RGCNN: Regularized Graph CNN for Point Cloud Segmentation

Gusi Te
Peking University
tegusi@pku.edu.cn

Wei Hu
Peking University
forhuwei@pku.edu.cn

Amin Zheng
MTlab, Meitu Inc.
zam@meitu.com

Zongming Guo
Peking University
guozongming@pku.edu.cn

ABSTRACT

Point cloud, an efficient 3D object representation, has become popular with the development of depth sensing and 3D laser scanning techniques. It has attracted attention in various applications such as 3D tele-presence, navigation for unmanned vehicles and heritage reconstruction. The understanding of point clouds, such as point cloud segmentation, is crucial in exploiting the informative value of point clouds for such applications. Due to the irregularity of the data format, previous deep learning works often convert point clouds to regular 3D voxel grids or collections of images before feeding them into neural networks, which leads to voluminous data and quantization artifacts. In this paper, we instead propose a regularized graph convolutional neural network (RGCNN) that directly consumes point clouds. Leveraging on spectral graph theory, we treat features of points in a point cloud as signals on graph, and define the convolution over graph by Chebyshev polynomial approximation. In particular, we update the graph Laplacian matrix that describes the connectivity of features in each layer according to the corresponding learned features, which adaptively captures the structure of dynamic graphs. Further, we deploy a graph-signal smoothness prior in the loss function, thus regularizing the learning process. Experimental results on the ShapeNet part dataset show that the proposed approach significantly reduces the computational complexity while achieving competitive performance with the state of the art. Also, experiments show RGCNN is much more robust to both noise and point cloud density in comparison with other methods. We further apply RGCNN to point cloud classification and achieve competitive results on the ModelNet40 dataset.

KEYWORDS

Graph CNN, graph-signal smoothness prior, updated graph Laplacian, point cloud segmentation

1 INTRODUCTION

The development of depth sensors like Microsoft Kinect and 3D scanners like LiDAR has enabled convenient acquisition of 3D point clouds, a popular signal representation of arbitrarily-shaped objects in the 3D space. Point clouds consist of a set of points, each of which is composed of 3D coordinates and possibly attributes such as color and normal. Thanks to the efficient representation, point clouds have been widely deployed in various fields, such as 3D immersive tele-presence, navigation for autonomous driving, free-viewpoint videos and heritage preservation [28]. Hence, the analysis of point clouds, such as point cloud segmentation, becomes active research topics in order to exploit the informative value of point clouds.

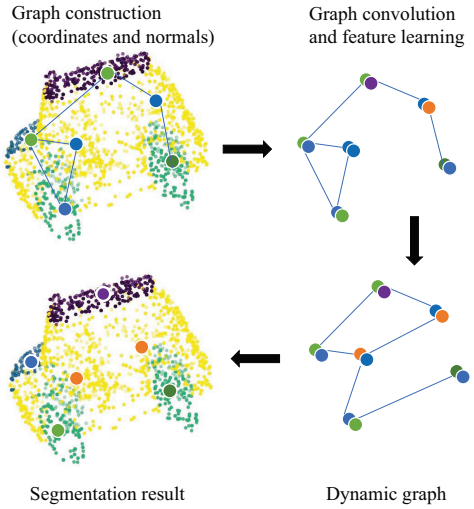


Figure 1: Illustration of the RGCNN architecture, which directly consumes raw point clouds (car in this example) without voxelization or rendering. It constructs graphs based on the coordinates and normal of each point, performs graph convolution and feature learning, and adaptively updates graphs in the learning process, which provides an efficient and robust approach for 3D recognition tasks such as point cloud segmentation and classification.

Previous point cloud segmentation works can be classified into model-driven segmentation and data-driven segmentation. Model-driven methods include edge-based [21], region-growing [22] and model-fitting [27], which are based on the prior knowledge of the geometry but sensitive to noise, uneven density and complicated structure. Data-driven segmentation, on the other hand, learns the semantics from data, such as deep learning methods [19]. Nevertheless, typical deep learning architectures require regular input data

MM '18, October 22–26, 2018, Seoul, Republic of Korea

ACM ISBN 978-1-4503-5665-7/18/10…$15.00
https://doi.org/10.1145/3240508.3240621

formats, such as images on regular 2D grids or voxels on 3D grids, in order to perform operations like convolution and pooling. For *irregular* 3D point clouds, most previous works convert them to regular 3D voxel grids [17] or collections of images [25] before feeding them into typical convolutional neural networks (CNN). This, however, introduces quantization error in the conversion process and renders the resulting data unnecessarily voluminous.

Recently, Graph Convolutional Neural Network (GCNN) has been proposed to generalize CNNs to graphs [14]. The key idea is to consider the convolution of graphs in the spectral domain, leveraging on spectral graph theory [4]. However, this requires the eigen-decomposition of graph Laplacian matrices [4] that describe the connectivity of graphs, which is computationally expensive. Hence, several methods propose to approximate the convolution in the spectral domain by spatial filtering, such as Chebyshev polynomials [9], Lanczos method [26], Cayley polynomials [15], etc. Nevertheless, the graph Laplacian matrix is always *fixed*[14], which is unable to represent the structures of dynamic graphs in the learning process. Also, though GCNN has shown its efficiency in semi-supervised classification, it hasn't been deployed to point cloud segmentation yet.

