Benchmarking and Analyzing Point Cloud Classification under Corruptions

Jiawei Ren [1]   Liang Pan [1]   Ziwei Liu [1]

Abstract

3D perception, especially point cloud classification, has achieved substantial progress. 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 on a wide range of representative point cloud models to understand their robustness and generalizability. Our benchmark results show that although point cloud classification performance improves 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 classifier robustness. We hope our comprehensive benchmark, in-depth analysis, and proposed techniques could spark future research in robust 3D perception. Code is available at https://github.com/jiawei-ren/modelnetc.

1. Introduction

Robustness to common corruptions is crucial to point cloud classification. Compared to 2D 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 (Wu et al., 2019; Yan et al., 2020). Furthermore, point cloud is widely employed in safety-critical applications such as autonomous driving. Therefore, robustness to 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 classification (Qi

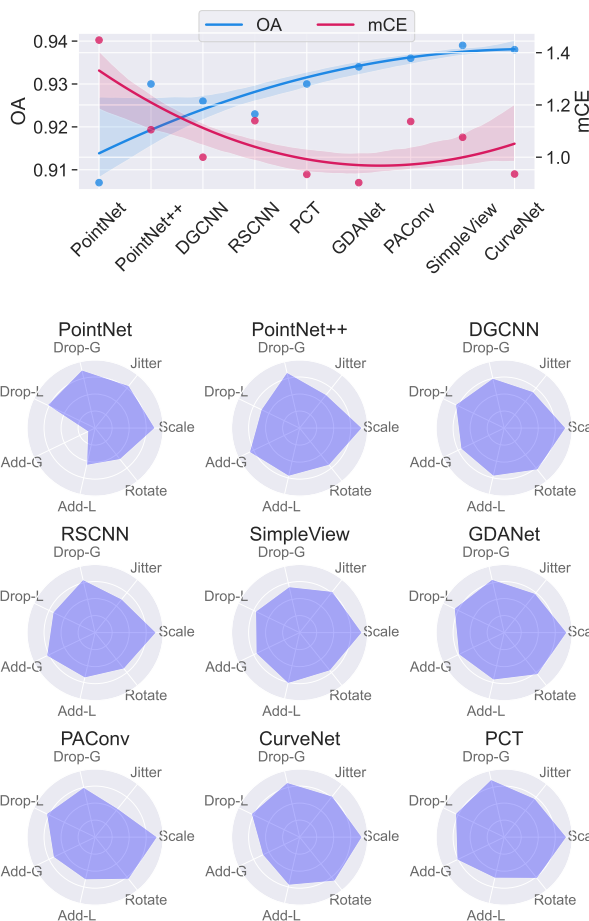[1]S-Lab, Nanyang Technological University. Correspondence to: Ziwei Liu <ziwei.liu@ntu.edu.sg>.

Figure 1. **Upper.** Blue curve shows overall accuracy (OA) on ModelNet40. Red curve shows mean Corruption Error (mCE) on proposed ModelNet-C. Methods are sorted in chronological order. OA gradually saturates but mCE is at the risk of increasing due to the lack of a standard test suite. **Lower.** Point cloud classifer's robustness to various corruptions in a radar chart. Proposed ModelNet-C allows fine-grained corruption analysis. Different architectures have diverse strengths and weaknesses to corruptions. "-G": -Global. "-L": -Local.

et al., 2017b; Simonovsky & Komodakis, 2017).

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

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

	Jitter	Drop Global	Drop Local	Add Global	Add Local	Scale	Rotate
PointNet (Qi et al., 2017b)	✓	✓		✓			
ECC (Simonovsky & Komodakis, 2017)	✓	✓					
PointNet++ (Qi et al., 2017a)		✓					
DGCNN (Wang et al., 2019)		✓					
RSCNN (Liu et al., 2019)		✓					✓
PointASNL (Yan et al., 2020)		✓		✓			
Orderly Disorder (Ghahremani et al., 2020)	✓						
PointAugment (Li et al., 2020)	✓	✓				✓	✓
PointMixup (Chen et al., 2020)	✓	✓				✓	✓
PAConv (Xu et al., 2021a)	✓					✓	✓
OcCo (Wang et al., 2021)		✓					
Triangle-Net (Xiao & Wachs, 2021)	✓	✓				✓	✓
Curve-Net (Xiang et al., 2021)	✓	✓					
RSMix (Lee et al., 2021)	✓	✓				✓	✓
PointWolf (Kim et al., 2021)	✓	✓	✓		✓		
GDANet (Xu et al., 2021b)		✓					✓
*Our Benchmark (**ModelNet-C**)*	✓	✓	✓	✓	✓	✓	✓

Protocol-1. Evaluate the robustness to a selected set of corruptions (Qi et al., 2017b;a; Wang et al., 2019; Chen et al., 2020; Kim et al., 2021), *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 (Reizenstein et al., 2021; Ahmadyan et al., 2021), *e.g.*, train on ModelNet40 (Wu et al., 2015) and test on ScanObjectNN (Uy et al., 2019). To exploit the naturally occurred 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 and ScanObjectNN may have different styles, which obfuscates the evaluation results.

