3DCTN: 3D Convolution-Transformer Network for Point Cloud Classification

Dening Lu, Qian Xie, Kyle Gao, *Graduate Student Member, IEEE*, Linlin Xu, *Member, IEEE*, and Jonathan Li, *Senior Member, IEEE*

***Abstract*— Point cloud classification is a fundamental task in 3D applications. However, it is challenging to achieve effective feature learning due to the irregularity and unordered nature of point clouds. Lately, 3D Transformers have been adopted to improve point cloud processing. Nevertheless, massive Transformer layers tend to incur huge computational and memory costs. This paper presented a novel hierarchical framework that incorporated convolutions with Transformers for point cloud classification, named 3D Convolution-Transformer Network (3DCTN). It combined the strong local feature learning ability of convolutions with the remarkable global context modeling capability of Transformers. Our method had two main modules operating on the downsampling point sets. Each module consisted of a multi-scale local feature aggregating (LFA) block and a global feature learning (GFL) block, which were implemented by using the Graph Convolution and Transformer respectively. We also conducted a detailed investigation on a series of self-attention variants to explore better performance for our network. Various experiments on ModelNet40 and ScanObjectNN datasets demonstrated that our method achieves state-of-the-art classification performance with a lightweight design. The code is publicly available at https://github.com/d62lu/3DCTN.**

***Index Terms*— Transformer, convolution-transformer, hierarchical transformer, point cloud classification, deep learning, self-attention mechanism, graph convolution.**

I. Introduction

With the popularization of sensors that are able to obtain geometric information of 3D scenes, such as 3D laser scanners and RGB-D cameras, 3D point cloud classification, as a fundamental 3D computer vision task, has become more and more important for many computer graphics and vision applications. 3D point cloud data can clearly express 3D geometry thanks to its simple yet flexible data structure. In recent years, 3D point cloud classification has been applied widely in many important fields, such as urban construction, autonomous driving, robotics, engineering survey and mapping.

Point cloud classification is highly dependent on global features. Compared with 2D images, point clouds have more complicated structures distributed in 3D space, with points arranged in an irregular and unordered manner. Therefore, it remains a challenging research topic to design deep learning networks to achieve effective global feature extraction.

To tackle the aforementioned challenges, a number of deep learning-based approaches on 3D point clouds have been proposed. Many existing works [1], [2], [3], [4], [5] focused on projecting the 3D point cloud to 2D parameter planes by using multi-view projections, or by designing discrete spatial convolutions with 3D space voxelization. Despite achieving great success in point cloud processing, such methods fail to leverage the sparsity of spatial point clouds, and massive projection operations tend to incur high computation cost and memory consumption. Using a new approach, Charles *et al.* [6] proposed PointNet to achieve the point cloud feature learning in a point-wise manner. PointNet consisted of several core modules: rigid transformations (T-Net), shared Multi-Layer Perceptrons (MLPs) and maxpooling, which ensured the network invariant to point permutation and shape rotation. After that, several variants [7], [8], [9] have been proposed to improve the performance of PointNet by introducing local feature extraction. To utilize the strong local feature extraction capability of convolutional neural networks (CNNs), many meaningful works, such as PointCNN [10], PointConv [11], and DGCNN [12], were proposed to define the 3D convolutional kernels or Graph Convolution to improve point cloud processing and analysis.

Transformers have recently contributed to impressive progress in Natural Language Processing (NLP) and computer vision. Transformers have proven to have a remarkable ability for global feature learning, and thus has been applied to various point cloud processing tasks, such as object classification, semantic scene segmentation, and object part segmentation [13], [14], [15], [16]. The core component of the Transformer is the self-attention mechanism, which first computes the similarities between any two embedded words, and then utilizes the corresponding similarities to compute the weighted sum of all words, as the new output. With this, each output word is able to establish connections with all input words, which is the reason why the Transformer

Manuscript received 1 March 2022; revised 27 June 2022; accepted 27 July 2022. This work was supported in part by the National Natural Science and Engineering Research Council of Canada under Grant RGPIN-2019-06744, Grant RGPIN-04726-2016, and Grant RGPIN-2022-03741; in part by the National Natural Science of China under Grant 41871380; and in part by the Chinese Scholarship Council under Grant 202106830030. The Associate Editor for this article was C. Wen. *(Corresponding authors: Linlin Xu; Jonathan Li.)*

Dening Lu, Kyle Gao, and Linlin Xu are with the Department of Systems Design Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: d62lu@uwaterloo.ca; y56gao@uwaterloo.ca; l44xu@uwaterloo.ca).

Qian Xie is with the Department of Computer Science, University of Oxford, Oxford OX1 3QD, U.K. (e-mail: qian.xie@cs.ox.ac.uk).

Jonathan Li is with the Department of Systems Design Engineering and the Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: junli@uwaterloo.ca).

Digital Object Identifier 10.1109/TITS.2022.3198836

is good at learning global feature. Therefore, current 3D point cloud Transformers tend to employ Transformer layers to replace all convolution operations in networks for better feature expression.

Despite recent success of 3D point cloud Transformers, the efficiency of the Transformer networks is still below similarly sized CNN counterparts because of massive linear transformation layers in Transformers. By introducing convolution to the ViT [17] structure, CvT [18] achieved better performance and robustness, while concurrently maintaining a high degree of computational and memory efficiency. Therefore, in this work, we hypothesized that combining the strong local modeling ability of CNNs with the remarkable global feature learning ability of Transformers may improve the accuracy for 3D point cloud classification with a lightweight design.

Therefore, we developed a new architecture for point cloud classification, called 3DCTN, to incorporate convolutions into Transformers, achieving the lightweight design and competitive results with state-of-the-art classification methods. Specifically, to avoid computational redundancy, our framework was designed as a hierarchical structure, which had two main modules both operating on the downsampling point sets. Each module consisted of two blocks: the multi-scale LFA block and GFL block, which were achieved by the Graph Convolution and Transformer respectively.

We evaluated the accuracy and efficiency of our classification network on the public synthetic dataset, ModelNet40 [1], and the real scanned dataset, ScanObjectNN [19]. Extensive results showed that our method achieves state-of-the-art classification performance. Additionally, we conducted a detailed investigation and analysis on a series of self-attention variants for better performance, and concluded that the Offset-Attention mechanism and subtraction-form vector attention operator outperform the other variants for our framework.

In summary, the main contributions of our work are as follows:

- We designed a highly expressive module combining Transformers and convolutions, to learn local and global features effectively for point cloud classification;
- Based on such modules, we proposed a multi-scale hierarchical framework, which is suitable for the global feature expression of unstructured point clouds;
- We conducted a detail investigation and analysis of a series of self-attention variants for better performance.

II. Related Work

A. 3D Point Cloud Classification

1) Volume-Based Methods: Similar as 2D image processing, VoxNet [2] introduced the 3D voxelization method to point cloud data processing, which quantized unstructured point clouds to regular volumetric grid forms. Then 3D convolutions were directly applied to point clouds for feature learning. Such methods fail to leverage the sparsity of spatial point clouds because of rasterization. It is also challenging to construct high-resolution voxelization models due to huge computation and memory costs. To address these issues, OctNet [3] proposed an unbalanced grid-octree structure, which allowed higher resolution ($256 \times 256 \times 256$) input than VoxNet. Choy *et al.* [20] used a sparse convolution method, only performing convolution operations at occupied voxels to reduce memory and computational footprint. Despite the great progress volume-based methods made, there are still problems with the loss of geometric information due to the transformation from irregular point clouds to regular 3D voxels.

2) Projection-Based Methods: Projection-based methods are also closely related to 2D image processing. MVCNN [21], as the pioneer of projection-based methods, projected 3D point clouds into multiple views. The features of each view were extracted by 2D CNNs, and then aggregated these features through maxpooling. To improve the robustness and accuracy of the view feature aggregation, several variants have been proposed. View-GCN [22] utilized the Graph Convolution Network (GCN) to establish the relationship between different projection views. Yu *et al.* [23] pointed out the limitations of view-based pooling, and proposed a patch-level pooling method by formulating the view-based 3D classification into a set-to-set matching problem. However, projection-based methods may incur the loss of geometric information during the projection process. Massive view projections tend to cause high computation cost and memory consumption. Additionally, the number and position of projection views is critical to the classification performance, but it is still challenging to chose projection views adaptively for the underlying geometric structure modeling.

