

# Adaptive Graph Convolution for Point Cloud Analysis

Haoran Zhou<sup>1</sup> Yidan Feng<sup>2</sup> Mingsheng Fang<sup>1</sup> Mingqiang Wei<sup>2\*</sup>

Jing Qin<sup>3</sup> Tong Lu<sup>1\*</sup>

<sup>1</sup>Nanjing University <sup>2</sup>Nanjing University of Aeronautics and Astronautics

<sup>3</sup>The Hong Kong Polytechnic University

## Abstract

Convolution on 3D point clouds that generalized from 2D grid-like domains is widely researched yet far from perfect. The standard convolution characterises feature correspondences indistinguishably among 3D points, presenting an intrinsic limitation of poor distinctive feature learning. In this paper, we propose Adaptive Graph Convolution (AdaptConv) which generates adaptive kernels for points according to their dynamically learned features. Compared with using a fixed/isotropic kernel, AdaptConv improves the flexibility of point cloud convolutions, effectively and precisely capturing the diverse relations between points from different semantic parts. Unlike popular attentional weight schemes, the proposed AdaptConv implements the adaptiveness inside the convolution operation instead of simply assigning different weights to the neighboring points. Extensive qualitative and quantitative evaluations show that our method outperforms state-of-the-art point cloud classification and segmentation approaches on several benchmark datasets. Our code is available at <https://github.com/hrzhou2/AdaptConv-master>.

## 1. Introduction

Point cloud is a standard output of 3D sensors, *e.g.*, LiDAR scanners and RGB-D cameras; it is considered as the simplest yet most efficient shape representation for 3D objects. A variety of applications arise with the fast advance of 3D point cloud acquisition techniques, including robotics [31], autonomous driving [20, 43] and high-level semantic analysis [35, 15]. Recent years have witnessed considerable attempts to generalize convolutional neural networks (CNNs) to point cloud data for 3D understanding. However, unlike 2D images, which are organized as regular grid-like structures, 3D points are unstructured and unordered, discretely distributed on the underlying surface of a sampled object.

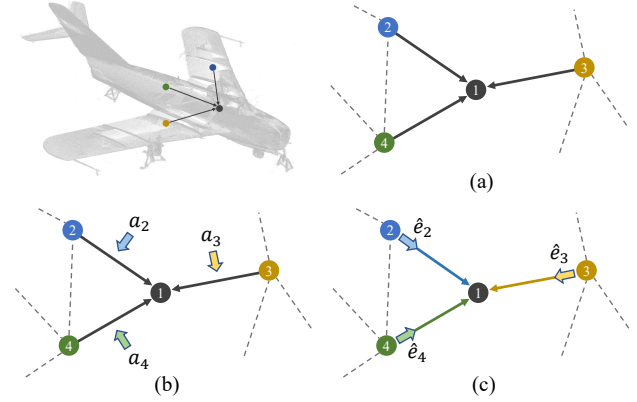


Figure 1. Illustration of adaptive kernels and fixed kernels in the convolution. (a) The standard graph convolution applies a fixed/isotropic kernel (black arrow) to compute features for each point indistinguishably. (b) Based on these features, several attentional weights  $a_i$  are assigned to determine their importance. (c) Differently, AdaptConv generates an adaptive kernel  $\hat{e}_i$  that is unique to the learned features of each point.

One common approach is to convert point clouds into regular volumetric representations and hence traditional convolution operations can be naturally applied to them [24, 30]. Such a scheme, however, often introduces excessive memory cost and is difficult to capture fine-grained geometric details. In order to handle the irregularity of point clouds without conversions, PointNet [27] applies multi-layer perceptrons (MLPs) independently on each point, which is one of the pioneering works to directly process sparse 3D points.

More recently, several researches have been proposed to utilize the graph-like structures for point cloud analysis. Graph CNNs [44, 21, 41, 4, 7] represent a point cloud as graph data according to the spatial/feature similarity between points, and generalize 2D convolutions on images to 3D data. In order to process an unordered set of points with varying neighborhood sizes, standard graph convolutions harness shared weight functions over each pair of points to extract the corresponding edge feature. This leads to a

\*Co-corresponding authors (mqwei@nuaa.edu.cn/lutong@nju.edu.cn)

fixed/isotropic convolution kernel, which is applied identically to all point pairs while neglecting their different feature correspondences. Intuitively, for points from different semantic parts of the point cloud (see the neighboring points in Fig. 1), the convolution kernel should be able to distinguish them and determine their different contributions.

To address this drawback, several approaches [41, 38] are proposed inspired by the idea of attention mechanism [3, 5]. As shown in Fig. 1 (b), proper attentional weights  $a_i$  corresponding to the neighboring points are assigned, trying to identify their different importance when performing the convolution. However, these methods are, in principle, still based on the fixed kernel convolution, as the attentional weights are just applied to the features obtained similarly (see the black arrows in Fig. 1 (b)). Considering the intrinsic isotropy of current graph convolutions, these attempts are still limited for detecting the most relevant part in the neighborhood.

