Complementary Multi-Branch CNNs Towards Real-World 3D Point Classification

Zifeng Tang[1,2#] , Fusheng Hao[1,2#], Qieshi Zhang[1,2*], Jin Zhang[1,2], Jun Cheng[1,2], Nan Li[1,2]

***Abstract*— Autonomous driving requires precise and efficient 3D point clouds processing techniques, where deep learning has shown great potential. Nonetheless, most of the existing works achieve high accuracy in synthetic data while performing unsatisfactorily in real-world datasets. On the other hand, though existing multi-view-based, volumetric-based, and point-based methods have achieved promising results, they still suffer from high computational complexity or low memory utilization efficiency. For example, the computational time and memory consumption of multi-view-based and volumetric-based methods increase at most cubically with the increasing of the input resolution, while half of the point-based methods' runtime is wasted due to their irregular memory access. To address these issues, we propose a novel method to fuse complementary CNNs, dubbed as Multi-Branch Convolutional Neural Networks (MBCNN), which achieves the current state-of-the-art performance while maintaining high efficiency. The underlying design of MBCNN is to utilize the point-based convolution as a local-feature branch and spherical convolution as a global-feature branch, to capture patterns in space and exploit the merit of the pointwise Multilayer Perceptron (MLP), which is robust to the perturbation. Besides, we design an efficient voxel-based indexing technique to prevent the feature-extracting phase from being blocked, thus significantly accelerating the procedure of finding a point's neighbor. Experimental results show that MBCNN achieves state-of-the-art performance on the real-world 3D point clouds dataset while reducing 30% of runtime in neighbor indexing.**

I. INTRODUCTION

3D data from the real world is usually captured by the LiDAR sensor in the format of point clouds, which is easy to be acquired and not sensitive to the ambient environment. Since the format is non-uniform and orderless, conventional ways (including 3D ShapeNets [1] and 3D-UNet [2]) transform each point to a relevant voxel grid by a specific mapping, followed by a 3D volumetric convolution to extract the feature. However, such methods are restricted due to the computational complexity of $O(N^3)$ as well as the massive memory space required. Since volumetric-based models can only improve the performance by raising the resolution, a cubical increase of memory will be incurred. For example, 3D-UNet [2] is required to be allocated more than 10 GB of GPU memory on a resolution of $64 \times 64 \times 64$ with the batch size of 16 [3]. 3D data could also be processed by another stream of traditional models, multi-view based models [4]. In these models, a 3D shape is firstly projected in multiple views. Then, features are extracted view-wise and aggregated for comprehensive analysis. The key challenge for multi-view based methods is how to fuse view-wise feature effectively. Overall, both streams of models mentioned above introduce explicit information loss and work inefficiently due to their unavoidable transformation before the process of feature extraction.

TABLE I
DIFFERENT STREAMS OF MODELS

Methods	Extra Loss	Local Feature Extraction	Memory Cost	Irregular Memory Access
Voxel-based Model	√	√	High	
Multi-view based Model	√	√	High	
Point-based MLP Model			Low	
Point-based Convolutional Model		√	Low	√

* We summarize the various characters from different streams of models.

In contrast, point-based models directly process original input points without any transformation, avoiding extra information loss. Qi et al. proposed PointNet [5], which is regarded as a pioneering work of point-based methods [4]. Generally, The point-based models in the early phase (pointwise Multi-Layer Perceptron (MLP) Methods) utilize a shared MLP that directly consume every single point, then followed by a max-pooling, which reserve the permutation invariance of the input points, and a perturbation to the input point clouds would not greatly change the output. However, point-based models tend to perform badly in consuming some large-scene datasets, since pointwise MLP networks neglect the extraction of features on the spatial relationship between points. To address this problem, a hierarchical network was proposed by Qi et al. [6] to extract fine geometric information from each point's neighborhood. Point-based convolution methods, such as RS-CNN [7], Kernal Point Convolution (KPConv) [8] etc., as another branch of point-based models start thriving. Similar to a conventional convolutional network, the models of this branch normally define convolutional on space, where the weights for neighbor points are determined by the geometric relationship. The introduction of convolution makes models acquire equivariance, which increases the generalization ability of models. However, the procedure of searching neighbor points is time-consuming. Up to 80% of the total time is spent on

