# Supplementary Materials for Robust 3D Shape Classification via Non-local Graph Attention Network

Shengwei Qin[1] Zhong Li[2,3*] Ligang Liu[4]

[1]School of Mechanical Engineering, Zhejiang Sci-Tech University
[2]School of Information Engineering, Huzhou University
[3]School of Science, Zhejiang Sci-Tech University
[4]School of Mathematical Sciences, University of Science and Technology of China

qsw.sise@gmail.com, lizhong@zjhu.edu.cn, lgliu@ustc.edu.cn

## Appendix 1

**Theorem 1.** *For any two points $x_i$ and $x_j$ on the point cloud model, their neighborhood matrices are $X_{is}, X_{js}$, and their Gram matrices are $G(X_{is}), G(X_{js})$, respectively. If $G(X_{is})$ and $G(X_{js})$ is close, which ensures that the difference of their $F$-norms is equal to or less than a fixed value,* i.e.,

$$\|G\left(X_{is}\right)-G\left(X_{js}\right)\|_{\mathrm{F}} \leq \frac{\sigma_C^2(X_{is})}{2} \tag{1}$$

*then there exists a rotation matrix $R$ such that the below inequality also holds.*

$$\min_R \|X_{is}-RX_{js}\|_{\mathrm{F}} \leq \frac{\sqrt{2}\sigma_C(X_{is})}{2} \tag{2}$$

*Namely, the minimal $F$-norm difference between $X_{is}$ and rotated $X_{js}$ is also equal to or less than a value related to $\sigma_C$. Here, $\sigma_C$ is the minimum singular value of the matrix $X_{is}$, $\|X\|_{\mathrm{F}}=\sqrt{\mathrm{Tr}\left(X^TX\right)}$.*

*Proof.* For two neighborhood matrices $X_{is}$, $X_{js}$ ($X_{is}$, $X_{js}$ $\in \mathbb{R}^{3\times k}$, $k\in\mathbb{N}^+$) composed of any two points $x_i$ and $x_j$, if the inequality Eq. (1) is guaranteed, then the left side of inequality Eq. (2) satisfies the following inequality (according to Theorem 3.2 in Pumir *et al.* [12]).

$$\begin{aligned}&\min_R\|X_{is}-RX_{js}\|_F \leq \\ &\frac{\sigma_C\left(X_{is}\right)}{\sqrt{2}}\left(1-\sqrt{1-\frac{2\|G\left(X_{is}\right)-G\left(X_{js}\right)\|_F}{\sigma_C^2\left(X_{is}\right)}}\right)\end{aligned} \tag{3}$$

Suppose $\min_R \|X_{is}-RX_{js}\|_{\mathrm{F}} \leq \rho\sigma_C(X_{is})$, where $\rho$ is an undetermined coefficient. If the inequality Eq. (2) holds, then the following inequality Eq. (4) is required to be valid,

$$\begin{aligned}&\frac{\sigma_C\left(X_{is}\right)}{\sqrt{2}}\left(1-\sqrt{1-\frac{2\|G\left(X_{is}\right)-G\left(X_{js}\right)\|_F}{\sigma_C^2\left(X_{is}\right)}}\right) \\ &\leq \rho\sigma_C(X_{is})\end{aligned} \tag{4}$$

By simplification, it follows

$$1-\sqrt{2}\rho \leq \sqrt{1-\frac{2\|G\left(X_{is}\right)-G\left(X_{js}\right)\|_F}{\sigma_C^2\left(X_{is}\right)}} \tag{5}$$

Obviously, when $\rho \geq \frac{\sqrt{2}}{2}$, the above inequality always keeps. We choose $\rho=\frac{\sqrt{2}}{2}$ to let $\min_R\|X_{is}-RX_{js}\|_{\mathrm{F}}$ satisfy the minimum upper bound. Namely, the inequality Eq. (2) is valid when $\|G\left(X_{is}\right)-G\left(X_{js}\right)\|_F \leq \frac{\sigma_C^2(X_{is})}{2}$. □

## Appendix 2

**Geometric Interpretation of Theorem 1.** For two points $x_i$ and $x_j$ with their neighborhood matrices $X_{is}$ and $X_{js}$, if there exists a rotation (or reflection) matrix $R$ satisfying with Theorem 1, then we can conclude the geometric structures consisting of two points and their neighbors are similar. For example, given the geometric design, the orange region (red point with its neighbors shown in an auxiliary Fig. 1 [25]) is similar to the blue region (blue points with their corresponding neighbors)*

The advantage of finding similar points in the geometric space is that more information can be collected especially when point clouds are sparse. As for the comparison shown in Fig. 2, the traditional feature extraction with

*Corresponding author.

