Sample-adaptive Augmentation for Point Cloud Recognition Against Real-world Corruptions (Supplementary Material)

This document contains additional details and experiments related to the main paper. Specifically, in Sec. 1, we elaborate on the corruptions settings used in ScanObjectNN-C (Section 4). In Sec. 2, we provide comprehensive details regarding the implementation of Anchor-Sets Fusion (Section 3.1.3). In Sec. 3, we provide further insights into the training setup details (Section 5). In Sec. 4, we present ablation studies on some critical components to demonstrate the effectiveness of the proposed model design. Lastly, in Sec. 5, we present full results of ModelNet-C, ScanObjectNN-C, and ShapeNet-C.

1. Corruption Settings of ScanObjectNN-C

In this section, we provide a detailed description of our implementation of corruptions and severity level settings, which is based on the guidelines proposed by ModelNet-C [12]. To facilitate the discussion, we present a visualization of corruption samples in Fig. 1.

Jitter. To introduce the Jitter corruption technique, we apply a Gaussian noise $\epsilon \in \mathcal{N}(0, \sigma^2)$ to the coordinates of each point in our dataset. We experiment with varying values of the noise parameter σ across five levels, namely $\sigma \in \{0.01, 0.02, 0.03, 0.04, 0.05\}$. By adjusting the noise level in this way, we aim to assess the impact of different levels of perturbation on the performance of our model.

Scale. The Scale corruption technique is implemented by randomly scaling the point clouds along the axes. Specifically, we sample scaling coefficients for each axis independently from a uniform distribution $U(1/S, S)$, where S takes on values from the set $S \in \{1.6, 1.7, 1.8, 1.9, 2.0\}$ across five levels. After scaling, the point clouds are renormalized to a unit sphere to maintain consistency across all samples.

Rotate. The Rotate corruption technique is implemented by randomly applying a rotation to the point clouds, which is described by an Euler angle (α, β, γ). The Euler angles are sampled independently from a uniform distribution $A(-\theta, \theta)$, where θ takes on values from the set $\theta \in \{\pi/30, \pi/15, \pi/10, \pi/7.5, \pi/6\}$ across five levels.

Drop-Global. The Drop-Global technique involves randomly shuffling the order of all points in a point cloud and then discarding the last $N * \rho$ points, where N represents the total number of points in the point cloud and $\rho \in \{0.25, 0.375, 0.5, 0.675, 0.75\}$ is a hyperparameter that determines the proportion of points to be dropped. We applied this technique at five different severity levels.

Drop-Local. We describe the Drop-Local corruption technique, which involves dropping a specified number of points from randomly selected local parts of a point cloud. The total number of points to be dropped is determined by $K \in \{100, 200, 300, 400, 500\}$ for the five levels. To determine the number of local parts to drop, we randomly sample C local areas from a uniform distribution, where $C \in U\{1, 2, ..., 8\}$. The local parts are then assigned a cluster size N_i such that the sum of all cluster sizes is equal to K, $K = \sum_{i=1}^{C} N_i$. For each local part, we randomly select a point as the center and drop its N_i-nearest neighbor points (including itself) from the point cloud.

Add-Global. We present the Add-Global corruption technique, which involves the uniform sampling of K points within a unit sphere, and subsequently adding them to the point cloud. The number of points to be added is determined by the selection of K, which can take on one of five distinct values, $K \in \{10, 20, 30, 40, 50\}$.

Add-Local. In the Add-Local corruption technique, we introduce K new points, where $K \in \{100, 200, 300, 400, 500\}$ for the five levels. The approach randomly selects C local centers from the shuffled point cloud, where $C \in U\{1, 2, ..., 8\}$. To ensure that K points are added, we assign N_i points to the i-th local part, where $K = \sum_{i=1}^{C} N_i$. The coordinates of the neighbouring points are generated from a Normal distribution $\mathcal{N}(\mu_i, \sigma_i^2 I)$, where μ_i is the i-th local center's coordinate and $\sigma_i \in \mathcal{U}(0.075, 0.125)$. Finally, each local part is added to the point cloud one by one.

