BENCHMARKING AND IMPROVING ROBUSTNESS OF 3D POINT CLOUD RECOGNITION AGAINST COMMON CORRUPTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks on 3D point cloud data have been widely used in the real world, especially in safety-critical applications. However, their robustness against corruptions is less studied. In this paper, we present **ModelNet40-C**, a comprehensive benchmark on 3D point cloud *corruption robustness*, consisting of 15 common and realistic corruptions. Our evaluation shows a significant gap between the performances on ModelNet40 and ModelNet40-C for state-of-the-art models. We identify a number of critical insights for future studies on corruption robustness in point cloud recognition. For instance, we unveil that Transformer-based architectures with proper training recipes achieve the strongest robustness. To bridge this gap, we further propose **RobustNet** and **PointCutMixup** that embrace the merits of existing architectural designs to further improve the corruption robustness in the 3D point cloud domain, after evaluating a wide range of augmentation and test-time adaptation strategies. Our codebase and dataset are open-sourced.

1 INTRODUCTION

Point clouds are one of the most acknowledged data format in 3D computer vision tasks, as they are inherently flexible representations and can be retrieved from a variety of sensors and computer-aided design (CAD) models. Because of these strengths, point clouds have been increasingly used in real-world applications, particularly in safety-critical areas like self-driving cars [71], robotics [40], medical imaging [62], and virtual and augmented reality (VR/AR) [34]. Processing of point clouds is thus crucial under these circumstances. For instance, autonomous vehicles rely on correct recognition of LiDAR point clouds to perceive the surroundings. Similar to 2D image classification, recent efforts demonstrate that deep learning models has dominated the point cloud recognition task.

As opposed to stellar progress on model architectures in 2D computer vision, deep 3D point cloud recognition is emerging where various architectures and operations are being proposed. PointNet [41] innovates to achieve end-to-end learning on point clouds. A few studies optimize the convolutional operation to be preferable for 3D point cloud learning [63, 32]. Various grouping operations are designed to learn local features [68, 63]. Transformer [59] blocks are also applied as backbones in point cloud recognition architectures [18, 74]. The most extensively utilized benchmark for comparing methods for point cloud recognition is ModelNet40 [66]. Although the accuracy on ModelNet40 over the past several years has been steadily improved, it merely shows a single perspective of model performance on the clean data and gets saturated. Given the importance of 3D point cloud in safety-critical applications, a comprehensive *robustness* benchmark for point cloud recognition models is necessary.

In the literature, the vast majority of research on robustness in 3D point cloud recognition has concentrated on the critical difficulties of robustness against adversarial examples. Adversarial training has been adapted to defend against various threats to point cloud learning [54, 53]. However, we find that the inevitable sensor inaccuracy and physical constraints will result in a number of *common corruptions* on point cloud data. For example, occlusion is a typical corruption for LiDAR and other scanning devices, rendering partially visible point clouds. Digital noises are also ubiquitous in 3D medical imaging. Such corruptions pose a even bigger threat in most real-world application scenarios. Therefore, it is imperative to study the corruption robustness of 3D point cloud recognition.

Summary of Our Contributions:

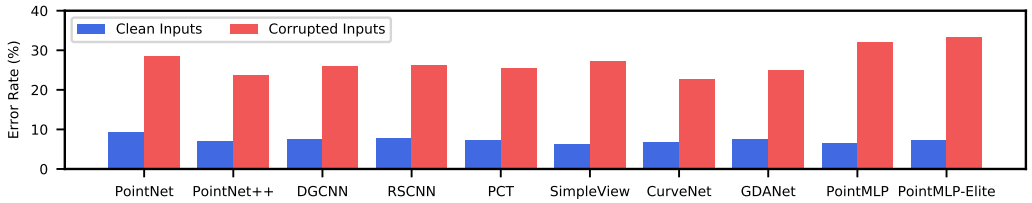

Figure 1: Despite the impressive results on clean inputs (*i.e.*, ModelNet40), state-of-the-art point cloud recognition models cannot deliver good performance on corrupted inputs (*i.e.*, ModelNet40-C). The error rate is **3**$\times$ larger on ModelNet40-C than ModelNet40. Standard deviation from 3 runs are always less than 0.3%.

In this paper, we create a systematic corruption robustness benchmark, ModelNet40-C, for 3D point cloud recognition and present an in-depth analysis. To construct the dataset, we meticulously design and formulate 75 corruptions (15 types with 5 severity levels) that cover the majority of real-world point cloud distortion cases. We further provide a taxonomy of these corruptions into three categories (*i.e*, density, noise and transformation) and discuss their application scenarios. We anticipate that ModelNet40-C will serve as a first step towards 3D point cloud corruption-resistant models.

We conduct extensive evaluation on our ModelNet40-C. Specifically, we compare 9 representative models including PointNet [41], PointNet++ [42], DGCNN [63], RSCNN [32], PCT [18], SimpleView [17], GDANet [69], CurveNet [68], and PointMLP [33]. We find that all models are extremely vulnerable to our created corruptions. As shown in Fig. 1, there are 3$\times$ error rate gaps between model performances on ModelNet40 and ModelNet40-C. Especially, the current state-of-the-art (SOTA), PointMLP, delivers the *worst* performance on our benchmark with an error rate of 31.9% (Fig. 1), which is a **4**$\times$ gap compared to its clean performance. This concerning result further demonstrate the urge for new benchmarks in the 3D point cloud community.

To mitigate such gaps, we propose a simple but effective strategy by combing PointCutMix-R and TENT, after evaluating a wide range of data augmentation and test-time adaptation methods. Our method on average achieves the lowest error rate of 15.2%. Specifically, we try augmentation (or regularization) strategies including PointCutMix-R/K [73], PointMixup [8], RSMix [26], and adversarial training [53] based strategies. Additionally, we employ test-time adaptation methods (*i.e.*, BN [45] and TENT [60]) to show their potential in improving corruption robustness. We examine *every* feasible combinations of architectures and methodologies, a total of *5,700* different configurations of experiments (76 strategies $\times$ 75 corruptions).

We summarize four conclusive insights below and eleven detailed findings in § 5.

• ***Insight 1***. Occlusion corruptions, rotation transformation, and background noises pose significant challenges for most point cloud recognition models (§ 5.1).

• ***Insight 2***. Different architectures and operations are vulnerable to different corruption types, which can be attributed to their design principles. (§ 5.2).

• ***Insight 3***. Different data augmentation strategies are especially advantageous for certain types of corruptions, which also correlate well with their design choices (§ 5.3).

• ***Insight 4***. Test-time adaptations (BN and TENT) are beneficial for enhancing the corruption robustness, particularly for hard corruptions like occlusions and rotations (§ 5.4).

Based on our benchmarking insights, we further propose **RobustNet** and **PointCutMixup**, embracing the merits of existing designs to improve the corruption robustness of point cloud recognition. RobustNet with PointCutMixup set the new SOTA robustness with an error rate of 14.9%. We hope our comprehensive benchmark and in-depth analysis will shed light and facilitate future research on the corruption robustness of point cloud recognition.

2 Related Work

Corruption Robustness of 2D Images. Deep neural networks are known to be vulnerable to adversarial examples and common corruptions [4]. [21, 20] developed corruption robustness benchmarking datasets CIFAR-10/100-C, ImageNet-C, and ImageNet-R to facilitate robustness evaluations of CIFAR and ImageNet classification models. [36] extended this benchmark to object detection models. [37] further proposed ImageNet-$\bar{\text{C}}$ dataset that is comprised of a set of corruptions that are perceptually dissimilar to ImageNet-C. Recently, [55] proposed a comprehensive benchmarking suite CIFAR-10/100-F that contains corruptions from different regions in the spectral domain. [22, 10, 5] proposed augmentation methods to improve the corruption robustness in 2D vision tasks.

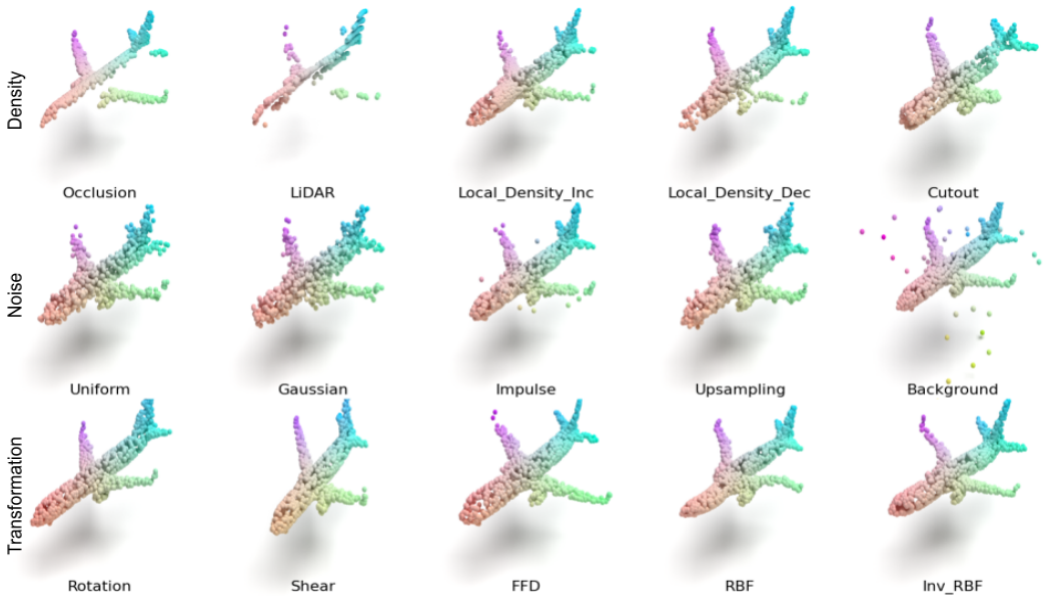

Figure 2: Visualizations of ModelNet40-C. Our ModelNet40-C dataset consists of 15 corruption types that represent different out-of-distribution shifts in real-world applications of point clouds. Similar to ImageNet-C [21], each corruption type has five severity levels. We carefully examine the generated point clouds and ensure they preserve their original semantics. More visualization samples are shown in Appendix A.

3D Point Cloud Deep Learning. Deep learning models are increasingly being proposed to process point cloud data. Early works attempted to use 3D voxel grids for perception, which have cubic complexity [35, 61]. PointNet [41] pioneered to leverage shared multi-layer perceptrons and a global pooling operation to achieve permutation-invariance and thus enable end-to-end training. [42] further proposed PointNet++ to hierarchically stack PointNet for multi-scale local feature encoding. PointCNN and RSCNN refactor the traditional pyramid CNN to improve the local feature learning for point cloud recognition [28, 32]. The graph data structure is also heavily used in point cloud learning [25, 49]. For example, DGCNN built a dynamic graph of point cloud data for representation learning [63]. PointConv and KPConv improve the convolution operation for point cloud learning [65, 57]. Recent work demonstrated that ResNet [19] on multi-view 2D projections of point clouds could also achieve high accuracy [17]. PointTransformer and PCT advance Transformer [59] blocks into point cloud learning and achieve good performance [74, 18]. Various local clustering operations [68, 69, 33] also show enhancements on the clean performance.

Robustness Enhancements for 3D Point Cloud. Several recent efforts tackle improving the robustness of 3D point cloud learning [52]. [67] and [29] first demonstrated that point cloud recognition is vulnerable to adversarial attacks. [76] and [13] proposed to leverage input randomization techniques to mitigate such vulnerabilities. [54] conducted adaptive attacks on existing defenses and analyzed the application of adversarial training on point cloud recognition. [75] discovered that adversarial rotation greatly degrades the perception performance. [53] further showed that pre-training on self-supervised tasks enhances the adversarial robustness of point cloud recognition. Recent studies presented a framework that uses the Shapley value [44] to assess the quality of representations learned by different point cloud recognition models [47, 48]. Recent efforts also proposed certified adversarial defenses[30]. Besides, [56] proposed several simple corruption types (*e.g.*, random sampling) to benchmark the robustness of point cloud recognition models. However, their formulations cannot represent realistic distortions in the physical world and they do not provide in-depth analysis of these corruptions among the representative recognition models. In this work, we aim to present a more systematic benchmark of the realistic corruptions and rigorously analyze the corruption robustness of representative point cloud recognition models.

