A Novel GCN-based Point Cloud Classification Model Robust to Pose Variances

Huafeng Wang[a,b,*], Yaming Zhang[a], Wanquan Liu[c], Xianfeng Gu[d], Xin Jing[a], Zicheng Liu[b]

[a]*School of Information Technology, North China University of Technology, Beijing, China*
[b]*School of Software, Beihang University, Beijing, China*
[c]*Department of Computing, Curtin University, WA, 6102, Australia*
[d]*Department of Computer Science, Stony Brook, NY 11794-2424, USA*

Abstract

Point cloud data can be produced by many depth sensors, such as Light Detection and Ranging (LIDAR) and RGB-D cameras, and they are widely used in broad applications of robotic navigation and remote-sensing for the understanding of environment. Hence, new techniques for object representation and classification based on 3D point cloud are becoming increasingly in high demand. Due to the irregularity of the object shape, the point cloud-based object recognition is a very challenging task, especially the pose variances of a point cloud will impose many difficulties. In this paper, we tackle the challenge of pose variances in object classification based on point cloud by developing a novel end-to-end pose robust graph convolutional network. Technically, we first represent the point cloud using the spherical system instead of the traditional Cartesian system for simplicity of computation and representation. Then a pose auxiliary network is constructed with an aim to estimate the pose changes in terms of rotation angles. Finally, a graph convolutional network is constructed for object classification against the pose variations of point cloud. The experimental results show the new model outperforms the existing approaches (such as PointNet and PointNet++) on the classification task when conducting ex-

*Corresponding author Email-Address, wanghuafengbuaa@gmail.com
Email addresses: wanghuafengbuaa@gmail.com (Huafeng Wang), yaming@ncut.edu.cn (Yaming Zhang), W.Liu@curtin.edu.au (Wanquan Liu), gu@cs.stonybrook.edu (Xianfeng Gu), Liuzicheng@buaa.edu.cn (Zicheng Liu), xin2856jing@126.com (Xin Jing)

periments on the ModelNet40 dataset with a series of random rotations of a 3D point cloud. Specifically, we obtain 73.02% accuracy for classification task on the ModelNet40 with delaunay triangulation algorithm, which is much better than the current state of the art algorithms, such as Pointnet and PointCNN.
Keywords: `Point Cloud`, Pose robust, Graph Convolutional Network, Classification

1. Introduction

Since point cloud can provide detailed information for representation, one can extract more features from point cloud in comparison to 2D images for object classification [1]. Therefore, point cloud is widely used in various applications such as digital preservation, reverse engineering, surveying, architecture, 3D gaming, robotics, and virtual reality. With the rapid development of artificial intelligence, it would be of great importance to develop new techniques for object classification that can automatically process the spatial point cloud information. In response to this trend, the study of processing point cloud data is also booming. Specifically, due to the particularity of point clouds, most previous studies have focused on 3D feature representation, high descriptiveness, and efficient computation [2]. Until recent years, the research on the point cloud based object recognition using the deep learning techniques is attracting more and more researchers. Although deep learning methods have achieved better results in some specific object detection, semantic segmentation and recognition tasks based on point cloud data than traditional research [3], there are still many challenges in this area.

First, the convolution operational structure for 2D ordered images are not applicable to the unordered three-dimensional point cloud. In 2D image, the convolutional operation is conducted on 2D image in a sequential order and this can capture the shape representation effectively. However, one same point cloud can be represented by two completely different matrices in a point cloud library (PCL) [4], and this brings huge difficulty in processing point cloud

data. Furthermore, the traditional convolution structure strongly requires a highly ordered data as the input format, but the point cloud data is generally disordered in practice.

Second, the same point cloud with different rotations may have different representations. In general, the rigid change (rotation or translation) [5] of a given point cloud should not affect its representation of the overall shape in the space. That is, a given point cloud data can be multiplied by different rotation translation matrices, but eventually the shape they represent remains the same. Therefore, the designed network is required to be robust to pose variances brought by the point cloud rotation.

Third, as clustering methods (some commonly used clustering methods include K-means, Gaussian mixture model [6]) were adopted in the existing neural networks, such PointNet and PointCNN [7] and these clustering methods for point cloud would have significant impact on the performance. In fact, these clustering methods in PointCNN correspond to the down-sampling and pooling operations in a two-dimensional image convolution network. However, the down-sampling and pooling operations for the regular 2D image data are not applicable to the unordered point cloud. In literature, down-sampling of an unordered point cloud in a deep neural network usually down-samples the points regardless of their importance for the network output and often addresses down-sampling the raw point cloud before processing. As a result, some important points in the point cloud may be removed, while less significant points may be passed to next layers [8]. Therefore, it is necessary to have a new method designed specifically for the clustering of point cloud which can enable the network to extract multi-dimensional representative features.

Next, we will present several related progress in this field, pinpoint some critical issues in the current works and then propose our solutions. The remainder of this paper is organized as follows: Section II reviews some related works of 3D object recognition based on the point cloud, and then analyze the possible problems in these works. Then, the detailed methodologies of our proposed deep learning architecture are explained in Section III. Experimental results and

discussions on future research are given in Section IV and V respectively.

2. Related Works

As this study mainly focus on object classification for point cloud, only related literature on 3D object classification is reviewed here. So far, a few methods have been proposed for the object classification of point cloud, according to the principles and ideas of these methods, they can be roughly categorized as the following groups.

- Multi-view based methods [9]. In this cluster, the 3D object classification problem is transformed into multiple 2D object classification, so that those existing 2D image classification models can be used. Since the accuracy of this type of methods heavily depends on a number of typical different views taken, they are actually regarded as non-3D approaches. Also, such approaches require subsequent processing to resolve different view inconsistencies as well as possible outliers of different views, they are quite challenging in achieving high accuracy in general [9].

- Voxel-based 3D convolution methods [10]. In this group, 3D convolution operation is defined and performed for representing an object as a voxel in the space. Such representation allows us to use the convolution and pooling operations for the processing efficiency. However, due to the sparsity of object features in point cloud and the computational cost of the 3D convolution, the capacity of such voxel representation is strictly limited by the actual voxel resolution. In addition, this type of methods require a regular shape as a prior, thus limiting its further applications.

- Mesh-based non-Euclidean methods [11]. In this case, the 3D point cloud can be represented as a mesh structure, and then feature extraction is performed on the mesh which is used to construct the feature descriptors. Since the actual unordered point clouds in most cases do not have the connection information among their nodes, it is much difficult to obtain

the accurate mesh information by simply performing the surface reconstruction.

- Point cloud-based 3D classification neural networks, such as PointNet [12], PointNet++ [13] and their variants. These approaches mainly focus on extracting object shapes in point cloud directly. As an important end-to-end 3D classification network based on point cloud, PointNet has taken the rotation and disorder of point cloud data into consideration partially with some extent. For example, they exploit the symmetric functions such as the Max and $Average$ pooling to lessen the effects of disorder of the input data. Basically, PointNet learns a global representation in point cloud via computing individual point features from per-point Multi-Layer-Perceptron (MLP) first and then aggregating all features of the given object. As an improved version for the PointNet, PointNet++ presented a new feature extraction method and implemented feature learning in a different dimension. Unfortunately, PointNet++ has the difficulties for learning the relationship between two points as it only uses the symmetric functions to acquire the statistical characteristics of a multi-dimensional point set. Also, its complicated architecture leads to slow speed and not suitable for real-time applications. Zhang et al. [14] proposed a Graph-CNN architecture called PointGCN for classifying objects in 3D point cloud by exploring the existing graph convolution operation with two types of specifically designed pooling layers. Since this approach lacks shape alignments in the spectral domain, as previously addressed by Li et al. [15], the performance is almost the same as PointNet [12]. Nonetheless, it provides a GCN-based method for representing point cloud and thus has instructive intuition for subsequent research. Recently, many researches focus on the use of GCN for effective local feature extraction [16] and GCN-based point cloud semantic segmentation [17] with certain progress. However, the research based on GCN for object classification is still in infancy as the challenges in data sparseness, disorder and noisy in point

cloud, have not been well addressed and also the weak adaptability of the constructed graph network to diverse point cloud data has not fully investigated. In other words, the framework of GCN only provides us with a feasible intuition, it is still necessary to enhance the adaptability of GCN via defining specific operations on point cloud as did in [14] for complicated classification tasks, which is the specific aim of this paper.

