Supplementary for "Retro-FPN: Retrospective Feature Pyramid Network for Point Cloud Semantic Segmentation"

Peng Xiang[1*], Xin Wen[2*], Yu-Shen Liu[1†], Hui Zhang[1†], Yi Fang[3], Zhizhong Han[4]
[1]School of Software, Tsinghua University, Beijing, China
[2]JD.com, Beijing, China [3]New York University Abu Dhabi [4]Wayne State University
xiangp23@mails.tsinghua.edu.cn wenxin16@jd.com liuyushen@tsinghua.edu.cn
huizhang@tsinghua.edu.cn yfang@nyu.edu h312h@wayne.edu

In this supplementary material, we provide more details about Retro-FPN.

- We provide more ablation studies to analyze the effect of pyramid height and K-NN in Section 1.
- We provide more network details and experimental settings of Retro-FPN in Section 2.
- We provide the detailed results on the S3DIS [1], ScanNet v2 [4], and SemanticKITTI [2] datasets in Section 3.
- We provide more visualization results of retrospective refinement on the three datasets in Section 4.
- We release part of the source code as part of the supplementary materials.

1. Ablation Studies

In this section, we provide more ablation studies regarding the pyramid height of Retro-FPN and the K-NN search used by the cross-attention block.

1.1. The effect of pyramid height

We analyze the effect of pyramid height and show the results in Table 1. The height represents the number of pyramid layers to conduct retrospective refinement. "Height 4" denotes the last four layers (from the fourth layer to the first layer); "height 1" denotes only the final prediction layer (lowest-level), where retro-transformer degrades to self-attention. The results in Table 1 show that the segmentation performance significantly improves as the height increases, and reaches the top when all pyramid layers (Point Transformer [10] has 5 pyramid levels) are involved. The results demonstrate the importance of feature pyramid, of which the full-scale semantic information should be fully preserved and carefully refined to facilitate the per-point prediction.

Table 1. Effect of pyramid height.

heights	1	2	3	4	5
mIoU	70.7	70.4	71.5	72.1	**73.0**

1.2. The effect of K-NN

As mentioned in the Section 3.3, the refinement of each point is based on the semantic pattern among nearby points from the previous layer. Therefore, Table 2 shows the effect of neighbor number K, and it indicates that the performance of Retro-FPN are significantly improved as the K increases within certain extent ($K \leq 16$). However, as K becomes larger ($K = 32$), the performance will not increase substantially, and may degrade the results. The point distribution of different local regions may vary dramatically, and while searching more neighbour points can lead to a more robust semantic context for complex areas, an oversized neighbourhood could also introduce noise in ambiguous regions.

Table 2. Effect of K-NN.

K	2	4	8	16	32
mIoU	70.9	71.5	72.1	**73.0**	72.5

2. Detailed Settings

2.1. Network settings

There are three important network settings of Retro-FPN, including the segmentation head, K-NN search in the local cross-attention, the random sampling rates and the loss balance weight λ_l.

*Equal contribution.
†Corresponding authors. This work was supported by National Key R&D Program of China (2022YFC3800600), the National Natural Science Foundation of China (62272263, 62072268), and in part by Tsinghua-Kuaishou Institute of Future Media Data.

Table 3. Segmentation head. The negative slope of the Leaky ReLU is 0.1. The head width is also the feature dimension of the point-level semantic features.

Dataset	Method	Activation function	head width
S3DIS [1]	MinkowskiNet (5cm) [3] + Retro-FPN	ReLU	32
	KPConv *rigid* [8] + Retro-FPN	LeakyReLU	32
	KPConv *deform* [8] + Retro-FPN	LeakyReLU	32
	Point Transformer [10] + Retro-FPN	ReLU	32
ScanNet[4]	MinkowskiNet (5cm) [3] + Retro-FPN	ReLU	32
	Point Transformer V2[9] + Retro-FPN	ReLU	48
SemanticKITTI [2]	MinkowskiNet (5cm) [3] + Retro-FPN	ReLU	32

Table 4. K-NN and downsampling rates. The K is the number of neighbors used by the cross-attention, the tuple (12, 12, 16, 16, 12) denotes the K-NN used for five layers. The RS rates is random sampling rates used in each pyramid layer. The tuple (1, 1, 1, 1, 1) denotes that the points (voxels) of all intermediate layers are used by Retro-FPN. And the tuple (1, 4, 4, 4, 1) indicates that points in the 2nd, 3rd, and 4th layer are downsampled to a quarter using random sampling.

