# Supplementary Material to: Point-DAE: Denoising Autoencoders for Self-supervised Point Cloud Learning

Yabin Zhang, Jiehong Lin, Ruihuang Li, Kui Jia, and Lei Zhang, *Fellow, IEEE*

The following materials are provided in this supplementary file:

- More Results on the ModelNet40 Dataset (please refer to Section IV-C in the main paper).
- Detailed Definitions and Analyses on Studied Corruptions (please refer to Fig. 2 in the main paper).
- More Visualizations (please refer to Section IV-D in the main paper)

## I. More Results on ModelNet40 Dataset

We validate our Point-DAE on the popular synthetic ModelNet40 dataset [1], which includes about $12K$ clean objects shared by $40$ categories, under the fine-tuning protocol. The standard voting strategy [2] is adopted in the testing stage following [3]. Compared to the real-world ScanObjectNN dataset, recognizing synthetic samples is much easier since the point clouds are clean and complete. As illustrated in Tab. A1, our Point-DAE outperforms its competitors with different backbones, validating its effectiveness.

TABLE A1: Classification results on the ModelNet40 dataset. The overall accuracy, *i.e.*, OA (%) is reported. 'ST?' indicates the standard Transformer architecture.

| Methods | ST? | #Points | OA (%) |
|---|---|---|---|
| Supervised References | | | |
| PointNet [4] | – | 1K P | 89.2 |
| PointNet++ [5] | – | 1K P | 90.7 |
| PointCNN [6] | – | 1k P | 92.5 |
| PointConv [7] | – | 1K P+N | 92.5 |
| KPConv [8] | – | 1K P | 92.9 |
| RS-CNN [2] | – | 1k P | 92.9 |
| PointASNL [9] | – | 1k P | 92.9 |
| DensePoint [10] | – | 1k P | 93.2 |
| PosPool [11] | – | 5k P | 93.2 |
| DRNet [12] | – | 1k P | 93.1 |
| GBNet [13] | – | 1k P | 93.8 |
| CurveNet [14] | – | 1k P | 94.2 |
| PCT [15] | ✗ | 1k P | 93.2 |
| PatchFormer [16] | ✗ | 1k P | 93.5 |
| PointTransformer [17] | ✗ | 1k P | 93.7 |
| PointTransformer V2 [18] | ✗ | 1k P | 94.2 |
| NPCT [15] | ✓ | 1k P | 91.0 |

| Methods | ST? | #Points | OA (%) |
|---|---|---|---|
| Self-supervised Methods | | | |
| DGCNN [19] | – | 1k P | 92.5 |
| + Jigsaw [20] | – | 1k P | 92.3 |
| + OcCo [21] | – | 1k P | 93.0 |
| + STRL [22] | – | – | 93.1 |
| + Point-DAE (Ours) | – | 1k P | **93.8** |
| Transformer [23] | ✓ | 1k P | 91.4 |
| + OcCo [21], [24] | ✓ | 1k P | 92.1 |
| + Point-BERT [24] | ✓ | 1k P | 93.2 |
| + Point-BERT [24] | ✓ | 4k P | 93.4 |
| + Point-BERT [24] | ✓ | 8k P | 93.8 |
| + MaskPoint [25] | ✓ | 1k P | 93.8 |
| + Point-MAE [3] | ✓ | 1k P | 93.8 |
| + Point-DAE (Ours) | ✓ | 1k P | **94.0** |
| Hierarchical Transformer [26] | ✗ | 1k P | 92.1 |
| + Point-M2AE [26] | ✗ | 1k P | 94.0 |
| + Point-DAE (Ours) | ✗ | 1k P | **94.2** |

## II. Detailed Definitions and Analyses on Studied Corruptions

We investigate the $14$ studied corruptions in detail in this part. We first give their definitions and conduct individual magnitude analysis with a DGCNN encoder, where we only corrupt the input with the studied corruption.

Generally speaking, the affine transformation corruptions can be achieved by multiplying the augmented input with an affine transformation matrix $A$ as follows:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{1}$$

where $[x, y, z]$ and $[x', y', z']$ are the input and transformed points. Different affine transformation corruptions are distinguished by the used affine transformation matrix $A$, which is defined by the $12$ parameters (*i.e.*, $a_{ij}$).

