Stratified Transformer for 3D Point Cloud Segmentation Supplementary Material

Introduction

This is the supplementary material, which is divided into the following sections.

1. More experimental results are shown in Sec. 1.
2. We illustrate more ERF [18] visualizations in Sec. 2.
3. More visual comparisons are shown in Sec. 3.
4. The illustration of 3D shifted window is shown in Fig. 2.
5. The illustration of the contextual Relative Position Encoding (cRPE) is shown in Fig. 3.
6. The robustness study on ScanNetv2 is shown in Sec. 4.
7. More detailed memory complexity analysis and the position encoding implementation are given in Sec. 5.
8. More ablation studies are shown in Sec. 6.
9. Introduction of ScanNetv2 [5], S3DIS [1] and ShapeNetPart [2] are given in Sec. 7.
10. Implementation details for experiments on ShapeNet Part is given in Sec. 8.
11. Comparision of FLOPs with previous methods is given in Sec. 9.
12. Limitation analysis and future work are shown in Sec. 10.

1. More Experimental Results

More results on S3DIS and ScanNetv2 datasets are shown in Tables 1 and 2, respectively. We add per-class results on both datasets. Also, we add the result of ScanNetv2 with model ensembling in Table 2. Amazingly, ours achieves 74.7% mIoU in ScanNetv2 benchmark, and outperforms other semantic segmentation methods purely based on 3D points by a large margin.

2. More ERF Visualizations

As shown in Fig. 1, we demonstrate more visualizations of Effective Recetive Field (ERF) [18]. The illustrations are similar to Fig. 1 of the submission file. We follow the definition of ERF in [18] to calculate the gradient of every input point x_i with regard to the feature of interest, *i.e.*, $\frac{\partial y}{\partial x_i}$, which depicts how much the feature changes as input point x_i changes by a small amount. Afterwards, we adopt the colormap `CV2.COLORMAP_JET` for colorization, therefore, red region corresponds to high contribution.

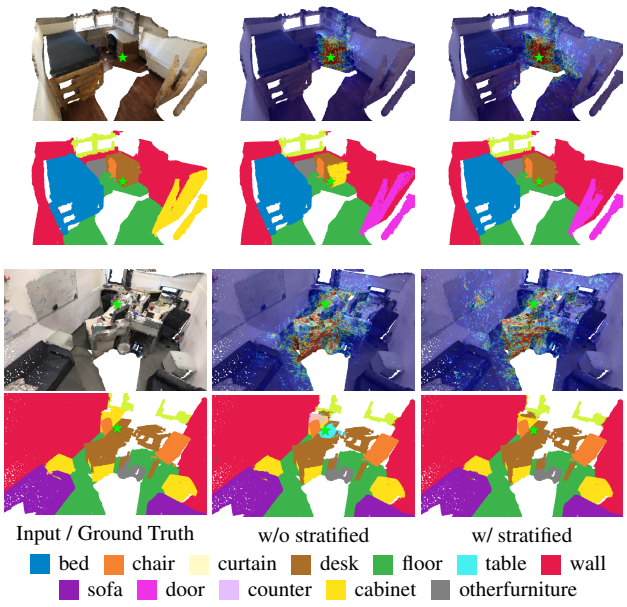


Figure 1. Visualization of Effective Receptive Field (ERF) [18], given the feature of interest (shown with green star) in the output layer. Red region corresponds to high contribution. **Left**: Input point cloud and the ground truth. **Middle**: The ERF and prediction of the model without stratified strategy and by only attending to its own window. **Right**: The ERF and prediction of the model with direct long-range dependency, using the stratified strategy. It reveals the fact that the stratified sampling strategy is able to capture long-range contexts.

