AGConv: Adaptive Graph Convolution on 3D Point Clouds

Mingqiang Wei, Zeyong Wei, Haoran Zhou, Fei Hu, Huajian Si, Zhilei Chen, Zhe Zhu, Jingbo Qiu, Xuefeng Yan, Yanwen Guo, Jun Wang, and Jing Qin

Abstract—Convolution on 3D point clouds is widely researched yet far from perfect in geometric deep learning. The traditional wisdom of convolution characterises feature correspondences indistinguishably among 3D points, arising an intrinsic limitation of poor distinctive feature learning. In this paper, we propose Adaptive Graph Convolution (AGConv) for wide applications of point cloud analysis. AGConv generates adaptive kernels for points according to their dynamically learned features. Compared with the solution of using fixed/isotropic kernels, AGConv improves the flexibility of point cloud convolutions, effectively and precisely capturing the diverse relations between points from different semantic parts. Unlike the popular attentional weight schemes, AGConv implements the adaptiveness inside the convolution operation instead of simply assigning different weights to the neighboring points. Extensive evaluations clearly show that our method outperforms state-of-the-arts of point cloud classification and segmentation on various benchmark datasets. Meanwhile, AGConv can flexibly serve more point cloud analysis approaches to boost their performance. To validate its flexibility and effectiveness, we explore AGConv-based paradigms of completion, denoising, upsampling, registration and circle extraction, which are comparable or even superior to their competitors. Our code is available at *https://github.com/hrzhou2/AdaptConv-master*.

Index Terms—Adaptive graph convolution, Point cloud analysis, Geometric deep learning

1 Introduction

Point clouds are a standard output of 3D sensors, e.g., LiDAR scanners and RGB-D cameras [1]. As the commonly-used shape representation, point clouds preserve the original geometric information in 3D space with a very simple yet flexible data structure [2], [3], [4]. A variety of applications arise with the fast advance of point cloud acquisition techniques, including robotics, autonomous driving and high-level semantic analysis. Recent years have witnessed considerable attempts to generalize convolutional neural networks (CNNs) to point clouds for 3D analysis and generation [5], [6]. 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.

The common ways of learning on point clouds are to convert them into regular 2D grids, 3D voxels or to develop hand-crafted feature descriptors, on which traditional 2D/3D CNNs can be naturally applied [7], [8], [9], [10]. Such solutions, however, often introduce excessive memory cost and are difficult to capture fine-grained geometric details. To handle the irregularity of point clouds without conversions, PointNet [11] applies multi-layer perceptrons (MLPs) independently on each point, which is the pioneering work to directly process sparse 3D points.

More recently, the very promising graph-like structures are explored for point cloud analysis. Graph CNNs (GCNs) [12], [13], [14], [15] describe a point cloud as graph data according to the spatial/feature similarity between points, and generalize 2D convolutions on images to 3D data. GCN-based methods have shown a great ability to understand the contextual features and achieved much higher processing accuracy (e.g., point cloud segmentation) than the pointwise feature-based methods. 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 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 a same point cloud (see the neighboring points in Fig. 1), the convolution kernel should be able to distinguish them and

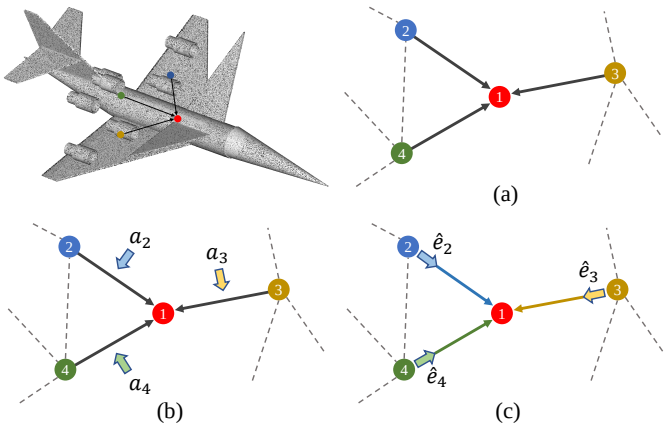


Fig. 1. Illustration of adaptive kernels and fixed kernels in the convolution. (a) The standard graph convolution applies a fixed/isotropic kernel (black arrow) to compute features for each point indistinguishably. (b) Based on these features in (a), several attentional weights a_i are assigned to determine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

M. Wei, Z. Wei, F. Hu, H. Si, Z. Chen, Z. Zhu, J. Qiu, X. Yan and J. Wang are with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China.
H. Zhou and Y. Guo are with the State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China.
J. Qin is Hong Kong Polytechnic University, Hong Kong, China.

determine their individual contributions.

To address this drawback, several approaches [14], [16] are proposed inspired by the idea of attention mechanism [17], [18]. 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)). In this regard, attentional convolutions cannot solve the inherent limitations of current graph convolutions, making it still difficult to capture the delicate geometric features of a point by considering its structural connections to its neighboring points distinctively rather than uniformly. Considering the intrinsic isotropy of current graph convolutions, these attempts are still limited for detecting the most relevant part in the neighborhood.

In this paper, we propose a novel graph convolution operator to more thoroughly address the inherent yet long-standing limitation of traditional GCNs, in order to more effectively capture the geometric features of a point by more precisely harnessing its geometric correlations with its neighboring points; we call the operator *adaptive graph convolution* (AGConv). In the proposed AGConv, we adaptively establish the relationship between a pair of points according to their feature attributes instead of using fixed kernels; *to our knowledge, this is the first time*. Such adaptiveness represents the diversity of kernels applied to each pair of points deriving from their individual features, which are capable of more accurately reflecting the underlying geometric characteristics of the objects when compared with the uniform kernels. Furthermore, we explore several design choices for feature convolution, offering more flexibility to the implementation of AGConv. AGConv can be easily integrated into existing GCNs for point cloud analysis by simply replacing the existing isotropic kernels with the adaptive kernels $\hat{e}_i$ generated from AGConv, as shown in Fig. 1 (c).

This paper is extended from our previous work [19]. The contents and key features newly added from [19] are listed as:

1) We detail the design of kernel function for AGConv (refer to Fig. 4), which helps to follow our released source code. We also fully discuss the influence brought by the spatial transform network (STN) for the segmentation task. In ablation study, we replace the AGConv layers with the attentional convolution layers to show its better performance. To verify the practicability of AGConv, more large-scale yet real-captured point clouds which possess complex structures are involved. That is, a challenging outdoor scene dataset named Paris-Lille 3D (NPM3D) is tested. In addition, we provide more visual results to demonstrate the merit of AGConv to capture key distinguishable features.

2) To validate the AGConv's effectiveness for point cloud completion, we improve ECG-Net [20] by replacing its original graph convolution with AGConv. The improved version of ECG-Net is called iECG-Net. iECG-Net employs the so-called coarse-to-fine strategy, i.e., first recovering its global yet coarse shape and then increasing its local details to output the missing point cloud of input. The difference from the original graph convolution is that, our AGConv can not only extract adequate spatial structure information, but also extract local features more efficiently and precisely. Therefore, the final completion results can better represent the missing parts on the tested benchmarks.

3) To validate the AGConv's effectiveness for point cloud denoising, we improve Pointfilter by replacing its original encoder with AGConv. The improved version of Pointfilter is called iPointfilter. iPointfilter is an encoder-decoder network, which directly takes the raw neighboring points of each noisy point as input, and regresses a displacement vector to encourage this noisy point back to its ground-truth position. The difference from the original encoder is that, AGConv can better obtain a compact representation for each input patch. Experiments show that our iPointfilter outperforms the state-of-the-art deep learning techniques in terms of noise-robustness and sharp feature preservation.

4) To validate the AGConv's effectiveness for point cloud upsampling, we improve PU-Transformer by add the AGConv module in the upsampling head. The improved version of PU-Transformer is called iPU-Transformer. iPU-Transformer takes a sparse point cloud as input, and generate a dense point cloud. The difference from the original upsampling head is that, the AGConv can better capture potential detailed features from the sparse point cloud. Therefore, iPU-Transformer can achieve a better upsampling effect in the parts with detail features.

5) To validate the AGConv's effectiveness for point cloud circle extraction, we improve Circle-Net [21] by replacing its original graph convolution module with AGConv. The improved version of Circle-Net is called iCircle-Net. Most of existing approaches leverage classification and fitting operations independently, not synergizing with each other to accurately extract geometric primitives. Differently, iCircle-Net is an end-to-end classification-and-fitting network, in which the two operations of classifying circle-boundary points and fitting the circle can synergize with each other to improve the performance of circle extraction. The difference from the original graph convolution is that, our AGConv can better perceive the circle spatial structure information. Therefore, iCircle-Net obtains better circle boundary detection precision.

6) To validate the AGConv's effectiveness for point cloud registration, we improve RGM [22] by utilizing AGConv in the local feature extractor instead of the original graph convolution. We denote the improved RGM with AGConv as iRGM. iRGM first utilizes a local feature extractor to obtain point-wise features, then both edge generator and graph feature extractor are leveraged to excavate graph features between the source and target point clouds. In addition, the AIS module predicts the soft correspondence matrix and the LAP solver converts soft correspondences to hard correspondences. Finally, the transformation is solved by SVD. Compared with the original version, AGConv can extract more discriminative and robust features for each point, thus boosting the performance of registration.

Extensive experiments demonstrate the effectiveness of our AGConv, achieving state-of-the-art performances in classification, segmentation, denoising, completion, upsampling, circle extraction and registration tasks on many benchmark datasets.