3) Point-Based Methods: Taking the 3D coordinates or/and normal features as input, point-based methods deal with the unstructured point clouds directly. The early work, PointNet [6], was introduced by Charles *et al.*, where a deep learning network with MLPs and maxpooling was designed to achieve feature learning. After that, to aggregate local features, PointNet++ [7] applied PointNet in a hierarchical manner and used query ball grouping to construct local neighborhoods. It proved that the hierarchical structure is effective for point cloud feature learning. To further leverage the local information of point clouds, [10], [11] introduced 3D convolution kernels to extract local features, instead of shared MLPs. Due to the unordered nature of neighbor points, PointCNN [10] introduced x-transformation to rearrange the points into a latent and potentially canonical order, followed by using typical convolution to extract local features from point clouds.

Another kind of point-based methods is the Graph-CNN. It establishes connections between local points with a graph, and then models the local geometric information. DGCNN [12] constructed dynamic neighborhoods by features extracted from the former layer, and performed the Graph Convolution on neighborhoods. Unlike DGCNN, ECC [24] defined dynamic convolution-like filters based on edge labels, which also achieved satisfactory results on various datasets. 3DGCN [25] introduced shift and scale-invariance properties to the deep learning networks, and defined learnable kernels with a graph max-pooling mechanism. DeepGCNs [26] applied residual/dense connections and dilated convolution to GCN frameworks to train very deep GCNs. It proved the

positive effect of network depth in GCNs. All these methods showed that the Graph Convolution is good at local feature information aggregating, but nearly none of the aforementioned methods are designed to model long-range context dependencies for the input data.

4) Transformer-Based Methods: Several Transformer-based methods for point cloud classification have been proposed recently. To learn global features of point clouds by Transformers, Point Cloud Transformer (PCT) [13] adopted the PointNet [6] architecture where shared-MLP layers were replaced with standard Transformer blocks. By utilizing the Offset-Attention mechanism and neighborhood information embedding, PCT achieved state-of-the-art performance in point cloud classification. Han *et al.* [15] proposed another point-wise approach to learn global features. Specifically, they used a multi-level Transformer to extract global features of target point clouds with different resolutions. Then, they concatenated these features and fed them into a multi-scale Transformer to obtain the final global features. Instead of extracting global features in a point-wise manner, Point Transformer (PT) [14] applied Transformer layers to local neighborhoods of point clouds, and extracted local features hierarchically through transition down modules. Finally, global features were obtained by a global average pooling operation. However, this method may have incurred information redundancy, since the Point Transformer block was applied to all input points of each layer. Additionally, as a pure Transformer architecture (without CNNs), it would have suffered from high computation and memory costs due to massive linear transformation layers.

B. Vision Transformers

Transformers have achieved a significant success in the field of computer vision as an alternative to CNNs. Many 3D Transformers [27], [28] were developed from 2D Transformers. Therefore, we make a brief introduction to vision Transformers in this subsection. Vision Transformer (ViT) [17] was the first to introduce a pure Transformer framework into the field of 2D image processing and achieved the better results compared with CNNs on large datasets. ViT divided the image into a series of patches, taken as input tokens for the network, followed by applying several Transformer blocks for feature learning. Each Transformer block consisted of two core stages: Multi-Head Attention and Feed Forward. To leverage local information and reduce computational complexity, Swin Transformer [29] proposed a window-based Transformer algorithm, i.e., applying the Transformer to fixed-size windows instead of the global image scale. To build connections with different non-overlapping windows, Swin Transformer introduced a shifted window module. Because of the hierarchical design and cross-window connection, Swin Transformer surpassed the previous state-of-the-art methods in terms of image classification. There also exist several vision Transformer variants [30], [31], [32], [33] which explored ways to better model local features, such as replacing the predefined positional embedding or constructing connections between multiple tokens. Reference [34] investigated a series of self-attention variants and assessed their effectiveness for image processing. By introducing convolutions into the Vision Transformer architecture, CvT [18] combined the benefits of Transformers with the benefits of CNNs for the image classification task. It achieved superior performance while maintaining computational efficiency. Similar approaches [35], [36] were also proposed in the 3D fields for point cloud segmentation and place recognition, but few of them explores the effectiveness of different self attention variants in such framework.

Inspired by CvT [18], we proposed a multi-scale framework that incorporates convolution operations into the Transformer for point cloud classification. Additionally, we also conducted a detailed investigation on different self attention variants to explore the one that best suites our framework, and presented our findings.

III. 3D Convolution-Transformer Network

In this section, we showed how to combine Transformers and convolutions in a hierarchical framework for 3D point cloud classification. We began by presenting the design of our hierarchical network architecture, followed by introducing the convolution-based local feature aggregating and Transformer-based global feature learning process.

A. Overview

The overall pipeline of our 3DCTN is shown in Fig. 1. We presented a hierarchical structure for point cloud classification to improve the efficiency and sensitivity to local geometric layout, which has been proven to be effective by many previous works [7], [14], [29]. Taking the original point cloud as input, the network had two modules operating on downsampling point sets. Each module had two blocks: the LFA block and GFL block, where the former block was based on the Graph Convolution, while the later one was based on the Transformer. In this way, the network effectively combined the strong local modeling ability of CNNs with the remarkable global feature learning ability of Transformers. The numbers of points in sampling point sets were set to $[N/4, N/16]$ for two modules respectively, where N is the number of input points. After two modules above, an additional convolution layer with 1×1 kernel was applied to extend the extracted feature to 1024 dimensions. Then a global max pooling was applied to obtain the final global feature for the target point cloud. Lastly, an MLP Head layer was utilized to get the global classification logits, which consisted of three linear layers with the batch normalization and ReLU.

B. Local Feature Aggregating Block

The LFA block was based on the Graph Convolution, and it was proposed to achieve local feature extraction. By aggregating the local features to corresponding center points (sampling points), this block is able to provide effective discriminative regional feature extraction.

As shown in Fig. 2, given the input point cloud, the farthest point sampling (FPS) was performed to obtain the point cloud subset, called the sampling point set. To ensure the

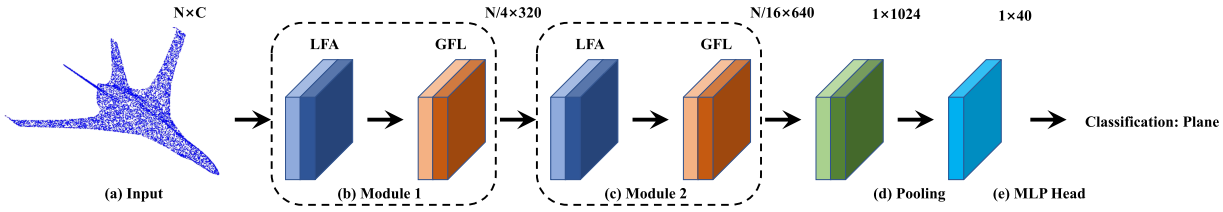


Fig. 1. Hierarchical structure of 3DCTN. It mainly consisted of two modules, and each of them has a Graph Convolution-based LFA block and a Transformer-based GFL block.