As observed above, GCN-based method still has much room to improve for complicated classification tasks. In this paper, we will focus on the object classification for 3D point cloud with pose variances, and the original GCN will be revised from different perspective for this task. Based on the experiments of this paper, one can see that the original GCN has limited capability for this task as evidenced with very low accuracy. In order to achieve this aim, as suggested by Defferrard et al. [18], a revised GCN requires to address the following fundamental issues for object classification for a poind cloud: (i) The graph construction. For example, considering that the input is a point cloud, there is no node connection information in this case, we need to build such connections. (ii) The graph pooling. The reason for graph pooling is that the original GCN itself does not change the number of nodes, but we need to propose a strategy of selecting the nodes being retained and passed to the next layer; in addition, the data on a graph cannot be operated with a conventional 3×3 pooling structure. Like the graph construction process, a pooling domain and a pooling center point should be determined, and then nodes can be retained accordingly. We will investigate these two critical issues plus other technical innovations in the remaining part of this paper.

The main contributions of this paper are:

- First, different from the point cloud representation of the Cartesian coordinate system, a novel rotation-independent auxiliary network is proposed with the aid of the spherical coordinate system;
- Second, in order to cope with the challenge of feature extraction caused by the disorder of point cloud data itself, a novel graph convolution network

was proposed;

- Third, in view of the particularity of point cloud data, how to effectively extract its global and local features and how to deal with the training problem of point cloud unbalanced data are also considered in this study.

3. The Proposed Approach

The original GCN is a method to extract the spatial features of the topology from the spectral domain [19], which can combine features on local surface patches,and these features are robust to the deformations of those patches in Euclidean space [20]. As addressed in [14], it is best to align the data for GCN, so that the graph network can be insensitive to diverse data of inputs. As we know, in an Euclidean space, if a picture in 3D point cloud is rotated, the new point cloud is no longer the original picture for the conventional GCN. To address this rotational problem, we propose a pose auxiliary network (PAN), which is one of the most important contributions of this paper to adapt the GCN to the point cloud classification task.

In general, there are three key aspects that need to be carefully considered in designing the architecture of the new network. First, in order to solve the problem brought by the rotations of the point cloud, a corresponding pose correction auxiliary network is designed first; second, a new set of convolution operations are designed to achieve the aim of feature robust to pose changing for the unordered points cloud; third, we design some new down-sampling or pooling methods to enhance the network's capacities for both global and local feature extraction. Besides of these critical issues, some techniques on dropout [21] to overcome over fitting and $BatchNormalization$ [22] to mitigate gradient diffusion, are also considered in this paper.

3.1. The Pose Auxiliary Network (PAN)

In this section, we develop a pose auxiliary network to serve the purpose of the rotation changes. Technically, we design a point cloud data coordinate

system transformation as part of the proposed PAN to achieve the rotation robust based on the Euler's rotation theorem [23]. In Comparison to PointNet and PointNet++, our model is not a simply regression aiming at an intermediate point cloud representation. Instead, we try to find a way to make the model pose independent. Theoretically, the geometry for the three-dimensional object, based on Euler's rotation theorem for any displacement of a rigid body with a point on the rigid body fixed, is equivalent to a single rotation about some axis that runs through the fixed point [23]. Furthermore, the composition of two rotations is also a rotation. Hence an arbitrary rotation could be described by only three parameters. In general, the mathematical representation of a point cloud rotation in a Cartesian coordinate system can be represented as follows,

$$\begin{bmatrix} x^s & y^s & z^s \end{bmatrix} = \begin{bmatrix} x^t & y^t & z^t \end{bmatrix} \times R \tag{1}$$

$$\begin{aligned} R = & \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \times \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \\ & \times \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2}$$

where $[x^t, y^t, z^t]$ is the original coordinates, $[x^s, y^s, z^s]$ is the rotated coordinates, R is the spatial rotation matrix. Formula (2) represents the specific definition of R, whose three matrices on the right side of the equation stand for the rotation around the X, Y, Z axis respectively. And R is a square matrix with full rank of 3. The spatial rotation matrix can be used to visually represent rotations at any angle along each axis. However, the matrix requires nine elements to perform a single rotation operation, and matrix multiplication may be compu-

tationally intensive. In order to reduce such computational costs, we will use another coordinate system, which can help convert a space coordinate system to a spherical coordinate system. The mathematical representation of the point cloud rotation in the spherical coordinate system is as follows,

$$\begin{pmatrix} r^s & \theta^s & \phi^s \end{pmatrix} = \begin{pmatrix} r^t & \theta^t & \phi^t \end{pmatrix} + \begin{pmatrix} 0 & \theta' & \phi' \end{pmatrix} \tag{3}$$

where (r^s, θ^s, ϕ^s) is the rotated coordinates, (r^t, θ^t, ϕ^t) is the original coordinates, $(0, \theta', \phi')$ is the rotation angle. Apparently, in this spherical coordinate system, only two parameters are required to determine a rotation operation. There is no doubt that the calculation process will be significantly simplified from a matrix multiplication to a vector addition. The point cloud view in both Cartesian coordinate system and spherical coordinate system are shown in figure 1, and one can see that the object shape in the spherical coordinate system is not as explicit as in the Cartesian coordinate system, however, its advantage in computation of rotation representation is obvious as demonstrated above.

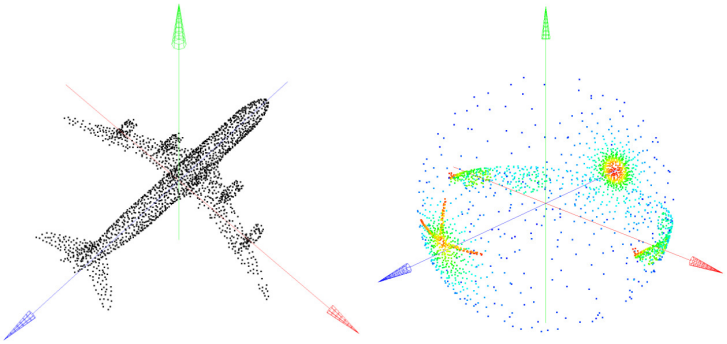


Figure 1: The same point cloud view in two coordinate systems (left: spatial Cartesian coordinate system; right: spherical coordinate system)

Furthermore, a point cloud on the spherical coordinate system can be equivalently mapped to a unit spherical point cloud with texture information (gray-scale), as shown in figure 1. That is, in the new coordinate system, (θ, ϕ) represents the angle at which any point exists in the unit sphere coordinate

system, and r is regarded as the gray level (pixel value) on the unit sphere.

