國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

學習三維圖卷積網路於點雲分析

Learning of 3D Graph Convolution Networks

for Point Cloud Analysis

林志皓

Zhi-Hao Lin

指導教授：王鈺強 博士

Advisor: Yu-Chiang Frank Wang, Ph.D.

中華民國 109 年 12 月

December 2020

國立臺灣大學（碩）博士學位論文
口試委員會審定書

學習三維圖卷積網路於點雲分析
Learning of 3D Graph Convolution Networks for Point Cloud Analysis

　　本論文係林志皓君（R08942062）在國立臺灣大學電信工程學研究所完成之碩（博）士學位論文，於民國 109 年 12 月 29 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

_____（簽名）

（指導教授）

林彥宇

_____　　_____

_____　　_____

_____　　_____

所　　長　_____（簽名）

致謝

　　在許多人的鼓勵和幫助下，這篇論文才終於得以完成。感謝我的指導教授王鈺強老師，一直以來都非常用心的指導每個學生。我從大學的專題研究開始，就跟老師一起做研究，然而回想當時的狀況，自己並不是相當認真，報告的技巧也滿糟糕的。但是老師一直都很用心地告訴我該怎麼改進，不管是表達的方式技巧，做研究的心態，撰寫論文的方式，還有待人處事的細節，都能感受到老師的用心。我也因此在即將畢業的這個時候，感受到自己確實進步了許多。很感謝老師對於題目的開放態度，讓我能嘗試各種可能性，也因為老師對於每一篇論文的堅持，影響我對研究的態度，凡事盡力以赴，即使結果不如人意，就修正過再試試。

　　感謝 VLL 的夥伴們，我一直覺得實驗室的討論風氣很棒，互相激勵各種想法，也能在一起去參加會議時玩得很開心，或是在研究生的平凡日常中一起約吃飯、打球、幫彼此慶生。大家不管在研究或是將來申請上，都給了很多超棒的建議。感謝聖喻，從大學專題開始就一起在 3D 組努力，不論是討論想法、跑實驗、研究別人的論文細節，都幫助了我許多。很懷念當初趕投稿，互吐苦水的日子。之後研究還有很長一段路，VLL 的大家要一起努力，做出好研究，發出好論文。

　　感謝家人朋友，會一直關心我的近況，維持我心理狀態的健康。在之前投稿結果出來的時候，也能夠一起分享喜悅。特別感謝媽媽努力工作，讓我和妹妹能夠專心的讀大學及研究所，雖然可能不太知道我都在幹嘛，但對於我的選擇都很支持。在各方面我還有許多不足的地方，這篇論文代表碩士生涯的結束，但不是研究的終點，接下來我也會繼續努力，做出屬害的研究。

<div align="right">2020.12.29 林志皓</div>

中文摘要

　　在自駕車、無人機、擴增實境等應用當中，電腦如何利用三維資料來感知空間資訊，是至關重要的問題。其中，點雲在三維電腦視覺領域是一種相當普遍且重要的資料型式。然而，不像是二維影像由規則的相素組成，點雲是由一堆點所形成的集合，能有效率地描述物體表面的資訊，卻也因為無規則排列，在分析處理上具有相當大的挑戰。

　　雖然近年來有許多研究團隊及學者，針對分析點雲提出了各種演算法，並也在一些簡單的任務達到了不錯的表現，然而這些演算法皆會在點雲被平移、縮放、旋轉等變化後，準確率會大為下降，這在許多應用當中都是不被允許的。在這篇論文中，我們提出了**三維圖卷積網路**，該演算法能夠在不同解析度下分析點雲的幾何特徵，經過適當的訓練，能夠進行點雲的分類、語意分割、室內場景辨識。藉由實驗在虛擬或是真實感測到的點雲資料上，我們驗證了三維圖卷積網路能夠用相對輕量化的模型，達到與其他團隊一樣甚至更好的表現，並在各種劇烈的平移、縮放下維持準確率不變。

Learning of 3D Graph Convolution Networks for Point Cloud Analysis

Zhi-Hao Lin

Advisor: Yu-Chiang Frank Wang

Graduate Institute of Communication Engineering

National Taiwan University

Taipei, Taiwan

December 2020

ii

Abstract

Point clouds are among the popular geometry representations in 3D vision. However, unlike 2D images with pixel-wise layouts, such representations containing unordered data points which make the processing and understanding the associated semantic information quite challenging. Although a number of previous works attempt to analyze point clouds and achieve promising performances, their performances would degrade significantly when data variations like shift and scale changes are presented. In this thesis, we propose **3D Graph Convolution Networks (3D-GCN)**, which is designed to extract local 3D features from point clouds across scales, while shift and scale-invariance properties are introduced. The novelty of our 3D-GCN lies in the definition of learnable kernels with a graph max-pooling mechanism. I show that 3D-GCN can be applied to 3D classification and segmentation tasks, with ablation studies and visualizations verifying the design of 3D-GCN.
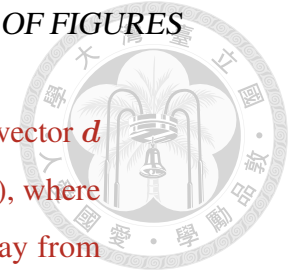
Contents

List of Figures

List of Tables

ix

Chapter 1

Introduction

3D vision has been an active research topic, closely related to applications such as augmented reality, drones and self-driving vehicles [4, 5]. Existing 3D data representations include the use of voxel, mesh or point cloud features. Voxels [6, 7] describe 3D objects as voxel grids but generally suffer from insufficient resolution and high memory costs. Meshes are common in animations, while such representations are not directly associated with the 3D sensor outputs [8, 9]. Finally, 3D point clouds focus on describing shape information of 3D objects and can be easily acquired by 3D sensors, but the resulting unordered set of 3D points might limit the subsequent analysis tasks.

With the recent remarkable progress of deep learning techniques, in particular the Convolutional Neural Network (CNN), promising performances have been observed in a variety of computer vision tasks [10, 11]. However, image data are generally presented in terms of grid structures (e.g., pixels or cells over scales), which makes the convolution operation feasible. For irregular and unstructured data like 3D point clouds, it is not possible to learn and deploy kernels with fixed sizes or patterns on such data. In order to process such an unstructured/unordered set of points, PointNet [1] applies multiple fully connected layers to encode 3D point clouds, followed by global max-pooling operation, and shows impressive results for 3D data recognition and segmentation. Since global pooling operation

1
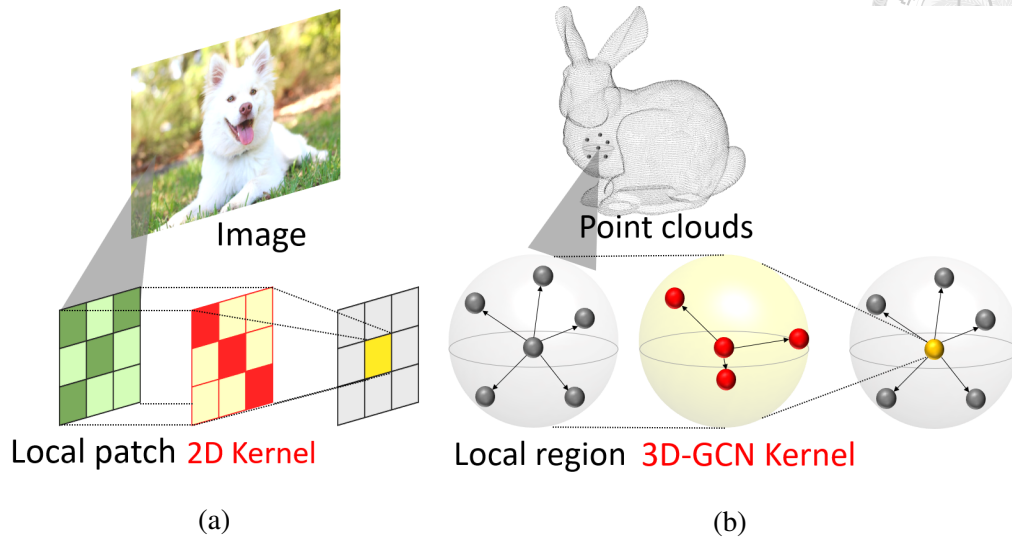
Figure 1.1: **Convolution in (a) 2D images and (b) 3D graphs.** Note that standard 2D CNN cannot be easily applied to handle 3D point cloud data, since the kernels in 3D graph convolution networks need to exhibit additional deformation in shape due to unstructured inputs.

is deployed, locally structured information of 3D point cloud might not be properly observed. Moreover, it is not designed to be invariant to global transformation like shift or scaling, as we later discuss and verify.

To better describe local information of 3D data, some research works utilize mesh data and their corresponding graph structures (i.e., vertices and associated edges) for extracting desirable information. For example, [12] choose to propagate and aggregate features of adjacent vertices for deriving the final representation, while others [8, 13] define kernels with fixed patterns to observe local information. Motivated by the above works, [14, 15, 16, 17, 2] attempt to construct graph-like structures for 3D point clouds. Since such methods consider global coordinates in representing their graphs, shift and scaling effects would degrade the performances of their models.

In this thesis, we propose a novel deep learning model of 3D Graph Convolution Networks (3D-GCN) for processing and learning structural information of 3D point clouds. Motivated by 2D CNN, we aim at deriving *deformable 3D kernels*, whose
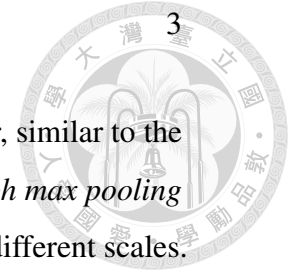
shape and weights are learnable during the training stage. Moreover, similar to the max pooling operation in standard CNN, we perform a unique *graph max pooling* operation in 3D-GCN to summarize the processed features across different scales. As a result, our 3D-GCN is able to observe and extract structural information of unordered 3D point clouds with arbitrary shape and size. As detailed and confirmed later, our 3D-GCN is invariant to 3D point cloud shift and scaling changes, which are the key properties for real-world 3D vision applications.

We now summarize our main contributions as follows:

- we propose a 3D Graph Convolution Network (3D-GCN) for processing 3D point cloud data, exhibiting shift and scale-invariant properties for promising classification/segmentation performances.

