Supplementary Material for
Differentiable Convolution Search for Point Cloud Processing

Xing Nie[1,2], Yongcheng Liu[1], Shaohong Chen[4], Jianlong Chang[3],
Chunlei Huo[1*], Gaofeng Meng[1,2,5], Qi Tian[3], Weiming Hu[1], Chunhong Pan[1],

[1] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences.
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences. [3] Huawei Cloud & AI.
[4] Xidian University. [5] Centre for Artificial Intelligence and Robotics, HK Institute of Science & Innovation, CAS.
Email: niexing2019@ia.ac.cn, {yongcheng.liu, clhuo}@nlpr.ia.ac.cn

1. Outline

This supplementary material provides further investigations for the proposed PointSeaNet. To be concrete, further analysis experiments are presented in Sec. 2. More details of our network architectures and training parameters are described in Sec. 3. Moreover, additional segmentation results on the S3DIS and ShapeNetPart benchmark are provided in Sec. 4. Finally, we show more result visualizations of shape part segmentation, large-scale scene segmentation and normal estimation in Sec. 5.

2. Further Investigations

In this section, we provide further investigations of PointSeaNet on two aspects, *i.e.*, the impact of the number of cells for architecture search and the influence of the dropout technique.

Number of cells for architecture search. To better understand the effect of stacked cells for architecture search, we conduct architecture evaluation at different numbers of cells during the search phase for shape classification on ModelNet40, which is different from the ablation study on stacked cells during the evaluation process (Sec. 4.3) in the main paper. All the search procedures are conducted on one NVIDIA TITAN Xp. The quantitative results are summarized in Tab. 1. To allow fair comparison, we choose the same setting for each case in Tab. 1 during the evaluation phase, including 6 stacked cells for architecture evaluation. From this table, we have the following two observations. First, the impact of the number of cells on architectural parameters is limited, yet it significantly causes search cost with the increase of the cells. Second, PointSeaNet can achieve an evaluation accuracy of 93.7% with only 1 cell for architecture search. This adequately demonstrates that PointSeaNet can achieve superior evaluation performance with only a few parameters for architecture search.

# Cells[1]	OA (%)	#params (M)	Search Cost (GPU-days)
1	93.7	7.30	0.16
2	**94.2**	8.45	0.25
3	94.1	9.05	0.57
4	94.0	9.95	0.81

Table 1. The comparisons of different numbers of cells for architecture search. All the results are reported during the evaluation phase.

ratio(%)	0	10	20	30	40	50	60	70
acc.	93.7	93.8	93.7	94.0	94.1	**94.2**	93.9	93.4

Table 2. The results (%) of dropout with different ratios applied on dropout $\boldsymbol{p}_i'$ in Eq. (4) in the main paper.

Dropout on $\boldsymbol{p}_i'$ in Eq. (4). Dropout technique is capable of driving the network to represent as an ensemble of a large number of subsets and reducing the risk of model overfitting. To show its impact on PointSeaNet, we select different radios on $\boldsymbol{p}_i'$ in Eq. (4) in the main paper for shape classification on the ModelNet40 benchmark. Tab. 2 summarizes the results. As can be seen, the best result of 94.2% can be obtained with a dropout ratio of 50%. Moreover, it is noticeable that PointSeaNet can achieve a decent result of 93.7% even without the dropout technique, outperforming most handcrafted methods.

3. Network Architectures and Parameters

In this section, we provide more details on classification network, segmentation network and training parameters.

*Corresponding author.

[1]Different from the ablation study in the main paper, here, #cells denotes the number of cells during the search phase, with 6 stacked cells for architecture evaluation.

