2024 IEEE International Conference on Multimedia and Expo (ICME) | 979-8-3503-9015-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICME57554.2024.10687520

MPVNN: Multi-resolution Point-Voxel Non-parametric Network for 3D Point Cloud Processing

1st Keli Wen
School of Electronic and Computer Engineering, Peking University
Shenzhen, China
keliwen@stu.pku.edu.cn

2nd Nan Zhang
School of Electronic and Computer Engineering, Peking University
Shenzhen, China
zhangnan@stu.pku.edu.cn

3rd Ge Li
School of Electronic and Computer Engineering, Peking University
Shenzhen, China
geli@ece.pku.edu.cn

4rd Wei Gao*
School of Electronic and Computer Engineering, Peking University
Shenzhen, China
gaowei262@pku.edu.cn

***Abstract*—Recent non-parametric networks have demonstrated the feasibility of point cloud tasks through skillfully combining Farthest point sampling, k-nearest neighbor, and pooling algorithms. However, these point-based algorithms struggle to capture features containing fine-grained spatial structures within a point cloud, thereby limiting the performance of non-parametric networks. To address it, we propose the Multi-resolution Point-Voxel Non-parametric Network (MPVNN) to enhance the structural information of voxelized point clouds at varying resolutions. Specifically, we design a Multi-Resolution Voxel Encoder (MRVEnc) that processes point clouds with multiple resolutions and adopts trilinear interpolation to synthesize features with respect to 8 voxel grid vertices. Then, we amalgamate features derived from the point-based algorithms with those from MRVEnc to establish a point memory bank for subsequent tasks. Extensive experiments on ModelNet40, 3D-FUTURE, and ShapeNetPart demonstrate the superior performance of MPVNN in both classification and segmentation tasks.**

***Index Terms*—Point Cloud Representation, Non-parametric Network, Multi Resolution Feature Learning**

I. Introduction

Point cloud analysis is a cornerstone of various downstream tasks such as robotics, autonomous driving, and AR/VR [1]–[3]. The point cloud is a collection of data points in 3D space, each representing a part of an object's surface, together forming a representation of the object's shape. Unlike 2D images represented by regular dense grids, they exhibit randomness in distribution and are permutation-invariant, making it infeasible to design the model to process point clouds directly.

Wei Gao is the corresponding author (gaowei262@pku.edu.cn). This work was supported in part by Natural Science Foundation of China (No. 62172021), in part by Shenzhen Science and Technology Program (KQTD20180411143338837).

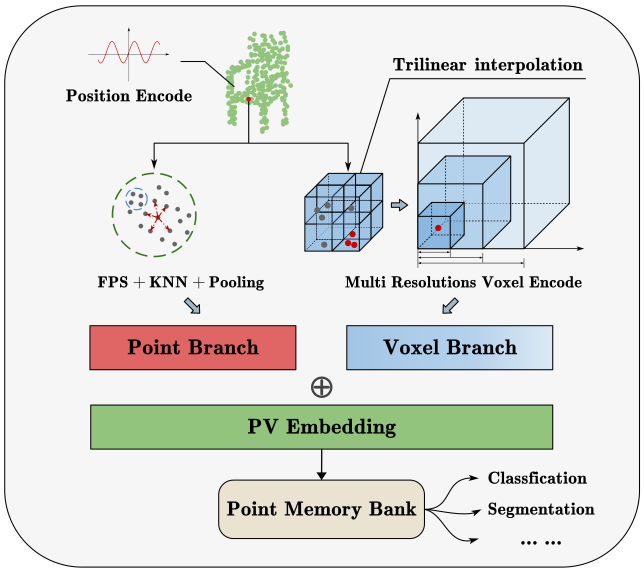

Fig. 1: The illustration of our non-parametric point cloud processing framework MPVNN. It utilizes trigonometric functions for original position encoding of the point clouds, employs non-parameter operations (FPS, KNN, and Pooling) to aggregate features in the Point branch, and applies trilinear interpolation to update Voxel branch features at different resolutions. The final PV Embedding, concatenated from these features, can be utilized for various 3D tasks.