* Corresponding authors
#These authors contribute equally to the work.
[1]CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems Shenzhen Institutes of Advanced Technology, CAS, China
[2]The Chinese University of Hong Kong, Hong Kong, China
qs.zhang@siat.ac.cn

structuring orderless input points, which involves frequent random memory access, instead of extracting features [3]. Such random memory access needs to be avoided since it can result in a cache miss. Moreover, some fusion-strategy based models such as Point-Voxel CNN (PVCNN) [3] and Sparse Point-Voxel Convolution (SPVCNN) [9] have been developed. Combining the advantages: point-based methods and volumetric-based methods, alleviating the large memory footprint and frequent irregular memory access problems.

Additionally, there are many other schemes have also been proposed, including Spherical CNNs [10]. Though similar to multi-view based models, in Spherical CNNs, the information of the 3D object is projected to a single sphere rather than multiple planes. Then, the spherical convolution is applied to the sphere and its rotation group (SO(3)), which is theoretically solid and shows excellent performance in the perturbed dataset due to its rotational equivariance [10]. One of the limits of these models is that it cannot process point clouds directly due to its implementation.

We design a novel feature-fusion strategy, Multi-Branch CNN (MBCNN) for 3D shape classification to ensure models with high accuracy and low latency at the same time. Being inspired by the PVCNN and concept of ensemble learning, we construct a multi-branch model, including local-feature, global-feature, and point-feature branch. For the local-feature branch, we propose a novel *voxel-based indexing* approach to find neighbors of specific points, which contribute to reduce computation complexity and mitigate the effect of irregular memory access. A point-based convolution is used to extract the local patterns embedded in the indexed point clouds. For the global-feature branch, we propose a novel *point-based spherical CNN*, that projects the normalized points to a sphere and exploits the spherical convolution to capture the global patterns in point clouds. For the point-feature branch, we exploit the pointwise MLP method to obtain each points' information. Overall, the underlying design of MBCNN is to employ the local-feature and global-feature to capture the patterns in space and exploit the merit of the pointwise MLP that resilient to the perturbation in a point set so that the MBCNN architecture could achieve great performance in the real-world datasets with interference. The main contributions of this work are the following:

- Conventional point-based convolution models are accelerated by utilizing the voxel-based indexing, which reduces 30% time in searching neighbors.
- A point-based spherical CNN is proposed to directly process point clouds.
- In MBCNN, a fusion strategy is utilized to gain features more comprehensively. We obtain an accuracy of 81.5% in the hardest subset of ScanObjectNN [11], surpassing the state-of-the-art methods.

II. Related Work

A. Point Based Methods

Due to its high efficiency and effectiveness, PointNet [5] is considered as a milestone in 3D deep learning, initiating the era of Point-based 3D models. These models exploit the characters of symmetric function, such as max-pooling, to process orderless points in point cloud data, which could facilitate the robustness of the model, since the network learns to summarize a shape by a sparse set of key points [5], and the output of this model should not significantly change due to a perturbation. However, PointNet is limited, since it lacks the ability to extract local spatial relationship among the points. To solve this drawback, other research tried to pile multiple layers of PointNet in [6] [12] [13]. Current point-wise MLP models with only layers of perceptron might not work well in dealing with large-scene tasks but can support other learning methods by providing robust supplementary information. Moreover, 3D convolution methods [8], [14]–[17] are introduced as new options, defining convolutional kernels on a continuous space, where the weight of each point is determined by its spatial distribution. Especially, KPConv shows high performance in extracting the local relationship between neighborhood points, achieving high accuracy in several datasets, while its computational complexity is significantly lower than traditional voxel-based convolution.

In these point-based methods, although the expressiveness is improved by aggregating the neighborhood information, it will result in high computational cost and irregular memory access. To accelerate the indexing procedure, a KD-tree is utilized in some models, which could reduce the computational cost of finding neighbor points from $O(N^2)$ to $O(Nlog(N))$. However, due to its tree structure, which involves frequent irregular memory access, which means that the required data probably has not been stored in the cache yet, and finally results in a cache miss and increases the latency (Fig. 1). Thus, finding a more efficient indexing algorithm is meaningful.