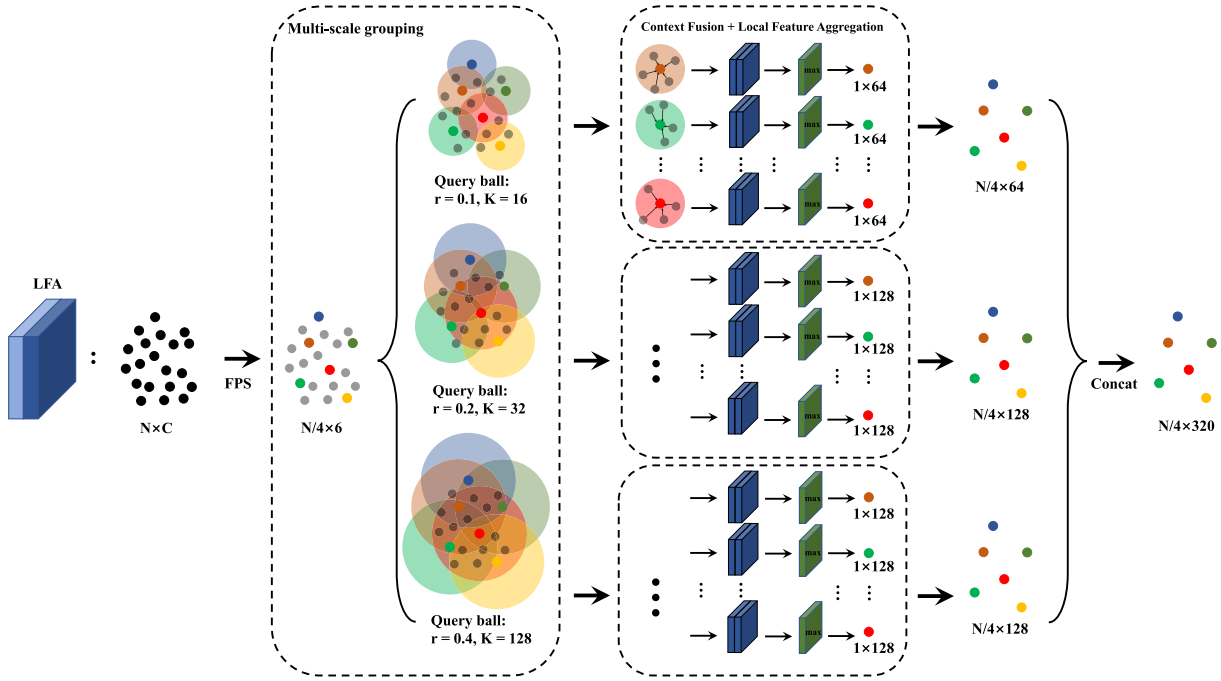


Fig. 2. Multi-scale LFA block. Taking Module 1 in Fig. 1 as an example, the LFA block had three key steps: Multi-scale grouping, Context fusion, and Local feature aggregation.

diversity of the receptive fields for sampling points, multi-scale neighborhoods of each sampling point were constructed by query ball grouping [7]. For each neighborhood of a sampling point, we first presented a context fusion method to encode and combine the coordinates and feature information of the neighborhood, which has been proven to be effective in [37]. Then, we adopted Edge Convolution [12] to aggregate the local features.

1) Context Fusion: Given a neighborhood χ_i of a sampling point x_i, each neighbor point x_j has two kinds of contexts: coordinate context P_j and feature context F_j. The former is used to describe the geometric distribution in 3D space, and the later is used to analyze the semantic information for point cloud classification. To leverage these contexts, we combined both P_j and F_j as:

$$\mathbb{C}_j = concat(F_j, P_j), \tag{1}$$

where $\mathbb{C}_j$ is the combined feature of x_j. Based on such combined features, we define the relationship between x_i and x_j as:

$$\Delta\mathbb{C}_{ij} = concat(F_j - F_i, \mathbb{C}_i). \tag{2}$$

By this way, we are able to encode comprehensive local details for further feature aggregation.

2) Local Feature Aggregation: Having the combined features of points in χ_i, a directed graph $\Psi = \{B, E\}$ was used to describe the local structure of χ_i, where B represents the neighbor points $\{x_j\}_{j=1}^{j=K}$, K is the number of points in χ_i, varying with different modules, and E denotes edge feature operating on the relationship between x_i and x_j. The computation of the edge feature E can be defined as:

$$E_{ij} = f(\Delta\mathbb{C}_{ij}), \tag{3}$$

where $f(*)$ is a nonlinear operator with a set of learnable parameters. There are various ways to choose $f(*)$ [12], and in our work, we defined $f(*)$ as:

$$f(\Delta\mathbb{C}_{ij}) = Conv(\Delta\mathbb{C}_{ij}), \tag{4}$$

where $Conv$ means a point-wise convolution with 1×1 kernels.

After computing all edge features, maxpooling was used to extract the new feature of x_i, which is expressed as:

$$y_i = \underset{B}{maxpooling}\ E_{ij}. \tag{5}$$

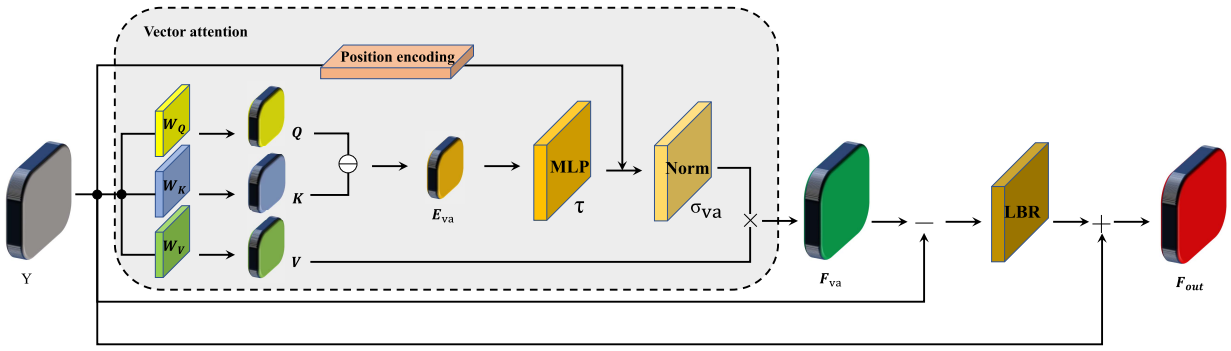


Fig. 3. Transformer-based GFL block, which adopted the offset-attention mechanism.

As such, we can aggregate the local information to the corresponding sampling points for accurate feature representation.

C. Global Feature Learning Block

Taking the aggregated features $Y = \{y_i\}_{i=1}^{i=S}$ of sampling points as input, where S is the number of sampling points, the GFL block adopted the Offset-Attention mechanism with the vector attention operator. Additionally, it also incorporated a learnable position encoding to adaptively capture position information. There was no input (word) embedding in the GFL block, since Y from the LFA block can be considered as the embedded input for the GFL block. In following subsections, we first elaborate on the Offset-Attention mechanism, which was proposed by [13] and achieved a great improvement for point cloud classification. Then, we introduce the learnable position encoding.

1) Offset-Attention: Fig. 3 shows the detailed structure of the Offset-Attention mechanism. Unlike the standard self-attention mechanism, the main idea of Offset-Attention was to adopt a similar operation as a Laplacian matrix $L = D - E$ [38] to replace the adjacency matrix E, where D is the diagonal degree matrix. In particular, the Offset-Attention mechanism can be defined as:

$$F_{out} = OA(Y) = LBR(Y - V_A(Y)) + Y, \tag{6}$$

where F_{out} is the final output of the Offset-Attention mechanism, LBR combines $Linear$, $BatchNorm$, and $ReLU$ layers, $V_A(*)$ represents the vector attention operator which is described in detail later, and $Y - V_A(Y)$ is an offset operator [13] analogous to the Laplacian matrix above. Experiments (Sec. IV-D) showed that the Offset-Attention mechanism outperforms other self-attention mechanisms.

Generally, there are two kinds of self-attention operators: vector attention and scalar attention, where the later has been applied in many previous 3D Transformer works [13], [15], while the former has been proven to be more effective than other operators in the fields of image processing [34] and 3D point cloud processing [14].

