Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 31(12): 7365–7384.
DOI: 10.3934/era.2023373
Received: 19 September 2023
Revised: 05 November 2023
Accepted: 08 November 2023
Published: 17 November 2023

Research article

A feature fusion-based attention graph convolutional network for 3D classification and segmentation

Chengyong Yang[1,3], Jie Wang[1], Shiwei Wei[2,*] and Xiukang Yu[1]

[1] School of Information Science and Engineering, Guilin University of Technology, Guangxi 541006, China

[2] School of Computer Science and Engineering, Guilin University of Aerospace Technology, Guangxi 541004, China

[3] Network and Information Center, Guilin University of Technology, Guangxi 541006, China

* **Correspondence:** Email: swwei@guat.edu.cn.

Abstract: Among all usual formats of representing 3D objects, including depth image, mesh and volumetric grid, point cloud is the most commonly used and preferred format, because it preserves the original geometric information in 3D space without any discretization and can provide a comprehensive understanding of the target objects. However, due to their unordered and unstructured nature, conventional deep learning methods such as convolutional neural networks cannot be directly applied to point clouds, which poses a challenge for extracting semantic features from them. This paper proposes a feature fusion algorithm based on attention graph convolution and error feedback, which considers global features, local features and the problem of the features loss during the learning process. Comparison experiments are conducted on the ModelNet40 and ShapeNet datasets to verify the performance of the proposed algorithm, and experimental results show that the proposed method achieves a classification accuracy of 93.1% and a part segmentation mIoU (mean Intersection over Union) of 85.4%. Our algorithm outperforms state-of-the-art algorithms, and effectively improves the accuracy of point cloud classification and segmentation with faster convergence speed.

Keywords: 3D point cloud; point cloud classification and segmentation; attention graph convolution; error feedback mechanism; feature fusion

1. Introduction

With the rapid development of 3D sensing technology, 3D vision has gradually become a current research hotspot [1]. Three-dimensional data plays an extremely important role in various fields,

including remote sensing [2], unmanned aerial vehicles [3], autonomous driving [4] and image segmentation [5]. Especially in the medical image segmentation field, for instance, optical coherence tomography (OCT) is an important standard for clinical retinal diagnosis [6] and widely used in imaging, cardiology, dermetology, etc. [7]. Traditional medical image segmentation methods are known for their information deficiency and low accuracy [8]. Therefore, enhancing accuracy and speed in medical image segmentation is a primary research focus. Currently, mainstream representations of 3D data include voxels, meshes and point clouds [9]. Voxel representation can be regarded as quantized and fixed-sized point clouds, similar to three-dimensional space pixels [10]. However, it has drawbacks such as insufficient resolution and high memory cost. Mesh representation is composed of polygons constructed by adjacent point clouds of an object [11], representing non-Euclidean data that is not directly related to the output of 3D sensors. Compared to voxels and meshes, 3D point cloud data avoids the complexity of data description [12] and becomes the preferred representation for 3D data due to its ability to effectively describe geometric information in 3D space, easy acquisition and no need for discretization. However, point cloud data possess characteristics such as disorder, non-structured nature and complex scenes [13] in 3D space, making analysis tasks based on point cloud data challenging.

In the early stages of point cloud research, manual feature extraction methods were used to extract point cloud features [14]. For instance, point feature histograms (FSPH, shapeme histogram), etc., were employed. However, manual feature extraction methods are usually limited to solving specific problems. With the increasing types and quantities of 3D point cloud data, manual extraction methods suffer from drawbacks like low efficiency and accuracy, making it difficult to extend them to other point cloud tasks.

With the widespread application of deep learning, especially the extensive use of convolutional neural networks (CNNs) in 2D vision research [15], 2D vision research has reached a relatively mature stage. However, for unordered and non-structured 3D point clouds, regular 2D convolutional kernels cannot be directly applied. To address these challenges in 3D point cloud research, early research attempted to transform unordered point clouds into regular structures (such as multi-view, voxel meshes, etc.) that can be handled by CNNs [16, 17]. Although these methods achieved acceptable feature extraction results, they often resulted in partial information loss during the point cloud data transformation process, and also faced issues of computational complexity and high workload [18].

In 2016, the PointNet network model was proposed and could be directly used to handle point clouds [19], which effectively extracts global features of point clouds through multi-layer perceptrons (MLPs), and performs well in point cloud classification and segmentation tasks. Consequently, direct processing of point clouds became a new research focus [20], and some improved methods had been continually proposed for addressing point clouds, such as PointNet++ [21], PointCNN [22], and DGCNN [23].

New methods for directly processing point cloud data have emerged, and at present they mainly focus on how to reduce processing losses and enhance classification accuracy. MFNet [24] introduces a RBFLayer to extract point cloud features, incorporating spatial location, geometric features of neighboring points, and feature correlations. This comprehensive approach effectively captures rich point cloud features. CSANet [25] constructs a point cloud feature extraction network by stacking cross self-attention network (CSA) modules and incorporates a multi-scale fusion (MF) module for feature fusion in the segmentation process. Multiscale receptive fields graph attention network (MRF-GAT) [26] employs a multi-scale graph attention network to capture an extensive range of point cloud features, demonstrating its effectiveness in partial feature extraction. Methods based on convolutional

neural networks tend to lose original point cloud information during data transformation and involve complex computations. On the other hand, methods based on PointNet can directly handle point cloud data, avoiding information loss during data transformation. However, PointNet and most of its derivatives often overlook information from neighboring points. Graph-based methods, while improving point cloud data processing, mainly focus on feature extraction, with limited consideration for global and local feature extraction and potential feature loss in the local feature extraction process.

This study proposes a novel graph attention convolutional network that combines attention mechanisms with graph convolution for local feature extraction from point clouds. This study also introduces a global attention module for capturing point cloud local contextual features and employ a feedback feature fusion module for local feature fusion, enabling point cloud classification and segmentation tasks. Our main contributions can be summarized as follows:

1) This study designs a graph attention convolution operation that focuses on both the center point and its neighboring points. By incorporating attention mechanisms into graph convolution, this study can effectively extracts features.

2) This study develops a feature fusion method by employing an error feedback mechanism, which can effectively reduce biases introduced during feature fusion.

3) This study introduces a global attention module for extracting global contextual features from the point cloud.

Experimental results conducted on standard datasets demonstrate that our method performs better than state-of-the-art methods in terms of model training, point cloud classification and segmentation.

The rests of this paper are organized as follows: Section 2 reviews existing work related to different methods. Section 3 provides a detailed description of our method's implementation. In Section 4, this paper discusses experimental results and introduce the details of ablation experiments. Finally, this paper gives a conclusion in Section 5.

2. Related work

In this section, this paper reviews some of the current methods for 3D point cloud analysis, which can be broadly categorized into two types: methods based on original point clouds and methods based on graph structures.

2.1. Method based on the original point clouds

PointNet is a deep neural network that can directly process disordered point cloud data [19]. It uses multiple shared MLPs for each point for feature extraction, and then uses max-pooling to aggregate all learned point features into global features. Experiments have shown that PointNet has good computational performance and speed in 3D point cloud tasks, and it has the characteristics of light weight, high efficiency and speed, but it ignores the local characteristics of point clouds in 3D space.