*Although the way of finding points is similar to GS-Net [25], the network architecture and experiments results of our NLGAT are superior to those by GS-Net [25].

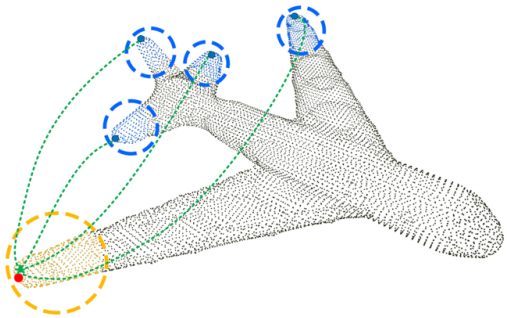


Figure 1. Geometric interpretation of Thm 1 on point clouds.

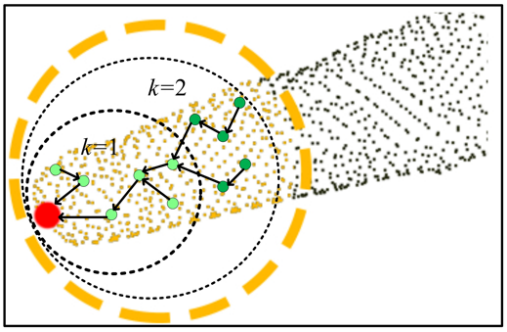



Figure 2. Node neighborhood feature aggregation process [6].

progressive feature aggregation involves many points in the flat region, which is equivalent to adding noise to the aggregated points during the feature computation, and the similar faraway points will be scoped by stacking hundreds of network layers, while our method can directly find similar points according to the condition of Theorem 1.

## Appendix 3

The detailed steps of multi-scale Gram matrices construction are described in the following algorithm 1*.

## Appendix 4

A series of feature representations are obtained through the following aggregation operations with a Gram matrix input.

$$\begin{cases} E_{ii} = h_\theta\left(x_i, x_i\right) \\ E_{ij} = h_\theta\left(x_i, x_{ij}\right) \\ E_{it} = h_\theta\left(x_i, x_{it}\right) \end{cases} \tag{6}$$

where $h_\theta : R^C \times R^C \rightarrow R^{C'}$ is the feature aggregation function, $C^{'}$ is the feature dimensional, $\theta$ is the learnable parameter. $E_{ii}$ is the self-feature learning, $E_{ij}$ $(1 \leq j \leq k)$ is the first-order neighborhood feature learning, and $E_{it}$ $(1 \leq t \leq k-1)$ is the feature learning of ssimilar geometric structures in the space without the first-order neighborhood.

The learned features (Eq. (6)) are then fused by a single-layer convolutional network and normalized by using the Softmax function to generate attention coefficients as follows.

$$\begin{cases} \alpha_{ij} = \frac{\mathcal{G}(E_{ii}+E_{ij})}{\sum_j \mathcal{G}(E_{ii}+E_{ij})} \\ \alpha_{it} = \frac{\mathcal{G}(E_{it})}{\sum_j \mathcal{G}(E_{ii}+E_{ij})+\sum_t \mathcal{G}(E_{it})} \end{cases} \tag{7}$$

where $\mathcal{G}(\cdot) = e^{LeakyReLU(\cdot)}$, and LeakyReLU$(\cdot)$ is the nonlinear activation function Leaky ReLU.

*Note that for a sparse point cloud with only 8 points, a Gram matrix is directly input to the network for feature learning during the experiments.

## Appendix 5

**Experimental settings.** The experimental results are evaluated on a server with NVIDIA Tesla K80 GPU. We refer to other classification methods for each experiment to normalize the point cloud model into the standard $[-1, 1]$ as the network input. We optimize the network with Adam optimizer and use the cross-entropy as the loss function. The main parameters in NLGAT are set as shown in Tab. 1. In addition, the three main convolution kernels in NLGAT are listed as follows.

