
Article:
Zhang, W., Su, S., Wang, B. et al. (2 more authors) (2020) Local k-NNs pattern in omni-direction graph convolution neural network for 3D point clouds. Neurocomputing, 413. pp. 487-498. ISSN 0925-2312

https://doi.org/10.1016/j.neucom.2020.06.095

Local k-NNs Pattern in Omni-Direction Graph Convolution Neural Network for 3D Point Clouds

Wenjing Zhang[a], Songzhi Su[*,a], Beizhan Wang[a], Qingqi Hong[a] and Li Sun[b]

[a]School of Informatics, Xiamen University, 361005, China
[b]Department of Computer Science, The University of Sheffield, S1 4DP, UK

ARTICLE INFO

Keywords:
3D point cloud
Spatial layout
Omni-Directional k-NNs pattern
Graph convolution neural network

ABSTRACT

Effective representation of objects in irregular and unordered point clouds is one of the core challenges in 3D vision. Transforming point cloud into regular structures, such as 2D images and 3D voxels, are not ideal. It either obscures the inherent geometry information of 3D data or results in high computational complexity. Learning permutation invariance feature directly from raw 3D point clouds using deep neural network is a trend, such as PointNet and its variants, which are effective and computationally efficient. However, these methods are weak to reveal the spatial structure of 3D point clouds. Our method is delicately designed to capture both global and local spatial layout of point cloud by proposing a Local k-NNs Pattern in Omni-Direction Graph Convolution Neural Network architecture, called LKPO-GNN. Our method converts the unordered 3D point cloud into an ordered 1D sequence, to facilitate feeding the raw data into neural networks and simultaneously reducing the computational complexity. LKPO-GNN selects multi-directional k-NNs to form the local topological structure of a centroid, which describes local shapes in the point cloud. Afterwards, GNN is used to combine the local spatial structures and represent the unordered point clouds as a global graph. Experiments on ModelNet40, ShapeNetPart, ScanNet, and S3DIS datasets demonstrate that our proposed method outperforms most existing methods, which verifies the effectiveness and advantage of our work. Additionally, a deep analysis towards illustrating the rationality of our approach, in terms of the learned the topological structure feature, is provided. Source code is available at https://github.com/zwj12377/LKPO-GNN.git

1. Introduction

3D point cloud has a broad range of emerging applications, such as autonomous driving[1], 3D reconstruction, urban planning[2], etc. Learning and representation of point clouds is the core research problem in semantic-related tasks e.g. object classification[3, 4], object detection[5] and instance segmentation [6, 7, 8]. One natural and naive representation of point clouds is set, which is unordered, irregular, and sparse, making its understanding difficult. This paper explores graph-based deep learning architecture for effective reasoning of 3D point cloud.

Deep learning has dominated most of computer vision tasks in recent years. A proven track record can be found that deep learning methods can effectively analyze the large-scale and high-dimensional regular or Euclidean data[9]. In particular, Convolutional Neural Networks (CNNs) is able to extract meaningful statistical patterns from big regular grid structure data, such as images, audios and videos. Unfortunately, point clouds are irregular and unordered, thus making it difficult to be directly processed using conventional CNNs. As a result, most existing methods convert 3D point cloud to 2D images using casting and rendering [10] or voxel representation by down-sampling [11, 12, 13]. Though all the methods above have achieved good performance, there are still problems have not been solved. Firstly, rendering 3D point cloud or shape into 2D images will miss out the 3D geometry information. Secondly, applying 3D CNNs on volumetric representation is constrained by its high computational complexity. Finally, these operations are still based on sparse volumes, and it is challenging to deal with large-scale problems.

Recently, the Graph Neural Network (GNN) has become increasingly popular in many areas [14] and achieves state-of-the-art performance in modelling node relationships. Graph structure elegantly represents the feature information and spatial layout of the points simultaneously. In order to combine the local and global features of 3D point cloud, this paper proposes local k-Nearest Neighbors (k-NNs) pattern in Omni-Direction for point cloud object classification and segmentation with graph CNN, referred as LKPO-GNN.

*Corresponding author
ssz@xmu.edu.cn (S. Su)
ORCID(s):

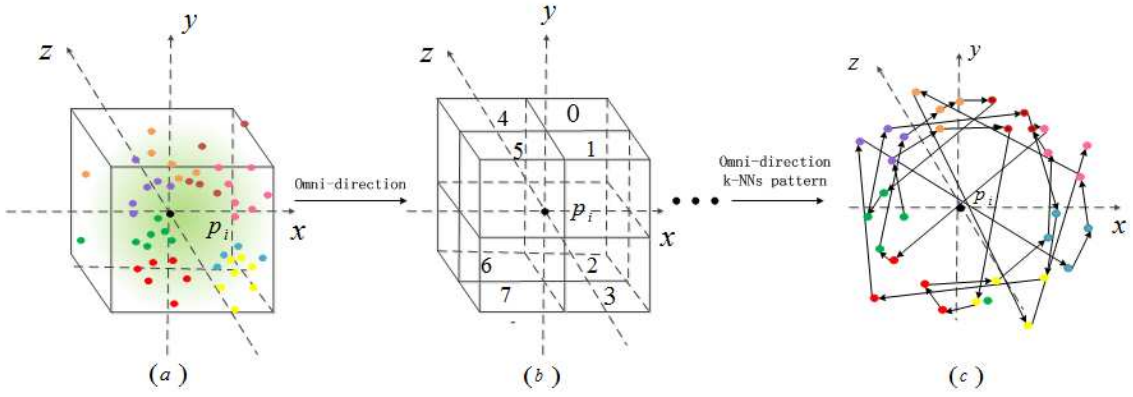


Figure 1: (*a*) The 3D Space of the Center Point p_i; (*b*) Omni-Direction of Center Point p_i; (*c*) Omni-Directional k-NNs Patten of Center Point p_i.

Given a centroid p_i of a set of 3D points (the black point in Figure 1(a)), the local region of point p_i is a sphere with radius R. We first partition the sphere into eight octants (sub-regions), called Omni-Direction, as shown in Figure 1(b). And then we find $k/8$ neighboring points of p_i according to the coordinate in each octant. The obtained k points are sorted by the distances to p_i from small to large, as shown in Figure 1(c)). Finally, The unordered points are transformed into an ordered 1D sequence, which usually maintains the original spatial layout. The details will be presented in Section 3. k-NN is a traditional non-parametric density estimation method, and we find this simple and naive method is suitable for solving the spatial structure problem well. Since we choose the k-NN points in different orientations, so we name our proposed representation "Local K-NNs Pattern in Omni-Direction" (LKPO).

Stacking multiple convolution layers can capture global information to some extend. In this paper, we use GNN to propagate the local information represented by LKPO.

Three main contributions of our paper are: 1) We proposed a novel architecture, i.e. using k-NN points in Omni-Direction, to represent and learn enriched local shape information of point clouds; 2) We proposed a method to convert the 3D unordered points into an ordered 1d sequence, to generate unified data for learning and also reduce the computational complexity; 3) GNN is employed to propagate the certain local geometric structure modelled by our proposed LKPO to obtain the global spatial layout.

2. Related work

2.1. Deep Learning on ordered Sets

Deep Neural Networks (DNNs) have been widely used in computer vision and natural language processing, and obtained promising performance. DNNs can learn significant statistical patterns from large-scale data in 3D Euclidean space. These learned patterns are known as feature representations. Extending DNNs to 3D data is challenging due to the fact that volumetric representations are spatial sparseness and computation complexity. Spectral methods are proposed by FPNN[11] and Vote3D[12] to solve the sparsity problem, but these methods are difficult to handle very large-scale 3D point cloud scenes. Multi-view CNNs [12, 13, 10] have tried to convert 3D point cloud or shapes into 2D images, then use 2D CNN to solve classification and instance segmentation problems. These methods have achieved good performance on 3D shape classification and retrieval tasks, however, it is nontrivial to extend them to point cloud classification and shape completion. Later, Riegler et al.[15] proposed OctNet to address these issues. OctNet represents point clouds with a hybrid of shallow grid octrees (depth = 3). Compared to its dense peers, OctNet reduces the computational and memory costs to a large degree, and is applicable to high-resolution inputs up to 256×256×256. Whereas OctNet must process point clouds as regular 3D volumes due to its 3D-CNN kernels. There is no such constraint in our proposed Local k-NNs Pattern in Omni-Direction with Graph CNN.