Dataset	Method	K	RS rates	λ_l
S3DIS [1]	MinkowskiNet (5cm) [3] + Retro-FPN	12	(1, 1, 1, 1, 1)	1.0
	KPConv *rigid* [8] + Retro-FPN	12	(1, 1, 1, 1, 1)	1.0
	KPConv *deform* [8] + Retro-FPN	16	(1, 1, 1, 1, 1)	1.0
	Point Transformer [10] + Retro-FPN	16	(1, 1, 1, 1, 1)	1.0
ScanNet [4]	MinkowskiNet (5cm) [3] + Retro-FPN	16	(1, 1, 1, 1, 1)	$2.4 - 0.4l$
	Point Transformer V2[9] + Retro-FPN	(12, 12, 16, 16, 12)	(1, 4, 1, 1, 1)	1.0
SemanticKITTI [2]	MinkowskiNet (5cm) [3] + Retro-FPN	12	(1, 4, 4, 4, 1)	1.0

Segmentation head. The segmentation head is an essential component of semantic segmentation networks, which serves to transform the point-level semantic features into per-point labels. In the Point Transformer [10], KPConv [8], and MinkowskiNet [3] backbones, there is only a single segmentation head at the end of the networks, which mainly consists of layers of activation functions and linear transformations. In Retro-FPN, each pyramid level has its own segmentation head. To reduce computation cost, we use a single activation function (Batch Normalization [6] is optional) followed by a linear transformation as the segmentation head (as illustrated in Section 3.2) for Retro-FPN. We keep the activation function the same as the backbone networks, which aims to fully preserve the backbone performance. Moreover, because Retro-FPN focuses on point-level semantic information, a small head width (i.e., dimension C of the point-level semantic features $\mathcal{H}^l$) suffices to characterize the single category information, which also helps to significantly reduce the computation cost. Therefore, we set the head width to 32 for backbones (except Point Transformer V2[9]) across the S3DIS [1], ScanNet V2 [4], and the SemanticKITTI [2] datasets. The detailed settings of the segmentation head are shown in Table 3.

K**-NN**. The local cross-attention in the retro-transformer relies on K-NN to search neighbor points. To balance the trade-off between inference time and performance, we try to keep a small K to search nearest neighbors for all backbones. The detailed settings are shown in Table 4

Random sampling rates. As mentioned in Section 3.4 in our main paper, intermediate layers may contain too many points (voxels) due to small downsampling rates, which leads to substantial computation cost and limited receptive fields. Therefore, we use random sampling to reduce intermediate points (voxels). The random sampling rates (RS rates) are shown in Table 4. Since the Point Transformer [10], KPConv [8], and MinkowskiNet [3] have 5 pyramid layers, we report the RS rates of the five layers. The tuple (1, 1, 1, 1, 1) denotes that the points (voxels) of all intermediate layers are used by Retro-FPN, and the tuple (1, 4, 4, 4, 1) indicates that points in the 2nd, 3rd, and 4th layer are downsampled to a quarter using random sampling.

Loss balance weight. Except the MinkowskiNet backbone on ScanNet v2 dataset, we typically set the loss balance weight to 1.0. For MinkowskiNet backbone on ScanNet, we set $\lambda_l = 2.4 - 0.4l$ so that the information from the last layer (layer 1) plays the key role.

2.2. Experimental settings

We implement our algorithm based on the PyTorch platform. All experiments are conducted on RTX 3090 GPUs. Since we have integrated Retro-FPN with Point Transformer [10], KPConv [8] and MinkowskiNet [3], we separately describe the implementation details according to different backbones. Further, to have a fair comparison, we keep the experimental settings the same as the backbone networks.