- The shape and weights of each kernel in our 3D-GCN are learnable during training, which show capabilities in describing local structural information from unordered 3D point clouds.

- A novel graph max pooling is also introduced in 3D-GCN, allowing extraction and summarization of point cloud features across different scales for improved performances.

Chapter 2

Related work

2.1 Multi-view and voxelized methods

Since standard Convolutional Neural Networks cannot be directly applied to handle unstructured data like 3D point clouds, existing works typically choose to convert point clouds into proper representation for further processing purposes. For example, [7, 6, 18, 19, 20] register point features into voxel grid or directly use voxelized 3D shapes as the model input, so that standard 3D CNN can be performed on such input data. Although octree-based methods like [21, 22] have been proposed to refine the resolution of the predicted output, voxel-based methods are generally known to suffer from insufficient resolution and enormous memory consumption for 3D voxel representation.

Alternatively, a number of works [23, 24, 25, 26, 16] choose to project 3D shapes onto 2D planes in multiple views, followed by 2D CNN for feature extraction. While impressive results are reported, it cannot be easily extended to 3D data segmentation or reconstruction [11, 10].

5

2.2   Point-cloud based methods

Various computer vision and computer graphic tasks involve shape analysis, which require a robust representation of local geometry. Since it is difficult to describe point cloud shapes due to the irregular structure, some previous works proposed various hand-crafted feature descriptors designed for different tasks and exhibit distinct properties. Some works uses descriptors which is derived from the point coordinates in 3D space, including methods like shape context [27], distance-based descriptors [28], point feature histograms [29, 30], and normal histograms [31]. A comprehensive overview of hand-designed feature descriptors could be found in [32, 33].

There are also learning-based methods. PointNet [1] has multiple shared fully connected layers to handle unordered 3D point inputs, followed by channel-wise max-pooling to extract global features to represent 3D point cloud data. While PointNet is able to handle 3D point cloud data without limitation to its unordered property, it essentially learns key point representation of the input object for deriving the final features. Local geometric information is not directly encoded, and this model would be sensitive to input translation and scaling variation, as we later verify.

To alleviate the aforementioned problem, researchers propose to sort the 3D points into an ordered list, where neighboring points have smaller euclidean distances in the 3D space. For example, [34] sorts all the points along different dimensions, then employ Recurrent Neural Networks (RNN) to extract features from the resulting sequences. [35, 36] convert the 3D points into an 1D list via kd-tree according to their coordinates, followed by 1D CNNs to extract the corresponding features. Nevertheless, sorting a 3D point set into an 1D list is not trivial; moreover, local geometric information may not be easily preserved in such ordered lists.
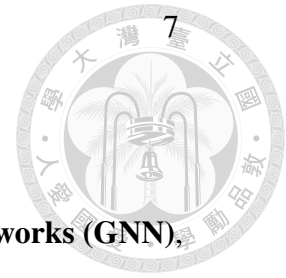
2.3 Geometry-based methods

Geometry-based methods are highly-related to **Graph Neural Networks (GNN)**, which extracts features from graphs composed of nodes and edges. Graphs are representative data structure but are tricky for processing due to its irregularity. [37] proposed **Graph Convolution Network (GCN)**, generalizing convolution operations over graphs, where features are updated by averaging over adjacent nodes. Some other works [38, 39, 40] proposed GCN variants with different feature aggregation algorithm. Since the details are out of the scope of this thesis, we refer readers to [41] for comprehensive discussion.

Aside from other applications, GCN could also be applied in shape analysis. [42, 43, 12] used GCN to extract features from the mesh object, where mesh is of graph structures. While point clouds shapes are not graphs since points are not connected, such edges could be constructed by considering various metrics, including euclidean distances. Different from the aforementioned works which take the entire 3D data as the input, another branch of methods [3, 2, 14, 15, 44, 45, 17, 46, 47] choose to learn local geometric information from a subset of 3D points. By dividing the 3D points into smaller groups, this type of approach extracts features from each local group for representation purposes. For example, PointCNN [45] learns to construct a transformation matrix for local points set, aligning points in a certain order, followed by 1D convolution operation. While good performances are reported in their work, the feature extraction strategy named $\mathcal{X}$-Conv are vulnerable to point order since the operation is not permutation-invariant. PointNet++ [3] divides 3D point clouds into several ball regions, and apply [1] to each ball for local feature extraction. DGCNN [2] dynamically constructs local graphs by identifying the nearest neighbors of points in the feature space, followed by the EdgeConv operation for feature extraction. Shen *et al.* [14] establish graphs according to euclidean distances and learn geometry information with kernel correlation. RS-CNN [15] applies weighted sum of neighboring point features, where each weight is learned with MLPs according to geometric relation between two points.

ShellNet [47] divides local points set into several shell area, aggregating features extracted from each shell. SPH3D [46] proposed discrete spherical kernels in 3D space, which is a deformed version of traditional 3D CNN. While the kernels in [46] are constructed in predefined structure, we proposed deformable kernels whose structure change in the training to describe various local geometry. These works attempt to extract geometrical information within local regions of 3D point clouds. However, existing methods typically use exact coordinates of points or distance vectors as the input features, and thus the model performance would be influenced by shifting and scaling effects, which would not be preferable for real-world applications like scene segmentation and multi-object detection. In this thesis, we propose a novel 3D-GCN with learnable 3D graph kernels and Graph Max-Pooling mechanism, resulting in effective geometric features across different scales while exhibiting scale and shift-invariance.

Chapter 3

3D Graph Convolution

In this thesis, we propose a 3D Graph Convolution Network (3D-GCN) for point cloud analysis. Our 3D-GCN is designed to process and analyze geometric patterns of 3D point clouds, while sharing several principles and properties with those of 2D Convolution Networks. In this section, we will introduce our proposed network, including the definition of 3D receptive fields representing local geometry of point clouds, learnable kernels, and the 3D convolution operation which enables the kernels to extract features from each receptive field. These components are used to construct model structures performing various tasks, which will be discussed in the following section.

3.1   Notations

For the sake of completeness, we define the notations used in this thesis as follows. A point cloud instance is viewed as a set, which contains a total of N points $\mathcal{P} = \{\boldsymbol{p}_n \mid n = 1, 2, \ldots, N\}$ located on the surface of an object of interest. Note that $\boldsymbol{p}_n$ denotes the n-th point of this instance, and its attributes may describe coordinates (x_n, y_n, z_n), normal vector $(\nu_n^x, \nu_n^y, \nu_n^z)$, or RGB color information (r_n, g_n, b_n). For example, if the point cloud describes only the coordinates of each point on the object surface, we have $\boldsymbol{p}_n = (x_n, y_n, z_n)$, and thus a 3D point cloud

9
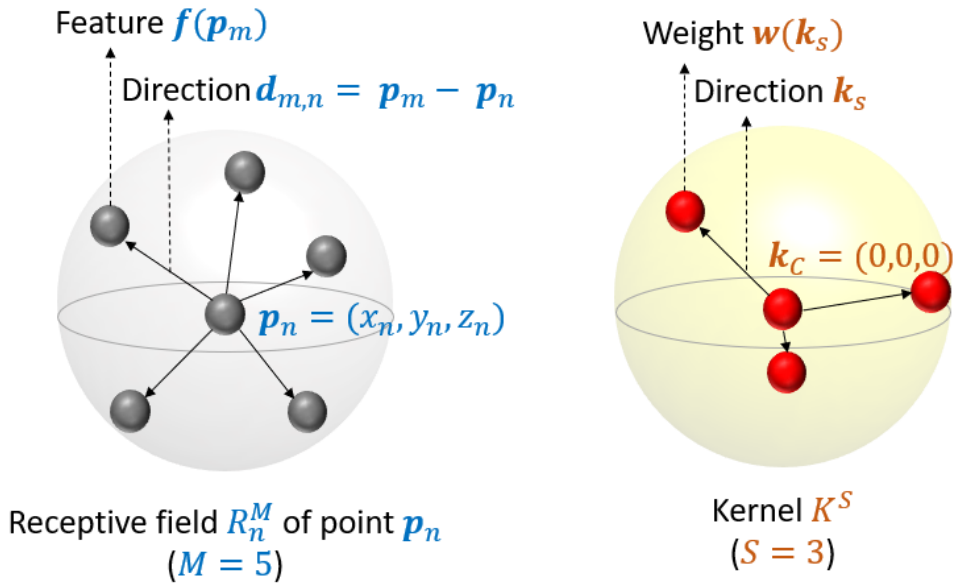
Figure 3.1: **Illustration of receptive field** R_n^M **and kernel** K^S**.** We have R_n^M indicates the M neighboring points for the nth point $\boldsymbol{p}_n$, and kernel K^S composes of S supports with center at $\boldsymbol{k}_C = (0, 0, 0)$. Note that directional vector $\boldsymbol{d}_{m,n}$ and $\boldsymbol{k}_s$ are used to measure the similarity in (3.4).

object is represented by a matrix of size $N \times 3$.

For classification tasks, 3D-GCN takes the point cloud input and produces the predicted output scores c for each class of interest. On the other hand, for semantic segmentation, we need to predict the part/scene label for each point in a 3D object. Thus, the output would be of size $N \times c$, which also indicates that the 3D-GCN performs point-wise classification in the task of semantic segmentation.

3.2   Receptive Fields in 3D-GCN

In 2D cases, square patches describe local patterns of images, and features are represented by either intensity, RGB value, or high dimensional vectors. Similarly, receptive fields of point clouds are defined to indicate various local geometry.