On the basis that the spherical coordinate system is relatively convenient for calculating the rotation of the point cloud, we propose a pose auxiliary network which aims to learn the corresponding angle correction value (θ^f, ϕ^f) from the point cloud data rotated at any angle along each axis. Then we use them as the input to the future GCN as shown in the figure 2. In this case, the input data format of the auxiliary network is the (θ, ϕ) part of the spherical coordinate system. In order to accelerate the model convergence speed, we normalize the point cloud input data in spherical coordinate format. The structure of the auxiliary network is shown in figure 2 and the details can be referred to Alg.1.

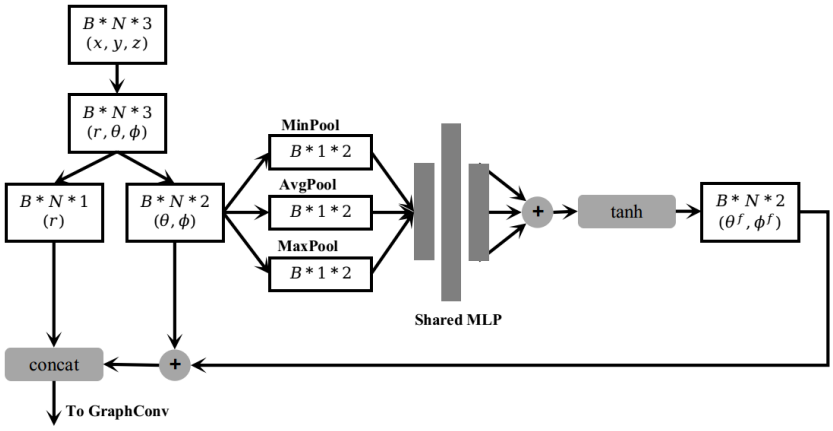


Figure 2: A diagram of the auxiliary network structure

In summary, by mapping the 3D point cloud from the Cartesian coordinate system to the spherical coordinate system, the new proposed auxiliary network (PAN) can learn angle corrections for the new inputs. Then, the obtained angle correction information will be concatenated with the original point cloud data and entered into the subsequent graph convolution network for feature extraction.

3.2. The Graph Representation Approach

As it is well-known that the loss of local geometry in 3D point cloud has a negative impact on classification performance [12]. In this paper, we use a

graph convolution module to achieve the preserving the local geometric details of 3D object that are conducive to the classification task.

For graph convolution, the input is a topological graph. In order to extract the point cloud features using the graph convolution, an undirected weighting map $G = (V, E, W)$ needs to be constructed for the unordered point cloud data X with N nodes/points, where V is the vertex set, E is the edge set, and W is the weighted adjacency matrix that stands for the edge weight. The vertex set V of the undirected weighted graph contains all nodes $v_i = x_i, i \in [0, N-1]$ in X, and the edge set E is built from the vertex set V by applying the K nearest neighbor (KNN) algorithm [24]. In the construction procedure, for each vertex $v_i, i \in [0, N-1]$ in the vertex set V, we select the K nodes $(v_j, j \in [0, K-1])$ with the smallest distances in terms of the Euclidean distance, and then build the edge set $(v_i, v_j) \in E$. The edge weights are calculated by the following radial basis functions,

$$W(v_i, v_j) = \begin{cases} \exp(-\frac{\|v_i - v_j\|^2}{\delta^2}), & (v_i, v_j) \in E \\ 0, & (v_i, v_j) \notin E \end{cases} \tag{4}$$

where δ is the mean of the Euclidean distance between all connected nodes in the edge set E:

$$\delta = \frac{1}{N \times K} \sum_{i=1}^{N} \sum_{j=1}^{K} \|v_i - v_j\|^2, (v_i, v_j) \in E \tag{5}$$

Based on the adjacent matrix W, we can choose the symmetrically normalized Laplace matrix of $L = I_n - D^{-1/2} W D^{-1/2}$ to perform a spectral field operation on the graph as did in [25], where D is the degree (out-degree and in-degree) matrix and I_n is the identity matrix. In fact, the graph convolution method is inspired by the Fourier transformation on a graph, and the essence of such operation is to convert the convolution of graph signals in the time

domain into products on the spectral domain according to the graph Fourier transformation, which can be represented as follows,

$$(f * h)_G = U((U^T h) \odot (U^T f)) \tag{6}$$

where $\odot$ is the Hadamard product, which means the element by element multiplication of the corresponding positions of two vectors, matrices and tensors with the same dimensions, U is the eigenvector of the Laplace matrix L. f is the feature map, and h is the convolution kernel. In order to reduce both the network complexity and training time, we will choose the convolution formula based on the Chebyshev polynomial fitting in this paper, which is defined as follows,

$$\begin{aligned} y_{output} &= \zeta(U \sum_{k=0}^{K-1} \beta_k T_k(\tilde{\Lambda})x), \\ \tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - I_n, \tilde{\Lambda} &\subset [-1, 1]; T_0 = I_n; T_1 = \tilde{\Lambda}; \\ T_k(\tilde{\Lambda}) &= 2\tilde{\Lambda}T_{k-1}(\tilde{\Lambda})x - T_{k-2}(\tilde{\Lambda})x \end{aligned} \tag{7}$$

where ζ is the activation function, k is the number of terms of Chebyshev polynomials [18], T_k is the Chebyshev polynomials, β_k is the trainable weight, I_n is the identity matrix, $\tilde{\Lambda}$ is the eigenvalue matrix of normalized Laplace polynomials. In order to reduce the number of network parameters, we choose the Chebyshev polynomials with $k = 3$ (Please refer to Fig. 3, which shows the different computational scopes when different k values are selected). Meantime, for speeding up the convergence of the model, we select *Relu* as the activation function. Finally, the mathematical expression of the convolution layer we use

is as follows,

$$y_{output-new} = Relu(\beta_0 x + \beta_1 \tilde{L} + \beta_2 (2\tilde{L}^2 - 1)x + b), \quad \tilde{L} = \frac{2L}{\lambda_{max}} - I_n \tag{8}$$

In the above formula, b is the bias of the convolution layer and $\tilde{L}$ is the normalized Laplace matrix. λ_{max} is the maximum eigenvalue of $\tilde{L}$, and I_n is the identity matrix with the same rank as $\tilde{L}$. The above formula can intuitively explain the purpose of the graph convolution proposed in this paper, that is, the weighted summation of the feature information of the third-order neighbor nodes for each node in the graph is taken as the new feature information of the node. Compared to traditional convolutions, in addition to the feature information of each node as input, graph convolution also requires information of the symmetric normalized Laplacian matrix constructed based on the previous nodes.

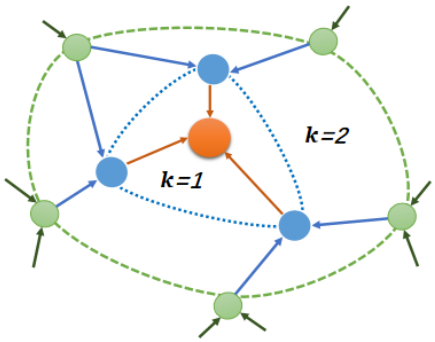


Figure 3: The illustration for calculation ranges with different k values

The adaptive design of the graph convolution (*GraphConv*) operation can be referred to figure 4. The *GraphConv* operation will not change the number of nodes in the graph, only changes the feature dimension of the nodes. The subsequent graph pooling operation (*GraphPool*) mainly obtains the pooled node feature matrix by selecting the appropriate pooling function according to the pooling domain range of each pooling center point. Please refer to figure 5

and details will be provided in the next section.