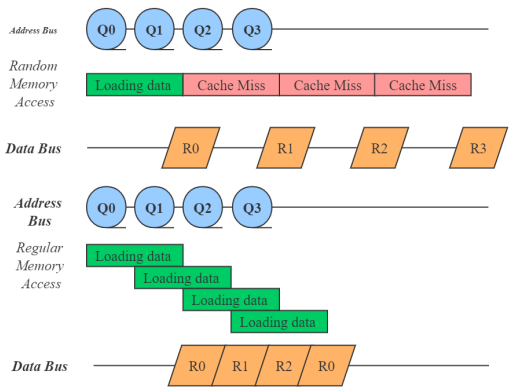


Fig. 1. **Cache miss caused by irregular memory access.** Irregular memory access is inefficient since it will cause cache miss, which means the data is frequently loaded from memory instead of the high-speed cache. As CPUs become much faster compared to main memory, stalls caused by cache miss show more potential computation.

B. Group Equivariant CNNs

Cohen et.al propose a series of group equivariant convolution [10] [18] [19], which was initially proposed to

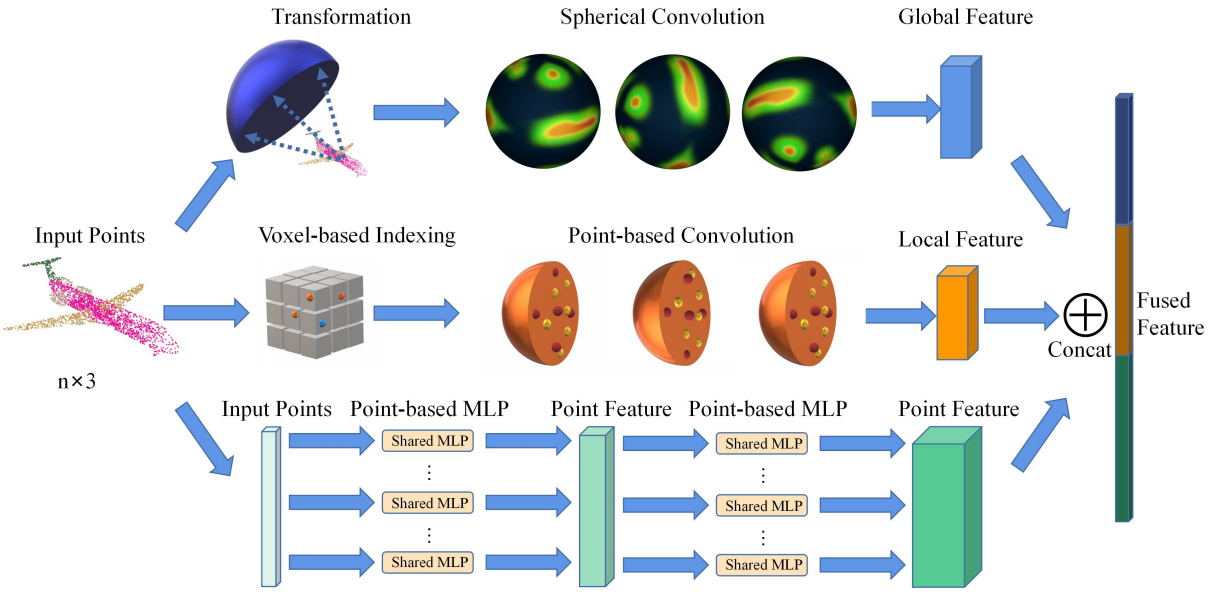


Fig. 2. **The fusion of 3 various branches.** MBCNN is composed of a point-feature branch (lower branch), a local-feature branch (middle branch) and a global-feature branch (upper branch).

augment convolution with rotation equivariance [10]. It was not utilized for 3D object classification untill [10], which defines a novel spherical convolution.