Differently, we propose to adaptively establish the relationship between a pair of points according to their learned features. This adaptiveness represents the diversity of kernels unique to each pair of points rather than relying on the predefined weights. To achieve this, in this paper, we propose a new graph convolution operator, named *adaptive graph convolution* (AdaptConv). AdaptConv generates adaptive kernels  $\hat{e}_i$  for points in the convolution which replace the aforementioned isotropic kernels (see Fig. 1 (c)). The key contribution of our work is that the proposed AdaptConv is employed inside the graph convolution rather than a weight function that is based on the resulting feature. Furthermore, we explore several choices for the feature convolving design, offering more flexibility to the implementation of the adaptive convolution. Extensive experiments demonstrate the effectiveness of the proposed AdaptConv, achieving state-of-the-art performances in both classification and segmentation tasks on several benchmark datasets.

## 2. Related Work

**Voxelization-based and multi-view methods.** The voxelization/projection strategy has been explored as a simple way in point cloud analysis to build proper representations for adapting the powerful CNNs in 2D vision. A number of works [24, 47, 16, 42] project point clouds onto regular grids, but inevitably suffer from information loss and enormous computational cost. To alleviate these problems, OctNet [30] and Kd-Net [14] attempt to use more efficient data structures and skip the computations on empty voxels. Alternatively, the multi-view-based methods [12, 34] treat a point cloud as a set of 2D images projected from multiple views, so as to directly leverage 2D CNNs for subsequent processing. However, it is fundamentally difficult to apply these methods to large-scale scanned data, considering the struggle of covering the entire scene from single-point per-

spectives.

**Point-based methods.** In order to handle the irregularity of point clouds, state-of-the-art deep networks are designed to directly manipulate raw point cloud data, instead of introducing an intermediate representation. In this way, PointNet [27] first proposes to use MLPs independently on each point and subsequently aggregate global features through a symmetric function. Thanks to this design, PointNet is invariant to input point orders, but fails to encode local geometric information, which is important for semantic segmentation tasks. To solve this issue, PointNet++ [29] proposes to apply PointNet layers locally in a hierarchical architecture to capture regional information. Alternatively, Huang et al. [9] sorts unordered 3D points into an ordered list and employs Recurrent Neural Networks (RNN) to extract features according to different dimensions.

More recently, various approaches have been proposed for effective local feature learning. PointCNN [19] aligns points in a certain order by predicting a transformation matrix for local point set, which inevitably leads to sensitivity in point order since the operation is not permutation-invariant. SpiderCNN [48] defines its convolution kernel as a family of polynomial functions, relying on the neighbors' order. PCNN [2] designs point kernels based on the spatial coordinates and further KPConv [36] presents a scalable convolution using explicit kernel points. RS-CNN [22] assigns channel-wise weights to neighboring point features according to the geometric relations learned from 10-D vectors. ShellNet [51] splits local point set into several shell areas, from which features are extracted and aggregated. Recently, [53, 6] utilize the successful transformer structures in natural language processing [37, 45] to build dense self-attention between local and global features.

The graph-based methods treat points as nodes of a graph, and establish edges according to their spatial/feature relationships. Graph is a natural representation for a point cloud to model local geometric structures but is challenging for processing due to its irregularity. The notion of Graph Convolutional Network is proposed by [13], which generalizes convolution operations over graphs by averaging features of adjacent nodes. Similar ideas [32, 44, 8, 19, 17] have been explored to extract local geometric features from local points. Shen et al. [32] define kernels according to euclidean distances and geometric affinities in the neighboring points. DGCNN [44] gathers nearest neighboring points in the feature space, followed by the EdgeConv operators for feature extraction, in order to identify semantic cues dynamically. MoNet [25] defines the convolution as Gaussian mixture models in a local pseudo-coordinate system. Inspired by the idea of attention mechanism, several works [38, 41, 39] propose to assign proper attentional weights to different points/filters. 3D-GCN [21] develops deformable kernels, focusing on shift and scale-invariant properties in

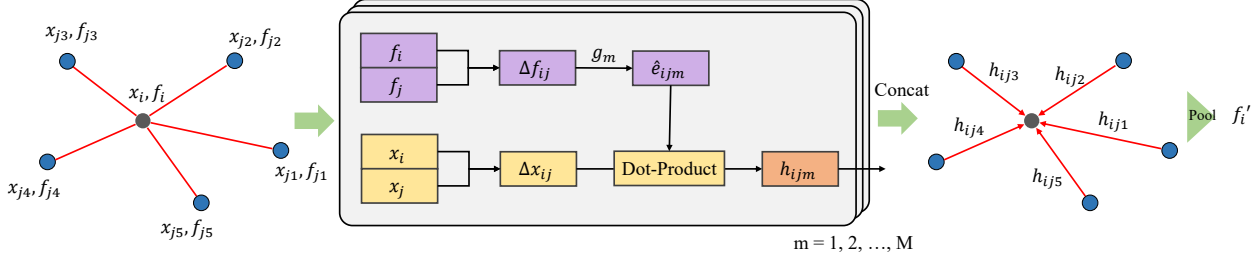


Figure 2. The illustration of AdaptConv processed in the neighborhood of a target point. An adaptive kernel  $\hat{e}_{ijm}$  is generated from the feature input  $\Delta f_{ij}$  of a pair of points on the edge, which is then convolved with the corresponding spatial input  $\Delta x_{ij}$ . Concatenating  $h_{ijm}$  of all dimensions yields the edge feature  $h_{ij}$ . Finally, the output feature  $f'_i$  of the central point is obtained through a pooling function. AdaptConv differs from other graph convolutions in that the convolution kernel is unique for each pair of points.

point cloud analysis.