3 Common Corruptions of 3D Point Cloud

In this section, we introduce the design principles of our 3D corruption benchmark. As mentioned in § 1, 3D point clouds are being utilized in various safety- and security-critical real-world applications [16, 1, 72, 9]. Extensive studies have been carried out to improve both architectures and training strategies for point cloud recognition on in-distribution data [41, 63, 8, 26]. However, there

has not been any systematic study on the model robustness against common corruption. To bridge this gap, we design 15 common corruptions for benchmarking *corruption robustness* of point cloud recognition models. It is worth noting that such designs are *non-trivial* since the manipulation space of 3D point clouds is completely different from 2D images where the corruptions come from the RGB modification [21]. In particular, we have three principles to design our benchmarks: **i)** Since we directly manipulate the position of points, we need to take extra care to preserve the *original semantics* of point clouds (Fig. 2). **ii)** we should ensure the constructed corruptions are *realistic* in various applications. **iii)** We should take *diversity* as an important factor to emulate a wide range of natural corruptions for 3D point clouds.

Our 15 corruption types can be naturally grouped into three categories (*i.e.*, density, noise, and transformation) , and we will introduce them in the following subsections.

3.1 Density Corruption Patterns

3D point cloud can be collected from various sensors like VR scanning devices and LiDAR for autonomous driving. Therefore, the testing point clouds may have different density patterns from the training samples. For example, VR scanning (in indoor scenes) and LiDAR sensors may suffer from occlusion, so that only a portion of the point cloud is visible [16, 11]. Besides, the direct reflection of lasers on metal materials will cause local missing points in LiDAR point clouds [31]. The local density of 3D scanned point clouds rely on how frequently the device passes that area [38]. We hence formulate five corruption types to cover the density corruption patterns: {`Occlusion`, `LiDAR`, `Local_Density_Inc`, `Local_Density_Dec`, `Cutout`}. Specifically, `Occlusion` and `LiDAR` both simulate occlusion patterns using ray tracing on the original meshes [78], and `LiDAR` additionally incorporates the vertically line-styled pattern of LiDAR point clouds [31]. `Local_Density_Inc` and `Local_Density_Dec` will randomly select several local clusters of points using k-nearest neighbors (kNN) to increase and decrease their density, respectively. Similarly, `Cutout` discards several randomly chosen local clusters of points using kNN [12].

3.2 Noise Corruption Patterns

Noise evidently exists in all real-world point cloud applications. For example, the inevitable digital noise of scanning sensors (*e.g.*, 3D medical imaging) [64] and the random reflections and inaccuracy of LiDAR lasers [16] will contribute to a substantial variation of points. Compression and decompression will potentially result in noisy point clouds as well [6]. Besides, real-time rendering in VR games is another source of noise [3]. Although noise is a common corruption pattern for both 2D and 3D data, the manipulation space is larger for point clouds since their numbers of points are adjustable. We thus formulate five noise perturbations: {`Uniform`, `Gaussian`, `Impulse`, `Upsampling`, `Background`}. As their names indicate, `Uniform` and `Gaussian` apply different distributional noise to each point in a point cloud. `Impulse` applies deterministic perturbations to a subset of points. `Upsampling` assigns new perturbation points around the existing points. `Background` randomly adds new points in the bounding box space of the pristine point cloud.

3.3 Transformation Corruptions Patterns

We use both linear and non-linear 3D transformations to formulate the corruptions. For the linear ones, we leverage 3D `Rotation` and `Shear` as our corruption types and exclude translation and scale transformations since they can be easily restored by normalization (*i.e.*, the inverse transformation matrix). `Rotation` of point clouds is common in the real world and the robustness against adversarial rotations has been investigated by a few studies [75, 47]. We here do not use aggressive rotations that might affect human perception as well, but instead enable a milder rotation ($\leq 15^\circ$) along xyz axes. We consider `Shear` on the xy plane to represent the motion distortion in 3D point clouds [70]. We utilize free-form deformation (FFD) [46] and radial basis function (RBF)-based deformation [15] for non-linear transformations. Such deformations are also common in VR/AR games and point clouds from generative models (GAN) [27, 77]. Specifically, we use multi quadratic ($\varphi(\boldsymbol{x}) = \sqrt{\boldsymbol{x}^2 + r^2}$) and inverse multi quadratic splines ($\varphi(\boldsymbol{x}) = (\boldsymbol{x}^2 + r^2)^{-\frac{1}{2}}$) as the representative RBFs to cover a wide range of deformation types. As a result, we in total formulate {`Rotation`, `Shear`, `FFD`, `RBF`, `Inv_RBF`} as our transformation-based corruptions.

4 MODELNET40-C DATASET AND ROBUSTNESS BENCHMARK

ModelNet40 is the most popular dataset for benchmarking point cloud recognition performance, containing 12,308 point clouds from 40 classes [66]. Point clouds from ModelNet40 are extracted from CAD models, rendering a perfectly clean dataset. [58] recently proposed ScanObjectNN consisting of point clouds scanned from real-world objects to show the performance gap between models trained on synthetic and real-world data. However, quantifying how models trained on the clean dataset perform under common corruptions encountered during the test time remains challenging. To this end, we use the ModelNet40 as our base dataset to construct ModelNet40-C. It is worth noting that our devised corruptions are general to be applied to other dataset, like ShapeNet [7].

Setup. We create ModelNet40-C with five severity levels for each corruption type, the same as ImageNet-C. Fig. 2 illustrates samples from ModelNet40-C with severity level four, and they clearly still preserve the semantics of the "airplane" class. Since it is hard to qualify and quantify the corruption severity for `LiDAR` and `Occlusion`, we instead leverage five different view angles to create their corrupted point clouds. The detailed construction of ModelNet40-C is introduced in Appendix A. These designed corruptions are applied to the *validation* set of ModelNet40, resulting in ModelNet40-C a 75$\times$ larger dataset to test the corruption robustness of pre-existing models. *Note that ModelNet40-C should be only used in the test time rather than in the training phase.*

Metrics. We use the error rate (ER) and class-wise mean error rate (mER) as the main metrics for ModelNet40-C benchmarking. We denote $\text{ER}^f_{\text{clean}}$ as the error rate for a classifier f on the clean dataset (*i.e.*, ModelNet40) and $\text{ER}^f_{s,c}$ as the error rate for f on corruption c with severity s. Similarly, $\text{ER}^f_c = \frac{1}{5}\sum_{s=1}^{5}\text{ER}^f_{s,c}$ and $\text{ER}^f_{\text{cor}} = \frac{1}{15}\sum_{c=1}^{15}\text{ER}^f_{\text{c}}$. The goal of ModelNet40-C is to evaluate the general robustness of point cloud learning models in various real-world scenarios.

5 EXPERIMENTS

In this section, we elaborate our comprehensive evaluation and rigorous analysis in detail. We benchmark corruption robustness of point cloud recognition from the perspectives of corruption types and model architectures. Moreover, we examine the effectiveness of data augmentations and test-time adaptation methods as mitigation solutions.

Table 1: Error Rates of Different Models with Training Strategies on the ModelNet40 (ER_{clean}). **Bold** and underline denote the best and runner-up results throughout this paper, respectively.

Model (%) ↓	Standard	PointCutMix-R	PointCutMix-K	PointMixup	RSMix	AT
PointNet	9.3	9.4	9.0	8.9	9.8	11.8
PointNet++	7.0	7.1	6.7	7.1	6.6	-
DGCNN	7.4	7.4	6.8	7.8	7.1	**8.1**
RSCNN	7.7	7.6	7.1	7.2	7.6	-
PCT	7.1	7.2	6.9	7.4	6.9	8.9
SimpleView	**6.1**	7.9	7.4	7.2	7.9	-
CurveNet	6.6	6.7	6.3	6.8	**6.1**	-
GDANet	7.5	7.5	6.9	7.6	7.2	-
PointMLP	6.3	**6.2**	**6.0**	**6.5**	**6.1**	-

Setup. We leverage 9 representative model architectures: PointNet [41], PointNet++ [41], DGCNN [63], RSCNN [32], PCT [18], SimpleView [17], GDANet [69], CurveNet [68], and PointMLP [33]. These 9 models stand for distinct architecture designs, and have achieved good accuracy on the clean dataset. They are also well-recognized by the 3D vision community, and have been extensively applied to complex tasks like semantic segmentation [38] and object detection [51, 50]. As suggested by [17], we adopt the same training strategy for all models. We utilize smoothed cross-entropy [63] as the loss function as it has been demonstrated to improve the recognition performance. We take 1024 points as input size in the training phase and use the Adam optimizer [24] with the `ReduceLROnPlateau` scheduler implemented in PyTorch [39]. We train 300 epochs and pick the best performant model for our further evaluation and follow [63] to use random translation and scaling as our default data augmentation. All training and testing experiments are done on a GeForce RTX 2080 GPU. We report the class-wise mean ER in Appendix C.

Besides, we try data augmentation and test-time adaption strategies and evaluate their effectiveness against our created corruptions. In particular, we leverage PointCutMix-R, PointCutMix-K [73], PointMixup [8], RSMix [26], and PGD-based adversarial training [53] as additional data augmentation strategies. We adopt the original hyper-parameter settings from their official implementations in our study. Detailed introduction can be found in Appendix C. We only enable adversarial training (AT) for PointNet, DGCNN, and PCT because others do not fit the AT framework, following [53].

Clean Performance. Table 1 shows the ER_{clean} of different model architectures with the adopted training strategies. All models achieve 90+% accuracy with standard training. As [17] indicate,

Table 2: Error Rates of Different Models on ModelNet40-C with Standard Training.

		Density Corruptions					Noise Corruptions					Transformation Corruptions				
Model (%) ↓	ER_{cor}	Occlusion	LiDAR	Density Inc.	Density Dec.	Cutout	Uniform	Gaussian	Impulse	Upsampling	Background	Rotation	Shear	FFD	RBF	Inv. RBF
PointNet	28.3	52.3	**54.9**	10.5	11.6	12.0	12.4	14.4	29.1	14.0	93.6	36.8	25.4	21.3	18.6	17.8
PointNet++	23.6	54.7	66.5	16.0	**10.0**	**10.7**	20.4	16.4	35.1	17.2	18.6	27.6	13.4	15.2	16.4	15.4
DGCNN	25.9	59.2	81.0	14.1	17.3	15.4	14.6	16.6	24.9	19.1	53.1	19.1	12.1	13.1	14.5	14.0
RSCNN	26.2	**51.8**	68.4	16.8	13.2	13.8	24.6	18.3	46.2	20.1	**18.3**	29.2	17.0	18.1	19.2	18.6
PCT	25.5	56.6	76.7	11.8	14.3	14.5	12.1	13.9	39.1	17.4	57.9	18.1	11.5	12.4	13.0	12.6
SimpleView	27.2	55.5	82.2	13.7	17.2	20.1	14.5	14.2	24.6	17.7	46.8	30.7	18.5	17.0	17.9	17.2
CurveNet	22.7	55.1	66.0	10.5	15.3	13.9	11.7	13.2	**23.7**	11.8	61.0	**15.8**	**9.8**	**10.7**	**11.4**	**10.6**
GDANet	25.6	60.5	72.1	11.0	14.5	13.8	16.5	19.1	28.9	18.8	52.6	17.4	11.5	12.0	13.1	12.7
PointMLP	31.9	64.3	95.2	12.1	14.6	14.4	25.7	35.9	49.3	42.5	56.9	19.7	11.5	11.1	12.8	11.9
PointMLP-Elite	33.4	64.8	93.3	14.0	18.2	18.7	21.7	31.3	46.8	36.2	81.1	19.9	13.2	12.9	14.4	13.8
RobustNet (Ours)	**22.2**	**51.8**	55.4	**9.7**	10.5	11.1	**10.4**	**11.1**	29.3	**10.3**	70.5	16.8	11.8	11.5	12.4	11.5

auxiliary factors will obscure the effect of architectures, and the performance gaps among different models are not significant on the ModelNet40 validation set. Data augmentation strategies like PointMixup claim to enhance the general model performance [73, 8]. However, with these factors controlled, we also do not find tangible improvements over these augmentation recipes. Besides, adversarially trained models are expected to perform slightly worse on the clean dataset compared to others [53]. It also initiates that model performances on ModelNet40 tend to saturate. Thus, it is a necessary to evaluate the model effectiveness from other perspectives (*e.g.*, robustness).