- If the data input size is $2k \times 2k \times N$, the size of the convolution kernel is $3 \times 3 \times N$, the number of convolution kernel is $N$, stride=4, padding=0, and the output size is $\lfloor \left(\frac{2k-3}{4}+1\right) \times \left(\frac{2k-3}{4}+1\right) \times N \rfloor$ (stride$=4$, padding$=0$).
- If the data input size is $1 \times N$, the size of the convolution kernel is $1 \times 1$, the number of convolution kernel is 1024, and the output size is $1024 \times N$.
- In other cases, the data input size is $3 \times N$. The size of the convolutional kernel is $3 \times 1$, and the number of convolutional kernel is 1024. The output size is $1024 \times N$.

| Hyper-parameter | Value |
|---|---|
| Window sizes of 2D Conv Layers | 3/3 |
| Values k in two k-max pooling Layers | 2/2 |
| Label Smoothing | 0.20 |
| Learning Rate | 0.02 |
| Batchsize | 16 |
| Max Epoch | 200 |
| Decay Step | 200000 |
| Decay Rate | 0.70 |

Table 1. Hyper-parameters in NLGAT

**Algorithm 1:** Multi-scale Gram Matrices Construction Based on Local Information Entropy

**Input:** Point cloud: $X \in \mathbb{R}^{3\times N}$, points:$N$, neighborhood ranges $k$ : 4 points.
**Output:** Multi-scale Gram matrices: $G_k(X)$

**for** $i = 1 : N$ **do**
  **for** $k = 4 : 4 : 64$ **do**
    $neighbor \leftarrow KNN(x_i)(x_i \in X)$;
    $\boldsymbol{M} \leftarrow \frac{1}{|neighbor|}\sum_{j=1}^{|neighbor|}(neighbor_j - \bar{x})(neighbor_j - \bar{x})^{\mathrm{T}}$ ; /* $\bar{x}$ is the neighborhood centroid coordinates of the current point. */
    $\lambda_1, \lambda_2, \lambda_3 \leftarrow SVD(\boldsymbol{M})$ ; /* SVD is the eigenvalue decomposition. */
    Compute the spatial distribution features.

$$a_{1D} = \frac{\sqrt{\lambda_1} - \sqrt{\lambda_2}}{\sqrt{\lambda_1}}, a_{2D} = \frac{\sqrt{\lambda_2} - \sqrt{\lambda_3}}{\sqrt{\lambda_1}}, a_{3D} = \frac{\sqrt{\lambda_3}}{\sqrt{\lambda_1}}.$$

    Compute the local information entropy of current point $x_i$ when its neighborhood size is $k$.

$$H_i(k) = -a_{1D}\ln(a_{1D}) - a_{2D}\ln(a_{2D}) - a_{3D}\ln(a_{3D}).$$

  **end**
**end**
The three optimal neighborhood ranges $k_{op}^l (l = 1, 2, 3)$ are obtained from the local information entropy of all points.

$$k_{op}^l \leftarrow \underset{k=\{4:4:64\}}{\arg\min^3}\left\{\max_{k=\{4:4:64\}}\{H_i(k)\}, i = 1, 2, \cdots N\right\}$$

where $\underset{k}{\arg\min^3} H(k)$ denotes finding the three dependent variables $k$ that obtain the minimum, the second minimum and the third minimum values of the function $H(k)$.
Construct three multi-scale Gram matrices $G_k(X)(k \in \{k_{op}^1, k_{op}^2, k_{op}^3\})$.

## Appendix 6

For a point cloud model $X$, the noise point $\hat{x}_i$ is generated by adding Gaussian noise $Y \sim \mathcal{N}(0, \sigma)$ to any point $x_i$: $\hat{x}_i = Y + x_i$, as shown in Fig. 3.

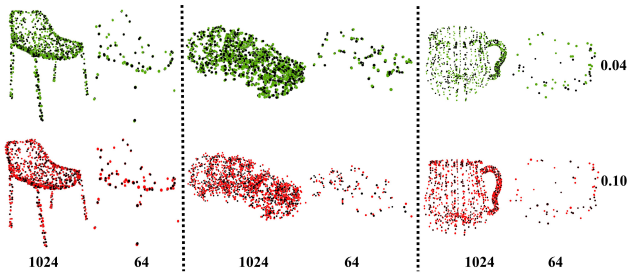



Figure 3. The noise injection effect of the dense point cloud (1024 points) and sparse point cloud (64 points) with different Gaussian noise parameters, where black points are the original input point cloud, green points are the noise injection results when the standard deviation is 0.04, and red points are the noise injection results when the standard deviation is 0.1.

