCo-DGCNN [15]	0.851	0.977	0.963	1.068	0.957	1.020	0.942	0.998	0.890
OcCo-PointNet [15]	0.832	1.130	1.108	1.037	0.964	1.102	1.221	1.125	1.351
OcCo-PCN [15]	0.815	1.173	1.228	1.042	1.081	1.181	1.065	1.148	1.465
Point-BERT [32]	0.855	1.033	0.938	1.098	0.873	0.927	1.170	1.199	1.025
Point-MAE [45]	**0.860**	0.927	**0.908**	1.035	**0.852**	**0.882**	0.776	1.031	1.003

TABLE 9: Results for Relative mCE (RmCE) on *ShapeNet-C*. **Bold**: best in column. Underline: second best in column. Blue: best in row. Red: worst in row. mIoU: average IoU over all corruptions.

Method	mIoU↑	RmCE↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN [9]	0.852	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PointNet [5]	0.833	1.056	0.355	0.880	0.087	1.152	1.566	1.206	2.144
PointNet++ [8]	0.857	1.850	0.685	1.329	0.176	7.860	0.830	1.169	0.901
PAConv [14]	0.859	0.848	0.560	1.336	0.764	0.789	0.597	**0.947**	0.940
GDANet [20]	0.857	0.785	0.264	1.115	0.806	0.842	**0.535**	0.952	0.979
PointTransformer [30]	0.840	0.933	0.981	1.051	0.728	1.077	1.133	1.054	**0.507**
Point-MLP [44]	0.853	0.810	0.474	1.428	0.217	0.961	0.882	1.109	0.601
OcCo-DGCNN [15]	0.851	0.804	0.224	1.196	0.630	1.078	0.894	0.988	0.617
OcCo-PointNet [15]	0.832	0.937	0.674	0.822	**−0.086**	0.909	1.281	1.117	1.842
OcCo-PCN [15]	0.815	0.882	0.846	**0.581**	0.026	0.766	0.934	1.070	1.951
Point-BERT [32]	0.855	0.895	0.283	1.356	0.213	0.619	1.303	1.360	1.128
Point-MAE [45]	**0.860**	**0.703**	**0.180**	1.209	0.222	**0.459**	0.650	1.088	1.114

7 Part Segmentation Robustness

We further study the point cloud part segmentation task to generalize the analysis to the generic point cloud processing. Based on *ShapeNet-C*, we benchmark on seven different architectures and three different unsupervised pretrain methods. **Architectures:** PointNet [5], PointNet++ [8], DGCNN [9], PAConv [14], GDANet [20], PointTransformers [30] and Point-MLP [44]. **Pretrain:** OcCo [15], Point-BERT [32], and Point-MAE [45]. We have not benchmarked augmentation techniques due to the insufficiency of existing works. We show the experiment results in Table 8 and Table 9.

Note that PointTransformers [30], Point-MLP [44], and Point-MAE [45] are three new methods that are not included in the previous anaylsis. PointTransformers [30] and Point-MLP [44] use attention and MLP for featureization and Point-MAE [45] use inpainting as a self-supervision signal. We use their official pretrained models when available and train the rest using their official codes.

7.1 Architecture Design

We observe that the most robust architecture on part segmentation is still GDANet [20], which aligns with the results on *ModelNet-C*. The result advocates that advanced grouping is a powerful design choice for robust point cloud processing. It is also noteworthy that PAConv [14] also shows a competitive performance, thanks to the position adaptive convolution mechanism during feature processing.

7.2 Self-Supervised Pretraining

Self-supervised pretraining demonstrates the robustness has a mixed performance. OcCo [15] and Point-BERT [32] have

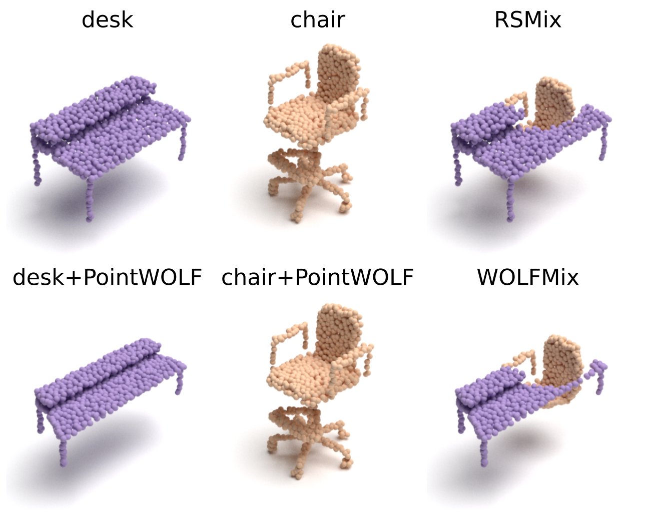


Fig. 12: Illustration of the proposed WOLFMix augmentation. Point clouds are first locally deformed and then rigidly mixed. Ground truth labels are mixed accordingly.

no significant effect on robustness, while Point-MAE [45] achieves the second-best result. It indicates that unsupervised pretrain has a promising potential in improving point cloud processing robustness but proper implementation is required.