Building upon PointNet, PointNet++ [21] introduces a hierarchical structure for feature extraction. It utilizes PointNet as a feature extractor and iteratively extracts features for small sets of local point clouds. This enables PointNet++ to learn more comprehensive features compared to PointNet. Despite its improved feature extraction performance in point cloud tasks, PointNet++ still processes individual points without considering context information between points, leading to high memory usage and computational burden. To address these limitations, Kd-Net [27] proposes using kd-tree indexing to

represent point cloud sets and capture potential feature relationships between local points. Although Kd-Net shows significant performance improvements, it lacks robustness against rotated point clouds.

PointCNN [22] introduces hierarchical convolution to "regularize" the disordered point cloud by learning an X-transform (a $K * K$ matrix) on the point cloud, and at the same time combines with the traditional convolution operation to construct an X-Conv block to act on the local region for feature extraction, PointCNN fully considers the local information of the point cloud. PPFNet [28] represents the point cloud's coordinates, normal, and point pair features as point sets. It encodes different point sets and inputs them into a small PointNet to extract local feature information. By generalizing contrastive loss to n-tuple loss, PPFNet maximizes the use of available correspondence relations, thus improving invariance.

SO-Net [29] simulates the spatial distribution of point clouds by building a self-organizing map (SOM). It designs a point cloud auto-encoder to hierarchically extract features for individual points and SOM nodes, and controls the degree of overlap of the sensory fields by a tunable parameter K. Larger K values result in more overlap and richer learned feature information. It improves network performance in point cloud classification tasks.

2.2. Method based on graph structures

The graph-based methods primarily treat the points in the point cloud as vertices in a graph, constructing graph data from the point cloud [30], which allows the features of point cloud to be learned directly in the spatial or spectral domain.

KCNet [31] first defines the point set kernel as a set of learnable 3D points, and then leverages the kernel correlation layer to learn the local 3D geometric structure of the point cloud by computing the affinity between neighboring point kernels within the kernel point set. The graph pooling layer utilizes the high-dimensional feature structure of local regions and performs multiple rounds of feature aggregation on the local neighborhood graph. KCNet enables learning of local geometric information while maintaining permutation invariance of 3D point clouds.

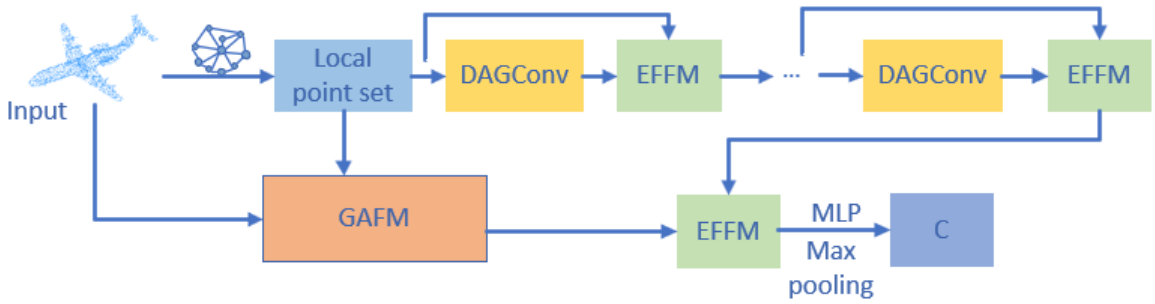
Most methods based on original points often do not consider the topological structure information of point clouds. To address this problem, DGCNN [23] designs an edge convolution operation by dynamically updating the graph structure of the point set in each convolutional layer. It performs convolution-like operations on point edges to learn local structures. Edge convolution can be embedded into existing point cloud processing frameworks. LDGCNN [32] optimizes the DGCNN network by connecting hierarchical features from different dynamic graphs. It adds shortcut connections between different edge convolution layers to learn local features, and cascading layer-wise features can alleviate the problem of gradient vanishing.

Res2Net [33] can deal with the challenge of information loss during dimension reduction by building a multi-scale CNN for point cloud classification. Chen et al. [34] propose a point neural network based on graph attention called GAPointNet, which uses the graph attention and multi-head mechanisms to capture local features. FatNet [35] stands out for its use of the same attention mechanism across two distinct forms of feature map aggregation. It also incorporates residual connections for enhanced information transfer between layers, enabling the training of deeper and more stable network architectures.

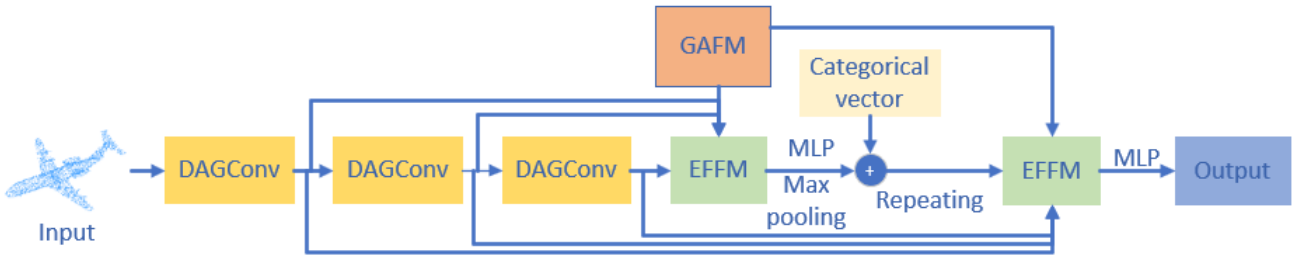
While previous studies have made significant contributions to enhancing local feature acquisition in point cloud data, they often concentrated on either global or local feature extraction without addressing

the challenge of feature loss during fusion. In this study proposed method, this study takes a comprehensive approach by simultaneously extracting global and local feature information from 3D point clouds. Furthermore, this study designs a feature fusion method based on error feedback to capture additional local features.

3. The proposed algorithm



(a) Classification model



(b) Segmentation model

Figure 1. The framework of our classification and segmentation model, where GAFM refers to global attention feature module, EFFM refers to error feedback feature fusion module, DAGConv refers to attention graph convolution and MLP refers to multi-layer perceptron.

In this section, this paper first gives the network framework and then introduce the global attentional feature module, the attention graph convolution module, and the error feedback feature fusion module. This study define the notations used in this paper as follows. The original point cloud instance is regarded as a collection of point clouds, which contains attributes such as coordinates, colors and normal vectors [36]. In this paper, the point clouds can be represented by the 3D coordinate attributes, which are described as (x_i, y_i, z_i). Given a 3D point cloud P with N points, $P = \{p_1, p_2, ..., p_N\} \in \mathbb{R}^{N\times 3}$, where the size of the point cloud is expressed as a matrix of $N \times 3$ and a single point can be noted as: $\{p_i = (x_i, y_i, z_i)|i = 1, 2, \ldots, N\}$. The framework of our classification and segmentation model is shown in Figure 1.

Figure 1(a) shows the classification model with an $n\times 3$ point cloud as the input. Firstly, the k-nearest neighbors (KNN) algorithm is used to compute the k closest neighbors for each input point. Then, four AGConv layers are employed for local feature extraction (note that: some convolution processes are

omitted). After each convolutional layer, the EFFM is applied for feature fusion. Additionally, a GAFM is utilized to learn global features of the point cloud. In the final EFFM module, local and global features are fused, generating classification scores for c categories. Figure 1(b) descripts the segmentation model, which is an extension of the classification model. Unlike the classification model, this study uses EFFM for feature fusion after three-layer convolution learning. The symbol $\oplus$ denotes feature concatenation. The segmentation model outputs classification scores for p semantic labels for each point.