In order to address the above problems, we propose a regularized graph convolutional neural network (RGCNN) for point cloud segmentation. As depicted in Fig. 1, RGCNN treats the features of points as graph signals, and takes the feature matrix and adjacency matrix of irregular point clouds as the input. Specifically, we choose the coordinates and normals of each point as the features to represent the underlying geometry of point clouds. The output is the per point segmentation labels for each point of the input. Leveraging on the basic framework of GCNN with truncated Chebyshev approximation, we design a three-layer GCNN with high-order Chebyshev polynomials. In particular, we incorporate a *graph-signal smoothness prior* into the loss function, which regularizes the learning process. This essentially combines data-driven methods with model-driven ones. Further, we prove the *spectral smoothing* property of this prior, which essentially enforces Laplacian smoothing in the spectral domain. Besides, instead of fixing the graph structure as in previous works (e.g., [32]), we *update* the graph Laplacian matrix in each layer, thus capturing the dynamic topology of graphs. We also prove the permutation-invariance property of the proposed RGCNN, so that when the input permutes, the output permutes in the same way. Finally, we extend the architecture of RGCNN for the application of point cloud classification.

While details are presented in the paper, the key contributions are as follows:

- To the best of our knowledge, we are the first to design GCNN for point cloud segmentation, which is suitable for consuming unordered 3D point clouds;
- We regularize each layer of RGCNN by adding graph-signal smoothness prior in the loss function, and prove the spectral smoothing property of this prior;
- We update the graph Laplacian matrix in each layer of RGCNN, in order to adaptively capture the structure of dynamic graphs.
- Extensive experiments show that RGCNN significantly reduces the computation complexity while achieving competitive results with the state of the art. It is also much more robust to both low density and noise in comparison with other methods.

The rest of the paper is organized as follows. Section 2 provides a review on previous works of point cloud coding and introduce GCNN as the basic framework. We present the problem statement of point cloud segmentation in Section 3, and then elaborate on the proposed RGCNN in Section 4. The proposed loss function and theoretical analysis is discussed in Section 5. Next, performance evaluation and comparison is presented in Section 6. Finally, we conclude the paper in Section 7.

2 RELATED WORK

We will first review previous works on point cloud segmentation, and then introduce GCNN, which inspires the proposed method.

2.1 Point Cloud Segmentation

Previous works on point cloud segmentation can be mainly classified into two categories: model-driven methods and data-driven methods.

Model-driven methods: This class of approaches segment point clouds by assuming certain models of the underlying geometry. According to different models, they are further categorized as follows.

- **Edge-based methods**: Rabbani et al. propose an edge-based method in [21], which outlines the borders of different regions by edge detection and then groups points inside the borders to deliver final segments. Jiang et al. [13] propose scan-line grouping methods to represent surfaces, and divide them by edges. This achieves good performance on range images, but is unsuitable for point clouds with uneven density. Although edge-based methods are fast, they are sensitive to noise and uneven density.
- **Region growing methods**: Starting from one or more points with specific characteristics, these methods grow around adjacent and similar points. Further, these methods can be divided into top-down approaches and bottom-up approaches. The difference includes the initial choice of points and how they grow afterwards. The main disadvantage lies in the selection of initial points and in the presence of complicated structures.
- **Model fitting methods**: This category is based on the observation that many man-made objects can be decomposed into geometric primitives like planes, cylinder and spheres. Points that conform to some primitive shapes are treated as one segment. Two popular algorithms include the Hough Transform [27] and the Random Sample Consensus approach [3]. However, details may fail to be modelled into easily recognizable geometric primitives.

Model-driven methods are primarily limited by the assumed prior knowledge. Also, objects with complex structures are great challenges for small-scale model datasets.

Data-driven methods This class is concerned with Artificial Intelligence algorithms based on empirical and training data. As 3D point clouds are irregular, traditional data-driven methods cannot

be applied directly. Preprocessing, such as feature extraction, is thus required. Features of high quality can greatly improve the learning efficiency.

- **Segmentation based on clustering**: Clustering groups (3D) points into clusters based on attributes or features. Biosca et al. deploy unsupervised clustering and fuzzy algorithms for laser point clouds [1]. Filin et al. proposes an approach using normal vectors derived from a neighborhood system called slope adaptive [7], where the slopes of normal vectors and height difference between each point and its neighbors are applied as features. Ma et al. exploit spectral clustering for point cloud segmentation [16], and propose a novel approach to find k-nearest-neighbors for graph construction. Clustering-based segmentation achieves better performance than model-driven methods under complex scenes. However, there lacks the view of local information.
- **Deep learning segmentation**: Deep learning methods have shown great potential in 3D shape recognition, such as view-based learning [25], 3D ShapeNets [30], VoxNet [17] and VoxelNet [33], which are mainly designed for point cloud classification.
  Regarding point cloud segmentation, Qi et al. come up with PointNet, a neural network which consumes point clouds directly [19]. The key breakthrough is the proposed symmetric function applied to the raw point cloud data. However, PointNet processes each point identically and independently. Hence, PointNet++ is proposed in [20] by introducing hierarchical grouping, which achieves better performance.

2.2 Graph Convolutional Neural Network

As CNN only deals with data defined on regular grids, it is extended to graphs for irregular data, which is referred to as GCNN. The key challenge is to define convolution over graphs, which is difficult due to the irregularity.