Method	Input	OA	mAcc	mIoU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [20]	point	-	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
SegCloud [23]	point	-	57.4	48.9	90.1	96.1	69.9	0.0	18.4	38.4	23.1	70.4	75.9	40.9	58.4	13.0	41.6
TangentConv [22]	point	-	62.2	52.6	90.5	97.7	74.0	0.0	20.7	39.0	31.3	77.5	69.4	57.3	38.5	48.8	39.8
PointCNN [14]	point	85.9	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 [32]	point	87.0	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
HPEIN [11]	point	87.2	68.3	61.9	91.5	98.2	81.4	0.0	23.3	65.3	40.0	75.5	87.7	58.5	67.8	65.6	49.4
GACNet [25]	point	87.8	-	62.9	92.3	98.3	81.9	0.0	20.4	59.1	40.9	85.8	78.5	70.8	61.7	74.7	52.8
PAT [30]	point	-	70.8	60.1	93.0	98.5	72.3	**1.0**	41.5	85.1	38.2	57.7	83.6	48.1	67.0	61.3	33.6
ParamConv [26]	point	-	67.0	58.3	92.3	96.2	75.9	0.3	6.0	**69.5**	63.5	66.9	65.6	47.3	68.9	59.1	46.2
SPGraph [12]	point	86.4	66.5	58.0	89.4	96.9	78.1	0.0	42.8	48.9	61.6	84.7	75.4	69.8	52.6	2.1	52.2
SegGCN [13]	point	88.2	70.4	63.6	93.7	98.6	80.6	0.0	28.5	42.6	74.5	88.7	80.9	71.3	69.0	44.4	54.3
MinkowskiNet [4]	voxel	-	71.7	65.4	91.8	**98.7**	86.2	0.0	34.1	48.9	62.4	81.6	**89.8**	47.2	74.9	74.4	58.6
PAConv [28]	point	-	73.0	66.6	94.6	98.6	82.4	0.0	26.4	58.0	60.0	89.7	80.4	74.3	69.8	73.5	57.7
KPConv [24]	point	-	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
PointTransformer [33]	point	90.8	76.5	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
Ours	point	**91.5**	**78.1**	**72.0**	**96.2**	**98.7**	85.6	0.0	**46.1**	60.0	**76.8**	**92.6**	84.5	**77.8**	75.2	**78.1**	**64.0**

Table 1. More results on S3DIS Area5 for semantic segmentation.