Given input features $Y = \{y_i\}_{i=1}^{i=S}$, we first computed *Query*, *Key* and *Value* matrices, $Q = \{q_i\}_{i=1}^{i=S}$, $K = \{k_i\}_{i=1}^{i=S}$, $V = \{v_i\}_{i=1}^{i=S}$, as:

$$\begin{aligned} Q &= W_Q \times Y, \\ K &= W_K \times Y, \\ V &= W_V \times Y, \end{aligned} \tag{7}$$

where W_Q, W_K, W_V are three learnable weight matrices. After that, the standard scalar attention can be formulated as:

$$\begin{aligned} F_{sa} &= E_{sa} \times V \\ &= \sigma_{sa}(Q \times K^{-1} + \rho_{sa}) \times V, \end{aligned} \tag{8}$$

where F_{sa} is the output feature of the scalar attention, E_{sa} is the adjacency matrix calculated by the scalar product between Q and K^{-1}, σ_{sa} is a normalization function: *scale* + *softmax*, and ρ_{sa} represents positional encoding. Essentially, F_{sa} is generated by computing the weighted sum of all vectors in V, according to the adjacency matrix E_{sa}.

Unlike the way of generating the adjacency matrix in the scalar attention, the vector attention used in our paper performed a channel-wise subtraction between Q and K, which can be described as:

$$\begin{aligned} F_{va} &= E_{va} \cdot V \\ &= \sigma_{va}(\tau(Q \ominus K) + \rho_{va}) \cdot \bar{V}, \end{aligned} \tag{9}$$

where F_{va} is the output feature of the vector attention, E_{va} is the channel-wise adjacency matrix, $\bar{V}$ is the expanded *Value* matrix to ensure the same shape of two terms in the equation. Specifically, we assumed that the shape of E_{va} was (B, N, N, D), where B means the batch size, D means the feature dimension of q_i. Accordingly, the shape of V was (B, N, D). To ensure the consistent shape, we expanded V to $\bar{V}$, which had the same shape, (B, N, N, D), as E_{va}. $\tau(*)$ represents an MLP operation to produce the attention map, σ_{va} is a normalization function: *softmax* + l_1 *normalization*, ρ_{va} represents positional encoding which is detailed in the next subsection, and $Q \ominus K$ is defined as:

$$Q \ominus K = \begin{bmatrix} q_1 - k_1 & q_1 - k_2 & \dots & q_1 - k_S \\ q_2 - k_1 & q_2 - k_2 & \dots & q_2 - k_S \\ \dots & \dots & \dots & \dots \\ q_S - k_1 & q_S - k_2 & \dots & q_S - k_S \end{bmatrix} \tag{10}$$

In contrast to the scalar product, $Q \ominus K$ was designed to measure the difference of corresponding channels between two

feature vectors like q_m and k_n. Compared with the scalar attention, the vector attention tends to be more flexible and expressive since each channel of the output feature can be modulated according to the channel-wise adjacency matrix.

2) Position Encoding: In the vector attention, a learnable position encoding was introduced to fuse the local spatial information, which is essential for local feature representation. Similar as standard 2D-aware position embedding [17], we defined the 3D position encoding scheme based on the relative coordinates $\mathbb{P} = \{P_i\}_{i=1}^{i=S}$ of points in χ_i, which can be expressed as:

$$\rho_{va} = \xi(\mathbb{P} \ominus \mathbb{P}), \tag{11}$$

where $\mathbb{P} \ominus \mathbb{P}$ is a matrix representing 3D relative coordinates of points in χ_i:

$$\mathbb{P} \ominus \mathbb{P} = \begin{bmatrix} P_1 - P_1 & P_1 - P_2 & \dots & P_1 - P_S \\ P_2 - P_1 & P_2 - P_2 & \dots & P_2 - P_S \\ \dots & \dots & \dots & \dots \\ P_S - P_1 & P_S - P_2 & \dots & P_S - P_S \end{bmatrix} \tag{12}$$

and ξ represents an MLP operation which consisted of two linear layers separated by batch normalization and ReLU. It was used to extend the feature dimension of relative coordinates from 3 to the same dimension as Q and K, to achieve channel-wise summation in Eq.(9).

IV. Experiments

In this section, we first introduced the implementation of our algorithm, including hardware configuration and hyperparameter settings. Secondly, we presented the performance of our network on the public synthetic and real-scanned datasets, ModelNet40 [1] and ScanObjectNN [19], and compared it with the state-of-the-art works in point cloud classification in terms of accuracy, model size, and processing efficiency. Thirdly, we showed the results from a series of ablation studies to verify the effectiveness of each main component in our framework. Fourthly, we presented detailed investigation and analysis on a series of self-attention variants (Fig. 5) in our network for better performance. Lastly, we illustrated the interpretability of our network by heat map visualization results, which showed that our method is able to understand different shapes by their distinctive features.

A. Implementation Details

We implemented the classification network with Pytorch and trained it on a NVIDIA Tesla V100 GPU. The network was trained with the SGD Optimizer, with a momentum of 0.9 and weight decay of 0.0001. The initial learning rate was set to 0.01, with a cosine annealing schedule to adjust the learning rate at every epoch. We trained the network for 250 epochs and set the batch size as 16. All the involved parameters of our method were empirically set for better performance.

B. Comparison to the State of the Art

We compared our method with the state-of-the-art works including Transformer-based methods and other deep learning-based methods, in terms of classification accuracy, model size, and efficiency.

1) Datasets and Metrics: The ModelNet40 [1] dataset is widely used in 3D point cloud classification. It consists of 12311 CAD-like models in 40 object categories, which have been split into 9843 training models and 2468 testing models. For our this experiment, each model was downsampled to 1024 points with normals by FPS, as input of the network, following PointNet [6]. Since point clouds in ModelNet40 were generated from the corresponding 3D meshes, the normals of point clouds could be obtained directly from the mesh normals. To further evaluate the generalization performance of the method to real objects, the real-scanned dataset, ScanObjectNN [19], was also used in our experiments. It contains $\sim$15, 000 objects that are categorized into 15 categories with 2902 unique object instances. Since each object was segmented from the scene point cloud, point clouds usually include massive outliers like background points, and were corrupted by occlusions and noises. Therefore, it was more challenging to perform shape classification on this dataset. We used the hardest variant of the dataset (PB_T50_RS), and adopted the original train/test split as in [19]. In common with the ModelNet40 dataset, we downsampled each model in the ScanObjectNN dataset to 1024 points, as input of our network. We noted that point clouds in PB_T50_RS have no normal information, we only took the 3D coordinates of point clouds as input.

For evaluation metrics, we utilized the mean accuracy operated on each category ($mAcc$) and the overall accuracy (OA) operated on all classes, which are formulated as:

$$mAcc = \frac{\sum_{i=1}^{K} \frac{T_i}{N_i}}{K}, \qquad OA = \frac{T}{N}, \tag{13}$$

where T is the number of all correctly predicted point clouds, $T = \sum_{i=1}^{K} T_i$, T_i is the number of correctly predicted point clouds in class i, K is the number of classes in the dataset, N is the number of all point clouds in the dataset, $N = \sum_{i=1}^{K} N_i$ and N_i is the number of point clouds in class i. Additionally, we adopted the total number of parameters, FLOPs (FLOating Point operations), and FPS (Frame Per Second) of the benchmarked networks to evaluate the model size and efficiency.

2) Performance Comparison: We compared our method with the state-of-the-art Transformer-based methods and other deep learning-based methods. As shown in Table. I,[1] our method achieved the highest value of $mAcc$ and a competitive value of OA on ModelNet40, thanks to the combination of the Graph Convolution and Transformer. Specifically, our method obtained 91.6% Top-1 $mAcc$, 1.0% higher than PointTransformer [14] and 0.6% higher than GBNet [46]. The results indicate that our method has robust classification performance for different shapes. Additionally, our method also achieved the best results in terms of both $mAcc$ and OA on ScanObjectNN. This demonstrates that our method has a strong generalization performance.

[1]Missing entries are due to lack of source code for particular benchmarked models.

