Attention EdgeConv For 3D Point Cloud Classification

Yen-Po Lin, Yang-Ming Yeh, Yu-Chen Chou, Yi-Chang Lu
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, 106319 Taiwan
{r07943127, d05943006, r07943018, yiclu}@ntu.edu.tw

***Abstract*—Due to the irregular and unordered properties of 3D point cloud data, it is more challenging to extract geometric features between points. In this paper, we propose techniques to improve the ability of capturing point cloud features, so that higher accuracy and better stability of point cloud classification tasks can be achieved. First, we design two attention modules: the Point-wise Attention Module determines the correlation between points, and the Channel-wise Attention Module allows the model to focus on important features with limited resources. With these attention modules, we not only achieve the state-of-the-art accuracy of 93.7% on the ModelNet40 dataset but also reduce the error rates on the ScanObjectNN dataset. Secondly, we propose a guideline to dynamically adjust the size of the KNN. Using the proposed Dynamic-K method, we can significantly increase the accuracy of classification when dealing with low-resolution objects.**

***Index Terms*—3D Point Cloud Classification, Attention Mechanism, Deep Neural Network**

I. Introduction

In recent years, demands in automated driving technology make a rapid growth of 3D point cloud research. 3D point cloud data have the following properties that make them difficult to process:

1) Unorderedness: 3D point cloud data are simply a collection of points without particular order. An object consisting of N points has $N!$ possible representations. Therefore, the classifier must be invariant to the order, or it may cause different results for the same object in different descriptive orders.
2) Irregularity: 3D point cloud data are arranged irregularly. Each point in a point cloud object is not confined to a fixed grid location, so the relative positions and distances between points are nonuniform. This leads to the failure of most traditional methods applied to 2D images, such as convolution operations and sliding windows.
3) Sparsity: In robotics and autopilot scenarios, a LiDAR sensor scans the surrounding scene with scan lines. Its coverage of sampling points can be very sparse relative to the scale of the scene, which makes it impossible to extract features efficiently after converting point cloud to a standard grid.

To overcome issues caused by these properties of point cloud data, several approaches have been proposed, such as PointNet [1], PointNet++ [2], DGCNN [3], SpiderCNN [4], and RS-CNN [5]. In this paper, we propose techniques to improve the results of the object classification task on point cloud data, and we summarize our contributions as follows:

- We significantly improve the ability to capture 3D point cloud geometry features by the Point-wise Attention Module (PAM) and the Channel-wise Attention Module (CAM). Using these two attention mechanisms, we can obtain state-of-the-art results on both ModelNet40 [6] and ScanObjectNN [7] datasets.
- We propose the Dynamic-K method, which significantly improves the robustness of the KNN-based classification when facing low-resolution objects.

II. Related Works

Volumetric Methods: Due to the excellent feature capturing ability of convolutional operations on standard grids, it is intuitive to convert irregular and unordered data types (i.e. point clouds) into standard volumetric grids. Through the conversion, 3D convolution can be directly applied to point cloud data. Well-known methods such as 3D ShapeNets [6] and VoxNet [8] fall into this category.

However, volumetric methods have two drawbacks: high memory consumption and high computational complexity. These disadvantages limit the volumetric methods to be only suitable for low-resolution input data. As a result, Wang *et al.* proposed O-CNN [9], which first divides a space into eight quadrants and stores the information of whether there is a point in these eight quadrants. If the eight quadrants are all empty, the octree will not have any child nodes. Otherwise, it goes into a quadrant and repeats the above process. By doing so, it can save a lot of memory space. This allows the O-CNN [9] to construct a deeper neural network and further increase the input resolution. While these methods are effective in resolving the problem of irregular alignments and unorderness, they are still not as powerful as those methods that work directly with point cloud data.

Direct Processing of Point Cloud Objects: The first study that directly deals with point cloud data is PointNet [1], which uses symmetric operations and shared Multi-Layer Perceptron (MLP) to handle the unordered and irregular properties of point cloud data. In order to improve the stability of the model for various transformations, PointNet also introduced a framework called T-Net. The goal of T-Net is to simulate various affine transformations in the training process to make the model more robust.