Method	Input	Val mIoU	Test mIoU	bath	bed	bksf	cab	chair	cntr	curt	desk	door	floor	othr	pic	ref	show	sink	sofa	tab	toil	wall	wind
PointNet++ [21]	point	53.5	55.7	73.5	66.1	68.6	49.1	74.4	39.2	53.9	45.1	37.5	94.6	37.6	20.5	40.3	35.6	55.3	64.3	49.7	82.4	75.6	51.5
3DMV [6]	point	-	48.4	48.4	53.8	64.3	42.4	60.6	31.0	57.4	43.3	37.8	79.6	30.1	21.4	53.7	20.8	47.2	50.7	41.3	69.3	60.2	53.9
PanopticFusion [19]	point	-	52.9	49.1	68.8	60.4	38.6	63.2	22.5	70.5	43.4	29.3	81.5	34.8	24.1	49.9	66.9	50.7	64.9	44.2	79.6	60.2	56.1
PointCNN [14]	point	-	45.8	57.7	61.1	35.6	32.1	71.5	29.9	37.6	32.8	31.9	94.4	28.5	16.4	21.6	22.9	48.4	54.5	45.6	75.5	70.9	47.5
PointConv [27]	point	61.0	66.6	78.1	75.9	69.9	64.4	82.2	47.5	77.9	56.4	50.4	95.3	42.8	20.3	58.6	75.4	66.1	75.3	58.8	90.2	81.3	64.2
JointPointBased [3]	point	69.2	63.4	61.4	77.8	66.7	63.3	82.5	42.0	80.4	46.7	56.1	95.1	49.4	29.1	56.6	45.8	57.9	76.4	55.9	83.8	81.4	59.8
PointASNL [29]	point	63.5	66.6	70.3	78.1	75.1	65.5	83.0	47.1	76.9	47.4	53.7	95.1	47.5	27.9	63.5	69.8	67.5	75.1	55.3	81.6	80.6	70.3
SegGCN [13]	point	-	58.9	83.3	73.1	53.9	51.4	78.9	44.8	46.7	57.3	48.4	93.6	39.6	6.1	50.1	50.7	59.4	70.0	56.3	87.4	77.1	49.3
RandLA-Net [9]	point	-	64.5	77.8	73.1	69.9	57.7	82.9	44.6	73.6	47.7	52.3	94.5	45.4	26.9	48.4	74.9	61.8	73.8	59.9	82.7	79.2	62.1
KPConv (*rigid*) [24]	point	-	68.4	84.7	75.8	78.4	64.7	81.4	47.3	77.2	60.5	59.4	93.5	45.0	18.1	58.7	80.5	69.0	78.5	61.4	88.2	81.9	63.2
JSENet [10]	point	-	69.9	88.1	76.2	82.1	66.7	80.0	52.2	79.2	61.3	60.7	93.5	49.2	20.5	57.6	85.3	69.1	75.8	65.2	87.2	82.8	64.9
SparseConvNet [8]	voxel	69.3	72.5	64.7	**82.1**	**84.6**	72.1	**86.9**	53.3	75.4	60.3	61.4	**95.5**	**57.2**	32.5	71.0	87.0	72.4	**82.3**	62.8	**93.4**	**86.5**	68.3
MinkowskiNet [4]	voxel	72.2	73.6	85.9	81.8	83.2	70.9	84.0	52.1	**85.3**	66.0	**64.3**	95.1	54.4	28.6	73.1	**89.3**	67.5	77.2	**68.3**	87.4	85.2	**72.7**
FusionNet [31]	point	-	68.8	70.4	74.1	75.4	65.6	82.9	50.1	74.1	60.9	54.8	95.0	52.2	**37.1**	63.3	75.6	71.5	77.1	62.3	86.1	81.4	65.8
Ours	point	**74.3**	**73.7**	**92.2**	77.7	83.1	**72.7**	83.3	**54.5**	82.0	**68.8**	62.4	95.4	52.7	27.4	**75.2**	85.7	**72.6**	77.8	66.2	88.8	85.0	71.2
Ours (*ensemble*)	point	**74.8**	**74.7**	90.1	80.3	84.5	**75.7**	84.6	51.2	82.5	**69.6**	**64.5**	**95.6**	**57.6**	26.2	74.4	86.1	**74.2**	77.0	**70.5**	89.9	86.0	**73.4**

Table 2. More results on ScanNetv2 for semantic segmentation.

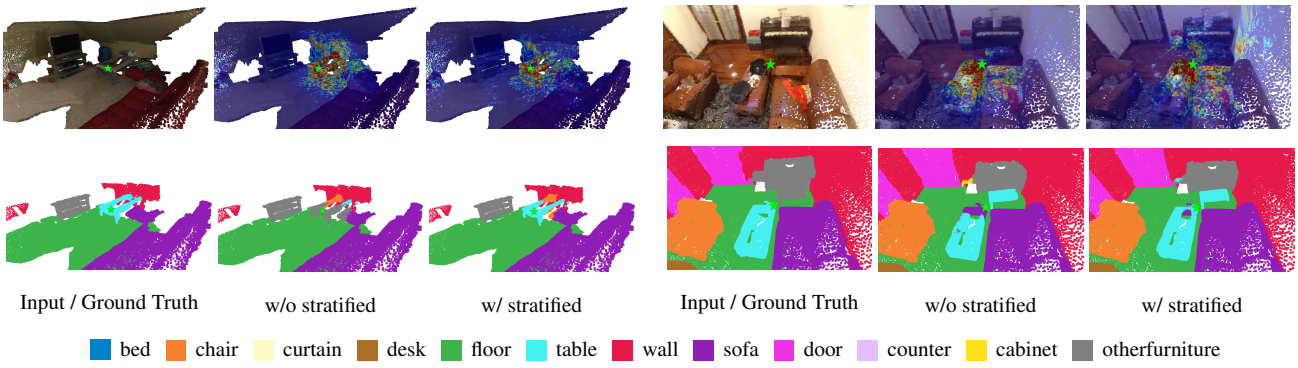


Figure 1. Visualization of Effective Receptive Field (ERF) (Cont.)