A 3D point cloud object with N points is denoted as $\mathcal{P} = \{\boldsymbol{p}_n \mid n = 1, 2, \ldots, N\}, \boldsymbol{p}_n \in \mathbb{R}^3$. To describe the feature derived at each point in 3D-GCN,
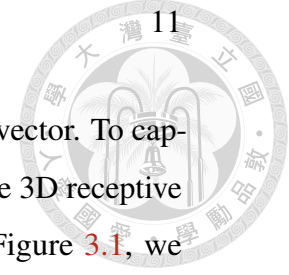
we have $\boldsymbol{f}(\boldsymbol{p}) \in \mathbb{R}^D$ denote the associated D-dimensional feature vector. To capture local geometric information of each point $\boldsymbol{p}_n$, we determine the 3D receptive field of $\boldsymbol{p}_n$ by a set of M neighboring points. As illustrated in Figure 3.1, we denote R_n^M as the receptive field of point $\boldsymbol{p}_n$ with size M as:

$$R_n^M = \{\boldsymbol{p}_n, \boldsymbol{p}_m \,|\, \forall \boldsymbol{p}_m \in \mathcal{N}(\boldsymbol{p}_n, M)\}, \tag{3.1}$$

where $\mathcal{N}(\boldsymbol{p}_n, M)$ denotes M nearest neighbors of $\boldsymbol{p}_n$ based on the euclidean distance $\|\boldsymbol{p}_m - \boldsymbol{p}_n\|_2$, and we have the corresponding *directional vector* $\boldsymbol{d}_{m,n} = \boldsymbol{p}_m - \boldsymbol{p}_n$ calculated for later convolution purposes, which indicates the relative position neighboring and center points. We note that, given a 3D point cloud object, we only need to determine the receptive fields for each point once. If the pooling operation is performed in later stages (as discussed in Sect. 4.2), receptive fields for the pooled point clouds at that scale needs to be constructed again. In 3D-GCN, the features within the receptive field of $\boldsymbol{p}_n$ with size M is expressed as $\{\boldsymbol{f}(\boldsymbol{p}_n), \boldsymbol{f}(\boldsymbol{p}_m) \,|\, \forall \boldsymbol{p}_m \in \mathcal{N}(\boldsymbol{p}_n, M)\}$. These features will be calculated and updated during the convolution operation, as later discussed.

It is worth noting that, recent models for 3D point clouds [3, 17] select neighboring points for each point within a sphere with predefined radius r. While parameter r can be tuned to properly describe the local structures of 3D point clouds, their models are vulnerable to different point density, and cannot deal with scale variations as we later discuss and verify in Sect. 5.

3.3 Learnable Kernels in 3D-GCN

In standard 2D CNN models, the kernel is composed of weight parameters in grids, sharing the same patterns across image patches (as depicted in Figure 1.1a). However, for 3D point cloud data, the data points are viewed as an unordered set, and it is limited to extract diverse geometric features with fixed-structured kernels.

To perform convolution in 3D point cloud structures, we propose deformable *3D Graph Convolution Kernel* K^S, where S denotes the number of supports in that
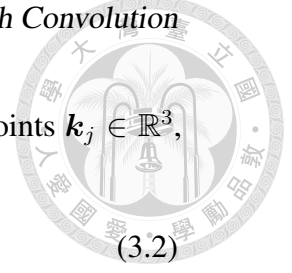
kernel. More precisely, we have K^S composed of $S + 1$ kernel points $\boldsymbol{k}_j \in \mathbb{R}^3$, i.e.,

$$K^S = \{\boldsymbol{k}_C, \boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_S\}. \tag{3.2}$$

where $\boldsymbol{k}_C = (0, 0, 0)$ is the center of the kernel, and $\boldsymbol{k}_1$ to $\boldsymbol{k}_S$ denote the associated supports. Directional vectors $\boldsymbol{k}_s = \boldsymbol{k}_s - \boldsymbol{k}_C, s = 1, 2, \ldots, S$ are utilized in the convolution operation.

In 2D CNN, each element in a kernel is the learned weight which describes the spatial patterns of interest. In our 3D-GCN, we define the weight vector $\boldsymbol{w}(\boldsymbol{k}) \in \mathbb{R}^D$ for each kernel point $\boldsymbol{k}$. Thus, the weighted-sum of features $\boldsymbol{f}(\boldsymbol{p})$ using the corresponding weights would achieve the convolution operation. As illustrated in Figure 3.1, except for $k_C = (0, 0, 0)$, the kernel in 3D-GCN is now defined as $\{\boldsymbol{w}(\boldsymbol{k}_C), (\boldsymbol{k}_s, \boldsymbol{w}(\boldsymbol{k}_s)) \,|\, s = 1, 2, \ldots, S\}$, where each element is learned via training. Kernels are deformable due to the learnable directional vectors $\boldsymbol{k}_s$, enabling 3D-GCN to extract complicated patterns.

Compared to 2D image patches and the CNN kernels are grid-structured, the receptive fields R_n^M and the learnable kernels K^S in 3D cloud can be viewed as simple graphs. Graph nodes are composed of point features $\boldsymbol{f}(\boldsymbol{p})$ or weight vectors $\boldsymbol{w}(\boldsymbol{k})$. The nodes in such graphs are connected by edges, which are directional vectors $\boldsymbol{d}_{m,n}$, $\boldsymbol{k}_s$, indicating the relations between nodes. Note that the support number S of a kernel is not necessarily equal to the neighbor number M of a receptive field R_n^M, while convolution operation is well-defined in the Sect. 3.4.

3.4   3D Graph Convolution

In 2D CNN, the convolution operation can be regarded as calculating the *similarity* between the 2D kernel and the associated image patch. Larger output values indicate higher visual similarity, describing strips, curves, or semantic patterns. With the aforementioned receptive field and kernel definitions for 3D point cloud data, we define 3D Graph Convolution by calculating the similarity between R_n^M
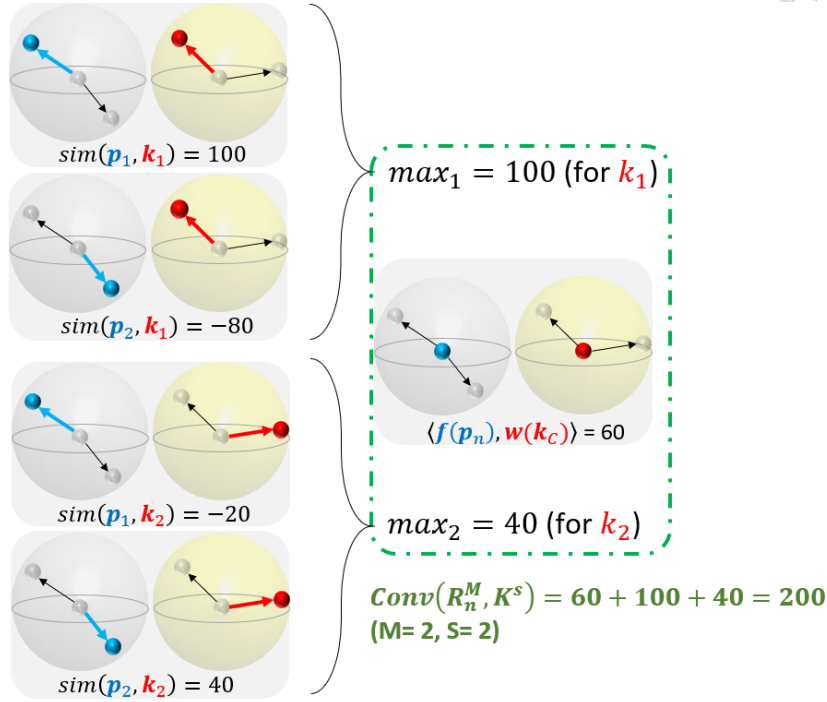
Figure 3.2: **3D Graph Convolution.** As in (3.4), $sim(\boldsymbol{p}_m, \boldsymbol{k}_s)$ calculates the inner product between $\boldsymbol{f}(\boldsymbol{p}_m)$ and $\boldsymbol{w}(\boldsymbol{k}_s)$ based on the cosine similarity between $\boldsymbol{d}_{m,n}$ and $\boldsymbol{k}_s$. For each support $\boldsymbol{k}_s$, the largest sim output among all neighbors $\boldsymbol{p}_m$ is obtained. Summing up with $\langle \boldsymbol{f}(\boldsymbol{p}_n), \boldsymbol{w}(\boldsymbol{k}_C) \rangle$ produces the final convolution output (i.e., (3.5)).

and K^S, denoted as $Conv(R_n^M, K^S)$.

However, unlike 2D CNN in which both kernel and image patches are of the same grid structures, it is not trivial to perform convolution in 3D graph structures, where no apparent one-to-one relations between graphs are provided. Thus, to measure the similarity between the features within the receptive field R_n^M (i.e., $\boldsymbol{f}(\boldsymbol{p}_n), \boldsymbol{f}(\boldsymbol{p}_m), \forall \boldsymbol{p}_m \in \mathcal{N}(\boldsymbol{p}_n, M)$ as defined in (3.1)) and weight vectors of kernel K^S centered at $\boldsymbol{k}_C$ with S supports (i.e., $\boldsymbol{w}(\boldsymbol{k}_C), \boldsymbol{w}(\boldsymbol{k}_s), \forall s = 1, 2, \ldots, S$), we consider all possible pairs between $(\boldsymbol{p}_m, \boldsymbol{k}_s)$. Thus, $Conv(R_n^M, K^S)$ in 3D-GCN is defined as:

$$Conv(R_n^M, K^S) = \left\langle \boldsymbol{f}(\boldsymbol{p}_n), \boldsymbol{w}(\boldsymbol{k}_C) \right\rangle + g(\mathcal{A}), \tag{3.3}$$
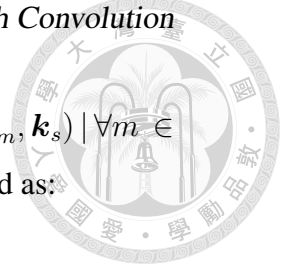
where $\langle \cdot \rangle$ denotes the inner-product operation, and $\mathcal{A} = \{sim(\boldsymbol{p}_m, \boldsymbol{k}_s) \mid \forall m \in (1, M), \forall s \in (1, S)\}$. Note that the $sim()$ function in (3.3) is defined as:

$$sim(\boldsymbol{p}_m, \boldsymbol{k}_s) = \left\langle \boldsymbol{f}(\boldsymbol{p}_m), \boldsymbol{w}(\boldsymbol{k}_s) \right\rangle \frac{\left\langle \boldsymbol{d}_{m,n}, \boldsymbol{k}_s \right\rangle}{\|\boldsymbol{d}_{m,n}\| \|\boldsymbol{k}_s\|}, \tag{3.4}$$

which calculates the inner product between $\boldsymbol{f}(\boldsymbol{p}_m)$ and $\boldsymbol{w}(\boldsymbol{k}_s)$ based on their cosine similarity $\cos\theta = \frac{\left\langle \boldsymbol{d}_{m,n}, \boldsymbol{k}_s \right\rangle}{\|\boldsymbol{d}_{m,n}\| \|\boldsymbol{k}_s\|}$. It can be observed that large $sim()$ value results from large inner product in feature space $\mathbb{R}^D$ and similar direction in euclidean space $\mathbb{R}^3$. The function g in (3.3) sums up the maximum similarity $sim(\boldsymbol{p}_m, \boldsymbol{k}_s)$ among all neighboring points for each support $\boldsymbol{k}_s$ in that kernel. With the above definitions, the 3D Graph Convolution operation in our 3D-GCN is derived as:

$$\begin{aligned} Conv(R_n^M, K^S) &= \left\langle \boldsymbol{f}(\boldsymbol{p}_n), \boldsymbol{w}(\boldsymbol{k}_C) \right\rangle \\ &+ \sum_{s=1}^{S} \max_{m \in (1, M)} \left\{ sim(\boldsymbol{p}_m, \boldsymbol{k}_s) \right\}. \end{aligned} \tag{3.5}$$

Recall that the neighbors M and supports S are the hyper-parameters (similar to the kernel sizes in 2D CNN), and are flexible to be unequal in our formulation. Figure 3.2 illustrates 3D Graph Convolution operation in our 3D-GCN.