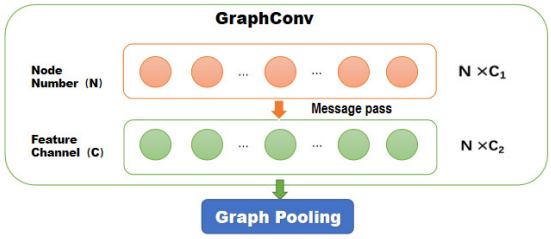


Figure 4: The structure of the graph convolution operation (GraphConv)

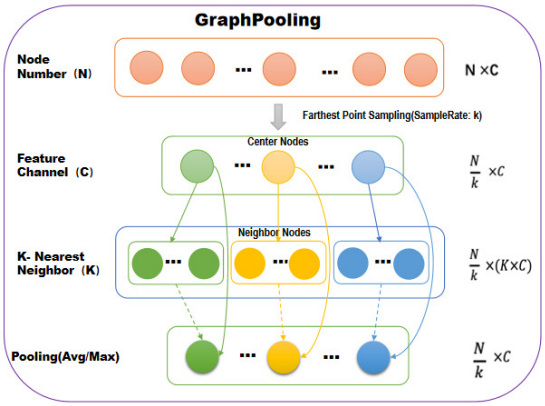


Figure 5: The structure of the graph pooling operation (GraphPool)

3.3. Graph based down-sampling and pooling operations

According to the descriptions in last section, we know that the graph convolution will not change the number of nodes, but only the feature channels of each node. Therefore, we need to design a down-sampling method to reduce the number of nodes, and premeditate using pooling method to obtain the suitable dimension of the features for those selected nodes.

Different from regular data, which utilizes the uniform sampling to reduce the amount of data, the unordered sparse feature of point cloud data requires that the selected down-sampling method not only can reduce the amount of node data, but also retain the spatial features of the origin cloud data. Therefore,

we choose the FPS (farthest point sampling) as the down-sampling algorithm for unordered point cloud since this technique can yield an elegant result with efficient implementation as its complexity remains tractable even when we are modeling a non-uniform data [23]. The specific procedure of this algorithm is described as follows:

(1) Randomly select a point v_i from the origin cloud data X first, and add it to the resulted set Y; (2) Then delete this point v_i from set X and select the farthest point from the origin cloud data to join in the resulted set, and delete this point again from the origin cloud data X; (3) By iterating the process until the number of points in the resulted set reaches a given threshold, and the final result set will be the point cloud data after sampling.

The comparison for one poind cloud before and after sampling is shown in figure 6, which illustrates that the point cloud data after FPS retains the sparse and spatial features of the origin point cloud.

Figure 6: Farthest point sampling (left:origin cloud, 2048 points; right:after sampling, 128 points))

Before the pooling operation, we need to determine the pooling center for each point and the pooling scope. Of course, we can take the FPS sampling result set Y as the centers of the pooling point set, and the sphere space with the pooling center as the spherical center and the radius of R as the pooling scope (as shown in figure 7). Finally, we can obtain the K nearest neighbor for each pooling center.

The whole process can be referred to Alg.2 in Appendix. What needs to be emphasized is that down-sampling is conducive to the reduction of vertex number, and pooling is conducive to the extraction of vertex features.

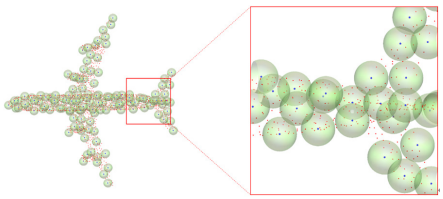


Figure 7: Illustration of the pooling scope

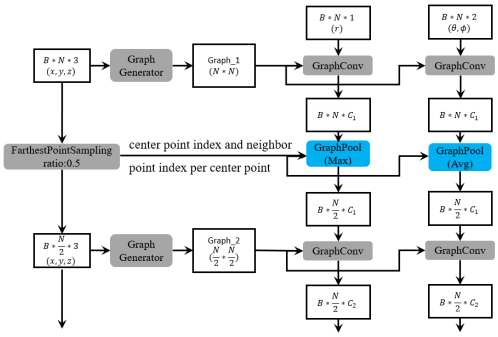


Figure 8: Multi-dimensional feature extraction network structure diagram

3.4. The Pipelined Architecture of the New Model

As detailed in Section 3.1, we exploit the point cloud coordinate system transformation and PAN to solve the robust problem of point cloud rotation. Then, with the help of FPS and proposed pooling method, one can finally extract the features of unordered point clouds on a multi-dimensional level. With these preparations, we can propose the following network architecture (as shown in Fig.9) to solve the classification problem for the unordered point cloud.

The overall network architecture (as shown in figure.9) consists of a pose auxiliary network (*RotateBlock*), three feature extraction units: *ParallelBlock* and two fully connected layers (*Dense_1* and *Dense_2*). Among them, *RotateBlock* is a new designed PAN module as described previously; *ParallelBlock* is the feature extraction module, which includes two *BasicBlocks*. And the *BasicBlock* consists of the graph convolution operations and the corresponding pooling operations; the FPS module and graph generator module provide *ParallelBlock* with the required graph data.

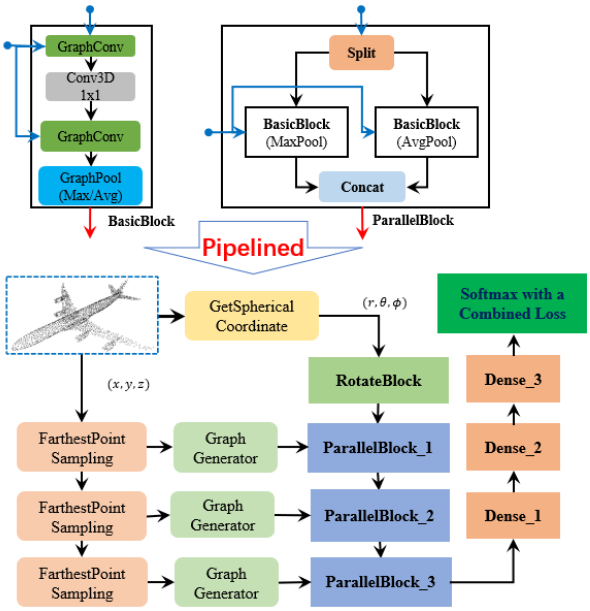


Figure 9: Architecture of point cloud classification network based on graph convolution

Technically, *ParallelBlock* is a feature extraction unit, which contains two sets of parallel *BasicBlocks* (basic feature extraction modules), which respectively extract features on the spatial information and texture information of point cloud data. Each *BasicBlock* consists of two *GraphConv* (graph convolution module, please refer to figure 4), one $Conv3D\_1 \times 1$ (1×1 convolution module) and one *GraphPool* (graph convolution pooling module, please refer to figure 5). The combined structure of two *GraphConv* and one $Conv3D\_1 \times 1$ enlarges the convolution receptive field of the graph while making the network parameter increase less, that is, a small amount of network complexity increases in exchange for a stronger network generalization ability. At the same time, the *BasicBlock* uses *GraphPool* to implement graph convolution operations based on two kinds of features, as well as corresponding to the maximum pooling (texture information) and the average pooling (spatial information) operations.