2.2. Deep Learning on Unordered Sets

Charles R. Qi [16] designed a deep learning framework that can directly consume unordered point sets as inputs, then use a simple symmetric function to aggregate the information from each point. However, PointNet[16] doesn't

capture local structures induced by the metric space points live in, limiting its ability to recognize fine-grained patterns and generalizability to complex scenes. In order to solve this problem, PointNet++ [17] was proposed and used a hierarchical neural network that applied PointNet recursively on a nested partitioning of the input point set. By exploiting metric space distance, this network is able to learn local feature with increasing contextual scales. However, it lacks sufficient ability to reveal the spatial structural characteristics of 3D point cloud. Subsequently, the SO-Net[18] models the spatial distribution of point cloud by building the irregular point cloud into an $m \times m$ 2D rectangular map. and performs hierarchical feature extraction on individual points and SOM nodes, to obtain the feature vector for the map. KCNet[19] also based on PointNet and introduced a point-set template to learn geometric correlations of points in 3D point clouds. Similarly, Li et al proposed PointCNN [20] to learn an x-transformation from the input points to attack the desertion of shape information and variance to point ordering of 3D point cloud. Jiang et al. [21] designed a PointSIFT module adaptive to scale of shape, encoding information of different orientations. These methods are related to our work in terms of directly using the coordinates of points as input. Here, for fairness, we remove the normal information of point cloud in experiment based on PointCNN.

2.3. Deep Learning using Graph Neural Networks

GNN-based methods can be grouped into spectral[22, 23, 24] and spatial [14] approaches. These methods have different mechanisms in modelling the correspondence between filter weights and nodes in local graph neighborhoods.

Point cloud can be formulated as a graph structure [14], and applying DNN on the graph is a trend. Currently, most Graph Neural Network (GNN) follow a pattern of cyclic recursive neighborhood aggregation in which each point aggregates the eigenvectors of its neighbors to compute its new eigenvector. After the k-round aggregation iteration, the point is represented by its transformed feature vector, which captures the structural information of the k-hop network neighboring points of the point. Then, you can use pooling to get a representation of the entire graph structure, such as summing the eigenvectors of all the points in the graph. Many GNN variants based on different neighborhood aggregation and graph-level pooling schemes have been proposed by many scholars[25, 26, 27, 15, 28, 29]. As a rule of thumb, these GNNs have achieved optimal performance in many tasks, such as point classification, link prediction, and graph classification. In regards to processing point clouds with deep learning using tree structures, Klokov et al. [14] built a kd-tree for the input point cloud, followed by hierarchical feature extractions from the leaves to root. In our proposed method, instead of the randomization of the tree construction as [14], we use deterministic structured geometric relationships between the points, obtaining richer structure information of 3D point clouds and also reducing the computation cost.

2.4. A Critical Analysis

Deep learning using ordered sets usually converts 3D point clouds to 2D images or 3D volumetric grids. However, 3D convolution and rendering 2D images are often time- and memory- consuming. Compared to using ordered sets, deep learning using unordered sets directly takes raw point clouds as the input without converting them to other formats. In these approaches, most of them only use neighborhood information to capture the local information. Compared to GNNs applied to 3D point objects, learning on unordered sets is unlikely to capture the rich local and global geometrical layout in a 3D point cloud. The input of GNNs are unordered sets, but the difference is that these points are input to the neural network in the form of a graph. Deep learning using GNNs can learn the rich structural information of a centroid through its k-top connected neighboring points in the graph, therefore, achieving better representation of spatial features in the topological graph.

3. Method

In the pipeline of 3D point clouds understanding, the key problem is the extraction of the topological features of the 3D point cloud and its representation in the computer. Topology refers to the structure of its shape or the spatial relationship of points, as shown in Figure 2. Although the point cloud of chair is unordered and irregular, each point has its own characteristic information. Additionally, it has certain local structural information between it and the surrounding points. Obviously, the characteristics of these points and their surrounding points are mostly consistent in Omni-Direction. Thus, in the point cloud classification and instance segmentation tasks, both global and local features are essential in an effective feature representation[14, 22]. This phenomenon indicates that the closer the point is to the centered point, the more similar the feature representation is. This topological relationship between points in space is extremely important in the study of point clouds. If this factor is ignored, directly convolving against feature associated

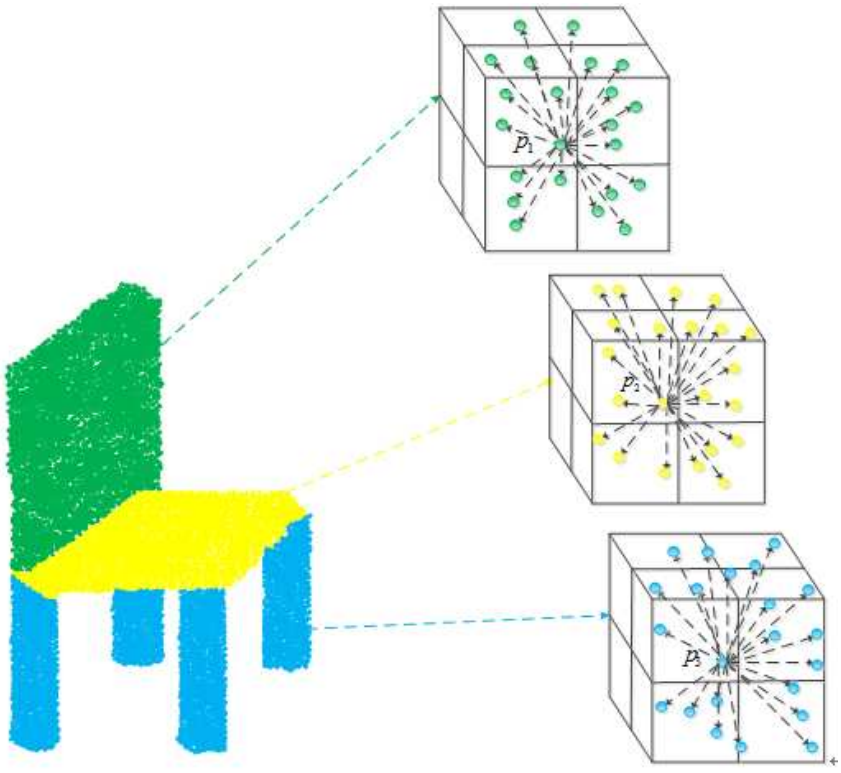


Figure 2: A Point Cloud of Chair. The chair can be divided into three sub-portions, which are indicated by green, yellow and blue, respectively. For example, points p_1, p_2, p_3 represent random point of three sub-portions of chair. Here, neighbor $(p_1) = \{p_{1i} \mid dist(p_1, p_{1j}) < d, p_{1j} \in \{p_{1t}\}_{t=1}^{k}, d \in R\}$ in Omni-Direction. Following this way, the neighbor point p_1 can be expressed as $(p_2) = \{p_{2j} \mid dist(p_2, p_{2j}) < d, p_{2j} \in \{p_{2t}\}_{t=1}^{k}, d \in R\}$ and neighbor $(p_3) = \{p_{3j} \mid dist(p_3, p_{3j}) < d, p_{3j} \in \{p_{3t}\}_{t=1}^{k}, d \in R\}$. Obviously, it is observed that points p_1,p_2,p_3 and their neighbor points are mostly consistent characteristics.

with the point and its neighbor points will result in desertion shape information and variance to point structural. To address these problems, we propose to design a novel deep Omni-Direction with LPKO-GNN architecture for point clouds.

Here we presents the details of our method by a graph representation of 3D point clouds that propagates topological features between k-NNs points in Omni-Direction. A point cloud is represented as a set of 3D points, $\{p_i | i = 1, 2, 3, \ldots, N\}$, where each point p_i is a vector of its (x, y, z) coordinate plus extra feature channels such as color, normal etc. Usually, more channel features result in better classification performance. In order to demonstrate modeling ability of geometric structure of points, we only use the (x, y, z) coordinates as input. Inspired by PointSIFT [21], we represent the shape pattern centered at a point by k neighborhoods in Omni-Direction to obtain the spatial structure between points and its neighbors. Given the input point cloud $P = \{p_1, p_2, ..., p_N\}$, we choose a subset $\tilde{P}$ that contains m points to define the local regions of centroid by iterative Farthest Point Sampling(FPS)[30], where $2 < m <= N$. Then the Ball Query[17] is used for finding all points that are within a radius R to the centroids. Next, similar to PointNet[16], we use Multi-Layer Perceptron(MLP) and global pooling layer to obtain the rich local feature vector of m centroids, respectively. Here, these local feature vector of m centroids carry rich local information.