2. Implementation of Anchor-Sets Fusion

We assemble the set of locally deformed point clouds $\mathcal{Q}' = \left\{q_i' \in \mathbb{R}^{N\times 3} | i = 1, ..., M\right\} \in \mathbb{R}^{M\times N\times 3}$ into a whole point cloud $\overline{\mathcal{P}} = \{\overline{p}_j \in \mathbb{R}^3 | j = 1, ..., N\} \in \mathbb{R}^{N\times 3}$ by

Nadaraya-Watson kernel regression. Specifically, we first convert point clouds back to origin space by adding anchors' coordinates to their corresponding subsets, obtaining transformed points $T = \{t_{i,j} \in \mathbb{R}^3 | i = 1, ..., M; j = 1, ..., N\} \in \mathbb{R}^{M\times N\times 3}$. Given input point cloud $\mathcal{P} = \{p_j \in \mathbb{R}^3 | j = 1, ..., N\} \in \mathbb{R}^{N\times 3}$ and anchor points $\mathcal{D} = \{d_i \in \mathbb{R}^3 | i = 1, ...M\} \in \mathbb{R}^{M\times 3}$

$$\overline{p}_j = \frac{\sum_{i=1}^{M} K_h(||p_j - d_i||_2) t_{i,j}}{\sum_{i=1}^{M} K_h(||p_j - d_i||_2)}, \tag{1}$$

where $K_h(||p_i - d_j||_2)$ denotes the coefficient of transformed points, $||p_i - d_j||_2$ denotes the $L2$ distance between j-th point p_j and i-th anchor d_i. $K_h(\cdots)$ is a pre-defined kernel function with bandwidth h.

$$K_h(x) = \exp(\frac{-x^2}{2h^2}), \tag{2}$$

Here, we set bandwidth h to 0.5.

3. Training Setup Details

ModelNet-C. AdaptPoint trains a classifier utilizing stochastic gradient descent (SGD) optimizer with a batch size of 32 for a total of 300 epochs. The optimizer utilizes a learning rate of 0.1, a momentum value of 0.9, and weight decay of 2e-4. The CosineAnnealingLR [6] scheduler is employed to decrease the learning rate to the minimum value of 0.005. The imitator and discriminator models are optimized utilizing the Adam optimizer [3] with betas=(0.5, 0.99) and learning rates of 0.0001 and 0.0004, respectively.

ScanObjectNN-C. AdaptPoint trains a classifier model with a batch size of 32 using the Adam optimizer for a total of 250 epochs. The optimizer is configured with a learning rate of 0.002, beta values of (0.9, 0.999), and weight decay of 0.05. The learning rate is decayed to the minimum value of 1e-4 using the CosineLRScheduler scheme. Furthermore, the imitator and discriminator models are optimized using the Adam optimizer [3]. The imitator optimizer is set to have betas of (0.5, 0.99) and a learning rate of 0.0001. Likewise, the discriminator optimizer utilizes the same optimization strategy, but with a learning rate of 0.0004.

ShapeNet-C. In this experiment, PointMAE [8] was utilized as the baseline, and the model is trained for 300 epochs using the AdamW optimizer. A batch size of 16 is set for the training process. The optimizer was configured with a learning rate of 0.0002 and weight decay of 0.05. Additionally, the learning rate was decayed to the minimum value of 1e-6 by employing the CosineLRScheduler scheme. The imitator and discriminator models were optimized using the Adam optimizer [3] with betas of (0.5, 0.99). The imitator was trained using a learning rate of 0.0001, and the same optimization strategy was employed to train the discriminator with a learning rate of 0.0004.

Evaluation metrics. We used evaluation metrics (mCE, mOA and RmCE) as specified in previous work [12].