5.1 Comparison Among Corruption Types

Validity. We first benchmark robustness of 9 standard trained models to verify the validity of our ModelNet40-C. The detailed results are presented in Fig. 8 and 9 in Appendix C. We find that the $ER_{s,c}$ gradually increases as the severity level increasing for each corruption, which justifies our hyper-parameter setting. As shown in Fig. 1, there is a $3\times$ performance degradation between the overall ER_{cor} and ER_{clean} for all benchmarked models. Table 2 presents the detailed ER_c of the 10 models (including a variant of PointMLP) evaluated on ModelNet40-C with standard training. As mentioned before, there is a significant increase in ER_c on each corruption compared to ER_{clean}. The gap ranges from 17.6% to 89.7% among different corruptions. From the perspective of the corruption, we obtain several interesting observations.

• ***Insight 1.1.*** `Occlusion` and `LiDAR` pose a major threat for 3D point cloud recognition.

From Table 2, $ER_{Occlusion}$ and ER_{LiDAR} reach 57.5% and 75.6% on average, respectively. Occlusion happens in most real-world application of 3D point clouds. Moreover, we find the models have poor performance regardless of the occlusion directions, suggesting a general vulnerability.

• ***Insight 1.2.*** `Rotation` is still challenging for 3D point cloud recognition even with small angles.

Rotation is a well-known threat for point cloud recognition by several recent studies [75, 47]. Existing studies allow a rotation angle (*e.g.*, $\geq 45°$). However, such rotated point clouds confuse human perception without RGB information. In our study, we find that a small rotation ($\leq 15°$) still causes a high ER on point cloud recognition models ranging from 15.8% to 36.8%.

• ***Insight 1.3.*** `Impulse` and `Background` are surprisingly troublesome to point cloud recognition.

We find $ER_{Impulse}$ (34.8%) and $ER_{Background}$ (54.0%) are abnormally high for most architectures. Although they are even less perceptible than Gaussian and uniform noise since only a small portion of points are affected. However, the magnitudes of `Impulse` and `Background` noises are high, suggesting that a small portion of outliers will greatly affect point cloud recognition performance.

5.2 Comparison Among Model Architectures

As presented in Table 2, there is no overarching model that dominates ModelNet40-C, unlike robustness benchmarking in 2D vision [21]. Point cloud recognition models have various designs and no consensus has been reached as deep learning in the 3D space is a relatively nascent field. The model performances on ModelNet40-C are found to be in good alignment with their design attributes. It is worth noting that we train all models with 3 random seeds and the standard deviation of the test results are always less than 0.3%.

• ***Insight 2.1.*** PointNet achieves strong performance on density corruptions, but fails on others.

PointNet does not encode local feature, and several publications have studies it from the perspectives of adversarial robustness [53] and representation quality [47]. Such a design has been regarded as a main drawback of PointNet. However, we find it robust against the variations in density. Table 2 presents that PointNet achieves an ER of 28.3% on density corruptions, and overall outperforms the runner-up by 11.7%. Such results can be attributed to the locality of density corruptions. Compared

Table 3: Error Rates of Different Models on ModelNet40-C with Different Data Augmentation Strategies.

	Standard	PointCutMix-R				PointCutMix-K				PointMixup				RSMix				AT			
Model (%) ↓	ER_{cor}	ER_{cor}	Density	Noise	Trans.	ER_{cor}	Density	Noise	Trans.	ER_{cor}	Density	Noise	Trans.	ER_{cor}	Density	Noise	Trans.	ER_{cor}	Density	Noise	Trans.
PointNet	28.3	21.8	30.5	18.0	16.9	21.3	26.8	21.8	15.4	25.4	**28.3**	28.9	19.0	22.5	**24.8**	27.3	15.5	25.9	**28.8**	28.4	20.5
PointNet++	23.6	19.1	28.1	12.2	17.0	20.2	26.3	16.9	17.3	19.3	30.8	14.3	12.9	23.3	27.0	19.3	23.7	-	-	-	-
DGCNN	25.9	17.3	28.9	11.4	11.5	17.3	29.1	11.9	**10.9**	20.4	32.1	16.8	12.3	18.1	28.8	13.0	**12.6**	20.7	36.8	**13.8**	11.5
RSCNN	26.2	17.9	**25.0**	13.0	15.8	21.6	28.3	19.0	17.6	19.8	29.7	15.5	14.1	21.2	26.8	17.4	19.3	-	-	-	-
PCT	25.5	16.3	27.1	10.5	**11.2**	16.5	25.8	12.6	11.1	19.5	30.3	16.7	11.5	**17.3**	25.0	12.0	15.0	**18.4**	29.3	14.7	**11.1**
SimpleView	27.2	19.7	31.2	11.3	16.5	20.6	29.1	15.6	17.0	21.5	32.7	17.1	14.8	20.4	28.4	14.6	18.3	-	-	-	-
CurveNet	22.7	16.9	26.4	11.0	13.4	17.9	**24.3**	15.4	13.9	19.9	31.2	16.7	11.6	19.9	26.8	16.4	16.5	-	-	-	-
GDANet	25.6	17.5	28.7	10.4	13.5	18.6	28.4	14.4	13.0	19.5	30.5	15.4	12.5	19.7	28.9	14.0	16.1	-	-	-	-
PointMLP	31.9	19.2	30.6	14.3	12.7	20.9	29.1	20.0	13.7	20.0	31.3	18.6	**10.3**	22.2	30.1	20.1	16.3	-	-	-	-
RobustNet(Ours)	**22.2**	**15.3**	25.3	**9.1**	11.4	**16.0**	25.3	**11.8**	**10.9**	**19.0**	30.5	**14.0**	12.6	**17.3**	25.1	**11.9**	15.0	-	-	-	-
Average	25.9	18.1	28.2	12.1	14.0	19.1	26.8	15.9	14.4	20.5	30.8	17.5	13.2	20.5	27.2	16.6	17.0	-	-	-	-

to other models that embeds complex local features, PointNet is less sensitive to local changes of the input point cloud. On the other hand, PointNet indeed fails on other corruptions, rendering itself the worst performant model on average. Our analysis complements the existing understanding of PointNet, we believe the usage of PointNet should be determined by the application scenarios.

• ***Insight 2.2.*** Ball query-based clustering operation is robust against `Background` noise.

As mentioned in § 5.1, `Background` is a challenging corruption to point cloud recognition. However, we find that PointNet++ and RSCNN are specially robust against it, which have outperform other models by 70.7%. We discover that the ball query of neighboring points is the key to such robustness. Compared to kNN that has deterministic k points to cluster, ball query fixes the radius to reject faraway points in the bounding box space. This design helps models tackle the root cause of the `Background` corruption.

• ***Insight 2.3.*** Curve-based clustering operations are robust against transformation corruptions.

Transformer [59] has recently reformed the 2D vision [14]. PCT leverages multiple Transformer blocks as its backbone, which leverage self-attention modules to embed robust global features. PCT reaches the ER of 13.5% on transformation corruptions. In comparison to density and noise corruptions, transformation corruptions are mild and have a minor effect on the local smoothness. Transformer has been demonstrated to have large capacity and a global receptive field, and we believe this design contribute to its resilience to global corruption of point clouds. Curve-based clustering is also effective in reducing the ER for transformation corruptions, as it better learns the geometry of a point cloud rather than blindly grouping nearby points as a cluster. CurveNet thus achieves the best performance with an ER=11.7%.

Besides above, we find that SimpleView cannot achieve better robustness under common corruptions than other architectures, despite it high performance on clean data (Table 2), suggesting point cloud-specific designs are indeed desired. The *worst* performance of PointMLP (ICLR'22) further alerts the community that it is critical to evaluate model quality from multiple perspectives.

5.3 Data Augmentation Strategies

As mentioned earlier, we use five additional data augmentation strategies to train 9 models. In this section, we examine how these training recipes combined with different models perform on ModelNet40-C, and Table 3 presents the overall results. Due to the space limit, we group the evaluation results per corruption patterns and the detailed results are shown in Fig. 8 and 9 in Appendix C. Several interesting insights can be concluded from our experiments.

• ***Insight 3.1*** Data augmentation strategies generally improve the corruption robustness.

As Table 3 indicates, all the augmentation recipes enhance the overall corruption robustness from 19.6% to 28.4%. Data augmentation methods enrich the training set the resulting model more general. Combined with results in Table 1, our paper suggests different conclusions from the original claims: mixing-based augmentations have little gain on clean performance but help generalize to common corruptions.

• ***Insight 3.2.*** No single data augmentation can rule them all. Different augmentation methods have expertise on distinct corruption patterns.

As Table 3 presents, PointCutMix-R performs the best on noise corruptions (ER $= 12.4\%$), PointMixup specializes the transformation corruptions (ER $= 13.2\%$), and RSMix is especially robust against density corruptions (ER $= 27.4\%$). Such results also relate to the design of augmentation strategies. In details, given two point cloud samples $\boldsymbol{x}_a$,$\boldsymbol{x}_b$ from class a and b, PointCutMix-R simply merges ($\oplus$) two randomly selected ($\odot$) subsets together based on hyper-parameter λ

($\boldsymbol{x}_{aug} = \lambda \odot \boldsymbol{x}_a \oplus (1-\lambda) \odot \boldsymbol{x}_b$). The two subsets will overlap in the resulting point cloud $\boldsymbol{x}_{aug}$. Each point cloud subset can be regarded as a special noise by the other. Thus, it naturally includes noise corruptions with mixing into data augmentations. PointMixup leverages interpolation-based mixing that the transition between two point clouds ($\boldsymbol{x}_{aug} = \lambda \boldsymbol{x}_a + (1-\lambda)\zeta(\boldsymbol{x}_a, \boldsymbol{x}_b)$, where $\zeta(\boldsymbol{x}_a, \boldsymbol{x}_b)$ finds the shortest path for every pair in $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$). The augmented point cloud is thus locally smooth, which aligns with the transformation corruptions. In contrast, RSMix acts similarly with PointCutMix-K but guarantee a *rigid* mixing of two partial point clouds. There will be no overlaps and each point cloud subset is clustered and isolated in the 3D space. Such patterns correspond to density corruptions in point cloud data.

• ***Insight 3.3.*** AT does not show superiority on corruption robustness for 3D point cloud recognition.

Adversarial training improves robustness on noise corruptions since we rely on point shifting attacks in the inner maximization stage. [47] suggest that adversarial rotation training improves the robustness against random rotations. We here motivate future research to present general methods that improve both adversarial and corruption robustness for point cloud learning.

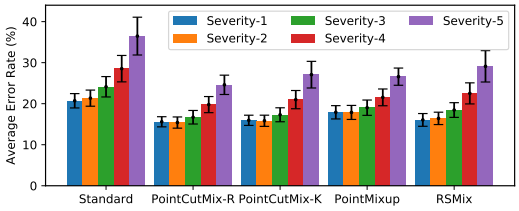

Figure 3: Average Error Rates over 9 Models and 13 Corruptions (except for `Occlusion` and `LiDAR`). We exclude `Occlusion` and `LiDAR` since they do not have different severity levels.