3.1. Farthest Point Sampling Method

Farthest point sampling(FPS) iteratively get the next sampled point in the middle of the least-known sampling domain. The FPS method was first introduced for generic graph clustering algorithm[30], and then applied to 2D images [31] and further extended to 3D point cloud [17]. The FPS method has been widely used for a variety of isometry-invariant surface processing tasks[32]. For example, [33] explores this sampling strategy to efficiently esti-

mate geodesic distances. [34] and [35] use FPS for the context representation of shape recognition. Given an point set $\xi = \{p_i | i = 1, 2, 3, \dots, n\}$, $p_i \in R^d$, we can use FPS to choose a subset of points $\left\{p_{i_1}, p_{i_2}, p_{i_3}, \dots, p_{i_n}\right\}$, such that p_{i_j} is the most distant point (in euclidean distance metric) from the set $\left\{p_{i_1}, p_{i_2}, p_{i_3}, \dots, p_{i_{j-1}}\right\}$ with regard to the rest points. In general, the most widely-used subsampling method is random subsampling. However, it is difficult to cover the entire point set. Compared to random sampling, the advantage of FPS is that a better convergence can be achieved over the whole point set. The drawback of this method is that the iterative subsampling relies on previously selected points. In [17], the robustness of FPS method to this randomness is investigated, and the results show a good robustness to random sampling. Hence, we use FPS to sample 3D point clouds.

3.2. Ball Query Method

Ball Query[17] finds all points that are within the radius R to the query point. For example, given a set of 3D points ξ, we select a subset $\vartheta = \{p_j | j = 1, 2, 3, \dots, m\}$, $m < N$ and $p_j \in R^d$ using FPS as the centroids for a point cloud. Here, we set I as the number of points to be obtained by the Ball Query. For each centroid p_j in subset ϑ, we compute the distance(in Euclidean distance) from each p_i in ξ to it. When the distance of p_i to p_j is smaller than R, point p_i is within in the radius R with respect to the center point p_j. This process will be repeated until the number of points is I. Comparing with k-NN, the local neighborhood developed by Ball Query method can generate 3D regions with a fixed scale to learn more generalizable local features across space, which is extremely important for recognition tasks using learned local features. When dealing with non-uniform point set, the generalization ability of feature learning using k-NN cannot be guaranteed. It is worth noting that Ball Query usually takes slightly more time than k-NN methods.

3.3. Local k-NNs Pattern in Omni-Direction

Given a center point $p_i(p_{ix}, p_{iy}, p_{iz}) \in \tilde{P}$, the 3D point cloud space ψ_i centered at p_i is partitioned into eight octants, indicating different orientations in Euclidean space. These different orientations are defined as Omni-Direction. The k neighbors in Omni-Direction of p_i is referred as its local Omni-Directional k-NNs pattern. The distance D_{ji} from the center point p_j to p_i can be represented as follows according to euclidean distance metric:

$$D_{ji} = \sqrt{\left(p_{jx} - p_{ix}\right)^2 + \left(p_{jy} - p_{iy}\right)^2 + \left(p_{jz} - p_{iz}\right)^2} \tag{1}$$

Here, $p_i, p_j \in \tilde{P}$, and $j \neq i$. If $D_{ji} \leq R$, the coordinate relationship between point p_j and point p_i can be expressed as:

$$int\tilde{x} = \left(p_{jx} > p_{ix}\right) \tag{2}$$

$$int\tilde{y} = \left(p_{jy} > p_{iy}\right) \tag{3}$$

$$int\tilde{z} = \left(p_{jz} > p_{iz}\right) \tag{4}$$

Here $\tilde{x}$, $\tilde{y}$ and $\tilde{z}$ are binary values (0 or 1). For example, when $p_{jx} > p_{ix}$, $\tilde{x} = 1$, otherwise $\tilde{x} = 0$. There are $8 = 2^3$ permutations of a set of three Integer values $\{\tilde{x}, \tilde{y}, \tilde{z}\}$. Next the number index id of points in local Omni-Directional k-NNs pattern can be obtained:

$$id = \tilde{x} * (k/2) + \tilde{y} * (k/4) + \tilde{z} * (k/8) \tag{5}$$

Here, $k = 8v, v \in R+$. If $v = 1$, $k = 8$, and the space ψ_i is partitioned into 8 octants. The details of id index are:

$$\{\tilde{x}, \tilde{y}, \tilde{z}\} = \{0, 0, 0\}; id = 0 * (k/2) + 0 * (k/4) + 0 * (k/8) = 0 \tag{6}$$

$$\{\tilde{x}, \tilde{y}, \tilde{z}\} = \{0, 0, 1\}; id = 0 * (k/2) + 0 * (k/4) + 1 * (k/8) = k/8 = 1 \tag{7}$$

$$\{\tilde{x}, \tilde{y}, \tilde{z}\} = \{0, 1, 0\}; id = 0 * (k/2) + 1 * (k/4) + 0 * (k/8) = 2k/8 = 2 \tag{8}$$

$$\{\tilde{x}, \tilde{y}, \tilde{z}\} = \{0, 1, 1\}; id = 0 * (k/2) + 1 * (k/4) + 1 * (k/8) = 3k/8 = 3 \tag{9}$$

$$\{\tilde{x}, \tilde{y}, \tilde{z}\} = \{1, 0, 0\}; id = 1 * (k/2) + 0 * (k/4) + 0 * (k/8) = 4k/8 = 4 \tag{10}$$

$$\{\tilde{x}, \tilde{y}, \tilde{z}\} = \{1, 0, 1\}; id = 1 * (k/2) + 0 * (k/4) + 1 * (k/8) = 5k/8 = 5 \tag{11}$$

$$\{\tilde{x}, \tilde{y}, \tilde{z}\} = \{1, 1, 0\}; id = 1 * (k/2) + 1 * (k/4) + 0 * (k/8) = 6k/8 = 6 \tag{12}$$

$$\{\tilde{x}, \tilde{y}, \tilde{z}\} = \{1, 1, 1\}; id = 1 * (k/2) + 1 * (k/4) + 1 * (k/8) = 7k/8 = 7 \tag{13}$$

If $v > 1(k > 8)$, all of the possibilities for the value of the index number of the space ψ_i are k, which can be expressed as $\{0, 1, 2, ..., k-1\}$. The possibilities for the value of set $\{\tilde{x}, \tilde{y}, \tilde{z}\}$ are $8 = 2^3$. The possibilities of the rest index number for ψ_i are $k-8$, which can be computed by adding the id obtained from each set $\{\tilde{x}, \tilde{y}, \tilde{z}\}$ to the each value in the set $[0, k/8)$. The details of id index for the space ψ_i can be expressed by (5) as follows:

$$\tilde{x}, \tilde{y}, \tilde{z} = \{0, 0, 0\}; id = 0 \tag{14}$$

$$id + 1 = 0 \tag{15}$$

$$id + 2 = 0 \tag{16}$$

$$\vdots$$

$$id + (k/8) - 1 = 0 \tag{17}$$

$$\tilde{x}, \tilde{y}, \tilde{z} = \{0, 0, 1\}; id = k/8 \tag{18}$$

$$id + 1 = k/8 \tag{19}$$

$$\vdots$$

$$id + (k/8) - 1 = k/8 \tag{20}$$

$$\tilde{x}, \tilde{y}, \tilde{z} = \{0, 1, 0\}; id = k/4 \tag{21}$$

$$id + 1 = k/4 \tag{22}$$

$$\vdots$$

$$id + (k/8) - 1 = k/4 \tag{23}$$

$$\vdots$$

$$\tilde{x}, \tilde{y}, \tilde{z} = \{1, 1, 1\}, id = 7k/8 \tag{24}$$

$$id + 1 = 7k/8 \tag{25}$$

$$\vdots$$

$$id + (k/8) - 1 = 7k/8 \tag{26}$$

Thus, by this operation, we can obtain the local k-NNs pattern of center point p_i in Omni-Direction. Next, we compute the distances D_{ji} for each centroid p_j to center point p_i. Then, all D_{ji} are sorted ascendingly, and the corresponding local feature vectors are selected in the Omni-Direction, where k-NNs pattern of a center point p_i is used as the feature for these k-NNs points. Finally, these feature vectors for k-NNs points are connected to represent the explicit structural information of the point p_i within a local area (i.e. its k-NNs points in Omni-Direction). Thus, we can obtain m graphs, and each graph m is pair (V, E) with $V = \{V_1, V_2, ..., V_k\}$ and the set of edges are $E \subseteq V_1 \times V_2, V_2 \times V_3, ..., V_{k-1} \times V_k$. Algorithm 9 gives the pseudocode for graph representation of a 3D Point Cloud.

