# Self-positioning Point-based Transformer for Point Cloud Understanding (Supplement)

Jinyoung Park[1*], Sanghyeok Lee[1*], Sihyeon Kim[1], Yunyang Xiong[2], Hyunwoo J. Kim[1†]
[1]Korea University, [2]Meta Reality Labs
{lpmn678, cat0626, sh_bs15, hyunwoojkim}@korea.ac.kr
yunyang@fb.com

In this supplementary material, we provide implementation details (Appendix A), and describe further experimental results (Appendix B).

## A. Implementation details

In this section, we describe the implementation details of our experiments. We implement our Self-positioning Point-based Transformer (SPoTr) using Pytorch [1] and OpenPoints [2], which is a library for dealing with point cloud data. For training SPoTr, we use 1,024 points for shape classification (ScanObjectNN dataset [3]), 2,048 points for part segmentation (SP-Part dataset [4, 5]), and 40K points for scene segmentation (S3DIS dataset [6]) as the input.

### A.1. Shape classification

For shape classification, we train our models with a batch size of 32 for 250 epochs and AdamW optimizer [7] using NVIDIA RTX Titan 24GB GPU. We use CrossEntropy loss with label smoothing following existing methods [8]. We opt 0.002 for the initial learning rate, which is decayed by the cosine annealing strategy [9]. During training, each sample is scaled in a range of [0.9, 1.1]. We stack 4 SPoTr blocks ($l = 0$) and the dimensionality is doubled for each block. The detailed architecture used for shape classification is described in Table 1.

| | stage 0 (MLP) | stage 1 (SPoTr Block) | stage 2 (SPoTr Block) | stage 3 (SPoTr Block) | stage 4 (SPoTr Block) |
|---|---|---|---|---|---|
| shape classification | dim : 48 | dim : 96<br>$l$ : 0<br>SP points : 24<br>$\gamma$ : 16<br>$\tau$ : 0.1 | dim: 192<br>$l$ : 0<br>SP points : 24<br>$\gamma$ : 16<br>$\tau$ : 0.1 | dim: 384<br>$l$ : 0<br>SP points : 24<br>$\gamma$ : 16<br>$\tau$ : 0.1 | dim: 768<br>$l$ : 0<br>SP points : 24<br>$\gamma$ : 16<br>$\tau$ : 0.1 |

Table 1. **Detailed architecture specifications for shape classification.**

### A.2. Part segmentation

For part segmentation, we train our models for 150 epochs with a batch size of 4 per GPU with 8 GPUs using NVIDIA RTX Titan 24GB GPU. We opt AdamW optimizer [7] with an initial learning rate of 0.001 and a learning rate decay at [90, 120] epochs with a decay rate of 0.5. We use point cloud scaling, jittering, and height appending following [10] for data augmentation. Following previous works [11], we apply a U-net designed architecture with SPoTr block and Feature propagation block for segmentation. The details of our model for part segmentation are described in Table 2.

*First two authors have equal contribution.
†is the corresponding author.

| | stage 0 (MLP) | stage 1 (SPoTr Block) | stage 2 (SPoTr Block) | stage 3 (SPoTr Block) | stage 4 (SPoTr Block) |
|---|---|---|---|---|---|
| shape part segmentation | dim : 128 | dim: 256<br>$l$ : 0<br>SP points : 16<br>$\gamma$ : 16<br>$\tau$ : 0.1 | dim: 512<br>$l$ : 0<br>SP points : 16<br>$\gamma$ : 16<br>$\tau$ : 0.1 | dim : 1024<br>$l$ : 0<br>SP points : 16<br>$\gamma$ : 16<br>$\tau$ : 0.1 | dim : 2048<br>$l$ : 0<br>SP points : 16<br>$\gamma$ : 16<br>$\tau$ : 0.1 |

Table 2. **Detailed architecture for shape part segmentation.**

### A.3. Scene segmentation

For scene segmentation, we train our models for 100 epochs with a batch size of 2 per GPU with 4 GPUs using NVIDIA Tesla V100 32GB GPU. The model is trained by CrossEntropy loss with label smoothing. We use the initial learning rate of 0.01, which is decayed by the cosine annealing scheme [9]. We use point cloud scaling, jittering, rotation, color drop, and height appending following [10] for data augmentation. Similar to part segmentation, we apply a U-net designed architecture with SPoTr block and Feature propagation block for scene segmentation. The details of our model for scene segmentation are described in Table 3.