TABLE I
CLASSIFICATION RESULTS ON MODELNET40 AND SCANOBJECTNN

Methods	ModelNet40			ScanObjectNN			Parameters (MB)	FLOPs (GB)	FPS
	Input Size	mAcc (%)	OA (%)	Input Size	mAcc (%)	OA (%)			
Other Learning-based Methods									
3DShapeNets [1]	1024	77.3	84.7	-	-	-	-	-	-
PointNet [6]	1024	86.0	89.2	1024	63.4	68.2	3.47	0.45	**614**
PointNet++ [7]	1024	88.2	91.9	1024	75.4	77.9	1.74	4.09	16
diffConv [39]	1024	90.4	93.2	-	-	-	2.08	**0.16**	-
CurveNet [40]	1024	90.4	93.1	-	-	-	-	-	-
PointCNN [10]	1024	88.1	92.2	1024	75.1	78.5	**0.6**	1.54	14
DGCNN [12]	1024	90.2	92.2	1024	73.6	78.1	1.81	2.43	279
FatNet [41]	1024	90.6	93.2	-	-	-	-	-	-
DRNet [42]	1024	-	93.1	1024	78.0	80.3	-	-	-
Transformer-based Methods									
PATs [43]	1024	-	91.7	-	-	-	-	-	-
LFT-Net [16]	1024	89.7	93.2	-	-	-	-	-	-
PointTransformer [44]	1024	89.0	92.8	1024	75.3	77.6	13.86	9.36	17
MLMST [15]	1024	-	92.9	-	-	-	-	-	-
PointCloudTransformer [13]	1024	90.3	93.2	1024	77.1	80.5	2.80	2.02	**125**
PointTransformer [14]	1024	90.6	93.7	1024	78.2	80.8	9.14	17.14	15
CloudTransformers [45]	1024	90.8	93.1	-	-	-	22.91	12.69	12
GBNet [46]	1024	91.0	**93.8**	1024	77.8	80.5	8.38	9.02	102
Ours (Single-scale)	1024	**91.2**	92.7	1024	**79.0**	**81.3**	**1.82**	**1.08**	30
Ours (Multi-scale)	1024	**91.6**	93.2	1024	**79.5**	**81.5**	4.21	3.76	24

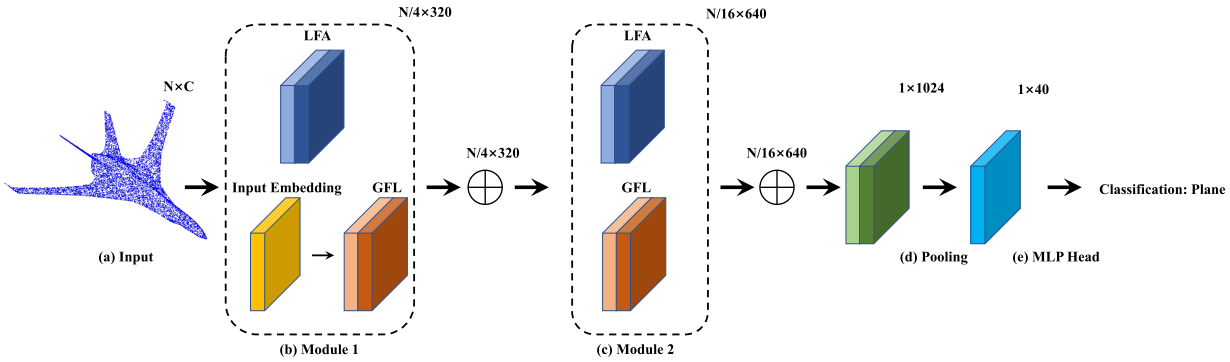


Fig. 4. Parallel pattern of our method. An additional MLP-based input embedding was added before the first GFL block.

As shown in Table. I, thanks to our hierarchical structure and light-weight LFA blocks, our method with single-scale neighborhood had fewer parameters and FLOPs than other Transformer-based approaches. This confirms the lightweight design of our network. Our single-scale model had only 30.4% of the parameters and used only 11.5% of the FLOPs, compared with PointTransformer. However, since downsampling and neighborhood building operations in the LFA block were very time-consuming, our method had no remarkable superiority in terms of FPS. Improving the efficiency of our network by optimizing the sampling and grouping operations could be one of our potential future research directions.

C. Ablation Study

In this section, we presented the results of various ablation experiments to evaluate the effectiveness of each main component of our framework on ModelNet40.

1) Hierarchical Structure: The proposed hierarchical structure can aggregate local features effectively, with the significant reduction of computational and memory costs. As shown in Table. II (Row 2), for the network without the hierarchical structure, it is inevitable to require more computational and memory costs, since the multi-scale local feature aggregating and global feature learning were performed on each point. This results suggest that the proposed hierarchical structure is effective to reduce the model size for our framework.

Additionally, we also compared different hierarchical patterns: cascade design and parallel design, where the former is shown in Fig. 1, and the later is shown in Fig. 4. For the parallel-design network, since the features from the LFA block could not be taken as the embedded input to the GFL block, we added an additional MLP-based input embedding before the first GFL block. From the results shown in Table. II, we saw that the cascade design (Row 9) achieved a slight improvement of accuracy and efficiency compared with the parallel design (Row 2). This was because the first GFL block lacks local information, causing poor feature representation. Moreover, the additional input embedding modules increased the model size of the parallel-design network.

2) Multi-Scale Strategy: We compared the multi-scale local feature aggregating with the single-scale (middle scale) way, and showed the results in Table. II (Row 3). Fig. 2 shows

TABLE II
ABLATION STUDY

Ablation		mAcc (%)	OA (%)	Parameters (MB)	FLOPs (GB)	FPS
Hierarchical structure	✗	-	-	4.22	92.36	-
	Hierarchical (parallel design)	91.2	93.1	4.42	3.84	21
Multi-scale neighborhood	✗	91.2	92.7	1.82	**1.08**	**30**
Local feature aggregating	✗	87.5	90.6	4.19	1.74	27
	Transformer-based	-	-	4.99	51.54	-
Global feature learning	✗	91.0	93.1	**1.76**	2.40	28
	Graph Convolution-based	90.4	92.7	2.87	2.51	27
Position encoding	✗	90.9	93.1	4.22	4.06	24
Self-attention mechanism	Standard mechanism	90.4	91.9	4.22	4.06	25
Self-attention operator	Scalar attention	90.5	93.1	4.21	2.65	27
3DCTN		**91.6**	**93.2**	4.22	4.06	24

TABLE III
CLASSIFICATION RESULTS OF DIFFERENT PARAMETER SETTINGS FOR THE MULTI-SCALE STRATEGY

Setting	LFA1		LFA2		ModelNet40		ScanObjectNN	
	Radiuses	K-neighbors	Radiuses	K-neighbors	mAcc (%)	OA (%)	mAcc (%)	OA (%)
1-1	[0.1, 0.2, 0.4]	[16, 32, 64]	[0.2, 0.4, 0.8]	[16, 32, 64]	91.4	92.8	79.2	80.1
1-2	[0.1, 0.2, 0.4]	[16, 32, 64]	[0.2, 0.4, 0.8]	[32, 64, 128]	**91.6**	**93.2**	**79.5**	**81.5**
1-3	[0.1, 0.2, 0.4]	[16, 32, 64]	[0.4, 0.8, 1.0]	[64, 128, 192]	91.5	93.0	79.4	81.2
2-1	[0.1, 0.2, 0.4]	[16, 32, 128]	[0.2, 0.4, 0.8]	[16, 32, 64]	90.9	92.2	78.0	80.2
2-2	[0.1, 0.2, 0.4]	[16, 32, 128]	[0.2, 0.4, 0.8]	[32, 64, 128]	90.7	92.9	77.8	79.6
2-3	[0.1, 0.2, 0.4]	[16, 32, 128]	[0.4, 0.8, 1.0]	[64, 128, 192]	91.4	92.9	79.1	81.0
3-1	[0.2, 0.4, 0.6]	[32, 64, 128]	[0.2, 0.4, 0.8]	[16, 32, 64]	91.2	92.5	78.7	80.5
3-2	[0.2, 0.4, 0.6]	[32, 64, 128]	[0.2, 0.4, 0.8]	[32, 64, 128]	90.8	92.7	78.8	81.4
3-3	[0.2, 0.4, 0.6]	[32, 64, 128]	[0.4, 0.8, 1.0]	[64, 128, 192]	91.0	92.7	79.1	80.8

that the multi-scale aggregating built three neighborhoods with different radiuses for each sampling point, while the single-scale strategy only used the middle-scale neighborhood to achieve local feature aggregating. From the comparison results, the classification accuracy of the single-scale strategy was lower than the multiple-scale strategy. This demonstrates that multiple receptive fields are able to improve the local feature aggregating. In terms of the lightweight design, single-scale strategy led to a significant reduction of 57% in parameters, and 73% in FLOPs.