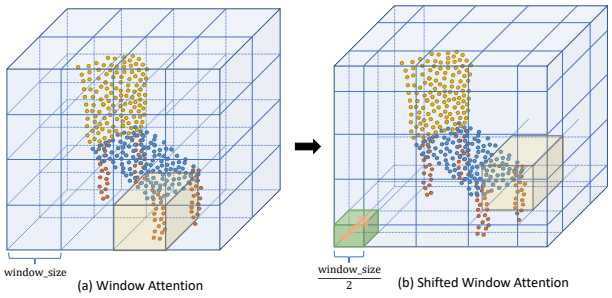


Figure 2. Illustration of the shifted window operation in 3D. (a) Window attention is performed. (b) The window is shifted in the successive Transformer block.

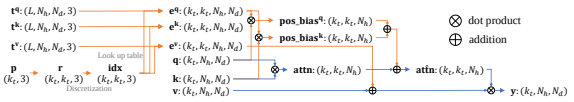


Figure 3. Illustration of contextual Relative Position Encoding.

Method	None	Perm.	90°	180°	270°	+0.2	−0.2	×0.8	×1.2	jitter
Minkowski	72.22	72.23	71.92	71.97	72.00	71.99	72.03	69.92	70.86	**71.43**
Ours	**73.90**	**74.09**	**74.00**	**73.96**	**73.79**	**73.98**	**73.95**	**73.26**	**72.72**	69.49

Table 3. Robustness study on ScanNetv2.

3. More Visual Comparisons

As shown in Fig. 4, we show more visual comparisons with MinkowskiNet [4] on ScanNetv2 validation set. As highlighted with the yellow box, ours is able to accurately recognize the objects, while the MinkowskiNet fails, which visually demonstrates the superiority of our method.

4. Robustness Study on ScanNetv2

As shown in Table 3, we evaluate the robustness of our method and also MinkowskiNet [4] on ScanNetv2 with different perturbations in testing, including permutation, rotation, shift, scale, and jitter. Obviously, ours is more robust when encountering most perturbations, especially with scaling. As for jitter, ours drops too much, because we do not adopt random jitter as data augmentation during training.

5. Memory Complexity Analysis and Position Encoding Implementation

Our implementation. As mentioned in Sec.4 of the submission file, we pre-compute all the query-key pairs that need dot product. Concretely, we use two indices $\mathbf{index}_q, \mathbf{index}_k \in \mathcal{R}^M$ to refer to the query and key features, respectively. Hence, we yield the attention map $\mathbf{attn} \in \mathcal{R}^{M \times N_h}$ via dot product. Without loss of generality, we discuss the memory complexity analysis based on the vanilla version Transformer Block for simplicity. Since each query only attends to the keys within its window, we have a total of $\sum_{i=1}^{n} k_i^2$ query-key pairs that need dot product, where n denotes the number of non-empty windows and k_i denotes the number of points within the i-th window. Therefore, the memory complexity is formulated as

$$O(M \cdot N_h) = O(\sum_{i=1}^{n} k_i^2 \cdot N_h)$$

Vanilla padding-based implementation. The vanilla implementation pads the keys to the maximum number over all windows with dummy tokens for each query point. Therefore, the complexity would be $O(\sum_{i=1}^{n} k_i \cdot k_{max} \cdot N_h)$, where $k_{max} = \max_{i=1}^{n} k_i$ denotes the maximum number of points over all windows.