Point Transformer. To have a fair comparison, we keep the training settings the same for the "Point Transformer" and "Point Transformer + Retro-FPN". Specifically, we train the networks using SGD optimizer with momentum and weight decay of 0.9 and 0.0001, respectively. We use a batch size of 4 and train for 100 epochs with an initial learning rate of 0.1, which is dropped by 0.1 at epochs 60 and 80.

KPConv. KPConv conducts semantic segmentation by segmenting small subclouds contained in spheres. During training, the spheres are randomly sampled. During testing, the spheres are regularly picked. Then, the segmentation results of all subclouds are merged into the whole scene through a voting scheme. For both KPConv *deform* and KPConv *rigid*, we set the radius of the sampled spheres to be 1.5m, and the batch size is 6. The other experimental settings are the same as the KPConv paper, including voting augmentation during testing.

MinkowskiNet. For MinkowskiNet, we conduct experiments on the S3DIS [1], ScanNet v2 [4], and SemanticKITTI [2] datasets. For experiments on the S3DIS and the ScanNet v2 dataset, we train the networks using SGD optimizer with an initial learning rate of 0.1, batch size is 16, momentum and wight decay are 0.98 and 0.0001, respectively. During testing, we follow BPNet[5] and MinkowskiNet[3] to obtain the test predictions with voting augmentation. The learning rate is decayed with poly learning rate scheduler and multi-step learning rate scheduler on S3DIS and ScanNet, respectively. For experiments on the SemanticKITTI [2] dataset, we first train "MinkowskiNet + Retro-FPN" for 15 epochs (on sequences 00-07 and 09-10) with a starting learning rate 0.24 and cosine learning rate decay, the batch size is 4. Then, the model is finetuned for another 15 epochs with a starting learning rate 0.096 and cosine learning rate decay, where we select the best mIoU during finetuning as the result of the validation set (sequence 08). To obtain results on the testing set (online test), we further finetune the model on both the training and validation sets (sequences 00-10) for 10 epochs with a starting learning rate 0.032 and cosine learning rate decay. Following the same practice as SPVNAS [7] and Cylinder3D [11], 5 view rotation augmentation is used for getting the online test predictions.

3. Detailed Experimental Results

In this section, we provide the detailed experimental results on the S3DIS [1] Area 5 and 6-fold evaluation, the ScanNet v2 [4] and SemanticKITTI [2] datasets. The results are listed in Table 5, 6, 7, and 8. Note that the per-category IoUs listed in Table 7 are results on the test set, which are cited from the ScanNet online website[*].

4. More Visualization Results

In this section, we provide more visualization results of the refining process conducted by Retro-FPN. In Figure 1, we present more visual comparison between the backbone network and Retro-FPN. The refining process of the improved areas are lighlighted in blue circles. At last, figure 2 shows the visual results on S3DIS [1] Area 5, and Figure 3 visualizes the refined results on ScanNet v2 [4] dataset, Figure 4 shows the segmentation results in all pyramid layers. The segmentation results of each pyramid layer are highlighted in blue circles.

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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1534–1543, 2016. 1, 2, 3, 4, 6, 7

[2] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 3, 5, 6, 9

[3] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 3, 4, 5

[4] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 1, 2, 3, 5, 8

[5] Wenbo Hu, Hengshuang Zhao, Li Jiang, Jiaya Jia, and Tien-Tsin Wong. Bidirectional projection network for cross dimension scene understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14373–14382, 2021. 3

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015. 2

[7] Haotian* Tang, Zhijian* Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3D architectures with sparse point-voxel convolution. In *European Conference on Computer Vision*, 2020. 3

[8] 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 *Proceedings of the IEEE/CVF international*

[*]http://kaldir.vc.in.tum.de/scannet_benchmark/

Table 5. Quantitative results on the S3DIS [1] dataset, evaluated on Area 5 validation. Red number means better results than baseline.