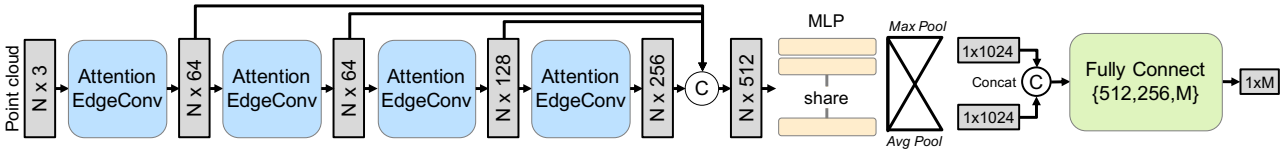
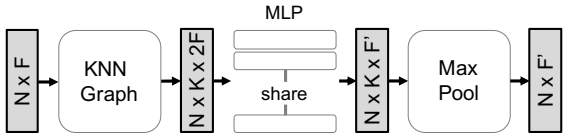
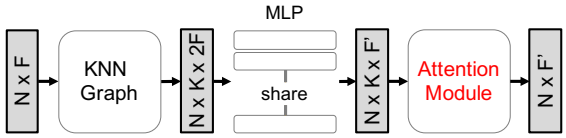


Fig. 1: The proposed network architecture: It takes an object with N points as the inputs and calculates the edge features by Attention EdgeConv. After four layers of Attention EdgeConv, our network aggregates all of the features to obtain the global features. In the last step, our network generates classification scores for M classes by fully connected layers.



(a) EdgeConv [3]



(b) Attention EdgeConv

Fig. 2: Comparison between the EdgeConv and our proposed Attention EdgeConv.

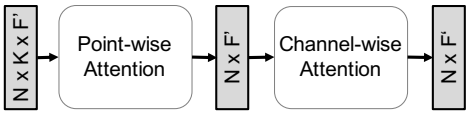


Fig. 3: Attention Module: each Attention Module consists a Point-wise Attention Module (PAM) and a Channel-wise Attention Module (CAM).

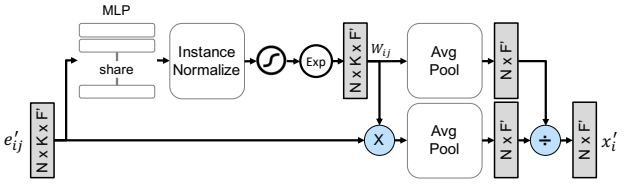


Fig. 4: Point-wise Attention Module (PAM)

Although PointNet performs well, it has certain drawbacks. The most obvious one is that it only captures global features and does not take the correlation between points into account. This limits its ability in capturing local features. Later many studies have aimed to improve the problem of lacking local messages in PointNet. PointNet++ [2] constructs a hierarchical neural network to extract features recursively via the designed sampling and grouping layers. Wang *et al.* proposed a method named DGCNN [3], which constructs KNN Graphs in each layer to capture the correlation between points in different feature dimensions. Liu *et al.* proposed RS-CNN [5], which randomly selects the neighbors within a certain radius in order to make the model more robust to non-uniformly sampled objects.

Inspired by the above-mentioned studies, in the following sections, we introduce new schemes to further improve the results of point cloud classification tasks.

III. The Proposed Design

The proposed network architecture is shown in Fig. 1. The details of our design are discussed in the following subsections.

A. Attention EdgeConv

EdgeConv, proposed by [3], is an effective method for capturing local information. When calculating features of one particular point, EdgeConv takes the information of that point and its K nearest points. With this technique, the points can form a small local graph within a small area, providing local features. The experiments have shown that combining global and local features to update the features of each point is effective.

However, the last operation in EdgeConv is Max Pooling, which does not well preserve the features of the edges. In response to this observation, we propose two attention modules to replace the Max Pooling. The two attention modules, a Point-wise Attention Module (PAM) and a Channel-wise Attention Module (CAM), can enhance the ability of EdgeConv in capturing geometric features. The PAM aims to edge weights while the CAM focuses on distinctive features under limited computing resources. The architecture of Attention EdgeConv and the design of Attention Module are shown in Fig. 2 and Fig. 3, respectively.

1) Point-wise Attention Module (PAM): The Max Pooling operation in EdgeConv aims to extract the features of an edge with the largest correlation regarding the center point. However, this causes the loss of feature information for the remaining $K-1$ edges. Therefore, we have the model learn a set of weights for each edge so that all edges can provide the local information.

To learn the weights of the edges, we design a spatial attention mechanism for point cloud objects, named Point-wise Attention Module (PAM), as shown in Fig. 4.