Before deep learning methods became prevalent, traditional methods used feature descriptors like PFH [4] and VFH [5] to efficiently compute point cloud features from information about position and curvature, but these methods perform poorly

in complex and large-scale scenarios. PointNet is the first deep-learning-based method to directly process raw point cloud data. It utilizes learnable operators like Multilayer Perceptrons (MLPs) to extract point features. Subsequent works [6]–[11] have refined operators by introducing hierarchical structures, graph network, dynamic convolution kernels, curve grouping operations and self-attention mechanism to extract local structures. Additionally, PointMLP [12] introduces a geometric affine module for feature normalization and demonstrates more complex models and a larger number of learnable parameters could improve modeling capabilities. The heavy computational requirements have motivated research into non-parametric approaches for point cloud analysis.

Recently, Point-NN is introduced as the first non-parametric network, employing non-parametric encoders such as FPS, k-NN, and trigonometric functions for 3D feature extraction, along with a point-memory bank for task-specific recognition. The favorable results achieved by Point-NN across multiple datasets showcase the potential of non-parametric networks. However, Point-NN struggles to capture features containing fine-grained spatial structures within a point cloud, potentially limiting performance in certain scenarios. In this paper, we propose the Multi-resolution Point-Voxel Non-parametric Network (**MPVNN**), which incorporates a Multi-Resolution Voxel Encoder (**MRVEnc**) to enable voxel-based feature extraction at various resolutions. Our contributions are summarized below:

- We propose a novel non-parametric network MPVNN to better capture fine-grained spatial structures information.
- We design MRVEnc, which handles point clouds with multiple resolutions and adopts trilinear interpolation to calculate features, to enable MPVNN to obtain voxel features of point clouds at varying resolutions.
- We boost overall efficiency by implementing point memory bank sparsification, integrated with the Distributed Data Parallelism framework.
- We conduct extensive experiments on the ModelNet40, 3D-FUTURE, and ShapeNetPart datasets, achieving considerable improvements in performance.

II. Related Work

Point cloud analysis has initially been divided into two distinct approaches. Due to the inherent irregularity and unordered nature of point clouds, some methods have focused on transforming raw point clouds into voxels or images, thereby translating 3D tasks into the more explored 2D domain. However, these transformations can result in the loss of important details, and the methods that involve converting to voxels or images are not designed to work directly with raw point cloud data. Pioneering the direct use of raw point cloud data, PointNet [13] was the first to employ Multilayer Perceptrons (MLPs) for this purpose, and its successor, PointNet++ [6], enhanced feature extraction through a hierarchical structure. Subsequent research has concentrated on designing methods to extract fine-grained local features. Graph-based methods, such as DGCNN [7] and GACNet [14], utilize graph neural networks, incorporating point and edge features to model local relationships. Convolution-based methods like Kpconv [8] employ dynamic convolutional kernels for adaptively aggregating neighborhood features. Moreover, transformer-based networks [15], [16] have emerged, utilizing self-attention mechanisms for feature extraction. In this progression, PointMLP [12] introduced a geometric affine module to normalize features, and Curvenet [11] offered an innovative approach to feature aggregation, including a novel curve grouping and aggregation operator.

Before the advent of deep learning in the field, traditional descriptors such as PFH [4], FPHF [17], and VFH [5] leveraged basic features like points, normals, and curvatures. Although computationally efficient, these methods were less effective in large-scale scenes due to the prevalence of similar normals or curvatures. In most point cloud processing scenarios, non-parametric methods like farthest point sampling (FPS), k-Nearest Neighbors (KNN), and pooling operations are prevalent. Based on this, Point-NN [18] introduced a fully non-learnable, non-parametric network. This training-free network demonstrated commendable performance across various tasks. However, Point-NN was limited in capturing point cloud structures at different scales. To address it, we propose the MPVNN, incorporating MRVEnc for voxel-based feature extraction at varying resolutions, aiming to capture the structural information of point clouds more effectively.

III. Method

A. Preliminary

Point-NN introduces an approach in 3D point cloud processing by leveraging non-learnable components to construct a Non-Parametric Encoder. The Non-Parametric Encoder is composed of two main components: Raw-point Embedding and Local Geometry Aggregation.