2 Related work

Although achieving the tremendous success in 2D grid-like structures, deep learning is still not well explored for 3D point cloud analysis. We will review previous researches categorized as voxelization-, projection-, graph- and point-based methods.

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 [23], [24], [25], [26] project point clouds onto regular grids, but inevitably suffer from both the information loss and enormous computational cost. To alleviate these problems, OctNet [7] and Kd-Net [27] attempt

to exploit more efficient data structures and skip the computations on empty voxels. Alternatively, the multiview-based methods [28], [29] 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 perspectives.

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 [11] 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 the semantic segmentation related tasks. To solve this issue, PointNet++ [30] proposes to apply PointNet layers locally in a hierarchical architecture to capture the regional information. Alternatively, Huang et al. [31] sorts unordered 3D points into an ordered list and employs Recurrent Neural Networks (RNN) to extract features according to different dimensions. In order to process a set of points that are unordered and discrete, there also exist efforts of sorting the 3D points into an ordered list. [27], [32] propose to apply the Kd-tree structure to build a 1D list for points according to their coordinates. Although alleviating the unstructured problem, the sorting process is critical to the weight functions, and local geometric information may not be easily preserved in a specific ordered list.

More recently, various approaches have been proposed for effective local feature learning. PointCNN [33] aligns points in a certain order by predicting a transformation matrix for the local point set, which inevitably leads to sensitivity in point orders since the operation is not permutation-invariant. SpiderCNN [34] defines its convolution kernel as a family of polynomial functions, relying on the neighbors' order. PCNN [35] designs point kernels based on the spatial coordinates and further KPConv [36] presents a scalable convolution using explicit kernel points. RS-CNN [37] assigns channel-wise weights to neighboring point features according to the geometric relations learned from 10-D vectors. ShellNet [38] splits a local point set into several shell areas, from which features are extracted and aggregated. Recently, Zhao et al. [39] and Guo et al. [40] independently utilize the successful transformer structures in natural language processing [41], [42] to build dense self-attentions between the local and global features.

Graph-based methods. 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. The notion of Graph Convolutional Network is proposed by [43], which generalizes convolution operations over graphs by averaging features of adjacent nodes. Similar ideas [12], [33], [44], [45], [46] have been explored to extract local geometric features from local points. Shen et al. [44] define kernels according to Euclidean distances and geometric affinities in the neighboring points. DGCNN [12] gathers nearest neighboring points in the feature space, followed by the EdgeConv operators for feature extraction, in order to identify semantic cues dynamically. MoNet [47] defines convolution as Gaussian mixture models in a local pseudo-coordinate system. Inspired by the attention mechanism, several works [14], [16], [48] propose to assign proper attentional weights to different points/filters. 3D-GCN [13] develops deformable kernels, focusing on shift and scale-invariant properties in point cloud analysis. diffConv [49] operates on spatially-varying and density-dilated neighborhoods, which are further adapted by a learned masked attention mechanism.

In order to expand the receptive field of graph convolutions, they either use graph pooling [13], [36] to gradually reduce the point numbers or a dynamic graph mechanism [12], [50] to connect similar points in the feature space. Both methods will change the graph construction structure which is easier to propagate local features throughout the point cloud. Therefore, the neighborhood of each point is varying between layers, determined dynamically according to the sampling strategy or feature similarity. That is, the relationships (edge features) are largely diverse among points not only in the neighborhood of a central point but also between any pair of points in the point cloud. Previous methods try to use a static function over each pair of points, neglecting their different feature correspondence from the previous layers. Differently, we propose to adaptively establish this varying relationship between a pair of points according to their feature attributes. This adaptiveness represents the diversity of weight kernels applied on each pair of points deriving from their individual features.

Convolution on point clouds. Many methods are proposed to define a proper convolution on point clouds. To improve the basic designs using fixed MLPs in PointNet/PointNet++, a variety of works [14], [16], [36], [37], [48] try to introduce weights based on the learned features, with more variants of convolution inputs [12], [34], [47]. However, the limitation caused by the isotropy of convolution still exists for those relying on fixed kernels.

Convolution on meshes. Polygonal meshes are a compelling representation form of 3D geometry. MeshCNN [51] operates directly on irregular triangular meshes, performing convolution and pooling operations designed in the harmony with the unique mesh properties. SubdivNet [52] performs 3D geometry learning on meshes with loop subdivision sequence connectivity using a mesh pyramid structure for feature aggregation. MeshNet [53] applies mesh convolution to expand the receptive field of the surface, which is realized by accumulating the information of adjacent surfaces. SpiralNet++ [54] formulates the order of aggregating neighbouring vertices, and fuses neighbouring node features with local geometric structure information. Nevertheless, the mesh convolution is vulnerable to adversarial remeshing attacks, and sometimes, the operation needs a trade-off between the mesh quality and the base mesh size.

Other methods [55], [56], [57] try to learn a dynamic weight for the convolution. However, their ideas are to approximate weight functions from the direct 3D coordinates while AGConv uses features to learn the kernels, which represents more adaptiveness. In addition, their implementations are heavily memory consuming when convolving with the high-dimensional features. Thus, the main focus of this work is to handle the isotropy of point cloud convolutions, by developing an adaptive kernel that is unique to each point in the convolution.

3 METHODOLOGY

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

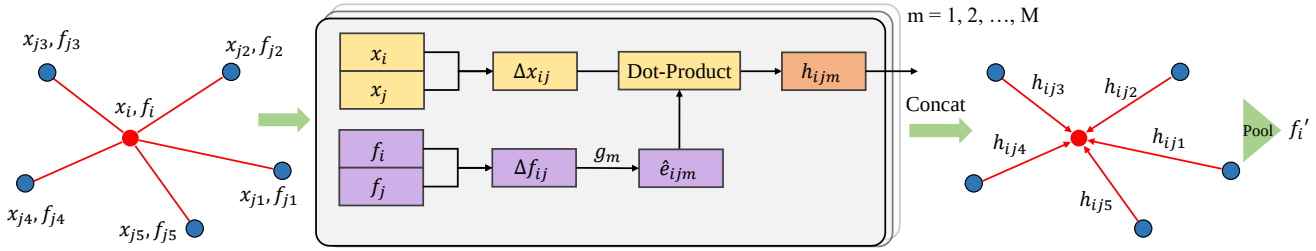


Fig. 2. The illustration of AGConv processed in the neighborhood of a target point x_i. An adaptive kernel $\hat{e}_{ijm}$ is generated from the feature input Δf_{ij} of a pair of points on the edge, which is then convolved with the corresponding spatial input Δx_{ij}. Concatenating h_{ijm} of all dimensions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

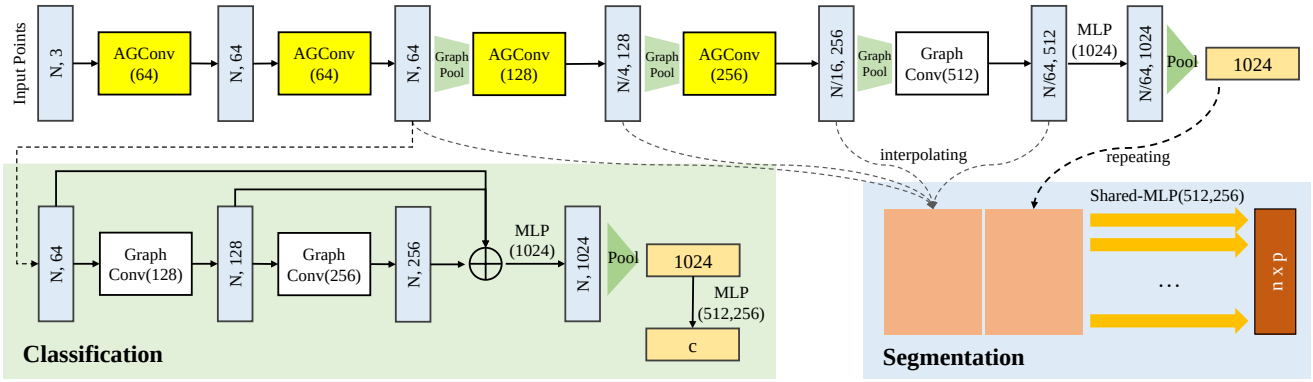


Fig. 3. AGConv for classification and segmentation. GraphConv denotes our standard convolution without an adaptive kernel. The segmentation model uses pooling and interpolating to build a hierarchical graph structure, while the classification model applies a dynamic structure [12].

3.1 Adaptive graph convolution

We denote the input point cloud as $\mathcal{X} = \{x_i | i = 1, 2, ..., N\} \in \mathbb{R}^{N \times 3}$ with the corresponding features defined as $\mathcal{F} = \{f_i | i = 1, 2, ..., N\} \in \mathbb{R}^{N \times D}$. Here, x_i processes the $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ 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 $\mathcal{V} = \{1, ..., N\}$ is the set of points (nodes), 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 AGConv 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 AGConv dynamically generates a kernel using a function over the point features (f_i, f_j):

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

Here, $m = 1, 2, ..., M$ indicates one of the M output dimensions corresponding to a single filter defined in our AGConv. In order to combine the global shape structure and feature differences captured in a local neighborhood [12], 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. $g(\cdot)$ 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):

$$h_{ijm} = \sigma \left\langle \hat{e}_{ijm}, \Delta x_{ij} \right\rangle, \tag{2}$$

where Δx_{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 σ is a nonlinear activation function. As shown in Fig. 2 (middle), the m-th adaptive kernel $\hat{e}_{ijm}$ is combined with the spatial relations Δx_{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 are efficiently incorporated into each layer, combined with the feature correspondences extracted dynamically from our kernel. Stacking h_{ijm} of each channel yields the edge feature $h_{ij} = [h_{ij1}, h_{ij2}, ..., h_{ijM}] \in \mathbb{R}^M$ between the connected points (x_i, x_j). Finally, we define the

output feature of the central point x_i by applying aggregation over all the edge features in the neighborhood (see Fig. 2 (right)):

$$f_i' = \max_{j \in \mathcal{N}(i)} h_{ij}, \tag{3}$$

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

3.2 Feature decisions

In our method, AGConv 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 Δx_{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 AGConv can also capture the relationships between feature dimensions and spatial coordinates which are from different domains. Note that, this is another option in our AGConv 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 Δx_{ij} with Δf_{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); and (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. 4.5.