| | stage 0 (MLP) | stage 1 (SPoTr Block) | stage 2 (SPoTr Block) | stage 3 (SPoTr Block) | stage 4 (SPoTr Block) |
|---|---|---|---|---|---|
| scene segmentation | dim : 64 | dim: 128<br>$l$ : 4<br>SP points : 16<br>$\gamma$ : 16<br>$\tau$ : 0.5 | dim: 256<br>$l$ : 4<br>SP points : 16<br>$\gamma$ : 16<br>$\tau$ : 0.5 | dim : 512<br>$l$ : 4<br>SP points : 16<br>$\gamma$ : 16<br>$\tau$ : 0.5 | dim : 1024<br>$l$ : 4<br>SP points : 16<br>$\gamma$ : 16<br>$\tau$ : 0.5 |

Table 3. **Detailed architecture for scene segmentation.**

## B. Further experimental results

### B.1. Detailed semantic segmentation results

In this section, we provide more detailed semantic segmentation results on SN-Part [4] and S3DIS [6], respectively. For category-wise mIoU, we present Table 4 for SN-Part and Table 5 for S3DIS.

### B.2. Sensitivity analysis for self-positioning points (SP points)

To analyze the sensitivity of SPoTr to the number of Self-Positioning points (SP points), we compare the performance of the model with the diverse number of SP points in Table 6. We empirically found that 24 SP points are most effective for our SPoTr architecture on SONN dataset.

### B.3. Efficiency comparison

To investigate the efficiency of SPoTr, we compare our method with recent baselines [20, 24] on ScanObjectNN [3]. Specifically, we use DeepSpeed [25] library as a profiler for calculating the number of parameters and FLOPS during inference. For comparison, we also opt a light version of SPoTr (SPoTr*), where the channel size of each layer is 2/3. The results are summarized in Table 7. It is worth noting that SPoTr achieves the best performance (88.6%) with fewer parameters of 3.3(M) than 13.2(M) of PointMLP and 6.8(M) of RepSurf. Further, although SPoTr* only requires 1.6(M) parameters and 5.5 GFLOPS, it still shows significant gains over the previous best methods (+2.2%). In short, we demonstrate that SPoTr is a computation-efficient and memory-efficient method.

| Method | mIoU | air plane | bag | cap | car | chair | ear phone | guitar | knife | lamp | laptop | motor bike | mug | pistol | rocket | skate board | table |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PointNet [12] | 83.7 | 83.4 | 78.7 | 82.5 | 74.9 | 89.6 | 73.0 | 91.5 | 85.9 | 80.8 | 95.3 | 65.2 | 93.0 | 81.2 | 57.9 | 72.8 | 80.6 |
| PointNet++ [11] | 85.1 | 82.4 | 79.0 | 87.7 | 77.3 | 90.8 | 71.8 | 91.0 | 85.9 | 83.7 | 95.3 | 71.6 | 94.1 | 81.3 | 58.7 | 76.4 | 82.6 |
| PointCNN [13] | 86.1 | 84.1 | 86.5 | 86.0 | 80.8 | 90.6 | 79.7 | 92.3 | 88.4 | 85.3 | 96.1 | 77.2 | 95.2 | 84.2 | 64.2 | 80.0 | 83.0 |
| DGCNN [14] | 85.1 | 84.2 | 83.7 | 84.4 | 77.1 | 90.9 | 78.5 | 91.5 | 87.3 | 82.9 | 96.0 | 67.8 | 93.3 | 82.6 | 59.7 | 75.5 | 82.0 |
| RSCNN [8] | 86.2 | 83.5 | 84.8 | 88.8 | 79.6 | 91.2 | 81.1 | 91.6 | 88.4 | 86.0 | 96.0 | 73.7 | 94.1 | 83.4 | 60.5 | 77.7 | 83.6 |
| KPConv [10] | 86.4 | 84.6 | 86.3 | 87.2 | 81.1 | 91.1 | 77.8 | 92.6 | 88.4 | 82.7 | 96.2 | 78.1 | 95.8 | 85.4 | 69.0 | 82.0 | 83.6 |
| PointASNL [15] | 86.1 | 84.1 | 84.7 | 87.9 | 79.7 | 92.2 | 73.7 | 91.0 | 87.2 | 84.2 | 95.8 | 74.4 | 95.2 | 81.0 | 63.0 | 76.3 | 83.2 |
| PCT [16] | 86.4 | 85.0 | 82.4 | 89.0 | 81.2 | 91.9 | 71.5 | 91.3 | 88.1 | 86.3 | 95.8 | 64.6 | 95.8 | 83.6 | 62.2 | 77.6 | 83.7 |
| PAConv [17] | 86.1 | 84.3 | 85.0 | 90.4 | 79.7 | 90.6 | 80.8 | 92.0 | 88.7 | 82.2 | 95.9 | 73.9 | 94.7 | 84.7 | 65.9 | 81.4 | 84.0 |
| AdaptConv [18] | 86.4 | 84.8 | 81.2 | 85.7 | 79.7 | 91.2 | 80.9 | 91.9 | 88.6 | 84.8 | 96.2 | 70.7 | 94.9 | 82.3 | 61.0 | 75.9 | 84.2 |
| CurveNet [19] | 86.8 | 85.1 | 84.1 | 89.4 | 80.8 | 91.9 | 75.2 | 91.8 | 88.7 | 86.3 | 96.3 | 72.8 | 95.4 | 82.7 | 59.8 | 78.5 | 84.1 |
| PointMLP [20] | 86.1 | 83.5 | 83.4 | 87.5 | 80.5 | 90.3 | 78.2 | 92.2 | 88.1 | 82.6 | 96.2 | 77.5 | 95.8 | 85.4 | 64.6 | 83.3 | 84.3 |
| **SPoTr** | **87.2** | 85.8 | 86.9 | 89.3 | 82.2 | 92.0 | 82.4 | 91.8 | 88.6 | 85.7 | 96.2 | 77.6 | 96.3 | 85.3 | 64.0 | 78.0 | 84.1 |