As we pointed out in the subsequent experimental section, the ModelNet40 [26] dataset used in the experiment has an uneven distribution of sample cat-

egories. In fact, the data of the real scenarios will also face this situation. Therefore, in order to solve the problem of uneven distribution of sample categories in the data set, a penalty weight for the sample categories is constructed. That is to say, there are K point cloud samples in the point cloud dataset A, and the number of samples of each type is $n_1, n_2, ...n_K$, then the weight calculation formula of the kth sample is as follows,

$$Weight_k = \frac{\sum_{i=1}^{K} n_i}{n_k} \times \sum_{j=1}^{K} \frac{n_j}{\sum_{i=1}^{K} n_i} \times K + 1 \tag{9}$$

In the above formula, multiplying the coefficient K and adding the value 1 makes $Weight_k > 1, k \in [1, K]$. That is, the weight will not reduce the size of the original loss value, nor will it affect the original convergence speed.

As illustrated in figure 9, the new model exploits the Softmax and a combined loss function. The first term is the Softmax cross entropy loss, which represents the accuracy of the point cloud classification based on probability and its cross entropy. The $loss$ term is expressed by minimizing the negative log-likelihood,

$$L_{soft}^{loss} = -\sum_{i=1}^{K} y_i \ln \hat{y}_i, \hat{y}_i = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{10}$$

where y_i is the probability distribution of the true sample labels, and $\hat{y}_i$ is the probability distribution of the normalized prediction output.

To prevent the network from over-fitting during the training, the weight of $L2$ regularization as the second term for the combined $loss$, is expressed as follows,

$$L_2^{loss} = \frac{1}{2} \sum_{i=1}^{M} w_i^2 \tag{11}$$

where M is the number of trainable weights in the network, and w_i is a trainable

weight value. Then, the combined Loss for this network is as follows,

$$Loss = L_{soft}^{loss} \times Weight_k + \mu L_2^{loss}, k \in [1, K] \tag{12}$$

where the first term is the loss of the Softmax cross entropy with penalty weights, and second term is L_2 regularization multiplied by learning rate μ.

4. Experimental Results and Analysis

4.1. Point cloud dataset preparation

For training the proposed model and comparison with others, we conduct experiments on the well-known ModelNet40 dataset [26]. This dataset is a collection of CAD models with 40 object categories. The dataset includes 9,840 objects for training and 2,468 objects for testing. Each CAD model in the ModelNet40 dataset is stored in an object file format (OFF). The OFF file stores the vertex, face, and edge information of the CAD model. This study is based on point cloud data, so only the vertex information of the model in the ModelNet40 dataset is used.

With the given setup for training and testing samples, the object types in training set and testing set are not evenly distributed To cope with this uneven distribution of the object categories in the dataset, we introduce a penalty weight for object categories in training. In the training stage, different categories are given different weights according to their number of samples, which implies that the category with smaller number of samples will assign the larger weight, and the category with larger number of samples will have the smaller weight. Then the weighted loss function is calculated according to the weight as did in [27]. Also, as for the evaluation, in order to reduce the impact of the uneven distribution of sample categories in the dataset, we use the Receiver Operating Characteristic Curve (ROC) and Precision Recall Curve (PR) as evaluation indicators.

4.2. Training Procedure and Hyper-parameters Tuning

The training configuration is detailed as follows.

Loss Function. The soft-max cross entropy is used as the loss function defined in formula 10.

Regularization . In order to prevent the over-fitting in the training process, a term with L_2 regularization of all weights is added to the original loss function. In the same time, the dropout of $keep\_prob = 0.9$ is used in the convolution layer and full connection layer respectively.

KNN parameters. The graph of the point cloud data is constructed by the k-nearest neighbor algorithm. In order to balance the training time and network performance, we use 20 nearest neighbors of each point as the input of graph convolution.

Points' number . A single point cloud in the ModelNet40 dataset contains 2048 points. We use the FPS to reduce the number of points in a point cloud to 512, and then use them for training.

Rotation of testing and Training epoches . In order to make the network robust to the pose variances of the point cloud, we rotate the point cloud of all the inputs around the three spatial coordinate axes of x, y, z respectively in the range of $(-\pi, \pi)$ angle randomly (only applied to testing set). Since we observed that the network has converged when the process iterates 50 epochs, 50 is used as the training epoch.

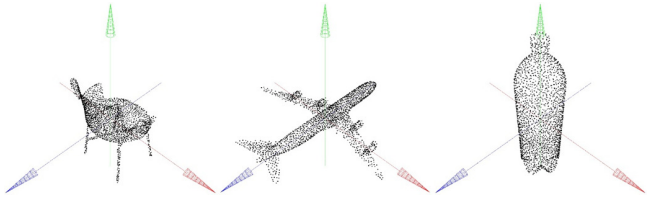
Learning rate. The initial learning rate is 0.01, the decay period is 100 batches, and the decay rate is 0.97.

4.3. Evaluation

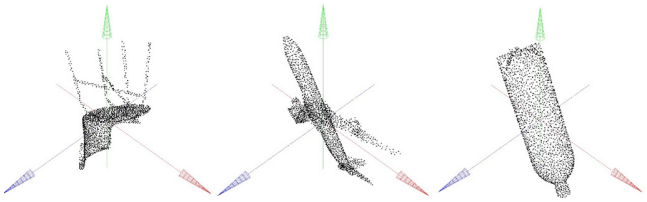
In the literature, some typical end-to-end network models for the object classification with point clouds are: PointEdge[28], PointNet, PointNet++, PointCNN.

4.3.1. Experiments on the point cloud input without rotation

According to the experimental setup, the input point cloud is not rotated, and the fixed pose given in the data set is maintained (as shown in figure 10 and refer to figure 10.a). The results of this new model compared with other point cloud classification models are shown in figure 11.

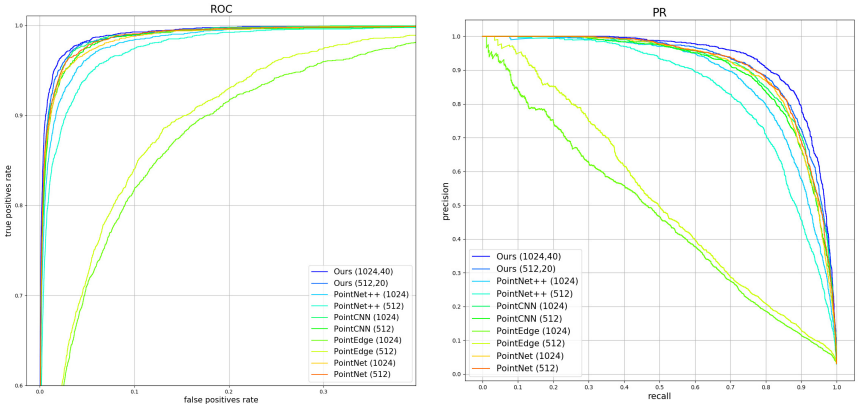


(a) Samples without rotation



(b) Samples with random rotation

Figure 10: The demonstration of dataset with rotation or without rotation



(a) ROC (b) PR

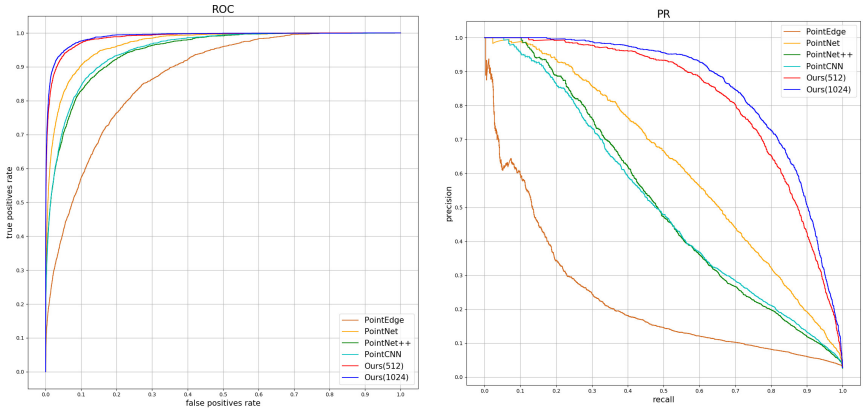
Figure 11: ROC and PR curves for the classification task on the pose-fixed point cloud

In figure 11, the performances of PointNet, PointNet ++, PointEdge, PointCNN,

and the new model under the input point cloud points of 1024 and 512 are compared respectively. It can be seen that the new method performs the best in all these experiments. At the same time, we explored the impact of point cloud sparseness and node connectivity on the new model. As shown in figure 11, the performance of the proposed model with the number of input point cloud nodes 1024 and the node connectivity 40 is much better than the model with number of input point cloud nodes 512, and the node connectivity 20. This demonstrates that the performance of the proposed method is better than other models in the case of fixed pose. This is due to the fact that we considered the uneven distribution of the training and testing samples in our model besides of the new features we extracted.