Though spherical convolution's rotation equivariance shows great potential in 3D deep learning, Cohen, et al.'s experiment [10] in Spherical CNNs is based on the synthetic 3D object (mainly constructed by polygon meshes). By utilizing a set of straight lines from the centroid to the sphere, the 3D meshes are projected to an enclosing sphere. For each intersection between line and mesh, three kinds of signals are collected: the distance between intersection and sphere, the inner product and outer product between the mesh's normal and line's direction vector. Thus, data preprocessing in [10] is not applicable in the real-world task, due to the data format is mainly point clouds. Therefore, it is worth finding a new method to preprocess point data with spherical convolution.

C. Ensemble Learning

A collection of various models working as a unit is called ensemble learning. Normally, ensemble learning is utilized to improve the performance of a model or reduce the likelihood of an unfortunate selection of a poor one [20]. By using ensemble learning, models tend to have lower error and be less over-fitting. A voting classifier is one of the simplest ways to combine multiple models. Technically, it is a wrapper rather than an actual classifier, which allows various models to be trained and evaluated in parallel to exploit the distinction of each model.

III. Multi-Branch CNN

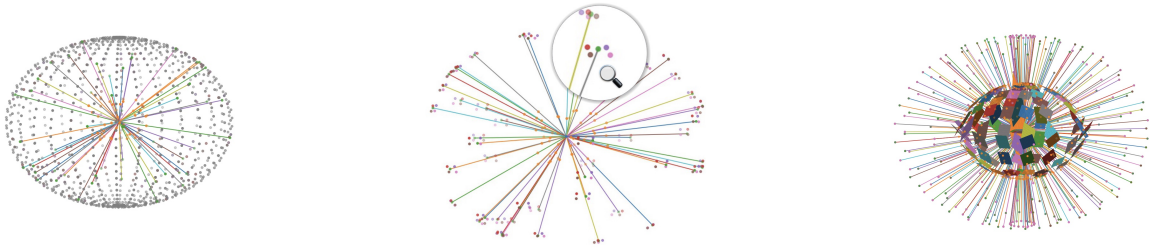
We managed to construct a fusion architecture named Multi-Branch CNN (MBCNN) to combine various features from different perspectives. From the pointwise perspectives, we utilize MLP to extract the point feature of every single point. To ensure that the model is capable of detecting the local relationship between points, a point-based Convolution is introduced as a local-feature branch in our MBCNN. In the end, a spherical convolution is employed to aggregate the global feature as supplementary signals to guide the classification in 3D deep learning (Fig. 2).

A. Local-Feature Branch

Since Kernal-Point convolution performs well in learning local signals. We utilize Kernal-Point convolution as a coarse branch in feature extraction. Nevertheless, KPConv is still facing a high computational cost due to unavoidable irregular memory access.

1) Voxel-Based Indexing Methods: To solve the irregular memory access problem, methods like KD-tree are proposed, its computational complexity is $O(Nlog(N))$, which is faster than the conventional Brute-force search. Despite that, the irregular memory access of searching neighbors could reduce the efficiency of GPU due to its waiting. Thus, we propose a novel idea to index the neighborhood points in point clouds to accelerate the process of finding neighbors (Fig.

Fig. 3. **Voxel-Based Indexing**



(a) Projection from points to sphere (b) Interpolation. (c) Constructing planes by linear regression

Fig. 4. **Transformation of the input points**

3). Being inspired by the voxel-based convolution, we firstly constructed a mapping between each point and its relative voxel grid. Then, we can avoid search all the points again in finding neighborhood points (Fig. 3), by simply searching the neighborhood voxel grids. Overall, for finding each point's neighbor, the computational complexity is significantly reduced from $O(Nlog(N))$ to $O(N)$. In terms of the implementation, a three-dimensional array is constructed, and each element contains a linked list or an array. When each element contains a linked list, there is no too much memory would be wasted due to the pointer's lightness. Nonetheless, by such implementation, we only reduce the computation complexity, yet still have an irregular problem. On the other hand, when each element is an array, we can avoid frequent irregular memory access, but facing a larger memory footprint. Based on the Eq. 1, we can easily transform the points to a relevant grid depending on their coordinate, where r denotes the resolution of the voxel, $\mathbb{I}[\cdot]$ refer to a binary indicator to judge whether the point falls into the voxel grid, x_k, y_k, z_k denote the coordinates of each point, u, v, w denote the index of the corresponding grid:

$$\begin{aligned} V_{u,v,w} = \mathbb{I}[&floor(x_k \times r) = u, \\ &floor(y_k \times r) = v, \\ &floor(z_k \times r) = w]. \end{aligned} \tag{1}$$

The computational complexity is both $O(N)$ during insertion of a new point and the search of neighbors.