Method	mIoU	ceil.	floor	wall	beam	col.	wind.	door	table	chair	sofa	book.	board	clut.
MinkowskiNet (5cm) [3]	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
+ Retro-FPN	69.5	94.3	97.0	85.3	0.0	27.0	59.5	77.5	81.2	89.4	74.1	75.2	80.5	62.0
KPConv *rigid* [8]	65.4	92.6	97.3	81.4	0.0	16.5	54.5	69.5	90.1	80.2	74.6	66.4	63.7	58.1
+ Retro-FPN	69.7	94.0	98.2	84.0	0.0	31.9	62.6	78.4	92.0	80.0	74.7	79.9	71.1	59.9
KPConv *deform* [8]	67.1	92.8	97.3	82.4	0.0	23.9	58.0	69.0	81.5	91.0	75.4	75.3	66.7	58.9
+ Retro-FPN	70.7	94.8	98.5	84.2	0.0	40.3	58.9	79.2	92.1	83.0	77.5	76.6	72.7	61.6
PointTransformer [10]	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
+ Retro-PFN	73.0	95.6	98.5	88.1	0.0	44.7	64.4	80.4	83.8	91.9	84.2	75.1	80.0	62.1

Table 6. Quantitative results on the S3DIS [1] dataset, evaluated on 6-fold cross validation. Red number means better results than baseline.

Method	mIoU	ceil.	floor	wall	beam	col.	wind.	door	table	chair	sofa	book.	board	clut.
PointTransformer [10]	73.5	94.3	97.5	84.7	55.6	58.1	66.1	78.2	77.6	74.1	67.3	71.2	65.7	64.8
+ Retro-PFN	77.3	95.8	97.7	86.6	68.0	61.0	69.4	81.4	77.1	81.7	75.3	69.9	72.8	67.5

conference on computer vision, pages 6411–6420, 2019. 2, 4

[9] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. In *NeurIPS*, 2022. 2

[10] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 1, 2, 4

[11] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3D convolution networks for lidar segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 9939–9948, 2021. 3

Table 7. Quantitative results on the ScanNet [4] dataset, evaluated on the test set. Red number means better results than baseline.

Method	mIoU	bath.	bed	bksf.	cab.	chair	ctr.	curt.	desk	door	floor	oth.	pic.	ref.	shw.	sink	sofa	tab.	toil.	wall	win.
MinkowskiNet (2cm) [3]	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
+ Retro-FPN	74.4	84.2	80.0	76.7	74.0	83.6	54.1	91.4	67.2	62.6	95.8	55.2	27.2	77.7	88.6	69.6	80.1	67.4	94.1	85.8	71.7

Table 8. Quantitative results on the SemanticKITTI [2] dataset, evaluated on the test set. Red number means better results than baseline.

Method	mIoU	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
MinkowskiNet (5cm) [3]	64.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
+ Retro-FPN	68.0	97.5	50.2	52.3	57.7	62.0	65.7	69.4	43.3	91.1	69.1	76.7	34.5	92.5	68.7	85.8	73.9	70.3	64.9	65.9