**Convolution on point clouds.** State-of-the-art researches have proposed many methods to define a proper convolution on point clouds. To improve the basic designs using fixed MLPs in PointNet/PointNet++, a variety of works [38, 41, 39, 36, 22] try to introduce weights based on the learned features, with more variants of convolution inputs [44, 25, 48]. Other methods [33, 46, 10] try to learn a dynamic weight for the convolution. However, their idea is to approximate weight functions from the direct 3D coordinates while AdaptConv uses features to learn the kernels, which represents more adaptiveness. In addition, their implementation is heavily memory consuming when convolving with high-dimensional features. Thus, the main focus of this paper is to handle the isotropy of point cloud convolutions, by developing an adaptive kernel that is unique to each point in the convolution.

### 3. Method

We exploit local geometric characteristics in point cloud analysis by proposing a novel adaptive graph convolution (AdaptConv) in the spirit of graph neural networks (Sec. 3.1). Afterwards, we discuss several choices for the feature decisions in the adaptive convolution (Sec. 3.2). The details of the constructed networks are shown in Sec. 3.3.

#### 3.1. Adaptive graph convolution

We denote the input point cloud as  $\mathcal{X} = \{x_i | i = 1, 2, \dots, N\} \in \mathbb{R}^{N \times 3}$  with the corresponding features defined as  $\mathcal{F} = \{f_i | i = 1, 2, \dots, N\} \in \mathbb{R}^{N \times D}$ . Here,  $x_i$  processes the  $(x, y, z)$  coordinates of the  $i$ -th point, and, in other cases, can be potentially combined with a vector of additional attributes, such as normal and color. We then compute a directed graph  $\mathcal{G}(\mathcal{V}, \mathcal{E})$  from the given point cloud where  $\mathcal{V} = \{1, \dots, N\}$  and  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  represents the set of edges. We construct the graph by employing the  $k$ -nearest neighbors (KNN) of each point including self-loop. Given the input  $D$ -dimensional features, our AdaptConv layer is designed to produce a new set of  $M$ -dimensional features

with the same number of points while attempting to more accurately reflect local geometric characteristics than previous graph convolutions.

Denote that  $x_i$  is the central point in the graph convolution, and  $\mathcal{N}(i) = \{j : (i, j) \in \mathcal{E}\}$  is a set of point indices in its neighborhood. Due to the irregularity of point clouds, previous methods usually apply a fixed kernel function on all neighbors of  $x_i$  to capture the geometric information of the patch. However, different neighbors may reflect different feature correspondences with  $x_i$ , particularly when  $x_i$  is located at salient regions, such as corners or edges. In this regard, the fixed kernel may incapacitate the geometric representations generated from the graph convolution for classification and, particularly, segmentation.

In contrast, we endeavor to design an adaptive kernel to capture the distinctive relationships between each pair of points. To achieve this, for each channel in the output  $M$ -dimensional feature, our AdaptConv dynamically generates a kernel using a function over the point features  $(f_i, f_j)$ :

$$\hat{e}_{ijm} = g_m(\Delta f_{ij}), j \in \mathcal{N}(i). \quad (1)$$

Here,  $m = 1, 2, \dots, M$  indicates one of the  $M$  output dimensions corresponding to a single filter defined in our AdaptConv. In order to combine the global shape structure and feature differences captured in a local neighborhood [44], we define  $\Delta f_{ij} = [f_i, f_j - f_i]$  as the input feature for the adaptive kernel, where  $[\cdot, \cdot]$  is the concatenation operation. The  $g(\cdot)$  is a feature mapping function, and here we use a multilayer perceptron.

Like the computations in 2D convolutions, which obtain one of the  $M$  output dimensions by convolving the  $D$  input channels with the corresponding filter weights, our adaptive kernel is convolved with the corresponding points  $(x_i, x_j)$ :

$$h_{ijm} = \sigma \langle \hat{e}_{ijm}, \Delta x_{ij} \rangle, \quad (2)$$

where  $\Delta x_{ij}$  is defined as  $[x_i, x_j - x_i]$  similarly,  $\langle \cdot, \cdot \rangle$  represents the inner product of two vectors outputting  $h_{ijm} \in \mathbb{R}$  and  $\sigma$  is a nonlinear activation function. As shown in Fig. 2 (middle part), the  $m$ -th adaptive kernel  $\hat{e}_{ijm}$  is combined

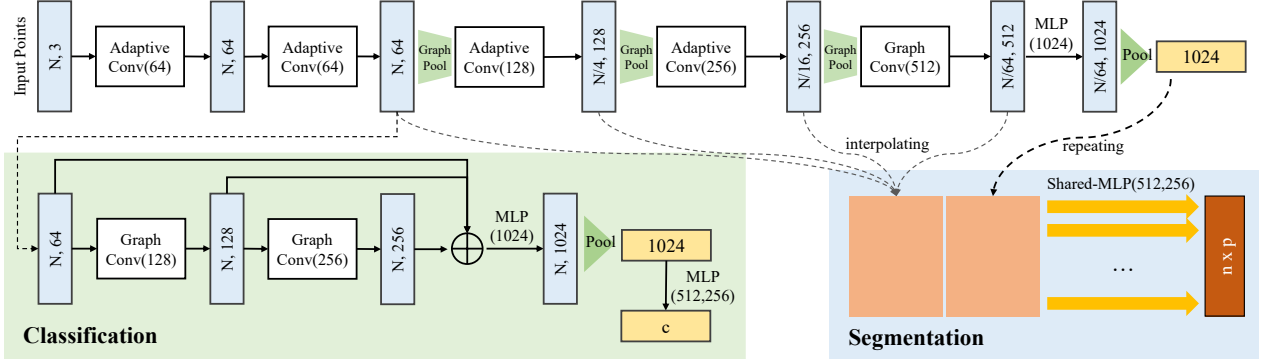


Figure 3. AdaptConv network architecture for classification and segmentation tasks. The GraphConv layer denotes our standard convolution without an adaptive kernel. The segmentation model uses pooling and interpolating to build a hierarchical graph structure, while the classification model applies a dynamic structure [44].