For point p_i in the input 3D point cloud instance P, this study uses the KNN algorithm to select the k nearest neighbors of the point p_i, denoted by $p_{i,j} = \{p_{i,1}, p_{i,2}, ..., p_{i,k}\}$, $j \in k$, to construct a local neighborhood graph, i.e., $G(V, E)$, with as the center point of the vertex $V = \{1, 2, ..., k\} \in N$ and edge $E = \{e_{i1}, e_{i2}, ..., e_{ik}\}$. The process of constructing the neighborhood graph of KNN is shown in Figure 2.

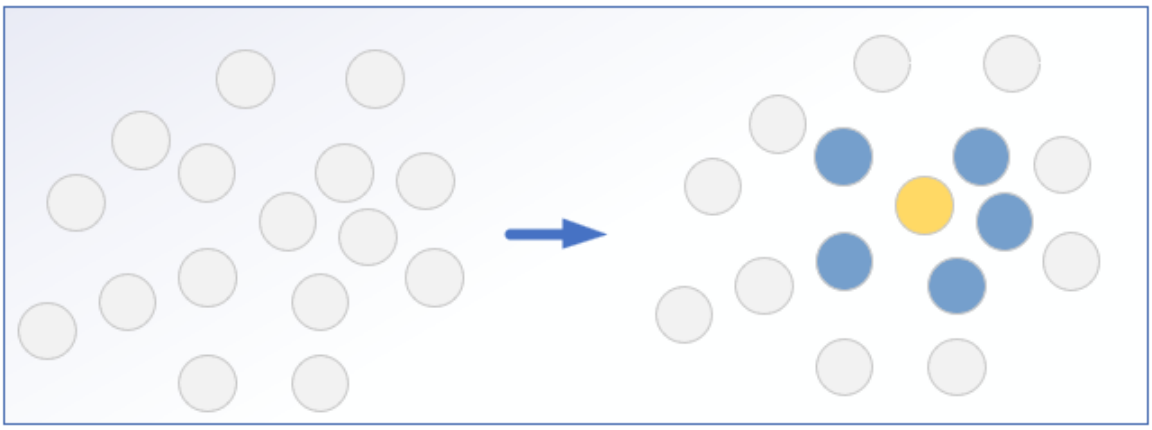


Figure 2. The process of constructing a local neighborhood graph by KNN, where the gray circles represent point cloud nodes, the yellow circle represents the central node and the blue circles around the central node represent its neighbor nodes.

Figure 2 illustrates the process of constructing local neighborhood graphs using the KNN within a point cloud instance. It calculates the Euclidean distance between each point to select its K closest neighboring points. The parameter $k = 5$, the center node p_i is marked in yellow, and the neighborhood points $p_{i,j}$ are marked in blue.

3.1. Global attention feature module

In the global attention feature module, to better aggregate effective feature information from the point cloud, this work considers the spatial relationships among all points. In the given instance, the Euclidean matrix is obtained by calculating the Euclidean distance between points along the three-dimensional spatial directions. The global attention feature module is shown in Figure 3.

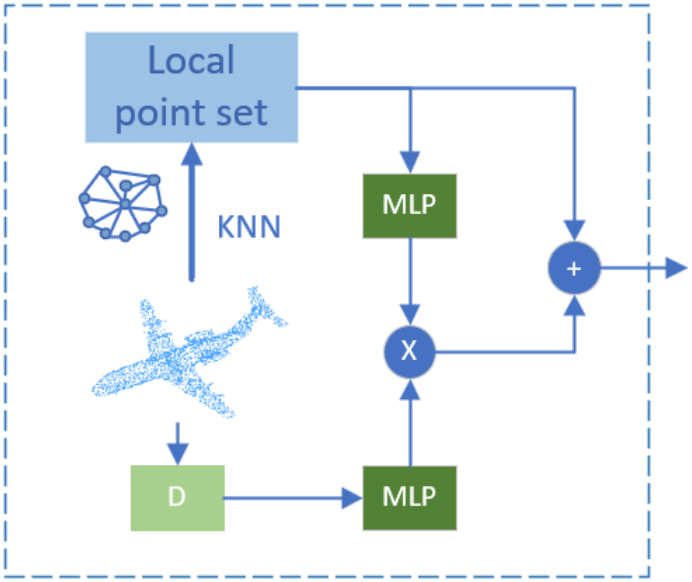


Figure 3. Global attention feature module, where KNN refers to the k-nearest neighbor algorithm, MLP refers to the multi-layer perceptron, D refers to the Euclidean distance matrix.

This work calculates the Euclidean distance between points along each coordinate direction in the point cloud, and then use $softmax$ to perform normalization to obtain the global normalized distance matrix. For a point cloud with N points, this work uses Eq (3.1) to calculate the Euclidean distance matrix of the points p_i and p_j in each coordinate direction:

$$D = \sum_{i=1}^{N}\sum_{j=1}^{N}[p_{x_i} - p_{x_j}, p_{y_i} - p_{y_j}, p_{z_i} - p_{z_j}]^T, \tag{3.1}$$

where p_{x_i} is the coordinate of the point p_i in the x-axis direction, p_{y_i} is the coordinate of the point p_i in the y-axis direction, and p_{z_i} is the coordinate of the point p_i in the z-axis direction.

This work uses the $softmax$ function to normalize the distance matrix calculated in Eq (3.1), and the normalized distance matrix is obtained according to Eq (3.2).

$$D'_{i,j} = softmax(D_{ij}) = \frac{exp(D_{ij})}{\sum_{j=1}^{N} exp(D_{ij})}, \tag{3.2}$$

where $D_{i,j}$ is the distance between the points p_i and p_j, and D'_{ij} is the normalized distance between p_i and p_j.

This work utilizes an MLP as a non-linear transformation function to compute the global features, as described in Eq (3.3).

$$F_{g_{ij}} = MLP(D'_{ij}) = \sum_{j=1}^{N} W_{g_{ij}} * D'_{ij} + b_i, \tag{3.3}$$

where $F_{g_{ij}}$ is the global attention weight of points p_i and p_j, $W_{g_{ij}}$ is the weight matrix and b_i is the bias.

This work uses an MLP to learn and map the local point set L_p to obtain $\widetilde{L_p}$, then matrix multiply the mapping result $\widetilde{L_p}$ with the global attention weights, and finally perform a concatenation operation with L_p to obtain the global features, which are defined by Eq (3.4).

$$F_g = (\widetilde{L_p} \otimes F_{g_{ij}}) \oplus L_p, \tag{3.4}$$

where F_g are the global features.

3.2. *Attention graph convolution module*

Our attention graph convolution module consists of multiple attention graph convolution layers (AGConv). The AGConv layer is responsible for extracting local features from the point cloud while considering both the central point p_i and its k neighboring points' features during the aggregation process. In the graph convolution, this work introduces an attention mechanism to obtain attention coefficients for each neighborhood point with respect to the central point. This allows us to learn more valuable information about local features. The process of the attention graph convolution operation is illustrated in Figure 4.