2) Point-Based Convolution: We utilize Kernal Point Convolution in our local-feature branch. For each point, KPConv selects its neighbor points within a specific radius as input, then convolve them with some predefined kernel points with different weights. Additionally, a deformable kernal is introduced, enabling the model to automatically adapt to the input data geometry.

B. Global-Feature Branch

The main challenge of combining the spherical convolution is that the original Spherical CNNs framework is not based on point clouds. Thus, in analogy to data sampling methods in Spherical CNNs, we design a new spherical-convolution-based framework, to directly extract features from point clouds. Eventually, the feature extracted by Spherical CNNs is combined as a global feature.

1) Transformation:

- **Projecting points to sphere:** In analogy to the sampling method in Spherical CNNs, we design an inverse render model (Fig. 4(a)), which inversely send a ray from input points to the sphere. In the beginning, we initiate an enclosing sphere with grids. Then for each point, there is a ray sent from the centroid to this point and finally, intersect with the sphere on a point.
- **Interpolation:** Based on an intersection, we can use interpolation to share the value (the distance between sphere and point) with the four nearest grids on the sphere depending on their relative weights (Fig. 4(b)). The weight of each grid is determined by the euclidean distance between the intersection and grid Eq. (2), since the geodesic distance is approximately equal to Euclidean distance when the grids are dense, $Weight_i$ denotes the weight of each grid, $grid_i$ denotes the location of each grid, p denotes the location of the intersection:

$$Weight_i = \frac{\|grid_i - p\|_2}{\sum_{i=0}^{3} \|grid_i - p\|_2}. \tag{2}$$

- **Linear Regression:** As mentioned before, in Spherical CNNs, the angle of each intersection between meshes' normal and ray is utilized to augment the raw data. However, there is no concept of "mesh" and its normal in point clouds. Under this circumstance, we need to construct the angle features manually. In the last step, through interpolation, each grid contains a value indicating the distance between sphere and point. According to the distance information, the shape of the point clouds can be estimated: based on a grid and other 8 grids around it, a related position within the sphere can be computed, then we can use these 9 points for linear regression. Finally, the regression output a plane that is almost equal to mesh, and an angle signal with the plane's normal (Fig. 4(c)). It is worth mentioning that during the implementation of linear regression, we did not directly use the sklearn library since it is too slow. Instead, we independently implement a C extension by simply calculating an inversed matrix Eq. (3) to get the coefficients (when $X^T X$ is a full-rank matrix):

$$\hat{w} = (X^T X)^{-1} X^T y, \tag{3}$$

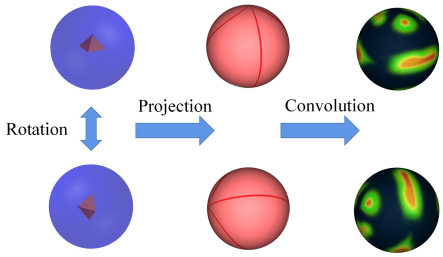


Fig. 5. **Rotation equivariance of Spherical CNNs.** In this figure, we show that when the object is rotated, the output of convolution is transformed accordingly.

and it is 20,000 times faster than directly use sklearn library.

2) Spherical Convolution: After transformation, the grids on the sphere have been assigned with three-channel features. Accordingly, the spherical convolution is applied to the feature due to the isomorphic space of rotation on the sphere, outputting features that are on a SO(3). It is proved that spherical CNNs show excellent performance in the classification of the perturbed 3D object due to its equivariance Fig. 5. The equivariance of spherical convolution can be defined in Eq.(4), where L_R denotes the rotation operator, f and ψ denote an input feature map and a filter respectively. There is a commutation relation between f and ψ, which indicates that the performance of spherical convolution is not affected by rotation:

$$[[L_R f] * \psi](x) = [L_R[f * \psi]](x). \tag{4}$$

C. Point-Feature Branch

Several shared MLP layers are used to extract pointwise features independently, followed by a max-pooling layer to aggregate the most important features, which is insensitive to real-world noises, serving as robust supplementary information for the global and local features.