In this way, the Omni-Direction with local k-NNs pattern graph of each central point would be obtained. Here, the center points and its local Omni-Directional k-NNs pattern maintain the initial spatial structure, which will served as feature for the subsequent task of point clouds classification and segmentation.

Three common tasks of point cloud are:

Task 1: Object classification. The input is the 3D point cloud directly sampled from the shape, the output is n scores for all n candidate classes.

Task 2: Object part segmentation. The input can be shape representation of point clouds, the output is a part category label for each point.

Task 3: semantic segmentation. The input is a 3D scan model represented by point clouds, the output is a semantic object class for each point.

In the following section, we present our architecture for 3D point clouds.

Algorithm 1 Graph Representation of a 3D Point Cloud.

Input: A point cloud P with N points, the number of center points m, k points in Omni-Direction

Output: m graphs and each graph with k vertices and $k-1$ edges

1: Downsample $\vartheta_{\mathrm{m}} = \left\{p_j | j = 1, 2, 3, \ldots, m\right\}$ from P using FPS;
2: **for** $i = 1; i <= m; i++$ **do**
3: Compute the rich local feature vector $f_i = \phi\left(\vartheta_i\right)$;
4: Compute local k-NNs pattern $g_i = \left\{s_t = \theta\left(\vartheta_i\right)\right\}_{t=1}^{k}$ in Omni-Direction;
5: Sort g_i according to the Euclidean distance between s_t to ϑ_i in ascending order;
6: Select f_i in g_i corresponding to the order;
7: Connect points in f_i in this order to represent the explicit structural information of the point ϑ_i and its k-NNs points in Omni-Direction;
8: **end for**
9: **Return** m graphs and each graph with k vertices and $k-1$ edges

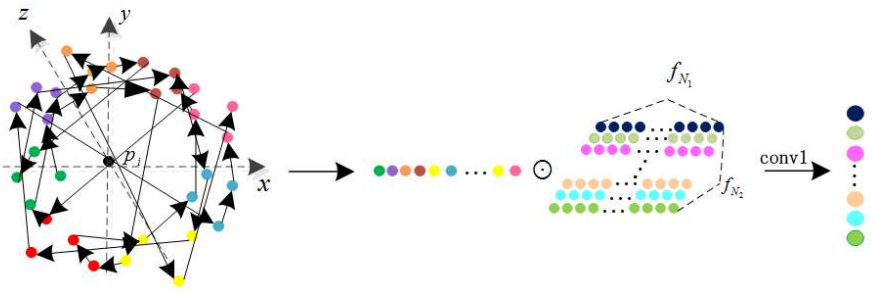


Figure 3: This figure illustrates the convolution operation over channels. Here, given a graph G_i constructed by Omni-Directional k-NNs pattern of the center point p_i, and a set of fixed-size convolution filters, an example of the proposed convolution is conducted. The f_{N_2} is the number of filters and f_{N_1} is the number of filter elements. Finally, a deterministic non-linear feature transform, that implicitly represents the structured geometrical shape of a point p_i within an area of k-NNs pattern in Omni-Direction, can be learned by back-propagation through time.

3.4. Local k-NNs Pattern in Omni-Direction GNN

For m graphs using method described in Section 3.3, each graph $G_i, i \in \{1, 2, ..., m\}$ consists of a set of vertices and a set of edges (V, E) i.e the vertices $V = \{V_1, V_2, ..., V_k\}$ and the edge $E \subseteq V_1 \times V_2, V_2 \times V_3, ..., V_{k-1} \times V_k$. The details of convolution are shown as Figure 3.

$$V_{G_i} = \{p_{i1}, p_{i2}, ..., p_{ik}\} \tag{27}$$

In the Figure 3, V_{G_i} means the vertices of the graph G_i constructed by the local Omni-Directional k-NNs pattern of center point p_i, which contains the rich geometric information. Next, we transform these vertices into an order sequence(see Figure 3), which carriers certain origin geometric structure. Finally, we design a set of fixed-size convolution filters(as described in Figure 3). The feature representation of $X_{G_i}^t$ for graph G_i can be obtained by (28).

$$X_{G_i}^t = \sigma\left(\sum_{f=1}^{F}\left(\sum_{h=1}^{k} W_h^{f,t} \odot Y_{G_i}^f\right) + B^t\right) \tag{28}$$

Here, σ represents the activation function, and $Y_{G_i}^f = V_{G_i}$. t means tth feature for G_i, and f represent the number of filters. $\odot$ is the element-wise multiplication. B^t is the tth bias. After we obtain $X_{G_i}^t$, the center points p_i and its k-NNs in Omni-Direction maintain the initial spatial structure would be obtained. The idea is to apply convolution filters which slide over the vertices of graph G_i to extract local structure features within the graph G_i in a manner analogous to the standard convolution operation on grid 2D data. In particular, our LKPO-GNN can map graphs to point cloud space to effectively preserve local spatial structures information in the original space. Additionally, our

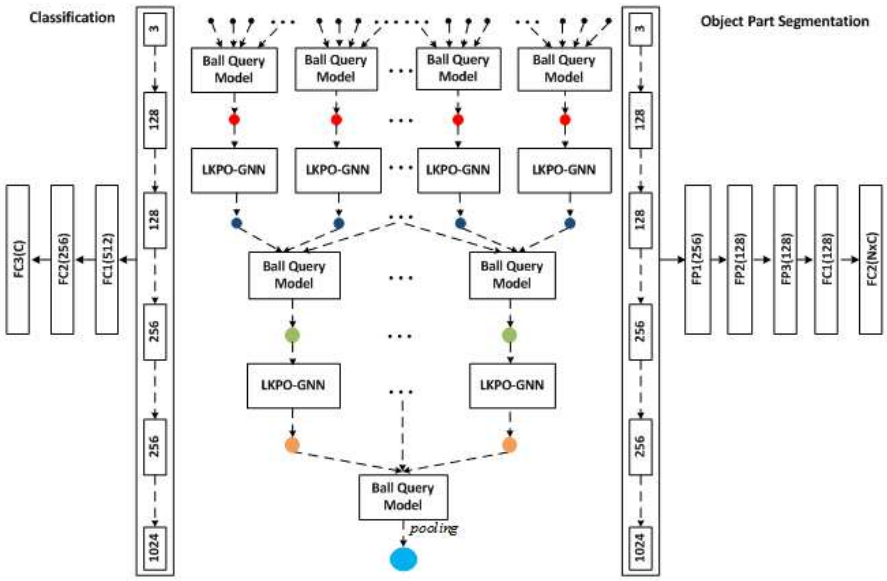


Figure 4: The architecture of the proposed LKPO-GNN for classification and object part segmentation. In this Figure, the networks take n points as the input. Firstly, Ball Query Module is to extract the rich feature information of the center points, and then use LKPO-GNN to get the deep structural features between k-NNs points in Omni-Direction by a graph representation of 3D point clouds. After several steps of processing by Ball Query Modules and LKPO-GNN Modules, point features are aggregated by max pooling. For classification, a triple-layer perceptron i.e. FC1(512), FC2(256) and FC3(C) is employed using feature with rich structure information (shown in blue color at the bottom). The feature is reused and connected with fully-connected layers i.e. FP1(256),FP2(128),and FP3(128), and after that, another two fully-connected layers i.e. FC1(128) and FC2(NxC) are used for the object part segmentation.

method converts unordered intersections between points of 3D point cloud into an ordered 1D sequence, making it easier to combine with neural networks and also reduce the computational complexity.