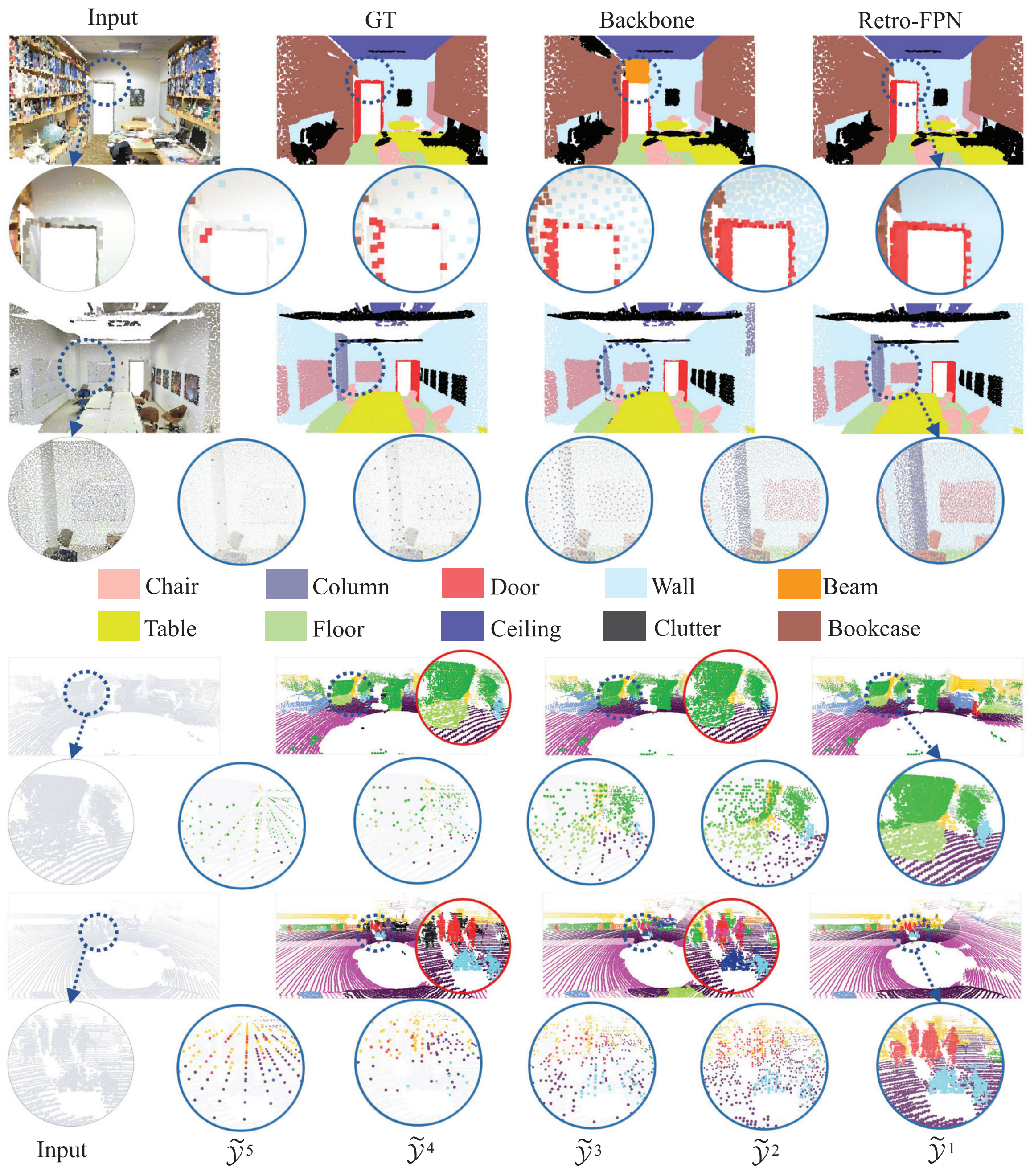