3.5   Property analysis

3D graph convolution kernel K^S is applied to every receptive fields R_n^M centered at each points $\boldsymbol{p}_n$, exhibiting **weight-sharing** property, which is also essential in traditional 2D CNN. Some previous works [35, 45] extract permutation-dependent point features, however, 3D-GCN formulates receptive fields via nearest-neighbor algorithm, and take advantages of symmetric functions such as $\boldsymbol{max}(), \boldsymbol{sum}()$ in 3.5, these operations are **permutation-invariant**, which is desirable in point cloud analysis.

While existing works like [15, 1, 3, 14, 17] report promising performances, they typically consider global coordinates or require point cloud normalization to alleviate such data variances, which would limit their invariance properties

Normalized 3D object   Normalized 3D object
with noisy data
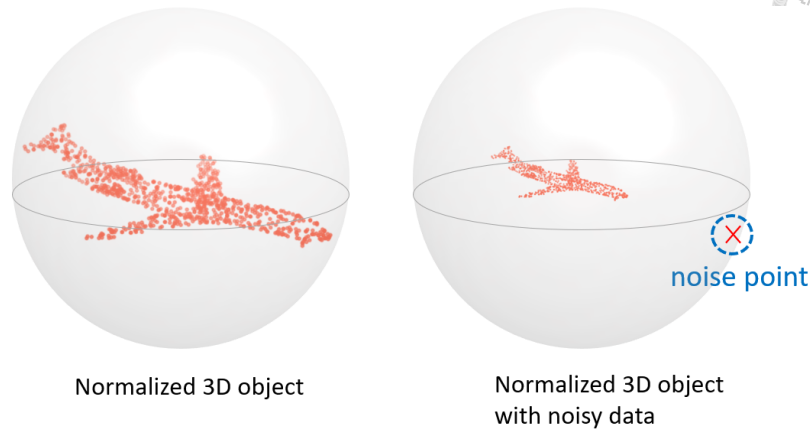
Figure 3.3: **Illustration of the lack of invariance property.** Recent models like PointNet [1] require techniques like zero-mean normalization for 3D point cloud representation, which might be sensitive to noisy 3D input points (as verified in Sect. 5).

(see examples in Figure 3.3). On the other hand, convolution formulation (3.5) of 3D-GCN utilizes directional vectors $\boldsymbol{d}_{m,n} = \boldsymbol{p}_m - \boldsymbol{p}_n$ within R_n^M instead of global coordinates, which introduces **shift-invariant** property. In addition, the similarity function in (3.4) simply calculates the cosine similarity between $\boldsymbol{d}_{m,n}$ and $\boldsymbol{k}_s$ regardless of their lengths. Therefore, the **scale-invariant** property can be jointly observed by our 3D-GCN.

Last but not the least, 3D-GCN is more robust to shape rotations, compared to other convolution-based methods. [3, 2, 17] generally take global and/or relative coordinates between 3D points as input features, and rotation would result in the significant differences in their convolution outputs. Take Fig. 3.4 with support S = 1 for our learned kernel $\boldsymbol{k}$ as an example, if $\boldsymbol{k}$ is close to the z-axis, rotation in yaw (with respect to z-axis) would produce negligible difference when calculating cosine similarity $\cos \theta = \frac{\langle \boldsymbol{d}, \boldsymbol{k} \rangle}{\|\boldsymbol{d}\|\|\boldsymbol{k}\|}$. On the other hand, if k is away from z-axis, small rotation variations would not result in significant changes in the associated cosine similarities. The above observations can be supported by results and comparisons in Sect. 5.
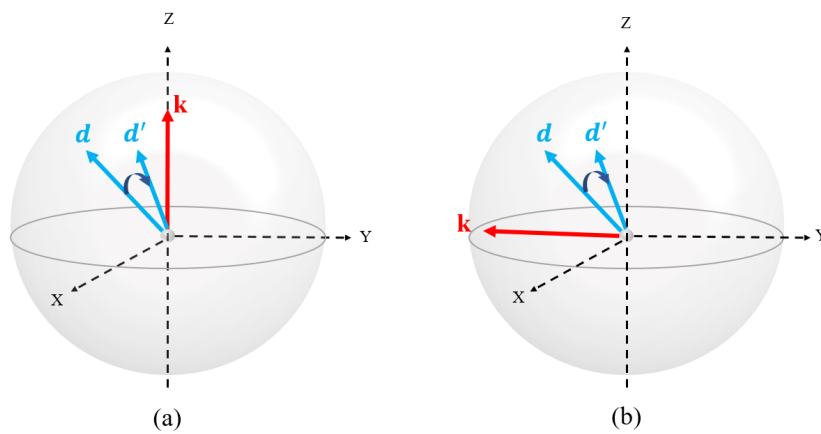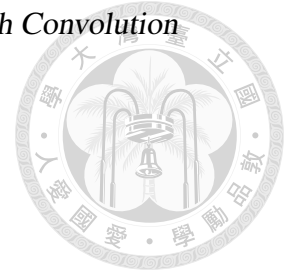
Figure 3.4: **Illustration of rotation effects on 3D-GCN** Directional vector d of a 3D point and the kernel k (with support number as 1), where (a) and (b) consider the learned kernels in along and away from z-axis, respectively.

Chapter 4

3D Graph Convolution Network

With the fundamental elements of 3D-GCN defined in the previous section, we now explain how the convolution and pooling are performed for processing 3D point cloud data, and how the classification and segmentation tasks are conducted.

4.1   3D Graph Convolution

A 3D Graph Convolution layer is composed of a pre-determined number L of kernels K^S, each with a constant support number, which can be denoted as $\{K_i^S | i = 1, 2, \ldots, L\}$, and has large convolution response with diverse geometry and semantics. Taking the 3D point cloud input $\mathcal{P} \in \mathbb{R}^{N \times 3}$ with the corresponding D-dimensional features $\mathcal{F}^{in} \in \mathbb{R}^{N \times D}$, our 3D Graph Convolution Layer applies (3.5) with each kernel respectively, producing output features $\mathcal{F}^{out} \in \mathbb{R}^{N \times L}$. Thus, each output channel $i = 1, 2, \ldots, L$ can be expressed as:

$$ConvLayer((\mathcal{P}, \mathcal{F}^{in}), K_i^S) = (\mathcal{P}, \mathcal{F}_i^{out}), \tag{4.1}$$

where $\mathcal{F}_i^{out} \in \mathbb{R}^{N \times 1}$ is the i-th channel of $\mathcal{F}^{out}$. To initialize the convolution and learning process for point clouds without intrinsic feature $\mathcal{F}^{in}$, we simply set $\boldsymbol{f}(\boldsymbol{p}) = 1, \forall \boldsymbol{p} \in \mathcal{P}$, and $\boldsymbol{w}(\boldsymbol{k}) = 1, \forall \boldsymbol{k}$ in the first input layer. In other words, we only consider directional information to initialize 3D-GCN operations.

17
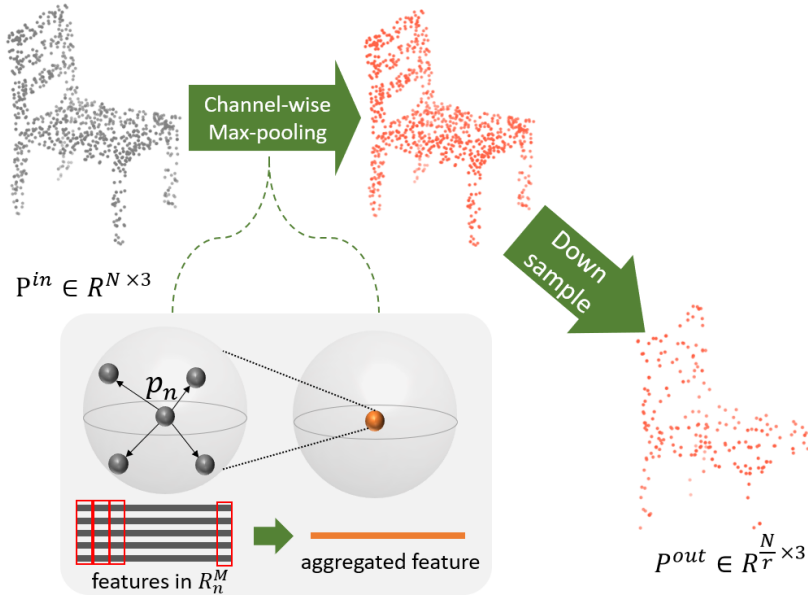
Figure 4.1: **Graph Max-Pooling.** This pooling process performs channel-wise max-pooling from the features in the receptive field of each $\boldsymbol{p}_n \in \mathcal{P}^{in}$, followed by randomly sampling a subset from $\mathcal{P}^{in}$ with a sampling rate r.

4.2 3D Graph Max-Pooling

Pooling operation plays an important role in standard 2D CNN, which summarizes the dominant responses within each scale for later high-level processing purposes, resulting in fine-to-course feature extraction. In 3D-GCN, we also propose a down-pooling operation, 3D Graph Max-Pooling, for performing similar mechanisms in 3D point clouds.