8 Conclusion

In this work, we establish a comprehensive test suite *PointCloud-C* for robust point cloud classification under corruptions, including *ModelNet-C* for classification and *ShapeNet-C* for part segmentation. We systematically benchmarked and analyzed representative point cloud classification methods. By analyzing benchmark results, we propose two effective strategies, RPC and WOLFMix, for improving robustness. As the SoTA methods for point cloud classification on clean data are becoming less robust to random real-world corruptions, we highly encourage future research to focus on classification robustness so as to benefit real applications.

9 Acknowledgment

This work is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG2-PhD-2021-08-018), NTU NAP, MOE AcRF Tier 2 (T2EP20221-0033), and under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s).

References

[1] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, “Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud,” in *IEEE International Conference on Robotics and Automation*, 2019, pp. 4376–4382.

[2] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, “Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 5589–5598.

[3] L. Kong, J. Ren, L. Pan, and Z. Liu, “Lasermix for semi-supervised lidar semantic segmentation,” *arXiv preprint arXiv:2207.00026*, 2022.

[4] L. Kong, N. Quader, and V. E. Liong, “Conda: Unsupervised domain adaptation for lidar segmentation via regularized domain concatenation,” *arXiv preprint arXiv:2111.15242*, 2021.

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[6] M. Simonovsky and N. Komodakis, “Dynamic edge-conditioned filters in convolutional neural networks on graphs,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, July 2017.

[7] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, “3d shapenets: A deep representation for volumetric shapes,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[8] C. Qi, L. Yi, H. Su, and L. J. Guibas, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” in *Advances in Neural Information Processing Systems*, 2017.

[9] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph cnn for learning on point clouds,” *ACM Transactions on Graphics*, 2019.

[10] Y. Liu, B. Fan, S. Xiang, and C. Pan, “Relation-shape convolutional neural network for point cloud analysis,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8895–8904.

[11] M. Ghahremani, B. Tiddeman, Y. Liu, and A. Behera, “Orderly disorder in point cloud domain,” in *European Conference on Computer Vision*, vol. 12373, 2020, pp. 494–509.

[12] R. Li, X. Li, P. Heng, and C. Fu, “Pointaugment: An auto-augmentation framework for point cloud classification,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6377–6386.

[13] Y. Chen, V. T. Hu, E. Gavves, T. Mensink, P. Mettes, P. Yang, and C. G. M. Snoek, “Pointmixup: Augmentation for point clouds,” in *European Conference on Computer Vision*, vol. 12348, 2020, pp. 330–345.

[14] M. Xu, R. Ding, H. Zhao, and X. Qi, “Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[15] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner, “Unsupervised point cloud pre-training via occlusion completion,” in *IEEE/CVF International Conference on Computer Vision*, 2021.

[16] C. Xiao and J. P. Wachs, “Triangle-net: Towards robustness in point cloud learning,” in *IEEE/CVF Winter Conference on Applications of Computer Vision*. IEEE, 2021, pp. 826–835.

[17] T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai, “Walk in the cloud: Learning curves for point clouds shape analysis,” in *IEEE/CVF International Conference on Computer Vision*, October 2021, pp. 915–924.

[18] D. Lee, J. Lee, J. Lee, H. Lee, M. Lee, S. Woo, and S. Lee, “Regularization strategy for point cloud via rigidly mixed sample,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 900–15 909.

[19] S. Kim, S. Lee, D. Hwang, J. Lee, S. J. Hwang, and H. J. Kim, “Point cloud augmentation with weighted local transformations,” in *IEEE/CVF International Conference on Computer Vision*, 2021, pp. 548–557.

[20] M. Xu, J. Zhang, Z. Zhou, M. Xu, X. Qi, and Y. Qiao, “Learning geometry-disentangled representation for complementary understanding of 3d object point cloud,” in *AAAI Conference on Artificial Intelligence*, 2021, pp. 3056–3064.

[21] J. Reizenstein, R. Shapovalov, P. Henzler, L. Sbordone, P. Labatut, and D. Novotny, “Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction,” in *IEEE/CVF International Conference on Computer Vision*, 2021.

[22] A. Ahmadyan, L. Zhang, A. Ablavatski, J. Wei, and M. Grundmann, “Objectron: A large scale dataset of object-centric videos in the wild with pose annotations,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7822–7831.

[23] M. A. Uy, Q. Pham, B. Hua, D. T. Nguyen, and S. Yeung, “Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data,” in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1588–1597.

[24] H. Zhou, K. Chen, W. Zhang, H. Fang, W. Zhou, and N. Yu, “Dup-net: Denoiser and upsampler network for 3d adversarial point

clouds defense," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1961–1970.

[25] X. Dong, D. Chen, H. Zhou, G. Hua, W. Zhang, and N. Yu, "Self-robust 3d point recognition via gather-vector guidance," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 513–11 521.