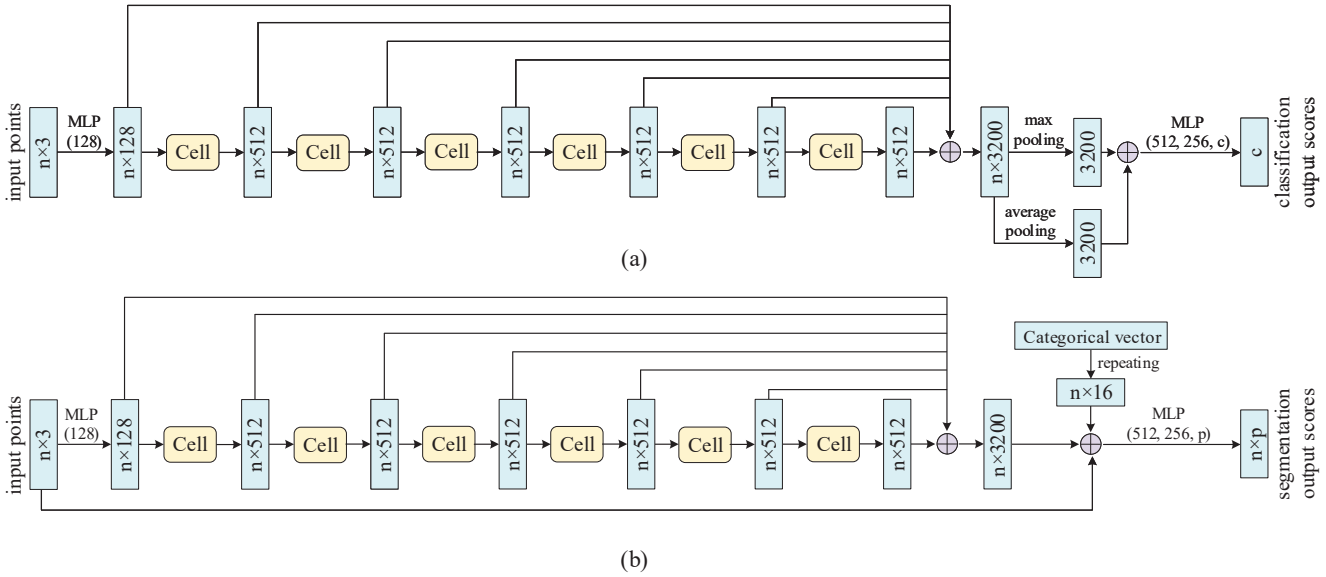


Figure 1. The architectures of PointSeaNet applied in the classification (a) and segmentation (b) of point clouds. c is the number of classes and p is the semantic labels. $\oplus$ denotes concatenation.

PointSeaNet Classification Network. The network architecture used for shape classification takes raw point cloud as input, as shown in Fig. 1(a). It is first composed of a shared $\mathrm{MLP}(128)$ network (with layer output size 128) on each point. Then, we stack the searched cell 6 times to learn underlying geometric features. Those features extracted by the first MLP and each cell are concatenated to provide contextual semantic information. After the last cell, the features are aggregated by a global max pooling and average pooling, and then the outputs of the aggregation are concatenated. Finally, a shared $\mathrm{MLP}(512, 256, c)$ network is used as the classifier, where c indicates the number of classes.

PointSeaNet Segmentation Network. The segmentation network is an extension to the PointSeaNet classification one, as illustrated in Fig. 1(b). The input points and point features (the output after the last cell) are concatenated in the channel dimension. As used in [9, 11, 16], we also add a one-hot vector that contains the object class of input points. This vector is repeated n times and then concatenated with the point features. Finally, the output segmentation scores are obtained by a shared $\mathrm{MLP}(512, 256, p)$ network without any global pooling layer, where p denotes the semantic labels. No dropout is applied and other hyperparameters are the same as the classification network.

Training Parameters. In our experiment, there are 6 nodes with 3 intermediate nodes for each cell architecture, where the output node is defined as the depth-wise concatenation of its four precedents. Each intermediate node must select two input nodes from its precedents. PointSeaNet is obtained through two stages, a search phase and an evaluation phase. During the search phase, we stack each identical cell 2 times to search the optimal convolution and cell architecture for 50 epochs. After search, we stack the searched cell architecture 6 times to form a large backbone network, and then train it from scratch for 400 epochs. We use the same settings for all the NAS methods. Usually, NAS methods use different settings from the handcrafted ones. We report the best results for two class of methods. Our PointSeaNet is implemented with Pytorch. Neighboring points in local point subset are gathered by *k nearest neighbor* in the first operation of each cell. During the search phase, 2 cells with 32 initial channels and $k = 9$ are used to search the optimal cell architecture for 50 epochs with batch size 16. SGD is employed with an initial learning rate 0.005, momentum 0.9 and weight decay 3×10^{-4}. A cosine annealing is used to schedule the learning rate with the minimum learning rate 1×10^{-4}. For the evaluation phase, we stack the searched cell 6 times and apply $k = 20$ to form a larger network, and then the final network is trained from scratch for 400 epochs with batch size 128. The Adam optimization algorithm with learning rate 0.001 and weight decay 1×10^{-4} is employed, where a cosine annealing with the minimum learning rate 1×10^{-5} is used to schedule the learning rate. Regarding the epsilon-greedy algorithm, we set the probability ε to be 0.5, which is responsible for balancing the greedy algorithm and random algorithm when choosing the essential association candidates in convolution search.