3.5. Our Proposed Network Architecture

For unstructured data, kd-tree[14] is an efficient method to store the point neighborhood relationship. The tree structure of points not only provides guidelines for neural network architectures that can be used to process the 3D point cloud, but also possesses the much desired attributes of permutation and translation invariance for neural networks. In this section, we present our architecture for 3D point cloud classification and segmentation. Our approach uses similar techniques with convolutions over graph, in particular our method uses Omni-Directional deterministic structured geometric relationships between the points. Our core module is LKPO-GNN, which propagates topological features between k-NNs points in Omni-Direction by a graph representation of 3D point clouds. In our network, all convolution layers before the last layer are followed by batch normalization and ReLU activations, implemented with CUDA for speed-up in our experiments. Especially important is that the global and local spatial layout of 3D point cloud can be extracted by LKPO-GNN Architecture.

3.5.1. Classification and Object Part Segmentation Network

Ball Query and feature propagation(FP)[17] are two core modules of our proposed network. FP was used to propagate features from subsampled points to the original points. The input of our network is the 3D coordinates of $n \times 3$ points. Firstly, we can use Ball Query Module to obtain the center points with the rich information of point cloud. The number of center points on Ball Query Modules are $n = 512$,$n = 128$, and radius $r_1 = 0.2, r_2 = 0.4$. Secondly, we use LKPO-GNN to propagate the rich topological features between k-NNs points in Omni-Direction by a graph representation of 3D point clouds. The number of points on LKPO-GNN modules are $n = 512$,$n = 128$, and $r_1 = 0.2, r_2 = 0.4$ respectively. After several steps of processing by Ball Query Modules and LKPO-GNN modules, the center point in 3D point cloud scene becomes less and less, and the feature information of the center point aggregation become richer. These feature information aggregate rich global spatial structure features. Then, we use FP method to obtain

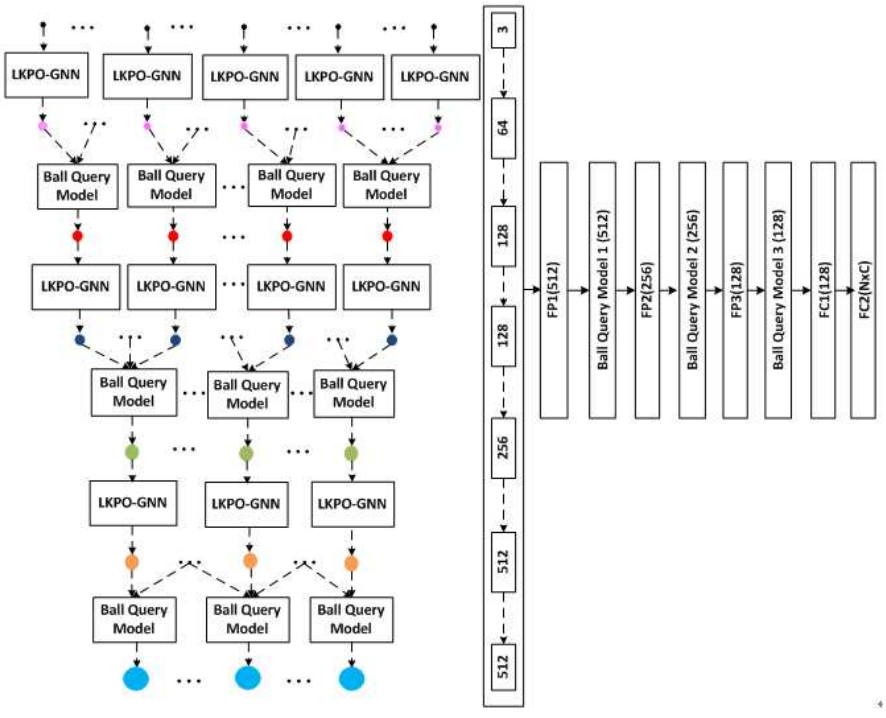


Figure 5: Illustration of LKPO-GNN Architecture for Semantic Segmentation. For semantic segmentation, the network takes n points as input. Firstly, LKPO-GNN is applied to get deep structural features between k-NNs points in Omni-Direction using a graph representation, and then Ball Query Module is employed to extract the rich feature information of the points. After several steps of applying LKPO-GNN modules and Ball Query modules alternately, point features can be aggregated by Ball Query module. Features at the bottom (blue) are of enriched structural information and back propagate features to the original set through a number of layers points by FP1(512), LKPO-GNN(512), FP2(256), LKPO-GNN(256), FP3(128), and LKPO-GNN(128) moduels. And then a couple-layer architecture, i.e. FC1(128) and FC2(NxC), are used for the semantic segmentation.

the original set of points. Finally, the fully-connected layers are used for the classification and object part segmentation tasks. Figure 4 shows the architecture for classification and object part segmentation for 3D point clouds.

3.5.2. Semantic Segmentation Network

Each shape is represented with $n \times 3$ points in the semantic segmentation task. We use the LKPO-GNN modules and Ball Query Modules iteratively to obtain the structural information of the center points. The number of points on LKPO-GNN modules are $n = 8192$,$n = 1024$,$n = 512$, respectively. The number of center points on Ball Query Modules are $n = 1024$,$n = 256$,$n = 64$. Next, the FP module is used to perform upsampling operations. Through the FP module, the point cloud scene is restored. The saved scene contains richer feature information, which would contributes to the subsequent segmentation processing. After upsampling, LKPO-GNN module is used to acquire the structural information of the sampling points and its its k-NNs in Omni-Direction. Finally, fully-connected layers are used to perform semantic segmentation of the 3D point cloud. The center point number of the three steps are $n = 256$,$n = 1024$,$n = 8192$ respectively. Comparing with [21], our module converts the unordered 3D point cloud into an ordered 1D sequence, making it easier to combine with neural networks and also significantly reduce the computational complexity. Here, thanks to [21], the contribution made in attacking the multi-direction information of 3D point cloud. That gives our method much inspiration in considering the local k-NNs pattern in Omni-Direction information of center points in 3D point cloud. In Figure 5, shows the architecture for segmentation for 3D point cloud.

3.6. Comparison to existing methods

In this subsection, we compare our LKPO-GNN with existing methods to highlight the differences. In the recent years, several methods [14, 17, 19, 20] have been proposed to model the structural information between points in 3D point clouds. All of these methods relate to our work in terms of directly accepting the spatial coordinates of points as input. There are three main differences of our LKPO-GNN between these methods. Firstly, in our method, the vertices of graph, made of Omni-Directional k-NNs pattern , are learned from the feature vectors of these points, rather than use the point coordinates in [14, 19]. In this way, each vertex in the graph carried the rich local geometrical information, which will be conducive to understanding the 3D point cloud scene. Secondly, the vertices of the graph in our method were gotten from different orientations of center point, such that the representation ability of the graph, which is one of key components in understanding 3D objects, will be better. However, in [14, 17, 19, 20], the local features represented by the points are mostly derived from the measurement of Euclidean distance without considering the direction for each point. Additionally, [14] built a kd-tree for the input point cloud, followed by hierarchical feature extractions from the leaves to root. This causes its performance to be heavily relied on the randomization of the tree construction. Different from [14], the vertices of the graph in our method considered deterministic structured geometric relationships. Thus, the performance of our method can avoid depending on the graph construction. Finally, since the graph in our method is built on a certain geometric structure, the sequential convolution kernel can be designed to convert the unordered intersections between points in 3D point cloud into an ordered 1D sequence, making it easier to combine with neural networks and also significantly reduce the computational complexity.

4. Experimental Results and Analysis

4.1. Dataset and Performance Evaluation

Four commonly used point cloud dataset are used in our experiments, ModelNet40[36], ScanNet[37], ShapeNetPart[38], and S3DIS[39]. We performed object classification on the ModelNet40, and semantic part segmentation on the ShapeNetPart dataset. Next, we completed the semantic segmentation based on ScanNet and S3DIS datasets. Object classification performance is evaluated by Average Class Accuracy (ACA%) and Overall Accuracy (OA%). Semantic segmentation of scene labeling is evaluated by per-voxel Accuracy(Accuracy%)[37], (OA%) and Mean IOU(MIOU%)[12] metrics.

In object classification, the ACA calculates the proportion of pixels that are correctly classified within each category, and then take averages over all categories. OA calculates the ratio between the model and the total number of predictions on all test sets. In semantic segmentation, our goal is to predict the semantic object label on per-voxel basis, OA and MIOU. The per-voxel Accuracy is based on voxel accuracy, which is similar to[37]. MIOU is mean IOU for all shapes from test dataset. IOU is a standard measure of semantic segmentation[40, 41, 42]. It calculates the intersection of the two sets and the union of the two sets, which are ground truth labels and predicted label respectively. For each shape m of category k, in order to calculate the shape's MIOU, firstly, for every part of category k, we should compute its IOU between ground truth and prediction. Then we take average of IOUs for all shapes in its category. We list below the experiment setting for each dataset as Table 1. Figure 6 shows part of the samples from four datasets.