## Appendix 7

Tab. 2 provides a comparison of the parameter size and inference times of various network models, with a list of network design considerations. Most methods consider only one or several states of point clouds for the network design, with fewer network parameters and shorter inference times. For example, the inference time of DGCNN with STN [19] increases to $22.7ms$, and the inference speed drops to $0.08M/ms$ compared with DGCNN without STN. Considering the generalization ability of complex point clouds, our NLGAT is designed with many matrix calculation and feature extraction modules, resulting in more network parameters and a longer reasoning time. Although the parameter complexity of NLGAT is high under the current hardware devices, the inference speed of NLGAT is competitive with other methods on multiple states.

## Appendix 8

**Classification Results for Point Clouds with Noise.** Fig. 4 shows the best classification result achieved in the ModelNet40 dataset (with 1024 points) when the Gaussian noise

| Model | Network design considerations | Params $(M)$ | Time $(ms)$ | Speed $(M/ms)$ |
|---|---|---|---|---|
| MVCNN [17] | U | 60.00 | 45.00 | 1.33 |
| PointNet [13] | U | 3.50 | 2.10 | 1.67 |
| PointNet + (MSG)(PointNet++) [14] | U | 1.70 | 192.00 | 0.01 |
| DGCNN(without STN) [19] | U | 1.80 | 5.30 | 0.34 |
| DGCNN [19] | U, R | 1.80 | 22.70 | 0.08 |
| PCA-RI [22] | U, R | 4.20 | 35.00 | 0.12 |
| GLR-Net [28] | U, R | 1.50 | 29.00 | 0.05 |
| ERI-Net [4] | U, R | 1.50 | 5.70 | 0.26 |
| Triangle-Net [21] | U, R, S, N | 2.00 | 6.90 | 0.29 |
| NLGAT | U, R, S, N | 62.60 | 183.10 | 0.34 |

Table 2. Model size and inference time (where four letters in network design considerations indicate the states of point clouds, *i.e.*, unorderedness (U), rotation(R), sparsity(S), noise(N)).

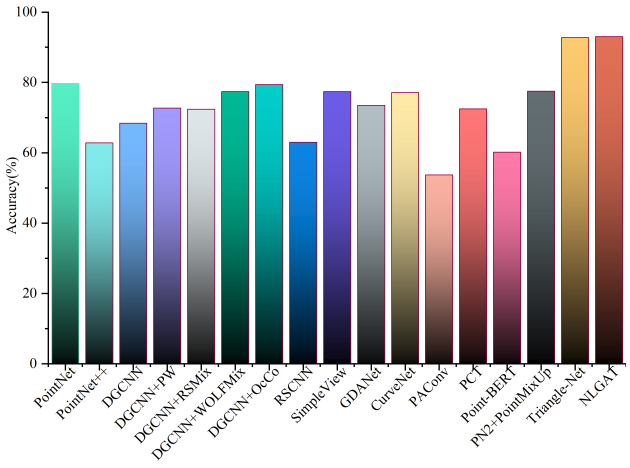



Figure 4. The classification results of different methods in the ModelNet40 dataset with Gaussian noise.

parameter varies from 0.01 to 0.05. It can be seen that the classification results of most methods [2, 3, 5, 7, 8, 10, 13, 14, 16, 18–20, 23, 24, 27] are below 80%, while Triangle-Net [21] and our NLGAT achieve a classification performance above 90%.