Explanation of reported baseline results. We did not employ multiple training/inference such as voting. Our reproduced PointNet++ performance on ScanObjectNN is elevated (86.2% vs. original 77.9%), owing to the integration of Label Smoothing, AdamW optimizer, and Cosine Decay during training. We reproduced PointNeXt using the same training strategy and achieved 87.3% OA, which is improved to 88.5% (↑ 1.2%) by using our method.

4. Additional Ablation Studies

Effect of Global Features in Deformation Controller. We investigated the impact of incorporating global structure information in Deformation Controller. Our experimental findings, presented in Tab. 1, demonstrate that the omission of global information leads to a notable decline in the AdaptPoint's performance, with a decrease from 78.3% to 83.3%. Our results indicate that global structure information plays a pivotal role in Deformation Controller.

Table 1. Effect of global feature in Deformation Controller

Cross-Anchor Feat	Global Feat	mCE(↓)
✓	✗	83.3
✓	✓	**78.3**

Effect of Global Features in Mask Controller In this study, we conducted additional experiments to investigate the impact of global structure information in Mask Controller. Our results, presented in Tab. 2, indicate that the absence of global information leads to a noticeable reduction in the AdaptPoint's performance, with a decline from 78.3% to 82.5%. Our findings suggest that global structure information plays a crucial role in Mask Controller.

Table 2. Effect of global feature in Mask Controller

Cross-Point Feat	Global Feat	mCE(↓)
✓	✗	82.5
✓	✓	**78.3**

Combining AdaptPoint with RSMix. The mix strategy known as RSMix [4] has been shown to significantly enhance point cloud model robustness. Our AdaptPoint, on the other hand, is a type of deformation data augmentation. Given that these two strategies have demonstrated improved performance when used in conjunction, we sought to evaluate their combined effectiveness using the results presented in Tab. 5. However, our experimental findings indicate that the combination of AdaptPoint and RSMix did not yield substantial improvements.

Effect of cross interaction in Imitator. This study aims to explore the significance of cross interaction in Imitator through further experiments. Tab. 8 presents the results, which reveal that the exclusion of either interaction module results in a decline in network performance. Moreover,

Table 3. Classification results of mOA(%) on ModelNet-C.

Method	mOA(↑)	Sca	Jit	Drop-G	Drop-L	Add-G	Add-L	Rot
DGCNN [13]	76.4	90.6	68.4	75.2	79.3	70.5	72.5	78.5
PointNet [9]	65.8	88.1	**79.7**	87.6	77.8	12.1	56.2	59.1
RSCNN [5]	73.9	89.9	63.0	80.0	68.6	79.0	68.3	68.2
SimpleView [1]	75.7	91.8	77.4	69.2	71.9	71.0	76.8	71.7
GDANet [16]	78.9	92.2	73.5	80.3	81.5	74.3	71.5	78.9
CurveNet [14]	77.9	91.8	77.1	82.4	78.8	60.3	72.5	82.6
PAConv [15]	73.0	91.5	53.7	75.2	79.2	68.0	64.3	79.2
PointNet++ [10]	75.1	91.8	62.8	84.1	62.7	81.9	72.7	69.8
+PointWOLF [2]	80.5	**92.3**	57.3	83.3	73.0	87.3	81.2	89.0
+RSMix [4]	79.9	91.6	47.9	88.0	84.7	92.3	91.0	63.8
+WOLFMix [12]	85.1	91.1	56.7	88.6	87.3	91.2	91.9	89.1
+AdaptPoint	85.9	89.7	67.7	90.9	86.2	91.0	89.0	87.0
PointNeXt [11]	80.5	91.5	59.0	79.0	80.2	92.6	92.4	68.6
+PointWOLF [2]	81.4	91.7	54.2	71.8	78.5	92.2	92.3	89.0
+RSMix [4]	79.7	90.6	49.7	78.5	88.8	**93.2**	92.7	64.4
+WOLFMix [12]	82.3	92.1	50.7	70.4	88.3	93.0	**93.1**	88.6
+AdaptPoint	84.7	90.5	63.1	85.0	85.7	91.3	91.1	86.2
RPC [12]	79.5	92.1	71.8	87.8	83.5	72.6	72.2	76.8
+PointWOLF [2]	84.1	91.7	66.1	80.1	78.3	92.0	92.0	88.3
+RSMix [4]	84.2	90.7	68.1	82.1	87.1	92.2	92.0	76.9
+WOLFMix [12]	86.5	90.5	69.4	89.5	**89.4**	90.2	86.8	**89.7**
+AdaptPoint	**87.7**	91.2	75.5	**91.0**	84.9	91.9	92.0	87.4