Our proposed PAM first uses a shared MLP f_θ to obtain a set of weights. Then we use Instance Normalization and Sigmoid modules in series, expressed as σ_+ here, to make the model more efficient and stable. After that, we apply an

TABLE I: Classification results on ModelNet40 [6] dataset: +norm denotes data with normal vectors; +voting denotes post-processing with multi-votes is applied.

Architecture	Input Type	# Points	Avg Class Acc	Overall Acc	Params
PointNet [1]	xyz	1k	86.0%	89.2%	3.5M
Kd-Net [10]	xyz	1k	-	90.6%	-
PointCNN [11]	xyz	1k	88.1%	92.2%	0.6M
PCNN [12]	xyz	1k	-	92.3%	8.2M
DensePoint [13]	xyz	1k	-	92.8%	-
DensePoint [13] + voting	xyz	1k	-	93.2%	-
DGCNN [3]	xyz	1k	90.2%	92.9%	1.84M
KPConv [14]	xyz	1k	-	92.9%	14.3M
RSCNN [5]	xyz	1k	-	92.9%	1.41M
RSCNN [5] + voting	xyz	1k	-	93.6%	1.41M
Ours	**xyz**	**1k**	**91.1%**	**93.7%**	**1.95M**
SO-Net [15]	xyz	2k	87.3%	90.9%	-
PointNet++ [2]	xyz + norm	5k	-	91.9%	1.48M
SpiderCNN [4]	xyz + norm	5k	-	92.4%	-
SO-Net [15]	xyz + norm	5k	90.8%	93.4%	-
DGCNN [3]	xyz	2k	90.7%	93.5%	1.84M

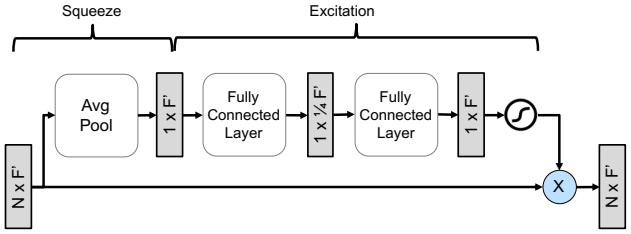


Fig. 5: Channel-wise Attention Module (CAM)

exponential function to the weights to ensure that non-negative values can be attained. Input edge features e'_{ij} for the point i are multiplied with these weights W_{ij} and then normalized to produce the weighted features x'_i. The equation is shown below:

$$x'_i = \frac{\sum_{j=1}^{\mathrm{K}} W_{ij} \times e'_{ij}}{\sum_{j=1}^{\mathrm{K}} W_{ij}}, \tag{1}$$

where

$$W_{ij} = F\left(e'_{ij}\right) = \exp\left(\sigma_+\left(f_\theta\left(e'_{ij}\right)\right)\right). \tag{2}$$

2) Channel-wise Attention Module (CAM): In addition to the PAM that helps us determine the weight of each edge, we adopt the Squeeze-and-Excitation technique, a channel-wise attention mechanism proposed by [16], to extract more features without adding too many resources. The CAM is shown in Fig. 5, which can be split into two main parts.

First is the Squeeze part, where we take an average of the information from all points. This is to make use of the correlations between channels rather than correlations in the spatial distribution. This can also block out the information spatially distributed, making the model more reliable in terms of scale.

The Excitation part is implemented with two fully connected layers. The first fully connected layer f_ϕ with ReLU function compresses the total F' channels into F'/r channels to reduce the amount of computation. The second fully connected layer g_ϕ reverts the F'/r channels back to the F' channels, and then uses the Sigmoid function σ to limit the output weights ranging from 0 to 1. Here, r is the reduction ratio.

$$s = \sigma\left(g_\Phi\left(ReLU\left(f_\phi\left(x'_i\right)\right)\right)\right). \tag{3}$$

The final output y_i of the PAM is obtained by rescaling the input x'_i with the scalar s:

$$y_i = s \cdot x'_i. \tag{4}$$

B. The Dynamic-K Method

In order to obtain local geometric information, KNN is usually adopted to construct a local receptive field [3], [11], [15]. However, this receptive field varies not only with the k value in KNN but also with the number of input points. Previous methods selected the KNN size during the training stage and fix the size for evaluation. As the result, when dealing with low-resolution objects, this scheme may cause a significant change in the receptive field as shown in Fig. 6. Thus a notable decrease in classification accuracy [17] would be observed.