4. Additional Segmentation Results

We provide more details of our PointSeaNet on the S3DIS (Tab.3 and Tab.4) and ShapeNetPart (Tab. 5) benchmark, which contain segmentation scores of each class. On S3DIS [1], our PointSeaNet and PointSeaNet† outperform all the compared methods with significant advan-

tage, which set new state of the arts in point-based methods over 9 categories on the 6-fold cross-validation. On ShapeNetPart [19], our PointSeaNet significantly surpasses the second best method, *i.e.*, Densepoint [9], with $1.5\uparrow$ in class mIoU and $1.4\uparrow$ in instance mIoU, respectively. Note that, our two searched models also set new state of the arts in point-based methods over fourteen categories. In addition, PointSeaNet achieves better performance than PointSeaNet† in most classes, which adequately validates the effectiveness of integrating *epsilon-greedy* into the convolution search.

5. More Visualizations

In this section, we show more visualization results of large-scale scene segmentation (Fig. 2), shape part segmentation (Fig. 3) and normal estimation (Fig. 4), respectively. As can be seen, compared with PointNet++ [12] and PointNet [11], our PointSeaNet obtains superior normal estimation results. Additionally, although the indoor scenes implied in irregular points are varied and they may be very confusing to recognize, our PointSeaNet can also segment them out with decent accuracy. However, PointSeaNet could also be less effective for several intractable shapes, such as lamps in normal estimation and motorbikes in shape part segmentation.

References

[1] I. Armeni, O. Sener, A.R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, 2016. 2

[2] M. Atzmon, H. Maron, and Y. Lipman. Point convolutional neural networks by extension operators. *ACM TOG*, 37(4):1–12, 2018. 4

[3] B. Graham, M. Engelcke, and L. Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 4

[4] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 2020. 4

[5] Q. Huang, W. Wang, and U. Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *CVPR*, 2018. 4

[6] R. Klokov and V. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *ICCV*, 2017. 4

[7] J. Li, B.M. Chen, and G. H. Lee. So-net: Self-organizing network for point cloud analysis. In *CVPR*, 2018. 4

[8] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, 2018. 4

[9] Y. Liu, B. Fan, G. Meng, J. Lu, S. Xiang, and C. Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *ICCV*, 2019. 2, 3, 4

[10] Y. Liu, B. Fan, S. Xiang, and C. Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019. 4

[11] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2, 3, 4, 7

[12] C.R. Qi, L. Yi, H. Su, and L.J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 3, 4, 7

[13] Y. Shen, C. Feng, Y. Yang, and D. Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*, 2018. 4

[14] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M. Yang, and J. Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *CVPR*, 2018. 4

[15] H. Thomas, C.R. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L.J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. 4

[16] 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 TOG*, 38(5):1–12, 2019. 2, 4

[17] S. Xie, S. Liu, Z. Chen, and Z. Tu. Attentional shapecontextnet for point cloud recognition. In *CVPR*, 2018. 4

[18] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, 2018. 4

[19] L. Yi, V.G. Kim, D. Ceylan, I.C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM TOG*, 35(6):1–12, 2016. 3

[20] L. Yi, H. Su, X. Guo, and L.J. Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, 2017. 4

[21] H. Zhao, L. Jiang, C. Fu, and J. Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019. 4