3.3 Network architectures for classification and segmentation

We design individual network architectures for the point cloud classification and segmentation tasks using the proposed AGConv layer. The network architectures are shown in Fig. 3. The standard graph convolution layer with a fixed kernel uses the same feature inputs Δf_{ij} as in the adaptive kernels.

Dynamic graph update. Following [12], we update the graph structure 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 $\mathcal{N}(i) = \{j_{i_1}, j_{i_2}, ..., j_{i_k}\}$ such that the corresponding features $f_{j_{i_1}}, f_{j_{i_2}}, ..., f_{j_{i_k}}$ are closest to f_i. This encourages the network to organize the graph semantically, grouping together similar points in the feature space but not solely considering their proximity in the spatial inputs. Thus, the receptive field of local points is expanded, leading to a propagation of local information throughout the point cloud. Note that, in the convolution with adaptive kernel in Eq. 2, Δx_{ij} corresponds to the feature pair (f_i, f_j) which may not be spatially close.

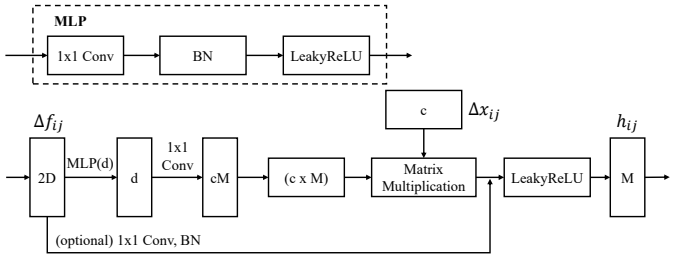


Fig. 4. Kernel function in our adaptive convolution. We apply a two-layer MLP for the adaptive weight matrix. The output edge feature is obtained by matrix multiplication between Δx_{ij} and the weight matrix. Optional ResNet block: shortcut 1×1 convolution and batch normalization layer.

Kernel function for adaptive convolution. In our experiments, the AGConv kernel function g_m is implemented as a two-layer shared MLP with residual connections to extract important geometric information. It is an inevitable choice to use a shared mapping as the kernel function. However, g_m is not the convolution kernel (fixed kernel) that is applied to points, but is to explore the different feature correspondences for different pairs of points. In the implementation, we process all g_m $(m = 1, 2, ..., M)$ together and obtain the adaptive kernels for the following convolution (see Fig. 4). The first layer is one shared MLP(d) for all g_m, and we organize the kernels as a weight matrix $(c \times M)$ which is then applied to the corresponding Δx_{ij} of dimension c by matrix multiplication. After a LeakyReLU, the edge feature h_{ij} is obtained and finally we apply Eq. 3 for the output feature of the central point. The ResNet connection is an optional block used in our segmentation model. From this perspective, the generation of adaptive kernels can be regarded as the generation of a weight matrix which directly produces the M-dimensional feature.

Graph pooling. For the segmentation task, we reduce the number of points progressively in order to build the network in a hierarchical architecture. The point cloud is subsampled by the furthest point sampling algorithm [11] with a sampling rate of 4, 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 sub-cloud can be simply obtained by a max-pooling function within its neighborhood. Alternatively, we can use an AGConv layer to aggregate this pooled features. To predict point-wise labels for the segmentation purpose, we interpolate deeper features from the subsampled cloud to the original points. Here, we use the nearest upsampling scheme to get the features for each layer, which are concatenated for the final point-wise features.

Segmentation network. Our segmentation network architecture is illustrated in Fig. 3. The AGConv 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. In addition, our segmentation network includes a spatial transformer network [58] before the convolution layers. It processes the input points and outputs a 3×3 matrix in order to apply a global transformation. We apply standard graph convolutions (64, 128, 1024), followed by a max pooling function and fully-connected layers (512, 256). The output matrix is initialized as an identity matrix. Here, it is also

Method	mcIoU(%)	mIoU(%)
w/o STN	83.2	86.2
STN	83.4	86.4

TABLE 1
Segmentation results for models using STN.

Method	Input	#points	mAcc(%)	OA(%)
3DShapeNetParts [24]	voxel	-	77.3	84.7
VoxNet [23]	voxel	-	83.0	85.9
Subvolume [59]	voxel	-	86.0	89.2
PointNet [11]	xyz	1k	86.0	89.2
PointNet++ [30]	xyz, normal	5k	-	91.9
Kd-Net [27]	xyz	1k	-	90.6
SpecGCN [60]	xyz	1k	-	92.1
SpiderCNN [34]	xyz, normal	5k	-	92.4
PointCNN [33]	xyz	1k	88.1	92.2
SO-Net [61]	xyz, normal	5k	-	**93.4**
DGCNN [12]	xyz	1k	90.2	92.9
KPConv [36]	xyz	6.8k	-	92.9
3D-GCN [13]	xyz	1k	-	92.1
PointASNL [62]	xyz, normal	1k	-	93.2
Ours	xyz	1k	**90.7**	**93.4**

TABLE 2
Classification results on ModelNet40. Our network achieves the best results according to both mAcc and OA.

possible to replace these graph convolutions with AGConv layers, but this does not lead to a significant improvement. Since the input contains normals as additional attributes, we apply the 3×3 matrix separately to the point and normal dimensions. The STN module can be seen as a global adaptive kernel that is convolved with all input points similar as in our AGConv. We report the results of networks with and without STN in Tab. 1.

Classification network. The classification network uses a similar encoder as in the segmentation model (see Fig. 3). For sparser point clouds used in ModelNet40, we simply apply dynamic graph structures [12] 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 $\mathcal{N}(i) = \{j_1, j_2, ..., j_k\}$ such that the corresponding features $f_{j_1}, f_{j_2}, ..., f_{j_k}$ 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 AGConv for point cloud classification, part segmentation and indoor/outdoor segmentation.

4.1 Classifcation

Data. We evaluate our model on ModelNet40 [24] for classification. It 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 [11]. 1024 points are uniformly sampled for each object and we only use the $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ coordinates of sampled points as input. Data augmentation includes shifting, scaling and perturbing of the points.

Network configuration. The network architecture is shown in Fig. 3. Following [12], 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. Shortcut connections are included and one shared fully-connected layer (1024) is applied to aggregate the multi-scale features. The global feature is obtained using a max-pooling function. All layers are with LeakyReLU and batch normalization. We use the SGD optimizer with the momentum set to 0.9. The initial learning rate is 0.1 and is dropped until 0.001 using cosine annealing [63]. The batch size is set to 32 for all training models. We use PyTorch [64] for 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. 2. The evaluation metrices on this dataset are the mean class accuracy (mAcc) and the overall accuracy (OA). Our model achieves the best scores on this dataset. For a clear comparison, we show the input data types and the number of points corresponding to each method. Our AGConv only considers the point coordinates as input with a relatively small size of 1k points, which already outperforms other methods using larger inputs.

4.2 Part segmentation

Data. We further test our model for the part segmentation task on the ShapeNetPart dataset [65]. This dataset contains 16,881 shapes from 16 categories, with 14,006 for training and 2,874 for testing. Each point is annotated with one label from 50 parts and each point cloud contains 2-6 parts. We follow the experimental setting of [30] and use their provided data for the benchmarking purpose. 2,048 points are sampled from each shape. The input attributes include the point normals apart from the 3D coordinates.

Network configuration. Following [11], 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. We use spatial positions (without normals) as Δx_{ij} as in Sec. 3.2. Other choices will be evaluated later in Sec. 4.5.

Results. We report the mean class IoU (mcIoU) and mean instance IoU (mIoU) in Tab. 3. Following [11], IoU of a shape is computed by averaging 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 state-of-the-art performance compared with other methods.

4.3 Indoor scene segmentation

Data. Our third experiment shows the semantic segmentation performance of our model on the S3DIS dataset [75]. 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 [11], [66], [76], 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 noise and outliers. Thus, we follow the experimental settings of KPConv [36], and train the network using randomly

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 [27]	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 [11]	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++ [30]	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 [61]	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 [12]	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 [33]	-	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 [62]	-	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 [13]	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]	**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	83.4	**86.4**	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

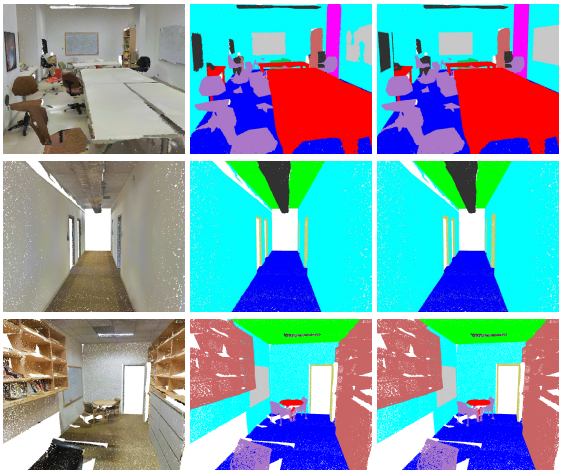
TABLE 3
Part segmentation results on ShapeNetPart evaluated by the mean class IoU (mcIoU) and mean instance IoU (mIoU).

