Supplementary Material for
PIDS: Joint Point Interaction-Dimension Search for 3D Point Cloud

In this appendix, we first evaluate PIDS on ModelNet40 to study its effect on classification benchmarks. Then, we elaborate the detailed structure of hand-crafted PIDS, built upon efficient 3D point operators within the PIDS search space. Finally, we discuss the detailed configuration of Dense-Sparse predictor. Our code is publicly available here.

9.1. Evaluation on ModelNet40 Classification

We run PIDS on ModelNet40 classification benchmark to search for highly representative classification models. On ModelNet40, a single training batch in ModelNet40 contains 16 sub-sampled point clouds. We adopt a similar training pipeline following the original KPConv [7] paper yet incorporates SimpleView [1] RSCNN training protocol for fair comparison. Specifically, the SimpleView-RSCNN protocol adopts random scaling & translation for data augmentation, employ cross-entropy loss to optimize, and utilize a voting scheme on the best test model to best exploit the potential of searched model.

Table 9. Overall Accuracy (OA) on ModelNet40. *: Use native protocol instead of SimpleView RSCNN evaluation protocol.

Architecture	Params (M)	OA (%)
PointCNN [3]	0.60	92.2
PointConv [9]	-	92.5
SPH3D-GCN* [2]	0.8	92.1
FPConv* [4]	2.1	92.5
RSCNN [5]	1.3	92.5
DGCNN [8]	1.8	92.8
KPConv [7]	14.9	92.9
SimpleView [1]	0.80	93.2
PointNet++ [6]	1.48	93.3
PIDS (second-order)	1.25	92.6
PIDS (NAS)	**0.56**	93.1
PIDS (NAS, 2×)	1.21	**93.4**

We demonstrate the evaluation results on the testing dataset of ModelNet40 in Table 9. PIDS explores more efficient model designs with higher performance, outperforming KPConv and other prior arts [1] with up to 1.3% OA while being up to 12× smaller in size.

9.2. Structure of First-order PIDS

Table 10. Structure of hand-crafted PIDS (first-order). "1/2" strides means up-sampling by 2×, and "O" denotes "Octahedron" kernel with 7 kernel points. All point operators employ a first-order point interaction.

Point-Op	Depth	Stride	In Width	Out Width	E	K
1	1	1	16	16	1	O
2	2	2	16	24	3	O
3	3	2	24	32	3	O
4	4	2	32	64	3	O
5	3	1	64	96	3	O
6	3	2	96	160	3	O
7	1	1	160	320	3	O
8	1	1/2	416	160	3	O
9	1	1/2	160	96	3	O
10	1	1/2	96	64	3	O
11	1	1/2	64	32	3	O

We follow the layer organization to manually craft first-order PIDS (Table 2), and demonstrate the detailed architecture in Table 10, echoing the design of search components in Section 3.3. The hand-crafted architecture contains a total of 11 point operators (Point-Op). Among them, 7 point operators serve as a classification backbone, and 4 additional point operators serve as a semantic segmentation head. The positional setting (i.e., depth, block width) of hand-crafted PIDS strictly follows MobileNet-V2, which is considered as a good hand-crafted model design with remarkable efficiency. For structural settings, we employ an expansion factor (E) of 3 for all point operators except for the first one. We leverage the 7-point Octahedron layout for each point operator for design efficiency. As a result, first-order PIDS serves as a strong baseline to compare with, when we evaluate the performance of NAS-crafted PIDS models.

9.3. Details of PIDS Search Space

We present the detailed settings of the PIDS interaction-dimension search space in Table 11. We jointly search the point interaction (kernel size and interaction type) and point dimension (block depth, block width, expansion factor) to find the best architecture.

Table 11. Configuration of the PIDS Search Space. "1/2" strides means up-sampling by 2×.

Hierarchy	Stage	Strides	Order Type	Kernel Type	Depth	Expansion Factor	Width
Backbone	**1**	1	first-order	Tetrahedron, Octahedron, Icosahedron	1	1	16
	2	2	first-order	Tetrahedron, Octahedron, Icosahedron	2, 3	2, 3, 4	16, 24
	3	2	first-order/second-order	Tetrahedron, Octahedron, Icosahedron	2, 3, 4	2, 3, 4	24, 32
	4	2	first-order/second-order	Tetrahedron, Octahedron, Icosahedron	3, 4, 5	2, 3, 4	24, 32, 40
	5	1	first-order/second-order	Tetrahedron, Octahedron, Icosahedron	2, 3, 4	2, 3, 4	40, 56, 72
	6	2	first-order/second-order	Tetrahedron, Octahedron, Icosahedron	3, 4, 5	2, 3, 4	64, 80, 96
	7	1	first-order/second-order	Tetrahedron, Octahedron, Icosahedron	1	2, 3, 4	160
Segmentation Head	**8**	1/2	first-order/second-order	Tetrahedron, Octahedron, Icosahedron	1	2, 3	64, 80, 96
	9	1/2	first-order/second-order	Tetrahedron, Octahedron, Icosahedron	1	2, 3	40, 56, 72
	10	1/2	first-order/second-order	Tetrahedron, Octahedron, Icosahedron	1	2, 3	24, 32, 40
	11	1/2	first-order	Tetrahedron, Octahedron, Icosahedron	1	2, 3	16, 24

9.4. Design of Dense-Sparse Predictor

We manually configure the architectural hyperparameters of our Dense-Sparse predictor without delicate architecture engineering. Specifically, we employ a 3-layer MLP with 64-128-256 layers as the dense architecture to extract dense neural architecture representations for both positional organizations and structural settings. The overall architecture that is responsible for processing fused features after dense-sparse interaction is a 2-layer MLP with 256-256 layers. We utilize ReLU as the activation function and apply Dropout of 0.5 before the final regression head to mitigate overfitting. To ensure fair comparison in Table 3, both the Dense predictor and the Sparse predictor also adopt a 2-layer MLP with 256-128 units to keep the same maximum projection dimension as the Dense-Sparse predictor.

References

[1] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. *arXiv preprint arXiv:2106.05304*, 2021.

[2] Huan Lei, Naveed Akhtar, and Ajmal Mian. Spherical kernel for efficient graph convolution on 3d point clouds. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[3] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on $\mathcal{X}-$transformed points. *arXiv preprint arXiv:1801.07791*, 2018.

[4] Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. Fpconv: Learning local flattening for point convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2020.

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

[6] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[7] 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 International Conference on Computer Vision*, pages 6411–6420, 2019.

[8] 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.

[9] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.