Y. Zhang, R. Li, and L. Zhang are with the Department of Computing, The Hong Kong Polytechnic University, HongKong. E-mails: csybzhang@comp.polyu.edu.hk, cslzhang@comp.polyu.edu.hk Correspondence to: L. Zhang

J. Lin and K. Jia are with the School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China.

**Rotate.** We randomly rotate the point cloud around the XYZ axes using the following affine transformation matrix $A$:

$$A = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) & 0 \\ 0 & \sin(\theta_x) & \cos(\theta_x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where $\theta_x, \theta_y, \theta_z \in [-\pi, \pi]$ determine the rotation angles around the X,Y, and Z axes, respectively. As illustrated in Fig. A1, better results are typically achieved with a wider range of rotation angles. Practically, we randomly sample $\theta_x, \theta_y, \theta_z$ values from the uniform distribution $U(-\pi, \pi)$ for each instance in every iteration.

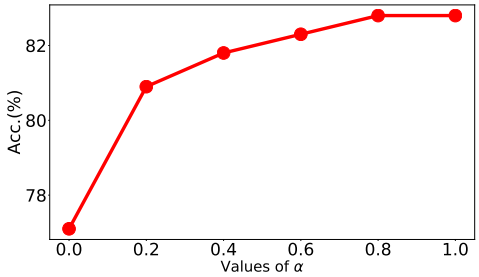



Fig. A1: Rotate corruption results with different rotation angles, where we randomly sample $\theta_x, \theta_y, \theta_z$ values from the uniform distribution $U(-\alpha\pi, \alpha\pi)$.

**Rotate-Z.** We randomly rotate the point cloud around the XYZ axes using the following affine transformation matrix $A$:

$$A = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

where $\theta_z \in [-\pi, \pi]$ determine the rotation angles around the Z axis. Given that Rotate-Z is a subset of the Rotate, we follow Rotate to randomly sample $\theta_z$ values from the uniform distribution $U(-\pi, \pi)$ for each instance in every iteration.

**Translate.** We randomly translate the point cloud along XYZ axes using the following affine transformation matrix $A$:

$$A = \begin{bmatrix} 1 & 0 & 0 & \tau_x \\ 0 & 1 & 0 & \tau_y \\ 0 & 0 & 1 & \tau_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{where } \tau_x, \tau_y, \tau_z \in \mathbb{R}. \tag{4}$$

The magnitude of Translate is decided by the values of $\tau_x, \tau_y, \tau_z$. We sample $\tau_x, \tau_y, \tau_z$ from the uniform distribution $U(-\tau_t, \tau_t)$. The results with different $\tau_t$ are presented in Fig. A2. We see that the results are stable within a wide range of $\tau_t$, *e.g.*, $\tau_t \in [0.3, 0.8]$. We set $\tau_t = 0.5$ in all experiments.

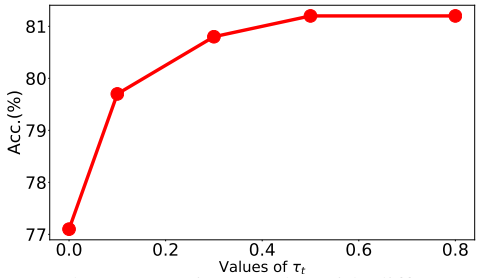



Fig. A2: Translate corruption results with different values of $\tau_t$.

**Reflect.** We randomly reflect the point cloud along the XYZ axes using the following affine transformation matrix $A$:

$$A = \begin{bmatrix} r_x & 0 & 0 & 0 \\ 0 & r_y & 0 & 0 \\ 0 & 0 & r_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{where } r_x, r_y, r_z \in \{-1, 1\}. \tag{5}$$

The point cloud is reflected along the X/Y/Z axis with $r_x/r_y/r_z = -1$. We randomly sample $r_x, r_y, r_z$ from $\{-1, 1\}$ for each instance in every iteration.

**Scale.** We randomly scale the point cloud along the XYZ axes using the following affine transformation matrix $A$:

$$A = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{where } s_x, s_y, s_z \in \mathbb{R}^+ \tag{6}$$

The magnitude of Scale is decided by the values of $s_x, s_y, s_z$. We sample $s_x, s_y, s_z$ from the uniform distribution $U(\frac{1}{\eta}, \eta)$, where $\eta \geq 1$. As illustrated in Fig. A3, a too-small $\eta$ and a too-large $\eta$ both lead to degenerated performance. We set $\eta = 2.0$ in all experiments.