Table 4. Classification Results of mOA(%) on ScanObjectNN-C.

Method	mOA(↑)	Sca	Jit	Drop-G	Drop-L	Add-G	Add-L	Rot
DGCNN [13]	62.8	57.8	45.6	62.2	69.7	54.0	77.3	73.3
+PointWOLF [2]	63.0	62.2	43.1	60.6	70.2	54.9	77.1	72.9
+RSMix [4]	63.8	56.4	47.0	65.5	74.1	51.7	77.6	73.9
+WOLFMix [12]	65.2	61.1	44.1	65.0	74.2	55.7	80.5	75.6
+AdaptPoint	66.1	61.7	41.5	72.7	77.9	57.0	78.8	72.9
PointNet++ [10]	64.1	62.1	40.0	79.2	61.3	56.4	79.5	70.5
+PointWOLF [2]	64.3	64.5	40.9	73.3	52.5	**59.6**	79.4	79.7
+RSMix [4]	66.0	62.4	45.2	79.0	70.0	56.5	79.8	69.3
+WOLFMix [12]	66.9	**66.4**	40.7	75.7	64.3	59.4	**81.9**	79.7
+AdaptPoint	67.9	63.6	38.8	**83.3**	76.5	57.5	80.4	75.1
RPC [12]	52.2	44.4	41.6	44.9	60.4	47.4	64.0	62.6
+PointWOLF [2]	49.1	43.9	38.4	46.2	59.5	37.8	60.7	57.2
+RSMix [4]	48.0	36.9	**48.4**	51.9	59.6	29.3	52.6	56.9
+WOLFMix [12]	55.6	54.1	36.1	53.7	67.5	43.2	71.7	63.1
+AdaptPoint	64.1	55.1	45.3	74.7	76.4	50.6	76.0	70.4
PointNeXt [11]	65.5	66.1	41.3	69.5	71.4	56.5	80.1	73.4
+PointWOLF [2]	65.8	65.6	38.6	66.0	72.0	56.3	81.0	81.0
+RSMix [4]	66.6	64.6	41.6	71.6	77.8	55.7	81.2	73.6
+WOLFMix [12]	66.4	65.4	35.1	66.0	76.4	58.9	81.8	**81.3**
+AdaptPoint	**70.0**	65.8	44.0	80.8	**81.0**	58.1	81.3	79.5

Table 5. Effect of combining Rsmix with AdaptPoint

AdaptPoint	Rsmix	mCE(↓)
✓	✗	**78.3**
✓	✓	85.9

when both modules are eliminated, there is a substantial deterioration in network performance. These findings demonstrate that cross interaction is a crucial factor in the AdaptPoint process.

5. Full Results

ModelNet-C. We present complete results for both the mOA and RmCE metrics in Tab. 3 and Tab. 6, respectively. **ScanObjectNN-C.** We show full results for the mOA metric and the RmCE metrics in Tab. 4 and Tab. 7, respectively.

Table 6. Classification results of RmCE(%) on ModelNet-C.