4.3.2. Experiments on the point cloud input with random rotations

In this experiment, the input point cloud has been randomly rotated in pose (as shown in figure 10.b). But the hyper-parameters are exactly the same as the last experiment. The performance of the corresponding methods after the point cloud rotations is shown in figure 12.



(a) ROC (b) PR

Figure 12: ROC and PR Curve

As shown in figure. 12, the new model's performance is much superior to other network models such as PonitNet, PointNet++ and so forth. The better

performance can be contributed to the strategy of providing the network with both the robust pose transformation and the much more comprehensive feature extraction structure. At the same time, it can be noticed that the new model tends to achieve better classification performance for the node-intensive point clouds and the graphs with high node connectivity according to the Fig. 12. It is worth pointing out that by observing the PR curves of Fig. 12 and Fig. 11, we can intuitively see that the rotation of the point cloud has a greater impact on all methods. However, the proposed model is relatively quite stable on the PR curve.

Table 1: Test accuracy

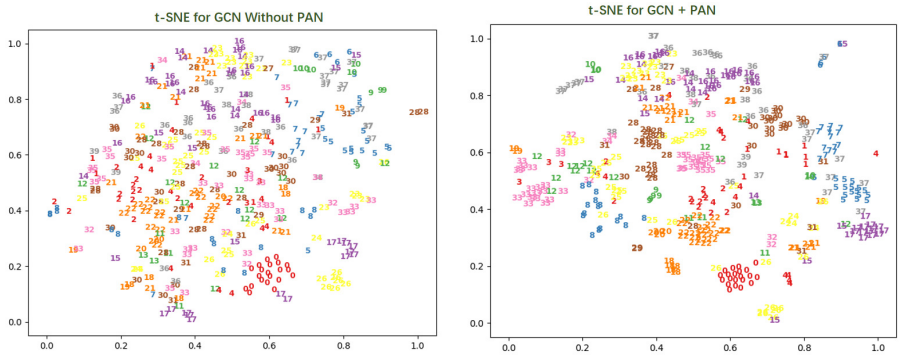
Model	Test Auccary
PointEdge	23.46%
PointNet	57.38%
PointNet++	46.92%
PointCNN	47.16%
Ours(512,K=20)	69.24%
Ours(1024,K=20)	73.07%

In addition, as can be seen from the table. 1, the network is superior to other models in terms of recognition accuracy. It is worth noting that the accuracy for PointNet is the second best to our method since it takes into account the random rotation by using an STN network [12]. Also, we noticed that our model can achieve even better accuracy by increasing the density of the input point cloud. However, for describing the same object, an increase with the number of point cloud nodes implies an increasing amount of workload for calculation, which leads to a slower network training. How to make a balance between speed and performance will be our future research.

4.3.3. Visualizing the features generated by GCN with PAN and without PAN

As shown in the previous experiments, the proposed approach can achieve much better performance, especially in the case with pose rotations. We believe this contributes to the PAN in the proposed approach. In this section, we denote the network without PAN as GCN and the proposed whole network

as GCN+PAN. In this experiment, the features extracted by GCN only and GCN+PAN were visualized, where the feature dimension was reduced to 2D by using t-SNE algorithm [29] in order to visualize the distribution of feature space properly (as shown in figure 13). Experimental results show that the GCN has a relatively poor differentiating capacity on ModelNet40 dataset. Conversely, due to the addition of the PAN module, GCN's feature expression capabilities have been greatly enhanced.



(a) GCN without PAN on ModelNet40 dataset (b) GCN with PAN on ModelNet40 dataset

Figure 13: The features after t-SNE for GCN without PAN or with PAN under random pose changing

4.3.4. Experiments on the efficiency of the model

Based on the results in Table 2, one can find that graphs with higher node-intensive point clouds and higher node connectivity tend to achieve better performance for point cloud object classification. By contrast, some relatively sparse point clouds and graphs with low connectivity can increase the efficiency. However, dense point clouds and highly connected graphs would significantly reduce model training speed.

Therefore, we infer that there is redundancy in dense point clouds and highly connected graphs. Then, in order to obtain a network model with better performance and faster training speed, the nodes and graphs in the training data need to be streamlined and optimized. Specifically, we use the FSP to reduce

the number of nodes for the point set in the training data. Though the KNN approach is used to build a graph in many papers including our paper, it will cause the edges that are not on the surface of the point cloud, thus they may not meet the conditions required by the point on a real point cloud surface. In general, point cloud surface reconstruction algorithms, such as Poisson Reconstruction (POR), Delaunay Triangulation (DT), etc. would behave differently in terms of computing resource consumption. In this section, we use the DT method to replace the KNN approach since DT has a much lower time complexity. The reconstruction result by DT randomly increment method is shown in figure 14.

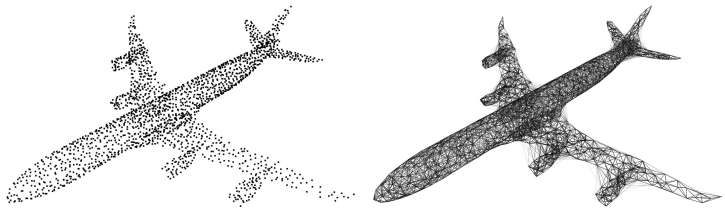


Figure 14: Unordered point cloud (left) and undirected weighted graph constructed by Delaunay Triangulation method (right)

Table 2: Performance comparison of the different graph construction algorithms

Vertex number	Type	Edge number	Time/epoch*	Test accuracy
512	KNN(K=40)	11471	904.59ms	70.87%
	KNN(K=20)	5633	888.71ms	69.27%
	Delaunay Triangulation	3012	873.26ms	70.20%
1024	KNN(K=40)	22971	1248.65ms	73.25%
	KNN(K=20)	10969	1193.25ms	73.07%
	Delaunay Triangulation	6150	1097.85ms	73.02%

* Run time measured on the configuration: Intel Xeon(R)CPU E5-2683v3,GeForce GTX 1080 Ti

As can be seen in the table 2, DT can construct a graph with fewer connected edges than the K-nearest neighbor algorithm, thereby it can increase the speed of network training and reduce the complexity of building the model, while keeping the similar performance. Because the DT's edge set can be regarded as a subset of the edge set obtained by the K nearest neighbor algorithm, there are not much differences between the K nearest neighbor composition algorithm

and DT with a node connectivity of 40.