- **Spectral methods**: The convolution over graphs is defined in the spectral domain, which is the multiplication of the signal on graph with the eigenvector matrix of the graph Laplacian matrix [9, 10]. The computation complexity, however, is high due to the eigen-decomposition of the graph Laplacian matrix in order to get the eigenvector matrix. Hence, it is improved by [5] through fast localized convolutions, where Chebyshev expansion is deployed to approximate graph Fourier transform (GFT). Susnjara et al. introduce the Lancoz method for approximation [26].
- **Spatial methods**: In this method , many techniques are introduced to implement convolution directly on each node and its neighbors. Gori et al. introduce recurrent neural networks that operate on graphs in [8]. Duvenaud et al. propose a convolution-like propagation to acculmulate local features [6]. Bruna et al. deploy the multiscale clustering of graphs in convolution to implement multi-scale representation [2]. Furthermore, Niepert et al. define convolution on a sequence of nodes and perform normalization afterwards [18]. Spatial methods provide strong localized filters, but it also means it is difficult to learn the global structure.

Spectral GCNN has shown its efficiency in semi-supervised classification [14], which outperforms many state-of-the-art methods significantly on citation networks and node classification. However, to the best of our knowledge, we are the first to extend it to point cloud segmentation.

3 PROBLEM STATEMENT

We design a graph neural network that directly undertakes irregular 3D point clouds as the input for segmentation. We represent a point cloud as a set of 3D points $\{\mathbf{p}_i\}_{i=1}^n$, where $\mathbf{p}_i \in \mathbb{R}^m$ is a vector denoting the i-th point's feature, such as coordinates, color, normal, etc. In the proposed method, we adopt the coordinates (x_i, y_i, z_i) and normal (n_i^x, n_i^y, n_i^z) as the feature for the i-th point, i.e., $\mathbf{p}_i = (x_i, y_i, z_i, n_i^x, n_i^y, n_i^z)^T$, and thus $m = 6$.

In the proposed RGCNN, the input is a $n \times m$ feature matrix $\mathbf{P}$ and a $n \times n$ adjacency matrix W. Assuming we have k semantic labels, RGCNN outputs $n \times k$ scores $\mathbf{S}$ for each of the n points and each of the k labels.

4 THE PROPOSED RGCNN

We first present the overall RGCNN architecture, and then elaborate on important modules including graph construction, graph convolution and feature learning respectively.

4.1 RGCNN Architecture

As depicted in Fig. 2, RGCNN consists of one general model for extracting features and two branches for segmentation and classification tasks respectively. It takes raw point clouds with coordinates and normals as the input, learns local features by graph convolution and then outputs the segmentation or classification score. In the segmentation branch, we deploy graph convolution to aggregate features, and then concatenate features from different layers to represent both local and global features. The per-point label is finally given in the output layer. In the classification branch, we additionally deploy global max pooling to collect global features and use multilayer perceptron (MLP) to get the final score. Specifically, RGCNN has three regularized graph convolution layers. Each layer consists of graph construction, graph convolution and feature filtering, which are elaborated in order as follows.

4.2 Graph Construction

As the graph construction has crucial effect on the efficiency of the network, we first discuss the proposed approach to construct graphs over point clouds.

Graph and Graph Laplacian. We consider an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{A}\}$ composed of a vertex set $\mathcal{V}$ of cardinality $|\mathcal{V}| = n$, an edge set $\mathcal{E}$ connecting vertices, and a weighted *adjacency matrix* $\mathbf{A}$. $\mathbf{A}$ is a real symmetric $n \times n$ matrix, where $a_{i,j}$ is the weight assigned to the edge (i, j) connecting vertices i and j. We assume non-negative weights, *i.e.*, $a_{i,j} \geq 0$.

The Laplacian matrix is defined from the adjacency matrix. Among different variants of Laplacian matrices, the *combinatorial graph Laplacian* used in [11, 12, 23] is defined as $\mathcal{L}_c := \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the *degree matrix*—a diagonal matrix with $d_{i,i} = \sum_{j=1}^n a_{i,j}$. One

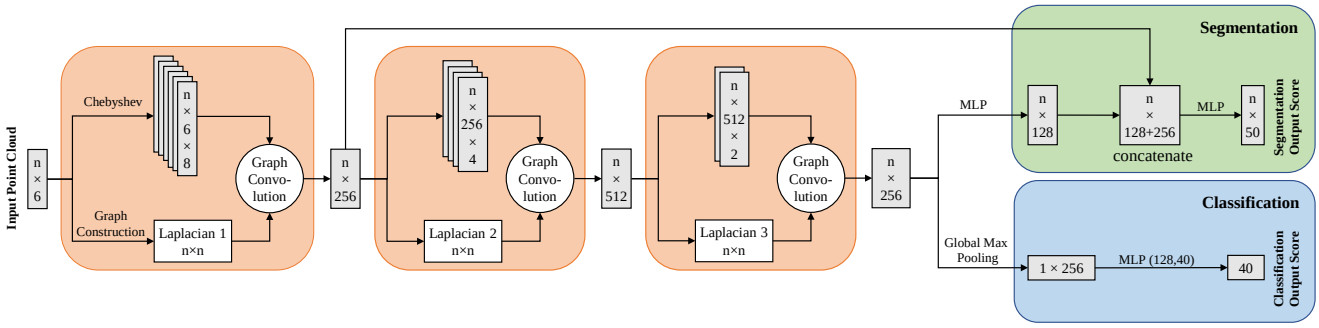


Figure 2: The architecture of the proposed RGCNN.

normalized graph Laplacian matrix is defined as $\mathcal{L} = \mathbf{D}^{-\frac{1}{2}}\mathcal{L}_c\mathbf{D}^{-\frac{1}{2}}$, which is used in the sequel because of its normalization property.