4.2. 3D Object Classification on ModelNet40

In this section, we evaluate our proposed method on classifying 3D point clouds uniformly sampled from 3D (ModelNet40) Euclidean spaces. 3D point clouds are sampled from mesh surfaces from ModelNet 40 shapes. 1024 points are used for ModelNet40 defautly. During training we augment 3D point cloud on-the-fly by randomly rotating the object along the up-axis and jitter the position of each point by Gaussian noise with zero mean and 0.02 standard deviation following. We train the network for 250 epochs for LKPO-GNN module with batch size 16, base learning rate is set to 0.001. Cross-entropy loss is minimized during training. The ADAM solver is adopted to optimize the network on two GPU TITAN Xp. Momentum is set to 0.9. In Table 2, we compare our model with several neural network methods designed for 3D point cloud.

The result shows that our model achieves state-of-the-art performance. Here, when $k = 32$, our LKPO-GNN outperforms the previous state-of-the-art [20] by 1.2% ACA and 0.2% in OA metric. There still a small gap between our method and MVCNN[10], and we think that is because multi-view perception can capture more details of a 3D point cloud. However, it's nontrivial to extend them to scene understanding or other 3D tasks such as point classification and shape completion [46]. In addition, when k=8, the result of [20] is better than our method in OA metric. That is because it can learn x-transformation from the input points to promote the input features associated with the points and

Table 1
Introduction of the Datasets

Databases	Description	Total Scenarios	Training	Testing
ModelNet40	CAD models 40 categories.	12311	9843	2468
ShapeNetPart	shapes, 16 categories, 50 parts.	16881	14006	2874
ScanNet	scanned and reconstructed, indoor scenes.	1513	1201	312
S3DIS	3D scans in 6 areas with 271 rooms, 13 categories.	23585	20291	3294

the permutation of the points into a latent and potentially canonical order. It indicates that the spatially-local structural information of the points in the 3D point cloud has an extremely important role in the classification. Compared to our approach, [20] is unable to achieve permutation-invariance, which is desired for point clouds. The main reason why our method achieves better results is that, Omni-Directional structured geometric relationships (between the points) are used to build the graph, thereby possessing the much desired attributes of permutation and translation invariance for 3D point cloud.

4.3. Object Part Segmentation on ShapeNetPart

In ShapeNetPart dataset, following the settings in [38], we evaluate our method on object part segmentation assume that category label for each shape is already known. Each shape is represented by 3D point cloud with input size of 1024 points, the task is to predict a part label for each point. The dataset includes 16881 shapes from 16 categories, annotated with 50 parts in total. These categories are labeled with two to five parts.We train the network for 200 epochs for LKPO-GNN respectively and use dropout with keep ratio 0.5 on the fully connected layer. The decay rate for batch normalization starts with 0.7 and is gradually increased to 0.99. Cross-entropy loss is minimized during training. We use adam optimizer with initial learning rate 0.001, momentum 0.9 and batch size 32.

Compared with state-of-the-art algorithms, quantitative results are provided in Table 3. We can see that our LKPO-GNN performs better than other competitive algorithms in most classes. When $k = 32$, our model outperforms [20] by 0.3% accuracy in mean IOU. In particular, our method acquires considerable improvement in bag, car, ear phone, lamp, laptop, motor, mug, pistol and skateboard. These shapes are mostly rigid and often contain rich geometric information. Notice our method achieves better results than [14, 19, 18, 20], which have been proposed to reason about the structural information between points in 3D point clouds. However, in these methods, the local features represented by the points are mostly derived from the measurement of Euclidean distance without taking into accounting the multi-direction deterministic structured geometric relationships between the points in 3D point cloud, limiting their ability to represent the local geometrical information, which is one of the key components in classification and segmentation of 3D point cloud. Additionally, [20] achieves good results by learning an x -transformation from the input points to promote the input features associated with the points. However, it needs more than twice the number of parametric layers and times as required by our method $k = 8$ to achieve the reported performance. This is a direct consequence of effective exploration of multi-direction deterministic structured geometric by LKPO-GNN. This factor should be considered in the study of 3D point cloud. In the future study, the author will be working on this part.

Figure 6: Part of samples from four datasets. From top to bottom line, these pictures are sampled from ModelNet40, ShapeNetPart, ScanNet and S3DIS separately.

4.4. 3D Semantic Segmentation on ScanNet

We formulate semantic segmentation as a per-point classification problem. For semantic segmentation in ScanNet dataset, each shape is represented by a 3D point cloud with 8192 points, as in[15]. Semantic segmentation distinguishes points of different classes. Furthermore, semantic segmentation assigns the same label to points belonging to the same instance.We train the network for 1000 epochs for LKPO-GNN respectively and use dropout with keep ratio 0.5 on the fully connected layer. The decay rate for batch normalization starts with 0.7 and is gradually increased to 0.99. Cross entropy loss is minimized during training. Adam optimizer is used with initial learning rate 0.001, momentum 0.9 and batch size 32.

The input is the 3D coordinates of $N = 8192$ points. The number of center points on LKPO-GNN modules are $N = 8192$, $N = 1024$, $N = 512$, respectively. The number of center points on Ball Query Modules are $N = 1024$, $N = 256$, $N = 64$, and then the FP module is used to perform upsampling operations. After upsampling, LKPO-GNN module is used to acquire the structural information of the sampling points and its neighboring points in local Omni-Directional k-NNs pattern. For fair comparison, the input of point cloud removes the normal information. We only use the coordinate information as input. In Table 4 we present the Accuracy(%) and MIOU(%) scores. This experiment demonstrates the effectiveness of our approach on Semantic Segmentation for 3D point cloud. We compare our method with previous state-of-the-art method [21] in Table 4. Our method achieves the best performance an ScanNet benchmark, and it outperforms the previous state-of-the-art model [21] by 0.2 % accuracy. In particular, it improves previous PointNet++ by 1.5 in MIOU. This validates the effectiveness of the LKPO-GNN. Although [21] method is designed to describe eight crucial orientations, it only stack eight orientation-encoding units. Thus, the ability of a certain geometric structure for points in 3D point cloud will be limited. We argue that the performance improvements come from the deterministic structured geometric relationships in Omni-Direction between the points in a 3D point cloud. Additionally, our method achieves better or comparable accuracy and computes more efficiently than [20, 21] with fewer parameters.

4.5. 3D Semantic Segmentation on S3DIS

In this section, we experiment on S3DIS dataset. We split the dataset into training and testing following the way of[16]. Here, in order to more clearly represent the structural information of the point cloud, we only consider the

Table 2
3D Object Classification Results on ModelNet40

Methods	Input	ACA	OA
SPH [43]	mesh	68.2	-
3DShapesNets[38]	volume	77.3	84.7
VoxNet[44]	volume	83.0	85.9
Subvolume[13]	volume	86.0	89.2
LFD[36]	image	75.5	-
MVCNN[10]	image	**90.1**	-
OctNet[15]	point	83.8	86.5
ECC[45]	point	83.2	87.4
PointNet[16]	point	86.2	89.2
PointNet++[17]	point	86.5	89.8
Kd-Net[14]	point	86.3	90.6
SO-Net[18]	point	87.3	90.9
KCNet[19]	point	-	91.0
PointCNN[20]	point	87.7	91.2
LKPO-GNN(k=8)	point	88.2	90.9
LKPO-GNN(k=32)	point	88.9	**91.4**

coordinate information of the point. In S3DIS dataset, each shape is represented by a 3D point cloud with 4096 points. We train the network for 50 epochs for LKPO-GNN respectively and use dropout with keep ratio 0.5 on the fully connected layer. The decay rate for batch normalization starts with 0.5 and is gradually increased to 0.99. Cross entropy loss is minimized during training. We use adam optimizer with initial learning rate 0.001, momentum 0.9 and batch size 24.