D. Feature Fusion

Based on three different branches of features, we concatenate them together as they are providing complementary signals to each other. Rather than using a simple voting classifier, we use several layers of MLPs to fuse the feature, since the weight of each individual learner is actually predefined. Thus, by introducing MLPs, the weight of individual learners can be adaptively defined by gradient descent. In the Eqs. (5)(6), w_i denotes connection weight of a neuron, η denotes learning rate. By epochs of training, the connection weight of neurons will be updated and the most useful information would be selected:

$$w_i \leftarrow w_i + \Delta w_i, \tag{5}$$

$$\Delta w_i = -\eta \frac{\partial Loss}{\partial w_i}. \tag{6}$$

TABLE II
MODELNET

Methods	ModelNet40 Overall Accuracy	ModelNet10 Overall Accuracy
3DmFV-Net [21]	91.6	95.2
SpecGCN [22]	91.5	-
PointNet [5]	89.2	-
PointNet++ [6]	90.7	-
SO-Net [23]	90.9	94.1
SpiderCNN [24]	90.5	-
PointCNN [15]	92.2	-
DGCNN [25]	92.2	-
KPConv [8]	91.2	-
KDNet [26]	91.8	94.0
MBCNN(ours)	91.9	94.1

IV. EXPERIMENT

A. Parameter Selection

- Global-feature branch: We set the initial bandwidth of grids on the sphere to $bw_0 = 128$, which means there are $65,536$ grids on the sphere. This point-based spherical CNN includes an spherical convolution followed by two rotation group convolution. The dimension of the output is set to $output_channel = 128$.
- Local-feature branch: We set the the size of initial subsampling grid to $dl_0 = 0.02$, and the size of the convolution is set to $r_{conv} = 2.5$. 5 convolutional layers are utilized in this branch, which outputs a 1024-dimension feature.
- Point-feature branch: MLP applied on points comprises 5 hidden layers with neuron sizes of $64, 64, 64, 128, 1024$, which is shared by all points. After max-pooling, linear transformation is applied to the points, outputting a 128-dimension channel feature.
- Feature fusion: The features of three branches are normalized to $[-1, 1]$ respectively and then concatenated as a fused feature with 1280 dimensions.

B. Datasets

1) Synthetic Datasets: We select ModelNet40 as our shape classification dataset [27], which is a collection of 12,311 meshed CAD models from 40 various categories. Additionally, we conduct experiments on ModelNet10 (a subset of ModelNet40) to prove the effectiveness of our model.

2) Real-World Datasets: We select ScanObjectNN [11] and ScanNet [28] as datasets for scene classification, which are both based on scanned indoor scene data. ScanObjectNN is a new benchmark dataset, containing around $15,000$ objects which are classified into 15 labels with $2,902$ unique instances. We conduct experiments on several variants of this dataset, which comprise data perturbed by randomly rotating and scaling. In addition, ScanNet is an RGB-D video dataset, containing more than $1,500$ scans. The raw data can be transformed to point clouds by preprocessing.