Comparing the above two methods, since $k_i << k_{max}$ for most windows practically, we have

$$O(M \cdot N_h) = O(\sum_{i=1}^{n} k_i^2 \cdot N_h) << O(\sum_{i=1}^{n} k_i \cdot k_{max} \cdot N_h)$$

which implies that our memory-efficient implementation is able to save large amount of memory.

Incorporate position encoding. By incorporating position encoding, we need to add two extra steps, *i.e.*, (1) dot product between the query/key features and the corresponding position encoding to produce the positional bias, (2) addition between the value features and the value position encoding, followed by weighted-sum aggregation.

For the first step, we use the relative position index $\mathbf{idx} \in \mathcal{R}^{k_t \times k_t \times 3}$ to look up to the learnable tables for qeury and key $\mathbf{t}_x^q$ ($\mathbf{t}_x^k$), $\mathbf{t}_y^q$ ($\mathbf{t}_y^k$) and $\mathbf{t}_z^q$ ($\mathbf{t}_z^k$), respectively, and sum up to yield the corresponding position encoding. Then, the position encoding performs dot product with the query/key features to obtain the positional bias. Note that we implement this process within a single CUDA kernel, where we take $\mathbf{idx} \in \mathcal{R}^{k_t \times k_t \times 3}$, $\mathbf{t}_x^q, \mathbf{t}_y^q, \mathbf{t}_z^q \in \mathcal{R}^{L \times N_h \times N_d}$, $\mathbf{q} \in \mathcal{R}^{N \times N_h \times N_d}$, $\mathbf{idx_q} \in \mathcal{R}^M$ as input and output $\mathbf{pos\_bias} \in \mathcal{R}^{M \times N_h}$ for the query features. The calculation is similar for the key features.

As for the second step, we extend the weighted sum process in Fig.7 (c) of the submission file to also incorporate the addition with the value features. Similar to the first step, we also use the relative position index $\mathbf{idx}$ to look up the learnable table for value. This process is also implemented with a single CUDA kernel.

weight decay	0.01	0.001	0.0001
mIoU	72.0	70.3	70.2

Table 4. Ablation study on weight decay evaluated on S3DIS.

Approach	S3DIS	ScanNetv2
grid sampling	69.6	72.3
farthest point sampling (fps)	**72.0**	**73.7**

Table 5. Ablation study on downsample approaches.

scale	4	8	16
mIoU	70.9	72.0	70.5

Table 6. Ablation study on the downsample scales in stratified sampling strategy.

6. More Ablation Studies

In the supplementary material, we show ablation studies on weight decay, downsample approaches and the downsample scale in the stratified sampling strategy as follows.

Large weight decay. In 2D vision Transformer such as ViT [7] and Swin Transformer [17], strong regularization is essential to achieve the best performance. We also use large weight decay in our framework. The ablation on weight decay in Table 4 shows the necessity of large weight decay.

Downsample. For the downsample layer, we compare the farthest point sampling (fps) with the grid sampling that is used in KPConv [24]. As shown in Table 5, we find that fps yields higher performance in both S3DIS and ScanNetv2 datasets. It demonstrates that uniform point distribution is beneficial for feature learning in our framework.

Stratified downsample scale. To further investigate the effect of different downsample scales in the stratified sampling strategy. We evaluate different downsample scales, *i.e.*, 4, 8 and 16 as shown in Table 6. We notice that setting the downsample scale to 8 achieves the best performance. We guess that setting the downsample scale to a lower value may cause too sparse distribution of distant points and thus limited benefits, while setting it to a higher value may lead to slower convergence in training.

7. Datasets Introduction

We use the S3DIS [1] and ScanNetv2 [5] datasets for the semantic segmentation task. Both of them are challenging large-scale indoor scenes datasets. The S3DIS dataset includes 271 rooms in 6 areas from three buildings, and 13 commonly seen categories are annotated. Following the common practice, we use the scenes in Area 5 for testing and others for training. The ScanNetv2 dataset contains 1201, 312 and 100 indoor RGB-D scenes for training, validation and testing, respectively. And it is annotated with semantic labels within 20 categories.