[26] J. Sun, Y. Cao, C. Choy, Z. Yu, A. Anandkumar, Z. M. Mao, and C. Xiao, "Adversarially robust 3d point cloud recognition using self-supervisions," in *Advances in Neural Information Processing Systems*, 2021.

[27] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[28] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *International Conference on Learning Representations*, 2019.

[29] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," 2020.

[30] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 259–16 268.

[31] K. Mazur and V. Lempitsky, "Cloud transformers: A universal approach to point cloud processing tasks," in *IEEE/CVF International Conference on Computer Vision*, 2021.

[32] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "Point-bert: Pre-training 3d point cloud transformers with masked point modeling," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 19 313–19 322.

[33] H. Liu, J. Jia, and N. Z. Gong, "Pointguard: Provably robust 3d point cloud classification," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6186–6195.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009.

[35] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz, "Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models," in *Advances in Neural Information Processing System*, 2019, pp. 9448–9458.

[36] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?" in *International Conference on Machine Learning*, vol. 97, 2019, pp. 5389–5400.

[37] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song, "Natural adversarial examples," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[38] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, D. Song, J. Steinhardt, and J. Gilmer, "The many faces of robustness: A critical analysis of out-of-distribution generalization," *IEEE/CVF International Conference on Computer Vision*, 2021.

[39] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva, "State of the art in surface reconstruction from point clouds," in *Eurographics 2014-State of the Art Reports*, vol. 1, 2014, pp. 161–185.

[40] Z. Zhang, B.-S. Hua, D. W. Rosen, and S.-K. Yeung, "Rotation invariant convolutions for 3d point clouds deep learning," *International Conference on 3D Vision*, pp. 204–213, 2019.

[41] C. Chen, G. Li, R. Xu, T. Chen, M. Wang, and L. Lin, "Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4989–4997, 2019.

[42] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3d shape collections," *ACM Transactions on Graphics*, vol. 35, no. 6, pp. 1–12, 2016.

[43] A. Goyal, H. Law, B. Liu, A. Newell, and J. Deng, "Revisiting point cloud shape classification with a simple and effective baseline," *International Conference on Machine Learning*, 2021.

[44] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu, "Rethinking network design and local geometry in point cloud: A simple residual mlp framework," *arXiv preprint arXiv:2202.07123*, 2022.

[45] Y. Pang, W. Wang, F. E. Tay, W. Liu, Y. Tian, and L. Yuan, "Masked autoencoders for point cloud self-supervised learning," *arXiv preprint arXiv:2203.06604*, 2022.

[46] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[47] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[48] D. Yin, R. G. Lopes, J. Shlens, E. D. Cubuk, and J. Gilmer, "A fourier perspective on model robustness in computer vision," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 255–13 265.

[49] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5648–5656.

[50] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR (Poster)*, 2018.

[51] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6023–6032.

Jiawei Ren is a Ph.D. student in Computer Science in the School of Computer Science and Engineering at Nanyang Technological University, Singapore, advised by Prof. Ziwei Liu. His research focuses on robust learning for 3D perception and generation. He obtained the B.Eng. degree in Electrical and Electronic Engineering from Nanyang Technological University, Singapore, where he was advised by Prof. Lap-Pui Chau. He has published several papers at top-tier conferences, including CVPR, ECCV, NeurIPS, ICML, and AAAI. He is the recipient of the Google Ph.D. fellowship and the AISG Ph.D. fellowship.

Lingdong Kong is a Ph.D. student in the School of Computing, Department of Computer Science, National University of Singapore. He received the B.Eng. degree from the South China University of Technology, Guangzhou, China, in 2019, and the M.Sc. and M.Eng. degrees from Nanyang Technological University, Singapore, in 2020 and 2022, respectively. His research interests include 3D perception, domain adaptation, and semi-supervised learning. He is the recipient of the DesCartes Ph.D. fellowship from CNRS and the National Scholarship from the Ministry of Education of China.

Liang Pan received the Ph.D. degree in Mechanical Engineering from the National University of Singapore in 2019. He is currently a Research Fellow at S-Lab, Nanyang Technological University, Singapore. Previously, He is a Research Fellow at the Advanced Robotics Centre at the National University of Singapore. His research interests include computer vision and 3D point cloud, with focuses on shape analysis, deep learning, and 3D human. He also serves as a reviewer for top computer vision and robotics conferences, such as CVPR, ICCV, ECCV, ICML, NeurIPS, IROS, and ICRA.

Ziwei Liu is currently an Assistant Professor in the School of Computer Science and Engineering at Nanyang Technological University, Singapore. Previously, he was a senior research fellow at the Chinese University of Hong Kong and a postdoctoral researcher at the University of California, Berkeley. Ziwei received his Ph.D. degree from the Chinese University of Hong Kong in 2017. His research revolves around computer vision/graphics, machine learning, and robotics. He has published extensively in top-tier conferences and journals in relevant fields, including CVPR, ICCV, ECCV, NeurIPS, IROS, SIGGRAPH, TOG, and TPAMI. He is the recipient of the Microsoft Young Fellowship, Hong Kong Ph.D. Fellowship, ICCV Young Researcher Award, and HKSTP best paper award.