Our 3D Graph Max-Pooling layer takes the receptive field of each point R_n^M, and applies channel-wise max-pooling to aggregate features $\boldsymbol{f}(\boldsymbol{p}), \forall \boldsymbol{p} \in R_n^M$, which could be formulated as:

$$\boldsymbol{f}'_i(\boldsymbol{p}_n) = \max\{\boldsymbol{f}_i(\boldsymbol{p}) \,|\, \forall \boldsymbol{p} \in R_n^M\}, i = 1, 2, \ldots, D \tag{4.2}$$

where the aggregated feature $\boldsymbol{f}'(\boldsymbol{p}_n) \in \mathbb{R}^D$ contains geometric information in a larger scale.

The aggregation operation is followed by sampling a subset of $\mathcal{P}$ with a sampling rate r. Some works [3, 45, 48] utilized Farthest Point Sampling (FPS)

technique, which iteratively select points according to its distance to others. While guaranteeing more uniform sampling results, such algorithm is time-consuming due to heavy computational complexity, and is not feasible in real-world applications. In 3D-GCN, we followed [49] and use **Random Sampling** technique, where the subset points are sampled randomly. The technique accelerates the process of point cloud process, which is more desirable for large point clouds, such as those sampled from scenes. Thus, this pooling process can be formulated as:

$$PoolLayer_r(\mathcal{P}^{in}, \mathcal{F}^{in}) = (\mathcal{P}^{out}, \mathcal{F}^{out}), \tag{4.3}$$

where $\mathcal{P}^{in} \in \mathbb{R}^{N \times 3}, \mathcal{P}^{out} \in \mathbb{R}^{(N/r) \times 3}$, and $\mathcal{F}^{in} \in \mathbb{R}^{N \times D}, \mathcal{F}^{out} \in \mathbb{R}^{(N/r) \times D}$. As depicted in Figure 4.1, this pooling layer enables us to learn multi-scale 3D point cloud features, and make learning and calculation more efficient, which are crucial factors in 3D deep learning models.

4.3 Classification Model

To train 3D-GCN to recognize 3D point cloud data as particular categories, we apply and combine multiple 3D graph convolution and max-pooling layers. After applied with the last 3D graph convolution layer, model obtains representative feature by global max-pooling, which is followed by adding multi-layer perceptron (MLP) for predicting the desirable outputs. Standard soft-max losses and back-propagation can be calculated to learn such 3D-GCN models (see Figure 4.2a for example architectures).

4.4 Semantic Segmentation Model

Semantic segmentation task had been noticed for decades in the fields of computer vision. In 2D world, previous works [50, 51, 52] proposed model aggregating cross-scale features from images, and achieve outstanding performance. Motivated by them, we proposed a U-shaped model structure, illustrated in Fig. 4.2b. Model
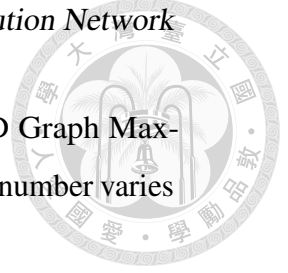
learns features in various scales by down sampling points with 3D Graph Max-pooling layer. However, fusing such features is not trivial since point number varies in different scales due to the pooling mechanism.

We provided a solution as following. For example, features of two scales are noted as points sets $\mathcal{P}_1 \subset \mathcal{P}_2$. For each points in $\mathcal{P}_2$, feature is updated by:

$$\boldsymbol{f}'(\boldsymbol{p}_2) = \Big(\boldsymbol{f}(\boldsymbol{p}_1), \boldsymbol{f}(\boldsymbol{p}_2)\Big), \forall \boldsymbol{p}_2 \in \mathcal{P}_2 \qquad (4.4)$$

$$\boldsymbol{p}_1 = \arg\min_p \{\|\boldsymbol{p} - \boldsymbol{p}_2\| \,|\, \forall \boldsymbol{p} \in \mathcal{P}_1\} \qquad (4.5)$$

where point features in coarse scale ($\mathcal{P}_1$) are **up-sampled** to the fine scale ($\mathcal{P}_2$). As a result, features from different scales are able to be fused via concatenation, followed by next 3D graph convolution layer. Decoder architecture of model take advantage of this mechanism, generating final dense prediction for each points. Experiment results in Sect. 5 validates the high performance of the model.

(a) **Classification**

(b) **Part segmentation**

Figure 4.2: **Architecture of 3D-GCN for (a) classification and (b) part segmentation.** Note that grey and yellow blocks denote point and feature inputs, respectively. Green arrows denote 3D Graph Convolution Layers, while green triangles denote the Graph Max-Pooling layer. We have MLP and outputs denoted in brown and blue, respectively. For part segmentation in (b), blocks in pink denote the up-sampled feature maps from the consecutive layer, which are concatenated with those at the layer of interest (shown in yellow) as the feature map for performing 3D graph convolution.

Chapter 5

Experiments

5.1 3D Model Classification

5.1.1 Dataset

We evaluate 3D-GCN for 3D shape classification on the ModelNet40 [7] dataset, which consists of 12311 CAD models of 40-categories, splitting into 9843 3D objects for training and 2468 for testing. To generate point clouds for training and testing, we sample 1024 points uniformly from the surface of each object without any normalization during training and testing.

5.1.2 Network configuration

Our 3D-GCN model structure for classification is shown in Figure 4.2a. The feature-extracting part is composed of 5 3D graph convolution layers, with kernel numbers $(32, 64, 128, 256, 1024)$ from low to high-level layers. We set the support number $S = 1$ for our kernels, and neighbor number $M = 25$ for the receptive fields. There are 2 3D Graph max-pooling layers in model structure, all with a fixed sample ratio $r = 4$. Following PointNet [1], the output feature of last 3D Graph Convolution Layer in our 3D-GCN is applied with global max-pooling, resulting a final feature representation of 1024 dimension. For classification, the MLP is of 2

Method	input	#points	Acc.(%)
ECC [12]	xyz	1k	87.4
PointNet [1]	xyz	1k	89.2
Kd-Net (depth=10) [35]	xyz	1k	90.6
PointNet++ [3]	xyz	1k	90.7
KCNet [14]	xyz	1k	91.0
MRTNet [36]	xyz	1k	91.2
DGCNN [2]	xyz	1k	92.9
SO-Net [53]	xyz	2k	90.9
KPConv rigid [17]	xyz	6.8k	92.9
SPH3D-GCN [46]	xyz	10k	92.1
PointNet++ [3]	xyz, normal	5k	91.9
SO-Net [53]	xyz, normal	5k	93.4
Ours	xyz	1k	92.1

Table 5.1: **Shape classification results on ModelNet40.** Note that "normal" denotes the normal vectors of object surfaces. We see that our method achieved comparable or improved results with inputs of size only 1k points.

layers, where Batchnorm and Dropout with drop ratio of 0.3 applied after the fist layer of MLP. We train our network with batch size 8, learning rate 0.0001 which is decayed half every 10 epochs, using the ADAM optimizer.
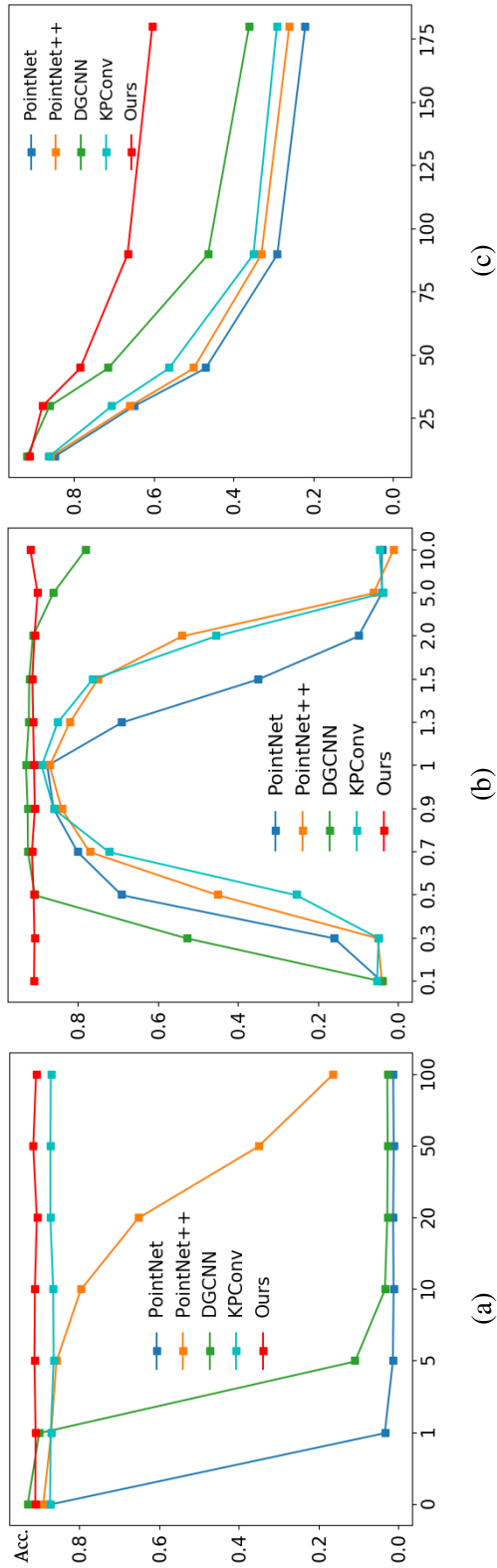
Figure 5.1: **Evaluation of invariance properties on ModelNet40.** **(a) Shift:** Objects randomly shifted within a distance along all directions (with unshifted version denoted as 0), **(b) Scale:** Objects scaled to different sizes (with the original size denoted as 1), **(c) Rotation:** Objects rotated along the upward direction (degree is denoted in this figure). Note that DGCNN in [2] was pre-trained on objects with scale variants (i.e., scale within [0.5, 1.5]), but it cannot handle unseen scale variants as shown in (b).

5.1.3   Results and Discussions

The classification results of our 3D-GCN are listed in Table 5.1, in which we also compare our results with a number of recent approaches including PointNet [1], PointNet++ [3], DGCNN [2] and KPConv [17]. From this table, we see that our 3D-GCN is generally comparable or performs favorably against several state-of-the-art models when the test data are without any shift or scale variations presented.