TABLE III
HARDEST SUBSET OF SCANOBJECTNN

Methods	OA	AA	bag	bin	box	cabinet	chair	desk	display	door	shelf	table	bed	pillow	sink	sofa	tolilet
3DmFV [21]	63.0	58.1	39.8	62.8	15.0	65.1	84.4	36.0	62.3	85.2	60.6	66.7	51.8	61.9	46.7	72.4	61.2
PointNet [5]	68.2	63.4	36.1	69.8	10.5	62.6	89.0	50.0	73.0	93.8	72.6	67.8	61.8	67.6	64.2	76.7	55.3
SpiderCNN [24]	73.7	69.8	43.4	75.9	12.8	74.2	89.0	65.3	74.5	91.4	78.0	65.9	69.1	80.0	65.8	90.5	70.6
PointNet++ [6]	77.9	75.4	49.4	84.4	31.6	77.4	91.3	74.0	79.4	85.2	72.6	72.6	75.5	**81.0**	**80.8**	90.5	**85.9**
DGCNN [25]	78.1	73.6	49.4	82.4	33.1	83.9	91.8	63.3	77.0	89.0	79.3	**77.4**	64.5	77.1	75.0	91.4	69.4
PointCNN [15]	78.5	75.1	**57.8**	82.9	33.1	83.6	**92.6**	65.3	78.4	84.8	**84.2**	67.4	80.0	80.0	72.5	**91.9**	71.8
BGA-PN++ [11]	80.2	77.5	54.2	**85.9**	39.8	81.7	90.8	76.0	84.3	87.6	78.4	74.4	73.6	80.0	77.5	**91.9**	**85.9**
BGA-DGCNN [11]	79.7	75.7	48.2	81.9	30.1	**84.4**	**92.6**	**77.3**	80.4	**92.4**	80.5	74.1	72.7	78.1	79.2	91.0	72.9
MBCNN(ours)	**81.5**	**77.8**	49.5	68.6	**68.1**	81.8	89.9	66.3	**87.9**	91.7	83.1	74.5	**83.8**	75.2	73.7	89.2	83.5

TABLE IV
ABLATION STUDY

Model	Global	Local	Point	Acc
A	✓			56.1
B		✓		79.8
C			✓	68.2
D	✓	✓		80.3
E		✓	✓	80.5
F	✓		✓	70.2
MBCNN(ours)	✓	✓	✓	**81.5**

TABLE V
THE EASIEST SUBSET OF SCANOBJECTNN AND SCANNET

Methods	ScanObjectNN Overall Accuracy	ScanNetV1 Overall Accuracy
3DmFV [21]	73.8	-
PointNet [5]	79.2	70.5
SpiderCNN [24]	79.5	-
KPConv [8]	-	73.7
PointNet++ [6]	84.3	-
DGCNN [25]	86.2	-
PointCNN [15]	85.5	-
MBCNN(ours)	**86.6**	**75.7**

C. Voxel-Based Indexing

For many point-based convolution models, the procedure of searching neighbors is arranged to the data loading phase to avoid extra training time, which is not illustrating the real-world conditions. Thus, we do not conduct our experiments in the training phase. Instead, we directly compare the neighbor searching time of different methods, and the total running time per batch is estimated.

D. Discussion

1) Synthetic Datasets: MBCNN has an excellent performance in ModelNet40/ModelNet10, achieving the accuracy of 91.9% and 94.1% respectively, which has almost the same performance as the state-of-the-art (including 3DmFV [21], DGCNN [25] and PointCNN [15]), without extra feature engineering and task-tunning.

2) Real-world Datasets: As shown in ScanObjectNN and ScanNet (Tables III, V), our MBCNN outperforms all former models, achieving the accuracy of 81.5% in the hardest subset of ScanObjectNN while achieving the accuracy of 86.6% in the easiest subset. From the accuracy of various classes in Table III, we notice that our fusion strategy shows higher effectiveness in more indecipherable classes.

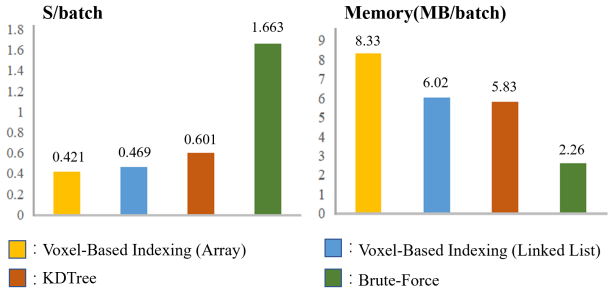


Fig. 6. **Performance of different methods in finding neighbors.**

Moreover, in Table IV, we analyze the performance of a combination of various branches, which proves the efficiency of our fusion strategy.

3) Voxel-based Indexing: As shown in Fig. 6, the procedure of finding neighbors is accelerated by utilizing voxel-based indexing, which is about 30% faster than the KDTree-based searching, and 70% faster than the Brute-Force searching, while there is no significant increasing memory footprint.