Protocol-3. Evaluate the robustness to adversarial attack (Zhou et al., 2019; Dong et al., 2020; Sun et al., 2021), *e.g.*, adversarial point shifting and dropping. Different 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 (Figure 2), 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 atomic corruptions to the validation set of ModelNet40 as our corruption test suite dubbed *ModelNet-C*. Inspired by the 2D image classification robustness benchmark (Hendrycks & Dietterich, 2019), we further create 5 severity levels for each atomic corruption and use the mean Corruption Error (mCE) metric for evaluation. Finally, based on the test suite, we benchmark 14 point cloud classification methods, including 9 architectures, 3 augmentations, and 2 pretrains. As shown in Figure 1, our benchmark results show that although point cloud classification performance on the clean ModelNet40 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 with the SoTAs. In particular, we present WOLFMix, a strong augmentation baseline that exploits both deformation-
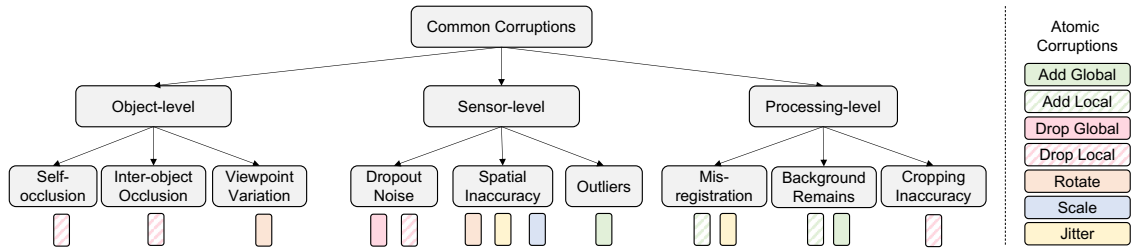
Figure 2. Corruption taxonomy. We break down common corruptions into detailed corruption sources on object-, senor- and processing levels, which are further simplified into a combination of seven atomic corruptions for a more controllable empirical analysis.

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 experiments, the performance gain by augmentations does not equally transfer to all model architectures. We identify the best combination from existing methods, and call for model design that fully exploits the augmentation power.

Our contributions are summarized as: **1)** We present the first systematically-designed test-suite *ModelNet-C* for point cloud classifier under corruptions. **2)** We comprehensively benchmark existing methods on their robustness to corruptions. **3)** We summarize several effective techniques, such as RPC and WOLFMix, to enhance 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. Point cloud classification serves as an fundamental task for 3D understanding from raw hardware inputs. Point cloud classifier has diverse architectural designs. There are MLP-based models (Qi et al., 2017b;a), convolution-based models (Liu et al., 2019; Xu et al., 2021a), graph-based models (Simonovsky & Komodakis, 2017; Wang et al., 2019) and recently proposed transformer-based models (Guo et al., 2020; Zhao et al., 2021; Mazur & Lempitsky, 2021). Besides, there is a rising discussion on point cloud augmentation, including mix-based augmentations (Chen et al., 2020; Lee et al., 2021), deformation-based augmentations (Kim et al., 2021) and auto-augmentations (Li et al., 2020). Moreover, self-supervised pre-train has drawn much research attention recently. Pre-trains obtained from pre-text tasks like occlusion reconstruction (Wang et al., 2021) and mask inpainting (Yu et al., 2021) provide better classification performance than random initialization.

Robustness in Point Cloud. Several attempts are made to improve point cloud classifier's robustness. Triangle-Net (Xiao & Wachs, 2021) designs feature extraction that is invariant to positional, rotational, and scaling disturbances. Although Triangle-Net achieves exceptional robustness un-

der extreme corruptions, its performance on clean data is not on par with SoTA. PointASNL (Yan et al., 2020) introduces adaptive sampling and local-nonlocal modules to improve robustness. However, PointASNL takes a fixed number of points in implementation. Other works improve a model's adversarial robustness by denoising and upsampling (Zhou et al., 2019), voting on subsampled point clouds (Liu et al., 2021), exploiting local feature's relative position (Dong et al., 2020) and self-supervision (Sun et al., 2021). Robust-PointSet (Taghanaki et al., 2020) evaluates the robustness of point cloud classifiers under different corruptions, and shows that basic data augmentations poorly generalize to "unseen" corruptions. However, our work shows that more advanced augmentation techniques, *e.g.*, mixing and local deformation, can substantially improve the robustness.

Robustness Benchmarks in Image Classification. Comprehensive robustness benchmark has been built for 2D image classification recently. ImageNet-C (Hendrycks & Dietterich, 2019) corrupts the ImageNet (Deng et al., 2009)'s test set with simulated corruptions like compression loss and motion blur. ObjectNet (Barbu et al., 2019) collects a test set with rich variations in rotation, background and viewpoint. ImageNetV2 (Recht et al., 2019) re-collects a test set following ImageNet's protocal and evaluates the performance gap due to the natural distribution shift. Moreover, ImageNet-A (Hendrycks et al., 2021b) and ImageNet-R (Hendrycks et al., 2021a) benchmark 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 2. 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
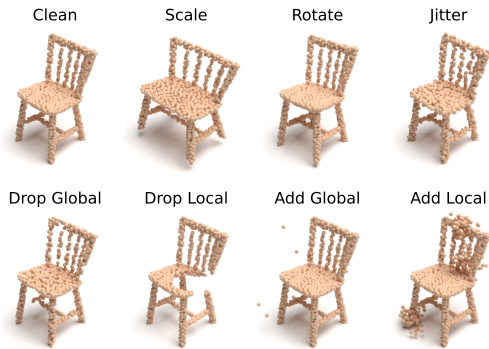
Figure 3. Examples of our proposed *ModelNet-C*. We corrupt the clean test set of *ModelNet-C* using seven types of corruptions with five levels of severity to provide a comprehensive robustness evaluation. Listed examples are from severity level 2. More visualizations on different severity levels can be found in the supplementary material.

perceiving with 3D sensors like LiDAR. As discussed in prior works (Wu et al., 2019; Berger et al., 2014), 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 of 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 the 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 noisy translation and random permutation are not considered in this work, because point cloud normalization and permutation-invariance are two basic properties of recent point cloud classification approaches.