Additionally, we also conducted experiments to choose the optimal parameter setting for the multi-scale strategy. As shown in Table. III, we saw that the parameter setting of 1-2 achieved the highest classification accuracy on both ModelNet40 and ScanObjectNN datasets. Since the multi-scale strategy aimed at aggregating local features, a large-scale neighborhood tended to include many irrelevant points, lowering the classification performance. Therefore, we chose the 1-2 setting as the optimal setting of the multi-scale strategy.

3) Local Feature Aggregating: We studied the effectiveness of the proposed LFA block (Sec. III-B), and Table. II (Row 4) shows the results. We first removed the LFA block, making the framework a pure Transformer network. From the results, without the local feature aggregating, the performance dropped significantly, which highlights the importance of the LFA block. Next we replaced the Graph Convolution in LFA with the Transformer block, and we saw that the total parameters and FLOPs of the network were much higher than the original. These results confirms that local feature aggregating based on the Graph Convolution has better local modeling ability, and is crucial in reducing the memory and computational footprint of our model.

4) Global Feature Learning: We also studied the choice of different global feature learning methods in the GFL block (Sec. III-C), and the results are shown in Table. II (Row 5). According to the results, the removal of the GFL block caused a drop in classification performance. When we replaced the Transformer block with the Graph Convolution, the performance dropped again, which confirms the superiority of the Transformer-based GFL block.

5) Position Encoding: The position encoding in Eq.(9) can introduce the spatial difference of the input words (points) to the attention map. We conducted an ablation study on that and showed results in Table. II (Row 6). Without the position encoding, both $mAcc$ and OA were lower than the original, indicating the effectiveness of such component.

6) Self-Attention Mechanism: We compared the Offset-Attention in the GFL block with the standard self-attention mechanism (shown in Fig. 5(a)) in Table. II (Row 7). From the results, by using the standard self-attention mechanism, we observed a 0.8% and 1.4% drop in $mAcc$ and OA respectively. This suggests that the Offset-Attention mechanism outperforms the standard self-attention mechanism. Additionally, we conducted a detailed investigation about

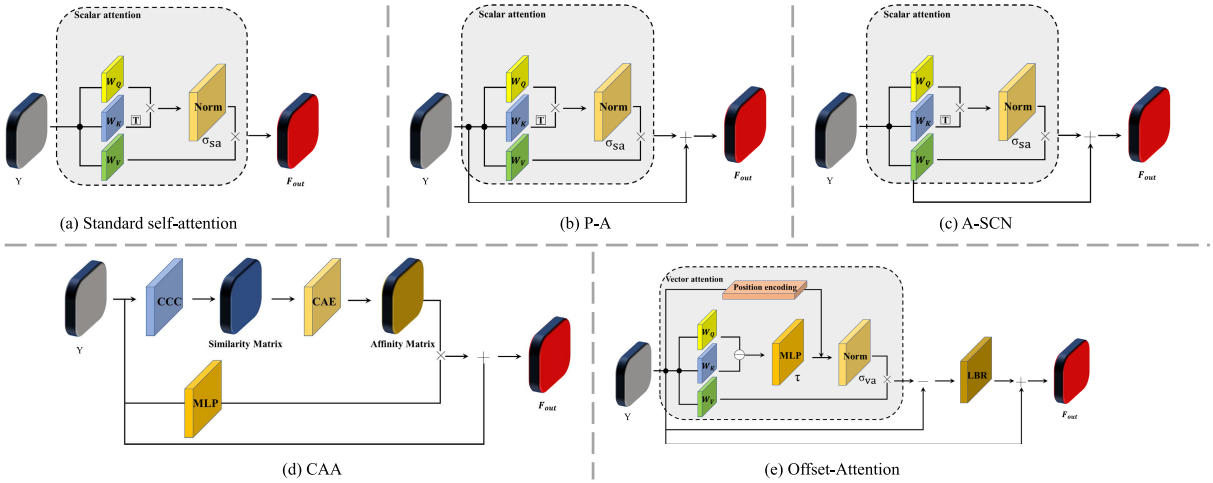


Fig. 5. Architectures of various self-attention mechanisms for 3D point cloud processing. (a) Standard self-attention. (b) P-A [35]. (c) A-SCN [47]. (d) CAA [46]. (e) Offset-Attention.

TABLE IV
INVESTIGATION OF SELF-ATTENTION MECHANISMS

Self-attention mechanisms	mAcc (%)	OA (%)	Parameters (MB)	FLOPs (GB)	FPS
Standard	90.4	91.9	4.22	4.06	25
A-SCN [47]	90.9	92.9	3.72	2.60	28
P-A [35]	90.8	92.8	4.21	2.65	27
CAA [46]	91.2	93.0	**2.27**	**2.45**	**28**
Offset-Attention [13]	**91.6**	**93.2**	4.22	4.06	24

different self-attention mechanisms for our framework, and the architectures of there mechanisms are shown in Fig 5. See Sec. IV-D for more results.

7) Self-Attention Operator: We also investigated the types of the self-attention operator used in the GFL block. Generally, there are two commonly-used self-attention operators: vector attention and scalar attention. As shown in Table. II (Row 8), the scalar attention led to a drop of accuracy (0.7% in $mAcc$ and 0.2% in OA), compared with the vector attention. This demonstrates the superiority of the vector attention.

D. Investigation on 3D Attentions

In this section, we presented a detailed investigation on self-attention mechanisms and operators for better performance in point cloud classification. This investigation is also expected to provide some benefit references for Transformer-based classification works.

1) Self-Attention Mechanisms: We collected a series of self-attention mechanisms widely used in the 3D point cloud processing. As shown in Fig. 5, the standard self-attention mechanism includes three linear layers to generate *Query*, *Key* and *Value* matrices. The attention map is estimated by comparing *query* and *key*, and then normalized to limit the variance of the matrix. The final output can be obtained by multiplying the attention matrix and *value* matrix. Attentional ShapeContextNet (A-SCN) [47] and Point-Attention (P-A) [35] had similar architectures to the standard self-attention mechanism, with a key difference being both applied residual connection operations to strengthen the connection between the input and output. Therefore, they both achieved better classification results than the standard self-attention mechanism. In contrast with the aforementioned point-wise mechanisms, Channel-wise Affinity Attention (CAA) [46] focused on the channel space, and achieved an outstanding performance in point cloud classification. It first used a Compact Channel-wise Comparator block (CCC) to generate the similarity matrix efficiently, followed by introducing a Channel Affinity Estimator block (CAE) to generate the affinity matrix which was able to sharpen the attention weights and reduce the redundant information. Lastly, the output was calculated by multiplying the affinity matrix and *value* matrix, with the same residual connection as P-A [35]. The results in Table. IV show the Offset-Attention mechanism achieves a better result than the other, which suggests that it is more suitable to our framework. Additionally, CAA [46] also achieved satisfactory performance terms of both accuracy and efficiency.