Graph signal. Graph signal refers to data residing on the vertices of a graph. In this paper, the graph signal is the features of each point in the point cloud, i.e., the feature vector $\mathbf{p}_i$ of the i-th point.

Graph Construction. Though there exist various ways to construct graphs, we choose complete graphs, which connect each point with all the other points in the point cloud and thus consider the relationship among all the points. The edge weight is defined based on the distance between features of points, which is able to measure the similarity among points in terms of structure. Specifically, the weight of an edge connecting points i and j is defined as

$$a_{i,j} = \exp(-\beta\|\mathbf{p}_i - \mathbf{p}_j\|_2^2), \tag{1}$$

where β is a scalar parameter. We empirically set $\beta = 1$ in the experiments.

4.3 Graph Convolution

The core of GCNN is graph convolution. Unlike images or videos, it is difficult to define convolution over graphs in the vertex domain, because a meaningful translation operator in the vertex domain is nontrivial to define due to the unordered vertices. Hence, inspired by [5], we start from filtering of graph signals in the spectral domain, and then deploy Chebyshev approximation to reduce the computational complexity.

Spectral filtering of graph signals. The convolution operator on a graph $*_{\mathcal{G}}$ is first defined in the spectral domain [2], specifically in the GFT domain. GFT is computed from the graph Laplacian matrix. As the graph Laplacian is symmetric and positive semi-definite, it admits a complete set of orthonormal eigenvectors. The GFT basis $\mathbf{U}$ is then the eigenvector set of the Laplacian matrix. The GFT of a graph signal $\mathbf{x}$ is thus defined as $\hat{\mathbf{x}} = \mathbf{U}^T\mathbf{x}$, and the inverse GFT follows as $\mathbf{x}' = \mathbf{U}\hat{\mathbf{x}}$.

Hence, the convolution between two graph signals $\mathbf{x}$ and $\mathbf{y}$ can be defined as the multiplication of the corresponding GFT coefficients, followed by the inverse GFT, i.e.,

$$\mathbf{x} *_{\mathcal{G}} \mathbf{y} = \mathbf{U}(\mathbf{U}^T\mathbf{x}) \odot (\mathbf{U}^T\mathbf{y}), \tag{2}$$

where $\odot$ is the element-wise Hadamard product. Then the spectral filtering of a graph signal $\mathbf{x}$ by g_θ is

$$\mathbf{y} = g_\theta(\mathcal{L})\mathbf{x} = g_\theta(\mathbf{U}\Lambda\mathbf{U}^T)\mathbf{x} = \mathbf{U}g_\theta(\Lambda)\mathbf{U}^T\mathbf{x}. \tag{3}$$

Chebyshev approximation for localized filtering. The spectral filtering, however, has two limitations: 1) it has high computational complexity of $O(n^3)$ due to the eigen-decomposition of the graph Laplacian; 2) it is not localized. Hence, Defferrard et al. propose to use truncated Chebyshev polynomials to approximate the spectral filtering [5]. The K-localized filtering operation is described as follows.

$$\mathbf{y} = g_\theta(\mathcal{L})\mathbf{x} = \sum_{k=0}^{K-1} \theta_k T_k(\mathcal{L})\mathbf{x}, \tag{4}$$

where θ_k denotes the k-th Chebyshev coefficient. $T_k(\mathcal{L})$ is the Chebyshev polynomial of order k. It is recurrently calculated by $T_k(\mathcal{L}) = 2\mathcal{L}T_{k-1}(\mathcal{L}) - T_{k-2}(\mathcal{L})$, where $T_0(\mathcal{L}) = 1, T_1(\mathcal{L}) = \mathcal{L}$. Now the computational complexity is reduced to $O(K|\mathcal{E}|)$.

4.4 Feature learning

Following the graph convolution, we generate a new feature vector for each point from a weight matrix, a bias and the ReLU activation function. This is formulated as follows:

$$y = \mathrm{ReLU}(g_\theta(\mathcal{L})\mathbf{x}\mathbf{W} + \mathbf{b}), \tag{5}$$

where $\mathbf{W} \in \mathbb{R}^{F_1 \times F_2}$ is the matrix of weight parameters of the trained network, and F_1 and F_2 are the dimensions of generated features in two connected layers respectively. $\mathbf{b} \in \mathbb{R}^{n \times F_2}$ is the bias, while ReLU is an activation function.

In practice, each output feature is calculated by $y_i = \sum_{j=1}^{F_2} w_{i,j}x_i$, $i = 0, \ldots, K-1$. When $K = 1$, it is equivalent to a one-layer perceptron shared by all the points, which plays an important role in some deep learning networks, such as PointNet. This works well in capturing features of individual points, but loses the neighborhood information. In our model, we take the neighborhood into consideration by graph convolution with truncated Chebyshev polynomials of order $K > 1$, thus incorporating local features.

Fig. 3 demonstrates that the feature space varies in different layers. We observe that a deeper layer is able to capture semantically similar structures better in the high-dimensional feature space.

5 THE PROPOSED LOSS FUNCTION AND THEORETICAL ANALYSIS

This section presents the proposed loss function, in which a graph-signal smoothness prior is added. We then provide theoretical analysis of the added prior, and also prove the permutation invariance property of RGCNN.

5.1 The proposed loss function

While the common error function in the loss function is the consistency of outputs with targets, i.e., the cross entropy, we propose to additionally incorporate a graph-signal smoothness prior as the regularization term. This prior essentially enforces the features of adjacent vertices to be more similar, which eases the segmentation task.