Method	RmCE(↓)	Sca	Jit	Drop-G	Drop-L	Add-G	Add-L	Rot
DGCNN [13]	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
PointNet [9]	148.8	130.0	**45.5**	17.8	97.0	355.7	171.6	224.1
RSCNN [5]	120.1	120.0	121.1	70.7	178.2	60.2	119.4	170.9
SimpleView [1]	118.1	105.0	68.2	142.0	165.4	103.6	85.1	157.4
GDANet [16]	86.5	60.0	82.2	75.3	89.5	86.4	109.0	102.8
CurveNet [14]	97.8	100.0	69.0	65.5	112.8	151.6	106.0	79.4
PAConv [15]	121.1	105.0	164.9	105.7	108.3	115.8	145.8	102.1
PointNet++ [10]	111.4	60.0	124.8	51.1	227.8	50.2	101.0	164.5
+PointWOLF [2]	77.6	60.0	149.6	58.6	154.1	28.1	61.2	31.9
+RSMix [4]	83.5	80.0	187.2	29.9	63.9	4.1	10.9	208.5
+WOLFMix [12]	51.8	100.0	150.4	25.9	43.6	8.6	6.0	28.4
+AdaptPoint	44.6	110.0	100.0	5.7	42.9	4.1	14.4	34.8
PointNeXt [11]	76.6	55.0	138.8	78.2	93.2	0.0	1.0	170.2
+PointWOLF [2]	60.6	25.0	157.0	117.2	103.0	0.0	-0.5	**22.7**
+RSMix [4]	90.6	130.0	179.8	84.5	33.1	0.0	2.5	204.3
+WOLFMix [12]	59.4	45.0	174.8	129.9	35.3	0.0	-0.5	31.2
+AdaptPoint	38.9	40.0	116.5	36.2	42.1	0.0	1.0	36.2
RPC [12]	77.8	45.0	87.6	29.9	71.4	92.3	103.5	114.9
+PointWOLF [2]	45.7	**15.0**	107.0	68.4	103.0	0.0	0.0	26.2
+RSMix [4]	54.3	75.0	99.6	58.0	38.3	0.0	1.0	108.5
+WOLFMix [12]	51.7	140.0	98.8	21.8	**29.3**	14.0	32.3	25.5
+AdaptPoint	**27.4**	35.0	67.8	**5.2**	52.6	**0.0**	**-0.5**	31.9

Table 7. Classification Results of RmCE(%) on ScanObjectNN-C.

Method	RmCE(↓)	Sca	Jit	Drop-G	Drop-L	Add-G	Add-L	Rot
DGCNN [13]	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
+PointWOLF [2]	98.4	83.5	105.7	105.6	95.5	96.5	99.8	101.8
+RSMix [4]	98.1	107.3	98.3	88.9	76.9	109.3	105.1	100.7
+WOLFMix [12]	92.3	93.2	107.1	94.2	81.1	99.0	78.5	93.3
+AdaptPoint	**74.4**	80.8	106.6	49.6	40.4	85.9	**65.6**	91.7
PointNet++ [10]	97.6	85.9	114.8	29.7	154.4	93.5	79.1	125.9
+PointWOLF [2]	97.7	78.5	113.6	56.0	211.2	**84.6**	84.6	55.1
+RSMix [4]	95.0	88.7	104.7	35.3	107.4	96.8	88.5	143.8
+WOLFMix [12]	85.8	**75.2**	116.3	49.7	143.6	88.0	65.5	62.4
+AdaptPoint	76.8	82.2	119.0	**14.5**	63.1	91.6	74.3	92.8
RPC [12]	102.1	108.2	82.3	126.0	88.8	85.8	126.6	97.3
+PointWOLF [2]	98.6	96.7	80.9	104.8	71.4	104.3	121.1	110.7
+RSMix [4]	82.7	102.5	**43.0**	58.1	**37.6**	114.3	153.7	69.9
+WOLFMix [12]	103.5	91.0	108.3	109.7	75.5	114.4	93.5	132.3
+AdaptPoint	82.2	101.5	95.0	37.3	44.1	103.4	88.8	104.9
PointNeXt [11]	93.9	75.8	114.4	75.5	98.6	96.7	84.7	111.2
+PointWOLF [2]	87.2	77.8	121.4	90.6	95.6	97.9	75.6	51.3
+RSMix [4]	90.3	83.9	115.6	69.6	64.0	101.7	81.3	116.3
+WOLFMix [12]	83.4	79.6	130.9	91.6	70.4	90.6	69.2	**51.2**
+AdaptPoint	74.5	80.8	110.6	32.4	46.5	95.4	84.1	71.4