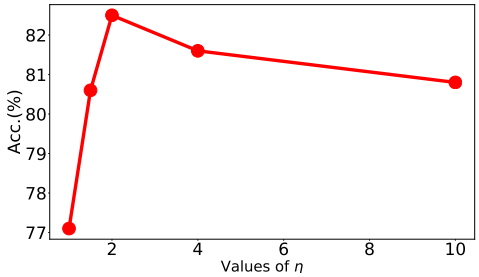



Fig. A3: Scale corruption results with different values of $\eta$.

**Shear.** We randomly shear the point cloud using the following affine transformation matrix $A$:

$$A = \begin{bmatrix} 1 & s_{xy} & s_{xz} & 0 \\ s_{yx} & 1 & s_{yz} & 0 \\ s_{zx} & s_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{where } s_{xy}, s_{xz}, s_{yx}, s_{yz}, s_{zx}, s_{zy} \in \mathbb{R}. \tag{7}$$

The magnitude of Shear is decided by the values of $s_{xy}, s_{xz}, s_{yx}, s_{yz}, s_{zx}, s_{zy}$. We sample them from the uniform distribution $U(-\eta, \eta)$. Results with different values of $\eta$ are presented in Fig. A4 and we set $\eta = 1.0$ in all experiments.

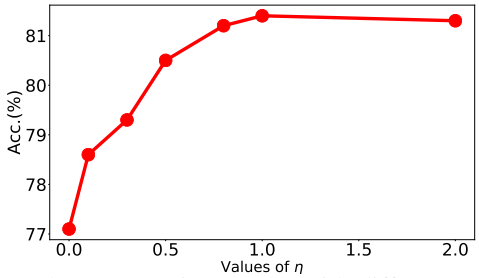



Fig. A4: Shear corruption results with different values of $\eta$.

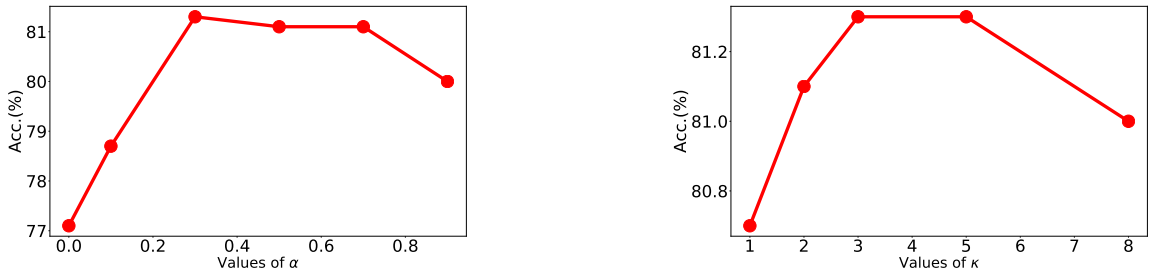
**Affine.** The complete affine transformation comprises the sub-transformations of Rotate, Translate, Scale, Reflect and Shear. In practice, we randomly select $K \in \{1, 2, 3\}$ items from the sub-transformations and then apply the selected ones to the input point cloud in random order. As expected, the complete affine transformation outperforms its sub-transformations, as presented in Tab. I of the main paper.

**Drop-Local.** The Drop-Local is detailed in Sec. III-B of the main paper and analyzed here. Specifically, we analyze the Drop-Local strategy by investigating the number of dropped clusters $\kappa$ and the masking ratio $\alpha$. Results with different $\alpha$ and $\kappa$ are illustrated in Fig. A5. We empirically set $\alpha$ and $\kappa$ as 0.3 and 5, respectively.

**Drop-Patch.** We follow [3] and [26] to define the Drop-Patch strategy, as detailed in Sec. III-C of the main paper.

**Scan.** We simulate the point cloud scanned with a LiDAR in the Scan corruption, where the point cloud becomes sparser ifas it is farther from the LiDAR position. Specifically, we randomly select one point on the unit sphere surface as the LiDAR position. Then, we measure the $L_2$ distance between the LiDAR position and each point, and the points are discarded with a probability (*i.e.*, dropping rate) proportional to the distances. Following [27], we multiply the basic dropping rate with a random gate parameter $g \in [1, 5]$, where the larger the gate parameter, the sparser the point cloud.