Method	OA	mIoU	ceil.	floor	wall	beam	col.	wind.	door	table	chair	sofa	book.	board	clut.
PointNet [11]	78.6	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	54.1	42.0	9.6	38.2	29.4	35.2
SPG [3]	85.5	62.1	89.9	95.1	76.4	62.8	47.1	55.3	68.4	**73.5**	69.2	63.2	45.9	8.7	52.9
PointCNN [8]	88.1	65.4	94.8	**97.3**	75.8	63.3	51.7	58.4	57.2	71.6	69.1	39.1	61.2	52.2	58.6
PointWeb [21]	87.3	66.7	93.5	94.2	80.2	52.4	41.3	64.9	68.1	71.4	67.1	50.3	62.7	62.2	58.5
KPConv [15]	-	70.6	93.6	92.4	83.1	63.9	54.3	66.1	76.6	57.8	64.0	**69.3**	74.9	61.3	60.3
RandLA-Net [4]	87.2	68.5	92.7	95.6	79.2	61.7	47.0	63.1	67.7	68.9	**74.2**	55.3	63.4	63.0	58.7
PointSeaNet†	89.6	71.2	94.3	95.4	80.1	62.7	**55.8**	65.6	74.0	65.3	68.2	66.9	72.5	**65.1**	59.3
PointSeaNet	**90.3**	**71.9**	**95.2**	93.4	**83.9**	**64.3**	51.1	**67.3**	**77.5**	67.9	70.1	63.6	**75.9**	64.2	**60.8**

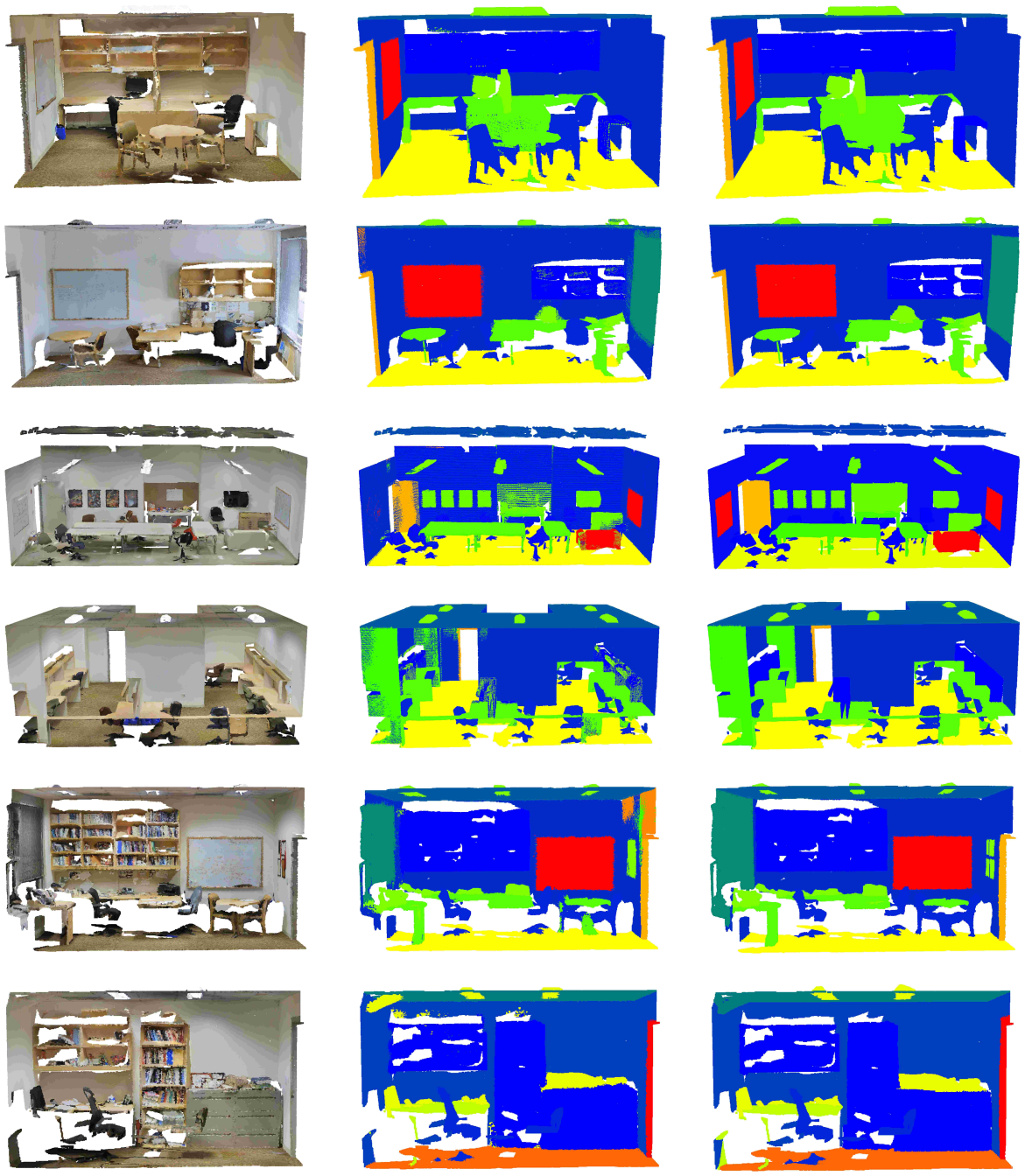
Table 3. Scene segmentation results (%) on S3DIS *6-fold* ('-': unknown).