with the spatial relations  $\Delta x_{ij}$  of the corresponding point  $x_j \in \mathbb{R}^3$ , which means the size of the kernel should be matched in the dot product, *i.e.*, the aforementioned feature mapping is  $g_m : \mathbb{R}^{2D} \rightarrow \mathbb{R}^6$ . In this way, the spatial positions in the input space can be efficiently incorporated into each layer, combined with the feature correspondences extracted dynamically from our kernel. Stacking the  $h_{ijm}$  of each channel yields the edge feature  $h_{ij} = [h_{iji}, h_{ij2}, \dots, h_{ijM}] \in \mathbb{R}^M$  between the connected points  $(x_i, x_j)$ . Finally, we define the output feature of the central point  $x_i$  by applying an aggregating function over all the edge features in the neighborhood (see Fig. 2 (right part)):

$$f'_i = \max_{j \in \mathcal{N}(i)} h_{ij}, \quad (3)$$

where max is a channel-wise max-pooling function. Overall, the convolution weights of AdaptConv are defined as  $\Theta = (g_1, g_2, \dots, g_M)$ .

### 3.2. Feature decisions

In our method, AdaptConv generates an adaptive kernel for each pair of points according to their individual features  $(f_i, f_j)$ . Then, the kernel  $\hat{e}_{ijm}$  is applied to the point pair of  $(x_i, x_j)$  in order to describe their spatial relations in the input space. The feature decision of  $\Delta x_{ij}$  in the convolution of Eq. 2 is an important design. In other cases, the inputs can be  $x_i \in \mathbb{R}^E$  including additional dimensions representing other valuable point attributes, such as point normals and colors. By modifying the adaptive kernel to  $g_m : \mathbb{R}^{2D} \rightarrow \mathbb{R}^{2E}$ , our AdaptConv can also capture the relationships between feature dimensions and spatial coordinates which are from different domains. Note that, this is another option in our AdaptConv design, and we use the spatial positions as input  $x_i$  by default in the convolution in our experiments.

As an optional choice, we replace the  $\Delta x_{ij}$  with  $\Delta f_{ij}$  in Eq. 2 with a modified dimension of  $\hat{e}_{ijm}$ . Therefore, the

adaptive kernel of a pair of points is designed to establish the relations of their current features  $(f_i, f_j)$  in each layer. This is a more direct solution, similar to other convolution operators, that produces a new set of learned features from features in the preceding layer of the network. However, we recommend xyz rather than feature in that: (i) the point feature  $f_j$  has been already included in the adaptive kernel and convolving again with  $f_j$  leads to redundancy of feature information; (ii) it is easier to learn spatial relations through MLPs, instead of detecting feature correspondences in a high-dimensional space (*e.g.* 64, 128 dimensional features); (iii) the last reason is the memory cost and more specifically the large computational graph in the training stage which cannot be avoided. We evaluate all these choices in Sec. 4.4.

### 3.3. Network architecture

We design two network architectures for point cloud classification and segmentation tasks using the proposed AdaptConv layer. The network architectures are shown in Fig. 3. In our experiments, the AdaptConv kernel function is implemented as a two-layer MLP with residual connections to extract important geometric information. More details are available in the supplemental material. The standard graph convolution layer with a fixed kernel uses the same feature input  $\Delta f_{ij}$  as in the adaptive kernels.

**Graph pooling.** For segmentation tasks, we reduce the number of points progressively in order to build the network in a hierarchical architecture. The point cloud is sub-sampled using furthest point sampling algorithm [27], and is applied by a pooling layer to output aggregated features on the coarsened graph. In each graph pooling layer, a new graph is constructed corresponding to the sampled points. The feature pooled at each point in the subcloud can be simply obtained by a max-pooling function within its neighborhood. Alternatively, we can use a AdaptConv layer to aggregate this pooled features. To predict point-wise labels for segmentation purpose, we need to interpolate deeper fea-



Method	Input	#points	mAcc(%)	OA(%)
3DShapeNets [47]	voxel	-	77.3	84.7
VoxNet [24]	voxel	-	83.0	85.9
Subvolume [28]	voxel	-	86.0	89.2
PointNet [27]	xyz	1k	86.0	89.2
PointNet++ [29]	xyz, normal	5k	-	91.9
Kd-Net [14]	xyz	1k	-	90.6
SpecGCN [40]	xyz	1k	-	92.1
SpiderCNN [48]	xyz, normal	5k	-	92.4
PointCNN [19]	xyz	1k	88.1	92.2
SO-Net [18]	xyz, normal	5k	-	<b>93.4</b>
DGCNN [44]	xyz	1k	90.2	92.9
KPCConv [36]	xyz	6.8k	-	92.9
3D-GCN [21]	xyz	1k	-	92.1
PointASNL [49]	xyz, normal	1k	-	93.2
Ours	xyz	1k	<b>90.7</b>	<b>93.4</b>

Table 1. Classification results on ModelNet40 dataset. Our network achieves the best results according to the mean class accuracy (mAcc) and overall accuracy (OA).

tures from subsampled cloud to the original points. Here, we use the nearest upsampling to get the features for each layer. The features are concatenated for the final point-wise features.