Moreover, we find **PCT** outperforms CurveNet with sophisticated augmentation-based training recipes. Such results align with recent studies in corruption robustness of 2D vision tasks as well [2], suggesting the superiority of Transformer-based design in 3D point cloud learning. Surprisingly, the simplest augmentation, PointCutMix-R, achieves the best overall robustness ($\text{ER}_{\text{corrup}}$=18.4%). As Fig. 3 shows, it is especially helpful on corruptions with high severity levels. We hope our analysis will facilitate future research on designing effective and robust training strategies.

5.4 Test-Time Adaptation Methods

Besides the introduced training-time strategies, we evaluate test-time adaptation methods on our ModelNet40-C. Specifically, we for the first time adapt the BN [45] and TENT [60] to point cloud recognition. BN updates the statistics of BatchNorm [23] layers (*i.e.*, μ and σ.) based on the incoming batch of testing point clouds. TENT updates both the statistics and weight parameters (*i.e.*, γ and β.) of BatchNorm layers to minimize the cross-entropy of the output layer. Table 4 presents the evaluation results.

Table 4: Error Rates of Different Models on ModelNet40-C with Test-time Adaptation Methods.

	BN				TENT			
Model (%) ↓	ER_{cor}	Density	Noise	Trans.	ER_{cor}	Density	Noise	Trans.
PointNet	25.9	26.8	29.6	21.2	27.3	27.8	31.0	23.0
PointNet++	16.9	**24.2**	12.7	13.8	16.7	**24.1**	12.3	13.6
DGCNN	21.1	31.4	19.5	12.5	20.9	30.7	19.4	12.5
RSCNN	20.0	26.4	16.7	17.0	19.4	26.0	15.9	16.4
PCT	19.5	27.9	18.2	12.4	18.0	26.9	15.4	11.7
SimpleView	19.8	29.8	13.8	15.8	16.9	28.1	**9.9**	12.8
CurveNet	18.2	25.9	18.3	**10.4**	17.4	25.8	16.8	**9.5**
GDANet	18.8	28.8	15.7	11.7	17.9	28.2	14.1	11.5
PointMLP	20.2	29.9	19.5	11.1	20.7	33.5	17.6	11.0
RobustNet(Ours)	**16.8**	**24.2**	**12.5**	13.7	**16.5**	**24.1**	12.0	13.5
Average	19.7	27.9	17.4	14.0	19.2	27.9	16.4	13.6

• ***Insight 4.1.*** Test-time adaptations overall perform worse than data augmentation strategies in improving robustness.

BN and TENT consistently help enhance the corruption robustness for point cloud recognition. However, we find they overall are not as effective as data augmentation strategies. Such observations are different from 2D vision tasks, and we attribute the reason to the nature of corruptions in 3D space. Corruption robustness benchmarks for 2D images are created by changing the RGB values. Corruptions in the 3D space directly modify both the numbers and positions of points. The distributional shift is thus large between corrupted and clean point clouds.

• ***Insight 4.2.*** Test-time adaptation is surprisingly useful for tough corruptions.

We find that TENT on average helps achieve the strongest robustness on `Occlusion` (ER=47.6%), `LiDAR` (ER=54.1%), and `Rotation` (ER=19.8%) corruptions, outperforming the best augmentation method by 6.7%, 1.9%, and 7.9% respectively. Especially, we find test-time adaptation methods achieve the best $\text{ER}_{\text{rotation},5} = 35.6\%$, which is a 27.1 % improvement over the best augmentation strategy. Augmentation strategies cannot handle these difficult corruptions, but test-time adaptation methods deliver a more consistent improvement.

We have so far demonstrated that PointCutMix-R and TENT obtain the best among the training- and test-time methods, in terms of the overall ER_{cor}. We here evaluate the performance of the combination of PointCutMix-R and TENT as they do not conflict with each other. As presented in Table 5, we find that the combined solution further improves the corruption robustness by 14.7%. To our best knowledge, there is no test-time adaptation designs specific for point cloud learning, and we hope our study will shed light on future research on corruption robustness in this area.

Table 5: Error Rates of Models on ModelNet40-C with PointCutMix-R and TENT.

Corruption (%) ↓	PointNet	PointNet++	DGCNN	RSCNN	PCT	SimpleView	CurveNet	GDANet	PointMLP	RobustNet
Density	24.8	**21.8**	24.7	22.7	22.8	23.5	22.7	24.2	29.2	22.1
Noise	15.7	9.5	9.2	10.7	9.0	9.7	8.9	9.4	10.0	**8.7**
Trans.	15.2	12.1	10.4	12.9	10.1	13.3	9.8	10.3	**9.5**	10.2
ER_{cor}	18.5	14.5	14.8	15.4	13.9	15.5	13.9	14.6	16.3	**13.7**

6 RobustNet and PointCutMixup

In this section, we introduce RobutNet architecture and PointCutMixup augmentation strategy based on insights from our benchmarking results.

RobustNet. We combine the robust designs inside PointNet++, CurveNet, and PCT modules to form our RobustNet architecture. Specifically, we first use one multi-scale grouping (MSG) block from PointNet++ as the first module in RobustNet. MSG is an essential design in PointNet++ to make it robust to noise corruptions § 5.2 due to the multi-scale ball queries. We then apply CurveNet module, CIC, to further aggregate local features in curves, which has demonstrated transformation. Four transformer blocks are finally attached in the end to improve the generality of the RobustNet backbone. Detailed hyper-parameters can be found in Appendix B.

Tables 2 and 3 present the results of RobustNet with standard training and different augmentations. We find that RobustNet achieves significantly better robustness with standard training, which outperforms the prior SOTA by 2.3%. Encouragingly, RobustNet with PointCutMix-R further improves the lowest error rate to 15.3%, which is 6.5% relative enhancement compared to the previous best result.

PointCutMixup. We find that PointCutMix-R is especially useful for noise corruptions, and PointMixup for transformation related corruptions. Our PointCutMixup involves three randomly selected point clouds: $\boldsymbol{x}_a$, $\boldsymbol{x}_b$, and $\boldsymbol{x}_c$. We first interpolate the first two point clouds $\boldsymbol{x}_{aug1} = \lambda \boldsymbol{x}_a + (1-\lambda)\zeta(\boldsymbol{x}_a, \boldsymbol{x}_b)$ using PointMixup, and then mix the the third point cloud with the first augmented one: $\boldsymbol{x}_{aug} = \eta \odot \boldsymbol{x}_{aug1} \oplus (1-\eta) \odot \boldsymbol{x}_c$. The corresponding label is $y_{aug} = \eta\lambda y_a + \eta(1-\lambda)y_b + (1-\eta)y_c$. Table 6 presents that RobustNet with PointCutMixup achieves the new SOTA ER to 14.9%.

Table 6: Error Rates of Models on ModelNet40-C with PointCutMix-R and TENT.

ER_{cor} (%) ↓	PointNet	PointNet++	DGCNN	RSCNN	PCT	SimpleView	CurveNet	GDANet	PointMLP	RobustNet
PointCutMix-R	21.8	19.1	17.3	17.9	16.3	**19.7**	16.9	17.5	19.2	15.3
PointCutMixup	**20.9**	**18.5**	**16.9**	**17.5**	**16.0**	20.0	**16.7**	**17.2**	**18.9**	**14.9**

7 Discussion and Conclusion

Through our systematic benchmarking and analysis, we found that the performance discrepancies of different point cloud recognition models across different corruptions are much larger than 2D architectures. This suggests future studies on a universal architecture design for 3D point cloud to be a worthwhile direction. In the future, we plan to extend our current benchmark to complex tasks like point cloud segmentation and object detection to facilitate research on robustness in the 3D domain. We notice a concurrent work [43] that also presents robustness dataset for point clouds. [43] has 7 corruption types without considering physical constraints, while our ModelNet40-C consists of 15 corruption types with physically realistic corruptions like `Occlusion` and `LiDAR`. We believe these two datasets will complement each other to benefit the community.

To conclude, we have presented ModelNet40-C, a comprehensive benchmark for corruption robustness of point cloud recognition models. We have unveiled the massive performance degradation on our ModelNet40-C for 9 representative models. We also provided critical insights on how different architecture and data augmentation designs affect model robustness on different corruptions. Our study on test-time adaptation in point cloud recognition shows its potential as a robustness strategy. Last but not least, we propose RobustNet and PointCutMixup that further enhance corruption robustness. We hope that our ModelNet40-C benchmark will benefit future research in developing robust 3D point cloud models.

ETHICS STATEMENT

Adversarial robustness has been extensively studies in the literature for both 2D and 3D model architectures. However, corruption robustness was rarely explored in the 3D space. Our new dataset and benchmark are thus beneficial for the 3D point cloud community to assess the corruption robustness of new models or training methods (*e.g.,* data augmentation strategies). The evaluation of existing models on corruption robustness is useful as well since we have uncovered that most SOTA models are extremely vulnerable to small corruptions (*e.g.,* PointMLP). We follow the licenses of usage for all the public models and datasets in our study.

REPRODUCIBILITY STATEMENT

To ensure the reproducibilityof our results, we have provided every datail of our dataset generation and training process in the Appendix. We also fixed the random seed in our evaluation for easier reproduction. Our codebase is attached in the supplementary materials and we also provide anoymous link for downloading our ModelNet40-C dataset.

REFERENCES

[1] H. B. Adallah, J.-J. Orteu, B. Dolives, and I. Jovančević. 3d point cloud analysis for automatic inspection of aeronautical mechanical assemblies. In *Fourteenth International Conference on Quality Control by Artificial Vision*, volume 11172, page 111720U. International Society for Optics and Photonics, 2019.

[2] Y. Bai, J. Mei, A. Yuille, and C. Xie. Are transformers more robust than cnns? In *NeurIPS*, 2021.

[3] D. Bonatto, S. Rogge, A. Schenkel, R. Ercek, and G. Lafruit. Explorations for real-time point cloud rendering of natural scenes in virtual reality. In *2016 International Conference on 3D Imaging (IC3D)*, pages 1–7. IEEE, 2016.

[4] S. Bulusu, B. Kailkhura, B. Li, P. K. Varshney, and D. Song. Anomalous example detection in deep learning: A survey. *IEEE Access*, 8:132330–132347, 2020.

[5] D. A. Calian, F. Stimberg, O. Wiles, S.-A. Rebuffi, A. Gyorgy, T. Mann, and S. Gowal. Defending against image corruptions through adversarial augmentations. *arXiv preprint arXiv:2104.01086*, 2021.

[6] C. Cao, M. Preda, and T. Zaharia. 3d point cloud compression: A survey. In *The 24th International Conference on 3D Web Technology*, pages 1–9, 2019.

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[8] Y. Chen, V. T. Hu, E. Gavves, T. Mensink, P. Mettes, P. Yang, and C. G. Snoek. Pointmixup: Augmentation for point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 330–345. Springer, 2020.

[9] Q. Cheng, P. Sun, C. Yang, Y. Yang, and P. X. Liu. A morphing-based 3d point cloud reconstruction framework for medical image processing. *Computer methods and programs in biomedicine*, 193:105495, 2020.

[10] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[11] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[12] T. DeVries and G. W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

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

[14] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[15] D. Forti and G. Rozza. Efficient geometrical parametrisation techniques of interfaces for reduced-order modelling: application to fluid–structure interaction coupling problems. *International Journal of Computational Fluid Dynamics*, 28(3-4):158–169, 2014.

[16] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[17] A. Goyal, H. Law, B. Liu, A. Newell, and J. Deng. Revisiting point cloud shape classification with a simple and effective baseline. *arXiv preprint arXiv:2106.05304*, 2021.

[18] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021.

[19] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] D. Hendrycks, S. Basart, N. Mu, S. Kadavath, F. Wang, E. Dorundo, R. Desai, T. Zhu, S. Parajuli, M. Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021.

[21] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[22] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019.

[23] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014.

[25] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4558–4567, 2018.

[26] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15900–15909, 2021.

[27] C.-L. Li, M. Zaheer, Y. Zhang, B. Poczos, and R. Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018.

[28] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.