Graph-signal Smoothness Prior. A graph signal $\mathbf{y}$ defined on a graph $\mathcal{G}$ is smooth with respect to the topology of $\mathcal{G}$ if

$$\sum_{i\sim j} a_{i,j}(y_i - y_j)^2 < \epsilon, \ \forall i, j, \tag{6}$$

where ϵ is a small positive scalar, and $i \sim j$ denotes two vertices i and j are one-hop neighbors in the graph. In order to satisfy Eq. (6), y_i and y_j have to be similar for a large edge weight $a_{i,j}$, and could be quite different for a small $a_{i,j}$. Hence, Eq. (6) enforces $\mathbf{y}$ to adapt to the topology of $\mathcal{G}$, which is thus coined *graph-signal smoothness prior*.

As $\mathbf{y}^T\mathcal{L}\mathbf{y} = \sum_{i\sim j} a_{i,j}(y_i - y_j)^2$ [24], Eq. (6) is concisely written as $\mathbf{y}^T\mathcal{L}\mathbf{y} < \epsilon$ in the sequel.

Loss Function. We add the aforementioned graph-signal smoothness prior in the loss function. In particular, the prior is computed from all the three graph convolution layers. The mathematical description is

$$E(\mathbf{y}^o, \mathbf{y}') = -\sum_{i=1}^{n} y_i^o \log(y_i{}') + \gamma \sum_{l=0}^{2} \mathbf{y}_l^T \mathcal{L}\mathbf{y}_l, \tag{7}$$

where $\mathbf{y}^o$ is the output score, $\mathbf{y}'$ is the ground truth label, and $\mathbf{y}_l$ is the feature map of the l-th layer. γ is the penalty parameter for the smoothness term, which is empirically set to 10^{-9} in our experiments.

5.2 Theoretical analysis

We provide analysis for the spectral property of the graph-signal smoothness prior and the permutation-invariance property of the proposed architecture.

Theorem 1. The graph-signal smoothness prior enforces more low-frequency components in the GFT domain.

Proof. While we have discussed that the added graph-signal smoothness prior regularizes the graph signal to be adapted to the structure of the graph, we further analyze the spectral behaviour of this prior. Specifically, as $\mathcal{L}$ is diagonalizable as $\mathbf{U}\Lambda\mathbf{U}^T$ as mentioned earlier, we have

$$\mathbf{y}^T\mathcal{L}\mathbf{y} = \mathbf{y}^T(\mathbf{U}\Lambda\mathbf{U}^T)\mathbf{y} = (\mathbf{U}^T\mathbf{y})^T\Lambda(\mathbf{U}^T\mathbf{y}) = \alpha^T\Lambda\alpha = \sum_{i=1}^{n}\lambda_i\alpha_i^2, \tag{8}$$

where $\alpha \in \mathbb{R}^n$ is the GFT transform coefficient vector, α_i is the i-th coefficient, and λ_i is the i-th eigenvalue of $\mathcal{L}$. The eigenvalues are often sorted as $\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$. It is known that larger eigenvalues correspond to higher-frequency GFT coefficients. For instance, λ_1 corresponds to the DC coefficient α_1, while λ_n corresponds to the highest-frequency coefficient α_n.

Hence, when we try to minimize the graph-signal smoothness prior in the loss function, the higher-frequency coefficients are weighted by a larger eigenvalue, and are thus penalized more heavily. This means that by adding this prior into the loss function, low-frequency components are better preserved, which leads to smoothing in the spectral domain. The smoothing operation enforces the features of vertices within each connected component of the graph similar, thus greatly easing the segmentation task.

Theorem 2. The proposed RGCNN is permutation-invariant, i.e., if the rows of the input feature matrix are permuted, the output permutes in the same way.

Theorem 2 indicates that the segmentation result of RGCNN is irrelevant to the order of the input, which is suitable for the unordered point cloud data. The proof is as follows.

Proof. Denote a $n \times n$ permutation matrix by $\mathbf{H}$. We prove if the input $n \times m$ feature matrix $\mathbf{P}$ is permuted by $\mathbf{H}$, i.e., $\mathbf{HP}$, then the $n \times k$ output $\mathbf{S}$ permutes as $\mathbf{HS}$.

Since the main operation of RGCNN is graph convolution, according to Eq. (4), we have

$$\mathbf{S}' = \sum_{k=0}^{K-1}\theta_k T_k(\mathcal{L})(\mathbf{HP}) = \mathbf{H}\sum_{k=0}^{K-1}\theta_k T_k(\mathcal{L})\mathbf{P} = \mathbf{HS} \tag{9}$$

Theorem 2 is hence proved.

5.3 Complexity analysis

Besides, another prominent superiority of our method is the simplicity of parameters, which reduces both time and spatial complexity. This is because the variables only relate to the Chebyshev order K and feature transform matrix $\mathbf{W}$. For each layer, the total number of parameters is approximately $O(N_1 * N_2 * K)$, where N_1 and N_2 denote the number of connecting feature channels. It is far less than PointNet which contains a T-Net to calculate the transform matrix.

6 EXPERIMENTAL RESULTS

In order to evaluate the performance of RGCNN, we carry out extensive experiments for point cloud segmentation, in terms of the segmentation accuracy and the robustness to density and noise. Further, we apply the network architecture to the classification task and provide comparison with the state-of-the-art methods. Finally, we provide analysis on the space and time complexity, and discuss the connections and differences of RGCNN with other competing methods.