**Drop-Global.** Similar to the popular masking strategy, we mask partial points of the input with the Drop-Global corruption. However, unlike the masking corruption, where points are masked nonuniformly, we uniformly drop some portion of points in the Drop-Global, resulting in a low-resolution point cloud. Specifically, all points are randomly shuffled, and the last $\lfloor \alpha w \rfloor$ points are dropped, where $w$ is the total number of points and $\alpha \in (0, 1)$ is the masking ratio. As shown in Fig. A6, a certain




(a) Drop ratio $\alpha$ (b) Cluster number $\kappa$

Fig. A5: Drop-Local corruption results with different values of masking ratio $\alpha$ and cluster number $\kappa$. We fix the cluster number $\kappa$ as 5 when analyzing the influence of $\alpha$, while fix the drop ratio as $0.3$ when studying the cluster number $\kappa$.

amount of improvement (*e.g.*, from 77.1% to 79.3%) is achieved with a small masking ratio $\alpha = 0.1$, which is inferior to the popular masking strategy (*e.g.*, 81.3%). This fact validates the advantages of nonuniform masking over uniform masking.

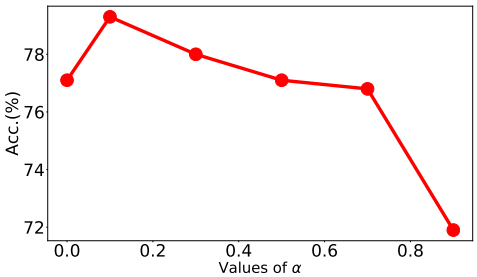



Fig. A6: Drop-Global corruption results with masking ration $\alpha$.

**Jitter.** Similar to affine transformation, the Jitter operation also corrupts all points. Specifically, a random jitter sampled from Gaussian distribution $N(0, \sigma^2)$ is added to each point. As shown in Fig. A7, adding point-wise noise via Jitter also introduces some improvement (*e.g.*, from 77.1% to 80.7%), but far lags behind that with affine transformation (*e.g.*, 84.4%).

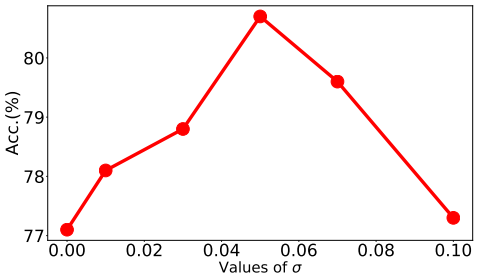



Fig. A7: Jitter corruption results with different values of $\sigma$.

**Add-Global.** Corrupting the point cloud by adding new points that do not belong to the vanilla input is another direction. We add such noise points in the global space and local regions, leading to the corruptions of Add-Global and Add-Local, respectively. Specifically, in Add-Global, we uniformly sample a number of $\lfloor \alpha w \rfloor$ noise points in the unit sphere, where $w$ is the total number of points and $\alpha \in \mathbb{R}^+$ is the adding ratio. Results with different values of $\alpha$ are shown in Fig. A8. Compared to the masking (*e.g.*, 81.3%) and affine transformation (*e.g.*, 84.4%) corruptions, adding noise points globally leads to less improvement.

**Add-Local.** We add noise points in local regions by adding several local clusters with the Add-Local corruption. Specifically, we first randomly set the number of added clusters $\kappa \in \mathbb{Z}^+$ and the total number of added points $\eta = \lfloor \alpha w \rfloor$, where $\alpha \in \mathbb{R}^+$ is the adding ratio. Then, for the $i$-th cluster, we randomly select a point in the input as the cluster center and randomly add $\eta_i$ points around the center point, where $\sum_{i=1}^{\kappa} \eta_i = \eta$ and $\eta_i > 0$. As illustrated in Fig. A9, adding noise points locally leads to less improvement compared to the masking (*e.g.*, 81.3%) and affine transformation (*e.g.*, 84.4%) corruptions. The results with Add-Global and Add-Local corruptions suggest that adding noise points to the clean point cloud input is a less effective sample corruption for the generation-based SSL framework.