[29] D. Liu, R. Yu, and H. Su. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 2279–2283. IEEE, 2019.

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

[31] J. Liu, Q. Sun, Z. Fan, and Y. Jia. Tof lidar development in autonomous vehicle. In *2018 IEEE 3rd Optoelectronics Global Conference (OGC)*, pages 185–190. IEEE, 2018.

[32] Y. Liu, B. Fan, S. Xiang, and C. Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.

[33] 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. In *International Conference on Learning Representations*, 2022.

[34] P. M. Maloca, J. E. R. de Carvalho, T. Heeren, P. W. Hasler, F. Mushtaq, M. Mon-Williams, H. P. Scholl, K. Balaskas, C. Egan, A. Tufail, et al. High-performance virtual reality volume rendering of original optical coherence tomography point-cloud data enhanced with real-time ray casting. *Translational vision science & technology*, 7(4):2–2, 2018.

[35] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[36] C. Michaelis, B. Mitzkus, R. Geirhos, E. Rusak, O. Bringmann, A. S. Ecker, M. Bethge, and W. Brendel. Benchmarking robustness in object detection: Autonomous driving when winter is coming. *arXiv preprint arXiv:1907.07484*, 2019.

[37] E. Mintun, A. Kirillov, and S. Xie. On interaction between augmentations and corruptions in natural corruption robustness. *arXiv preprint arXiv:2102.11273*, 2021.

[38] A. Nguyen and B. Le. 3d point cloud segmentation: A survey. In *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*, pages 225–230. IEEE, 2013.

[39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[40] F. Pomerleau, F. Colas, and R. Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015.

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

[42] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.

[43] J. Ren, L. Pan, and Z. Liu. Benchmarking and analyzing point cloud classification under corruptions. *arXiv preprint arXiv:2202.03377*, 2022.

[44] A. E. Roth. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[45] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge. Improving robustness against common corruptions by covariate shift adaptation. *arXiv preprint arXiv:2006.16971*, 2020.

[46] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, 1986.

[47] W. Shen, Q. Ren, D. Liu, and Q. Zhang. Interpreting representation quality of dnns for 3d point cloud processing. *Advances in Neural Information Processing Systems*, 34, 2021.

[48] W. Shen, Z. Wei, S. Huang, B. Zhang, P. Chen, P. Zhao, and Q. Zhang. Verifiability and predictability: Interpreting utilities of network architectures for point cloud processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10703–10712, June 2021.

[49] Y. Shen, C. Feng, Y. Yang, and D. Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4548–4557, 2018.

[50] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[51] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[52] J. Sun, Y. Cao, Q. A. Chen, and Z. M. Mao. Towards robust lidar-based perception in autonomous driving: General black-box adversarial sensor attack and countermeasures. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 877–894, 2020.

[53] J. Sun, Y. Cao, C. B. Choy, Z. Yu, A. Anandkumar, Z. M. Mao, and C. Xiao. Adversarially robust 3d point cloud recognition using self-supervisions. *Advances in Neural Information Processing Systems*, 34, 2021.

[54] J. Sun, K. Koenig, Y. Cao, Q. A. Chen, and Z. M. Mao. On adversarial robustness of 3d point cloud classification under adaptive attacks. *arXiv preprint arXiv:2011.11922*, 2020.

[55] J. Sun, A. Mehra, B. Kailkhura, P.-Y. Chen, D. Hendrycks, J. Hamm, and Z. M. Mao. Certified adversarial defenses meet out-of-distribution corruptions: Benchmarking robustness and simple baselines. *arXiv preprint arXiv:2112.00659*, 2021.

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

[57] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019.

[58] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1588–1597, 2019.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[60] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

[61] D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, volume 1, pages 10–15607. Rome, Italy, 2015.

[62] Y. Wang and J. M. Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3523–3532, 2019.

[63] 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 (tog)*, 38(5):1–12, 2019.

[64] K. Wolff, C. Kim, H. Zimmer, C. Schroers, M. Botsch, O. Sorkine-Hornung, and A. Sorkine-Hornung. Point cloud noise and outlier removal for image-based 3d reconstruction. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 118–127. IEEE, 2016.

[65] W. Wu, Z. Qi, and L. Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

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

[67] C. Xiang, C. R. Qi, and B. Li. Generating 3d adversarial point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9136–9144, 2019.

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

[69] 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. *arXiv preprint arXiv:2012.10921*, 2, 2021.

[70] W. Yang, Z. Gong, B. Huang, and X. Hong. Lidar with velocity: Motion distortion correction of point clouds from oscillating scanning lidars. *arXiv preprint arXiv:2111.09497*, 2021.

[71] T. Yin, X. Zhou, and P. Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11784–11793, 2021.

[72] J. Yu, C. Zhang, H. Wang, D. Zhang, Y. Song, T. Xiang, D. Liu, and W. Cai. 3d medical point transformer: Introducing convolution to attention networks for medical point cloud analysis. *arXiv preprint arXiv:2112.04863*, 2021.

[73] J. Zhang, L. Chen, B. Ouyang, B. Liu, J. Zhu, Y. Chen, Y. Meng, and D. Wu. Pointcutmix: Regularization strategy for point cloud classification. *arXiv preprint arXiv:2101.01461*, 2021.

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

[75] Y. Zhao, Y. Wu, C. Chen, and A. Lim. On isometry robustness of deep 3d point cloud models under adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1201–1210, 2020.

[76] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1961–1970, 2019.

[77] L. Zhou, Y. Du, and J. Wu. 3d shape generation and completion through point-voxel diffusion. *arXiv preprint arXiv:2104.03670*, 2021.

[78] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

A MODELNET40-C

We elaborate the creation of ModelNet40-C in this section. The detailed implementation can be found in our codebase. We fixed the random seed (*i.e.*, 666) in our implementation to create ModelNet40-C. We also tested two other random seeds (*i.e.*, 777 and 888), and found little variance in the benchmarking results.

`Occlusion` and `LiDAR` share similar general corruption features. We leverage five viewing angles to construct these two corruptions on ModelNet40. Specifically, we utilize ray tracing algorithms on the original meshed from ModelNet40 to generate the point cloud. Let the facing direction of the object as 0° pivoting the z axis, we use 0°, 72°, 144°, 216°, and 288° as our viewing angles, the viewing angles between the xy plane are randomly sampled from is $30^\circ - 60^\circ$. For `LiDAR`, we additionally render the generated point cloud into the vertically multi-line style to simulate the pattern of the LiDAR sensor.

For `Local_Density_Inc` and `Local_Density_Dec`, we first sample a number of anchor points based the severity level. We further find the kNN of the anchor points and up-sample or down-sample them to increase and decrease their local density, respectively. Similarly, `Cutout` discards the full kNN ($k = 50$) subsets of the anchor points to simulate the sensor limitations of LiDAR and other scanning devices.

`Gaussian` and `Uniform` noises are sampled from Gaussian and uniform distributions with different σ and ϵ based on the severity level. For the `Background` noise, we randomly sample different numbers of points in the edge-length-2 cube that bounds the point cloud based on the severity level. For `Impulse` noise, we first sample different numbers of points based on the severity level and assign the maximum magnitude of perturbation $\ell_\infty = 0.05$ to them. For the `Upsampling` noise, we first choose different numbers of points based on the severity level and generate new points around the selected anchors, bounded by $\ell_\infty = 0.05$.

For `Rotation` and `Shear`, we have introduced their construction in § 3. As mentioned, we allow relatively small transformations since we find larger ones will affect the human perception of the object class as well.

For deformation-based corruptions `FFD`, `RBF`, and `Inv_RBF`, we assign 5 control points along each xyz axis, resulting in 125 control points in total. We choose the deformation distance based on the severity level and randomly assign their directions in the 3D space. The deformations then are formulated based on the interpolation functions that we choose in § 3.

We visualize two additional groups of sample point clouds from ModelNet40-C in Fig. 4, Fig. 5 and 6.

Comparison with RobustPointSet [56]. We would like to emphasize *significant* differences between ModelNet40-C and RobustPointSet. First, the corruptions are more realistic in ModelNet40-C. For example, `Occlusion` in RobustPointSet does not consider physical law (*e.g.*, self-occlusion), while ModelNet40-C addressed it. Second, ModelNet40-C is a more diverse and fine-grained dataset containing 15 corruption types with 5 severity levels. Third, ModelNet40-C has provided a taxonomy of common corruptions, which will be beneficial for the community to understand the vulnerabilities of evaluated models and methods. Moreover, we have delivered more comprehensive and insightful benchmarking results.

B ROBUSTNET AND POINTCUTMIXUP

We detail the hyper-parameters of RobustNet and PointCutMixup in this section. For RobustNet, the first MSG block has three scales of ball clustering with number of anchor points = 512 and radii = [0.1, 0.2, 0.4]. RobustNet further stacks the features from multiple scales to form a [512,320] shaped feature tensor. RobustNet next uses two CIC modules from CurveNet to learn a feature map with a size of [256,256]. Four Transformer blocks are lastly attached to RobustNet. Other implementation parameters can be found in our codebase. For PointCutMixup, we use $\lambda = 0.5$ and $\eta = 0.5$.

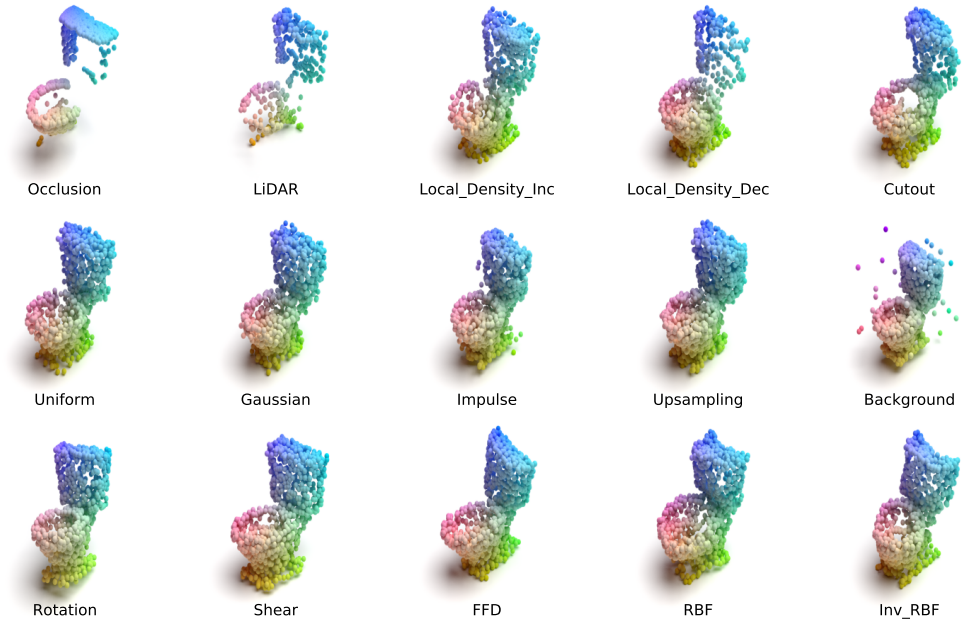

Figure 4: Visualization of Samples from ModelNet40-C - "Toliet" Class.