We also evaluated our method on ShapeNet Part [2] for the part segmentation task. It contains 16,881 shapes labeled with 16 class categories and 50 parts categories.

Method	mIoU ↑	FLOPs ↓
KPConv [24]	67.1	2042
PosPool [16]	66.7	2041
PAConv [28]	66.6	1253
Ours	**72.0**	1714

Table 7. FLOPs Comparison on S3DIS.

8. Inplementation Detail for ShapeNet Part

We follow PAConv [28] to adopt the official data splits. We use AdamW optimizer with initial learning rate, weight decay and betas set to 0.003, 0.01 and (0.9, 0.999). We train for 200 epochs with 2 RTX 2080Ti GPUs. The batch size and stratified downsample scale are set to 64 and 8, respectively. We adopt the same voting strategy as previous works [15, 28] in testing.

9. FLOPs Comparison

As shown in Table 7, we demonstrate the FLOPs comparison with previous methods such as KPConv [24], PosPool [16] and PAConv [28]. We follow [28] to take 4096 points as input and calculate the number of floating-point operations. The results show that our method outperforms others by a large margin and also maintains satisfying FLOPs, which demonstrates the superiority of our model.

10. Limitation Analysis and Future Work

Limitation analysis. Our method is able to yield strong performance in large-scale datasets, but may be a sub-optimal solution for small datasets, because less inductive bias is introduced in Transformer-based network.

Also, more advanced data augmentation has not been applied to our model during training, which may limit the performance. However, strong regularization is demonstrated to be crucial in training Transformer-based networks.

Moreover, despite smaller FLOPs, training is relatively slower compared to previous methods due to the lack of us-

age of advanced CUDA functionality such as shared memory.

Future work. In the future, we will extend our framework to many other 3D point cloud tasks such as 3D object detection, instance segmentation, *etc*. Also, to further boost the performance, we will explore advanced data augmentation for training. Besides, we will adopt more advanced CUDA functionality or methods to decrease the high memory throughput, hence accelerating the training process.

References

[1] 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, 4

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv:1512.03012*, 2015. 1, 4

[3] Hung-Yueh Chiang, Yen-Liang Lin, Yueh-Cheng Liu, and Winston H Hsu. A unified point-based framework for 3d segmentation. In *3DV*, 2019. 2

[4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 2, 3, 10

[5] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 1, 4

[6] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *ECCV*, 2018. 2

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 4

[8] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 2

[9] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020. 2

[10] Zeyu Hu, Mingmin Zhen, Xuyang Bai, Hongbo Fu, and Chiew-lan Tai. Jsenet: Joint semantic segmentation and edge detection network for 3d point clouds. In *ECCV*, 2020. 2

[11] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *ICCV*, 2019. 2

[12] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018. 2

[13] Huan Lei, Naveed Akhtar, and Ajmal Mian. Seggcn: Efficient 3d point cloud segmentation with fuzzy spherical kernel. In *CVPR*, 2020. 2

[14] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *NeurIPS*, 2018. 2

[15] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019. 4

[16] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. A closer look at local aggregation operators in point cloud analysis. In *ECCV*, 2020. 4