3.2. ModelNet-C: A Robustness Test Suite

ModelNet40 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 (Qi et al., 2017b), which samples 1024 points from each aligned CAD model and then normalizes them into a unit sphere. Based on ModelNet40 and the settings by (Qi et al., 2017b), we further corrupt the ModelNet40 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. Similar corruption operations are strictly *not allowed* during training.

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 example corrupted point clouds from *ModelNet-C* are shown in Figure 3. 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. The detailed description and implementation can be found in the appendix. 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 (Zhang et al., 2019; Chen et al., 2019), which is out of the scope of this work.

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 (Hendrycks & Dietterich, 2019), we use mean CE (mCE), as the primary metric. To compute mCE, we first compute CE:

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

where $\text{OA}_{i,l}$ is the overall accuracy on a corrupted test set i at corruption level l, $\text{OA}_{i,l}^{\text{DGCNN}}$ is baseline's overall accuracy mCE is the average of CE over all seven corruptions:

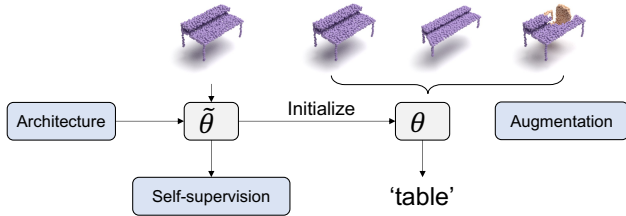$$\text{mCE} = \frac{1}{N}\sum_{i=1}^{N}\text{CE}_i, \tag{2}$$

Figure 4. Robust point cloud classification paradigm. Point cloud classification robustness to various corruptions largely depends on three main components, including architecture design, self-supervised pretraining and augmentation methods.

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:

$$\text{RCE}_i = \frac{\sum_{l=1}^{5}(\text{OA}_{\text{Clean}} - \text{OA}_{i,l})}{\sum_{l=1}^{5}(\text{OA}_{\text{Clean}}^{\text{DGCNN}} - \text{OA}_{i,l}^{\text{DGCNN}})}, \qquad (3)$$

$$\text{RmCE} = \frac{1}{N}\sum_{i=1}^{N}\text{RCE}_i, \qquad (4)$$

where OA_{Clean} is the overall accuracy on the clean test set.

3.4. Evaluation Protocol

Because most SoTA methods adopt the *DGCNN protocol* (Goyal et al., 2021), 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 test set during evaluation. We highlight that the same corruptions are not allowed during training to reflect model OOD generalizability. Following works are recommended to specify augmentations in training when reporting results on ModelNet-C.

4. Systematic Benchmarking

Implementation Details. We benchmark 14 methods in total, covering three key components for robust point cloud classification as shown in Figure 4. **Architectures**: Point-Net, PointNet++, DGCNN, RSCNN, SimpleView, GDANet, CurveNet, PAConv, PCT. **Pretrains**: OcCo, Point-BERT. **Augmentation**: PointMixup, RSMix, PointWOLF. For PointNet, PointNet++, DGCNN, RSCNN, and SimpleView, we use the pretrained models provided by Goyal et al. (2021). For CurveNet, GDANet, and PAConv, we use their official pretrained models. The rest of the models are trained using their official codes.

Table 2. Systematic study for architecture design.

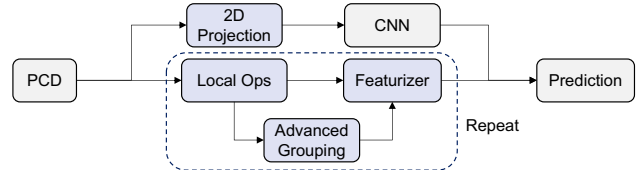|	Representation	Local Operations	Advanced Grouping	Featurizer	mCE($\downarrow$)
PointNet	3D	No	No	Conventional	1.422
PointNet++	3D	Ball-query	No	Conventional	1.072
DGCNN	3D	k-NN	No	Conventional	1.000
RSCNN	3D	Ball-query	No	Adaptive	1.130
PAConv	3D	k-NN	No	Adaptive	1.104
CurveNet	3D	k-NN	Curve	Conventional	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**

Figure 5. Key components in the 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-based backbones. This figure means to show how the key components are usually connected, but not to faithfully show every detailed architecture design.

Main Results. Benchmark results (mCE) are reported in Table 3, Table 4 and Table 5 for architechtures, pretrains and augmentations, respectively. RmCE and Overall Accuracy are reported in the appendix. In Figure 1, we sort benchmarked architectures in chronological order and visualize a second-order polynomial fitting results with 50% confidence interval. We observe that although new architecture's performance are 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 improvement robustness.