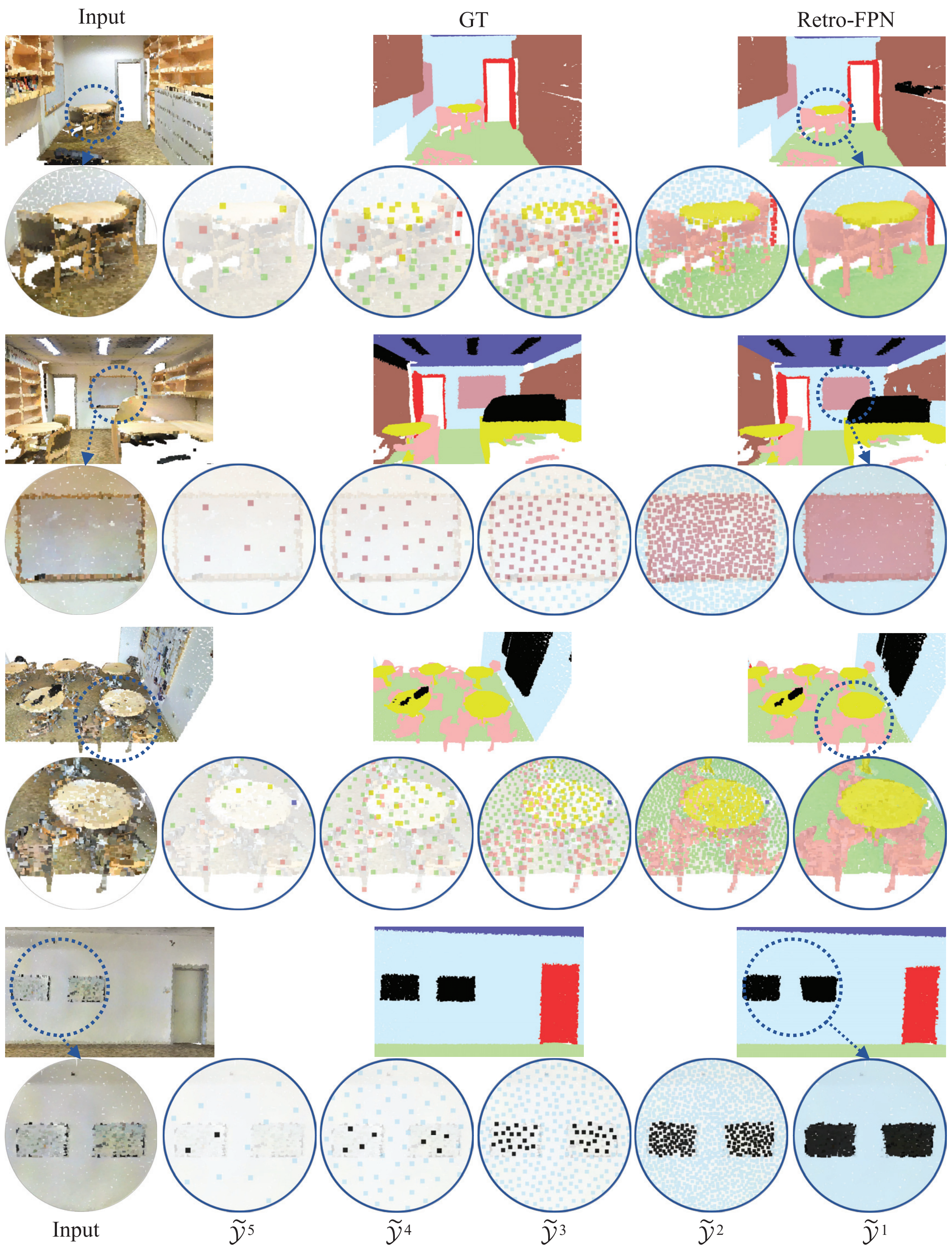