To further evaluate the invariance properties of our model, we compare to the above models using 3D point cloud data with 1024 points, normalized to a unit sphere with zero mean, without data augmentation. We test them under three different situations: coordinate shift, shape scaling, and shape rotation. The results are shown in Figure 5.1a, 5.1b, and 5.1c, respectively. From the results shown in these figures, we see that the performance of PointNet and DGCNN significantly dropped with coordinate shifts, which is caused by extracting features from global coordinates. When scale variants are presented, only our model was able to perform recognition with satisfactory performances. As for shape rotation, better invariance ability was exhibited by our 3D-GCN.

We also test the robustness of our 3D-GCN to the presence of outlier points, and show the results in Figure 5.2. Note that methods to be compared in Figure 5.2 require point cloud data to be normalized in a unit sphere before performing inference. Thus, as illustrated in Figure 3.3, such methods were not able to produce satisfactory results. On the other hand, 3D-GCN does not require such normalization during feature extraction. Therefore, our classification accuracy was clearly above those reported by other approaches (see Figure 5.2a).

5.2 3D Model Part Segmentation

5.2.1 Dataset

To evaluate the use of 3D-GCN for part segmentation, we consider the ShapeNet-Part dataset [54], which consists of 16881 CAD models from 16 object types, with each point in an object corresponding to a part label. With a total of 50 categories, 2 to 6 part categories are available for each object type. In our work, we sample 2048 points from each 3D model for training and testing.

5.2.2 Network configuration

The model architecture is shown in Figure 4.2b. The feature-extracting part is composed of 9 layers with 256 kernels, and two 3D Graph Max-pooling layers with a fixed sample ratio $r = 4$ are deployed. In the decoder architecture, features from coarse scale are up-sampled and concatenated, which is formulated as equation 4.4. We set the support number $S = 1$ for each kernel, and neighbor number $M = 50$ for the receptive field in 3D-GCN. Following [1, 2], we also have one-hot vectors indicating object type concatenated to the features, while single model was trained to classify 50 part types. We train the 3D-GCN with learning rate 0.001 and decayed half every 10 epochs, using the ADAM optimizer.

5.2.3 Results

We evaluate the segmentation performance in terms of mean intersection over union (mIoU), which is the average IoU of each part type in that object category. Note that the mIoU of each category is calculated by averaging mIoUs of all the shape instances. More specifically, class mIoU is the average of mIoU over all 16 categories, while instance mIoU is the average of mIoU over all instances. The part segmentation results are listed in Table 5.2. Note that without using global coordinates, our 3D-GCN achieved comparable or better results than recent

approaches did.

Furthermore, we demonstrate the robustness of 3D-GCN by visualizing the segmentation result under different transformations of an object. We shift the center/coordinates of each object by 100 and enlarge object size by 10 times, and Table 5.3 compares our segmentation visualization with others. We found that KPConv [17] and PointNet++ [3] failed to properly segment the corresponding parts in all cases. On the other hand, our 3D-GCN exhibited very promising invariance capabilities regardless of shift and scale variations. The quantitative results are reported in Table 5.4.
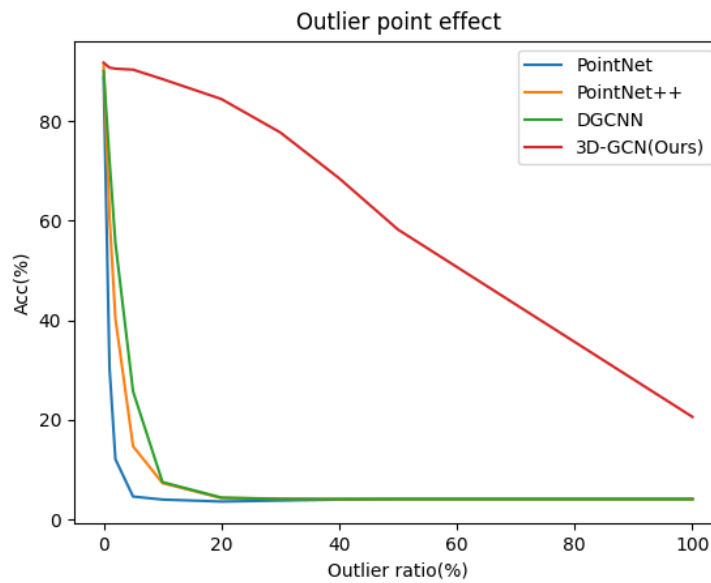
(a) Classification accuracy with outliers.

| Original | Ratio=5% | Ratio=10% | Ratio=50% | Ratio=100% |
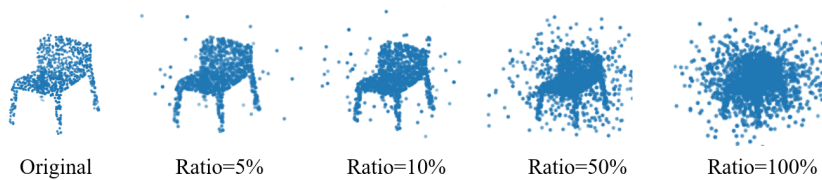
(b) Point cloud visualization with outliers.

Figure 5.2: **Effects on the presence of outlier points for the ModelNet40 dataset.** Outlier points of different ratio numbers are added to the 3D point cloud input. Take a point cloud input with 1000 points for example, 10% indicates additional 100 outliers introduced. Note that all the outlier points are sampled from a fixed Gaussian distribution.

Method	class mIoU	instance mIoU	air plane	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor bike	mug	pistol	rocket	skate board	table
Kd-Net [35]	77.4	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	84.9	87.4	86.7	78.1	51.8	69.9	80.3
MRTNet [36]	79.3	83.0	81.0	76.7	87.0	73.8	89.1	67.6	90.6	85.4	80.6	95.1	64.4	91.8	79.7	87.0	69.1	80.6
PointNet [1]	80.4	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
KCNet [14]	82.2	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
RS-Net [34]	81.4	84.9	82.7	86.4	84.1	78.2	90.4	69.3	91.4	87.0	83.5	95.4	66.0	92.6	81.8	56.1	75.8	82.2
SO-Net [53]	81.0	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++ [3]	81.9	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
DGCNN [2]	82.3	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
KPConv deform [17]	85.1	86.4	84.6	86.3	87.2	81.1	91.1	77.8	92.6	88.4	82.7	96.2	78.1	95.8	85.4	69.0	82.0	83.6
Ours	82.7	85.3	82.8	86.1	84.8	79.2	91.1	74.9	91.6	87.4	83.6	95.8	69.3	94.9	82.4	61.1	75.6	82.2

Table 5.2: **Part segmentation results on ShapeNetPart.** Note that while our method achieved comparable results as state-of-the-art models did, our model complexity was significantly less than others as discussed in Sect. 6.5.

Table 5.3: **Visualization of part segmentation on ShapeNetPart.** We compare our segmentation results with those produced by PointNet++ [3] and DGCNN [2]. In addition, shift (by 100), scale (by 10 times) and rotation (by 30 degree) variations are presented for evaluating the invariance capacity for each model.

Method	Shift					Scale					Rotation				
	1	5	10	50	100	0.1	0.5	1.5	5	10	30	60	90	120	150
PointNet [1]	23.7	17.1	16.1	16.0	15.8	30.8	66.6	73.1	40.8	35.1	68.0	58.5	54.5	50.8	50.0
PointNet++ [3]	43.3	27.9	23.9	17.6	15.1	29.1	54.7	73.8	38.8	36.2	74.5	66.6	62.0	58.8	58.0
DGCNN [2]	45.5	22.4	19.4	16.8	16.1	37.9	69.2	76.9	50.5	27.4	72.5	67.0	64.2	61.1	60.6
KPConv [17]	36.8	23.1	22.2	21.1	20.9	30.5	46.0	67.3	51.2	48.9	72.2	61.1	51.4	47.2	42.7
Ours	82.2	82.5	82.1	82.5	82.5	82.4	82.2	82.4	82.4	82.4	80.6	74.9	68.6	65.5	65.0

Table 5.4: **Part segmentation in terms of class mIoU with shift, scale and rotation variations.** Note that the 3D model is rotated around the **y**-axis, which is the upward direction.

5.3   Scene Segmentation

5.3.1   Dataset

We consider the Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS) [55], which consists of point clouds sampled from 6 different indoor area, containing 272 rooms in total. Each point belongs to one of 13 semantic categories, including ceiling, floor, chair, wall ... etc. We followed the setting of [1, 2], where all indoor areas are split into $1m \times 1m$ blocks, and each point is represented as a 9D vector of XYZ, RGB and normalized location in the room (from 0 to 1). 4096 points for each block are sampled during training and testing.

5.3.2   Network configuration

The model architecture is shown in Figure 4.2b, which is very similar to the one used in part segmentation task. The model is composed of 9 convolution layers with 160 kernels and 2 graph max-pooling layers. Batch normalization is applied after each convolution layer, and batch size equals 4 during training. We trained the model with learning rate 0.0001 and ADAM optimizer for total 50 epochs.