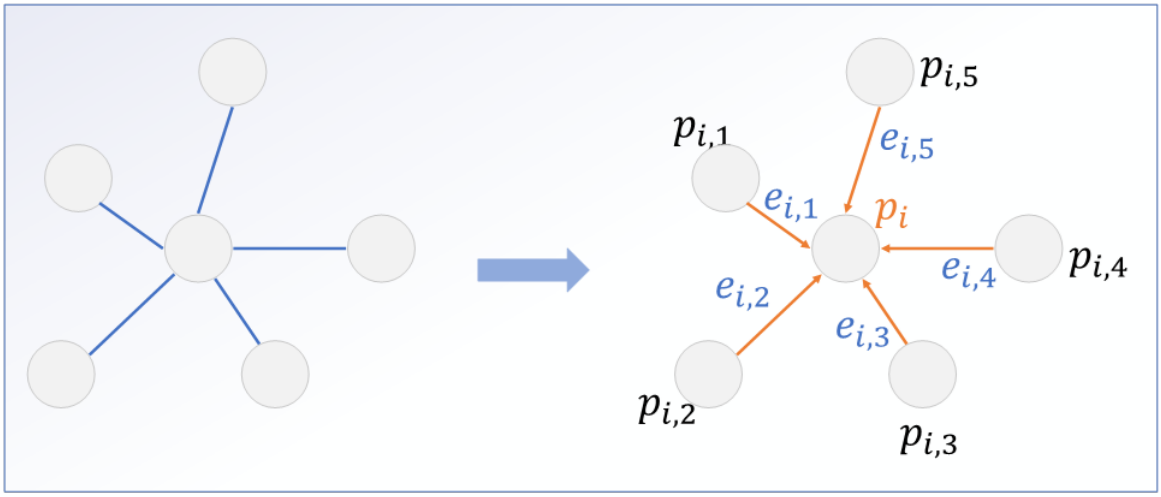


Figure 4. The attention graph convolution operation, where the gray circles represent point cloud nodes, the yellow directed arrows represent the neighbor relationship between the central node p_i and its 5 (assume $k = 5$) neighbor nodes p_{ij} ($j = 1...5$).

According to the attention mechanism, this work needs to convert neighborhood points $p_{i,j}$ and edges into the high-dimensional feature space to obtain enhanced representational capacity. This work uses a non-linear function $g(.)$ to represent the neighborhood relationship of the central node p_i, which is defined by Eq (3.5).

$$e_{i,j} = g(p_{i,j}, \theta), \tag{3.5}$$

where $e_{i,j}$ is the neighborhood relationship of the central node p_i, $p_{i,j}$ is the neighborhood p_j of the point p_i, θ is a set of learnable parameters of the filter, and $g(.)$ is a single-layer neural network.

According to the neighborhood relationship calculated in Eq (3.5), this work uses a nonlinear activation function to obtain the attention coefficient of the relationship $e_{i,j}$ between the center node p_i and the neighborhood node $p_{i,j}$, which is defined by Eq (3.6).

$$\alpha_{ij} = \sigma(e_{i,j}) = \sigma(g(e_{i,j}, \theta)) = LeakyReLU(g(e_{i,j}, \theta)), \tag{3.6}$$

where α_{ij} is the attention coefficient of p_i and $p_{i,j}$, and $\sigma(.)$ is a non-linear activation function. Here this work chooses LeakyReLU as the activation function.

According to the attention mechanism, the attention coefficients assigned by neighbors at different points need to be aligned. This work uses $softmax$ to normalize the attention coefficients α_{ij} of different $e_{i,j}$ to obtain the final attention coefficient, which is defined by Eq (3.7).

$$\alpha_{ij}^{'} = softmax(\alpha_{ij}) = \frac{exp(\alpha_{ij})}{\sum_{k \in N} exp(\alpha_{ij})}, \tag{3.7}$$

where $\alpha_{ij}^{'}$ is the attention coefficient after normalization.

This uses the normalization coefficient to calculate the contextual features of each point, which are defined by Eq (3.8).

$$\widehat{F}_i = \sigma(\sum_{j \in N} \alpha_{ij}^{'} e_{ij}), \tag{3.8}$$

where $\widehat{F}_i$ is the contextual features of point p_i and $\sigma(.)$ is the non-linear activation function. This work chooses LeakyReLU as the activation function.

3.3. *Feature fusion module*

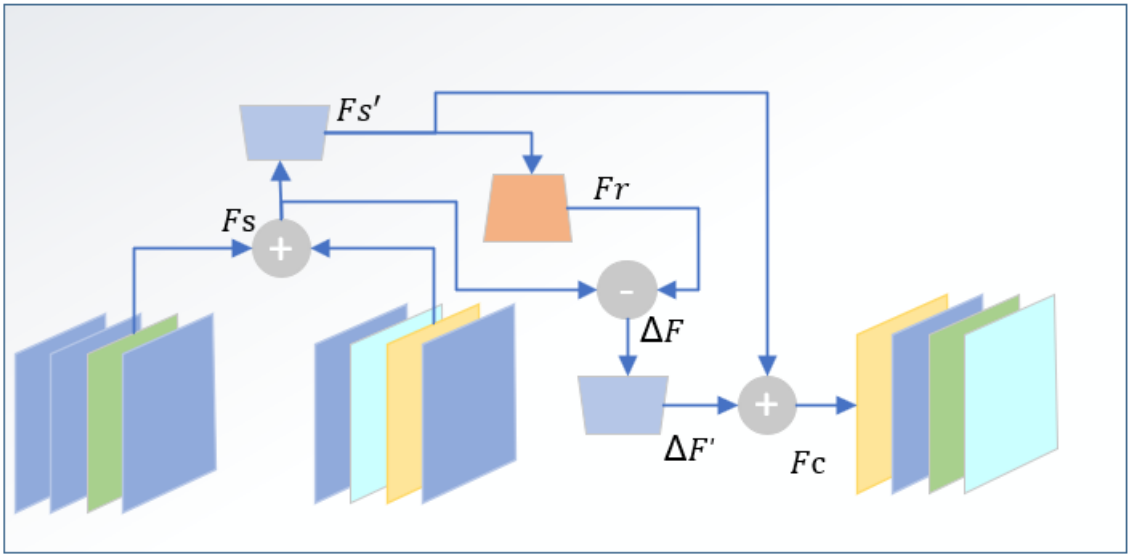


Figure 5. The feature fusion module. The blue trapezoid represents the projection operation, the orange trapezoid represents reverse-projection operation and the overlapping quadrangles of different colors represent features that need to be fused.

In previous research on point cloud classification, the common approach was to concatenate the extracted local features to form the global feature representation of the point cloud. However, simple feature concatenation may lead to the loss of some important information, resulting in classification performance deviating from the expected results. To address this issue, this work proposes a feature fusion module called the error feedback feature fusion module (EFFM) based on the error feedback mechanism [37]. The EFFM module is designed to better learn the contextual relationships among points in high-dimensional space. The feature fusion module is illustrated in Figure 5.