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	**83.4**	**86.4**

TABLE 4
Ablation studies on ShapeNetPart for part segmentation.

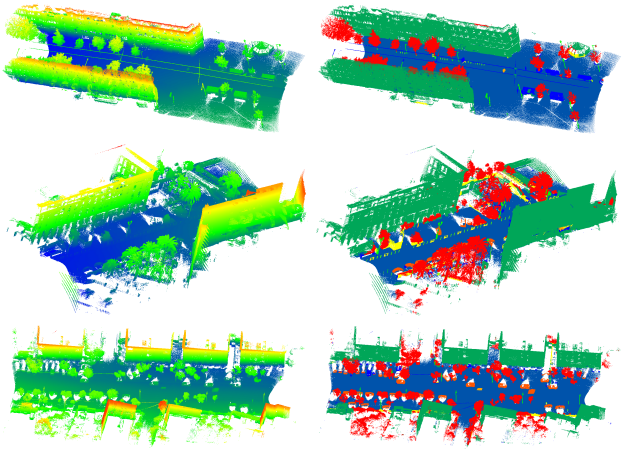
Method	mAcc(%)	OA(%)
GraphConv	88.8	92.5
Attention Point	88.5	92.1
Attention Channel	89.2	92.2
Ours	90.7	93.4

TABLE 5
Results of ablation networks on ModelNet40.



(a) Input (b) Prediction (c) Ground Truth

Fig. 5. Visualization of semantic segmentation results on S3DIS. We show the input point cloud, and labelled points mapped to RGB colors.



(a) Input (b) Prediction

Fig. 6. Visualization of semantic segmentation results on NPM3D. We show the input point cloud, and labelled points mapped to RGB colors.

sampled clouds in spheres. The subclouds contain more points with varying sizes, and are stacked into batches for training. During testing, spheres are uniformly picked in the scenes, and 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. 6. The IoU of each class is also provided. The proposed AGConv 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. 5 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 Outdoor scene segmentation

Data. Paris-Lille 3D (NPM3D) is a large-scale urban point cloud dataset acquired by a Mobile Laser System (MLS). It contains 160 million points in total scanned from four different cities. To help the segmentation and classification tasks, the point cloud has been annotated to 10 coarse classes. To ensure a fair comparison, test labels are hidden and they provide only an online benchmark.

Challenge. Compared with the indoor scenes, outdoor objects are often larger in scale, more complex in structure and contain much more noise. Moreover, the captured point clouds of outdoor scenes are sparse and have a large amount of object classes. However, single objects, especially the small targets, are usually

Method	OA	mAcc	mIoU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [11]	–	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 [66]	–	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 [33]	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 [67]	–	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 [68]	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 [69]	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 [14]	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 [62]	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	**90.0**	**73.2**	**67.9**	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 6
Semantic segmentation results on S3DIS evaluated on Area 5. We report the mean classwise IoU (mIoU), mean classwise accuracy (mAcc) and overall accuracy (OA). IoU of each class is also provided.

Method	mIoU	ground	building	pole	bollard	trash can	barrier	pedestrian	car	natural
RF_MSSF [70]	61.5	99.1	90.5	66.4	62.6	5.8	52.1	5.7	86.2	84.7
HDGCN [71]	68.3	99.4	93.0	67.7	75.7	25.7	44.7	37.1	81.9	89.6
MS3_DVS [72]	66.9	99.0	94.8	52.4	38.1	36.0	49.3	52.6	91.3	88.6
KP-edgeret [73]	69.9	99.2	92.9	60.0	65.7	49.0	38.6	48.6	85.4	89.7
ConvPoint [74]	75.9	99.5	95.1	71.6	88.7	46.7	52.9	53.5	89.4	85.4
KPConv [36]	75.9	-	-	-	-	-	-	-	-	-
Ours	**76.9**	99.4	97.3	67.8	77.1	49.4	59.4	55.5	93.2	93.2

TABLE 7
Semantic segmentation results on NPM3D. We report the mean classwise IoU (mIoU) and IoU of each class.

sampled with very few points. These challenges cannot make the downstream applications (e.g., segmentation, detection and recognition) operate smoothly. To alleviate the aforementioned problems, a deep network that possesses a powerful ability of extracting features is more than welcome. Following the experimental setup in Sec. 4.3, we have implemented AGConv by adopting the architecture of KPConv to realize the segmentation, and the specific details will not be described here.

Results. To demonstrate the effectiveness of AGConv, we compare our method with KPConv [36], ConvPoint [74], MS3_DVN [72], HDGCN [71], and RF-MSSF [70]. The final mIoU results and the IoU of each class are reported in Tab. 7. All quantitative results come from the official website of the dataset or the corresponding papers. After performing AGConv on KPConv, mIoU of the scene segmentation has been improved, which is superior to the competitors. This indicates that our adaptive convolution can identify effective information of each feature map, and capture different relations between points from different semantic components. We show visualized segmentation results in Fig. 6 where our method correctly identifies outdoor sparse point cloud scenes.

4.5 Ablation studies

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

Adaptive convolution vs Fixed kernels. We compare our AGConv with fixed kernel convolutions, including methods using the attention mechanism and standard graph convolution (DGCNN [14]), as discussed in the introduction. We train these models on ShapeNetPart for segmentation, and design several ablation networks by replacing AGConv layers with fixed kernel layers and keeping other architectures the same.

Specifically, [16] assigns attentional weights to different neighboring points and [14] further designs a channel-wise attentional function. We use their layers and denote these two ablations as Attention Point and Attention Channel in Tab. 4 respectively. We only replace the AGConv layers in our network and the feature inputs Δf_{ij} are the same as our model. Besides, we also show the result by using standard graph convolutions (GraphConv), which can be seen as a similar version of DGCNN [12]. From the comparison, we see that our method achieves better results than the fixed kernel graph convolutions.

Feature decisions. In AGConv, the adaptive kernel is generated from the feature input Δf_{ij}, and subsequently convolved with the corresponding Δx_{ij}. Note that, in our experiments, Δx_{ij} corresponds to the $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ spatial coordinates of the points. We have discussed several other choices of Δx_{ij} in Eq. 2 in Sec. 3.2, which can be evaluated by designing these ablations:

• Feature - In Eq. 2, we convolve the adaptive kernel $\hat{e}_{ijm}$ with their current point features. That is, Δx_{ij} is replaced with Δf_{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 ShapeNetPart, 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 scores are shown in Tab. 4. As one can see, $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is the most critical initial attribute (probably the only attribute) in point clouds, thus it is recommended to use them in the convolution with adaptive kernels. Although achieving a promising result, the computational cost for the Feature ablation is extremely high since the network expands heavily when it is

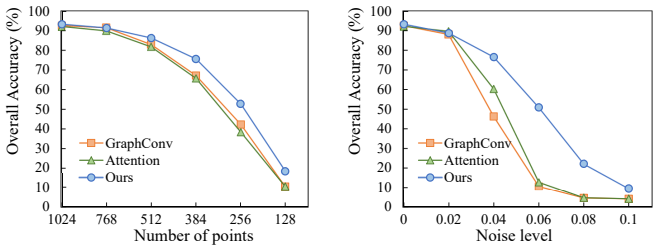


Fig. 7. Robustness test on ModelNet40 for classification. GraphConv indicates the standard graph convolution network. Attention indicates the ablation where we replace the AGConv layers with graph attention layers (point-wise). From the comparison, we can see that our model is more robust to point density 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 8
Our classification network with different k of nearest neighbors.

convolved with a high-dimensional feature.

Attentional convolution. To compare our model with attentional graph convolutions, we design several ablations which replace AGConv layers with attentional convolution layers. Following the design of [16], the output feature is formulated as:

$$f_i' = \max_{j \in \mathcal{N}(i)} a_{ij} * h(f_j), \tag{4}$$

where $h : \mathbb{R}^D \rightarrow \mathbb{R}^M$ is a shared MLP and a_{ij} is the attentional weight calculated as:

$$a_{ij} = \mathbf{softmax}_j(\alpha(\Delta f_{ij})). \tag{5}$$

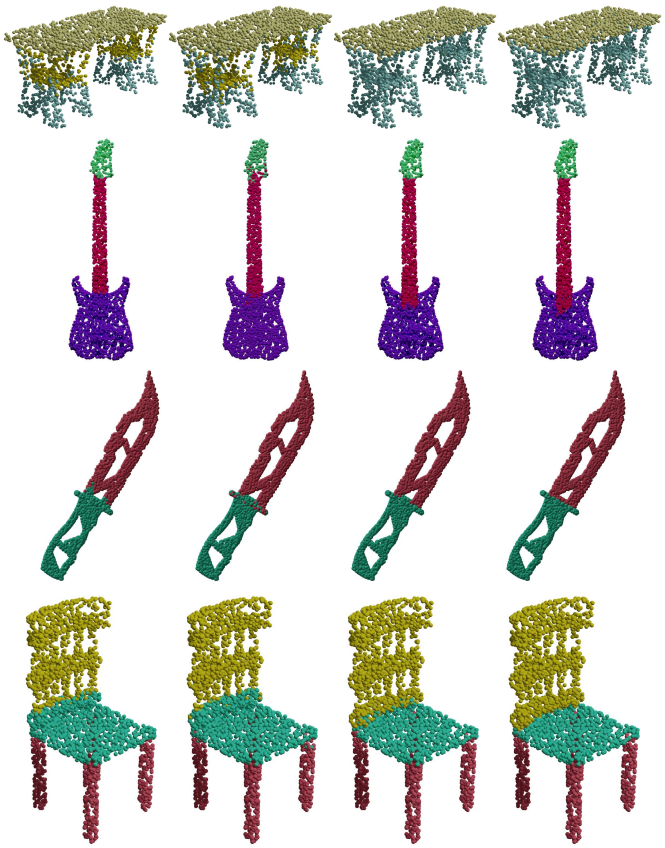
Here, $\alpha(\cdot)$ is a mapping function, and $\Delta f_{ij} = [h(f_i), h(f_j) - h(f_i)]$ since the attentional weights are applied to $h(f_j)$ instead of f_j. A softmax is used to make $\sum_j a_{ij} = 1, j \in \mathcal{N}(i)$. In Sec. 4.3 of the main paper, the point-wise attentional weight (Attention Point) uses $a_{ij} \in \mathbb{R}$, i.e., the function is $\alpha : \mathbb{R}^{2M} \rightarrow \mathbb{R}$. The channel-wise attenional weight (Attention Channel) uses $a_{ij} \in \mathbb{R}^M$ and, in this case, $*$ denotes the element-wise product.

The attentional weights a_{ij} are based on the produced features $h(f_j)$ in order to determine the different contributions of the neighboring points. However, since the applied convolution kernel $h(\cdot)$ is still a fixed/isotripic one as we discussed in the main paper, they still cannot solve the intrisic limitation of current graph convolutions. The results of these ablation networks trained on the ShapeNetPart dataset are given in Sec. 4.3. Furthermore, we show more comparisons on ModelNet40 for classification in Tab. 5.

4.6 Robustness test

We further evaluate the robustness of our model to point cloud density and noise perturbation on ModelNet40 [24]. We compare our AGConv with several other graph convolutions as discussed in Sec. 4.5. 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. 7, we can see that our method is robust to missing data and noise, thanks to the adaptive kernel in which the structural connections is extracted dynamically in a sparser area.

Also, we experiment the influence of different numbers k of the nearest neighboring points in Tab. 8. 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 does not improve the performance since the local information dilutes within a larger neighborhood.

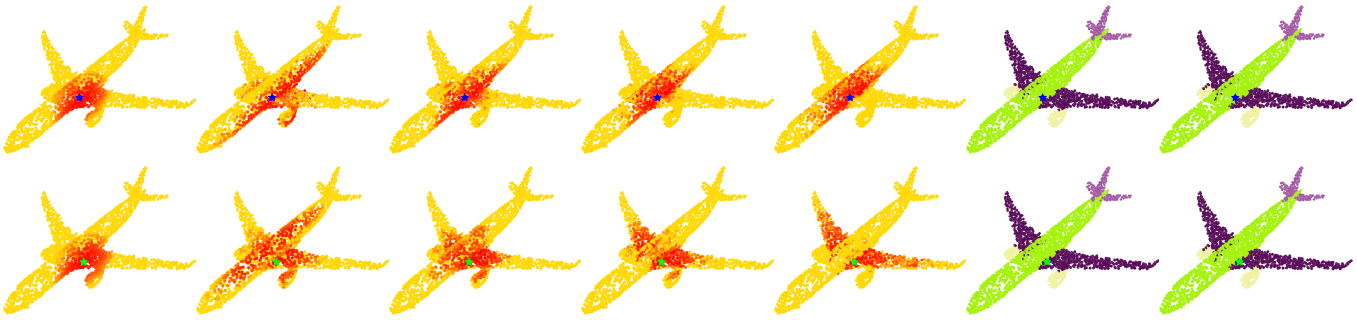


(a) DGCNN (b) Attention (c) Ours (d) GT

Fig. 8. Segmentation results on ShapeNet. The labelled points are visualized in different colors. We compare our adaptive graph convolution with DGCNN [12] (standard graph convolution) and attentional convolution network (Attention Point). Our method produces better results especially for points close to the object boundaries and edges.

Method	#parameters	OA(%)
PointNet [11]	3.5M	89.2
PointNet++ [30]	1.48M	91.9
DGCNN [12]	1.81M	92.9
KPConv [36]	14.3M	92.9
Ours	1.85M	**93.4**

TABLE 9
The number of parameters and overall accuracy of different models.



(a) Spatial (b) Layer1 (c) Layer2 (d) Layer3 (e) Layer4 (f) Ours (g) GT

Fig. 9. Visualize the Euclidean distances between two points (blue and green stars) and other points in the feature space (red: near, yellow: far).

Method	#parameters	Time(ms)	OA(%)
Baseline (w/o AGConv)	1.81M	93.1	92.5
AGConv (Layer2)	1.85M	129.1	93.4
AGConv (Layer3)	1.95M	168.4	93.0
AGConv (Layer4)	2.35M	276.0	93.2

TABLE 10
Number of parameters, forward pass time (per batch) and overall accuracy for different models using AGConv.

(a) Spatial (b) Layer1 (c) Layer2 (d) Layer4 (e) Ours (f) GT

Fig. 10. Visualize the Euclidean distances between the target point (green point in (a)) and other points in the feature space. Red color denotes a closer point and yellow one is far from the target. The feature distances in several layers of the network provide a clear insight that our network is able to distinguish points belonging to different semantic parts. It also captures non-local similar structures (see the handles in the second row).

4.7 Efficiency

To compare the complexity of our model with the state-of-the-arts, we show the parameter numbers and the corresponding results of networks in Tab. 9. These models are based on ModelNet40 for classification. Our model achieves the best performance of 93.4% overall accuracy and the model size is relatively small. Compared with DGCNN [12] which can be seen as a standard graph convolution version in our ablation studies, the proposed adaptive kernel performs better while being efficient.

4.8 Model complexity

The standard graph convolution adopted in this work contains $2DM$ parameters ($2D$ denotes the dimension of feature input Δf_{ij}). Here, D and M denote the input and output dimensions respectively. As described in Sec. 3.3, the kernel function uses a two-layer MLP which contains $dD + dcM$ parameters where c is the dimension of Δx_{ij} ($c = 3 \times 2$ for point coordinates $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ input). d is the dimension of hidden layer of the kernel function (see Fig. 2) and it can be adjusted to reduce the model size. In practice, we design the network architecture with two layers of AGConv which already achieves a pleasing performance. We further report the results and parameter numbers on ModelNet40 using different numbers of AGConv layers in Tab. 10. The adopted design (Layer2) significantly improves the network performance while the model size is relatively small. For the time performance evaluation, we also report the forward pass times of different models. The proposed AGConv layer is able to improve the performance of existing graph CNNs while being efficient.

5 Visualization and Learned Features

In this section, we provide visual results to further demonstrate the effectiveness of the proposed AGConv over fixed kernel methods. We first visualize the segmentation results on ShapeNetPart in Fig. 8. In this experiment, we compare the results of DGCNN [12], attentional graph convolution (Attention Point described in Sec. 4.5) and AGConv. Our results are better in challenging regions, such as part boundaries and object edges. This verifies that our method is able to capture distinguishable features for points belonging to different parts. More visualizations on ShapeNetPart are given in Fig. 21.

To achieve a deeper understanding of AGConv, we explore the feature relations in several intermediate layers of the network to see how AGConv can distinguish points with similar spatial inputs. In this experiment, we train our model on ShapeNetPart for segmentation. In Fig. 9, 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. 9, 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. As shown in Fig. 10, when the point is close to edges between different semantic parts, our network encourages it to have distinguishable features which captures better geometric information. Thus, it is separated from other parts of the objects, as shown in the first row of Fig. 10. Also, we see that in the second row of Fig. 10, points belonging to the same semantic part share similar features while they may not be spatially close. Note that, Fig. 10(a) indicates the spatial distances with regard to the central point.

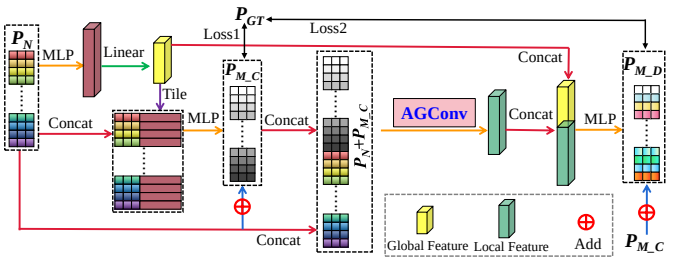


Fig. 11. ECG-Net [20] is improved by our AGConv for point cloud completion (call iECG-Net). iECG-Net can better extract local features to recover the shape details by AGConv. iECG-Net 1) first generates the coarse result P_{M_C}, and 2) concatenates P_{M_C} with the input P_N to produce $P_N + P_{M\_C}$, and 3) produces the final yet fine result P_{M_D}.