Figure 1. More visual comparison of the segmentation results on both the indoor [1] (the first two examples) and the outdoor [2] (the last two examples) datasets. The circles highlighted in blue visualize the segmentation process of the improved areas. The first two examples show the comparison between "Point Transformer" and "Point Transformer + Retro-FPN" on the S3DIS [1] dataset. The last two examples show the comparison between "MinkowskiNet" and "MinkowskiNet + Retro-FPN" on the SemanticKITTI [2] dataset.



🟥	Chair	🟪	Column	🟥	Door	🟦	Wall	🟪	Sofa	🟧	Beam
🟨	Table	🟩	Floor	🟦	Ceiling	⬛	Clutter	🟫	Bookcase		

Figure 2. More visualization results of the refining process on S3DIS [1] Area 5. The circular areas highlighted in blue visualize the predicted labels for each pyramid layer.

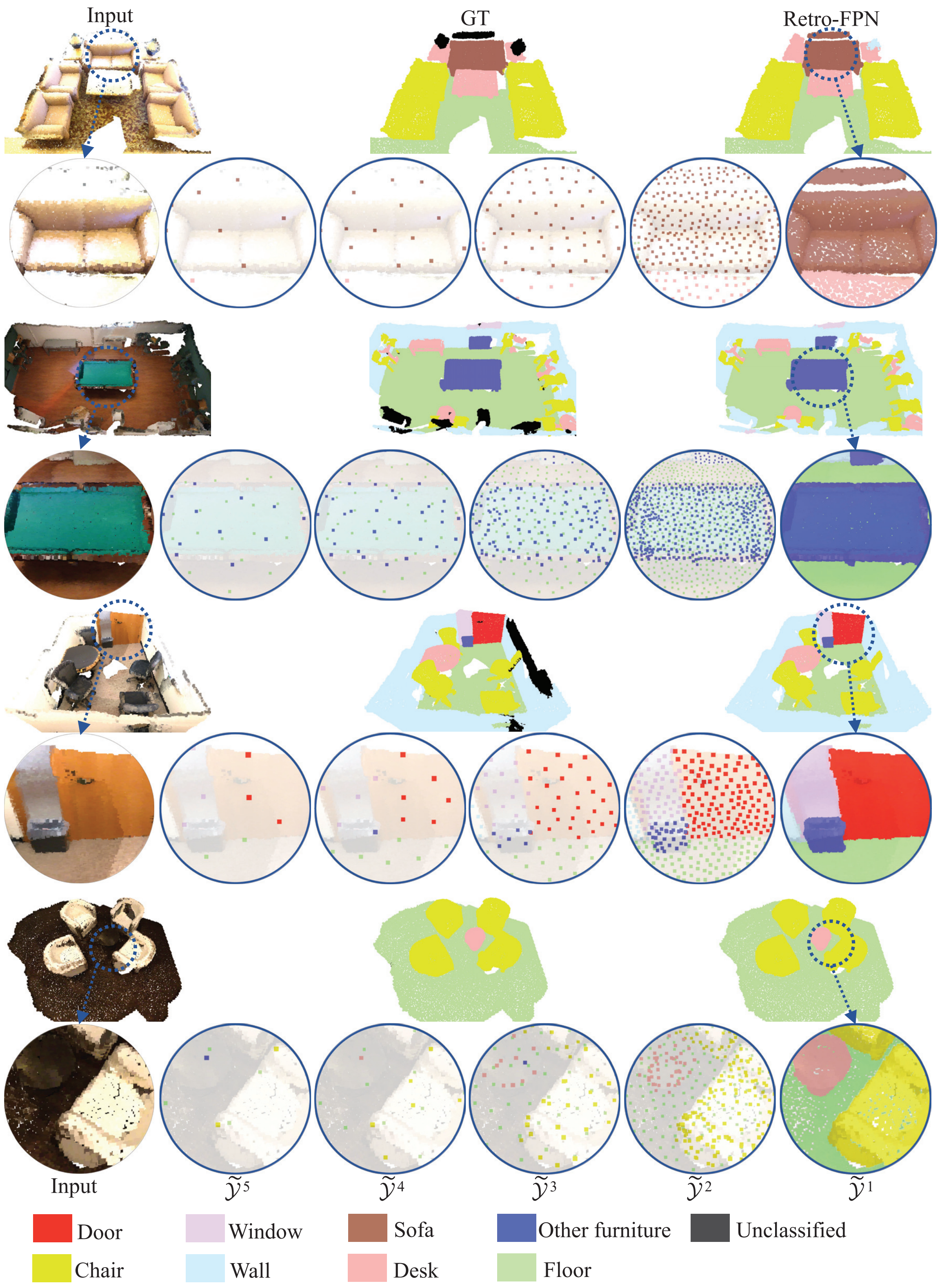


Figure 3. More visualization results of the refining process on ScanNet v2 [4]. The circular areas highlighted in blue visualize the predicted labels for each pyramid layer.