Method	OA	mIoU	ceil.	floor	wall	beam	col.	wind.	door	table	chair	sofa	book.	board	clut.
PointNet [11]	-	41.1	88.8	97.3	69.8	**0.1**	3.9	46.3	10.8	52.6	58.9	40.3	5.9	26.4	33.2
SPG [3]	86.4	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
PointCNN [8]	85.9	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]	87.0	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 [15]	-	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
PointSeaNet†	88.1	68.2	92.6	97.3	82.9	0.0	24.8	63.2	69.4	88.5	88.1	74.5	76.8	68.6	59.6
PointSeaNet	**89.2**	**69.0**	**93.1**	98.0	**84.6**	0.0	26.3	**64.6**	**70.9**	89.4	86.9	**76.1**	**77.2**	**69.9**	**60.3**

Table 4. Scene segmentation results (%) on S3DIS *Area-5* ('-': unknown).

Method	input	class mIoU	inst. mIoU	aero	bag	cap	car	chair	ear	guit	knif	lamp	lapt	moto	mug	pist	rock	skate	table
Kd-Net [6]	4k	77.4	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
PointNet [11]	2k	80.4	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
RS-Net [5]	-	81.4	84.9	82.7	**86.4**	84.1	78.2	90.4	69.3	91.4	87.0	83.5	95.4	66.0	92.6	81.8	56.1	75.8	82.2
SCN [17]	1k	81.8	84.6	83.8	80.8	83.5	79.3	90.5	69.8	91.7	86.5	82.9	96.0	69.2	93.8	82.5	62.9	74.4	80.8
PCNN [2]	2k	81.8	85.1	82.4	80.1	85.5	79.5	90.8	73.2	91.3	86.0	85.0	95.7	73.2	94.8	83.3	51.0	75.0	81.8
SPLATNet [14]	-	82.0	84.6	81.9	83.9	88.6	79.5	90.1	73.5	91.3	84.7	84.5	96.3	69.7	95.0	81.7	59.2	70.4	81.3
KCNet [13]	2k	82.2	84.7	82.8	81.5	86.4	77.6	90.3	76.8	91.0	87.2	84.5	95.5	69.2	94.4	81.6	60.1	75.2	81.3
DGCNN [16]	2k	82.3	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
RS-CNN [10]	2k	84.0	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
Densepoint [9]	2k	84.2	86.4	84.0	85.4	90.0	79.2	91.1	**81.6**	91.5	87.5	84.7	95.9	74.3	94.6	82.9	**64.6**	76.8	83.7
PointNet++ [12]	2k,nor	81.9	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
SyncCNN [20]	mesh	82.0	84.7	81.6	81.7	81.9	75.2	90.2	74.9	93.0	86.1	84.7	95.6	66.7	92.7	81.6	60.6	**82.9**	82.1
SO-Net [7]	1k,nor	80.8	84.6	81.9	83.5	84.8	78.1	90.8	72.2	90.1	83.6	82.3	95.2	69.3	94.2	80.0	51.6	72.1	82.6
SpiderCNN [18]	2k,nor	82.4	85.3	83.5	81.0	87.2	77.5	90.7	76.8	91.1	87.3	83.3	95.8	70.2	93.5	82.7	59.7	75.8	82.8
PointSeaNet†	2k	85.2	87.3	86.8	86.2	89.3	80.9	91.8	81.5	92.9	88.8	85.3	95.8	**76.6**	94.7	83.9	64.4	79.2	84.7
PointSeaNet	2k	**85.7**	**87.8**	**87.6**	85.9	**91.7**	**81.8**	**92.1**	81.3	**93.4**	**89.3**	**86.7**	**96.9**	75.2	**95.9**	**84.8**	64.1	78.9	**85.6**

Table 5. Shape part segmentation results (%) on ShapeNetPart (nor: normal, '-': unknown).