**Segmentation network.** Our segmentation network architecture is illustrated in Fig. 3. The AdaptConv encoder includes 5 layers of convolutions in which the last one is a standard graph convolution layer, as well as several graph pooling layers. The subsampled features are interpolated and concatenated for the final point features which are fed to the decoder part.

**Classification network.** The classification network uses a similar encoder part as in the segmentation model (see Fig. 3). For sparser point clouds used in the ModelNet40 classification dataset, we simply apply dynamic graph structures [44] without pooling and interpolation. Specifically, the graph structure is updated in each layer according to the feature similarity among points, rather than fixed using spatial positions. That is, in each layer, the edge set  $\mathcal{E}_l$  is recomputed where the neighborhood of point  $x_i$  is  $\mathcal{N}(i) = \{j_{i_1}, j_{i_2}, \dots, j_{i_k}\}$  such that the corresponding features  $f_{j_{i_1}}, f_{j_{i_2}}, \dots, f_{j_{i_k}}$  are closest to  $f_i$ . This encourages the network to organize the graph semantically and expands the receptive field of local neighborhood by grouping together similar points in the feature space.

## 4. Evaluation

In this section, we evaluate our models using AdaptConv for point cloud classification, part segmentation and indoor segmentation tasks. Detailed network architectures and comparisons are provided.

### 4.1. Classification

**Data.** We evaluate our model on ModelNet40 [47] dataset for point cloud classification. This dataset contains 12,311 meshed CAD models from 40 categories, where 9,843 models are used for training and 2,468 models for testing. We follow the experimental setting of [27]. 1024 points are uniformly sampled for each object and we only use the  $(x, y, z)$  coordinates of the sampled points as input. The data augmentation procedure includes shifting, scaling and perturbing of the points.

**Network configuration.** Following [44], we recompute the graph based on the feature similarity in each layer. The number  $k$  of neighborhood size is set to 20 for all layers. Afterwards, we collect features from all layers and apply one shared fully-connected layer (1024). Then, a max pooling function is applied to get the global feature for the point cloud, after which fully-connected layers (512, 256) are used for the classification output. In the last two fully-connected layers, we use dropout with a keep probability of 0.5. All layers are with LeakyReLU and batch normalization. We use SGD optimizer with momentum set to 0.9. The initial learning rate is 0.1 and is dropped until 0.001 using cosine annealing [23]. The batch size is set to 32 for all training models. We use PyTorch [26] implementation and train the network on a RTX 2080 Ti GPU. The hyperparameters are chosen in a similar way for other tasks.

**Results.** We show the results for classification in Tab. 1. The evaluation metrics on this dataset are the mean class accuracy (mAcc) and the overall accuracy (OA). Our model achieves the best result on this dataset. For a clear comparison, we show the input data types and the numbers of points corresponding to each method. Our AdaptConv only considers the point coordinates as input with a relatively small size of 1k points, which already outperforms other methods with larger inputs.

### 4.2. Part segmentation

**Data.** We further test our model for part segmentation task on ShapeNet part dataset [50]. This dataset contains 16,881 shapes from 16 categories, totally annotated with 50 parts. Each point cloud is labelled with 2-6 parts. We follow the experimental setting of [29] and use their provided data for benchmarking purpose. The dataset contains 16,880 shapes with 14,006 for training and 2,874 for testing. The input attributes include the point normals apart from the 3D coordinates.

**Network configuration.** Following [27], we include a one-hot vector representing category types for each point. It is stacked with the point-wise features to compute the segmentation results. Other training parameters are set the same as in our classification task. Note that, we use spatial positions (without normals) as  $\Delta x_{ij}$  as discussed in Sec. 3.2. Other choices will be evaluated later in Sec. 4.4.

Method	mcIoU	mIoU	air plane	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor bike	mug	pistol	rocket	skate board	table
Kd-Net [14]	77.4	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	87.4	86.7	78.1	51.8	69.9	80.3
PointNet [27]	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
PointNet++ [29]	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
SO-Net [18]	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
DGCNN [44]	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
PointCNN [19]	-	86.1	84.1	86.4	86.0	80.8	90.6	79.7	92.3	88.4	85.3	96.1	77.2	95.3	84.2	64.2	80.0	83.0
PointASNL [49]	-	86.1	84.1	84.7	87.9	79.7	92.2	73.7	91.0	87.2	84.2	95.8	74.4	95.2	81.0	63.0	76.3	83.2
3D-GCN [21]	82.1	85.1	83.1	84.0	86.6	77.5	90.3	74.1	90.9	86.4	83.8	95.6	66.8	94.8	81.3	59.6	75.7	82.8
KPConv [36]	<b>85.1</b>	<b>86.4</b>	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	83.4	<b>86.4</b>	84.8	81.2	85.7	79.7	91.2	80.9	91.9	88.6	84.8	96.2	70.7	94.9	82.3	61.0	75.9	84.2

Table 2. Part segmentation results on ShapeNet dataset evaluated as the mean class IoU (mcIoU) and mean instance IoU (mIoU).