Table 8. Effect of cross interaction in Imitator

cross-anchor	cross-point	mCE(↓)
✗	✗	81.4
✗	✓	79.1
✓	✗	78.5
✓	✓	**78.3**

ShapeNet-C. The complete results of the RmCE metric are presented in Tab. 9 for ShapeNet-C dataset.

Table 9. Segmentation Results of RmCE $(\%)$ on ShapeNet-C.

Method	RmCE(↓)	Sca	Jit	Drop-G	Drop-L	Add-G	Add-L	Rot
DGCNN [13]	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
PointNet [9]	105.6	35.5	**88.0**	8.7	115.2	156.6	120.6	214.4
PointNet++ [10]	185.0	68.5	132.9	17.6	786.0	83.0	116.9	90.1
PAConv [15]	84.8	56.0	133.6	76.4	78.9	59.7	94.7	94.0
GDANet [16]	78.5	26.4	111.5	80.6	84.2	53.5	95.2	97.9
PT [18]	93.3	98.1	105.1	72.8	107.7	113.3	105.4	50.7
Point-MLP [7]	81.0	47.4	142.8	21.7	96.1	88.2	110.9	60.1
Point-BERT [17]	89.5	28.3	135.6	21.3	61.9	130.3	136.0	112.8
Point-MAE [8]	70.3	**18.0**	120.9	22.2	**45.9**	65.0	108.8	111.4
+AdaptPoint	**29.3**	23.3	115.6	**-9.6**	47.4	**0.0**	**0.3**	**28.6**