Similarly, We only use the coordinate information as input. The input is the 3D coordinates of $n = 4096$ points. Then cross use the LKPO-GNN modules and Ball Query Modules to obtain multi-direction deterministic structured geometric relationships of the center points. The number of center points on LKPO-GNN modules are $N = 4096$, $N = 1024$, $N = 256$, respectively. The number of center points on Ball Query Modules are $N = 1024$, $N = 256$, $N = 64$. Finally, the FP module is used to perform upsampling operations. The point number in three steps are $N = 256$, $N = 1024$, $N = 4096$ respectively. After upsampling, LKPO-GNN module is used to acquire the deterministic structured geometric of the sampling points and its neighboring points in local Omni-Direction k-NNs pattern. The results of semantic segmentation for S3DIS is represented by Table 5. One can see that our method achieves the state-of-the-art in OA and MIOU. Our LKPO-GNN improves previous SPG method by 0.3 % in OA, and [21] method by 0.8 % in MIOU. The detailed percategory IOU results show that the LKPO-GNN is able to achieve better performances in door, table, sofa, and clutter. SPGraph offers a compact yet rich representation of contextual relationships between object parts and achieved good results. While the vertices of the graph in this method can not obtain the multi-direction of the point in 3D point cloud, which will limit its representation ability of the graph.

4.6. Space and Time Complexity

We further compare the space and time complexity with other methods. Here we choose our object part segmentation model to test the space and time complexity. Specifically, we record the forward times and modelsize of batch 1, 1024 points for these methods using TensorFlow on a Titan XP GPU. The forward times and modelsize are reported in Table 6. Table 6 shows that our model has the fastest forward time with acceptable model size compared to

Table 3
Object Part Segmentation Results on ShapeNetPart

Methods	MIOU	aero	bag	cap	car	chair	ear phone	guitar	knife	Lamp	Laptop	motor	mug	pistol	rocket	skate board	table
Wu[42]	-	63.2	-	-	-	73.5	-	-	-	74.4	-	-	-	-	-	-	74.8
3DCNN[38]	79.4	75.1	72.8	73.3	70.0	87.2	63.5	88.4	79.6	74.4	93.9	58.7	91.8	76.4	51.2	65.3	77.1
Yi[47]	81.4	81.0	78.4	77.7	75.7	87.6	61.9	92.0	85.4	82.5	95.7	70.6	91.9	**85.9**	53.1	69.8	75.3
Kd-net[14]	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	**87.2**	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
PointNet[16]	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
SSCNN[48]	84.7	81.6	81.7	81.9	75.2	90.2	74.9	**93.0**	86.1	84.7	95.6	66.7	92.7	81.6	**60.6**	**82.9**	82.1
KCNet[19]	84.7	82.8	81.5	86.4	77.6	90.3	76.8	91.0	**87.2**	84.5	95.5	69.2	94.4	81.6	60.1	75.2	81.3
SO-Net[18]	84.9	82.8	77.8	88.0	77.3	90.6	73.5	90.7	83.9	82.8	94.8	69.1	94.2	80.9	53.1	72.9	**83.0**
PointNet++ [17]	84.9	82.9	81.3	85.3	78.5	90.6	73.3	91.2	86.9	82.5	95.3	71.8	95.0	82.1	57.7	75.4	81.6
PointCNN [20]	85.3	82.3	79.9	**88.1**	78.1	**90.9**	74.6	91.0	86.7	83.2	95.6	71.3	95.4	81.7	57.2	73.4	81.6
LKPO-GNN(k=8)	85.3	82.5	**81.8**	87.7	**78.8**	90.7	75.4	90.8	87.1	83.5	95.6	**72.0**	**95.6**	81.7	55.9	75.8	82.8
LKPO-GNN(k=32)	**85.6**	82.6	80.8	86.9	78.6	**90.9**	**77.7**	90.8	86.9	**84.9**	**95.8**	71.7	94.6	82.4	56.1	76.0	82.8

Table 4
Semantic Segmentation on ScanNet

Methods	Accuracy	MIOU
3DCNN[38]	73.0	-
PointNet[16]	73.9	-
PointNet++[17]	84.0	56.9
PointCNN [20]	84.8	-
PointSIFT[21]	85.1	54.5
LKPO-GNN(k=8)	**85.3**	**58.4**

[16],[19],[18], [17] and [20] methods. Though [17] and [19] have less space consumption than our model, our method outperforms the [17] and [19] by 0.7 and 0.9 respectively in the mean IOU metric. Hence, our model is amenable to real-time object part segmentation tasks.

5. Conclusions

We proposed a deep neural architecture called LKPO-GNN. LKPO extracts local topological structure from 3D point clouds, and GNN propagates LKPO to obtain global information. Our proposed method can obtain deeper feature representation and then improve the classification and segmentation performance. We apply our LKPO-GNN on four datasets (ModelNet40, ShapeNetPart, ScanNet and S3DIS datasets), and perform on three point cloud-based applications (classification, object part segmentation, and semantic segmentation). The experimental results show the simplicity and effectiveness of our model. Future work will focus on: 1) investigating the possibility to optimise the architecture using Neural Architecture Search(NAS) technologies; 2) accelerating the neighbour searching procedure in Omni-Directional k-NNs pattern by Approximate Nearest Neighbor (ANN).

Table 5
Segmentation Results on S3DIS Dataset

Methods	OA	MIOU	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
PointNet[16]	80.6	54.8	87.9	97.1	65.1	50.0	41.1	63.9	47.4	63.9	65.4	29.9	44.8	9.6	46.4
RSNet[7]	-	51.9	**93.3**	**98.3**	**79.1**	0.00	15.7	45.4	50.1	65.5	67.9	22.5	52.5	41.0	43.6
SPGraph[49]	85.5	62.1	89.9	95.1	76.4	62.8	**47.1**	55.3	68.4	**73.5**	69.2	63.2	45.9	8.7	52.9
PointCNN[20]	-	62.7	85.6	85.2	77.1	**63.8**	34.8	56.1	69.3	60.8	71.2	**64.3**	43.2	47.9	56.3
PointSIFT[21]	83.6	63.8	86.5	86.3	71.9	54.5	30.0	**65.4**	66.3	64.6	77.9	52.1	53.7	**58.8**	61.6
LKPO-GNN(k=8)	**85.8**	**64.6**	83.4	85.6	73.1	63.3	36.7	64.4	**70.5**	65.9	**79.1**	46.3	**54.5**	55.9	**61.7**

Table 6
Complexity Comparison

Methods	ModelSize(MB)	Forward Times(ms)
PointNet[16]	98.2	128.26
KCNet[19]	25.7	36.5
SO-Net[18]	59.5	68.3
PointNet++[17]	**16.2**	24.63
PointCNN [20]	95.4	80.43
LKPO-GNN(k=8)	28.2	**19.94**
LKPO-GNN(k=32)	51.1	27.64

References

[1] D. Zermas, I. H. Izzat, N. Papanikolopoulos, Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications (2017) 5067–5073.

[2] W. Tao, X. He, N. Barnes, Learning structured hough voting for joint object detection and occlusion reasoning, in: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, 2013.

[3] A. Garciagarcia, S. Ortsescolano, S. Oprea, V. Villenamartinez, P. Martinezgonzalez, J. Garciarodriguez, A survey on deep learning techniques for image and video semantic segmentation, Applied Soft Computing 70 (2018) 41–65.

[4] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M. Yang, J. Kautz, Splatnet: Sparse lattice networks for point cloud processing, arXiv: Computer Vision and Pattern Recognition (2018).

[5] J. Lahoud, B. Ghanem, 2d-driven 3d object detection in rgb-d images, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017.

[6] W. Wang, R. Yu, Q. Huang, U. Neumann, Sgpn: Similarity group proposal network for 3d point cloud instance segmentation, arXiv: Computer Vision and Pattern Recognition (2017).

[7] Q. Huang, W. Wang, U. Neumann, Recurrent slice networks for 3d segmentation on point clouds, arXiv: Computer Vision and Pattern Recognition (2018).

[8] H. Fan, H. Su, L. J. Guibas, A point set generation network for 3d object reconstruction from a single image, arXiv: Computer Vision and Pattern Recognition (2016).

[9] D. G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.

[10] H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3d shape recognition, arXiv: Computer Vision and Pattern Recognition (2015).

[11] Y. Li, S. Pirk, H. Su, C. R. Qi, L. J. Guibas, Fpnn: Field probing neural networks for 3d data, arXiv: Computer Vision and Pattern Recognition (2016).

[12] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3d shapenets: A deep representation for volumetric shapes, arXiv: Computer Vision and Pattern Recognition (2014).