Ablations	mcIoU(%)	mIoU(%)
GraphConv	81.9	85.5
Attention Point	78.0	83.3
Attention Channel	77.9	83.0
Feature	82.2	85.9
Normal	83.2	86.2
Initial attributes	83.2	86.1
Ours	<b>83.4</b>	<b>86.4</b>

Table 3. Ablation studies on ShapeNet dataset for part segmentation.

**Results.** We report the mean class IoU (mcIoU) and mean instance IoU (mIoU) in Tab. 2. Following the evaluation scheme of [27], the IoU of a shape is computed by averaging the IoU of each part. The mean IoU (mIoU) is computed by averaging the IoUs of all testing instances. The class IoU (mcIoU) is the mean IoU over all shape categories. We also show the class-wise segmentation results. Our model achieves the best performance compared with the state-of-the-arts.

### 4.3. Indoor scene segmentation

**Data.** Our third experiment shows the semantic segmentation performance of our architecture on the S3DIS dataset [1]. This dataset contains 3D RGB point clouds from six indoor areas of three different buildings, covering a total of 271 rooms. Each point is annotated with one semantic label from 13 categories. For a common evaluation protocol [35, 27, 15], we choose Area 5 as the test set which is not in the same building as other areas.

**Real scene segmentation.** The large-scale indoor datasets reveal more challenges, covering larger scenes in a real-world environment with a lot more noise and outliers. Thus, we follow the experimental settings of KPConv [36], and train the network using randomly sampled clouds in spheres. The subclouds contain more points with varying sizes, and are stacked into batches for training. In the test stage, spheres are uniformly picked in the scenes, and

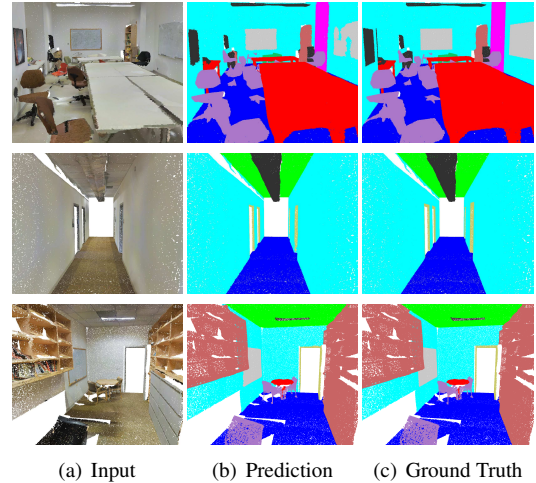


Figure 4. Visualization of semantic segmentation results on the S3DIS dataset. We show the input point cloud, and labelled points mapped to RGB colors.

we ensure each point is tested several times using a voting scheme. The input point attributes include the RGB colors and the original heights.

**Results.** We report the mean classwise intersection over union (mIoU), mean classwise accuracy (mAcc) and overall accuracy (OA) in Tab. 4. The IoU of each class is also provided. The proposed AdaptConv outperforms the state-of-the-arts in most of the categories, which further demonstrates the effectiveness of adaptive convolutions over fixed kernels. The qualitative results are visualized in Fig. 4 where we show rooms from different areas of the building. Our method can correctly detect less obvious edges of, e.g., pictures and boards on the wall.

### 4.4. Ablation studies

In this section, we explain some of the architecture choices used in our network, and demonstrate the effectiveness of AdaptConv compared to several ablation networks.

**Adaptive convolution vs Attention.** In order to tackle the isotropy of standard graph convolution based on a fixed

Method	OA	mAcc	mIoU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [27]	–	49.0	41.1	88.8	97.3	69.8	0.1	3.9	46.3	10.8	59.0	52.6	5.9	40.3	26.4	33.2
SegCloud [35]	–	57.4	48.9	90.1	96.1	69.9	0.0	18.4	38.4	23.1	70.4	75.9	40.9	58.4	13.0	41.6
PointCNN [19]	85.9	63.9	57.3	92.3	98.2	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
PCCN [43]	–	67.0	58.3	92.3	96.2	75.9	0.3	6.0	69.5	63.5	66.9	65.6	47.3	68.9	59.1	46.2
PointWeb [52]	87.0	66.6	60.3	92.0	98.5	79.4	0.0	21.1	59.7	34.8	76.3	88.3	46.9	69.3	64.9	52.5
HPEIN [11]	87.2	68.3	61.9	91.5	98.2	81.4	0.0	23.3	65.3	40.0	75.5	87.7	58.5	67.8	65.6	49.4
GAC [41]	87.7	–	62.8	92.2	98.2	81.9	0.0	20.3	59.0	40.8	78.5	85.8	61.7	70.7	74.6	52.8
KPConv [36]	–	72.8	67.1	92.8	97.3	82.4	0.0	23.9	58.0	69.0	81.5	91.0	75.4	75.3	66.7	58.9
PointASNL [49]	87.7	68.5	62.6	94.3	98.4	79.1	0.0	26.7	55.2	66.2	83.3	86.8	47.6	68.3	56.4	52.1
Ours	<b>90.0</b>	<b>73.2</b>	<b>67.9</b>	93.9	98.4	82.2	0.0	23.9	59.1	71.3	91.5	81.2	75.5	74.9	72.1	58.6

Table 4. Semantic segmentation results on S3DIS dataset evaluated on Area 5.

kernel, several prior works [38, 41, 39] utilize the attention mechanism. In this section, we compare our model with these graph attention networks (Attention). Since they may not focus on tasks of point cloud analysis, we design several ablations using attentional graph convolution layers in our framework to demonstrate the effectiveness of our adaptive convolution.