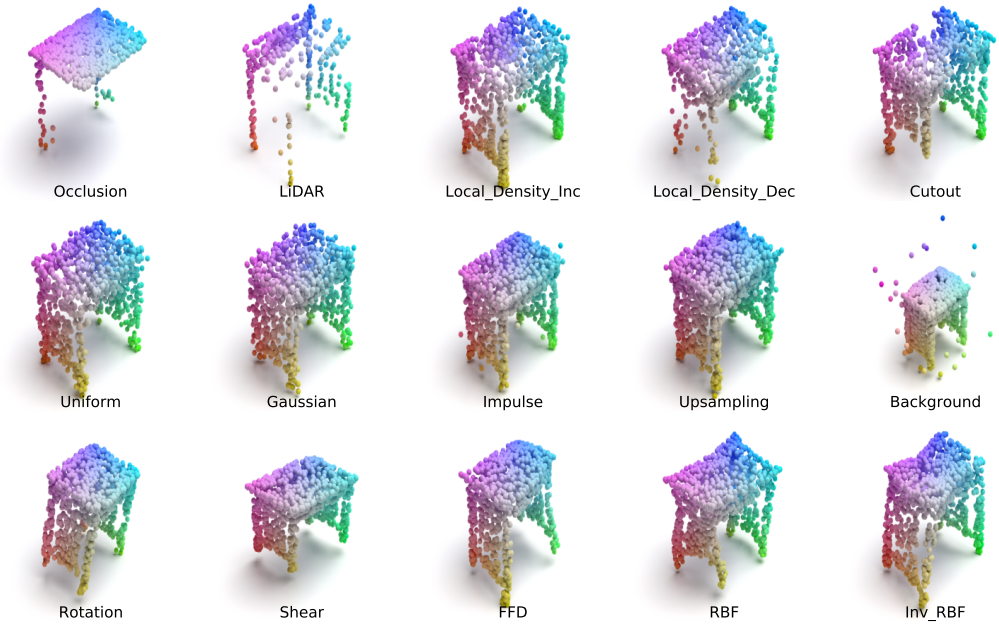

Figure 5: Visualization of Samples from ModelNet40-C - "Desk" Class.

C Experiments

C.1 Data Augmentation Setups

We introduce the detailed setting of our experiments and analysis in this section. For all mixing-based data augmentation strategies, we have a hyper-parameter λ to determine the weight of two samples to mix, as well as the weight of the virtual label vector:

$$y_{aug} = \lambda \cdot y_a + (1 - \lambda) \cdot y_b \tag{1}$$

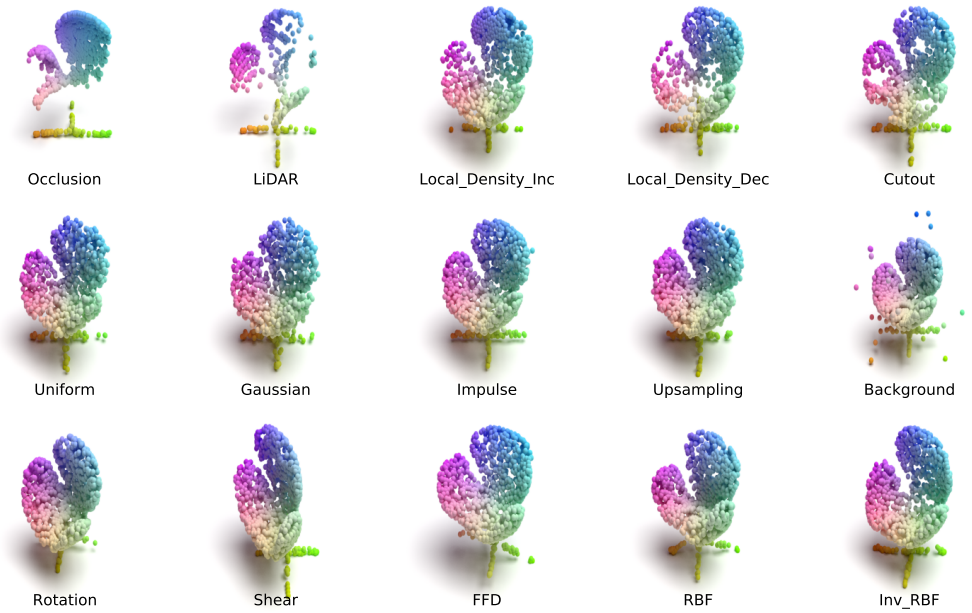

Figure 6: Visualization of Samples from ModelNet40-C - "Chair" Class.

where y_a and y_b are the label vectors for point cloud $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$. We set $\lambda = 0.5$, which has shown to achieve good results reported in [73, 8, 26]. For adversarial training, we use the point shifting attack in the adversarial inner maximization:

$$\boldsymbol{x}^{s+1} = \Pi_{\boldsymbol{x}+\mathcal{S}}(\boldsymbol{x}^s + \alpha \cdot \text{sign}(\nabla_{\boldsymbol{x}^s}\mathcal{L}(\boldsymbol{x}^s, y; f))); \quad \boldsymbol{x}^0 = \boldsymbol{x} + \text{U}(-\epsilon, \epsilon) \tag{2}$$

where $\epsilon = 0.05$, $\alpha = 0.01$ and we use seven steps PGD, as suggested by [53].

C.2 Evaluation Results

We utilized the same random seed (*i.e.*, 1) to run all the experiments and obtain the results. We agree with the reviewer that multiple tests are necessary to assess the performance. Thus, we use two other random seeds (*i.e.*, 2 and 3) to re-train all the adopted architectures and evaluate their robustness. As shown in Table 7, the standard deviations are small among all models and align well with our findings. Due to time and resource constraints, we did not re-test the augmentation methods. However, we believe the results should remain consistent since the performance gaps between augmentation strategies are already clear.

Table 7: Error Rates of Different Models on ModelNet40-C.

ER (%) ↓	PointNet	PointNet++	DGCNN	RSCNN	PCT	SimpleView	GDANet	CurveNet	PointMLP
Clean	9.4 ± 0.1	7.0 ± 0.2	7.3 ± 0.1	7.7 ± 0.2	7.1 ± 0.1	**6.0** ± 0.3	7.5 ± 0.2	6.6 ± 0.1	6.2 ± 0.2
Density	**28.2** ± 0.2	31.7 ± 0.3	37.4 ± 0.3	32.8 ± 0.4	34.8 ± 0.3	37.7 ± 0.3	34.4 ± 0.4	32.2 ± 0.2	40.1 ± 0.3
Noise	32.7 ± 0.3	**21.4** ± 0.2	25.7 ± 0.2	25.5 ± 0.2	28.1 ± 0.2	23.6 ± 0.2	27.2 ± 0.2	24.3 ± 0.3	42.1 ± 0.4
Trans.	20.2 ± 0.2	17.5 ± 0.1	14.6 ± 0.1	20.4 ± 0.2	13.5 ± 0.1	20.3 ± 0.2	13.3 ± 0.2	**11.7** ± 0.1	13.4 ± 0.2

We illustrate the confusion matrices of six representative models with standard training in Fig. 7 to show the validity of ModelNet40-C, where each cell (i, j) represents the proportion of groundtruth label j with prediction as label i. The values in the diagonal are still high, further validating the semantic maintenance of ModelNet40-C. We present our detailed evaluation results containing 5,700 data points. Fig. 8 and 9 shows the model comparison on all data augmentation strategies. The class-wise mean error rates (mER) are shown in Figure 10 and 11. The detailed results of test-time adaptation methods (ER and mER) are shown in Fig. 12, 13, 14, and 15.

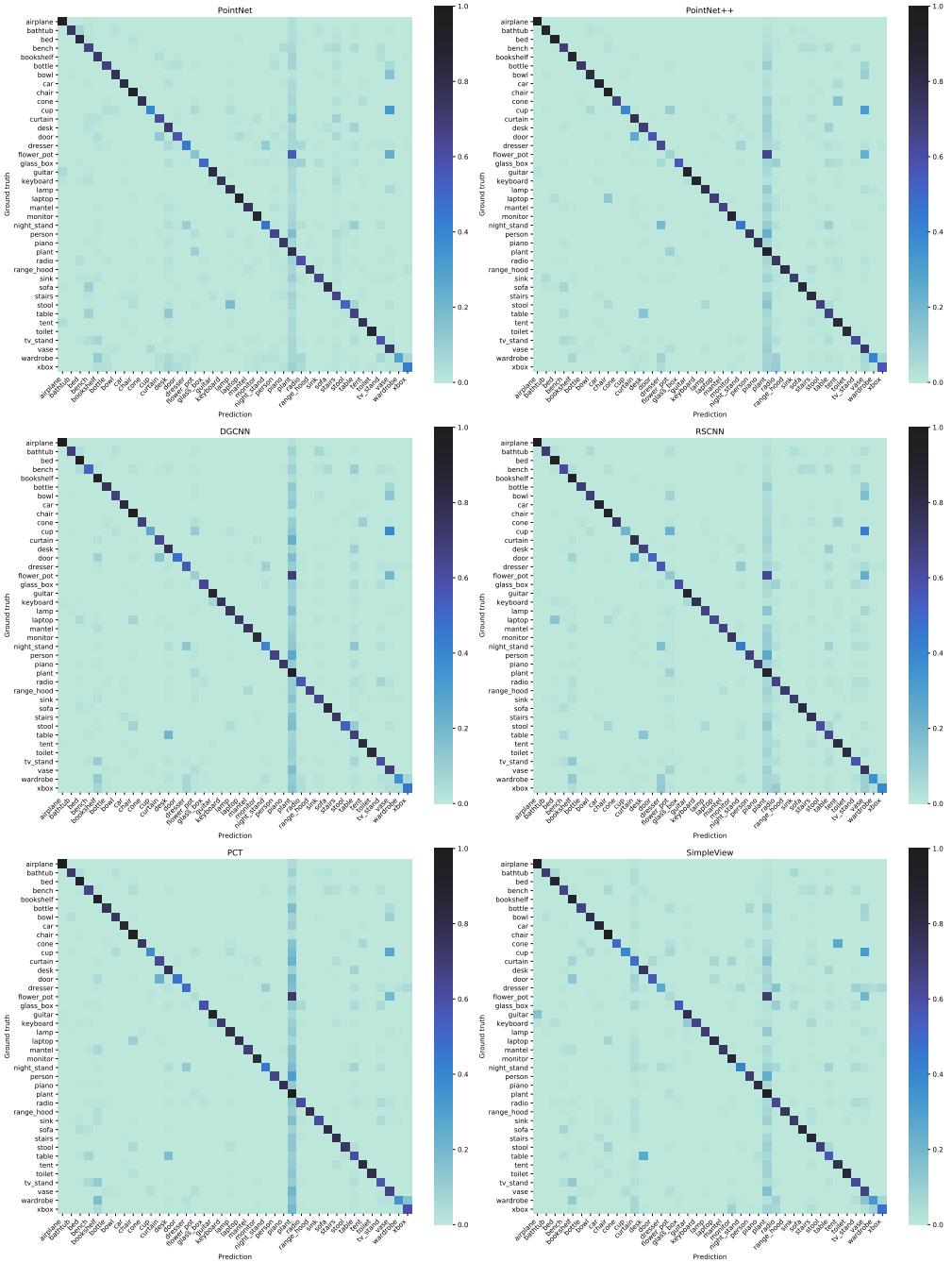

Figure 7: Confusion Matrices for Six Representative Models on ModelNet40-C with Standard Training.

