Improving the Robustness of Point Convolution on k-Nearest Neighbor Neighborhoods with a Viewpoint-Invariant Coordinate Transform Supplementary Material

Xingyi Li, Wenxuan Wu, Xiaoli Z. Fern, and Li Fuxin
School of Electrical Engineering and Computer Science, Oregon State University
{lixin, wuwen, xfern, lif}@oregonstate.edu

Input	mIOU
XYZ	23.36
VI descriptor	27.62

Table 1: mIOU results on ScanNet for DGCNN [8].

1 Experiment with DGCNN

To study the effectiveness of the VI descriptor on other point cloud convolution frameworks, we replace edge inputs with VI descriptors on DGCNN[8], and the mIOU on ScanNet validation set improved by about 4% (Table 1). It justified that the VI descriptor is quite useful in other point cloud networks as well. DGCNN does not work well on this dataset because it does not include downsampling and upsampling layers, hence unable to incorporate global context. Our literature search found no other reference applying it on this dataset. Other prior work, e.g. PointNet, did not consider point-point relations hence cannot benefit from VI. Since a proper convolutional network (e.g. PointConv) outperforms most of these non-convolutional networks on point clouds, we felt putting the emphasis of our analysis on PointConv is appropriate.

2 Additional details for the ScanNet experiment

2.1 Model Architectures

Table 3 demonstrates the 16-layer architecture of the network used in the paper. For general hyperparameters, we set the learning rate to 10^{-3} with Adam optimizer, the weight decay to 10^{-5}, the number of neighbor points for kNN to 8, and the number of sub-sampled points for each point cloud to 10^5. In the case of VI-PointConv, we will just substitute all PointConv layers to VI-PointConv layers.

2.2 4-Layer Architecture

Table 2 demonstrates a the same 4-layer model as in the paper [9].

Layer name	Layer description
Conv1	PointConv 48 kernels w/ ReLU
Subsampling1	voxel subsampling with voxel size 0.05
Conv2	PointConv 96 kernels w/ ReLU
Subsampling2	voxel subsampling with voxel size 0.1
Conv3	PointConv 144 kernels w/ ReLU
Subsampling3	voxel subsampling with voxel size 0.2
Conv4	PointConv 192 kernels w/ ReLU
Subsampling4	voxel subsampling with voxel size 0.4
BottleNeck Conv	PointConv 240 kernels w/ ReLU
Upsampling4	retrieve results before Subsampling4
DeConv1	PointDeConv 192 kernels w/ ReLU
Upsampling3	retrieve results before Subsampling3
DeConv2	PointDeConv 144 kernels w/ ReLU
Upsampling2	retrieve results before Subsampling2
DeConv2	PointDeConv 96 kernels w/ ReLU
Upsampling1	retrieve results before Subsampling1
DeConv2	PointDeConv 48 kernels w/ ReLU
Output Conv	1x1 convolution
Softmax layer	

Table 2: The network architecture (4-layer) for ScanNet experiments. Every convolutional layer is followed by a ReLU layer.

2.3 Extra Performance Reports for the 16 Layer Architecture

We provide more detailed results for the ScanNet experiment, starting from Table 5. The first row of each Table indicates the number of subsampled points, and the first column represents rotation angles. Note that the performance differences under each column are not significant, which indicates that frameworks are robust against rotations. However, the performance decreases as the number of sampled points gets smaller and smaller. When the 3D coordinates input is replaced with our novel viewpoint-invariant descriptor, the performance and robustness are significantly improved for PointConv networks.

3 Speed Analysis for SemanticKITTI Validation Dataset

We reported the running time on the SemanticKITTI Validation Dataset [1] for several popular related work in Table 4. The dataset contains $4,071$ point clouds and the model we selected is the one as in Table 3. Compared with PointConv [9], our approach increased 0.02% on the number of parameters and 15.9% on the running time. The setup is the same as in [2] with a single 2080 Ti GPU and $81,920$ points per point cloud. The benefit of our approach is that our model can accept any number of input points, and $1.4M$ is the maximal amount of points for each point cloud in SemanticKITTI dataset [1]. PointConv and VI-PointConv is faster than KPConv [7] due to k-nearest neighbors inducing the same amount of neighbors at each point, whereas with ϵ-ball the amount of neighbors is variable, reducing the efficiency of vectorization. Especially, in terms of the **pre-processing time**, which includes computing of the nearest neighbors, there is a significant benefit (4x faster) of kNN versus the ϵ-ball used in KPConv on the same amount of points ($81,920$ per point cloud), as shown in Fig. 1 (c) of the main paper. RandLA-Net is significantly faster than