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 5. 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. Further-

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

	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN (Wang et al., 2019)	0.926	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PointNet (Qi et al., 2017b)	0.907	1.422	1.266	**0.642**	0.500	1.072	2.980	1.593	1.902
PointNet++ (Qi et al., 2017a)	0.930	1.072	0.872	1.177	0.641	1.802	**0.614**	0.993	1.405
RSCNN (Liu et al., 2019)	0.923	1.130	1.074	1.171	0.806	1.517	0.712	1.153	1.479
SimpleView (Goyal et al., 2021)	**0.939**	1.047	0.872	0.715	1.242	1.357	0.983	**0.844**	1.316
GDANet (Xu et al., 2021b)	0.934	0.892	**0.830**	0.839	0.794	0.894	0.871	1.036	0.981
CurveNet (Xiang et al., 2021)	0.938	0.927	0.872	0.725	0.710	1.024	1.346	1.000	**0.809**
PAConv (Xu et al., 2021a)	0.936	1.104	0.904	1.465	1.000	1.005	1.085	1.298	0.967
PCT (Guo et al., 2020)	0.930	0.925	0.872	0.870	0.528	1.000	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. †: randomly initialized. Bold: best in column. Underline: second best in column. Blue: best in row. Red: worst in row.

	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN (Wang et al., 2019)	**0.926**	**1.000**	1.000	1.000	1.000	**1.000**	**1.000**	1.000	1.000
+OcCo (Wang et al., 2021)	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 (Yu et al., 2021)	0.922	1.248	**0.936**	1.259	**0.690**	1.150	1.932	1.440	1.326

more, 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 6a, the exploitation of the point cloud locality is a key component to robustness. Without local aggregations, PointNet (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 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 its 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 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 (Xu et al., 2021b) and Curve Grouping (Xiang et al., 2021), to introduce structural prior into architecture design. Frequency grouping uses a graph high-pass filter (Sandryhaila & Moura, 2014; Ortega et al., 2018) 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 6b, we observe that *both grouping techniques improve model robustness by a clear margin.* The idea of frequency group-

ing aligns with the observations in (Yin et al., 2019): 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 feature 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 conventional operators to 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 (Liu et al., 2019) and PAConv (Xu et al., 2021a) design adaptive kernels whose weights change with low-level features like spatial coordinates and surface normals. Based on self-attention, PCT (Guo et al., 2020) proposes the offset-attention operation, which achieves impressive performance for point cloud analysis. Despite the success of RSCNN and PAConv on clean point cloud classifications, they tend to be more sensitive to corruptions than conventional operators in our experiments shown in Figure 6c. 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 (Yu et al., 2021) also introduces an 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 result in the architecture analysis.

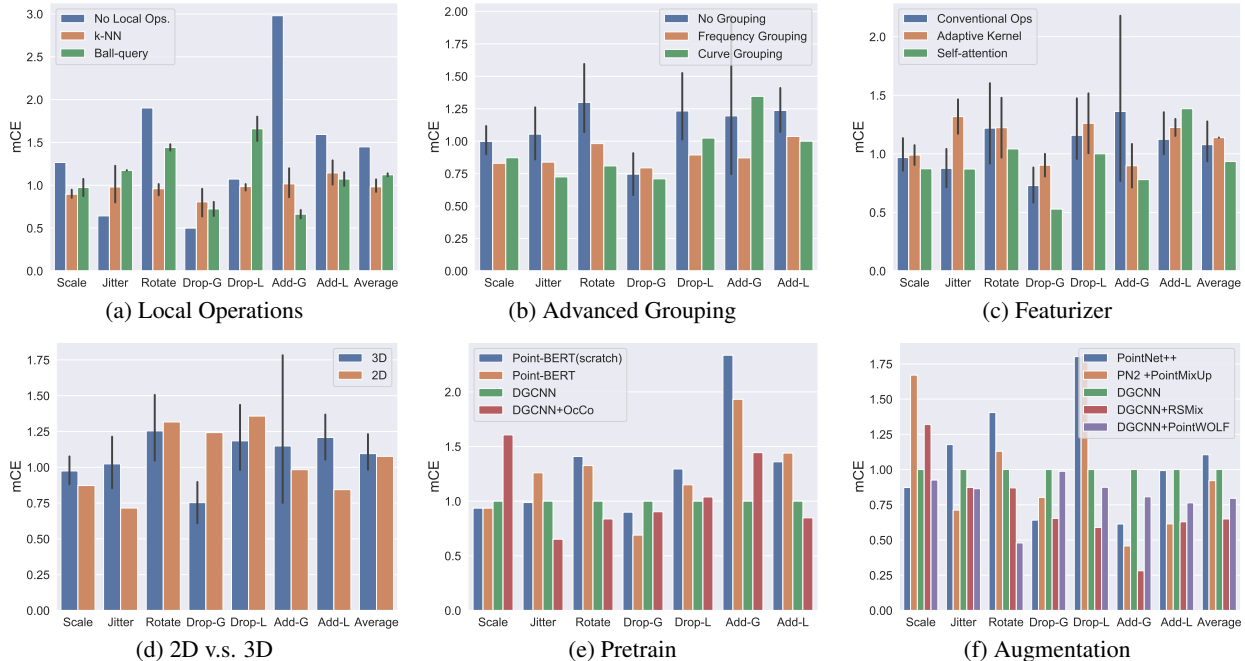**2D vs. 3D Representation.** A few methods (Qi et al., 2016;

Figure 6. Analysis on different architecture designs, pretrain strategies and augmentation strategies' effect to classifier's performance under different corruptions. "-G": Global. "-L": Local.