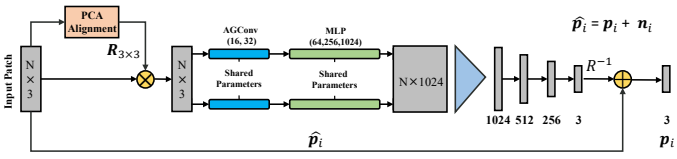


Fig. 12. Pointfilter [83] is improved by our AGConv for point cloud denoising (call iPointfilter). The main idea of iPointfilter is to project each noisy point onto the clean surface according to its neighboring structure. Given a noisy patch with N points, PCA is utilized for alignment and then the aligned patch is fed into iPointfilter. Both AGConv and MLP are used to extract features, and then all the features are aggregated by max pooling. Finally, three fully connected layers are used to regress a displacement vector between the noisy point cloud and the ground truth. The outputs of the first two layers are processed by Batch Normalization and Relu, and the last layer only uses the tanh activation function to constrain the displacement vector.

6 More Point Cloud Analysis Applications

AGConv, as a plug-and-play module, can flexibly serve more point cloud analysis approaches to boost their performance. We develop AGConv-based completion, denoising, upsampling, circle extraction and registration networks in this section. For each task, we introduce the network framework, data preparation and comparison in a concise way. *More details will be found in our prepared webpage.*

6.1 Point cloud completion

Raw point clouds captured by 3D sensors are often incomplete. When employing these untreated point clouds for semantic understanding, users may receive inaccurate or even wrong results. Point cloud completion aims to infer the whole underlying surface from a partial input. Moreover, the completion results should be uniform, dense and possess logically correct geometric structures.

Network framework. We leverage ECG-Net [20] as our network's backbone in Fig. 11. In our improved ECG-Net (iECG-Net), the original graph convolution module is replaced by the AGConv module. iECG-Net employs the so-called coarse-to-fine strategy, i.e., first recovering its global yet coarse shape and then increasing its local details to output the missing point cloud of input. First, we take the incomplete point cloud P_N as input, and use an encoder-decoder structure like PCN [77] to yield the coarse point cloud P_{M_C} to represent the missing part. Then, we concatenate the input incomplete point cloud P_N and the coarse missing part P_{M_C}. Finally, we take the concatenation result $(P_N + P_{M\_C})$ to the following detail refinement network that contains the AGConv layer. AGConv can not only extract adequate spatial structure information from $(P_N + P_{M\_C})$, but also extract local features more efficiently and precisely. The final completion result is P_{M_D} that represents the missing part. Thus, the whole complete point cloud is $(P_N + P_{M\_D})$. Please note that, in training, we calculate two losses from P_{M_C} and P_{M_D} with the ground truth P_{GT} by the Chamfer distance.

Data. We train and evaluate our iECG-Net in the benchmark dataset Shapenet-Part, which has 13 categories of different objects. Shapenet-Part has 14473 shape elements formatted by point clouds, in which 11705 point clouds are for training and 2768 for testing. In Shapenet-Part, the centers of all point clouds locate at the origin, and their coordinate values of xyz range within [-1,1]. We sample 2048 points uniformly from each point cloud as the complete shape. Then, we select some border points like (1,1,1) or (-1,1,1) as viewpoints and randomly choose a viewpoint for the point clouds in a same training batch, and remove a certain amount of points that are closest to the viewpoint. By such removal operation, we can produce the incomplete point clouds for training and testing. In our experiments, we aim at the problem of large-ratio incomplete point cloud completion. Thus, we set the ratio to be to 50% for both training and testing.

Comparison. To demonstrate the effectiveness of AGConv, we compare iECG-Net against several representative completion methods, including L-GAN [78], PCN [77], 3D-Capsual [79], TopNet [80], MSN [81], PF-Net [82], and ECG-Net [20]. In our experiments, we train all the methods without the category information for fairness. We test all methods in 13 categories in Tab. 12. Benefiting from AGConv, iECG-Net possesses higher average completion precision than its competitors in most cases. Moreover, the visualization results are given in Fig. 14, where our iECG-Net can better complete the large-missing parts.

6.2 Point cloud denoising

3D imaging devices are frequently used to capture the virtual models of physical objects. These models represented by point clouds are usually noisy due to measurement and reconstruction errors, and should be denoised to facilitate subsequent applications. Point cloud denoising aims to eliminate noise or spurious information from a noisy point cloud, while preserving its real geometry.

Network framework. We leverage Pointfilter [83] as our network's backbone in Fig. 12. In our improved Pointfilter (iPointfilter), the original encoder is replaced by the AGConv module. iPointfilter takes the noisy point as input, and outputs a displacement vector to move this noisy point to the underlying (noise-free) surface. The encoder attempts to obtain a complex representation for the input patch, and is mainly composed of two parts, a feature extractor to obtain features of different scales from the neighborhood, and a collector to aggregate the features as a latent vector. The extractor and collector are implemented by our AGConv layer and max pooling layer, respectively. The decoder is used as a regressor to return a displacement vector of the noisy point, which is realized by three fully connected layers.

Data. We train our iPointfilter on the benchmark dataset from Pointfilter [83], which contains 22 clean models (11 CAD models and 11 non-CAD models). Each model is generated from a random sampling of 100K points from the original surface. The clean models are then perturbed by Gaussian noise with the standard deviations from 0.0% to 2.5% of the bounding box's diagonal

Methods	MSE(10^{-3})	CD(10^{-5})
Noisy	44.77	104.12
Pointfilter	42.31	62.35
DGCNN	42.26	59.07
Ours	40.19	44.96

TABLE 11
Average errors of all filtered point clouds over our test synthetic models(20 models with 0.5% noise).

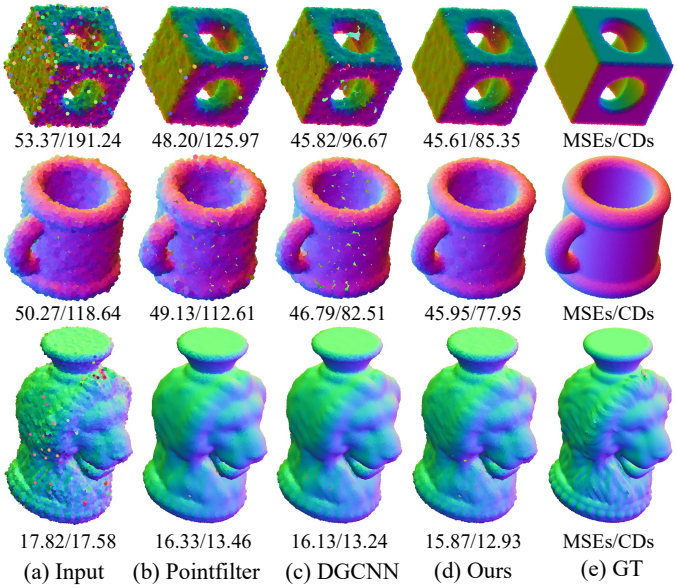


Fig. 13. Visual comparisons of point clouds with 0.5% noise (left: MSEs (10^{-3}), right: CDs (10^{-5})).

length. The training set consists of 132 models. In addition to the (x,y,z) coordinate of each point, the point normals of clean models are also required for training. To test the model, we randomly selected 20 models from the dataset of [84] and added different levels of Gaussian noise onto them.

Comparison. To evaluate the effectiveness of iPointfilter, we replace the encoder of Pointfilter with DGCNN [12]. As shown in Fig. 13, our methods can produce more evenly distributed results on the first two models and maintain more sharp features on the third models than the results of Pointfilter [83] and DGCNN [12].To comprehensively evaluate our iPointfiler, We also calculate the mean square error(MSE) and the chamfer distance(CD) over the 20 synthetic models in the test set. Table 11 shows our method averagely achieves the lowest errors.

6.3 Point cloud upsampling

Point cloud upsampling aims to generate dense point clouds from their sparse input. The generated data should recover the fine-grained structures at a higher resolution, and the upsampled points are expected to uniformly lie on the underlying surface.

Network framework. We leverage PU-Transformer [85] as our network's backbone in Fig. 15. In our improved PU-Transformer (iPU-Transformer), we add the AGConv module in the upsampling head. Given a sparse point cloud $P \in R^{N\times 3}$ as input, iPU-Transformer can generate a dense point cloud $Q \in R^{rN\times 3}$, where r denotes the upsampling scale. In Fig. 15, we first exploit

Fig. 14. Visualization of completion results on Shapenet-Part (4 categories: Car, Laptop, Table, and Chair).

AGConv and MLP to construct the upsampling head, which extracts a preliminary feature map from the input. Then, based on the feature map and the inherent 3D coordinates, the upsampling body gradually encodes a more comprehensive feature map via the cascaded Transformer encoders. Finally, in the upsampling tail, the shuffle operation [86] is used to form a dense feature map and reconstruct the 3D coordinates of Q via an MLP.

Data. We train and test iPU-Transformer on the PU1K dataset in PU-GCN [87]. PU1K covers 50 object categories, in which 1,020 3D meshes are used for training and 127 ones for testing. To match the patch-based upsampling methods, the training data is generated from patches of 3D meshes via Poisson disk sampling. Specifically, the training data includes a total of 69,000 samples (patches), where each sample has 256 points (low resolution) and a corresponding ground-truth of 1,024 points ($4\times$ high resolution).