In the feature fusion module, this work first cascades the local features extracted by the two AGConv to obtain the initial fusion features. This process is defined by Eqs (3.9) and (3.10).

$$F_s = concat\{F_{l-1}, F_l\}, \tag{3.9}$$

$$F_s = concat\{F_1, ..., F_l\}, \tag{3.10}$$

where F_s is the initial fused feature, F_l is the graph feature extracted in the l-th layer and F_{l-1} is the input feature from the previous layer. Equation (3.10) describes the EFFM process of fusion of multiple extracted features in a segmentation model.

Next, this work maps F_s to the high-dimensional feature space to obtain the geometric relationship of the local region in the high-dimensional space, and restore the initial fusion feature of the input by using reverse-projection or projection, which aims to map the feature back to the original feature space. After obtaining the reconstructed feature, this work computes the difference between the original fused feature and the reconstructed feature. This difference serves as the error signal, that is, the information lost during the initial fusion process. EFFM is used to constrain the feature learning process and minimize the deviation of the output feature as the training progresses. This process is defined by Eq (3.11).

$$\begin{aligned} F_s^{'} &= f_a(F_s), \\ F_r &= f_b(F_s^{'}), \\ \triangle F &= F_s - F_r, \end{aligned} \tag{3.11}$$

where $F_s^{'}$ is the feature of F_s after the projection operation, $f_a(.)$ is the projection operation, $f_b(.)$ is the reverse-projection, F_r is the reconstructed feature and $\triangle F$ is the difference between the original fusion feature and the reconstructed feature.

Finally, a projection operation is performed on the error signal, and a cascade operation is performed with the projected $F_s^{'}$ to obtain the final fusion feature F_c, which is defined by Eq (3.12).

$$F_c = F_s^{'} + f_a(\triangle F), \tag{3.12}$$

Please note that in the feature fusion module EFFM, this work sets the size of the point cloud to be constant.

In the last layer of local features extracted by AGConv via EFFM, this work needs to perform the final fusion of global features extracted by GAFM as the final point cloud features and perform our classification task, which is defined by Eq (3.13).

$$F_{all} = EFFC(F_c, F_g), \tag{3.13}$$

where F_{all} is the final feature of the point cloud.

4. Experimental results

In order to verify the performance of the proposed model, this study compares it with several state-of-the-art methods for point cloud classification and segmentation tasks. Extensive experiments are conducted on the two main datasets, ModelNet40 and ShapeNet [38]. The ModelNet40 dataset is well-structured and commonly used for point cloud classification tasks. On the other hand, the ShapeNet dataset contains point cloud data with complex backgrounds and missing points, making it suitable for part segmentation tasks in more challenging and complex scenes. Finally, to validate the effectiveness of each module in our proposed approach, this work also conducted ablation experiments accordingly. All experiments are conducted on an NVIDIA GTX 3060 GPU.

4.1. Classification

4.1.1. Datasets

The experiments of point cloud shape classification are performed on the dataset ModelNet40, which contains 40 object categories such as airplanes, chairs and cars, consisting of 9843 samples in the training set and 2468 samples in the test set. During the experiment, this represents individual point clouds using their 3-dimensional coordinates (x, y, z), and sample 1024 points from the surface of each object model to create the input data.

4.1.2. Details of method implementation

In our classification model, this work employs four AGConv layers to extract local features from the point cloud. The output of AGConv1 is used as the input of AGConv2, the output feature F_1 of AGConv1 and the output feature F_2 of AGConv2 are combined in the EFFM module for feedback feature fusion to obtain the fusion feature F_{12}. Next, F_{12} is used as the input of AGConv3, then this process is repeated until the final fused local feature F_c is obtained.

During the local feature learning process, this work utilizes the KNN algorithm to construct local neighborhood graphs for the input unordered point cloud. Experiments are conducted with different values of k to evaluate its impact on the classification results. Experimental results imply that setting k to 20 for the number of neighboring points around the center point yields favorable results.

The input consists of a set of 1024 points in three-dimensional space, which are transformed into a 32-dimensional embedding space. Multiple layers of AGConv are used for high-dimensional feature learning, and in the final EFFM module, the high-dimensional features learned are fused with the point cloud features extracted by the global attention feature module (GAFM). Then, a max pooling operation is performed to obtain the global features, which are then fed into an MLP layer for classification.

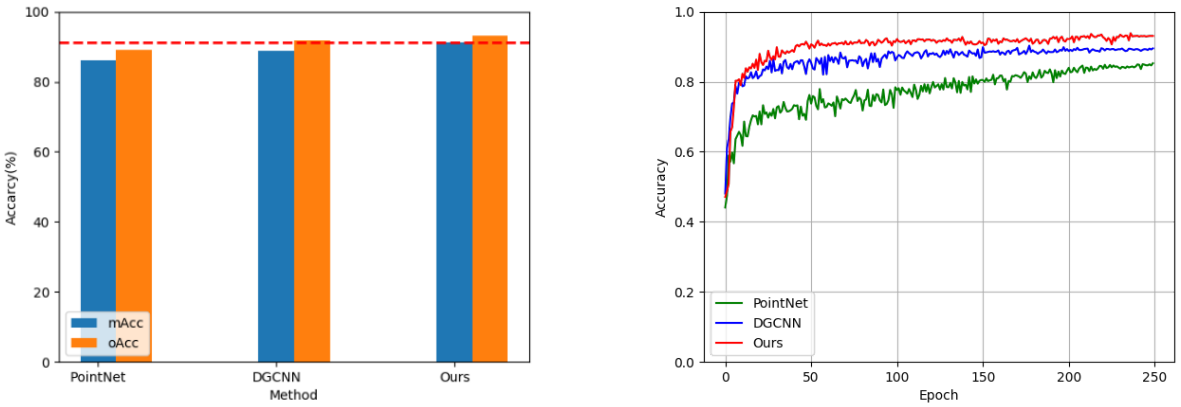
In the classification experiments, the initial learning rate is set to 0.001, the batch size is set to 32, and each training epoch is set to 250. The Adam optimizer is used for network training.

4.1.3. Experimental results

In point cloud classification experiments on the ModelNet40 dataset, our network model achieves superior performance compared to other methods under the same conditions, as shown in Table 1. This work uses the 1024 points represented by three-dimensional coordinates (x, y, z) as input. Extensive experimental results demonstrate that our overall accuracy of classification reaches 93.1%, indicating the effectiveness of our proposed model.

Table 1. Comparison of our model and other models.

Method	Points	Mean class accuracy (%)	overall class accuarcy (%)
kd-Net [31]	1024	86.3	90.6
PointNet [19]	1024	86.0	89.2
PointCNN [22]	1024	88.1	92.2
PointConv [39]	1024	-	92.5
KC-Net [31]	1024	-	91.0
PointNet++ [21]	5000	-	91.9
So-Net [29]	5000 + normal	90.8	90.4
SpiderCNN [40]	1024 + normal	-	92.4
DGCNN [23]	1024	88.9	91.7
Ours	1024	**91.2**	**93.1**



(a) (b)

Figure 6. Experiment results of classification. Figure 6(a). The mAcc/oAcc comparison histogram of PointNet, DGCNN and our model. Figure 6(b). The epoch and accuracy processes of PointNet, DGCNN and our model, where PointNet is marked in green, DGCNN is marked in blue, and Our model is marked in red.