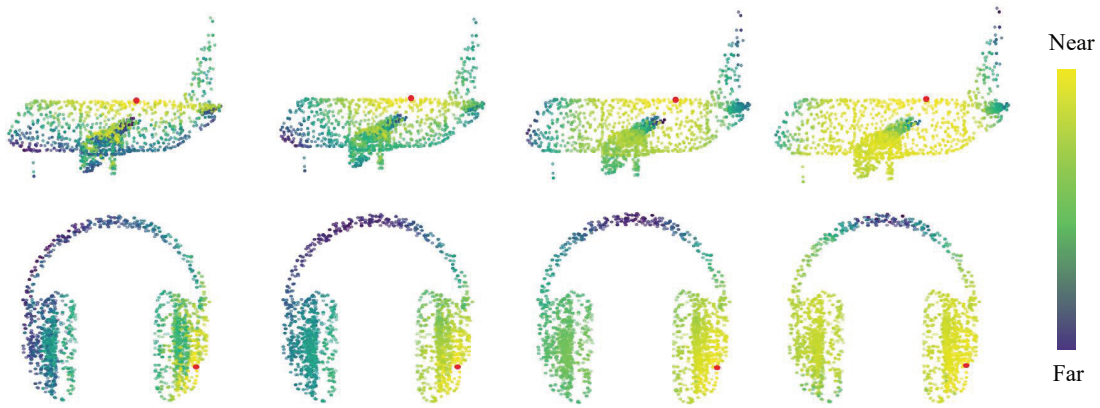


Figure 3: The feature space varies in different layers. The red point in each picture is the basic point. Different colors of the other points represent the Euclidean distance between each point and the basic point in the feature space.

Table 1: Segmentation results on ShapeNet part dataset (in mIoU).

	mean	aero	bag	cap	car	chair	earphone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skateboard	table
ShapeNet	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
PointNet	83.7	**83.4**	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++	**85.1**	82.4	79.0	87.7	**77.3**	**90.8**	71.8	91.0	85.9	83.7	95.3	**71.6**	94.1	81.3	58.7	76.4	**82.6**
SynSpecCNN	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
Ours	84.3	80.2	**82.8**	**92.6**	75.3	89.2	73.7	91.3	**88.4**	83.3	**96.0**	63.9	**95.7**	60.9	44.6	72.9	80.4

6.1 Experimental setup

Architecture parameters. In the architecture depicted in Fig. 2, the network comprises of three graph convolution layers with the Chebyshev order $K = (6, 5, 3)$ and dimensions of generated features $F = (128, 512, 1024)$, followed by three MLP layers $(512, 192, 50)$.

Training. We conduct experiments on ShapeNet part dataset [31]. This contains 16881 shapes from 16 categories, annotated with 50 labels in total. In the experiments, we first utilize random sampling to extract 2048 points from each model, which form the input point clouds. Then we feed the coordinates and normal of each point into our model as raw features. We follow the training/validation/test setting proposed in [31], assuming each category label is known for each sample. Our full model is trained on a single Nvidia GeForce GTX 1080Ti with 100 epochs.

Evaluation metric. We evaluate segmentation by mean Intersection of Union (mIoU). IoU is widely used in semantic segmentation to measure the ratio of the ground truth and prediction, and mean IoU is the average of IoU for each label appearing in the model categories. We compare our method with ShapeNet [31], PointNet [19], PointNet++ [20] and SynSpecCNN [32].

6.2 Point cloud segmentation results

The evaluation results are listed in Table 1. Our model outperforms the other competing methods in 5 categories, and achieves competitive results with the state-of-the-art. Further, we demonstrate some visual results in Table 2. It can be observed that RGCNN has better and more consistent segmentation results than PointNet for some challenging objects. More segmentation results of RGCNN are shown in Fig. 8.

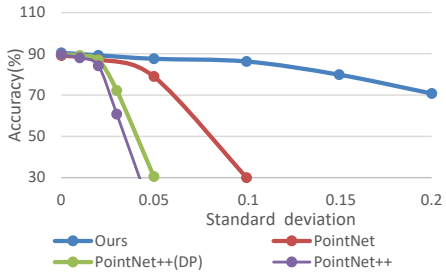


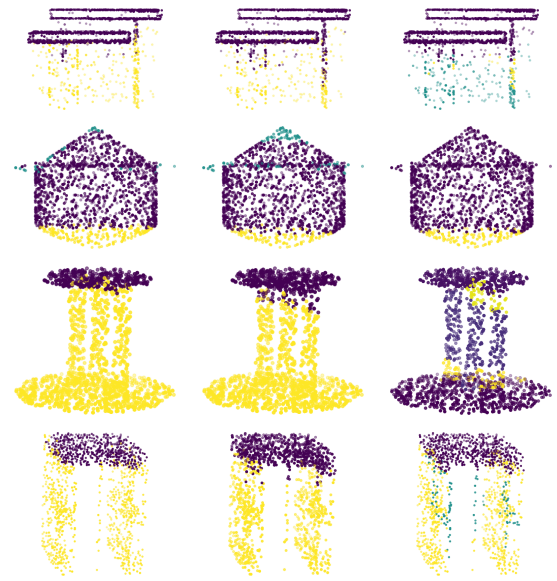
Figure 4: Accuracy with Gaussian noise.