Specifically, [38, 39] assign attentional weights to different neighboring points and [41] further designs a channel-wise attention. We denote these two ablations as Point and Channel in Tab. 3 respectively. We only replace the AdaptConv layers in our network and the feature inputs  $\Delta f_{ij}$  are the same as our model. Other network architectures are identical. Besides, we also show the result by using standard graph convolutions (GraphConv), which can be seen as a similar version of DGCNN [44]. From the comparison, our method achieves better results than other graph convolutions.

**Feature decisions.** In AdaptConv, the adaptive kernel is generated from the feature input  $\Delta f_{ij}$ , and subsequently convolved with the corresponding spatial positions  $\Delta x_{ij}$ . We have discussed several other choices of  $\Delta x_{ij}$  in Eq. 2 in Sec. 3.2. Note that, in our experiments,  $\Delta x_{ij}$  corresponds to the (x, y, z) spatial coordinates of points. Several ablation networks are designed:

- Feature - In Eq. 2, we convolve the adaptive kernel  $\hat{e}_{ijm}$  with their current point features. That is,  $\Delta x_{ij}$  is replaced with  $\Delta f_{ij}$  and the kernel function is  $g_m : \mathbb{R}^{2D} \rightarrow \mathbb{R}^{2D}$ . This makes the kernel learn to adapt to the features from previous layer and extracts the feature relations.
- Initial attributes - The point normals ( $n_x, n_y, n_z$ ) are included in the part segmentation task on ShapeNet, leading to a 6-dimensional initial feature attributes for each point. Thus, we design three ablations where we use only spatial inputs (Ours), only normal inputs (Normal) and both of them (Initial attributes). The kernel function is modified correspondingly.

The resulting IoU accuracies are shown in Tab. 3. Since the spatial coordinates are the basic attributes in 3D points, it is recommended to use them in the convolution with adap-

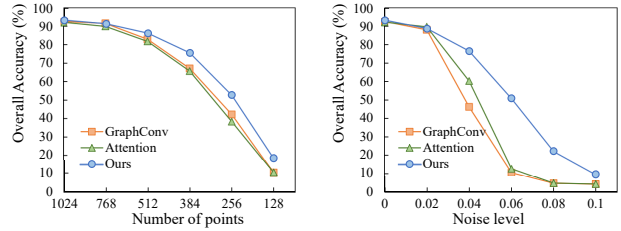


Figure 5. Robustness test on ModelNet40 for classification. GraphConv indicates the standard graph convolution network. Attention indicates the ablation where we replace the AdaptConv layers with graph attention layers (point-wise). From the comparison, we can see that our model is more robust to point density and noise perturbation.

Number $k$	mAcc(%)	OA(%)
5	89.4	92.8
10	90.7	93.2
20	90.7	93.4
40	90.4	93.0

Table 5. Results of our classification network with different numbers  $k$  of nearest neighbors.

tive kernels. Although achieving a promising result, the computational cost for the Feature ablation is extremely high since the network expands heavily when it is convolved with a high-dimensional feature.

#### 4.5. Robustness test

We further evaluate the robustness of our model to point cloud density and noise perturbation on ModelNet40 [47]. We compare our AdaptConv with several other graph convolutions as discussed in Sec. 4.4. All the networks are trained with 1k points and neighborhood size is set to  $k = 20$ . In order to test the influence of point cloud density, a series of numbers of points are randomly dropped out during testing. For noise test, we introduce additional Gaussian noise with standard deviations according to the point cloud radius. From Fig. 5, we can see that our method is robust to missing data and noise, thanks to the adaptive kernel in

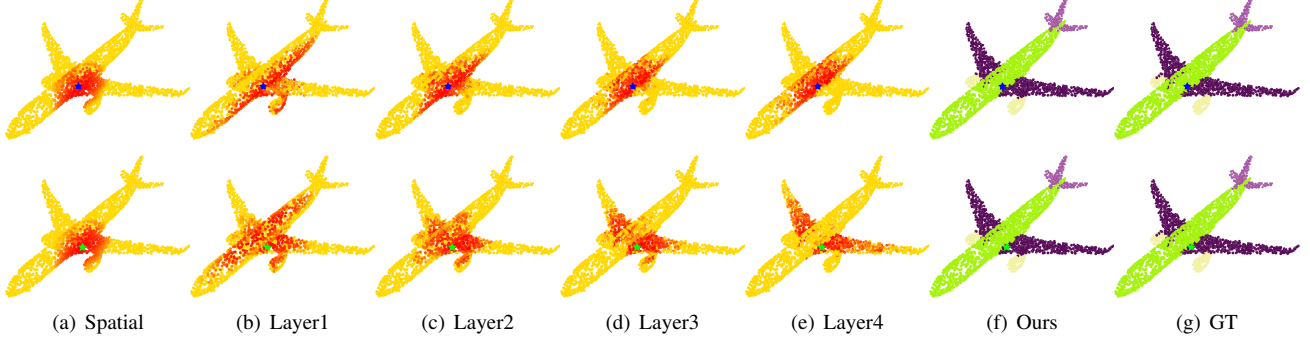


Figure 6. Visualize the Euclidean distances between two target points (blue and green stars) and other points in the feature space (red: near, yellow: far).