5. Conclusion and Discussion

In this paper, we propose an end-to-end cloud point classification model based on graph convolution networks. The network model architecture mainly includes the following aspects: (1) A pose correction auxiliary network is designed for rotation invariances based on the rigid body transformation theory in the spherical coordinate system when extracting point cloud convolution features; (2) Graph convolution operation based on the KNN method combined with Chebyshev polynomial fitting strategy is proposed to achieve the extraction of global and local features on unordered point clouds; (3) Based on the FPS and mapping the spherical coordinate system to the gray scale image, a convolution pooling structure of the graph is developed in order to further extract multi-dimensional features from the unordered point cloud. The PR curves in Fig. 11 and Fig. 12 respectively visually and quantitatively show the classification performances of different models when the point cloud target is rotated or not. By observing the different presentations of the PR curves in Fig. 11 and Fig. 12, two conclusions could be reached: one is that many existing point cloud classification models are actually sensitive to the target rotation represented by the 3D point cloud; the other is that our new model exhibits strong robustness to point cloud rotations as expected.

In general, the new network model can accurately recognize 40 types of point cloud objects, but there are still a large gap errors for a few classes. For example, the $flower\_pot$ class is more likely to be identified as plant and vase. By observing the original $flower\_pot$ point cloud model, we know that $flower\_pot$ consists of two parts, plant and vase. Therefore, this kind of misidentification reflects the problem that the point clouds for these two classes are not fine-grained in the ModelNet40 dataset. It also further validates the effectiveness of the new model for point cloud classification tasks corresponding to other single objects.

Currently, GCNs are mainly used to predict individual relations in social networks, model proteins for drug discovery, enhance predictions of recommendation engines, efficiently segment large point clouds, etc. The increasing proliferation of non-Euclidean data in real-world applications will give us more chances for applying GCNs in different domains, as the limited performance of CNNs when dealing with such data [30].

Moreover, as mentioned in the experiments, we found that the training results are better for the graphs with the dense point cloud and high connectivity, but the performance improvement is not very significant compared with the graph with sparse point cloud and low connectivity. This shows that there are a lot of redundant points and edges in the graph for training and we will explore a balance strategy in the future study.

In order to leverage the advantage of convolutional networks such as weight sharing and local connectivity, the raw point clouds are usually converted to some other representation, such as 3D occupancy grids or rendered 2D images. However, these conversions are often irreversible, and destroy the local geometric structure of the original point cloud. In this paper, we explored a new way of converting point clouds to a representation suitable for deep learning, without destroying any geometric information. Specifically, our new proposed method focusd on solving the three major problems in point cloud object classification: unordered, rotation sensitive and multi-dimensional feature learning. The experimental results illustrate that the new graph based CNN architecture performs better than other recent models in dealing with the classifications for random rotations of point cloud. As can be seen from figure 15 , our new model serves as a uniform feature extractor for the same object with different poses, which can be observed from the feature distribution after concatenation on figure 15. In the future, we will give much efforts on presenting a more feasible representation for the input topology graph by implementing much efficient surface reconstruction methods to obtain an even better classification performance. In addition, the down-sampling method proposed in this paper still has a lot of room for improvement in our future research work.

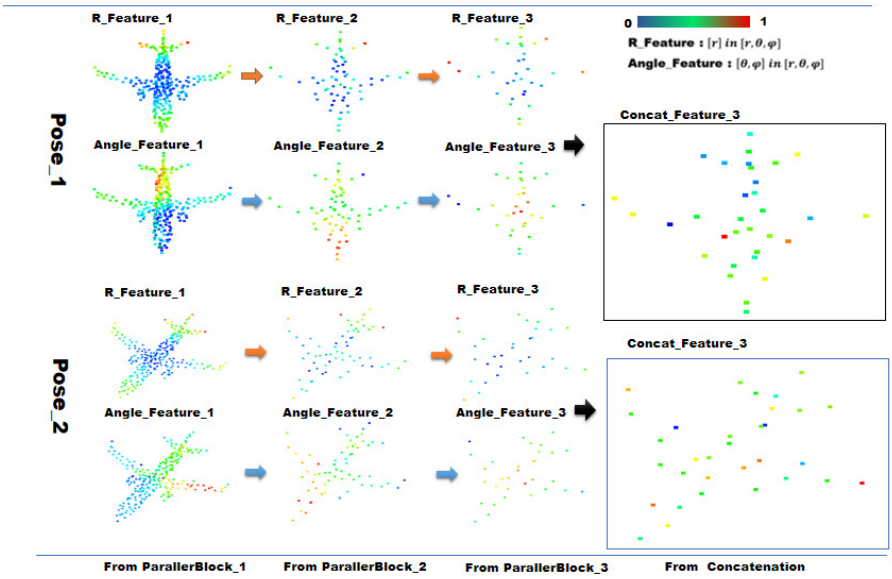


Figure 15: The visualization for the extracted critical feature of the new model under different poses

6. Acknowledgment

The authors would like to thank Mr. Haoyang Ma and Mr. Xin Jing for their help on the point cloud dataset preparation and thank Mr. Qi Zhang and Mr. Hanlin Li for their help on validating our new model. This work was partially supported by a grant from the National Natural Science Foundation of China (No.61573019, No.61703006, and No.61602321) and 2020 Hebei Provincial Science and Technology Plan Project (No. 203777116D). Also we would appreciate the support of campus funding in North China University of Technology.

7. Appendix

7.1. Algorithms for angle correction and multi-dimensional feature extraction

Algorithm 1 Angle correction algorithm for rotating auxiliary network

Require: $(x_i, y_i, z_i)_b$: Point cloud data as $[B, N, 3]$,$i \in [1, N], b \in [1, B]$;

Ensure: $(r_i, \theta_i', \phi_i')_b$: Point cloud data in the modified spherical coordinate format $[B, N, 3]$,$i \in [1, N], b \in [1, B]$;

for $b = 1, 2, \ldots, B$ **do**
  for $i = 1, 2, \ldots, N$ **do**
    $(r_i, \theta_i, \phi_i)_b \leftarrow SphericalCoordinate((x_i, y_i, z_i)_b)$
    $(r_i, \theta_i^n, \phi_i^n)_b \leftarrow Normalization((r_i, \theta_i, \phi_i)_b)$;
    $(r_i)_b, (\theta_i^n, \phi_i^n)_b \leftarrow Split((r, \theta^n, \phi^n))$;
  end for
  $(\theta_{avg}, \phi_{avg})_b \leftarrow Average((r_i, \theta_i^n, \phi_i^n)_b)$, $i \in [1, N]$;
  $(\theta_{min}, \phi_{min})_b \leftarrow Minimum((r_i, \theta_i^n, \phi_i^n)_b)$, $i \in [1, N]$;
  $(\theta_{max}, \phi_{max})_b \leftarrow Maximum((r_i, \theta_i^n, \phi_i^n)_b)$, $i \in [1, N]$;
  $(\theta^f, \phi^f)_b \leftarrow \sum_i SharedMLP((\theta_i, \phi_i)_b)$, $i \in \{avg, min, max\}$;
  $(\theta_i', \phi_i')_b \leftarrow (\theta_i^n, \phi_i^n)_b + (\theta^f, \phi^f)_b$ for each $i \in [1, N]$;
  $(r_i, \theta_i', \phi_i')_b = Concat((r_i)_b, (\theta_i', \phi_i')_b)$
end for

Algorithm 2 Algorithm for multi-dimensional feature extraction of point cloud data

Require:

$(x_i, y_i, z_i)_b$: Point cloud data of spatial rectangular coordinate system as $[b, N, 3]$,$i \in [1, N], b \in [1, B]$;