Table 4. **Part segmentation results on SN-Part.**

| Method | mAcc | mIoU | ceiling | floor | wall | beam | column | window | door | table | chair | sofa | bookcase | board | clutter |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PointNet [12] | 49.0 | 41.1 | 88.8 | 97.3 | 69.8 | 0.1 | 3.9 | 46.3 | 10.8 | 59.0 | 52.6 | 5.9 | 40.3 | 26.4 | 33.2 |
| PointCNN [13] | 63.9 | 57.3 | 92.3 | 98.2 | 79.4 | 0.0 | 17.6 | 22.8 | 62.1 | 74.4 | 80.6 | 31.7 | 66.7 | 62.1 | 56.7 |
| PointWeb [21] | 66.6 | 60.3 | 92.0 | 98.5 | 79.4 | 0.0 | 21.1 | 59.7 | 34.8 | 76.3 | 88.3 | 46.9 | 69.3 | 64.9 | 52.5 |
| KPConv [10] | 72.8 | 67.1 | 92.8 | 97.3 | 82.4 | 0.0 | 23.9 | 58.0 | 69.0 | 91.0 | 81.5 | 75.3 | 75.4 | 66.7 | 58.9 |
| PCT [16] | 67.7 | 61.3 | 92.5 | 98.4 | 80.6 | 0.0 | 19.4 | 61.6 | 48.0 | 76.6 | 85.2 | 46.2 | 67.7 | 67.9 | 52.3 |
| CT [22] | - | 67.9 | 94.2 | 97.7 | 82.7 | 0.0 | 34.4 | 62.8 | 68.4 | 89.8 | 80.4 | 78.2 | 61.4 | 67.7 | 64.9 |
| PointTransfomer [23] | - | 70.4 | 94.0 | 98.5 | 86.3 | 0.0 | 38.0 | 63.4 | 74.3 | 89.1 | 82.4 | 74.3 | 80.2 | 76.0 | 59.3 |
| **SPoTr** | **76.4** | **70.8** | 94.4 | 98.5 | 84.0 | 0.0 | 34.4 | 59.9 | 77.4 | 83.2 | 92.5 | 79.8 | 76.4 | 78.5 | 61.4 |

Table 5. **Scene segmentation results on S3DIS.**

| # SP points | 8 | 16 | 24 | 32 |
|---|---|---|---|---|
| Accuracy | 88.0 | 88.1 | 88.6 | 88.4 |

Table 6. **Sensitivity analysis for SP points on SONN.**

| SONN | OA ↑ | Param ↓ (M) | FLOPs ↓ (G) |
|---|---|---|---|
| PointMLP [20] | 85.7 | 13.2 | 31.4 |
| RepSurf [24] | 86.0 | 6.8 | 4.9 |
| SPoTr* | 88.2 | 1.6 | 5.5 |
| SPoTr | **88.6** | 3.3 | 12.3 |

Table 7. **Efficiency comparison on SONN.** SPoTr* is a light version of SPoTr

## B.4. Qualitative results

The visualization of part segmentation with SN-Part is provided in Figure 1. We also provide qualitative results on scene segmentation with S3DIS in Figure 2.