Raw-point Embedding The initial phase of the Non-Parametric Encoder involves the embedding of raw point data. For each point $p_i = (x_i, y_i, z_i) \in \mathbb{R}^{1\times 3}$, the embedding process is formulated using trigonometric functions, as follows:

$$\begin{aligned} \mathrm{PosE}(p_i) &= \mathrm{Concat}(fp_i^x,\ fp_i^y,\ fp_i^z), \\ fp_i^x[2m] &= \sin\left(\alpha x_i/\beta^{\frac{6m}{C_I}}\right), \\ fp_i^x[2m+1] &= \cos\left(\alpha x_i/\beta^{\frac{6m}{C_I}}\right), \end{aligned} \tag{1}$$

where $fp_i^x,\ fp_i^y,\ fp_i^z \in \mathbb{R}^{1\times\frac{C_I}{3}}$ denote the embeddings of three axes, C_I denotes the initial feature dimension and α, β control the magnitude and wavelengths, respectively.

Local Geometry Aggregation (LGA) starts with Farthest Point Sampling (FPS) and k-Nearest Neighbors (k-NN) to select and group local regions and then uses Pooling (Max and Average Pooling) for feature integration.

For more detailed, the first step in the local geometry aggregation process involves expanding the feature dimensions

to encode 3D semantics with increasing depth. This is achieved through concatenation:

$$fp_{cj} = \text{Concat}(fp_c,\ fp_j), \quad \text{for } j \in \mathcal{N}_c, \tag{2}$$

where fp_{cj} represents the expanded feature, blending the neighbor feature fp_j with the center feature fp_c.

The next phase focuses on encoding the spatial distribution of neighbors in the local region. This is accomplished by weighting each expanded feature fp_{cj} with its relative positional encoding:

$$fp_{cj}^w = \left(fp_{cj} + \text{PosE}(\Delta p_j)\right) \odot \text{PosE}(\Delta p_j). \tag{3}$$

Here, Δp_j represents the normalized coordinates of each neighbor. The element-wise multiplication $\odot$ integrates these positional encodings.

Finally, the feature will aggregated by using both max pooling and average pooling:

$$fp_c^a = \text{MaxP}\left(\{fp_{cj}^w\}_{j\in\mathcal{N}_c}\right) + \text{AveP}\left(\{fp_{cj}^w\}_{j\in\mathcal{N}_c}\right). \tag{4}$$

This step summarizes neighboring features resulting in local-aggregated centers $\{fp_c^a, p_c\}_{c=1}^{\frac{M}{2}}$ for the next stage, where the M denotes the number of points before downsampling. Ultimately, Point-NN yields a global feature representation fp_G with C_G^p feature dimension of the input point cloud.

The Point-NN architecture can effectively aggregate neighboring point-based features through k-NN within its Local Geometry Aggregation stage. However, it lacks the capability to capture varying resolutions of the point cloud. To overcome this limitation, we introduce a multi-resolution voxel-based encoder, facilitating the perception of features across different resolutions.

B. Multi Resolution Voxel-base Encoder

Our proposed **MRVEnc** (Multi-Resolution Voxel-based Encoder) introduces a voxel-based feature processing branch. Our encoder voxelizes the input point clouds at various resolutions to generate a set of voxel-based features.

We integrate our MRVEnc into the Point-NN framework. For each input point $p_i = (x_i, y_i, z_i) \in \mathbb{R}^{1\times 3}$, we first apply min-max normalization and z-score normalization to the point clouds, resulting in:

$$p_i' = \frac{\tilde{p}_i - \min(\{\tilde{p}_i\})}{\max(\{\tilde{p}_i\}) - \min(\{\tilde{p}_i\})}, \quad \tilde{p}_i = \frac{p_i - \mu(\{p_i\})}{\sigma(\{p_i\})}. \tag{5}$$

This gives new coordinates $p_i' = (x_i', y_i', z_i') \in \mathbb{R}^{1\times 3}$, all within the range $[0, 1]$. For a set of resolutions $\mathcal{R} = \{r_1, r_2, \cdots, r_n\}$, MRVEnc voxelizes point clouds into voxels with grid size r_i.