5.3. Scene Segmentation

Area 5

Method	Accuracy	mIoU(*)	mIoU	ceiling (21.7%)	floor (18.9%)	wall (27.4%)	beam (0.0%)	column (1.4%)	window (3.0%)	door (3.4%)	table (3.5%)	chair (2.0%)	sofa (0.3%)	bookcase (8.7%)	board (1.1%)	clutter (8.4%)
PointNet [1]	79.0	-	42.2	88.9	96.3	69.1	0.4	6.2	42.7	14.0	56.1	51.0	24.3	44.6	22.1	33.1
PointNet++ [3]	83.8	-	51.5	91.4	97.9	74.3	0.0	3.7	48.9	36.3	69.4	76.2	26.5	53.5	49.3	41.9
DGCNN [2]	83.2	-	49.0	91.1	97.3	74.5	0.0	11.9	49.5	33.5	66.9	69.4	20.5	47.5	34.7	40.8
Ours	84.6	51.9	51.9	91.4	97.1	75.9	0.1	22.3	43.5	30.1	71.5	79.4	21.9	53.7	42.9	44.9

6-fold

Method	Accuracy	mIoU(*)	mIoU	ceiling (21.4%)	floor (19.0%)	wall (25.6%)	beam (1.8%)	column (1.7%)	window (2.0%)	door (5.5%)	table (3.0%)	chair (3.6%)	sofa (0.4%)	bookcase (5.1%)	board (1.1%)	clutter (9.9%)
PointNet [1]	79.7	47.7	49.9	89.5	93.9	68.3	42.0	25.9	45.2	52.1	53.9	41.2	24.0	41.5	32.7	38.1
PointNet++ [3]	83.5	-	57.6	91.7	93.9	73.5	54.7	20.7	53.0	57.0	63.0	59.3	36.4	49.0	49.2	47.0
DGCNN [2]	83.4	56.1	56.0	92.0	94.5	73.9	50.4	32.5	54.9	59.2	62.5	53.9	16.6	45.7	46.3	45.4
Ours	85.8	60.8	60.8	91.7	95.5	77.2	53.0	38.4	52.3	59.0	67.6	70.8	28.1	51.5	51.9	53.2

Table 5.5: **Scene segmentation on S3DIS.** The results of area 5 and 6-fold are in the upper and lower tables, respectively. All the methods are reproduced with same data processing, training and evaluation procedures, while the results of original paper are also shown in column **mIoU(*)**. For each object category, the proportion of the whole dataset is denoted as $(x\%)$, with the IoU scores are shown in each entry.
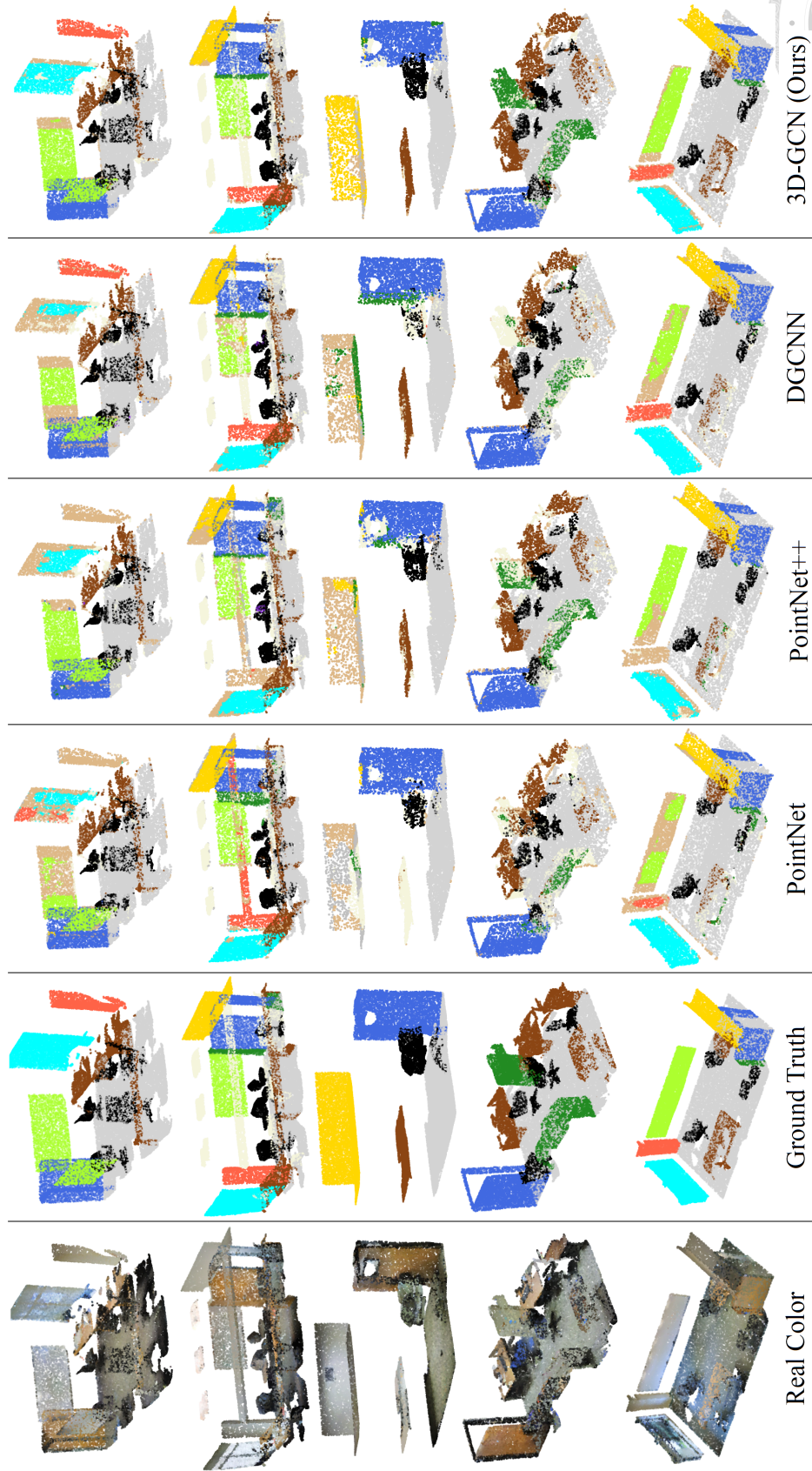
Table 5.6: **Visualization of scene segmentation.** From left to right, we show the input scene, ground truth segmentation, the results producd by PointNet [1], PointNet++ [3], DGCNN [2], and 3D-GCN, respectively. Note that each semantic category is illustrated in a distinct color, while the categories of **ceiling**, and **wall** are not shown for visualization clarity.
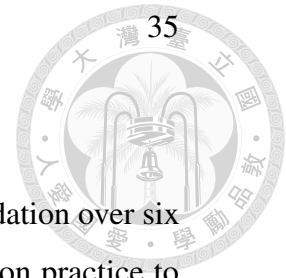
5.3.3 Results

We evaluated the performance on Area 5 explicitly and 6-fold validation over six areas, and results are shown in Table 5.5. Note that it is a common practice to separately analyze performance on Area 5, because this area is in the building that does not contain the other five areas, therefore it could be used to measure the generalization ability. For [1, 3, 2], we showed the performance reported in their paper, and the reproduced results are also listed. To reproduce, data processing and training details followed the same settings for fairness. We compare the overall accuracy, IoU score for each categories, and their mean (*i.e.* mIoU) in Table 5.5.

Since 3D-GCN is able to describe geometrical information with deformabel kernels, and provided properties such as shift- and scale-invariance, our algorithm is preferable in scene segmentation tasks, where multiple objects are contained in a scene and normalization operation can not be applied to each object. From Table 5.5, all methods perform similar on ceiling and floor, since they can be easily classified by the location in the room. However, 3D-GCN outperforms [1, 3, 2] with a large margin on categories like column, table and chair, which share category-related shape information but are distributed in different positions in rooms. Furthermore, our overall mIoU score outperforms others, which indicates that 3D-GCN has stronger capability to recognize local geometry.

Chapter 6

Ablation Study

6.1 Neighbor number M in receptive fields.

We now conduct experiments on 3D-GCN by varying the neighbor number when constructing the receptive fields in 3D-GCN. The results are shown in Table 6.1. From this table, we see that insufficient or excessive neighbor numbers would affect the performance of 3D-GCN in describing local structural information of 3D point clouds, thus moderate neighbor number leads to better performance.

6.2 Support number S in kernels

We conduct experiments on 3D-GCN by varying the support number S of the proposed learnable kernels. The results are shown in Table 6.2. From this table, we see that kernels with more support numbers (e.g., $S = 3$ and 5) generally showed comparable classification accuracy, the performance difference was marginal. More importantly, the model complexity would grow significantly with more support numbers, leading to large memory and computation loads. As a result, $S = 1$ in our work is a reasonable choice.

Neighboring number M	5	10	25	50
ModelNet40(%)	87.7	90.4	**92.1**	91.4
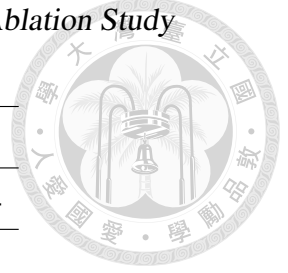
Table 6.1: **Effects on shape classification on ModelNet40 with varying neighboring number**. Note that insufficient numbers of neighbors are not expected to properly represent the receptive fields, while the performances would be less sensitive to larger numbers due to our learnable kernels.

Support number S	1	3	5
#params(M)	**0.89**	1.51	2.13
ModelNet10 (%)	93.3	**93.9**	93.6
ModelNet40 (%)	**91.7**	91.5	91.4

Table 6.2: **Performances of shape classification on ModelNet10 and ModelNet40 with varying support number S**

6.3 Learning of directional vector k_s for each kernel.

To demonstrate the power of learnable/deformable kernels in 3D-GCN, we consider three possible uses of directional vectors k_s in kernel K^S (we fix the support number S as 3). We first consider the inner product between the receptive field and the kernel as simply a correlation between the associated features, regardless of their geometry/cosine similarity. That is, (3.4) is simply replaced by $sim(\boldsymbol{p}_m, \boldsymbol{k}_s) = \left\langle \boldsymbol{f}(\boldsymbol{p}_m), \boldsymbol{w}(\boldsymbol{k}_s) \right\rangle$. The resulting accuracy is shown in the first column in Table 6.3. We next consider and assign 3 unit vectors along each axis (e.g., (1, 0, 0) along x-axis) as the 3 directional vectors k_s. Since these vectors are not learnable, and results shown in the second column of Table 6.3 was not satisfactory either. Finally, as shown in the last column of the table, we verify that the use of our learnable k_s

Directional vector	**A**	**B**	**C**
ModelNet10(%)	89.5	92.2	**93.9**
ModelNet40(%)	90.8	91.0	**91.5**

Table 6.3: **Effects on shape classification using learnable directional vector or not. A**: no directional information, **B**: assign three unit vectors (along 3 axes) as k_s, and **C**: our learnable directional vectors.

Function	ModelNet40 Acc.(%)	ShapeNetPart class mIoU(%)	S3DIS mIoU(%)
mean	89.5	80.2	58.3
max	**92.1**	**82.7**	**60.8**

Table 6.4: **Effect of the aggregation function. Max-** and **mean-** aggregation function are evaluated in the tasks of classification, part-segmentation, and scene-segmentation. We see that max-aggregation is preferable for all tasks.

would be desirable. Note that directional information is important for extracting geometric information, and learnable k_s makes kernel deformable and fitting the object of interest, which is why improved recognition performance can be achieved.