$(r_{i1}, r_{i2}, \ldots, r_{in}, a_{i1}, a_{i2}, \ldots, a_{in})_b$: Point cloud feature as $[B, N, 2C_1]$,$i \in [1, N]$,

$b \in [1, B], n \in [1, C_1]$;

Ensure:

$(x_j, y_j, z_j)_b$: Point cloud data $[B, \frac{N}{R}, 3]$, $j \in [1, \frac{N}{R}]$,

$b \in [1, B]$,R;

$(r'_{j1}, r'_{j2}, \ldots, r'_{jm}, a'_{j1}, a'_{j2}, \ldots, a'_{jm})_b$: $[B, \frac{N}{R}, 2C_2]$,

$j \in [1, \frac{N}{R}], b \in [1, B], m \in [1, C_2]$,$R$;

1: **for** $b = 1, 2, \ldots, B$ **do**

2: $G_b^{n*n} \leftarrow GraphGenerator((x_i, y_i, z_i)_b), i \in [1, N]$;

3: $(r_{i1}, r_{i2}, \ldots, r_{in})_b, (a_{i1}, a_{i2}, \ldots, a_{in})_b \leftarrow Split((r_{i1}, r_{i2}, \ldots, r_{ic}, a_{i1}, a_{i2}, \ldots, a_{ic})_b)$, $i \in [1, N], n \in [1, C_1]$;

4: $(r'_{i1}, r'_{i2}, \ldots, r'_{im})_b \leftarrow GraphConv((r_{i1}, r_{i2}, \ldots, r_{in})_b, G_b^{n*n})$, $i \in [1, N], n \in [1, C_1], m \in [1, C_2]$;

5: $(a'_{i1}, a'_{i2}, \ldots, a'_{im})_b \leftarrow GraphConv((a_{i1}, r_{i2}, \ldots, a_{in})_b, G_b^{n*n})$, $i \in [1, N], n \in [1, C_1], m \in [1, C_2]$;

6: $(r'_{j1}, r'_{j2}, \ldots, r'_{jm})_b \leftarrow MaxPool((r'_{i1}, r'_{i2}, \ldots, r'_{im})_b), i \in [1, N]$, $j \in [1, \frac{N}{R}], m \in [1, C_2]$;

7: $(a'_{j1}, a'_{j2}, \ldots, a'_{jm})_b \leftarrow AvgPool((a'_{i1}, a'_{i2}, \ldots, a'_{im})_b), i \in [1, N]$, $j \in [1, \frac{N}{R}], m \in [1, C_2]$;

8: $(x_j, y_j, z_j)_b \leftarrow FPS((x_i, y_i, z_i)_b), i \in [1, N]$, $j \in [1, \frac{N}{R}]$;

9: $(r'_{j1}, r'_{j2}, \ldots, r'_{jm}, a'_{j1}, a'_{j2}, \ldots, a'_{jm})_b = Concat((r'_{j1}, r'_{j2}, \ldots, r'_{jm})_b, a'_{j1}, a'_{j2}, \ldots, a'_{jm})_b)$, $j \in [1, \frac{N}{R}], m \in [1, C_2]$

10: **end for**

References

[1] L. W. H. T. W. P. Liu W, Sun J, Deep learning on point clouds and its application: A survey, Sensors (19) (2019) 4188–4207.

[2] H. Zhao, M. Tang, H. Ding, Hoppf: A novel local surface descriptor for 3d object recognition, Pattern Recognition 103 (2020) 107272. doi:https://doi.org/10.1016/j.patcog.2020.107272.
URL http://www.sciencedirect.com/science/article/pii/S0031320320300777

[3] A. M. Araújo, M. M. Oliveira, A robust statistics approach for plane detection in unorganized point clouds, Pattern Recognition 100 (2020) 107115. doi:https://doi.org/10.1016/j.patcog.2019.107115.
URL http://www.sciencedirect.com/science/article/pii/S0031320319304169

[4] R. B. Rusu, S. Cousins, 3d is here: Point cloud library (pcl), in: 2011 IEEE international conference on robotics and automation, IEEE, 2011, pp. 1–4.

[5] T. D. Barfoot, State Estimation for Robotics, Cambridge University Press, 2017.

[6] C. E. Rasmussen, The infinite gaussian mixture model, in: Advances in neural information processing systems, 2000, pp. 554–560.

[7] 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, 2018, pp. 820–830.

[8] E. Nezhadarya, E. Taghavi, R. Razani, B. Liu, J. Luo, Adaptive hierarchical down-sampling for point cloud classification, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[9] M. Gadelha, R. Wang, S. Maji, Multiresolution tree networks for 3d point cloud processing, in: The European Conference on Computer Vision (ECCV), 2018.

[10] J. Wu, C. Zhang, T. Xue, W. T. Freeman, J. B. Tenenbaum, Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling, in: NIPS'16: Proceedings of the 30th International Conference on Neural Information Processing, 2016, pp. 82–90.

[11] M. Simonovsky, N. Komodakis, Dynamic edge-conditioned filters in convolutional neural networks on graphs, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 29–38. doi:10.1109/CVPR.2017.11.

[12] C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[13] 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, 2017, pp. 5099–5108.

[14] Y. Zhang, M. Rabbat, A graph-cnn for 3d point cloud classification, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 6279–6283.

[15] L. Yi, H. Su, X. Guo, L. Guibas, Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6584–6592.

[16] G. Qian, A. Abualshour, G. Li, A. Thabet, B. Ghanem, Pu-gcn: Point cloud upsampling using graph convolutional networks (2019). arXiv:1912.03264.

[17] L. Wang, Y. Huang, Y. Hou, S. Zhang, J. Shan, Graph attention convolution for point cloud semantic segmentation, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[18] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in neural information processing systems, 2016, pp. 3844–3852.

[19] F. R. Chung, F. C. Graham, Spectral graph theory, no. 92, American Mathematical Soc., 1997.

[20] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Transactions on Neural Networks 20 (1) (2008) 61–80.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, The journal of machine learning research 15 (1) (2014) 1929–1958.

[22] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, arXiv preprint arXiv:1502.03167.

[23] Y. Eldar, M. Lindenbaum, M. Porat, Y. Y. Zeevi, The farthest point strategy for progressive image sampling, IEEE Transactions on Image Processing 6 (9) (1997) 1305–1315. doi:10.1109/83.623193.

[24] S. A. Dudani, The distance-weighted k-nearest-neighbor rule, IEEE Transactions on Systems, Man, and Cybernetics (4) (1976) 325–327.

[25] I. S. Dhillon, Y. Guan, B. Kulis, Weighted graph cuts without eigenvectors a multilevel approach, IEEE transactions on pattern analysis and machine intelligence 29 (11) (2007) 1944–1957.

[26] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, J. Xiao, 3d shapenets: A deep representation for volumetric shapes,

in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1912–1920. doi:10.1109/CVPR.2015.7298801.

[27] G. Lemaitre, F. Nogueira, C. K. Aridas, Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning, Journal of Machine Learning Research 18 (2017) 1 – 5.
URL https://hal.inria.fr/hal-01516244

[28] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, J. M. Solomon, Dynamic graph cnn for learning on point clouds, ACM Transactions on Graphics 38 (2019) 146:1–146:12.

[29] L. van der Maaten, G. Hinton, Visualizing high-dimensional data using t-sne, Journal of Machine Learning Research 9 (nov) (2008) 2579–2605, pagination: 27.

[30] A. T. B. G. Guohao Li, Matthias Muller, Deepgcns: Can gcns go as deep as cnns?, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 9267–9276.