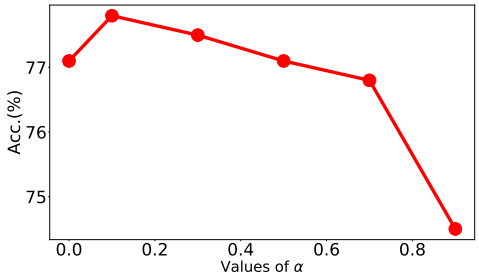
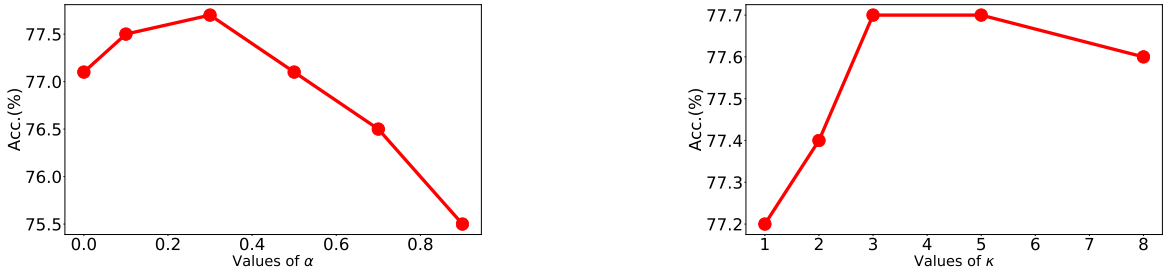



Fig. A8: Add-Global corruption results with different adding ratio $\alpha$.

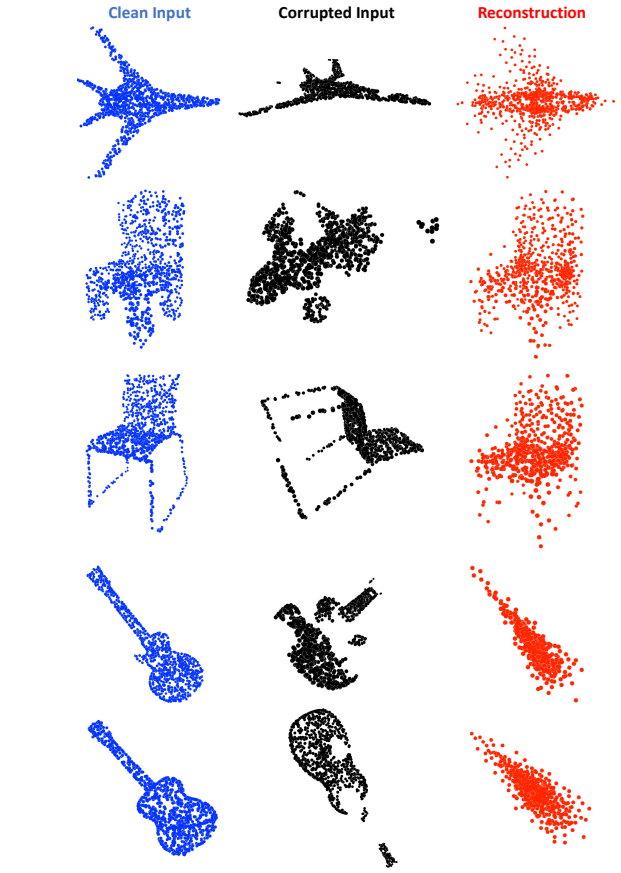



(a) Adding ratio $\alpha$ (b) Cluster number $\kappa$

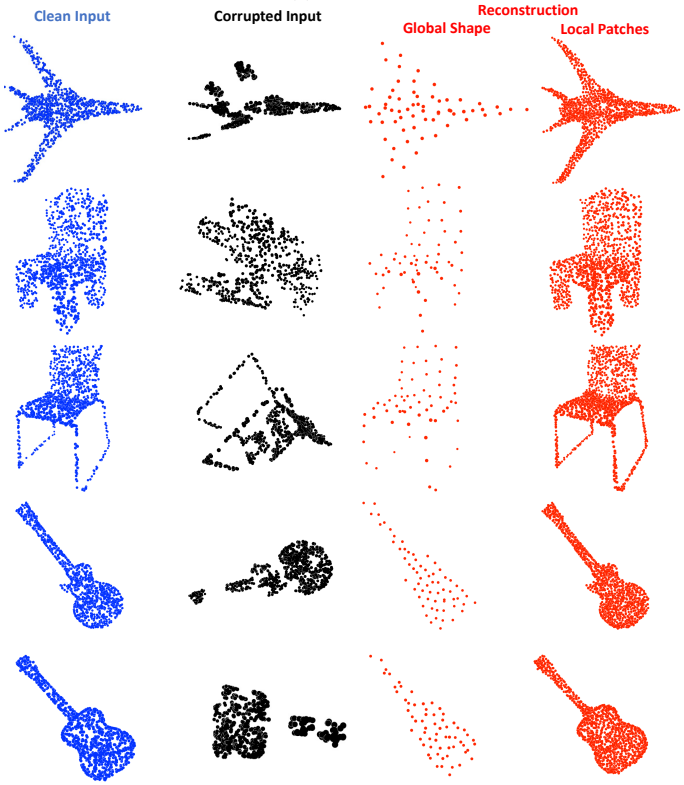
Fig. A9: Add-Local corruption results with different values of adding ratio $\alpha$ and cluster number $\kappa$. We fix the cluster number $\kappa$ as 3 when analyzing the influence of $\alpha$, while the adding ratio is fixed as $0.3$ when studying the cluster number $\kappa$.