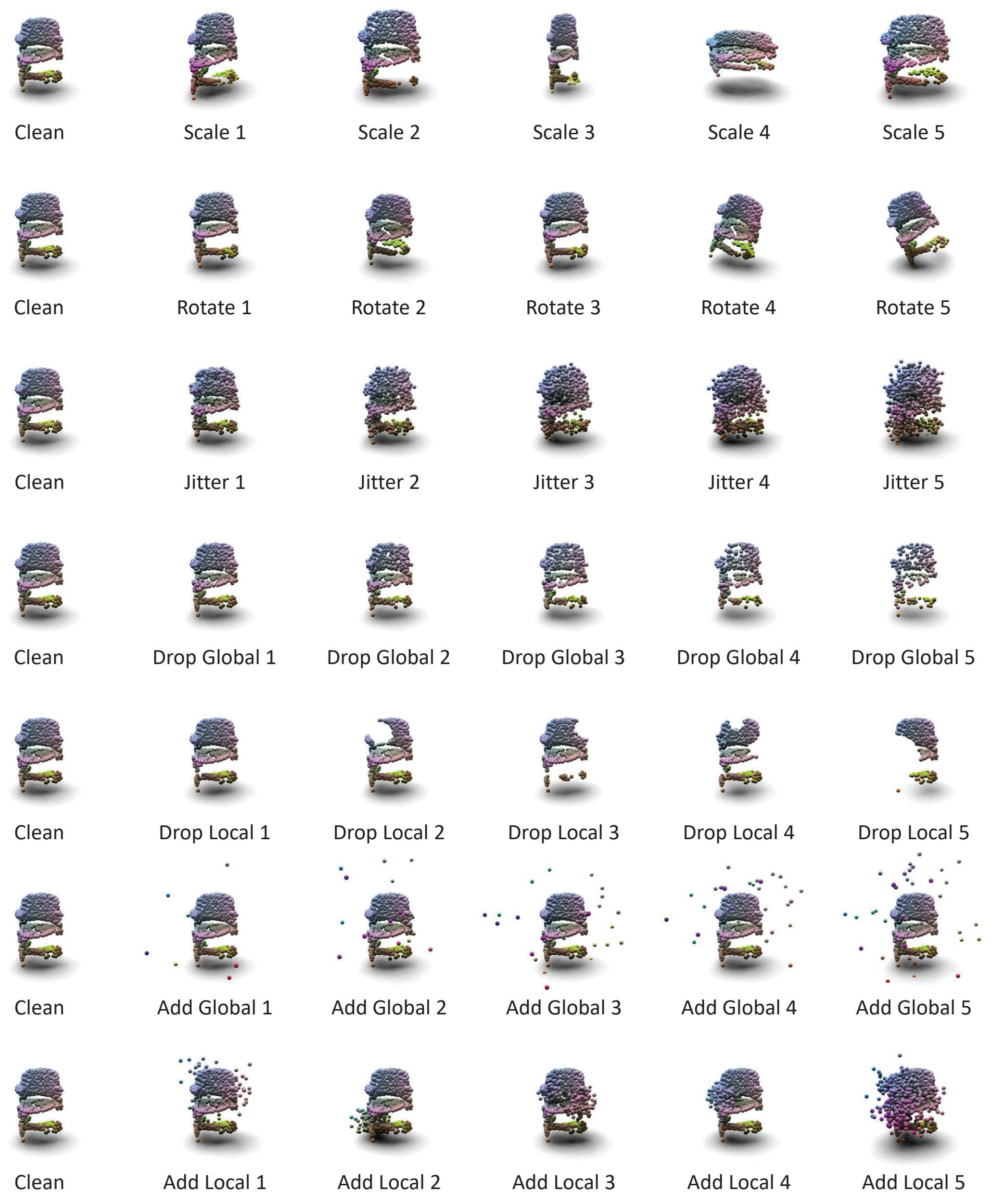

Figure 1. Visualization of corruption on all levels.

References

[1] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *ICML*, pages 3809–3820, 2021. 3

[2] Sihyeon Kim, Sanghyeok Lee, Dasol Hwang, Jaewon Lee, Seong Jae Hwang, and Hyunwoo J Kim. Point cloud augmentation with weighted local transformations. In *ICCV*, pages 548–557, 2021. 3

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[4] Dogyoon Lee, Jaeha Lee, Junhyeop Lee, Hyeongmin Lee, Minhyeok Lee, Sungmin Woo, and Sangyoun Lee. Regularization strategy for point cloud via rigidly mixed sample. In *CVPR*, pages 15900–15909, 2021. 2, 3

[5] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019. 3

[6] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 2

[7] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. In *ICLR*, 2022. 4

[8] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, pages 604–621. Springer, 2022. 2, 4

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 3, 4

[10] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 3, 4

[11] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In *NIPS*. 3

[12] Jiawei Ren, Liang Pan, and Ziwei Liu. Benchmarking and analyzing point cloud classification under corruptions. *arXiv preprint arXiv:2202.03377*, 2022. 1, 2, 3

[13] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 38(5):1–12, 2019. 3, 4

[14] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. In *ICCV*, 2021. 3

[15] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *CVPR*, 2021. 3, 4

[16] Mutian Xu, Junhao Zhang, Zhipeng Zhou, Mingye Xu, Xiaojuan Qi, and Yu Qiao. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. In *AAAI*, volume 35, pages 3056–3064, 2021. 3, 4

[17] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *CVPR*, pages 19313–19322, 2022. 4

[18] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 4