To provide a comprehensive visual representation of our experimental results, this work plots a bar chart in Figure 6(a) to compare the classification performance of PointNet, DGCNN and our proposed method. Additionally, this work includes a red dashed line as a reference, which represents our method's mAcc (mean accuracy) as the baseline for comparison.

In Figure 6(b), this work demonstrates the training process effects of PointNet, DGCNN and our method. This work tracks the classification accuracy of PointNet, DGCNN and our method on the ModelNet40 dataset throughout 250 epochs, and the result shows that our accuracy reaches a relatively stable state within the first 50 epochs. Moreover, not only does our method achieve higher accuracy compared to the other models, but the overall fluctuation is also minimal, indicating a consistently good performance and a strong classification effect.

To further validate the rationality of our method, this work will conduct additional experiments by varying some crucial parameters that could potentially influence the experimental results.

This work constructs the local neighborhood graph of the point cloud using the KNN algorithm, where the value of k determines the construction of the neighborhood graph and affects the acquisition of local features. To evaluate the optimal k value within a certain range, while keeping other parameters constant, this work conducts experiments by varying the k value. The experimental results are presented in Table 2. Additionally, to provide a more intuitive understanding of the impact of different k values on our results, this work compares the classification accuracy with DGCNN in Figure 7. The experiments demonstrate that our method achieves better results when $k = 20$, and in general, our method outperforms DGCNN when $k \in [15, 30]$.

Table 2. Classification results for different k values.

k	oAcc (%)
$k = 5$	87.92
$k = 10$	89.49
$k = 15$	91.74
$k = 20$	**93.15**
$k = 25$	92.16
$k = 30$	90.58

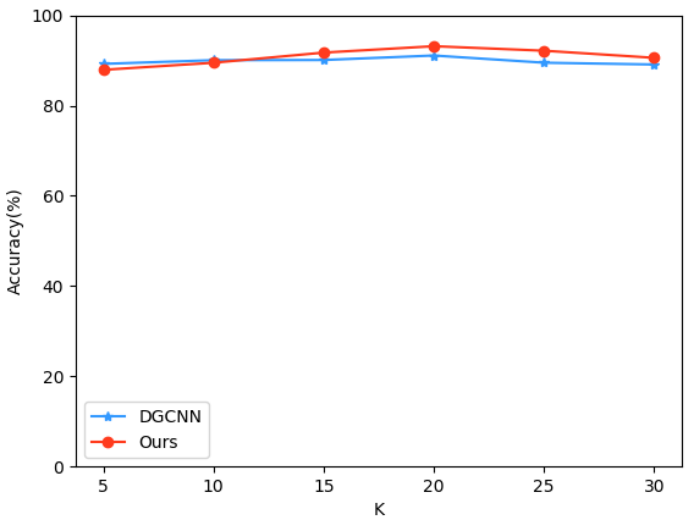


Figure 7. Comparison of classification effects with DGCNN under different k values.

In the experiment of changing the k value, this work finds that the model achieves the best results when $k = 20$. Therefore, in the next experiments, k is set to 20 by default. In addition to the influence of the k value on feature extraction, the input number of points will also affect the result of feature extraction. Under the condition of $k = 20$ and the other parameters remaining unchanged, this work varies the input number of points from {256,512,768,1024}, and the classification accuracy is shown in Figure 8.

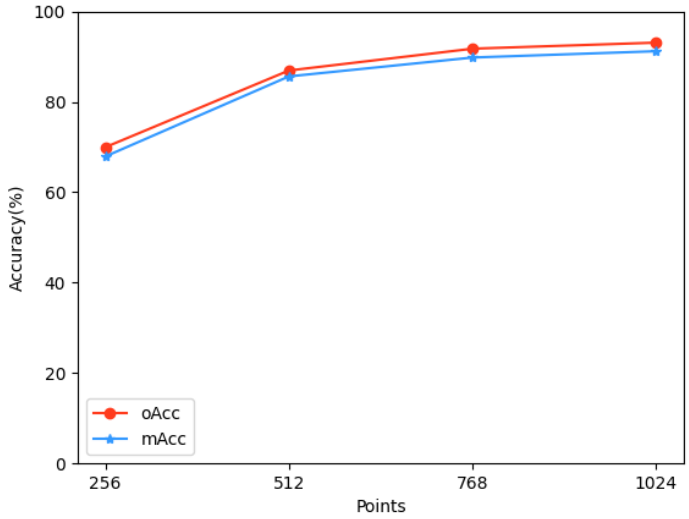


Figure 8. Classification accuracy under different numbers of points.

4.2. Part segmentation

4.2.1. Datasets

This work evaluated the performance of our model on the ShapeNet dataset [38] for point cloud part segmentation. The ShapeNet dataset, as a popular benchmark for point cloud semantic segmentation, consists of 16 major categories and is annotated with 50 parts, totaling 16,881 models, which are divided into training and testing sets. Unlike the classification experiments, in the ShapeNet dataset, each model samples 2048 points as the input.

4.2.2. Experimental results

In the segmentation model, this work employs three AGConv layers for feature extraction. Unlike the classification model, after the three AGConv layers, this work performs an additional feature fusion step. This fusion process includes features extracted by the three AGConv layers and the GAFM module. Except for the number of sampled points, the other parameters, such as batch size and activation functions, remain the same as in the classification model.

Similar to DGCNN, this work uses mIoU to evaluate the performance of our network for point cloud part segmentation. To better understand the part segmentation performance of different methods, this work categorizes the other compared methods into two types: CNNs and GNNs. The experimental results shown in Table 3 demonstrate that compared to five other state-of-the-art models [19, 21, 23,

41,42], our model achieves a higher mIoU value and shows more competitive results on the ShapeNet dataset. Figure 9 presents some part segmentation results, where black circles are used to highlight areas containing obvious segmentation errors.

Table 3. Part segmentation results.

Categories	Method	mIoU	Air plane	Bag	Cap	Car	Chair	Ear phone	Guitar	Knife	Lamp	Laptop	Motor bike	Mug	Pistol	rocket	Skate board	Table
CNNS	PointNet	83.7	83.4	78.7	82.5	74.9	89.6	73.0	**91.5**	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
	Pointnet++	85.1	82.4	79.0	87.7	77.3	**90.8**	71.8	91.0	85.9	83.7	95.3	**71.6**	94.1	**81.3**	58.7	**76.4**	82.6
	DGCNN	85.2	**84.0**	83.4	86.7	77.8	90.6	**74.7**	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
GNNS	3D-GCN	85.1	83.1	84.0	86.6	77.5	90.3	74.1	90.9	86.4	**83.8**	95.6	66.8	94.8	**81.3**	59.6	75.7	**82.8**
	RGCNN	84.3	80.2	82.8	**92.6**	75.3	89.2	73.7	91.3	**88.4**	83.3	**96.0**	63.9	**95.7**	60.9	44.6	72.9	80.4
	Ours	**85.4**	83.1	**86.6**	86.5	**80.3**	90.4	72.9	91.2	86.0	81.7	95.4	67.2	94.3	75.4	**65.2**	74.9	**82.8**