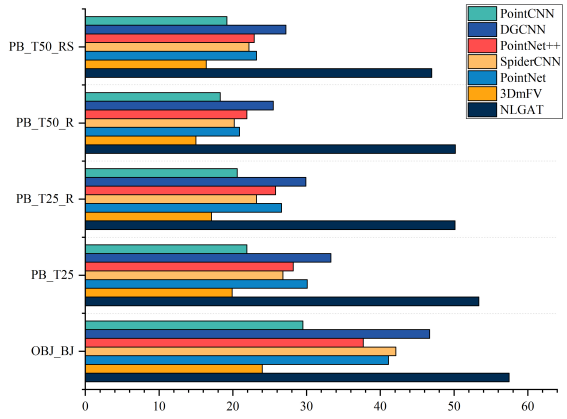
## Appendix 9

**Training on CAD and Testing on ScanObjectNN.** Fig. 5a shows that the classification results of most methods [1, 9, 13, 14, 19, 26] on five subsets(including background points) of ScanObjectNN are only about 30%, while the average result of our method is 51.62%. Although the average classification results of NLGAT on various subsets of ScanObjectNN are improved by 21.62%, this result (57.5% obtained on the OBJ_BJ subset) is not high compared with the results trained and tested on the CAD dataset (94.2% obtained on the ModelNet40). In conclusion, although NLGAT has a better generalization ability than the other methods in the current training mode, the data mapping relationship between the training dataset and the test dataset significantly impacts the network training.

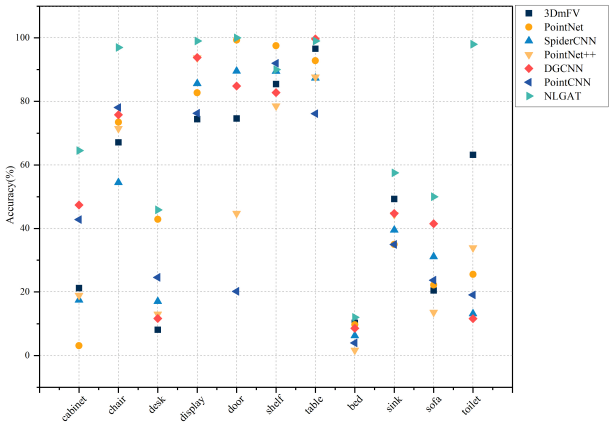
**Training on ScanObjectNN and Testing on CAD.** Fig. 5b provides the results on the 11 corresponding categories between two datasets to verify the above conclusion. Compared with the results of subset PB_T50_RS in Fig. 5a, the classification results of NLGAT are improved by 26.9% in the current training mode, and some categories can even reach an accuracy of over 90%. The average accuracy of NLGAT is 73.9%, which is 19.14% higher than the second-best result (DGCNN: 54.76%). These experimental results support our previous experimental conclusion that classification results would be improved when there is a more relevant mapping relationship between the data in the training and test datasets. Therefore, we train and test the real-world dataset ScanObjectNN to validate the network classification ability of NLGAT further.

## Appendix 10

**Comparison of Six Subsets of ScanObjectNN.** Fig. 6 shows the classification results in the dataset ScanObjectNN with different subsets. It can be seen that the classification accuracy of NLGAT is significantly improved compared with other methods, and the classification results are above 88% of all datasets. In particular, it reaches 95.0% on the dataset OBJ_ONLY and 94.2% on the dataset PB_T25. In addition, our result is 91.8% on the dataset PB_T25_R, which outperforms the PointCNN method (82.5%) by 9.3%, and the result (88.8%) by NLGAT has an 8.8% improvement compared with the result (80.0%) of DGCNN on the dataset PB_T50_R. Note that the classification results in the harshest test scenario PB_T50_RS are decreased compared with the dataset OBJ_BJ. DGCNN has only a 78.1% accuracy on the dataset PB_T50_RS and an 82.8% accuracy




(a) Mode1: Training on ModelNet40 and Testing on ScanObjectNN.




(b) Mode2: Training on ScanObjectNN and Testing on ModelNet40.

Figure 5. Comparison of generalization ability of the network based on the classification accuracy (unit: %).

on the dataset OBJ_BJ. NLGAT has an 88.4% accuracy on the dataset PB_T50_RS, which is a 3.8% reduction compared with the result (92.2%) on the dataset OBJ_BJ.

**Comparison of Subset PB_T50_RS of ScanObjectNN.** Tab. 3 shows the overall classification accuracy(OA) results of some SOTA methods on the dataset PB_T50_RS. Experiments have shown that our result (88.4%) is superior to the results of PointMLP (85.4%) [11] and PointNeXt (87.7%) [15].

| Methods | OA(%) |
|---|---|
| PointNet [13] | 68.2 |
| PointNet++ [14] | 77.9 |
| DGCNN [19] | 78.1 |
| PointMLP [11] | 85.4 |
| PointNetXt [15] | 87.7 |
| NLGAT | 88.4 |

Table 3. Comparison of overall classification accuracy (unit: %) in Subset PB_T50_RS of ScanObjectNN.