6.4 Aggregation function

In section 3.4, 3D graph convolution is formulated as equation 3.5, where the similarity values $sim(.,.)$ are aggregated by selecting **maximum** value for each kernel supports. We validated the design by experimenting **mean** value for aggregation instead, and the results are shown in Table 6.4. For classification and segmentation tasks, **max**-aggregation outperforms **mean**-aggregation. We conclude that the **max** operation is able to better preserve information in feature extracting procedure, sharing similar concept with the design of max-pooling layers in conventional 2D

Method	ModelNet40			ShapeNetPart			S3DIS		
	#params(M)	Time(s)	Acc(%)	#params(M)	Time(s)	class mIoU(%)	#params(M)	Time(s)	mIoU(%)
PointNet [1]	3.5	**0.17**	89.2	1.67	**0.17**	80.4	3.53	**0.18**	49.9
PointNet++ [3]	1.48	0.33	91.9	1.74	0.36	81.9	0.97	0.52	57.6
DGCNN [2]	1.81	0.18	**92.9**	**1.46**	0.18	82.3	0.99	**0.18**	56.0
Ours	**0.89**	**0.17**	92.1	1.64	0.21	**82.7**	**0.58**	0.23	**60.8**

Table 6.5: **Number of parameters in different models for various tasks.** The table list the parameter number in millions (**M**) and inference time for each model, while the performances are also shown.

CNN.

6.5 Visualization and Complexity analysis

In Figure 6.1, we visualize the points of an object which have large response values at each layer of our 3D-GCN. From low to high-level layers, we can see that responses were shifted from point to part levels, which confirms our ability in processing and summarizing 3D information across scales, which is equivalent to the use of 2D CNN in describing image data. On the other hand, we compare the number of parameters of recent 3D point cloud models, and list the comparison results in Table 6.5. From this table, we see that our model achieves comparable recognition performances as state-of-the-art models did, while our model required the fewest amount of parameters. It can be seen that our 3D-GCN performs favorably against recent approaches with only about half parameters required. This confirms both effectiveness and efficiency of our proposed 3D-GCN.
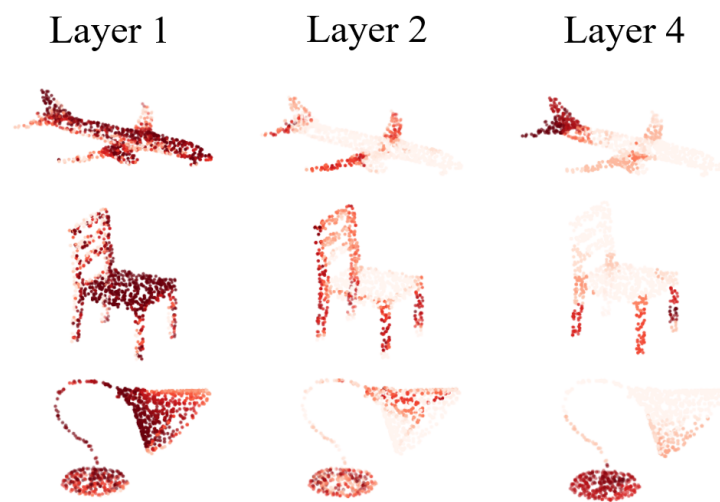
Layer 1   Layer 2   Layer 4

Figure 6.1: **Example kernel responses in different layers (segmentation on ShapeNetPart).** Note that points with larger responses are colored in darker red. As expected, the dominant responses are shifted from point (low) to part (high) levels in 3D-GCN.

Chapter 7

Conclusion

In this work, we introduced 3D-GCN which learns geometrical information of 3D point clouds across scales, and thus exhibits properties of shift and scale invariance. The technical contributions of our 3D-GCN lie in the design and learning of learnable kernels in 3D graphs, and the proposed scheme for graph max-pooling from 3D point clouds. While our model achieved comparable or improved classification and segmentation performances than recent state-of-the-art models did, we confirmed that our model is invariant to shift and scale changes and is computationally more efficient.
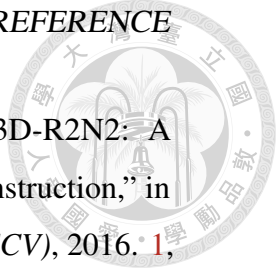
Reference

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. v, x, 1, 6, 7, 14, 15, 23, 24, 26, 27, 30, 32, 33, 34, 35, 40

[2] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, 2019. vi, ix, x, 2, 7, 15, 24, 25, 26, 27, 30, 31, 32, 33, 34, 35, 40

[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2017. ix, x, 7, 11, 14, 15, 18, 24, 26, 28, 30, 31, 32, 33, 34, 35, 40

[4] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[5] W. Luo, B. Yang, and R. Urtasun, "Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[6] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016. 1, 5

[7] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A Deep Representation for Volumetric Shapes," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1, 5, 23

[8] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst, "Geodesic convolutional Neural Networks on Riemannian Manifolds," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 1, 2

[9] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 5

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2012. 1, 5

[12] M. Simonovsky and N. Komodakis, "Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 7, 24
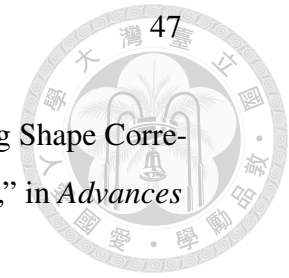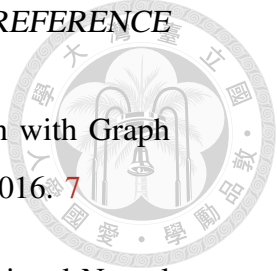
[13] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein, "Learning Shape Corre-
spondence with Anisotropic Convolutional Neural Networks," in *Advances
in Neural Information Processing Systems*, 2016. 2

[14] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining Point Cloud Local Struc-
tures by Kernel Correlation and Graph Pooling," in *Proceedings of IEEE
Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 7,
14, 24, 30

[15] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-Shape Convolutional Neural
Network for Point Cloud Analysis," in *Proceedings of IEEE Conference on
Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 7, 14

[16] M. Atzmon, H. Maron, and Y. Lipman, "Point Convolutional Neural Networks
by Extension Operators," *arXiv preprint arXiv:1803.10091*, 2018. 2, 5

[17] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J.
Guibas, "KPConv: Flexible and Deformable Convolution for Point Clouds,"
in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*,
2019. 2, 7, 11, 14, 15, 24, 26, 28, 30, 32

[18] H.-Y. Meng, L. Gao, Y. Lai, and D. Manocha, "VV-Net: Voxel VAE Net
with Group Convolutions for Point Cloud Segmentation," *arXiv preprint
arXiv:1811.04337*, 2018. 5

[19] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network
for Real-time Object Recognition," in *Proceedings of IEEE International
Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015. 5

[20] X. Roynard, J.-E. Deschaud, and F. Goulette, "Classification of Point
Cloud Scenes with Multiscale Voxel Deep Network," *arXiv preprint
arXiv:1804.03583*, 2018. 5

[21] G. Riegler, A. Osman Ulusoy, and A. Geiger, "OctNet: Learning Deep 3D Representations at High Resolutions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5

[22] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 72, 2017. 5

[23] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view Convolutional Neural Networks for 3D Shape Recognition," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015. 5

[24] A. Kanezaki, Y. Matsushita, and Y. Nishida, "RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[25] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and Multi-view CNNs for Object Classification on 3D Data," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[26] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent Convolutions for Dense Prediction in 3D," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5

[27] S. Belongie, J. Malik, and J. Puzicha, "Shape context: A new descriptor for shape matching and object recognition," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2001. 6

[28] H. Ling and D. W. Jacobs, "Shape classification using the inner-distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2007. 6
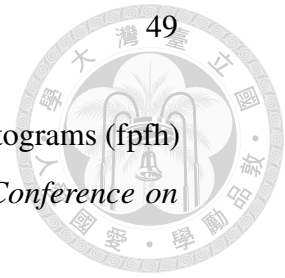
[29] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2009. 6

[30] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Aligning point cloud views using persistent feature histograms," in *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2008. 6

[31] F. Tombari, S. Salti, and L. Di Stefano, "A combined texture-shape descriptor for enhanced 3d feature matching," in *Proceedings of IEEE International Conference on Image Processing (ICIP)*. IEEE, 2011. 6

[32] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3d object recognition in cluttered scenes with local surface features: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2014. 6

[33] O. Van Kaick, H. Zhang, G. Hamarneh, and D. Cohen-Or, "A survey on shape correspondence," in *Computer Graphics Forum*. Wiley Online Library, 2011. 6

[34] Q. Huang, W. Wang, and U. Neumann, "Recurrent Slice Networks for 3D Segmentation of Point Clouds," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 6, 30

[35] R. Klokov and V. Lempitsky, "Escape From Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017. 6, 14, 24, 30

[36] M. Gadelha, R. Wang, and S. Maji, "Multiresolution Tree Networks for 3D Point Cloud Processing," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2018. 6, 24, 30

[37] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv preprint arXiv:1609.02907*, 2016. 7

[38] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2016. 7

[39] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of Neural Information Processing Systems (NIPS)*, 2017. 7

[40] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017. 7

[41] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020. 7

[42] N. Verma, E. Boyer, and J. Verbeek, "FeastNet: Feature-steered Graph Convolutions for 3D Shape Analysis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 7

[43] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or, "Meshcnn: a network with an edge," *ACM Transactions on Graphics (TOG)*, 2019. 7

[44] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise Convolutional Neural Networks," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 7

[45] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-Transformed Points," in *Advances in Neural Information Processing Systems*, 2018. 7, 14, 18

[46] H. Lei, N. Akhtar, and A. Mian, "Spherical kernel for efficient graph convolution on 3d point clouds," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 7, 8, 24

[47] Z. Zhang, B.-S. Hua, and S.-K. Yeung, "Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2019. 7, 8

[48] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 18

[49] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 19

[50] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 19

[51] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention.* Springer, 2015. 19

[52] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017. 19

[53] J. Li, B. M. Chen, and G. Hee Lee, "SO-Net: Self-Organizing Network for Point Cloud Analysis," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 24, 30
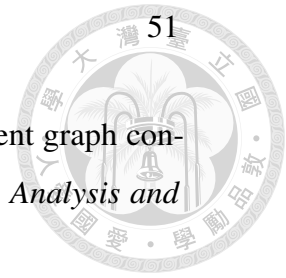
[54] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A Scalable Active Framework for Region Annotation in 3D Shape Collections," *SIGGRAPH Asia*, 2016. 27

[55] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 32