[17] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, 2021. 4

[18] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *NeurIPS*, 2016. 1

[19] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. In *IROS*, 2019. 2

[20] 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. 2

[21] 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. 2

[22] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *CVPR*, 2018. 2

[23] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *3DV*, 2017. 2

[24] 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. 2, 4

[25] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, 2019. 2

[26] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *CVPR*, 2018. 2

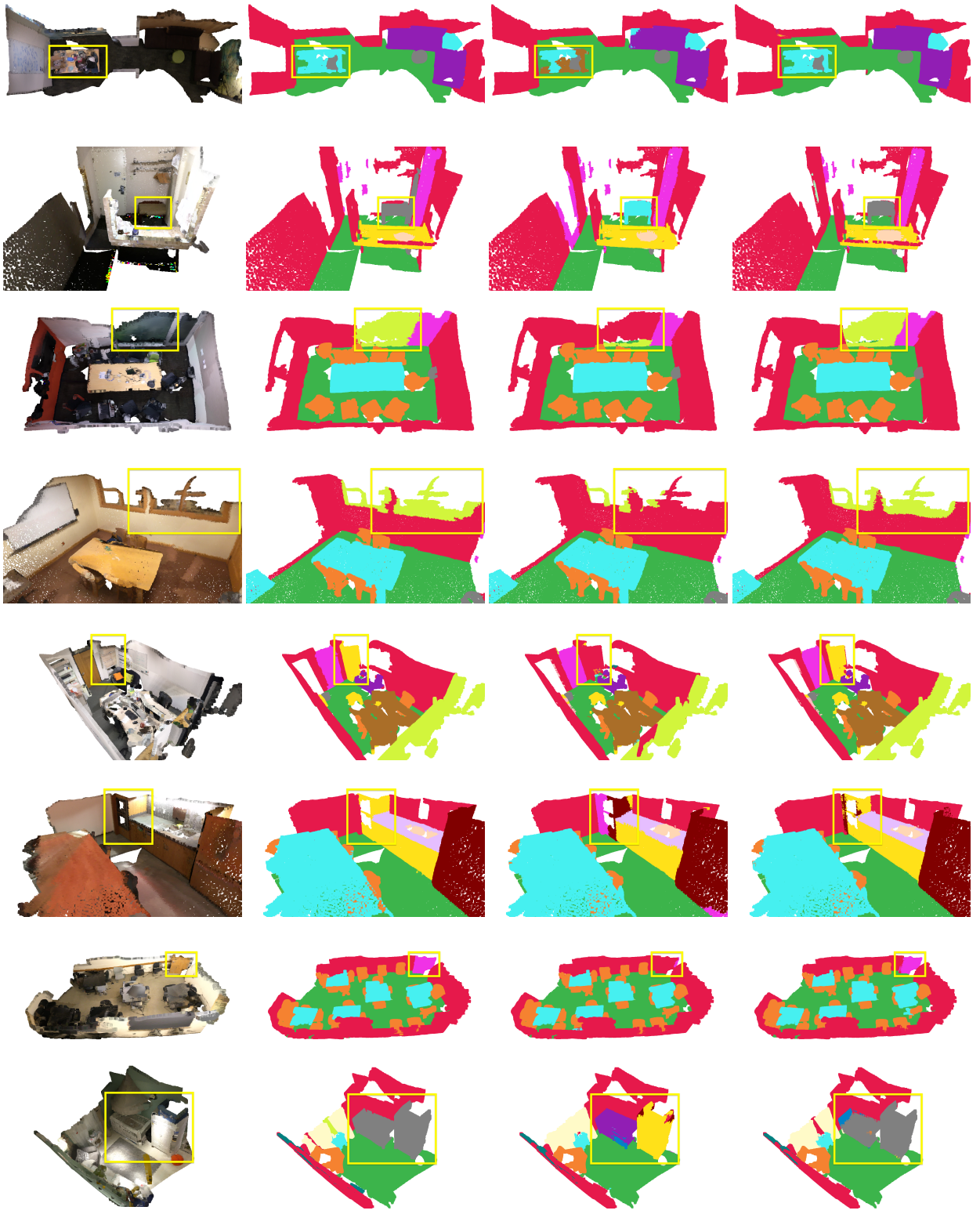
[27] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, 2019. 2

[28] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *CVPR*, 2021. 2, 4

[29] 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. 2


Input
Ground Truth
MinkowskiNet
Ours
bed
chair
curtain
desk
floor
table
wall
sofa
door
counter
cabinet
window
otherfurniture