Self-attention Operators. The effectiveness of different self-attention operators has been studied in the field of 2D image processing by [34], but there is still no such investigation in point cloud classification. Self-attention operators can be generally divided into two types: scalar attention and vector attention. The former operates on feature-level similarity estimation, while the later operates on channel-level

TABLE V
INVESTIGATION OF SELF-ATTENTION OPERATORS

Self-attention operators		mAcc (%)	OA (%)	Parameters (MB)	FLOPs (GB)	FPS
Scalar attention	Dot product	90.5	93.1	4.21	2.65	**26**
Vector attention	Concatenation	89.6	93.1	4.43	6.17	21
	Summation	90.6	92.9	4.22	4.06	24
	Subtraction	**91.6**	**93.2**	4.22	4.06	24
	Division	88.8	91.9	4.22	4.06	24
	Hadamard product	90.1	92.9	**4.22**	**4.06**	24

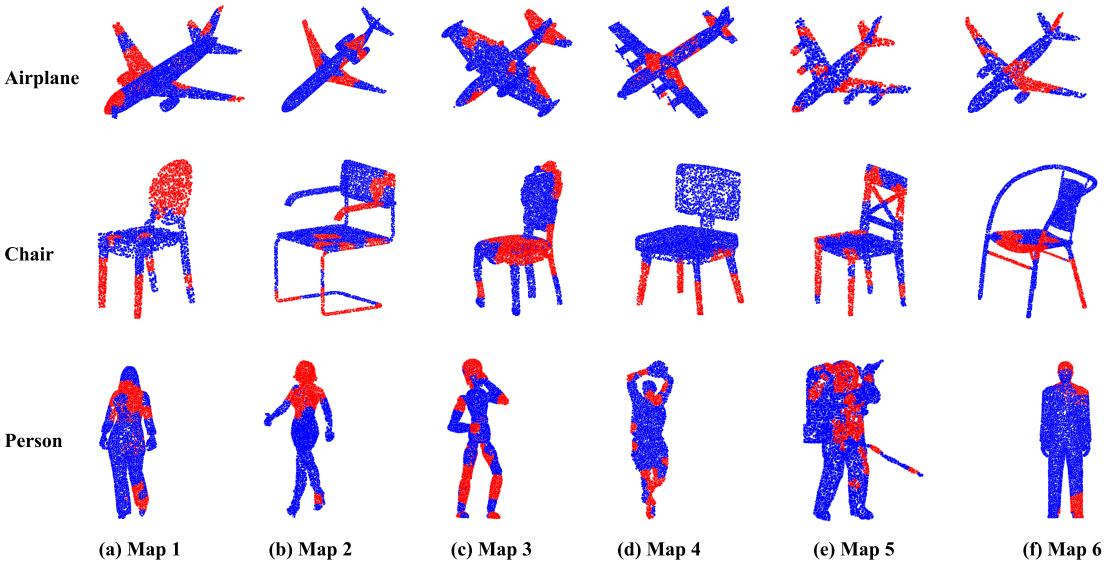


Fig. 6. Heat map visualization. First row: Airplane, Second row: Chair, Third row: Person. As can be seen, the attention (red) is focused on wings and tails for airplanes, on legs and backs for chairs, and on heads and limbs for persons.

estimation. The detailed definition of each operator is shown below:

$$\begin{aligned} &\textbf{scalar attention}: F_{sa} = E_{sa} \times V = \sigma_{sa}(Q \times K^{-1}) \times V, \\ &\textbf{vector attention}: F_{va} = E_{va} \cdot V = \sigma_{va}(\tau(\Delta(Q, K)) \cdot \bar{V}, \end{aligned} \tag{14}$$

where the representative form of the scalar attention is the *dot product*, and Δ in the second equation represents various forms of the vector attention, which can be formulated as:

$$\Delta = \begin{bmatrix} \delta(q_1, k_1) & \delta(q_1, k_2) & \dots & \delta(q_1, k_S) \\ \delta(q_2, k_1) & \delta(q_2, k_2) & \dots & \delta(q_2, k_S) \\ \dots & \dots & \dots & \dots \\ \delta(q_S, k_1) & \delta(q_S, k_2) & \dots & \delta(q_S, k_S) \end{bmatrix} \tag{15}$$

where δ can be expressed as different forms:

$$\begin{aligned} \textit{Concatenation}: \delta(q_i, k_i) &= [q_i, k_i], \\ \textit{Summation}: \delta(q_i, k_i) &= q_i + k_i, \\ \textit{Subtraction}: \delta(q_i, k_i) &= q_i - k_i, \\ \textit{Division}: \delta(q_i, k_i) &= q_i / k_i, \\ \textit{Hadamard product}: \delta(q_i, k_i) &= q_i \cdot k_i. \end{aligned} \tag{16}$$

We applied all operators above to our framework, to evaluate the their performance. The results in Table. V show that the scalar attention outperforms most channel-wise operators except the subtraction-form vector attention. The results suggestes that the subtraction-form vector attention operator is most suitable for our point cloud classification framework.

E. Heat Map Visualization

To highlight the interpretability of our network, we generated attention map visualization results by utilizing the Grad-CAM method [48] which is commonly used in the 2D fields. As shown in Fig. 6, heat maps show the different regions of interest (ROI) of the network for different types of point clouds. For the airplane category, our network focused more on airplane wings and tail, which are key discriminative geometric features. For the chair category, the ROI of our network were the chair leg and back, which makes them distinguishable from desks. For the person category, our network focused more on the head and limbs, which is also consistent with our common sense of person classification. Specially, as shown in Fig. 6 Row 3 (e), when there were interference point sets (like backpack), our network was still able to focus on the key areas of the person, which illustrates the robust performance of our network.

V. Conclusion

In this paper, we proposed a hierarchical framework that incorporated convolutions into Transformers for 3D point cloud classification. Taking a downsampled point cloud as input, our method had two main modules to extract the global feature progressively. In each module, we first aggregated the multi-scale local features by the Graph Convolution and then learned the global features by the Transformer. As such, Our network was able to combine the strong local modeling ability of CNNs with the remarkable global feature learning ability of Transformers. After that, we added an additional convolution layer with a 1×1 kernel, a maxpooling operation, and an MLP head layer sequentially, to generate final classification results. To explore the better classification performance, we conducted a detailed investigation on a series of self-attention variants. To demonstrate the effectiveness of the proposed method, we designed and performed a number of ablation experiments on main components of our framework. Extensive experiments on ModelNet40 and ScanObjectNN datasets proved the effectiveness of our method, and showed that it achieves state-of-the-art classification performance with a lightweight design.

Future work. There have been many studies on the combination of the convolution and Transformer in 2D images, and we believe that it will also be a promising research direction in 3D point cloud processing. In the future, we will further develop our framework and extend it to more complex 3D applications, such as point cloud segmentation and object detection.

References

[1] Z. Wu *et al.*, "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1912–1920, doi: 10.1109/CVPR.2015.7298801.

[2] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IROS*, Oct. 2015, pp. 922–928, doi: 10.1109/IROS.2015.7353481.

[3] G. Riegler, A. O. Ulusoy, and A. Geiger, "OctNet: Learning deep 3D representations at high resolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 3577–3586, doi: 10.1109/CVPR.2017.701.

[4] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," Mar. 2018, *arXiv:1803.10091*.

[5] T. Le and Y. Duan, "Pointgrid: A deep network for 3D shape understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9204–9214, doi: 10.1109/CVPR.2018.00959.

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 652–660, doi: 10.1109/CVPR.2017.16.

[7] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, Dec. 2017, pp. 5099–5108.

[8] A. Paigwar, O. Erkent, C. Wolf, and C. Laugier, "Attentional PointNet for 3D-object detection in point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1297–1306, doi: 10.1109/CVPRW.2019.00169.

[9] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 918–927, doi: 10.1109/CVPR.2018.00102.

[10] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 820–830.

[11] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9621–9630, doi: 10.1109/CVPR.2019.00985.

[12] 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 Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Nov. 2019, doi: 10.1145/3326362.

[13] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "PCT: Point cloud transformer," *Comput. Vis. Media*, vol. 7, no. 2, pp. 187–199, Jun. 2021, doi: 10.1007/s41095-021-0229-5.

[14] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16259–16268.

[15] X.-F. Han, Y.-J. Kuang, and G.-Q. Xiao, "Point cloud learning with transformer," Apr. 2021, *arXiv:2104.13636*.

[16] Y. Gao, X. Liu, J. Li, Z. Fang, X. Jiang, and K. M. S. Huq, "LFT-Net: Local feature transformer network for point clouds analysis," *IEEE Trans. Intell. Transp. Syst.*, early access, Feb. 1, 2022, doi: 10.1109/TITS.2022.3140355.

[17] A. Dosovitskiy *et al.*, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, May 2020, pp. 1–12.

[18] H. Wu *et al.*, "CvT: Introducing convolutions to vision transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 22–31, doi: 10.1109/ICCV48922.2021.00009.