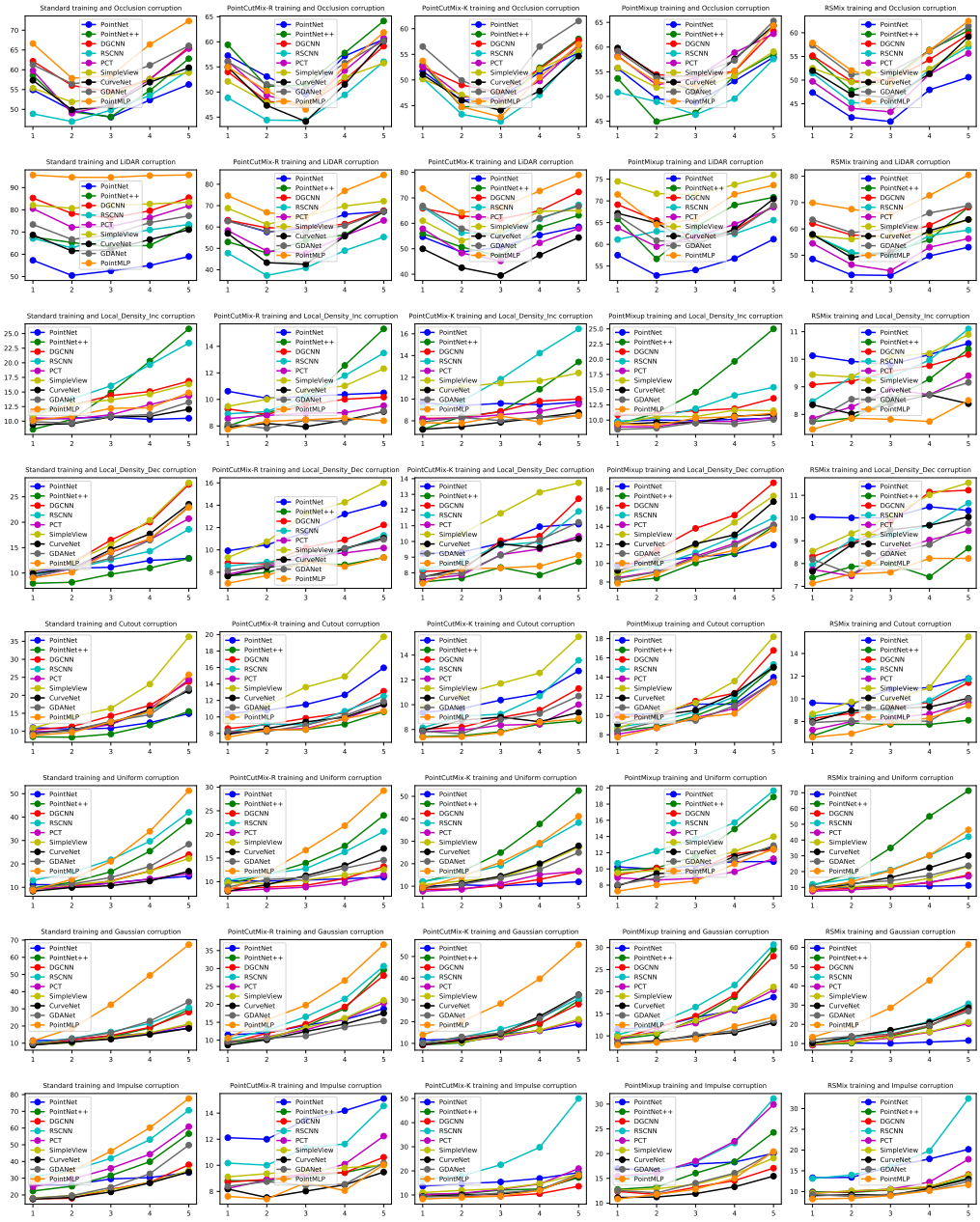

Figure 8: Error Rates of Different Models with Different Data Augmentation Strategies on ModelNet40-C.

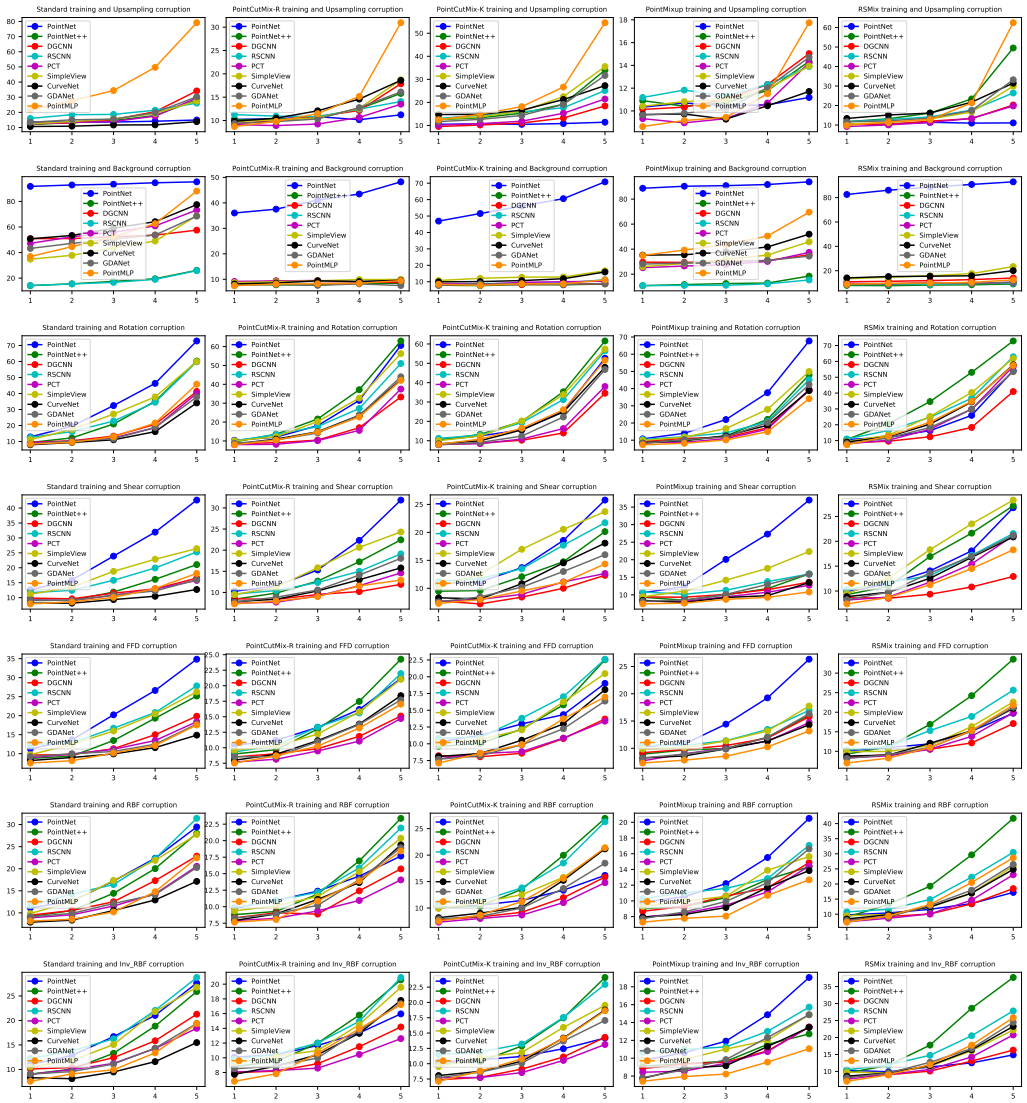

Figure 9: Error Rates of Different Models with Different Data Augmentation Strategies on ModelNet40-C (Cont'd).

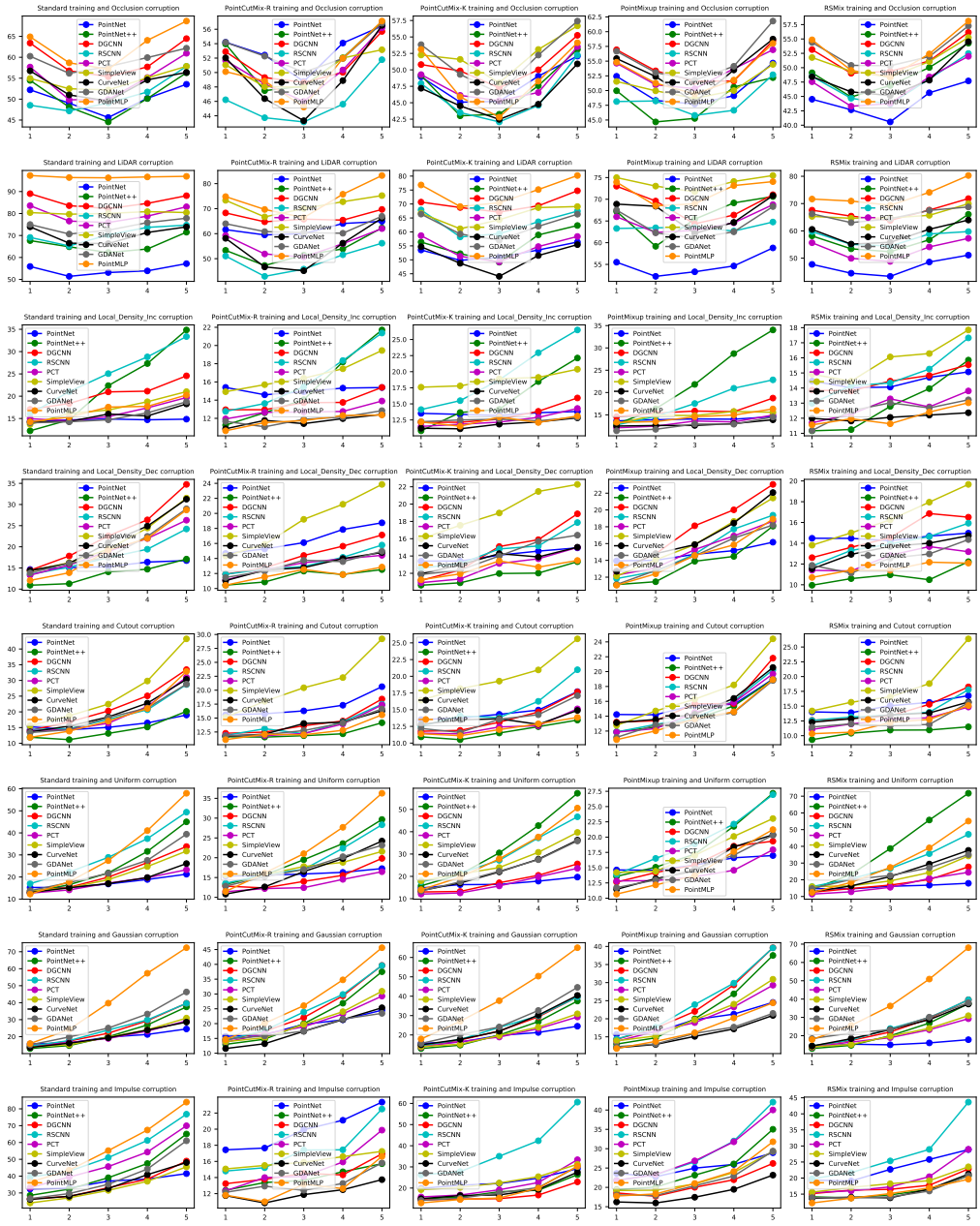

Figure 10: Class-wise Mean Error Rates of Different Models with Different Data Augmentation Strategies on ModelNet40-C.