Input
Ground Truth
MinkowskiNet
Ours
bed
chair
curtain
desk
floor
table
wall
sofa
door
counter
cabinet
window
otherfurniture


Input
Ground Truth
MinkowskiNet
Ours
bed
chair
curtain
desk
floor
table
wall
sofa
door
counter
cabinet
window
otherfurniture


Input
Ground Truth
MinkowskiNet
Ours
bed
chair
curtain
desk
floor
table
wall
sofa
door
counter
cabinet
window
otherfurniture

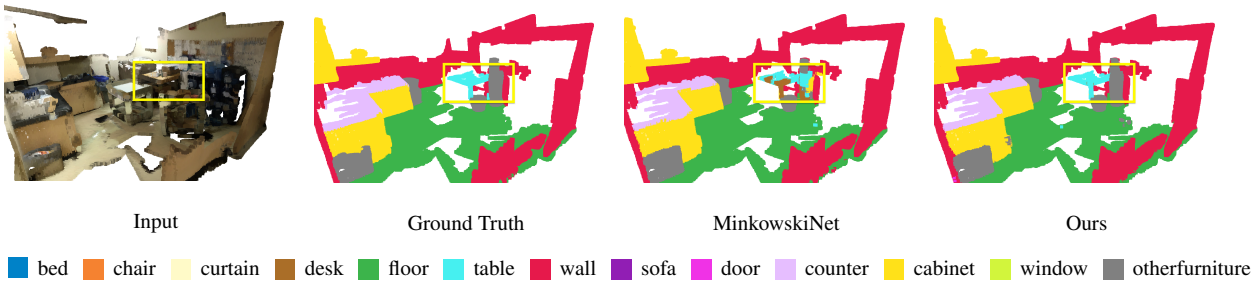


Figure 4. More visual comparisons with MinkowskiNet [4].

Notation	Meaning	Shape
N	input points number.	
k	average input point number within each window.	
k_t	input point number within the t-th window.	
N_h	the number of heads.	
N_d	the dimension of each head.	
N_c	the feature dimension.	
$\mathbf{x}$	MSA input points within the t-th window.	$(k_t, N_h \times N_d)$
$\mathbf{q}$, $\mathbf{k}$, $\mathbf{v}$	MSA query, key, value features.	(k_t, N_h, N_d)
$\mathbf{attn}$	MSA attention map prior to softmax.	(k_t, k_t, N_h)
$\hat{\mathbf{attn}}$	MSA attention map after softmax.	(k_t, k_t, N_h)
$\mathbf{y}$	MSA aggregated features.	(k_t, N_h, N_d)
$\hat{\mathbf{z}}$	MSA output features.	$(k_t, N_h \times N_d)$
s_{win}	the size of cubic window.	
s_{win}^{large}	the size of large cubic window in the stratified Transformer.	
s_{quant}	the quantization size.	
L	the length of learnable look-up tables.	
$\mathbf{p}$	the coordinates for points within the t-th window.	$(k_t, 3)$
$\mathbf{r}$	the relative coordinates for points within the t-th window.	$(k_t, k_t, 3)$
$\mathbf{t}$	learnable look-up tables.	$(L, N_h \times N_d)$
$\mathbf{idx}$	indices of look-up tables for points within the t-th window.	$(k_t, k_t, 3)$
$\mathbf{e}$	relative position encoding for points within the t-th window.	(k_t, k_t, N_h, N_d)
$\mathbf{pos\_bias}$	relative positional bias for points within the t-th window.	(k_t, k_t, N_h)

Table 8. Notation Table.

[30] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *CVPR*, 2019. 2

[31] Feihu Zhang, Jin Fang, Benjamin Wah, and Philip Torr. Deep fusionnet for point cloud semantic segmentation. In *ECCV*, 2020. 2

[32] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019. 2

[33] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 2