Method	#parameters	OA(%)
PointNet [27]	3.5M	89.2
PointNet++ [29]	1.48M	91.9
DGCNN [44]	1.81M	92.9
KPConv [36]	14.3M	92.9
Ours	1.85M	<b>93.4</b>

Table 6. The number of parameters and overall accuracy of different models.

which the structural connections can be extracted dynamically in a sparser area.

Also, we experiment the influence of different numbers  $k$  of the nearest neighboring points in Tab. 5. We choose several typical sizes for testing. Reducing the number of neighboring points leads to less computational cost while the performance will degenerate due to the limitation of receptive field. Our network still achieves a promising result when  $k$  is reduced to 5. On the other hand, with certain point density, a larger  $k$  doesn’t improve the performance since the local information dilutes within a larger neighborhood.

#### 4.6. Efficiency

To compare the complexity of our model with previous state-of-the-arts, we show the parameter numbers and the corresponding results of networks in Tab. 6. These models are based on ModelNet40 for classification task. From the table, we see that our model achieves the best performance of 93.4% overall accuracy and the model size is relatively small. Compared with DGCNN [44] which can be seen as a standard graph convolution version in our ablation studies, the proposed adaptive kernel performs better while being efficient.

### 5. Visualization and learned features

To achieve a deeper understanding of AdaptConv, we explore the feature relations in several intermediate layers of

the network to see how AdaptConv can distinguish points with similar spatial inputs. In this experiment, we train our model on ShapeNetPart dataset for segmentation. In Fig. 6, two target points (blue and green stars in 1-st and 2-nd rows respectively) are selected which belong to different parts of the object. We then compute the euclidean distances to other points in the feature space, and visualize them by coloring the points with similar learned features in red. We can see that, while being spatially close, our network can capture their different geometric characteristics and segment them properly. Also, from the 2-nd row of Fig. 6, points belonging to the same semantic part (the wings) share similar features while they may not be spatially close. This shows that our model can extract valuable information in a non-local manner.

### 6. Conclusion

In this paper, we propose a novel adaptive graph convolution (AdaptConv) for 3D point cloud. The main contribution of our method lies in the designed adaptive kernel in 3D graphs, which is dynamically generated according to the point features. Instead of using a fixed convolution kernel that captures the feature correspondences indistinguishably between points, our AdaptConv can produce learned features that are more effective to shape geometric details. We have applied AdaptConv to train end-to-end deep networks for several point cloud analysis tasks, outperforming the state-of-the-arts on several public datasets. Further, AdaptConv can be easily integrated into existing graph CNNs to improve their performance by simply replacing the existing kernels with the adaptive kernels.

**Acknowledgements.** This work was supported by the National Natural Science Foundation of China (No. 62032011, No.62172218, No. 61672273) and the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. PolyU 152035/17E and 15205919).



## References

- [1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016. 6
- [2] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018. 2
- [3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2
- [4] Kent Fujiwara and Taiichi Hashimoto. Neural implicit embedding for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11734–11743, 2020. 1
- [5] Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*, 2016. 2
- [6] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *arXiv preprint arXiv:2012.09688*, 2020. 2
- [7] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017. 1
- [8] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Point-wise convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–993, 2018. 2
- [9] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2635, 2018. 2
- [10] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29:667–675, 2016. 3
- [11] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10433–10441, 2019. 7
- [12] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3779–3788, 2017. 2
- [13] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2
- [14] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 2, 5, 6
- [15] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. 1, 6
- [16] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018. 2
- [17] Huan Lei, Naveed Akhtar, and Ajmal Mian. Spherical kernel for efficient graph convolution on 3d point clouds. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 2
- [18] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018. 5, 6
- [19] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on  $\chi$ -transformed points. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 828–838, 2018. 2, 5, 6, 7
- [20] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision*, pages 641–656, 2018. 1
- [21] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1800–1809, 2020. 1, 2, 5, 6
- [22] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019. 2, 3
- [23] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 5
- [24] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 1, 2, 5
- [25] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017. 2, 3
- [26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 5
- [27] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 77–85, 2017. 1, 2, 4, 5, 6, 7, 8
- [28] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and

- multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 5
- [29] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 2, 5, 6, 8
- [30] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3577–3586, 2017. 1, 2
- [31] 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. 1
- [32] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4548–4557, 2018. 2
- [33] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017. 3
- [34] 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. 2
- [35] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*, pages 537–547. IEEE, 2017. 1, 6, 7
- [36] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019. 2, 3, 5, 6, 7, 8
- [37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 2
- [38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. 2, 3, 7
- [39] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2598–2606, 2018. 2, 3, 7
- [40] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–66, 2018. 5
- [41] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10296–10305, 2019. 1, 2, 3, 7
- [42] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017. 2
- [43] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018. 1, 7
- [44] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 1, 2, 3, 4, 5, 6, 7, 8
- [45] Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019. 2
- [46] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 3
- [47] 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. 2, 5, 7
- [48] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018. 2, 3, 5
- [49] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5589–5598, 2020. 5, 6, 7
- [50] Li Yi, Vladimir G Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016. 5
- [51] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1607–1616, 2019. 2
- [52] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019. 7
- [53] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. *arXiv preprint arXiv:2012.09164*, 2020. 2