Goyal et al., 2021) 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 (Goyal et al., 2021) performs surprisingly well on clean 3D point clouds. In our experiments shown in Figure 6d, *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 straight-forward 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 (Yu et al., 2021) and OcCo (Wang et al., 2021). We study their robustness against corruptions in Figure 6e, which reveals that *pretrain signals can be transferred*, and hence benefiting classification under specific corruptions. During self-supervised pretrain, Point-BERT 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 pre-

train show better classification robustness when local part is missing. OcCo employs a similar reconstruction pretrain task, but with a different masking strategy. By observing from different camera viewpoints, OcCo masks the points that are self-occluded. Meanwhile, point clouds are also rotated with different camera angles. Consequently, the OcCo pretrained models are significantly more robust to rotation perturbations. Moreover, OcCo 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 6f, *mixing and deformation augmentations can bring significant improvements to model robustness.* PointMixUp (Chen et al., 2020) and RSMix (Lee et al., 2021) are two mix strategies. Similar to MixUp (Zhang et al., 2018) in 2D augmentation, PointMixup mixes two point clouds using shortest-path interpolation. Similar to CutMix (Yun et al., 2019) in 2D augmentation, RSMix 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 (Kim et al., 2021) enrich the data variation, which constantly improves classifier robustness on all evaluated corruptions.

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

	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN (Wang et al., 2019)	0.926	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
+PointWOLF (Kim et al., 2021)	0.926	0.814	0.926	0.864	0.988	0.874	0.807	0.764	0.479
+RSMix (Lee et al., 2021)	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++ (Qi et al., 2017a)	0.930	1.072	0.872	1.177	0.641	1.802	0.614	0.993	1.405
+PointMixUp (Chen et al., 2020)	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.

	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN (Wang et al., 2019)	0.926	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
+WOLFMix	0.932	0.590	0.989	0.715	0.698	0.575	0.285	0.415	0.451
PointNet (Qi et al., 2017b)	0.907	1.422	1.266	0.642	0.500	1.072	2.980	1.593	1.902
+WOLFMix	0.884	1.180	2.117	0.475	0.577	1.082	2.227	0.702	1.079
PCT (Guo et al., 2020)	0.930	0.925	0.872	0.870	0.528	1.000	0.780	1.385	1.042
+WOLFMix	0.934	0.574	1.000	0.854	0.379	0.493	0.298	0.505	0.488
GDANet (Xu et al., 2021b)	0.934	0.892	0.830	0.839	0.794	0.894	0.871	1.036	0.981
+WOLFMix	0.934	0.571	0.904	0.883	0.532	0.551	0.305	0.415	0.409
RPC	0.930	0.863	0.840	0.892	0.492	0.797	0.929	1.011	1.079
+WOLFMix	0.933	0.601	1.011	0.968	0.423	0.512	0.332	0.480	0.479

6. Boosting Corruption Robustness

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

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. The detailed architecture is shown in the appendix. 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. The implementation details are provided in the appendix.

WOLFMix: A Strong Augmentation Strategy. We design WOLFMix upon two powerful augmentation strategies, PointWOLF and RSMix. 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 7. By taking advantage of both rigid and non-rigid transformations, WOLFMix brings substantial robustness gain over standalone PointWOLF and RSMix in Table 5. Implementation details can be found in the appendix.

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 with different models. Although RPC achieves the lowest standalone mCE, its improvements by WOLFMix are less than
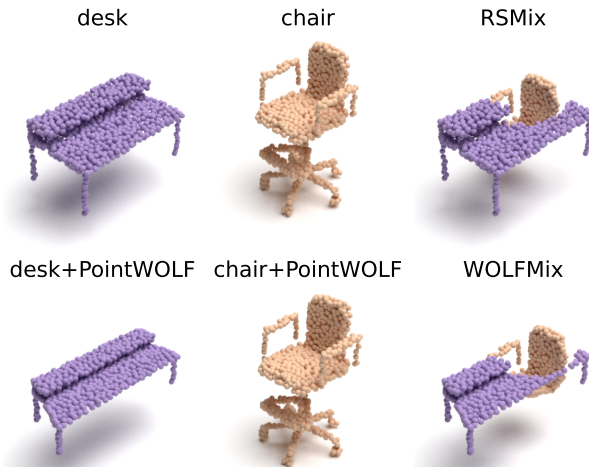

Figure 7. Illustration of the proposed WOLFMix augmentation. Point clouds are first locally deformed and then rigidly mixed. Ground truth labels are mixed accordingly.

WOLFMix for DGCNN, PCT and GDANet. PointNet enjoys 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 with WOLFMix achieves the best robustness in all existing methods, with an impressive 0.571 mCE.

7. Conclusion

In this work, we establish a comprehensive test suite *ModelNet-C* for robust point cloud classification under corruptions. 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.

8. Acknowledgment

References

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

Barbu, A., Mayo, D., Alverio, J., Luo, W., Wang, C., Gutfreund, D., Tenenbaum, J., and Katz, B. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *NeurIPS*, pp. 9448–9458, 2019.

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

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

Chen, Y., Hu, V. T., Gavves, E., Mensink, T., Mettes, P., Yang, P., and Snoek, C. G. M. Pointmixup: Augmentation for point clouds. In *ECCV (3)*, volume 12348 of *Lecture Notes in Computer Science*, pp. 330–345. Springer, 2020.