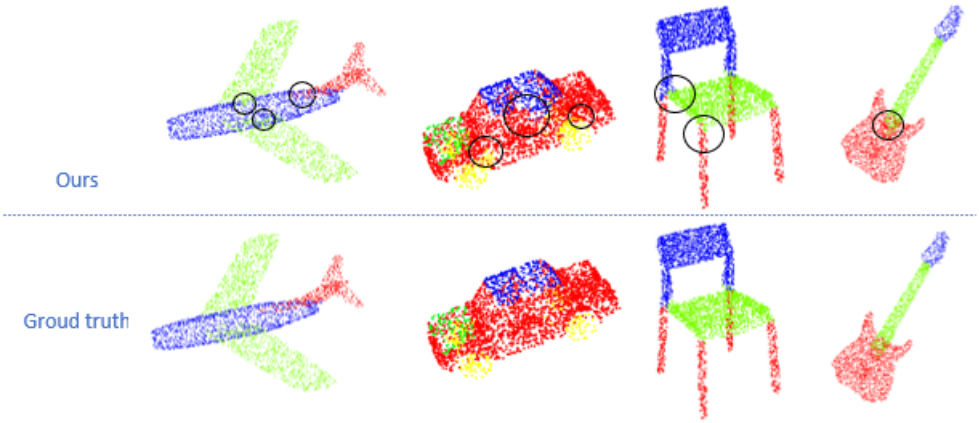


Figure 9. Part segmentation effect diagram.

4.3. Ablation experiment

In order to better assess the feasibility of our approach, this work conducts an ablation experiment on the ModelNet40 benchmark to analyze the effectiveness of each individual module in the 3D point cloud classification task. This work samples 1024 points from each point cloud instance as the input. Please note that, this work uses the optimal parameters determined from the classification experiments to evaluate the impact of different modules on classification accuracy. The experimental results presented in Table 4 provide insights into how each module affects the overall classification performance.

Table 4. Classification accuracy under different module composition.

AGConv	GAFM	EFFM	oAcc (%)
√			90.2
√	√		90.7
√		√	92.4
√	√	√	**93.1**

4.4. Model size and timing

In this section, this work assesses the complexity of our model by analyzing parameters, FLOPs (floating-point operations), and execution time. To ensure fair and comparable experiments, all evaluations were conducted on the same hardware configuration, which consisted of an NVIDIA GTX 3060 GPU and an Intel i5-12400 CPU with 6 cores running at 2.50 GHz. During the experiments, this work utilized 1024 points as input, with a batch size set at 32 and an initial learning rate of 0.001. Based on prior experimentation, this work determined that an optimal value for k is 20, which is used to construct the local neighborhood graph. Our model, primarily focused on global feature extraction from point clouds, possesses a slightly higher number of model parameters compared to PointNet++ and DGCNN. However, in the broader context, our model offers several advantages. For a detailed comparison of model complexity, including parameters, FLOPs and execution time, please refer to Table 5. In the table, 'M' denotes 'million', 'MS' signifies 'milliseconds' and 'B' stands for 'billion'.

Table 5. Complexity analysis of our classification method.

Method	Params (M)	Time (MS)	FLOPS (B)	Accuracy (%)
Pointnet	3.48	16.6	14.7	89.2
PointNet++	1.48	163.2	26.94	91.9
DGCNN	1.84	27.2	42.71	92.2
Ours	1.97	26.0	24.4	93.1

5. Conclusions

This work proposed a novel AGConv, which incorporates attention mechanisms into graph convolutional layers for learning local features of points. Additionally, this work introduced a GAFM module to capture global features of the point cloud, effectively considering both local and global information. Unlike previous methods that used concatenation to fuse features, which could lead to feature loss, this work introduced an EFFM module that learns an error signal to mitigate this issue. Extensive experiments demonstrated the effectiveness of our proposed method in learning both local and global features of point clouds. Compared to state-of-the-art methods, it effectively reduces feature loss during the fusion process and exhibits excellent performance in point cloud classification and segmentation tasks. Furthermore, our method shows fast convergence during training. The method proposed in this paper has been experimentally compared and can improve the classification and segmentation accuracy of point clouds to some extent. When creating a local neighborhood graph of a point cloud, this work experiments repeatedly and select the graph build scale with the best experimental results. This work only use the coordinates of the point cloud as input, ignoring the effects of other features of the point cloud on the experimental results. In future work, we will focus on the study of adaptive graph building methods and consider applying our methods to the classification and segmentation of large point clouds.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant No.62362016, the Fundation on Enhancement of Basic Research Ability of Young and Middle-aged Teachers in Guangxi Universities and Colleges under Grant No.2022KY0788 and the Natural Science Foundation of Guilin University of Aerospace Technology under Grant No.XJ21KT21.

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, M. Bennamoun, Deep learning for 3D point clouds: A survey, *IEEE Trans. Pattern Anal. Mach. Intell.*, **42** (2021), 4338–4364. https://doi.org/10.1109/TPAMI.2020.3005434
2. X. Yuan, J. Shi, L. Gu, A review of deep learning methods for semantic segmentation of remote sensing imagery, *Expert Syst. Appl.*, **169** (2021), 114417. https://doi.org/10.1016/j.eswa.2020.114417
3. H. Aasen, E. Honkavaara, A. Lucieer, P. J. Zarco-Tejada, Quantitative remote sensing at ultra-high resolution with UAV spectroscopy: A review of sensor technology, measurement procedures, and data correction workflows, *Remote Sens.*, **10** (2018), 1091. https://doi.org/10.3390/rs10071091
4. J. Balado, J. Martínez-Sánchez, P. Arias, A. Novo, Road environment semantic segmentation with deep learning from MLS point cloud data, *Sensors*, **19** (2019), 3466. https://doi.org/10.3390/s19163466
5. R. Meleppat, K. E. Ronning, S. J. Karlen, M. E. Burns, E. N. Pugh Jr, R. J. Zawadzki, *In vivo* multimodal retinal imaging of disease-related pigmentary changes in retinal pigment epithelium, *Sci. Rep.*, **11** (2015), 16252. https://doi.org/10.1038/s41598-021-95320-z
6. R. K. Meleppat, M. V. Matham, L. K. Seah, C. Shearwood, Quantification of biofilm thickness using a swept source based optical coherence tomography system, in *International Conference on Optical and Photonic Engineering*, **9524** (2015), 683–688. https://doi.org/10.1117/12.2190106
7. R. K. Meleppat, E. B. Miller, S. K. Manna, P. Zhang, E. N. Pugh Jr, R. J. Zawadzki, Multiscale Hessian filtering for enhancement of OCT angiography images, in *Ophthalmic Technologies XXIX*, **10858** (2019), 64–70. https://doi.org/10.1117/12.2511044
8. K. M. Ratheesh, L. K. Seah, V. M. Murukeshan, Spectral phase-based automatic calibration scheme for swept source-based optical coherence tomography systems, *Phys. Med. Biol.*, **61** (2016), 7652. https://doi.org/10.1088/0031-9155/61/21/7652
9. H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3D shape recognition, in *Proceedings of the IEEE International Conference on Computer Vision*, (2015), 95242L. https://doi.org/10.1109/ICCV.2015.114