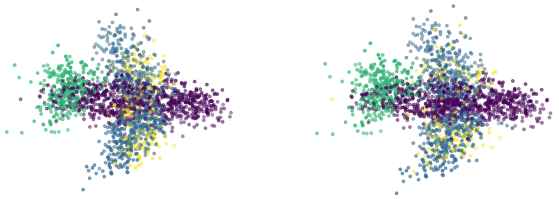
Also, we evaluate the proposed graph construction of fully-connected graphs. We test the commonly used k-nearest-neighbor graphs with $k = 30$. The resulting mean mIoU is 80.4%, which is much lower than using the proposed fully-connected graph. This confirms that the fully-connected graph is able to capture more abundant information, thus leading to better segmentation results.

6.3 Robustness test

Robustness to noise. In order to test the robustness of our model to random noise, we jitter the coordinates of the raw data with Gauss noise, with zero mean and standard deviation $\sigma \in [0.02, 0.2]$. Fig. 4 provides mIoU under different noise levels for PointNet, PointNet++, PointNet++(DP) with DP meaning random input dropout

Table 2: Visualization of segmentation results.





(a) Ground truth (b) Ours

Figure 5: The comparison between the ground truth and our segmentation result with the input perturbed with the noise level $\sigma = 0.1$.

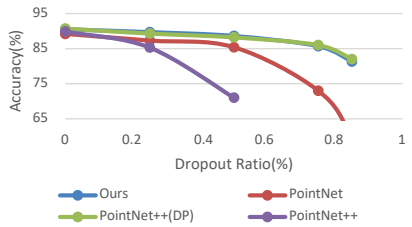
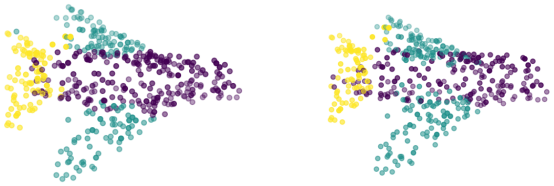


Figure 6: Accuracy with missing data.

during training [20] and our method. We see that while the performance of the other methods drops quickly with increasing noise variance, RGCNN is robust to noise even when the noise level is high. Also, our segmentation result is visually close to the ground truth from the macroscopic view even when $\sigma = 0.1$, as demonstrated in Fig. 5.



(a) Ground truth (b) Ours

Figure 7: The comparison between the ground truth and our segmentation result with the input under the missing ratio 0.75**.**

Robustness to density. We also test the robustness of our model to point clouds of low density. Random dropping is adopted to remove points with missing ratios $\{0.5, 0.75, 0.85, 0.95\}$. As depicted in Fig. 6, our accuracy keeps 85% even when the missing ratio is 0.75, which outperforms PointNet (73%) significantly. PointNet++ then amends this problem by randomly dropping out points in training time, which helps learn an optimized strategy to combine multi-scale features. With this strategy, PointNet++ achieves better performance, which is comparable to our robustness even though we have no such additional strategy. This is also visualized in Fig. 7, where our segmentation result is still satisfactory compared with the ground truth.

Hence, RGCNN is very robust to sparse and noisy point clouds. This gives credit to the proposed updated graph Laplacian and graph-signal smoothness prior in the loss function. This property is important in practical applications, since point clouds often suffer from noise or low density mainly due to inherent limitations of acquisition sensors.

6.4 Application to point cloud classification

We extend our model to the task of point cloud classification, as shown in the second branch of Fig. 2. It is tested on ModelNet40 dataset to predict the category of a given model. This dataset includes 12311 models from 40 categories, among which we utilize 9843 models for training and 2468 for testing. For each model, we select 1024 points with coordinates and normals randomly as the input point cloud, and then normalize each point cloud to a unit cube. Table 3 lists the classification results of different competing methods. It can be seen that our classification accuracy is better than PointNet and comparable to PointNet++.

6.5 Space and time complexity

We further compare the space and time complexity with other methods. Here, we choose our classification model to test the space and time complexity. Table 4 shows that our model has the fastest forward time with acceptable model size among these methods. Hence, our model is amenable to real-time classification tasks. Further, we test that the forward time will be even shorter (4.8 ms approximately) if we use fixed graphs instead.

Table 3: Classification results on ShapeNet40.

Metric	Mean Class Accuracy	Overall Accuracy
VoxNet[17]	83.0	85.9
PointNet [19]	86.0	89.2
PointNet++ [20]	-	90.7
DGCNN [29]	**90.2**	**92.2**
Ours	87.3	90.5

Table 4: Complexity comparison

Method	Model Size(MB)	Forward Time(ms)
PointNet	40	25.3
PointNet++	**12**	163.2
DGCNN	21	94.6
Ours	22.4	**7.5**

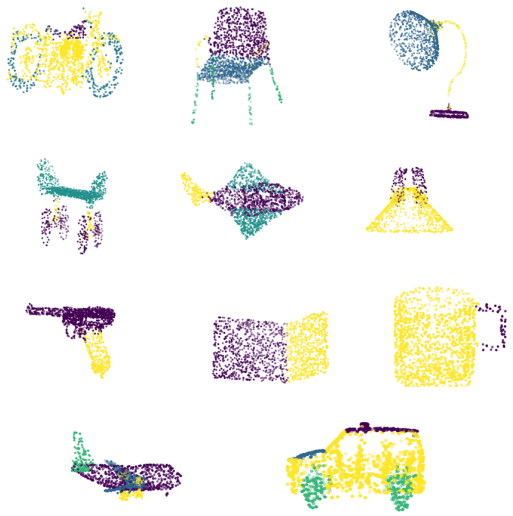


Figure 8: Segmentation results from RGCNN.