Results. We follow the common way utilized in PU-Net [88], PU-GCN [87], and PU-Transformer [85]. To be specific, we first cut the input point cloud into multiple seed patches covering all N points. Then, we apply our trained model to upsample the seed patches with a scale of r. Finally, the farthest point sampling algorithm is used to combine all upsampled patches as a dense output point cloud with rN points. We test the point clouds with 2,048 points for the $4\times$ upsampling experiments. We quantitatively evaluate the upsampling performance of PU-Net [88], PU-GCN [87], PU-Transformer [85] and our method in Tab. 13 based on three widely used metrics: (i) Chamfer distance (CD), (ii) Hausdorff distance (HD), and (iii) Point-to-Surface distance (P2F). iPU-Transformer obtains a lower value under these metrics than its competitors. Moreover, we visually compare our

Category	LGAN-AE	PCN	3D-Capsule	TopNet	MSN	PF-Net	ECG-Net	AGConv(Ours)
Airplane	2.814	2.626	2.991	2.251	1.698	**0.984**	1.095	1.010
Bag	8.837	8.673	8.492	7.887	9.745	**3.543**	3.995	4.121
Cap	7.609	7.126	7.706	6.524	5.491	5.473	4.668	**3.576**
Car	5.416	5.789	6.236	5.514	5.716	2.390	2.496	**2.356**
Chair	4.787	4.153	4.045	3.597	3.072	2.053	2.124	**1.916**
Guitar	1.251	1.113	1.294	0.976	0.836	**0.407**	0.478	0.442
Lamp	7.476	6.918	7.669	6.534	3.517	4.185	3.467	**3.182**
Laptop	3.376	3.262	3.627	2.671	1.619	1.448	1.408	**1.348**
Motorbike	4.156	4.012	4.048	3.546	2.963	1.923	2.034	**1.888**
Mug	6.516	6.845	7.051	6.781	8.795	**3.377**	3.775	3.478
Pistol	3.261	3.163	3.212	2.620	1.647	1.381	**1.237**	1.271
Skateboard	3.022	2.906	3.346	2.717	1.760	1.327	1.354	**1.247**
Table	4.781	4.746	5.157	4.036	4.342	2.053	1.982	**1.922**
Mean	4.869	4.717	4.990	4.281	3.938	2.349	2.316	**2.135**

TABLE 12
Completion results on Shapenet-Part with 13 categories by the CD. The last row show the mean CD loss of all these categories, scaled by 1000.

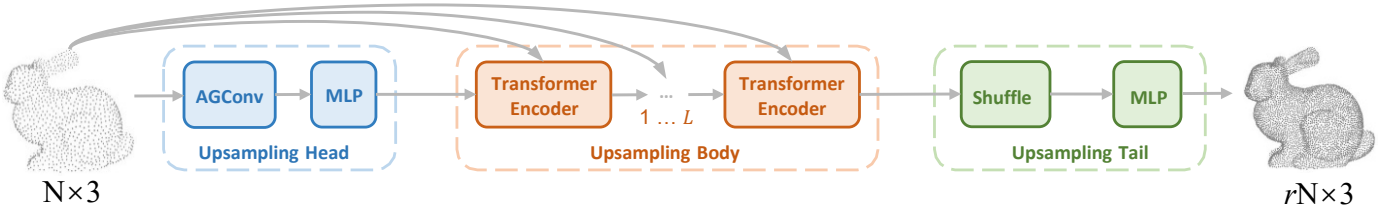


Fig. 15. PU-Transformer [85] is improved by our AGConv for point cloud upsampling (call iPU-Transformer).

Method	CD ($\times 10^{-3}$)	HD ($\times 10^{-3}$)	P2F ($\times 10^{-3}$)
PU-Net [88]	1.155	15.170	4.834
PU-GCN [87]	0.585	7.577	2.499
PU-Transformer [85]	0.451	3.843	1.277
Ours	**0.434**	**3.534**	**1.251**

TABLE 13
Quantitative comparisons on PU1K [87].

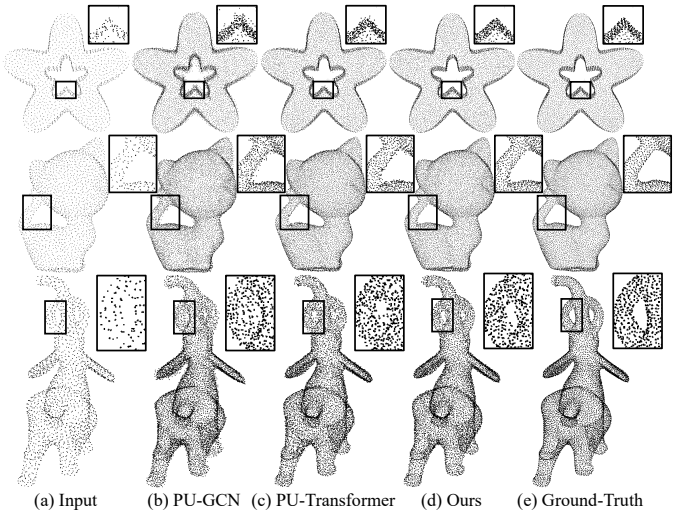


Fig. 16. Visualization of upsampling results on models (with 2,048 points for the $4\times$ upsampling) from PU-GAN [84].

iPU-Transformer with PU-GCN [87], and PU-Transformer [85] in Fig. 16. Benefiting from AGConv, iPU-Transformer can better upsample point clouds, which have detailed features.

6.4 Circle extraction

Geometric primitive extraction from man-made engineering objects is essential for many practically meaningful applications, such as reverse engineering and 3D inspection. The shape of circle is one of the fundamental geometric primitives of man-made engineering objects. Thus, extraction of circles from scanned point clouds is a quite important task in geometry data processing.

Network framework. We leverage Circle-Net [21] as our network's backbone in Fig. 17. In our improved Circle-Net (iCircle-Net), the original graph convolution module is replaced by the AGConv module. iCircle-Net leverages an end-to-end classification-and-fitting network: The circle-boundary learning module detects all potential circle-boundary points from a raw point cloud by considering local and global neighboring contexts of each point; the circle parameter learning module for weighted least squares is developed, without designing any weight metric to avoid the influence of outliers during fitting; the two modules are co-trained with a comprehensive loss to enhance the quality of extracted circles.

First, we build two different neighborhoods, for perceiving both local and global context information for each point. The deep features of the two patches are then extracted by AGConv and MLPs. Then, a transformer module is used to fuse the features of the two patches, and the fused features are exploited to regress each point's label. After classification, we use a neural network to estimate the weight of each point in the detected circle-boundary candidate points, which will be subsequently used for weighted least squares circle fitting.

Data. We train and evaluate our iCircle-Net in the benchmark

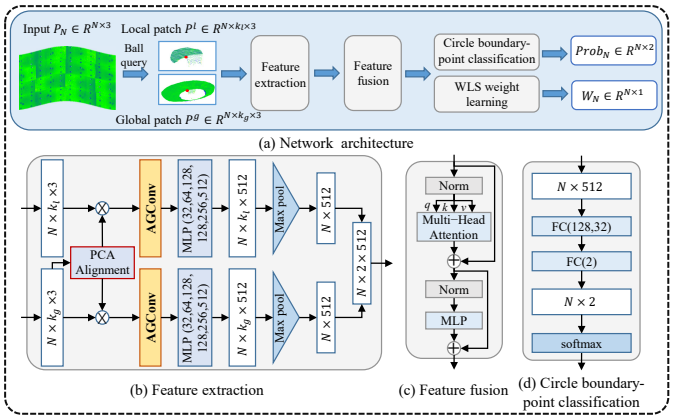


Fig. 17. Circle-Net [21] (a classification-and-fitting network) is improved by our AGConv for point cloud circle extraction (call iCircle-Net).

Method	$Precision(\%)$	$Recall(\%)$	$F1(\%)$
EC-Net [89]	32.60	26.82	29.43
PIE-NET [90]	77.95	69.79	73.64
Circle-Net [21]	86.06	77.62	81.58
Ours	**87.33**	**78.24**	**82.54**

TABLE 14
Quantitative comparisons of circle boundary detection on virtual scans.

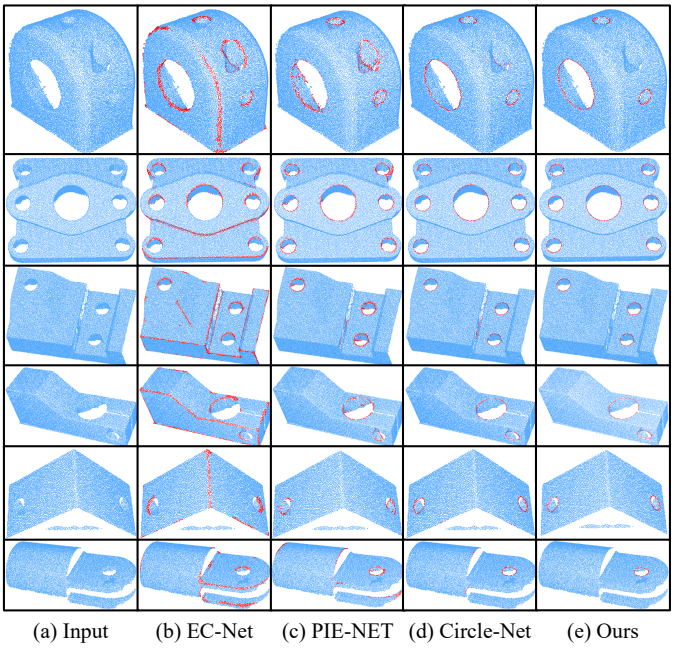


Fig. 18. Visualization of circle extraction results on six real scans.