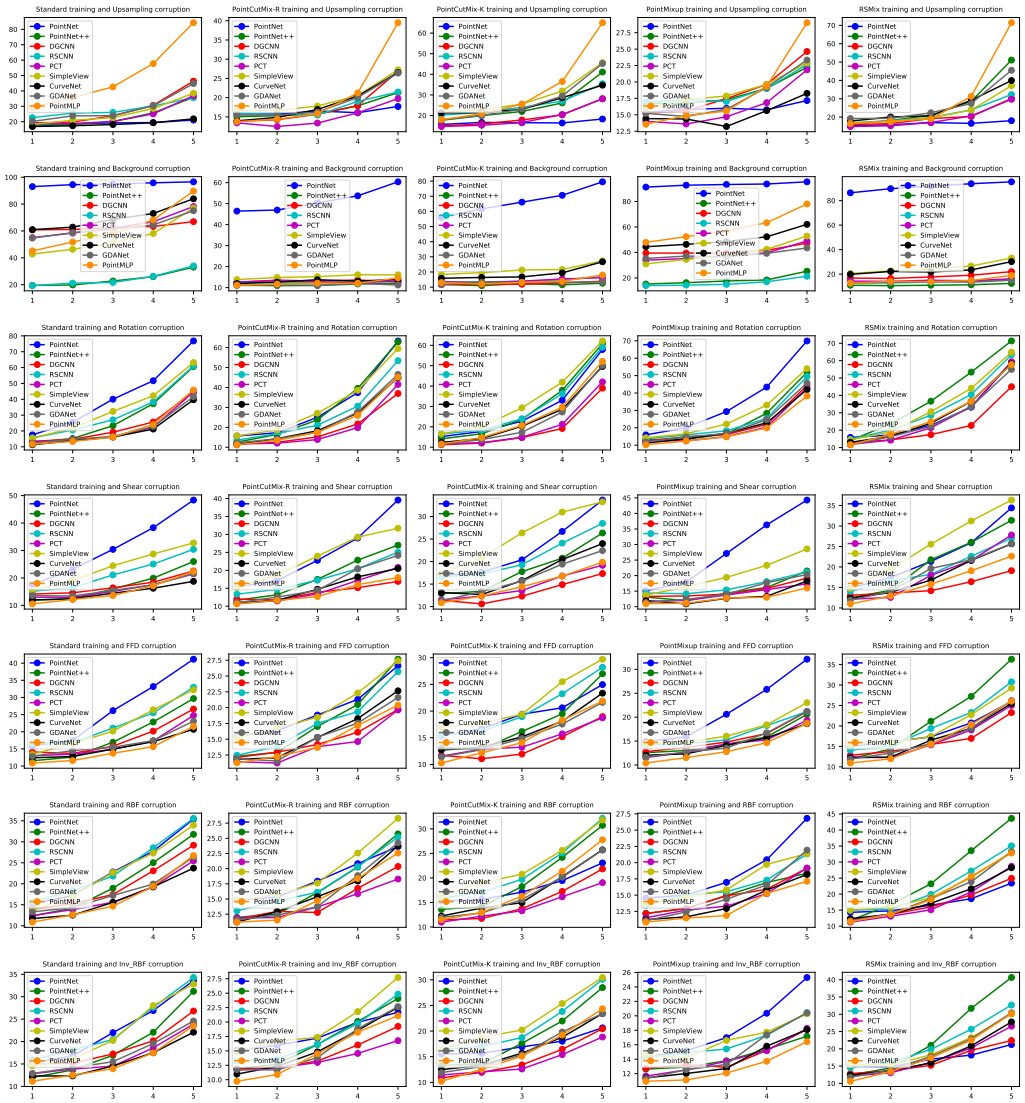

Figure 11: Class-wise Mean Error Rates of Different Models with Different Data Augmentation Strategies on ModelNet40-C (Cont'd).

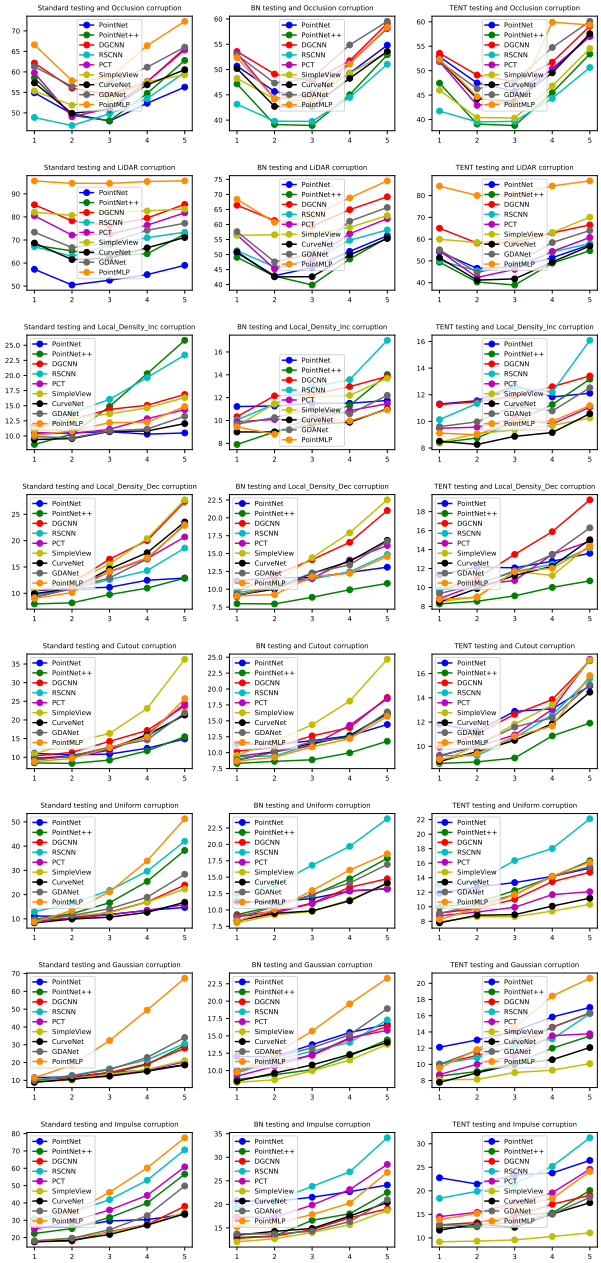

Figure 12: Error Rates of Different Models with Different Test-time Adaptation Methods on ModelNet40-C.

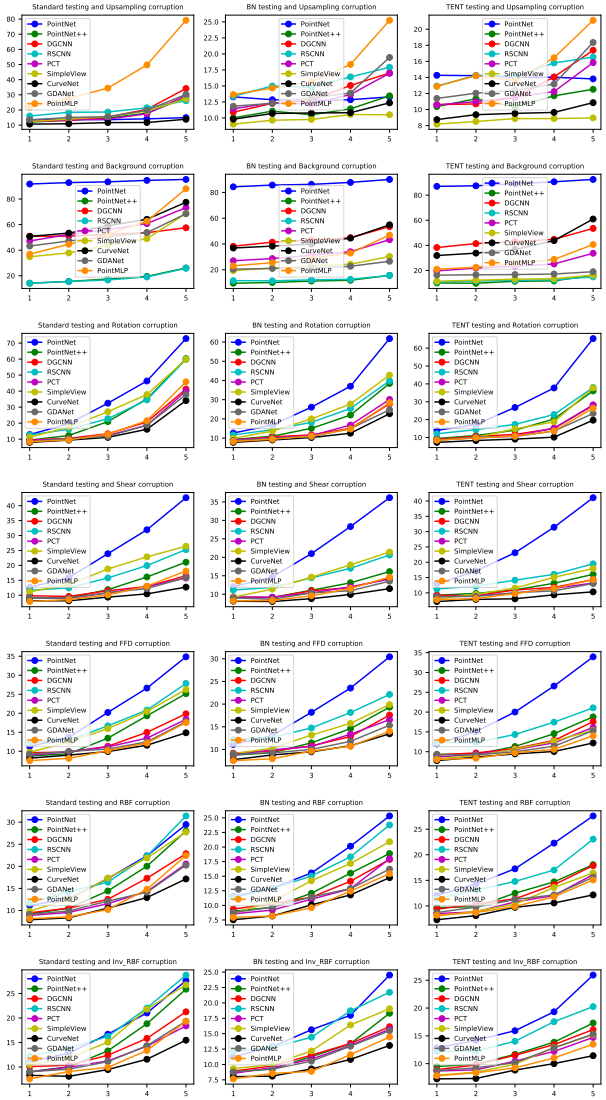

Figure 13: Error Rates of Different Models with Different Test-time Adaptation Methods on ModelNet40-C (Cont'd).

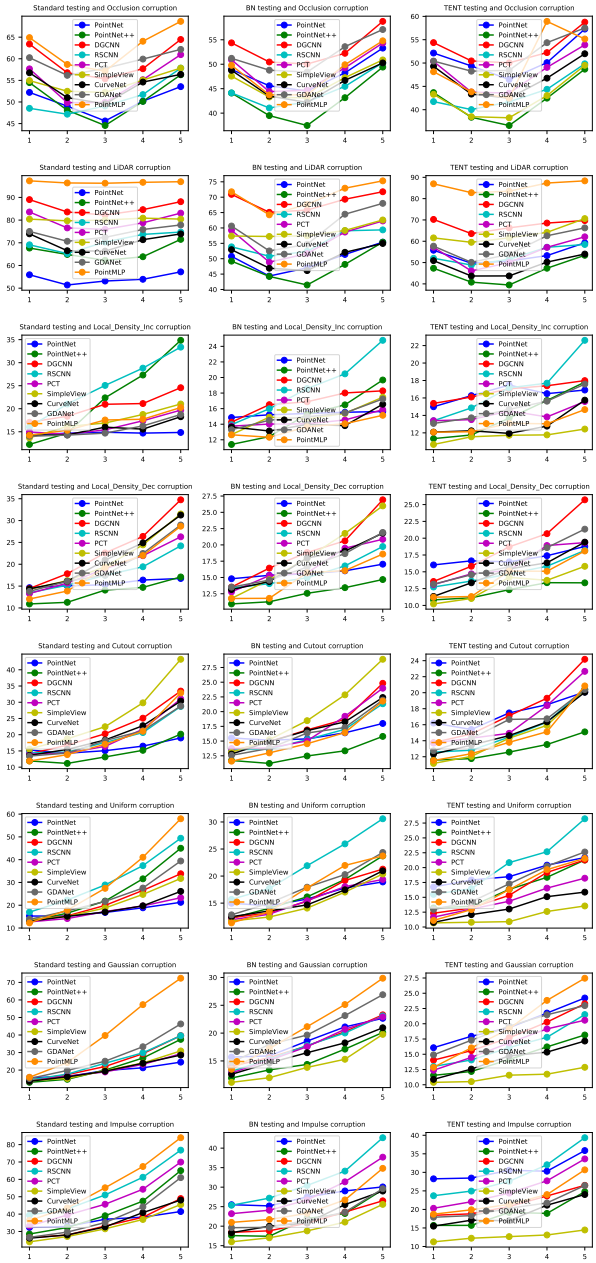

Figure 14: Class-wise Mean Error Rates of Different Models with Different Test-time Adaptation Methods on ModelNet40-C.

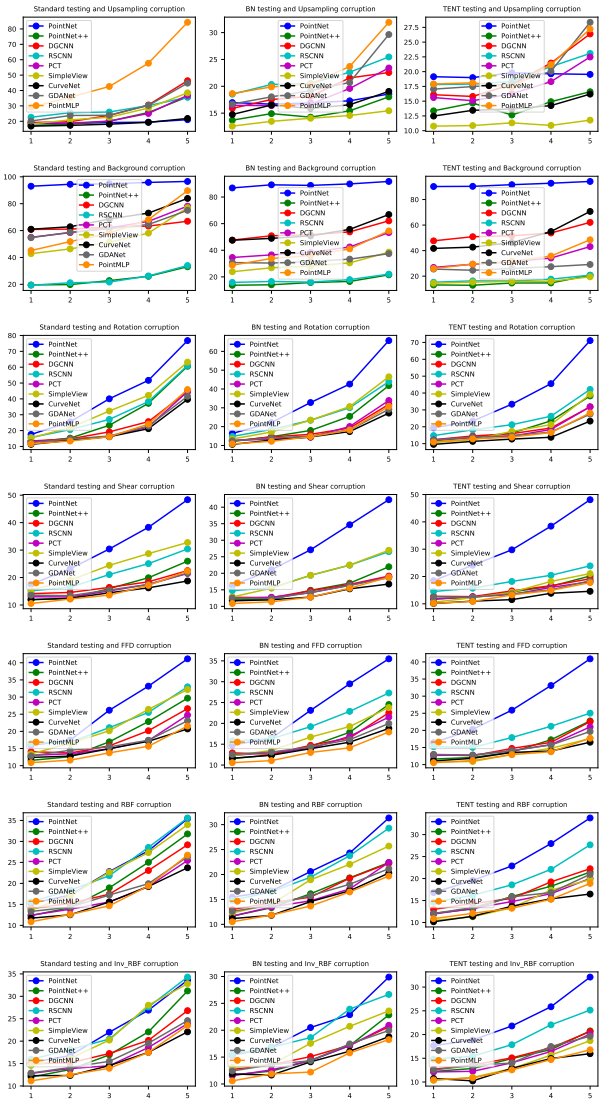

Figure 15: Class-wise Mean Error Rates of Different Models with Different Test-time Adaptation Methods on ModelNet40-C (Cont'd).