(a) Input (b) Prediction (c) Ground Truth

Figure 2. Large-scale scene segmentation results on S3DIS. In the first column we show the input point clouds of some complex scenes. In the second column, we provide the semantic segmentation results of our PointSeaNet. In the last column, we show the corresponding ground-truths.

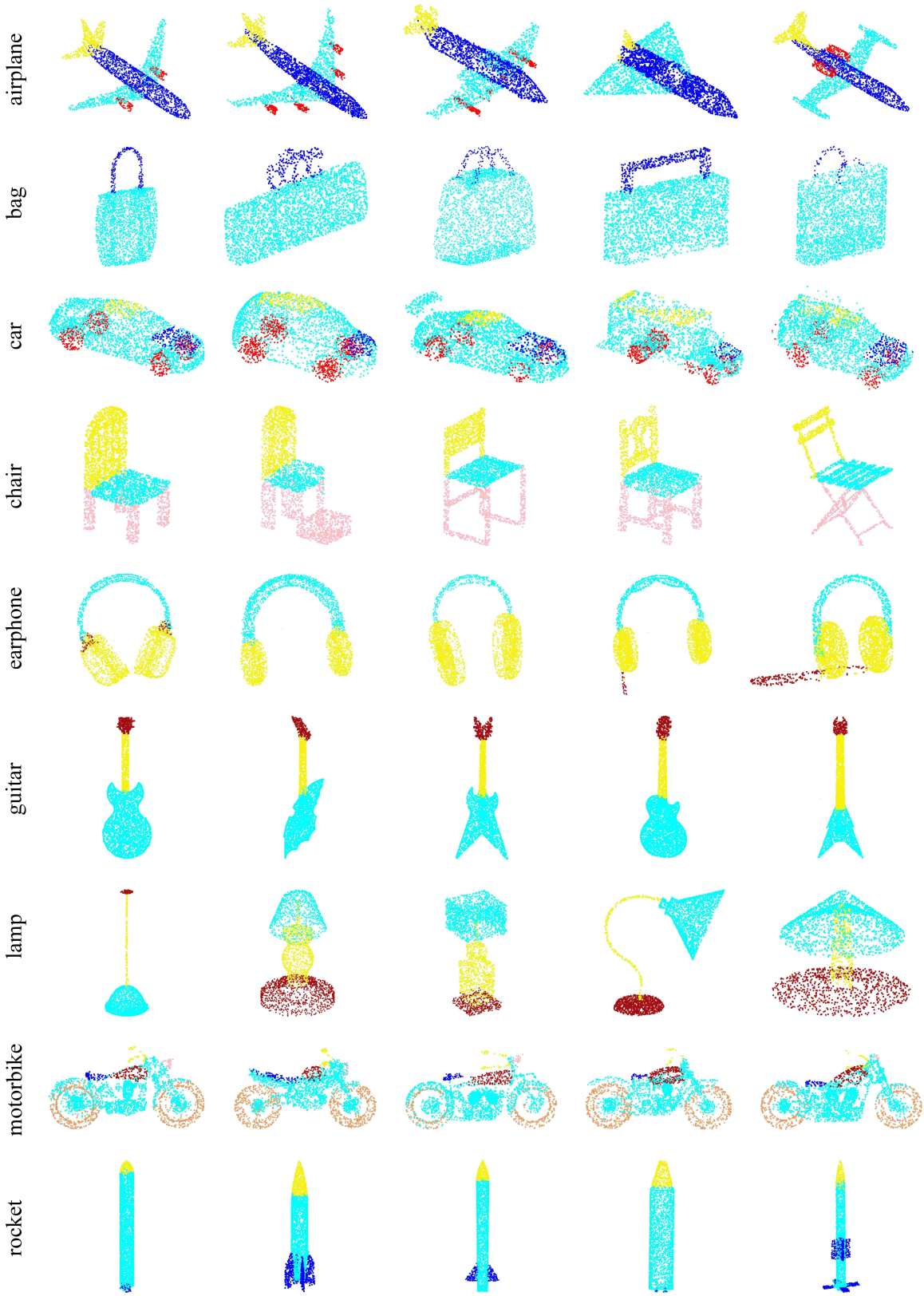


Figure 3. PointSeaNet shape part segmentation results on ShapeNetPart.