[19] 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 *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1588–1597, doi: 10.1109/ICCV.2019.00167.

[20] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 3075–3084, doi: 10.1109/CVPR.2019.00319.

[21] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 945–953, doi: 10.1109/ICCV.2015.114.

[22] X. Wei, R. Yu, and J. Sun, "View-GCN: View-based graph convolutional network for 3D shape analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 1850–1859, doi: 10.1109/CVPR42600.2020.00192.

[23] T. Yu, J. Meng, M. Yang, and J. Yuan, "3D object representation learning: A set-to-set matching perspective," *IEEE Trans. Image Process.*, vol. 30, pp. 2168–2179, 2021, doi: 10.1109/TIP.2021.3049968.

[24] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Dec. 2017, pp. 3693–3702, doi: 10.1109/CVPR.2017.11.

[25] Z.-H. Lin, S.-Y. Huang, and Y.-C.-F. Wang, "Convolution in the cloud: Learning deformable kernels in 3D graph convolution networks for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1800–1809, doi: 10.1109/CVPR42600.2020.00187.

[26] G. Li, M. Müller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9267–9276, doi: 10.1109/ICCV.2019.00936.

[27] K. Fu, P. Gao, R. Zhang, H. Li, Y. Qiao, and M. Wang, "Distillation with contrast is all you need for self-supervised point cloud representation learning," 2022, *arXiv:2202.04241*.

[28] X. Lai *et al.*, "Stratified transformer for 3D point cloud segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Mar. 2022, pp. 8500–8509.

[29] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted Windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 9992–10002, doi: 10.1109/ICCV48922.2021.00986.

[30] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, p. 12, doi: 10.1109/ICCV48922.2021.01196.

[31] L. Yuan *et al.*, "Tokens-to-token vit: Training vision transformers from scratch on ImageNet," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 539–547, doi: 10.1109/ICCV48922.2021.00060.

[32] W. Wang *et al.*, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 548–558, doi: 10.1109/ICCV48922.2021.00061.

[33] Y. Wang *et al.*, "End-to-end video instance segmentation with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8741–8750.

[34] H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10073–10082, doi: 10.1109/CVPR42600.2020.01009.

[35] M. Feng, L. Zhang, X. Lin, S. Z. Gilani, and A. Mian, "Point attention network for semantic segmentation of 3D point clouds," *Pattern Recognit.*, vol. 107, Nov. 2020, Art. no. 107446, doi: 10.1016/j.patcog.2020.107446.

[36] L. Hui, H. Yang, M. Cheng, J. Xie, and J. Yang, "Pyramid point cloud transformer for large-scale place recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 6078–6087, doi: 10.1109/ICCV48922.2021.00604.

[37] S. Qiu, S. Anwar, and N. Barnes, "PU-transformer: Point cloud upsampling transformer," Nov. 2021, *arXiv:2111.12242*.

[38] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*.

[39] M. Lin and A. Feragen, "DiffConv: Analyzing irregular point clouds with an irregular view," 2021, *arXiv:2111.14658*.

[40] A. A. M. Muzahid, W. Wan, F. Sohel, L. Wu, and L. Hou, "CurveNet: Curvature-based multitask learning deep networks for 3D object recognition," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 6, pp. 1177–1187, Jun. 2021, doi: 10.1109/JAS.2020.1003324.

[41] C. Kaul, N. Pears, and S. Manandhar, "FatNet: A feature-attentive network for 3D point cloud processing," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 7211–7218, doi: 10.1109/ICPR48806.2021.9412731.

[42] S. Qiu, S. Anwar, and N. Barnes, "Dense-resolution network for point cloud classification and segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 3813–3822, doi: 10.1109/WACV48630.2021.00386.

[43] J. Yang *et al.*, "Modeling point clouds with self-attention and Gumbel subset sampling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 3323–3332, doi: 10.1109/CVPR.2019.00344.

[44] N. Engel, V. Belagiannis, and K. Dietmayer, "Point transformer," *IEEE Access*, vol. 9, pp. 134826–134840, 2021, doi: 10.1109/ACCESS.2021.3116304.

[45] K. Mazur and V. Lempitsky, "Cloud transformers: A universal approach to point cloud processing tasks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, p. 10, doi: 10.1109/ICCV48922.2021.01054.

[46] S. Qiu, S. Anwar, and N. Barnes, "Geometric back-projection network for point cloud classification," *IEEE Trans. Multim.*, vol. 24, pp. 1943–1955, 2022, doi: 10.1109/TMM.2021.3074240.

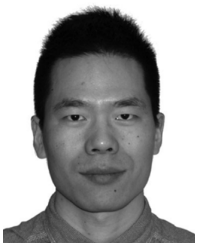
[47] S. Xie, S. Liu, Z. Chen, and Z. Tu, "Attentional ShapeContextNet for point cloud recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4606–4615, doi: 10.1109/CVPR.2018.00484.

[48] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. 29th IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2921–2929, doi: 10.1109/CVPR.2016.319.

Dening Lu received the B.Sc. and M.Sc. degrees in electrical engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), China, in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree in systems design engineering with the Geospatial Sensing and Data Intelligence Group, University of Waterloo, Canada. He has published papers in the IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT and ICCV. His research interests include 3D point cloud processing and deep learning.

Qian Xie received the B.Sc. and Ph.D. degrees in electrical engineering from the Nanjing University of Aeronautics and Astronautics (NUAA), China, in 2014 and 2021, respectively. He is currently a Research Associate with the Department of Computer Science, University of Oxford, U.K. He works with the Cyber-Physical Systems Group under the supervision of Prof. Niki Trigoni and Prof. Andrew Markham. Prior to Oxford, he went to Cardiff University, U.K., as a joint-trained Ph.D. Student in 2019 for 18 months. He publishes extensively in venues and journals such as CVPR, ICCV, IEEE ROBOTICS AND AUTOMATION LETTERS, IEEE TRANSACTIONS ON MULTIMEDIA, and the *International Journal of Computer Vision*. His research interests are 3D vision, point cloud processing, deep learning, and scene understanding.

Kyle (Yilin) Gao (Graduate Student Member, IEEE) received the bachelor's degree in mathematics from the University of Waterloo, Canada, in 2016, and the master's degree in physics from the University of Victoria, Canada, in 2020. He is currently pursuing the Ph.D. degree in systems design engineering with the Geospatial Sensing and Data Intelligence Group, University of Waterloo. He has published articles in the *International Journal of Applied Earth Observation and Geoinformation*. His research interests include computer vision and deep learning.



Linlin Xu (Member, IEEE) received the B.Eng. and M.Sc. degrees in geomatics engineering from the China University of Geosciences, Beijing, China, in 2007 and 2010, respectively, and the Ph.D. degree in geography from the Department of Geography and Environmental Management, University of Waterloo, Waterloo, ON, Canada, in 2014. He is currently a Research Assistant Professor with the Department of Systems Design Engineering, University of Waterloo. He has published various papers on high-impact remote sensing journals and conferences. His research interests include hyperspectral and synthetic aperture radar data processing and their applications in various environmental applications.

Jonathan Li (Senior Member, IEEE) received the Ph.D. degree in geomatics engineering from the University of Cape Town, South Africa, in 2000. He is currently a Professor of geomatics and systems design engineering with the University of Waterloo, Canada. He has supervised more than 120 master's and Ph.D. students as well as post-doctoral fellows to completion. He has coauthored more than 490 publications, more than 320 of which were published in refereed journals, including IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING, *ISPRS-JPRS*, *RSE*, and *JAG*. He has also published papers in flagship conferences in computer vision and AI, including CVPR, AAAI, and IJCAI. His main research interests include AI-based information extraction from earth observation images, LiDAR point clouds, 3D vision, and GeoAI. He is a fellow of The Canadian Academy of Engineering and the Engineering Institute of Canada. He is the Editor-in-Chief of the *International Journal of Applied Earth Observation and Geoinformation* (JAG) and an Associate Editor of the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, and *Canadian Journal of Remote Sensing*.