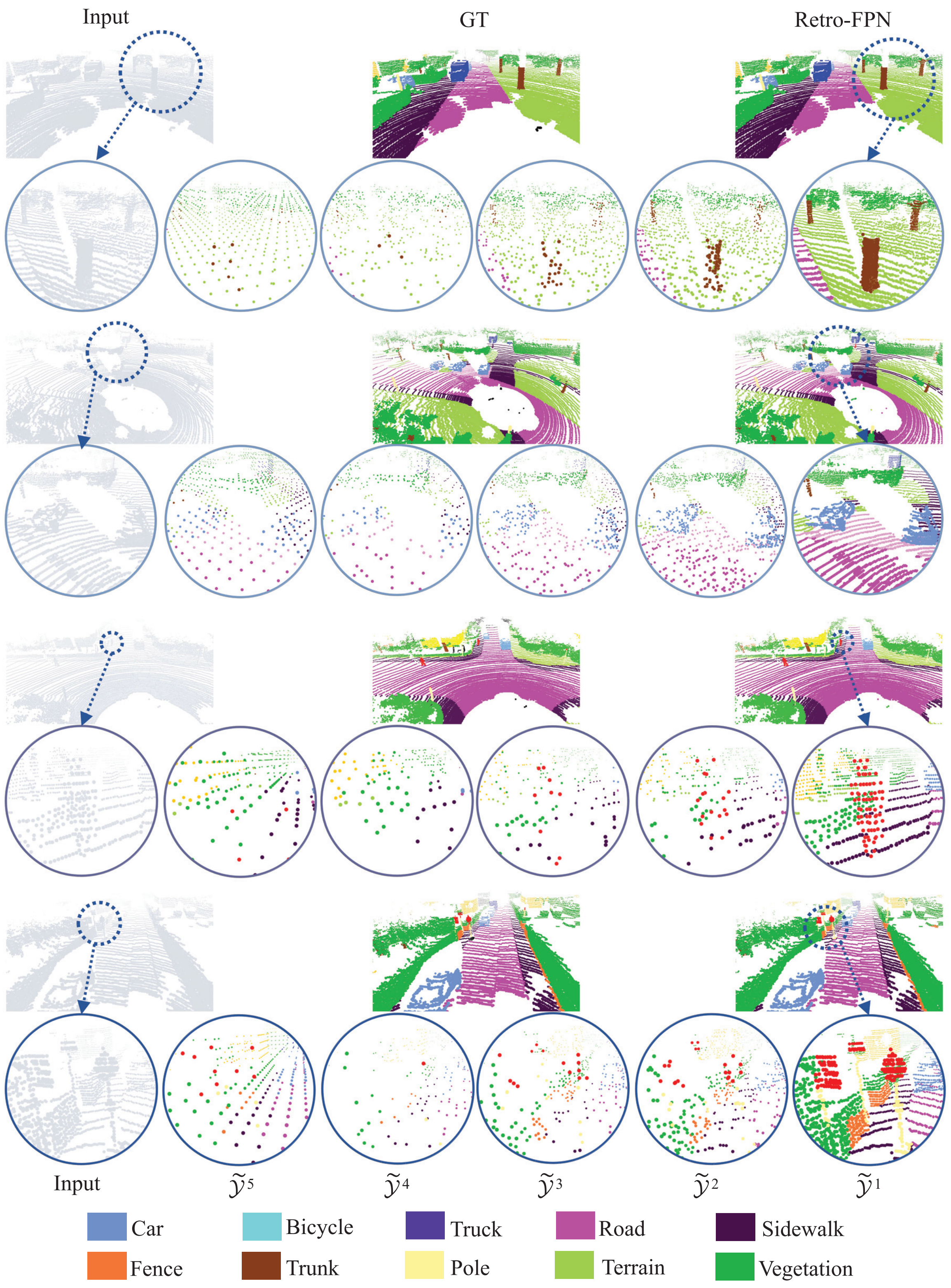


Figure 4. More visualization results of the refining process on SemanticKITTI [2]. The circular areas highlighted in blue visualize the predicted labels for each pyramid layer.