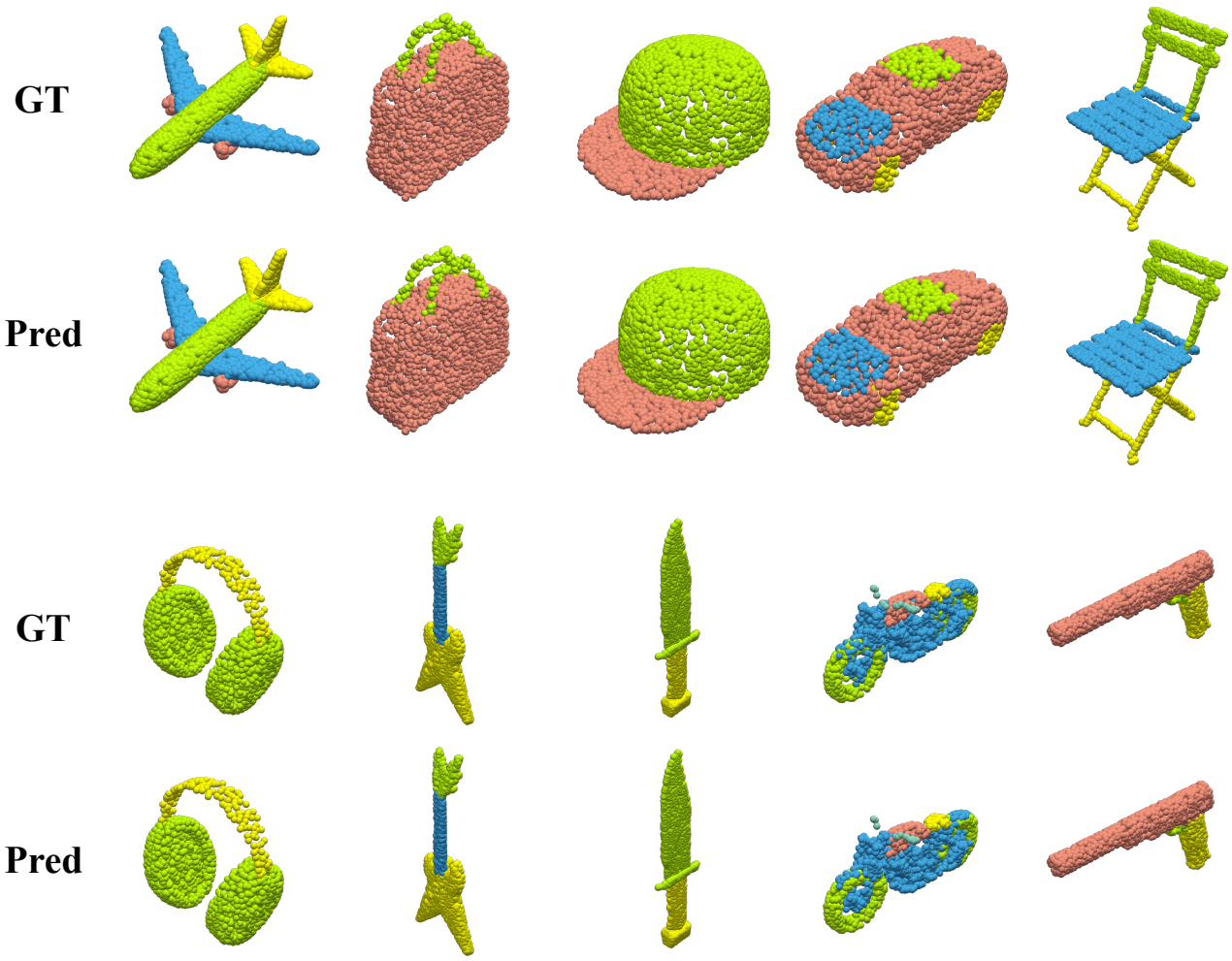


Figure 1. Qualitative results on SP-Part.