6.6 Discussion

Finally, we discuss the connections between our method and the other competing methods, as well as the advantages and limit in the following.

- In graph-based neural networks, the structure of the constructed graph plays an important role for tasks such as point cloud segmentation. Compared with existing graph-based methods in which the graph is fixed in general, our graph structure is dynamic to features in the learning process, thus adaptively capturing the generated features. Also, our graph construction is more computationally efficient.
- PointNet deploys MLP to extract the feature of each individual point and utilizes global pooling to extract the global feature, which is a special case in our model when the order of the Chebyshev polynomial is $K = 0$. Additionaly, our model is able to take the features of the K-hop neighborhood into consideration when $K \geq 1$.
- In SynSpecCNN, the connection between the spectral and spatial domain is learned, while our graph convolution is another form of spectral approximation but with more flexibility because of the dynamically updated graph Laplacian.
- The boundary between two segments is sometimes not quite sharp in our results, which limits the performance to some extent.

7 CONCLUSION

We propose RGCNN, a regularized graph convolutional neural network architecture that directly consumes irregular 3D point clouds. We introduce a graph-signal smoothness prior into the loss function, which essentially enforces Laplacian smoothing in both the spectral and spatial domain. Further, we update the graph Laplacian in each layer of the network in order to adaptively capture the dynamic graphs. Also, we prove the permutation-invariance property of RGCNN, which is suitable for the applications of unordered point clouds. Experimental results on the ShapeNet part dataset for point cloud segmentation validate the effectiveness of RGCNN, showing that RGCNN achieves competitive performance with the state of the art, with much lower computational complexity. We also evaluate that RGCNN is much more robust to both low density and noise than other competing methods. Further, we extend RGCNN for point cloud classification and achieve competitive results on ModelNet40 dataset.

8 ACKNOWLEDGEMENT

This work was supported by MSRA Collaborative Research under project ID FY18-Research-Sponsorship-029 and Alibaba Innovative Research under contract No. XTG201800108.

REFERENCES

[1] Josep Miquel Biosca and José Luis Lerma. Unsupervised robust planar segmentation of terrestrial laser scanner point clouds based on fuzzy clustering methods. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1):84–98, 2008.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[3] Dong Chen, Liqiang Zhang, P Takis Mathiopoulos, and Xianfeng Huang. A methodology for automated segmentation and reconstruction of urban 3-d buildings from als point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(10):4199–4217, 2014.

[4] F. K. Chung. Spectral graph theory. In *American Mathematical Society*, 1997.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[6] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[7] Sagi Filin. Surface clustering from airborne laser scanning data. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/A):119–124, 2002.

[8] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 729–734. IEEE, 2005.

[9] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[10] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[11] Wei Hu, Gene Cheung, Xin Li, and Oscar C. Au. Depth map compression using multi-resolution graph-based transform for depth-image-based rendering. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 1297–1300. IEEE, 2012.

[12] Wei Hu, Gene Cheung, Antonio Ortega, and Oscar C. Au. Multiresolution graph fourier transform for compression of piecewise smooth images. *IEEE Transactions on Image Processing*, 24(1):419–433, 2015.

[13] Xiaoyi Y Jiang, Urs Meier, and Horst Bunke. Fast range image segmentation using high-level segmentation primitives. In *Applications of Computer Vision, 1996. WACV'96., Proceedings 3rd IEEE Workshop on*, pages 83–88. IEEE, 1996.

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[15] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *arXiv preprint arXiv:1705.07664*, 2017.

[16] Teng Ma, Zhuangzhi Wu, Lu Feng, Pei Luo, and Xiang Long. Point cloud segmentation through spectral clustering. In *Information Science and Engineering (ICISE), 2010 2nd International Conference on*, pages 1–4. IEEE, 2010.

[17] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[18] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[19] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

[20] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017.

[21] Tahir Rabbani, Frank Van Den Heuvel, and George Vosselmann. Segmentation of point clouds using smoothness constraint. *International archives of photogrammetry, remote sensing and spatial information sciences*, 36(5):248–253, 2006.

[22] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008.

[23] Godwin Shen, W-S Kim, Sunil K Narang, Antonio Ortega, Jaejoon Lee, and Hocheon Wey. Edge-adaptive transforms for efficient depth map coding. In *Picture Coding Symposium (PCS), 2010*, pages 566–569. IEEE, 2010.

[24] D. A. Spielman. Lecture 2 of spectral graph theory and its applications. September 2004.

[25] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[26] Ana Susnjara, Nathanael Perraudin, Daniel Kressner, and Pierre Vandergheynst. Accelerated filtering on graphs using lanczos method. *arXiv preprint arXiv:1509.04537*, 2015.

[27] Fayez Tarsha-Kurdi, Tania Landes, and Pierre Grussenmeyer. Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. In *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, volume 36, pages 407–412, 2007.

[28] C. Tulvan, R. Mekuria, and Z. Li. Use cases for point cloud compression (pcc). In *ISO/IEC JTC1/SC29/WG11 (MPEG) output document N16331*, June 2016.

[29] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[30] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[31] Li Yi, Vladimir G Kim, Duygu Ceylan, I Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, Leonidas Guibas, et al. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 35(6):210, 2016.

[32] Li Yi, Hao Su, Xingwen Guo, and Leonidas Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. *arXiv preprint arXiv:1612.00606*, 2016.

[33] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *arXiv preprint arXiv:1711.06396*, 2017.