To overcome the above issue, we propose the Dynamic-K method, which allows the model to select the size of KNN during the evaluation stage following the equation below:

$$K_{test} = (K_{train} - Bias) \times \frac{N_{test}}{N_{train}} + Bias, \tag{5}$$

where N_{train} and N_{test} are the numbers of input points in the training and testing stages, while K_{train} and K_{test} are the sizes of the KNN models. As for $Bias$, it is used for setting the smallest size of the KNN. In our experiment, we set K_{train} to 20, N_{train} to 1024 and $Bias$ to 5. After applying the Dynamic-K method, we can keep the receptive field almost unchanged for low-resolution objects. Qualitative experimental results are shown in Section IV-E.

TABLE II: Classification results on ScanObjectNN [7] dataset. * : reported by [7].

	OBJ_ONLY	OBJ_BG	PB_T25	PB_T25R	PB_T50R	PB_T50_RS
3DmFV [18]*	73.8%	68.2%	67.1%	67.4%	63.5%	63.0%
PointNet [1]*	79.2%	73.3%	73.5%	72.7%	68.2%	68.2%
SpiderCNN [4]*	79.5%	77.1%	78.1%	77.7%	73.8%	73.7%
PointNet++ [2]*	84.3%	82.3%	82.7%	81.4%	79.1%	77.9%
DGCNN [3]*	**86.2%**	82.8%	83.3%	81.5%	80.0%	78.1%
PointCNN [11]*	85.5%	**86.1%**	83.6%	82.5%	78.5%	78.5%
Ours	84.2%	83.6%	**83.8%**	**82.7%**	**81.0%**	**79.7%**

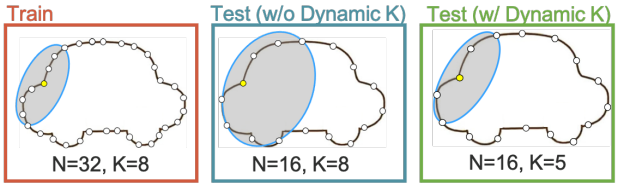


Fig. 6: An example of the Dynamic-K method. During the training process, the size of KNN is 8, the number of points is 32. If a lower- resolution object with 16 points is tested during evaluation, the receptive field would be inconsistent with one in the training phase. After applying the proposed Dynamic-K method, the receptive field will be controlled to a similar size as in the training phase.

C. Our Network Architecture

As shown in Fig. 1, we stack the neural network in the same way as DGCNN [3] to verify the effectiveness of the attention module. We use four Attention EdgeConv layers with channel sizes of 64, 64, 128, and 256 to capture the geometry features. We also use the same Dynamic Graph method as DGCNN. For each Attention EdgeConv layer, we reconstruct a new KNN Graph based on the current features to capture geometric relationships in different feature dimensions. In the training phase, we fix the size of KNN to 20. In addition, shortcut connections are included to extract multi-scale features, and the following shared MLP layer aggregates these features. Next, we use both Max Pooling and Average Pooling to obtain global features. Finally, we use three fully connected layers with 50% dropout as the classifier, where LeakyReLU and Batch Normalization [19] are used in each layer.

IV. Experiments

A. Training Details

We implement our network in Pytorch and perform 250 epochs training with batch size set to 32. To optimize the model weights, we use SGD with the initial learning rate of 0.1, momentum of 0.9, and weight decay of 0.0001. The learning rate is reduced to 0.001 using the strategy of Cosine Annealing [20], and our network is trained on two RTX2080Ti GPUs with 64 GB of DDR4 memory.

B. The ModelNet40 Result

We summarize the ModelNet40 results in Table I. Our architecture outperforms all xyz-input architectures on the ModelNet40 dataset. In addition, our method even outperforms SO-Net [15], which uses rich information such as $5K$ inputs with normal vectors. We also achieve higher scores compared to RS-CNN [5], which uses the post-processing of 10-vote evaluations that is repeated 300 times during testing.