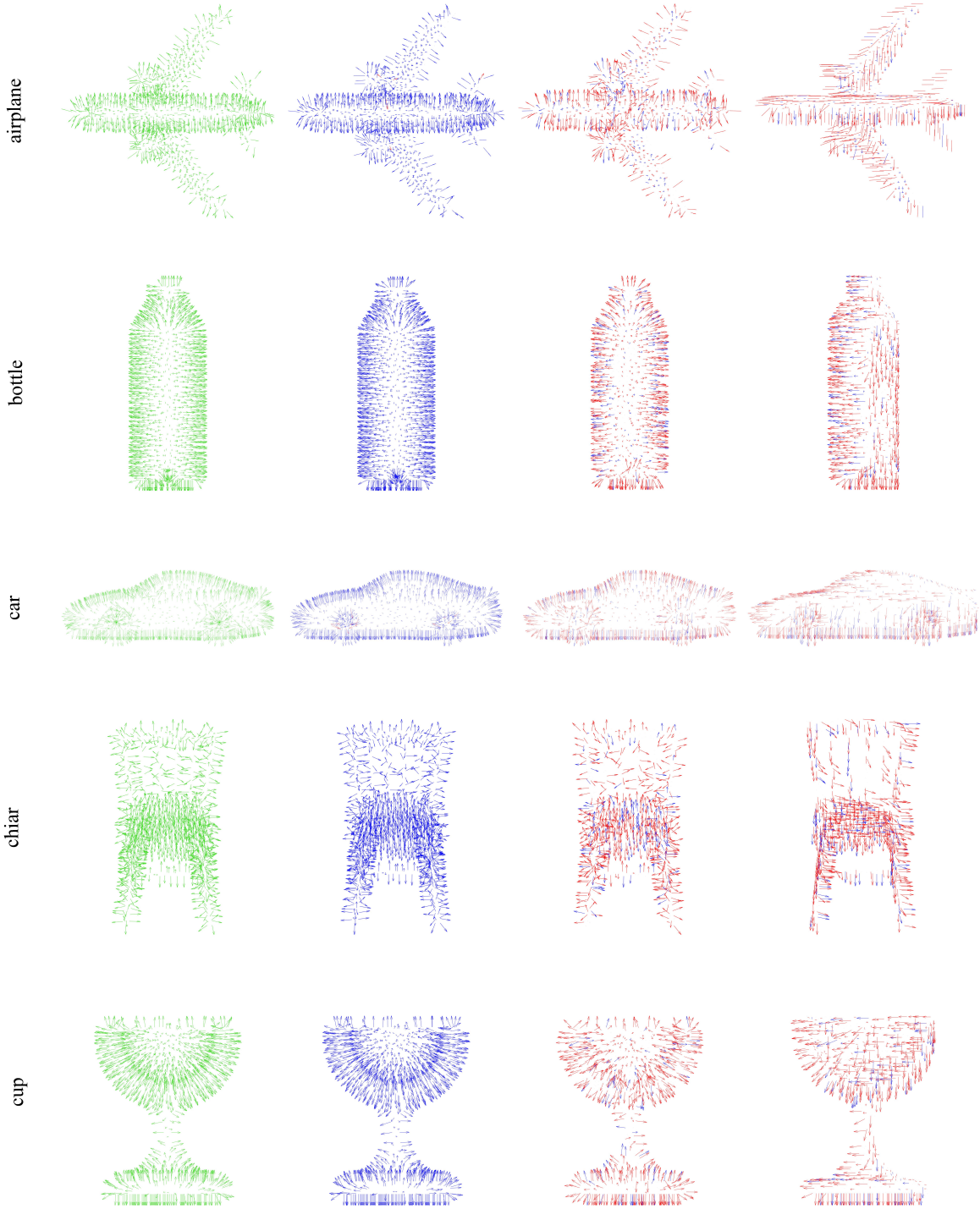


Figure 4. PointSeaNet normal estimation results on ModelNet40. Results from left to right: ground truth normals, PointSeaNet, PointNet++ [12], PointNet [11]. For clearness, only the predictions with angle less than 30% in blue and angle greater than 90% in red between ground truth normals are shown.