Deng, C., Litany, O., Duan, Y., Poulenard, A., Tagliasacchi, A., and Guibas, L. J. Vector neurons: A general framework for so (3)-equivariant networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12200–12209, 2021.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Dong, X., Chen, D., Zhou, H., Hua, G., Zhang, W., and Yu, N. Self-robust 3d point recognition via gather-vector guidance. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11513–11521. IEEE, 2020.

Ghahremani, M., Tiddeman, B., Liu, Y., and Behera, A. Orderly disorder in point cloud domain. In *ECCV (28)*, volume 12373 of *Lecture Notes in Computer Science*, pp. 494–509. Springer, 2020.

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

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

Hendrycks, D. and Dietterich, T. G. Benchmarking neural network robustness to common corruptions and perturbations. In *ICLR (Poster)*. OpenReview.net, 2019.

Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., and Gilmer, J. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021a.

Hendrycks, D., Zhao, K., Basart, S., Steinhardt, J., and Song, D. Natural adversarial examples. *CVPR*, 2021b.

Kim, S., Lee, S., Hwang, D., Lee, J., Hwang, S. J., and Kim, H. J. Point cloud augmentation with weighted local transformations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 548–557, October 2021.

Lee, D., Lee, J., Lee, J., Lee, H., Lee, M., Woo, S., and Lee, S. Regularization strategy for point cloud via rigidly mixed sample. In *CVPR*, pp. 15900–15909. Computer Vision Foundation / IEEE, 2021.

Li, R., Li, X., Heng, P., and Fu, C. Pointaugment: An auto-augmentation framework for point cloud classification. In *CVPR*, pp. 6377–6386. Computer Vision Foundation / IEEE, 2020.

Liu, H., Jia, J., and Gong, N. Z. Pointguard: Provably robust 3d point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6186–6195, 2021.

Liu, Y., Fan, B., Xiang, S., and Pan, C. Relation-shape convolutional neural network for point cloud analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8895–8904, 2019.

Mazur, K. and Lempitsky, V. Cloud transformers: A universal approach to point cloud processing tasks. In *International Conference on Computer Vision (ICCV)*, 2021.

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M., and Vandergheynst, P. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

Qi, C., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017a.

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., and Guibas, L. J. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2016.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017b.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. Do imagenet classifiers generalize to imagenet? In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5389–5400. PMLR, 2019.

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

Sandryhaila, A. and Moura, J. M. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014.

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Sun, J., Cao, Y., Choy, C., Yu, Z., Anandkumar, A., Mao, Z. M., and Xiao, C. Adversarially robust 3d point cloud recognition using self-supervisions. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

Taghanaki, S. A., Luo, J., Zhang, R., Wang, Y., Jayaraman, P. K., and Jatavallabhula, K. M. Robustpointset: A dataset for benchmarking robustness of point cloud classifiers. *arXiv preprint arXiv:2011.11572*, 2020.

Uy, M. A., Pham, Q., Hua, B., Nguyen, D. T., and Yeung, S. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, pp. 1588–1597. IEEE, 2019.

Wang, H., Liu, Q., Yue, X., Lasenby, J., and Kusner, M. J. Unsupervised point cloud pre-training via occlusion completion. In *International Conference on Computer Vision, ICCV*, 2021.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019.

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

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

Xiang, T., Zhang, C., Song, Y., Yu, J., and Cai, W. Walk in the cloud: Learning curves for point clouds shape analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 915–924, October 2021.

Xiao, C. and Wachs, J. P. Triangle-net: Towards robustness in point cloud learning. In *WACV*, pp. 826–835. IEEE, 2021.

Xu, M., Ding, R., Zhao, H., and Qi, X. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *CVPR*, 2021a.

Xu, M., Zhang, J., Zhou, Z., Xu, M., Qi, X., and Qiao, Y. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. In *AAAI*, pp. 3056–3064. AAAI Press, 2021b.

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

Yin, D., Lopes, R. G., Shlens, J., Cubuk, E. D., and Gilmer, J. A fourier perspective on model robustness in computer vision. In *NeurIPS*, pp. 13255–13265, 2019.

Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., and Lu, J. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. *arXiv preprint*, 2021.

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

Zhang, H., Cissé, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *ICLR (Poster)*. OpenReview.net, 2018.

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

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268, 2021.

Zhou, H., Chen, K., Zhang, W., Fang, H., Zhou, W., and Yu, N. Dup-net: Denoiser and upsampler network for 3d adversarial point clouds defense. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1961–1970, 2019.

A. Corruptions and Severity Level Settings

We elaborate on the implementation of corruptions and severity level settings in this section. A visualization is shown in Figure 8.

A.1. 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.

A.2. 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.

A.3. 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 sufficient to cover a range of rotation variations.

A.4. 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.

A.5. 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 neighbour points (including itself) from the point cloud.

A.6. 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.

A.7. 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 $\mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2 I)$, 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.

B. Implementation Details

We elaborate on implementation details for RPC and WOLFMix in this section.

B.1. RPC

B.1.1. DETAILED ARCHITECTURE

We show a detailed architecture of RPC in Figure 9.