Layer name	Layer description
Conv1-1	PointConv 48 kernels w/ ReLU
Conv1-2	PointConv 48 kernels w/ ReLU
Conv1-3	PointConv 48 kernels w/ ReLU
Conv1-4	PointConv 48 kernels w/ ReLU
Subsampling1	voxel subsampling with voxel size 0.05
Conv2-1	PointConv 96 kernels w/ ReLU
Conv2-2	PointConv 96 kernels w/ ReLU
Conv2-3	PointConv 96 kernels w/ ReLU
Conv2-4	PointConv 96 kernels w/ ReLU
Subsampling2	voxel subsampling with voxel size 0.1
Conv3-1	PointConv 144 kernels w/ ReLU
Conv3-2	PointConv 144 kernels w/ ReLU
Conv3-3	PointConv 144 kernels w/ ReLU
Conv3-4	PointConv 144 kernels w/ ReLU
Subsampling3	voxel subsampling with voxel size 0.2
Conv4-1	PointConv 192 kernels w/ ReLU
Conv4-2	PointConv 192 kernels w/ ReLU
Conv4-3	PointConv 192 kernels w/ ReLU
Conv4-4	PointConv 192 kernels w/ ReLU
Subsampling4	voxel subsampling with voxel size 0.4
BottleNeck Conv	PointConv 240 kernels w/ ReLU
Upsampling4	retrieve results before Subsampling4
DeConv4	PointDeConv 192 kernels w/ ReLU
Upsampling3	retrieve results before Subsampling3
DeConv3	PointDeConv 144 kernels w/ ReLU
Upsampling2	retrieve results before Subsampling2
DeConv2	PointDeConv 96 kernels w/ ReLU
Upsampling1	retrieve results before Subsampling1
DeConv1	PointDeConv 48 kernels w/ ReLU
Output Conv	1x1 convolution
Softmax layer	

Table 3: The network architecture (16-layer) for ScanNet experiments. Every convolutional layer is followed by a ReLU layer.

Method	**Feedforward time** (seconds per frame)	**Preprocessing time** (seconds per frame)	**Parameters** (millions)	**Maximum inference points** (millions)
PointNet [5]	0.047	-	0.8	0.49
PointNet++ [6]	2.41	-	0.97	0.98
PointCNN [4]	2.00	-	11	0.05
SPG [3]	10.70	-	0.25	–
KPConv [7]	0.18	0.83	14.9	0.54
RandLA-Net [2]	0.045	0.013	1.24	1.03
PointConv [9]	0.13	0.20	15.4896	1.4
VI-PointConv (ours)	0.15	0.20	15.4914	1.4

Table 4: The computation time, network parameters and maximum number of input points of different approaches for semantic segmentation on Sequence 08 of the SemanticKITTI [1] dataset.

Rotation angle	60k	40k	20k	10k	5k
0°	60.8	59.7	55.3	44.4	32.5
90°	60.8	59.7	54.8	44.7	32.1
180°	60.3	59.6	54.7	44.7	32.4
270°	60.8	59.6	55.2	45.0	32.0

Table 5: Performance results (mIoU,%) for the settings of KNN, VI descriptors as inputs for the MLP, and the network is trained with rotation augmentation.

Rotation angle	60k	40k	20k	10k	5k
0°	58.7	58.0	54.5	46.1	36.3
90°	58.7	58.0	53.6	46.0	36.3
180°	59.0	57.2	54.1	46.6	35.6
270°	58.1	57.9	54.5	46.3	36.2

Table 6: Performance results (mIoU,%) for the same settings as Table 5, expect that the network is trained without rotation augmentation.

Rotation angle	60k	40k	20k	10k	5k
0°	58.8	53.5	35.0	17.9	10.0
90°	58.6	53.2	34.3	17.6	9.9
180°	58.6	53.4	34.6	17.9	10.0
270°	58.8	53.5	34.5	17.8	9.9

Table 7: Performance results (mIoU,%) for the settings of KNN, (x, y, z) coordinates as inputs for the MLP, and the network is trained with rotation augmentation.

Rotation angle	60k	40k	20k	10k	5k
0°	54.2	51.1	40.4	27.3	16.1
90°	53.8	50.9	40.6	27.4	15.7
180°	54.0	51.1	40.6	27.3	16.1
270°	54.1	51.6	40.6	27.6	16.1

Table 8: Performance results (mIoU,%) for the same settings as Table 7, expect that the network is trained without rotation augmentation.

PointConv and VI-PointConv, however has significantly worse performance on the dataset.

References

[1] 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 *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[2] 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. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[3] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[4] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 820–830. Curran Associates, Inc., 2018.

[5] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016.

[6] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 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. *Proceedings of the IEEE International Conference on Computer Vision*, 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)*, 2019.

[9] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.