As illustrated in Figure 2, for any resolution r, we can obtain a set of voxel grid vertices $\{V_{ijk}\}$. We apply the Eq. (1) to perform positional encoding on the voxel grid vertices to obtain

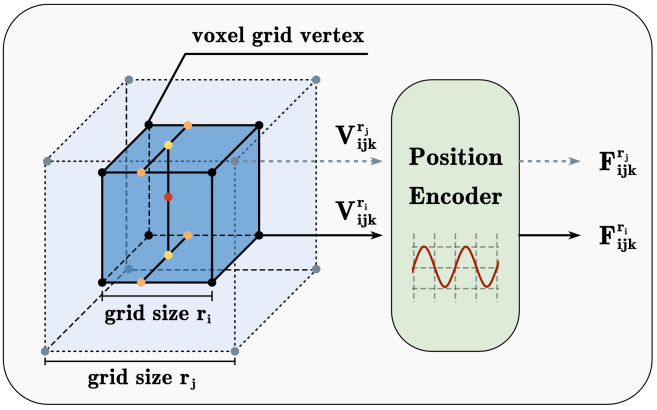

Fig. 2: The illustration of voxel feature updating in MRVEnc. The resolutions r_i, r_j are the grid size of the corresponding voxel. For each voxel grid vertex like $V_{ijk}^{r_i}$, we use the trigonometric function to generate the initial feature $F_{ijk}^{r_i}$. We calculate the voxel features by trilinear interpolation from feature and position of 8 corresponding voxel grid vertices.

the original features F_{ijk}, as follows:

$$\begin{aligned} V_{ijk}^r &= \text{Voxelize}(p_i', r), \\ F_{ijk}^r &= \text{PosE}(V_{ijk}^r) \in \mathbb{R}^{1\times C_I}. \end{aligned} \tag{6}$$

In each resolution $r \in \mathcal{R}$, we assign the point p_i' to a specific voxel. This process involves determining the voxel that contains p_i'. MRVEnc then uses the features $\{F_{ijk}^r\}$ of the 8 voxel grid vertices of the assigned voxel and their relative position information to calculate the Voxel-based feature fv_i^r by applying trilinear interpolation. We can finally obtain the feature f_G by concatenating these Voxel-based features fv_i^r with point-based features fp_i.

$$\begin{aligned} fv_i^r &= \text{Trilinear}(F_{ijk}^r, V_{ijk}^r, p_i') \in \mathbb{R}^{1\times C_I}, \\ f_{G_i} &= \text{Concat}\left(fp_i, \{fv_i^r\}\right). \end{aligned} \tag{7}$$

MRVEnc is integrated into each PV Block to effectively capture the changes in point cloud structure caused by FPS downsampling, which enhances the encoder's feature representation capabilities. Finally, we can obtain the global feature f_G through the 4-stage aggregation of the PV Blocks.

C. Point Memory Bank and Retrieve Optimization

Point Memory Bank We utilize the point-memory bank similar to Point-NN for classification and segmentation tasks. This involves creating a point-memory bank for the training set through the model, followed by retrieval based on similarity matching for prediction.

For the Shape Classification task, let's consider the training set containing N real point clouds $\{P_n\}_{n=1}^N$ of K categories. We encode them using the aforementioned model and convert their corresponding ground-truth labels $\{t_n\}_{n=1}^N$ into one-hot

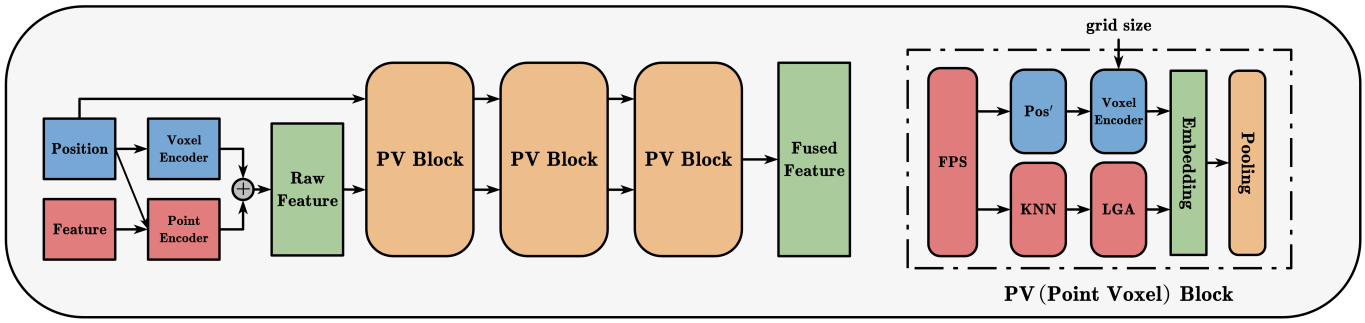

Fig. 3: The framework of the MPVNN. The red blocks are based on the Point-NN, focusing on feature aggregation for the Point branch. The blue blocks highlight our innovative MRVEnc. As detailed on the right, by feeding in varying grid sizes, MRVEnc encodes features of the Voxel branch at multiple resolutions within each PV Block. MPVNN finally integrates the features from both the Point and Voxel branches to obtain the fused features to improve the perception ability.