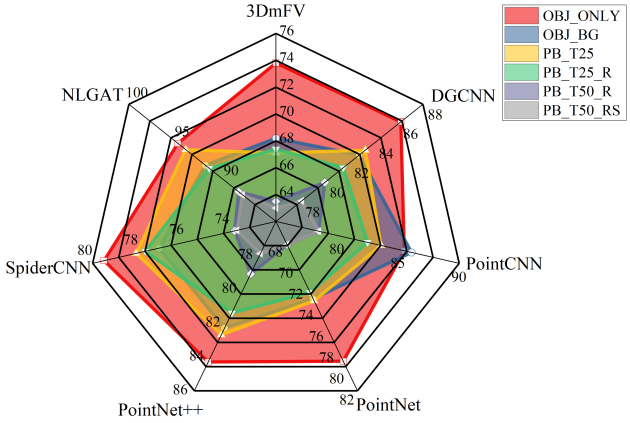



Figure 6. Comparison of overall classification accuracy (unit: %) when training and testing are done on ScanObjectNN, where the different colors correspond to six subsets, each axis represents the methods, and the axis coordinates are counted separately.

## References


[1] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, 3(4):3145–3152, 2018. 4

[2] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees GM Snoek. Pointmixup: Augmentation for point clouds. In *Proceedings of the European Conference on Computer Vision*, pages 330–345, 2020. 4

[3] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *International Conference on Machine Learning*, pages 3809–3820, 2021. 4

[4] Ruibin Gu, Qiuxia Wu, Wing WY Ng, Hongbin Xu, and Zhiyong Wang. Erinet: Enhanced rotation-invariant network for point cloud classification. *Pattern Recognition Letters*, 151:180–186, 2021. 4

[5] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. 4

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, page 1025–1035, 2017. 2

[7] Sihyeon Kim, Sanghyeok Lee, Dasol Hwang, Jaewon Lee, Seong Jae Hwang, and Hyunwoo J Kim. Point cloud augmentation with weighted local transformations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 548–557, 2021. 4

[8] 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 *Proceedings of the IEEE/CVF Conference on Computer*

*Vision and Pattern Recognition*, pages 15900–15909, 2021. 4

[9] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 1–11, 2018. 4

[10] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong 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. 4

[11] 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 *International Conference on Learning Representations*, 2022. 5

[12] Thomas Pumir, Amit Singer, and Nicolas Boumal. The generalized orthogonal procrustes problem in the high noise regime. *Information and Inference: A Journal of the IMA*, 10(3):921–954, 2021. 1

[13] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 4, 5

[14] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. 30:5099–5108, 2017. 4, 5

[15] 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 *Advances in Neural Information Processing Systems*, 2022. 5

[16] Jiawei Ren, Liang Pan, and Ziwei Liu. Benchmarking and analyzing point cloud classification under corruptions. *arXiv preprint arXiv:2202.03377*, 2022. 4

[17] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. pages 945–953, 2015. 4

[18] Hanchen Wang, Qi Liu, Xiangyu Yue, Joan Lasenby, and Matt J Kusner. Unsupervised point cloud pre-training via occlusion completion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9782–9792, 2021. 4

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

[20] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 915–924, 2021. 4

[21] Chenxi Xiao and Juan Wachs. Triangle-net: Towards robustness in point cloud learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 826–835, 2021. 4

[22] Zelin Xiao, Hongxin Lin, Renjie Li, Lishuai Geng, Hongyang Chao, and Shengyong Ding. Endowing deep 3d models with rotation invariance based on principal component analysis. In *IEEE International Conference on Multimedia and Expo*, pages 1–6, 2020. 4

[23] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3173–3182, 2021. 4

[24] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3056–3064, 2021. 4

[25] Mingye Xu, Zhipeng Zhou, and Yu Qiao. Geometry sharing network for 3d point cloud classification and segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12500–12507, 2020. 1

[26] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision*, pages 87–102, 2018. 4

[27] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19313–19322, 2022. 4

[28] Chen Zhao, Jiaqi Yang, Xin Xiong, Angfan Zhu, Zhiguo Cao, and Xin Li. Rotation invariant point cloud analysis: where local geometry meets global topology. *Pattern Recognition*, 127:1–11, 2022. 4