C. The ScanObjectNN Result

We summarize the ScanObjectNN result in Table II. In OBJ_ONLY and OBJ_BG, due to the small amount of data (only 2,902), our method suffers from the overfitting issue, which results in worse performance. However, in PB_T25, PB_T25_R, PB_T50_R, and PB_T50_RS, we achieved 83.8%, 82.7%, 81.0%, and 79.7% of accuracy, respectively, which outperformed all previous methods. This proves that our proposed method can perform well in synthetic datasets and real-world objects with various perturbations.

D. Ablation Study

We remove each part of the attention module and observe the effect on the classification results on ModelNet40 [6] dataset. The experiment results are shown in the Table III. According to the result, the combination of the PAM and the CAM produces the best performance. It should be noted that once all the PAM and CAM designs are removed, the proposed model will become DGCNN [3].

TABLE III: Effect of different parts of the attention module.

	Params	Overall Acc
DGCNN	1.84M	92.9%
PAM	1.90M	93.2%
CAM	1.85M	93.3%
PAM + CAM	1.95M	**93.7%**

E. Robustness Test

We test the robustness of our architecture on ModelNet40 [6] dataset. We use low-resolution objects as inputs to our model, which is trained with complete objects (1024 points). For a fair comparison, random sampling is applied to the input data. We summarize the results in Fig. 7. As demonstrated, our architecture outperformes DGCNN at all kinds of resolutions even without the use of the Dynamic-K method. After applying the Dynamic-K method, the accuracy numbers are further improved. The results demonstrate the effectiveness of the Dynamic-K method.

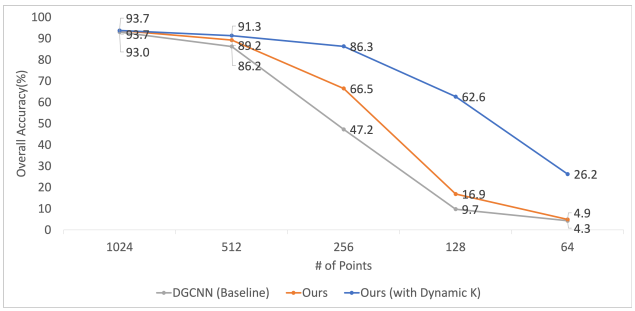


Fig. 7: Robustness test of our architecture on various number of input points. The model is trained with 1024 points and tested with random input subsampling.

V. Conclusion

In this work, we propose two different types of attention modules and the Dynamic-K method. The Point-wise Attention Module helps the model capture more accurate geometric features, while the Channel-wise Attention Module helps the model retain important features with limited resources. The Dynamic-K method allows our model to maintain the receptive field and therefore good performance even for low-resolution point cloud objects. Combining the two attention modules and the Dynamic-K method facilitates our model to surpass the state-of-the-art results.

Acknowledgment

This work is partially supported by National Taiwan University under Grant number 110L890605, and the Ministry of Science and Technology, Taiwan, under Grant numbers 107-2221-E-002-123-MY2 and 110-2221-E-002-097. The authors would also like to thank National Center for High-performance Computing (NCHC) for providing computational and storage resources.

References

[1] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[2] 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*, 2017, pp. 5099–5108.

[3] 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*, vol. 38, no. 5, pp. 1–12, 2019.

[4] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 87–102.

[5] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8895–8904.

[6] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1912–1920.

[7] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1588–1597.

[8] Daniel Maturana and Sebastian Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2015, pp. 922–928.

[9] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–11, 2017.

[10] Roman Klokov and Victor Lempitsky, "Escape from cells: Deep kd-networks for the recognition of 3d point cloud models," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 863–872.

[11] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in Neural Information Processing Systems*, 2018, pp. 820–830.

[12] Matan Atzmon, Haggai Maron, and Yaron Lipman, "Point convolutional neural networks by extension operators," *arXiv preprint arXiv:1803.10091*, 2018.

[13] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan, "Densepoint: Learning densely contextual representation for efficient point cloud processing," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5239–5248.

[14] 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*, 2019, pp. 6411–6420.

[15] Jiaxin Li, Ben M Chen, and Gim Hee Lee, "So-net: Self-organizing network for point cloud analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9397–9406.

[16] Jie Hu, Li Shen, and Gang Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[17] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski, "Monte carlo convolution for learning on non-uniformly sampled point clouds," *ACM Transactions on Graphics*, vol. 37, no. 6, pp. 1–12, 2018.

[18] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer, "3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3145–3152, 2018.

[19] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[20] Ilya Loshchilov and Frank Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.