## III. More Visualizations.

More reconstruction results with DGCNN and Transformer backbones are visualized in Fig. A10. Similar to the observations in Fig. 8 of the main paper, the DGCNN backbone mainly reconstructs the overall shape of the point cloud, while the Transformer backbone reconstructs well both the global shape and local details, thanks to its decomposition of the reconstruction objectives.




(a) DGCNN




(b) Transformer

Fig. A10: More reconstruction visualization on ShapeNet validation set with backbones of DGCNN and Transformer.

## REFERENCES


[1] 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*, 2015, pp. 1912–1920.

[2] 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*, 2019, pp. 8895–8904.

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

[4] 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*, 2017, pp. 652–660.

[5] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[6] 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*, vol. 31, 2018.

[7] 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*, 2019, pp. 9621–9630.

[8] 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*, 2019, pp. 6411–6420.

[9] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "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*, 2020, pp. 5589–5598.

[10] Y. Liu, B. Fan, G. Meng, J. Lu, S. Xiang, and C. Pan, "Densepoint: Learning densely contextual representation for efficient point cloud processing," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5239–5248.

[11] Z. Liu, H. Hu, Y. Cao, Z. Zhang, and X. Tong, "A closer look at local aggregation operators in point cloud analysis," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer, 2020, pp. 326–342.

[12] S. Qiu, S. Anwar, and N. Barnes, "Dense-resolution network for point cloud classification and segmentation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 3813–3822.

[13] ——, "Geometric back-projection network for point cloud classification," *IEEE Transactions on Multimedia*, vol. 24, pp. 1943–1955, 2021.

[14] 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*, 2021, pp. 915–924.

[15] 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*, vol. 7, no. 2, pp. 187–199, 2021.

[16] C. Zhang, H. Wan, X. Shen, and Z. Wu, "Patchformer: An efficient point transformer with patch attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 799–11 808.

[17] 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*, 2021, pp. 16 259–16 268.

[18] X. Wu, Y. Lao, L. Jiang, X. Liu, and H. Zhao, "Point transformer v2: Grouped vector attention and partition-based pooling," *arXiv preprint arXiv:2210.05666*, 2022.

[19] 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)*, vol. 38, no. 5, pp. 1–12, 2019.

[20] J. Sauder and B. Sievers, "Self-supervised deep learning on point clouds by reconstructing space," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[21] H. Wang, Q. Liu, X. Yue, J. Lasenby, and M. J. Kusner, "Unsupervised point cloud pre-training via occlusion completion," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9782–9792.

[22] S. Huang, Y. Xie, S.-C. Zhu, and Y. Zhu, "Spatio-temporal self-supervised representation learning for 3d point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6535–6545.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[24] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "Point-bert: Pre-training 3d point cloud transformers with masked point modeling," *arXiv preprint arXiv:2111.14819*, 2021.

[25] H. Liu, M. Cai, and Y. J. Lee, "Masked discrimination for self-supervised learning on point clouds," *arXiv preprint arXiv:2203.11183*, 2022.

[26] R. Zhang, Z. Guo, P. Gao, R. Fang, B. Zhao, D. Wang, Y. Qiao, and H. Li, "Point-m2ae: Multi-scale masked autoencoders for hierarchical point cloud pre-training," *arXiv preprint arXiv:2205.14401*, 2022.

[27] C. Huang, Z. Cao, Y. Wang, J. Wang, and M. Long, "Metasets: Meta-learning on point sets for generalizable representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8863–8872.