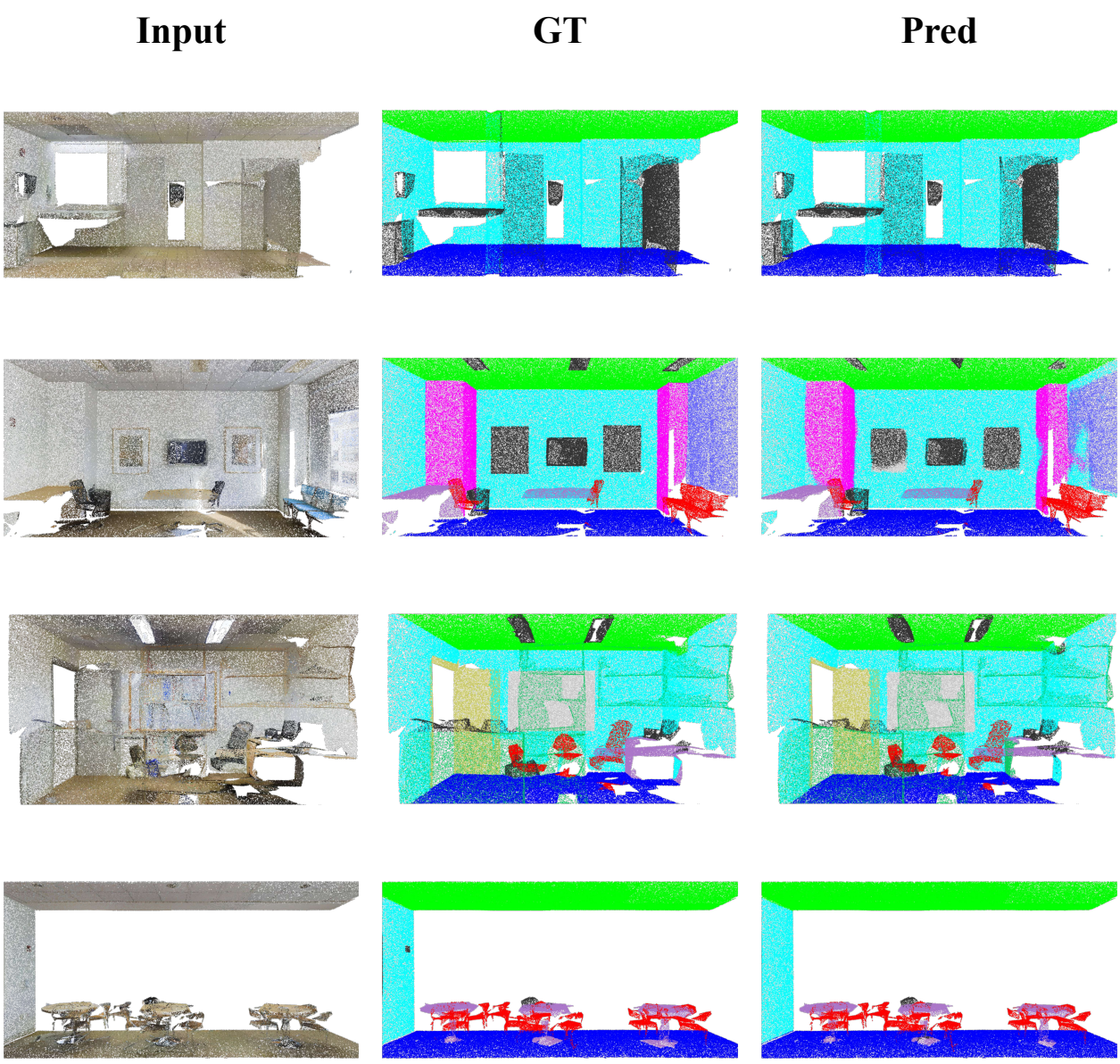


Figure 2. Qualitative results on S3DIS.

## References


[1] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 1

[2] 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 *NeurIPS*, 2022. 1

[3] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, 2019. 1, 2

[4] 3D Warehouse. Sketchup. https://3dwarehouse.sketchup.com/, 2022. 1, 2

[5] Sanghyeok Lee, Minkyu Jeon, Injae Kim, Yunyang Xiong, and Hyunwoo J Kim. Sagemix: Saliency-guided mixup for point clouds. In *NeurIPS*, 2022. 1

[6] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, 2016. 1, 2

[7] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 1

[8] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019. 1, 3

[9] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 1, 2

[10] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. 1, 2, 3

[11] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 1, 3

[12] 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

[13] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, 2018. 3

[14] 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):146:1–146:12, 2019. 3

[15] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *CVPR*, 2020. 3

[16] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *CVM*, 7(2):187–199, 2021. 3

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

[18] Haoran Zhou, Yidan Feng, Mingsheng Fang, Mingqiang Wei, Jing Qin, and Tong Lu. Adaptive graph convolution for point cloud analysis. In *ICCV*, 2021. 3

[19] 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

[20] 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. 2, 3

[21] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019. 3

[22] Kirill Mazur and Victor Lempitsky. Cloud transformers: A universal approach to point cloud processing tasks. In *ICCV*, 2021. 3

[23] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 3

[24] Haoxi Ran, Jun Liu, and Chengjie Wang. Surface representation for point clouds. In *CVPR*, 2022. 2, 3

[25] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *KDD*, 2020. 2