encoding. These are ultimately stored in two matrices, F_{mem} and T_{mem}, as follows:

$$\begin{aligned} F_{mem} &= \text{Concat}\left(\left\{\,\text{MPVNN}(P_n)\right\}_{n=1}^{N}\right), \\ T_{mem} &= \text{Concat}\left(\left\{\,\text{OneHot}(t_n)\right\}_{n=1}^{N}\right), \end{aligned} \tag{8}$$

where $F_{mem} \in \mathbb{R}^{N\times C_G}$ and $T_{mem} \in \mathbb{R}^{N\times K}$. At this point, we have obtained the encoded categorical knowledge of the 3D training set.

For a test point cloud, we extract its global features as $f_G^t \in \mathbb{R}^{1\times C_G}$ using the same method. First, we calculate the cosine similarity $S_{\cos}$ between the test feature and F_{mem}. Subsequently, using $S_{\cos}$ as weights, we extract the most probable label from T_{mem}:

$$\begin{aligned} S_{\cos} &= \frac{f_G^t F_{mem}^T}{\|f_G^t\| \cdot \|F_{mem}\|} \in \mathbb{R}^{1\times N}, \\ \text{logits} &= \varphi(S_{\cos} T_{mem}) \in \mathbb{R}^{1\times K}, \end{aligned} \tag{9}$$

where $\varphi(x) = \exp(-\gamma(1-x))$ is used as the activation function. In $S_{\cos}$, more similar feature memories receive higher scores and contribute more to the final classification logits. Our point-memory bank can adaptively distinguish different point cloud instances without any training.

Sparsification Due to the potential redundancy in the point-memory bank, we sparsify the bank using cosine similarity before retrieval to reduce memory usage and improve inference efficiency by:

$$\begin{aligned} F_m',\ T_m' &= \text{Filter}(F_{mem}, T_{mem}, f_{\cos}, \theta), \\ f_{\cos} &= \left(F_{mem} F_{mem}^T\right) / \left(\|F_{mem}\|^2\right), \end{aligned} \tag{10}$$

where F_m', T_m' denote the sparsified point-memory bank. The key to this sparsification lies in the choice of the threshold θ, points with a cosine similarity higher than the threshold θ are removed from the point-memory bank. We achieve this by sorting the element-wise cosine similarities within the bank and adaptively adjusting the threshold θ based on percentiles. After sparsifying the bank, experiments on different datasets in shape classification tasks show that we can maintain (or slightly compromise) the accuracy while reducing memory usage and improving the retrieval efficiency. For ModelNet40, we can save 50% of the memory usage and increase retrieval efficiency by 30% with only a 1% loss in accuracy.

TABLE I: **Shape Classification on Synthetic ModelNet40 [19]**. All compared methods take 1,024 points as input. Train Time and Test Speed (samples/second) are tested on one RTX 3090 GPU. We report the accuracy **without the voting strategy**. The comparison results are obtained from the original papers. † denotes the test speed in the DDP mode.

Method	Acc. (%)	Param.	Train Time	Test Speed
PointNet [13]	89.2	3.5 M	-	-
PointNet++ [6]	90.7	1.7 M	3.4 h	521
DGCNN [7]	92.9	1.8 M	**2.4 h**	617
RS-CNN [20]	92.9	1.3 M	-	-
DensePoint [21]	93.2	-	-	-
PCT [22]	93.2	-	-	-
GBNet [23]	93.8	8.4 M	-	189
CurveNet [11]	93.8	2.0 M	6.7 h	25
PointMLP [12]	**94.1**	12.6 M	14.4 h	189
Point-NN [18]	81.8	0.0 M	0	275
MPVNN	85.5	0.0 M	0	59 / 469†

IV. Experiments

A. Shape Classification

Dataset & Implementation The ModelNet 40 dataset [19] is widely recognized as the most prevalent benchmark for object shape classification, featuring a collection of meshed CAD models spanning various objects. It comprises 12311 models categorized into 40 classes. The 3D-FUTURE dataset [24]

TABLE II: **Shape Classification on 3D Furniture Shape [24]**. All compared methods take 1,024 points as input. The table presents the accuracy percentages for two levels of category granularity. **Acc. (%)** represents the accuracy for the broader super-category classification comprising 8 categories, while **Detailed-Acc. (%)** denotes the accuracy of the detailed classification into 50 categories. We report the accuracy **without the voting strategy**.