Figure 8. Corruptions on all levels. The severity of corruptions increases with the level. We average model's error on all levels for a comprehensive evaluation.
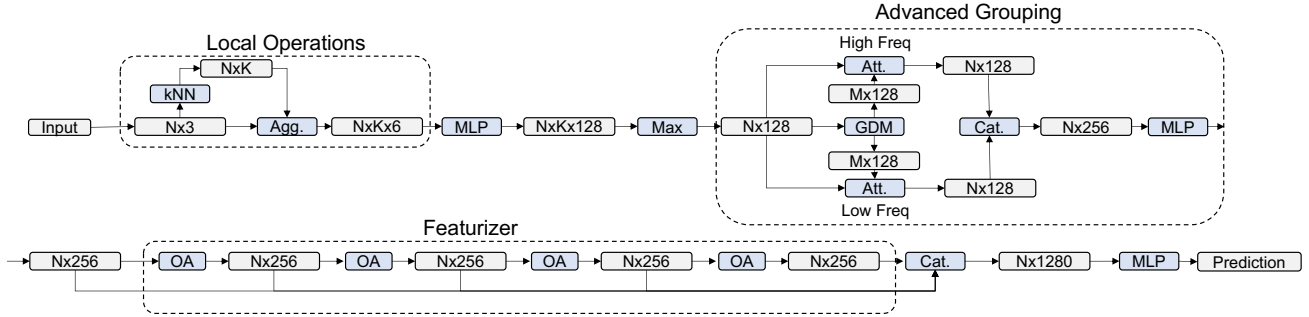
Figure 9. Detailed architecture of RPC. We design RPC following the conclusions we draw from the benchmark. It optimizes the use of existing building blocks in point cloud classifiers and serves as a strong baseline for corruption robustness.

B.1.2. HYPER-PARAMETERS

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 (Xu et al., 2021b). The number of points in each frequency component is set to 256.

B.1.3. TRAINING

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

B.2. WOLFMix

For the deformation step, we use the default hyper-parameters in PointWOLF (Kim et al., 2021). We set the number of anchors to 4, 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 is not used in training. For the mixing step, we use the default hyper-parameters in RSMix (Lee et al., 2021). We set RSMix probability to 0.5, β to 1.0, and the maximum number of point modifications to 512. For training, the number of neighbors in k-NN is reduced to 20 and the number of epochs is increased to 500 for all methods.

C. Additional Discussions

C.1. Correlation between ModelNet-C mCE and ScanObjectNN OA

We additionally evaluate all models on ScanObjectNN (Uy et al., 2019), and we use the *OBJ_BG PB_T25* variant to include both background remains and bounding box inaccuracy. The results are shown in Figure 10, and ModelNet-C mCE strongly correlates to ScanObjectNN OA, while ModelNet40 OA has nearly no correlations. Note that the results we report are lower than the results originally reported in Uy et al. (2019) due to different training protocols. Uy et al. (2019) uses random rotation and per-point jitter in training while we follow the *DGCNN protocol* (Goyal et al., 2021).

Table 7. Additional results of WolfMix.

	OA	mCE
PointNet++ (Qi et al., 2017a)	0.931	0.641
RSCNN (Liu et al., 2019)	0.918	0.601
SimpleView (Goyal et al., 2021)	0.922	0.676

C.2. More results of Wolfmix

We report additional results on RSCNN, SimpleView and PointNet++ with WolfMix in Table 7. The unequal benefits from augmentations motivate future research to explore the synergy between architecture and augmentation.
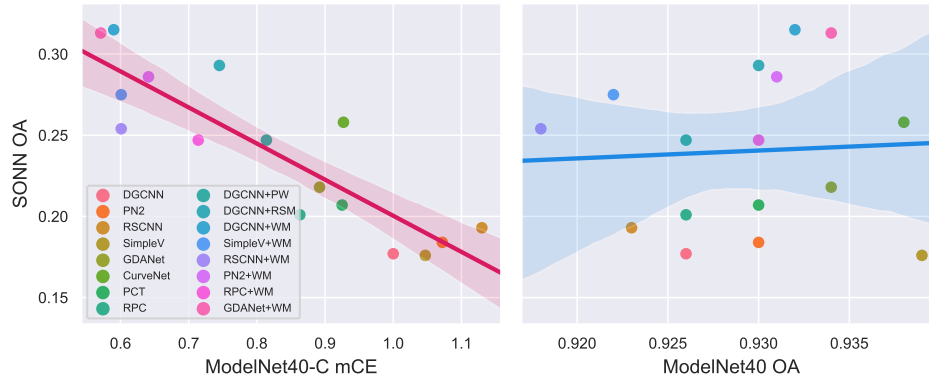
Figure 10. Correlation between ModelNet-C mCE and ScanObjectNN OA.

Table 8. More techniques.

	OA	mCE	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
PointASNL (Yan et al., 2020)	0.918	0.959	1.191	0.687	0.944	0.826	0.959	0.953	1.153
Vector Neuron (Deng et al., 2021)	0.908	1.345	1.287	1.601	1.875	1.754	0.902	1.567	0.428

C.3. Evaluation on specific techniques proposed for enhancing robustness

There are a few methods for robustness enhancement. TriangleNet (Xiao & Wachs, 2021)'s clean performance is not comparable to SoTA and PointASNL (Yan et al., 2020) requires a fixed number of points. Nonetheless, we manage to evaluate PointASNL with additional manual efforts and show the results in Table 8. PoinASNL shows outstanding performance to noisy jittering and achieves a competitive overall mCE result.