From the extensive experiments above, We notice that Models including 3DmFV [21], DGCNN [25] and PointCNN [15] perform evenly or outperform MBCNN in ModelNet40/ModelNet10. Nonetheless, these models are influenced heavily in the complex scene, while our MBCNN is more robust in processing real-world data. Besides, MBCNN is swifter than previous point-based convolution models due to voxel-based indexing.

V. CONCLUSION

In this paper, we proposed MBCNN (Multi-Branch Convolutional Neural Networks) for fast and efficient 3D Classification. A novel fusion strategy was proposed to employ the local-feature branch and global-feature branch to extract the spatial patterns and utilize the robustness of pointwise MLP so that the models could achieve a great performance in the real-world datasets. Moreover, a voxel-based indexing method was proposed to accelerate the procedure of neighbor finding in the local-feature branch. According to the experiments, it was proved that MBCNN has a better performance than the state-of-the-art in real-world classification tasks. Additionally, the ablation study demonstrated the effectiveness of our fusion strategy.

REFERENCES

[1] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D shapenets: A deep representation for volumetric shapes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[2] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: learning dense volumetric segmentation from sparse annotation," in *International Conference on Medical Image Computing and Computer-assisted Intervention (MICCAI)*. Springer, 2016, pp. 424–432.

[3] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel CNN for efficient 3D deep learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2019, pp. 965–975.

[4] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 652–660.

[6] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5099–5108.

[7] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8895–8904.

[8] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and deformable convolution for point clouds," in *IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 6411–6420.

[9] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, "Searching efficient 3D architectures with sparse point-voxel convolution," *arXiv preprint arXiv:2007.16100*, 2020.

[10] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical CNNs," *arXiv preprint arXiv:1801.10130*, 2018.

[11] M. A. Uy, Q.-H. Pham, B.-S. Hua, T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 1588–1597.

[12] R. Klokov and V. Lempitsky, "Escape from cells: Deep Kd-Networks for the recognition of 3D point cloud models," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 863–872.

[13] 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 Transactions On Graphics (TOG)*, vol. 38, no. 5, pp. 1–12, 2019.

[14] S. Lan, R. Yu, G. Yu, and L. S. Davis, "Modeling local geometric structure of 3D point clouds using Geo-CNN," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 998–1008.

[15] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on x-transformed points," in *Advances in Neural Information Processing Systems (NIPS)*, 2018, pp. 820–830.

[16] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *the European Conference on Computer Vision (ECCV)*, 2018, pp. 87–102.

[17] A. D. Pon, J. Ku, C. Li, and S. L. Waslander, "Object-centric stereo matching for 3D object detection," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8383–8389.

[18] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *International Conference on Machine Learning (ICML)*, 2016, pp. 2990–2999.

[19] T. S. Cohen and M. Welling, "Steerable CNNs," *arXiv preprint arXiv:1612.08498*, 2016.

[20] R. Polikar, "Ensemble learning. scholarpedia, 4 (1): 2776, 2009."

[21] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3DmFV: Three-dimensional point cloud classification in real-time using convolutional neural networks," *IEEE Robotics and Automation Letters (RAL)*, vol. 3, no. 4, pp. 3145–3152, 2018.

[22] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *the European conference on computer vision (ECCV)*, 2018, pp. 52–66.

[23] J. Li, B. M. Chen, and G. Hee Lee, "So-Net: Self-organizing network for point cloud analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 9397–9406.

[24] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "SpiderCNN: Deep learning on point sets with parameterized convolutional filters," in *the European Conference on Computer Vision (ECCV)*, 2018, pp. 87–102.

[25] B. Wu, Y. Liu, B. Lang, and L. Huang, "DgCNN: Disordered graph convolutional neural network based on the gaussian mixture model," *Neurocomputing*, vol. 321, pp. 346–356, 2018.

[26] R. Klokov and V. Lempitsky, "Escape from cells: Deep Kd-Networks for the recognition of 3d point cloud models," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 863–872.

[27] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D shapeNets: A deep representation for volumetric shapes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1912–1920.

[28] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5828–5839.