Method	Acc. (%)	Detailed-Acc. (%)	Param.
PointNet [13]	87.0	64.4	3.5 M
PointNet++ [6]	88.8	66.7	1.7 M
DGCNN [7]	88.2	62.2	1.8 M
CurveNet [11]	89.6	66.6	2.0 M
PointMLP [12]	89.8	67.6	12.6 M
Point-NN [18]	82.5	59.3	0.0 M
MPVNN	85.5	63.2	0.0 M

TABLE III: **Part Segmentation on ShapeNetPart [25]**. All compared methods take 2,048 points as input and are evaluated by mean IoU scores (%) across **classes** and **instances**. The comparison results are obtained from the original papers. † denotes the test speed in the DDP mode.

Method	Cls. mIoU	Inst. mIoU	Param.	Train Time	Test Speed
PointNet [13]	80.4	83.7	8.3 M	-	-
PointNet++ [6]	81.9	85.1	**1.8 M**	**26.5 h**	45
PAConv [22]	-	86.0	-	-	-
PointMLP [12]	84.6	86.1	16.8 M	47.1 h	119
CurveNet [11]	-	**86.6**	5.5 M	56.9 h	22
Point-NN [18]	69.2	70.4	0.0 M	0	51
MPVNN	72.7	74.0	0.0 M	0	12 / 91†

includes 16,563 unique and detailed 3D furniture instances, complete with high-resolution textures, across 5,000 different room settings. In our experiments, we converted these mesh models into point cloud datasets through Poisson disk sampling.

This dataset offers two types of categorizations for each object: (Super) Category, which encompasses 8 broad classes, and Detailed-Category, providing a more granular subdivision within each Category. For example, under the Category **Bed**, we find further classifications like **Kids Bed**, **Single Bed**, **Bunk Bed** and, so on. The Detailed-Category divides the dataset into 50 classes, presenting a significantly challenging task. In both datasets, we utilized only the coordinates of 1,024 uniformly sampled points as inputs for the network. We'll normalize these points into unit spheres before encoding them. In the shape classification task, we use 3 different grid sizes in the MRVEnc.

Experimental Result In Table I and II, the green rows detail the results of MPVNN for the ModelNet 40 and 3D-FUTURE datasets, respectively. MPVNN, adopting a parameter-free framework, demonstrates substantial improvements over Point-NN in both datasets. On ModelNet 40, our results show an increase in performance by over 3%, and for the 3D-FUTURE dataset, we maintained a performance increase of more than 3% across various categorization levels. These results solidify our model's capability to extract more nuanced global features from different point cloud structures.

Efficiency-wise, integrating the Multi-Resolution Voxel-based Encoder inevitably increased the computational load due to its more detailed structural analysis. However, our model's suitability for distributed computing led us to incorporate a distributed data parallel, enhancing inference speed by approximately $7.91\times$ on an 8 GPUs setup.

TABLE IV: **Ablation Study of MPVNN on ModelNet40.**

Interpolation Method	Acc (%)	NUM. Resolution	Acc (%)
Baseline	81.8	1	84.4
Nearest	84.7	2	84.8
Gaussian	85.0	**3**	**85.5**
Trilinear (Ours)	**85.5**	4	84.9

B. Part Segmentation

Dataset & Implementation Our method was validated on the ShapeNetPart dataset [25], encompassing 16881 shape models across 16 categories. Each model typically labeled fewer than 6 parts, cumulatively accounting for 50 distinct parts. Our data preparation followed the split scheme from [6], [13], using 12137 models, with the remaining models allocated for validation. From each model, 2048 points were uniformly sampled to serve as inputs. In the par segmentation task, we also use 3 different grid sizes in the MRVEnc. Other than extracting global features, part segmentation requires to classify each input point. Therefore, we append a symmetric decoder after the encoder, which is the same as [18].