C.4. Evaluation on works designed for rotation robustness

Robustness to arbitrary SO(3) rotations is out of the scope of our benchmark where we examine common corruptions like small view angle variation. Nevertheless, we evaluate Vector Neurons (Deng et al., 2021), a rotation-invariant model, and show the results in Table 8. Vector Neuron achieves impressive robustness to rotational corruption, but under-performs to other types of corruption.

D. Full Results

We show full results for the OA metric and the RmCE metric in Table 9 and Table 10.

Table 9. Full results for Overall Accuracy (OA). †: randomly initialized. Bold: best in column. Underline: second best in column. Blue: best in row. Red: worst in row. mOA: average OA over all corruptions.

	Clean ↑	mOA↑	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN (Wang et al., 2019)	0.926	0.764	0.906	0.684	0.752	0.793	0.705	0.725	0.785
PointNet (Qi et al., 2017b)	0.907	0.658	0.881	0.797	0.876	0.778	0.121	0.562	0.591
PointNet++ (Qi et al., 2017a)	0.930	0.751	0.918	0.628	0.841	0.627	0.819	0.727	0.698
RSCNN (Liu et al., 2019)	0.923	0.739	0.899	0.630	0.800	0.686	0.790	0.683	0.682
SimpleView (Goyal et al., 2021)	0.939	0.757	0.918	0.774	0.692	0.719	0.710	0.768	0.717
GDANet (Xu et al., 2021b)	0.934	0.789	0.922	0.735	0.803	0.815	0.743	0.715	0.789
CurveNet (Xiang et al., 2021)	0.938	0.779	0.918	0.771	0.824	0.788	0.603	0.725	0.826
PAConv (Xu et al., 2021a)	0.936	0.730	0.915	0.537	0.752	0.792	0.680	0.643	0.792
PCT (Guo et al., 2020)	0.930	0.781	0.918	0.725	0.869	0.793	0.770	0.619	0.776
RPC (Ours)	0.930	0.795	0.921	0.718	0.878	0.835	0.726	0.722	0.768
DGCNN+OcCo (Wang et al., 2021)	0.922	0.766	0.849	0.794	0.776	0.785	0.574	0.767	0.820
Point-BERT†	0.919	0.678	0.912	0.688	0.777	0.732	0.311	0.626	0.697
Point-BERT (Yu et al., 2021)	0.922	0.693	0.912	0.602	0.829	0.762	0.430	0.604	0.715
PN2+PointMixUp (Chen et al., 2020)	0.915	0.785	0.843	0.775	0.801	0.625	0.865	0.831	0.757
DGCNN+PW (Kim et al., 2021)	0.926	0.809	0.913	0.727	0.755	0.819	0.762	0.790	0.897
DGCNN+RSMix (Lee et al., 2021)	0.930	0.839	0.876	0.724	0.838	0.878	0.917	0.827	0.813
DGCNN+WOLFMix (Ours)	0.932	0.871	0.907	0.774	0.827	0.881	0.916	0.886	0.903
PointNet+WOLFMix	0.884	0.743	0.801	0.850	0.857	0.776	0.343	0.807	0.768
PCT+WOLFMix	0.934	0.873	0.906	0.730	0.906	0.898	0.912	0.861	0.895
GDANet+WOLFMix	0.934	0.871	0.915	0.721	0.868	0.886	0.910	0.886	0.912
RPC+WOLFMix	0.933	0.865	0.905	0.694	0.895	0.894	0.902	0.868	0.897

Table 10. Full results for Relative mCE. †: random initialized. Bold: best in column. Underline: second best in column. Blue: best in row. Red: worst in row.

	RmCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN (Wang et al., 2019)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
PointNet (Qi et al., 2017b)	1.488	1.300	0.455	0.178	0.970	3.557	1.716	2.241
PointNet++ (Qi et al., 2017a)	1.114	0.600	1.248	0.511	2.278	0.502	1.010	1.645
RSCNN (Liu et al., 2019)	1.201	1.200	1.211	0.707	1.782	0.602	1.194	1.709
SimpleView (Goyal et al., 2021)	1.181	1.050	0.682	1.420	1.654	1.036	0.851	1.574
GDANet (Xu et al., 2021b)	0.865	0.600	0.822	0.753	0.895	0.864	1.090	1.028
CurveNet (Xiang et al., 2021)	0.978	1.000	0.690	0.655	1.128	1.516	1.060	0.794
PAConv (Xu et al., 2021a)	1.211	1.050	1.649	1.057	1.083	1.158	1.458	1.021
PCT (Guo et al., 2020)	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 (Wang et al., 2021)	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 (Yu et al., 2021)	1.262	0.500	1.322	0.534	1.203	2.226	1.582	1.468
PN2+PointMixUp (Chen et al., 2020)	1.254	3.600	0.579	0.655	2.180	0.226	0.418	1.121
DGCNN+PointWOLF (Kim et al., 2021)	0.698	0.650	0.822	0.983	0.805	0.742	0.677	0.206
DGCNN+RSMix (Lee et al., 2021)	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	0.488	1.400	0.843	0.161	0.271	0.100	0.363	0.277
GDANet+WOLFMix	0.439	0.950	0.880	0.379	0.361	0.109	0.239	0.156
RPC+WOLFMix	0.517	1.400	0.988	0.218	0.293	0.140	0.323	0.255