10. M. Huang, P. Wei, X. Liu, An efficient encoding voxel-based segmentation (EVBS) algorithm based on fast adjacent voxel search for point cloud plane segmentation, *Remote Sens.*, **11** (2019), 2727. https://doi.org/10.3390/rs11232727
11. B. Xiong, W. Jiang, D. Li, M. Qi, Voxel grid-based fast registration of terrestrial point cloud, *Remote Sens.*, **13** (2021), 1905. https://doi.org/10.3390/rs13101905
12. C. Wen, X. Li, X. Yao, L. Peng, T. Chi, Airborne LiDAR point cloud classification with global-local graph attention convolution neural network, *ISPRS J. Photogramm. Remote Sens.*, **173** (2021), 181–194. https://doi.org/10.1016/j.isprsjprs.2021.01.007
13. S. A. Bello, S. Yu, C. Wang, J. M. Adam, J. Li, Review: Deep learning on 3D point clouds, *Remote Sens.*, **12** (2020), 1721. https://doi.org/10.3390/rs12111729
14. Z. Zhang, L. Zhang, X. Tong, B. Guo, L. Zhang, X. Xing, Discriminative-Dictionary-Learning-Based multilevel point-cluster features for ALS point-cloud classification, *IEEE Trans. Geosci. Remote Sens.*, **54** (2016), 7309–7322. https://doi.org/10.1109/TGRS.2016.2599163
15. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2016), 770–778. https://doi.org/10.1109/CVPR.2016.90
16. D. Maturana, S. Scherer, VoxNet: A 3D convolutional neural network for real-time object recognition, in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (2015), 922–928. https://doi.org/10.1109/IROS.2015.7353481
17. C. R. Qi, H. Su, M. Niebner, A. Dai, M. Yan, L. J. Guibas, Volumetric and multi-view CNNs for object classification on 3D data, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2016), 5648–5656. https://doi.org/10.1109/CVPR.2016.609
18. N. Qin, X. Hu, P. Wang, J. Shan, Y. Li, Semantic labeling of ALS point cloud via learning voxel and pixel representations, *IEEE Geosci. Remote Sens. Lett.*, **17** (2020), 859–863. https://doi.org/10.1109/LGRS.2019.2931119
19. C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3D classification and segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2017), 77–85. https://doi.org/10.1109/CVPR.2017.16
20. B. S. Hua, M. K. Tran, S. K. Yeung, Pointwise convolutional neural networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2018), 984–993. https://doi.org/10.48550/10.1109/CVPR.2018.00109
21. C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in *Advances in Neural Information Processing Systems*, **30** (2017), 5099–5108.
22. Y. Li, R. Bu, M. Sun, W. Wu, X. Di, B. Chen, PointCNN: Convolution on X-transformed points, in *Advances in Neural Information Processing Systems*, **31** (2018), 828–838.
23. Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, J. M. Solomon, Dynamic graph CNN for learning on point clouds, *ACM Trans. Graphics*, **38** (2019), 1–12. https://doi.org/10.1145/3326362
24. Y. Li, Q. Lin, Z. Zhang, L. Zhang, D. Chen, F. Shuang, MFNet: Multi-level feature extraction and fusion network for large-scale point cloud classification, *Remote Sens.*, **14** (2022), 5707. https://doi.org/10.3390/rs14225707

25. G. Wang, Q. Zhai, H. Liu, Cross self-attention network for 3D point cloud, *Knowledge-Based Syst.*, **2022** (2022), 247. https://doi.org/10.1016/j.knosys.2022.108769

26. X. Li, L. Wang, J. Lu, Multiscale receptive fields graph attention network for point cloud classification, *Complexity*, **2021** (2021), 1–9. https://doi.org/10.1155/2021/8832081

27. R. Klokov, V. Lempitsky, Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models, in *Proceedings of the IEEE International Conference on Computer Vision*, (2017), 863–872. https://doi.org/10.1109/ICCV.2017.99

28. H. Deng, T. Birdal, S. Ilic, PPFNet: Global context aware local features for robust 3D point matching, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2018), 195–205. https://doi.org/10.1109/CVPR.2018.00028

29. J. Li, B. M. Chen, G. H. Lee, SO-Net: Self-organizing network for point cloud analysis, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2018), 9397–9406. https://doi.org/10.1109/CVPR.2018.00979

30. C. Q. Huang, F. Jiang, Q. H. Huang, X. Z. Wang, Z. M. Han, W. Y. Huang, Dual-graph attention convolution network for 3-D point cloud classification, *IEEE Trans. Neural Networks Learn. Syst.*, **2022** (2022), 1–13. https://doi.org/10.1109/TNNLS.2022.3162301

31. Y. Shen, C. Feng, Y. Yang, D. Tian, Mining point cloud local structures by kernel correlation and graph pooling, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2018), 4548–4557. https://doi.org/10.1109/CVPR.2018.00478

32. K. Zhang, M. Hao, J. Wang, C. W. de Silva, C. Fu, Linked dynamic graph CNN: Learning on point cloud via linking hierarchical features, in *2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, (2021), 7–12. https://doi.org/10.1109/M2VIP49856.2021.9665104

33. W. Wang, T. Wang, Y. Cai, Multi-view attention-convolution pooling network for 3D point cloud classification, *Appl. Intell.*, **52** (2022), 14787–14798. https://doi.org/10.1007/s10489-021-02840-2

34. C. Chen, L. Z. Fragonara, A. Tsourdos, GAPointNet: Graph attention based point neural network for exploiting local feature of point cloud, *Neurocomputing*, **438** (2022), 122–132. https://doi.org/10.1016/j.neucom.2021.01.095

35. H. Wu, S. Chen, G. Chen, W. Wang, B. Lei, Z. Wen, FAT-Net: Feature adaptive transformers for automated skin lesion segmentation, *Med. Image Anal.*, **76** (2022), 102327. https://doi.org/10.1016/j.media.2021.102327

36. Z. Xie, J. Chen, B. Peng, Point clouds learning with attention-based graph convolution networks, *Neurocomputing*, **402** (2020), 245–255. https://doi.org/10.1016/j.neucom.2020.03.086

37. S. Qiu, S. Anwar, N. Barnes, Geometric back-projection network for point cloud classification, *IEEE Trans. Multimedia*, **24** (2022), 1943–1955. https://doi.org/10.1109/TMM.2021.3074240

38. Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, et al., 3D shapeNets: A deep representation for volumetric shapes, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (2015), 1912–1920. https://doi.org/10.1109/CVPR.2015.7298801

39. W. Wu, Z. Qi, F. Li, PointConv: Deep convolutional networks on 3D point clouds, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2019), 9621–9630. https://doi.org/10.1109/CVPR.2019.00985

40. Y. Xu, T. Fan, M. Xu, L. Zeng, Y. Qiao, SpiderCNN: Deep learning on point sets with parameterized convolutional filters, in *Proceedings of the European Conference on Computer Vision (ECCV)*, (2018), 87–102.

41. Z. H. Lin, S. Y. Huang, Y. C. F. Wang, Convolution in the cloud: Learning deformable Kernels in 3D graph convolution networks for point cloud analysis, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2020), 1800–1809. https://doi.org/10.1109/CVPR42600.2020.00187

42. G. Te, W. Hu, A. Zheng, Z. Guo, RGCNN: Regularized graph CNN for point cloud segmentation, in *Proceedings of the 26th ACM International Conference on Multimedia*, (2018), 746–754. https://doi.org/10.1145/3240508.3240621

AIMS Press