Experimental Result The introduction of a Multi-Resolution Voxel-based Encoder in MPVNN is anticipated to be particularly beneficial for tasks like Part Segmentation, as it is expected to significantly aid in capturing spatial information at various scales. To demonstrate our model's performance on the ShapeNetPart dataset for Part Segmentation tasks, we employ both Class-level and Instance-level mean Intersection over Union (Cls. mIOU and Inst. mIOU, respectively) as metrics [12]. We observed that MPVNN achieves an approximate 3% improvement in both Cls. mIOU and Inst. mIOU, as detailed in the green row of Table III. This result indicates that our design effectively captures more representative and finer-grained features.

C. Ablation Study

We conducted ablation studies on ModelNet40 on various interpolation methods and the number of resolutions, with the

detailed results presented in Table IV. Our results demonstrate that trilinear interpolation is the most effective method for MPVNN. We also analyzed how varying the number of resolutions (NUM. Resolution = $|\mathcal{R}|$) impacts performance. As the number of resolutions increases, computational costs rise substantially. Thus, using **three** distinct resolutions strikes an optimal balance between MPVNN accuracy and efficiency.

V. Conclusions

We conduct an analysis of Point-NN, a non-parametric network for 3D point cloud analysis. Our findings indicate that the existing network architecture struggles to capture features of point clouds across multiple scales and resolutions. By integrating a Multi-Resolution Voxel-based Encoder into the foundation of Point-NN, our **MPVNN** is now capable of encoding the structural information of point clouds at varying scales. Furthermore, we optimize the point-memory bank through sparsification, enabling our model to maintain (or slightly compromise) accuracy while optimizing memory footprint and inference efficiency. Our proposed **MPVNN** has markedly improved the performance in point cloud classification and segmentation over Point-NN.

References

[1] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE TPAMI*, 2020.

[2] S. Zhang, H. Cao, Y. Liu, S. Cai, Y. Zhang, Y. Li, and X. Chi, "Sn-graph: A minimalist 3d object representation for classification," in *ICME*, 2021.

[3] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, and J. Li, "Deep learning for lidar point clouds in autonomous driving: A review," *IEEE TNNLS*, 2020.

[4] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Persistent point feature histograms for 3d point clouds," in *Proc 10th Int Conf Intel Autonomous Syst (IAS-10)*, 2008.

[5] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3d recognition and pose using the viewpoint feature histogram," in *IEEE/RSJ international conference on intelligent robots and systems*, 2010.

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

[7] 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*, 2019.

[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 *ICCV*, 2019.

[9] L. Xun, X. Feng, C. Chen, X. Yuan, and Q. Lu, "Graph attention-based deep neural network for 3d point cloud processing," in *ICME*, 2021.

[10] N. Zhang, Z. Pan, T. H. Li, W. Gao, and G. Li, "Improving graph representation for point cloud segmentation via attentive filtering," in *CVPR*, 2023.

[11] T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai, "Walk in the cloud: Learning curves for point clouds shape analysis," *ICCV*, 2021.

[12] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu, "Rethinking network design and local geometry in point cloud: A simple residual mlp framework," *ICLR*, 2022.

[13] 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.

[14] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, "Graph attention convolution for point cloud semantic segmentation," in *CVPR*, 2019.

[15] X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, "Point-bert: Pre-training 3d point cloud transformers with masked point modeling," in *CVPR*, 2022.

[16] X. Lai, J. Liu, L. Jiang, L. Wang, H. Zhao, S. Liu, X. Qi, and J. Jia, "Stratified transformer for 3d point cloud segmentation," in *CVPR*, 2022.

[17] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *ICRA*, 2009.

[18] R. Zhang, L. Wang, Y. Wang, P. Gao, H. Li, and J. Shi, "Parameter is not all you need: Starting from non-parametric networks for 3d point cloud analysis," *CVPR*, 2023.

[19] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *CVPR*, 2015.

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

[21] 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.

[22] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *Computational Visual Media*, 2021.

[23] S. Qiu, S. Anwar, and N. Barnes, "Geometric back-projection network for point cloud classification," *IEEE TMM*, 2021.

[24] H. Fu, R. Jia, L. Gao, M. Gong, B. Zhao, S. Maybank, and D. Tao, "3d-future: 3d furniture shape with texture," *IJCV*, 2021.

[25] 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*, 2016.