[13] C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan, L. J. Guibas, Volumetric and multi-view cnns for object classification on 3d data, arXiv:

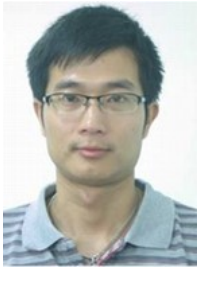
Computer Vision and Pattern Recognition (2016).
[14] R. Klokov, V. Lempitsky, Escape from cells: Deep kd-networks for the recognition of 3d point cloud models, arXiv: Computer Vision and Pattern Recognition (2017).
[15] G. Riegler, A. O. Ulusoy, A. Geiger, Octnet: Learning deep 3d representations at high resolutions, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[16] C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, arXiv: Computer Vision and Pattern Recognition (2016).
[17] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, arXiv: Computer Vision and Pattern Recognition (2017).
[18] J. Li, B. M. Chen, G. H. Lee, So-net: Self-organizing network for point cloud analysis, arXiv: Computer Vision and Pattern Recognition (2018).
[19] Y. Shen, C. Feng, Y. Yang, D. Tian, Mining point cloud local structures by kernel correlation and graph pooling, arXiv: Computer Vision and Pattern Recognition (2017).
[20] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, B. Chen, Pointcnn: Convolution on $\mathcal{X}$-transformed points, arXiv: Computer Vision and Pattern Recognition (2018).
[21] M. Jiang, Y. Wu, C. Lu, Pointsift: A sift-like network module for 3d point cloud semantic segmentation., arXiv: Computer Vision and Pattern Recognition (2018).
[22] J. Bruna, W. Zaremba, A. Szlam, Y. Lecun, Spectral networks and locally connected networks on graphs, arXiv: Machine Learning (2013).
[23] T. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv: Machine Learning (2016).
[24] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) 2278–2324.
[25] A. Dai, C. R. Qi, M. Niesner, Shape completion using 3d-encoder-predictor cnns and shape synthesis, arXiv: Computer Vision and Pattern Recognition (2016).
[26] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoue, H. Nguyen, R. Ohbuchi, et al., A comparison of methods for non-rigid 3d shape retrieval, Pattern Recognition 46 (1) (2013) 449–461.
[27] Y. Lipman, D. Cohenor, D. Levin, H. Talezer, Parameterization-free projection for geometry reconstruction, international conference on computer graphics and interactive techniques 26 (3) (2007) 22.
[28] T. Yu, J. Meng, J. Yuan, Multi-view harmonized bilinear network for 3d object recognition, arXiv: Computer Vision and Pattern Recognition (2018) 186–194.
[29] L. Yu, X. Li, C. Fu, D. Cohenor, P. Heng, Pu-net: Point cloud upsampling network., arXiv: Computer Vision and Pattern Recognition (2018).
[30] T. F. Gonzalez, Clustering to minimize the maximum intercluster distance, Theoretical Computer Science 38 (1985) 293–306.
[31] Y. Eldar, M. Lindenbaum, M. Porat, Y. Y. Zeevi, The farthest point strategy for progressive image sampling, IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society (1997).
[32] P. Kamousi, S. Lazard, A. Maheshwari, S. Wuhrer, Analysis of farthest point sampling for approximating geodesics in a graph, Computational Geometry 57 1–7.
[33] Z. B. Azouz, P. Bose, C. Shu, S. Wuhrer, Approximations of geodesic distances for incomplete triangular manifolds, canadian conference on computational geometry (2007) 177–180.
[34] L. Liu, L. Zhang, Y. Xu, C. Gotsman, S. J. Gortler, A local/global approach to mesh parameterization, symposium on geometry processing 27 (5) (2008) 1495–1504.
[35] A. Elad, R. Kimmel, On bending invariant signatures for surfaces, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (10) (2003) 1285–1295.
[36] P. Wang, C. Sun, Y. Liu, X. Tong, Adaptive o-cnn: A patch-based deep representation of 3d shapes, arXiv: Computer Vision and Pattern Recognition (2018).
[37] H. Lin, H. Chen, Q. Dou, L. Wang, J. Qin, P. Heng, Scannet: A fast and dense scanning framework for metastatic breast cancer detection from whole-slide images, arXiv: Computer Vision and Pattern Recognition (2017).
[38] A. X. Chang, T. Funkhouser, L. J. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al., Shapenet: An information-rich 3d model repository, arXiv: Graphics (2015).
[39] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, S. Savarese, 3d semantic parsing of large-scale indoor spaces, computer vision and pattern recognition (2016) 1534–1543.
[40] Y. Arakawa, H. Kakeya, M. Isogai, K. Suzuki, F. Yamaguchi, Space-shared communication based on truly 3d information space, international conference on image processing 3 (1999) 31–35.
[41] E. Shelhamer, J. Long, T. Darrell, Fully convolutional networks for semantic segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (4) (2017) 640–651.
[42] J. Sun, M. Ovsjanikov, L. J. Guibas, A concise and provably informative multi-scale signature based on heat diffusion, symposium on geometry processing 28 (5) (2009) 1383–1392.
[43] M. Kazhdan, T. Funkhouser, S. Rusinkiewicz, Rotation invariant spherical harmonic representation of 3d shape descriptors, symposium on geometry processing (2003) 156–164.
[44] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, M. Niesner, Scannet: Richly-annotated 3d reconstructions of indoor scenes, arXiv: Computer Vision and Pattern Recognition (2017).
[45] M. Simonovsky, N. Komodakis, Dynamic edge-conditioned filters in convolutional neural networks on graphs, arXiv: Computer Vision and Pattern Recognition (2017).
[46] X. Chen, H. Ma, J. Wan, B. Li, T. Xia, Multi-view 3d object detection network for autonomous driving, arXiv: Computer Vision and Pattern Recognition (2016).

[47] P. Wang, Y. Liu, Y. Guo, C. Sun, X. Tong, O-cnn: octree-based convolutional neural networks for 3d shape analysis, ACM Transactions on Graphics 36 (4) (2017) 72.
[48] L. Yi, H. Su, X. Guo, L. J. Guibas, Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation, arXiv: Computer Vision and Pattern Recognition (2016).
[49] L. Landrieu, M. Simonovsky, Large-scale point cloud semantic segmentation with superpoint graphs., arXiv: Computer Vision and Pattern Recognition (2017).

Wenjing Zhang born in 1989. She received B.S. degree from Automation School of Xi'an Kedagaoxin University in 2011 and her M.S. degree from Computer Science School of Xi'an Polytechnic University in 2014. Ph. D. candidate in the Software School of Xiamen University.Her current research foucus on deep learning and artificial intelligence.

Song-Zhi Su is an associate professor at Artificial Intelligent Department of Xiamen University, China. He received the B.S. degree in Computer Science and Technology from Shandong University, China, in 2005. He received M.S. and Ph.D. degrees in Computer Science in 2008 and 2011, both from Xiamen University, Fujian, China. He joined the Faculty of Xiamen University as an assistant professor in 2011. His research interests include visual object detection, face recognition, 3d scene perception and understanding, deep learning and its applications on computer vision.

Beizhan Wang born in 1965. Doctor, Professor and Ph. D. supervisor in the Department of Software Engineering Xiamen University. He has published over 30 journal articles and his main research interests include software architecture, data warehose, and network performance evaluation.



Qingqi Hong is an Associate Professor at the School of Informatics, Xiamen University, China. He received his PhD degrees in Computer Science from the University of Hull, UK. His current research interests include machine learning, medical imaging processing, 3D printing, and so on.

Li Sun is a Lecturer (Assistant Professor) of Computer Science at University of Sheffield. He was a Postdoctoral Research Fellow at the Oxford Robotics Institute, University of Oxford. Li was awarded the prestigious national scholarship from the Scottish government and received Ph.D. in robotics from the University of Glasgow in 2016. He worked as a postdoctoral research fellow at Lincoln Centre of Autonomous Systems, University of Lincoln (2017-2018) and Extreme Robotics Lab, University of Birmingham (2016-2017). Since 2012, he has worked as a main researcher/contributor of several EU/EPSRC funded projects. Li's research focuses on the core challenges in the emerging robot vision to enable the robot to manipulate complex industrial objects e.g. deformable object and waste-like objects or drive in the dynamic, real-life environments e.g. warehouse, on-road/off-road driving.