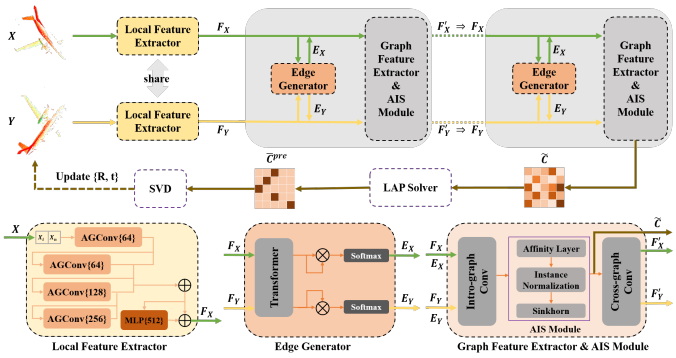


Fig. 19. RGM [22] is improved by our AGConv for point cloud registration (called iRGM).

dataset from Circle-Net [21], which contains 55 CAD models, including curved and flat thin-walled planes with multiple circular structures. Each CAD model is virtually scanned by a simulated scanner, which is developed by Blender, to generate virtually scanned data with different noise intensities and different resolutions. In addition, the circular structures in CAD models have different radii and depths, and the simulated raw data is scanned from different views to mimic more general scanning scenarios. The ground-truth circle primitives is extracted directly from CAD models since the virtually scanned data is consistent with the corresponding CAD models. Through the above schemes, the virtual point cloud data is similar to the real-scanned. Totally, $4,500$ point clouds are created for training.

Results. To demonstrate the effectiveness of AGConv, we compare iCircle-Net against several representative circle extraction methods, including EC-Net [89], PIE-NET [90], and Circle-Net [21]. In our experiments, we test all methods with a variety of virtually-scanned clouds in Tab. 14. Benefiting from AGConv, iCircle-Net possesses higher circle boundary detection precision than its competitors. Moreover, the visualization results on several real-scanned point clouds are given in Fig. 18, where our iCircle-Net can obtain less detection errors.

6.5 Point cloud registration

Point cloud registration aims to find a rigid transformation to align two point clouds.

Network framework. We utilize RGM [22] as our network's backbone in Fig. 19. In our improved RGM (iRGM), we replace the original graph convolution with the proposed AGConv module. iRGM consists of four components: a local feature extractor, an edge generator, a graph feature extractor & AIS module, and an LAP-SVD. First, we use the shared local feature extractor with AGConv to extract discriminative features for each point in $\mathbf{X}$ and $\mathbf{Y}$. Then, the edge generator produces edges and builds both the source graph and target graph, and the graphs are inputted into the graph feature extractor. The AIS module predicts the soft correspondence matrix $\widetilde{C}$ between nodes of the two graphs. Finally, the soft correspondences are converted to hard correspondences using the LAP solver, and the transformation is solved by SVD. We also update the transformation iteratively, similar to ICP.

Data. All experiments of our iRGM are conducted on ModelNet40 [24], which includes 12,311 meshed CAD models from 40 categories. Following RGM [22], we randomly sample 2,048 points from the mesh faces and re-scale the points into a unit sphere. Each category consists of official train/test splits. To select models for evaluation, we take 80% and 20% of the official train split as the training set and validation set, respectively, and the official test split for testing. For each object in the dataset, we randomly sample 1,024 points as the source point cloud $\mathbf{X}$, and then apply a random transformation on $\mathbf{X}$ to obtain the target point cloud $\mathbf{Y}$ and shuffle the point order. For the transformation applied, we randomly sample three Euler angles in the range

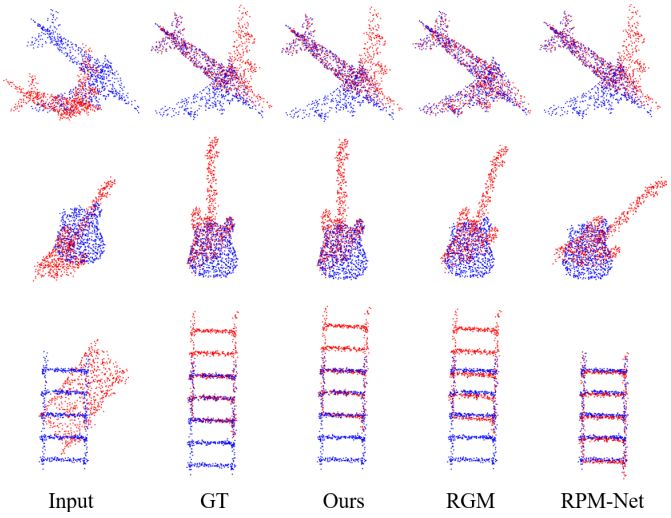


Fig. 20. Visualization of point cloud registration on ModelNet40 [24].

Method	MAE(R)	MAE(t)	MIE(R)	MIE(t)
RPM-Net [91]	0.869	0.0082	1.711	0.0175
RGM [22]	0.541	0.0046	1.032	0.0096
Ours	**0.493**	**0.0041**	**0.967**	**0.0088**

TABLE 15
Quantitative comparisons of registration methods on ModelNet40 [24].

of $[0, 45]^\circ$ for rotation and three displacements in the range of $[-0.5, 0.5]$ along each axis for translation. Moreover, we focus on the challenging partial-to-partial case, which occurs frequently in real-world applications. In order to generate partial overlapping pairs, we create a random plane passing through each point cloud independently, translate it along its normal, and retain 70% of the points as RPM-Net [91].

Results. To demonstrate the effectiveness of AGConv, we compare iRGM against two SOTAs, including RPM-Net [91] and RGM [22]. In our experiments, we train all methods in the same way and evaluate their performance over four metrics: the mean isotropic errors (MIE) of rotation and translation, and the mean absolute errors (MAE) of rotation and translation, as shown in Tab. 15. The robust features extracted by AGConv boost the performance of iRGM, which surpass the two methods over all metrics. The visualization results are given in Fig. 20, it is obvious that our iRGM aligns two point clouds more accurately.

7 Conclusion

Deep learning on 2D images has boomed because of its superiority in solving computer vision tasks. Deep learning on 3D point clouds has also drawn much interests in recent years. However, it is still far from being satisfactory to leverage the potential of deep learning for understanding point clouds. In this paper, we propose a novel adaptive graph convolution (AGConv) for point cloud analysis. The main contribution of our method lies in the designed adaptive kernel in the graph convolution, which is dynamically generated according to the point features. Instead of using a fixed kernel that captures correspondences indistinguishably between points, our AGConv can produce learned features that are more flexible to shape geometric structures. We have applied AGConv to train end-to-end deep networks for several point cloud analysis tasks, including the low-level geometry processing tasks, i.e., completion, denoising, upsampling and registration, and the high-level geometry processing tasks, i.e., classification, segmentation and circle extraction. In all these tasks, AGConv outperforms the state-of-the-arts on the benchmark datasets. Collecting-and-annotating large-scale point clouds is time-consuming and expensive. To alleviate it, we attempt to propose unsupervised learning approaches to learn features from unlabeled point cloud datasets by AGConv and geometry domain knowledge in future.

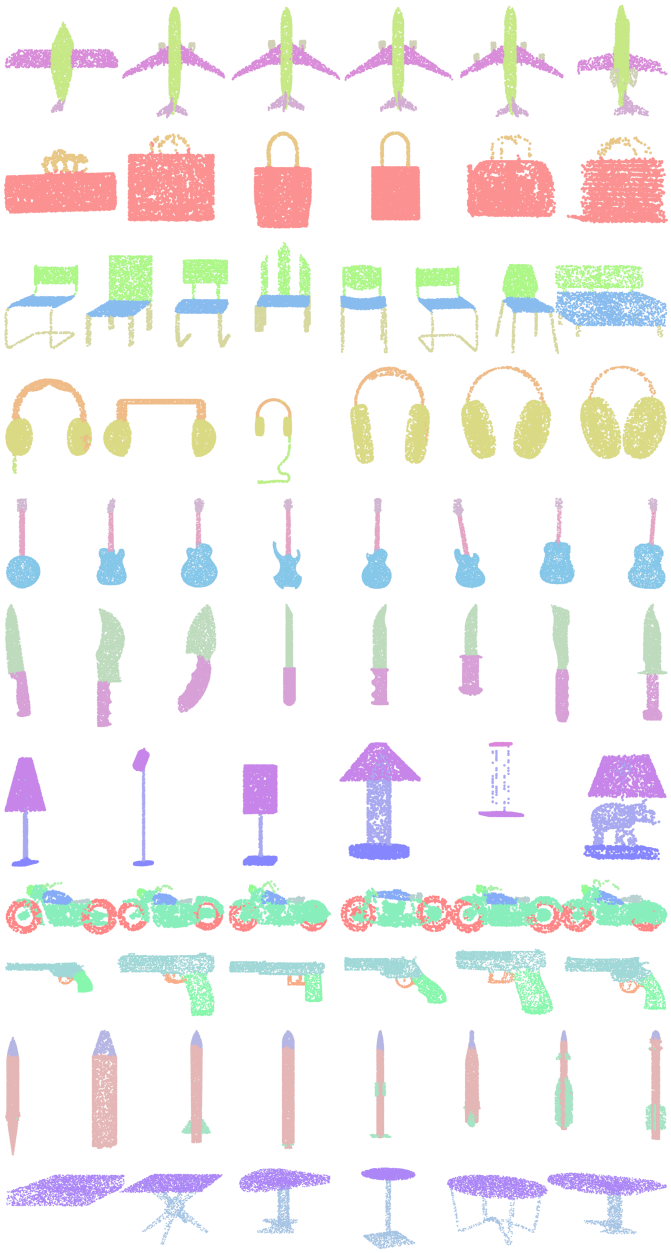


Fig. 21. More part segmentation results on ShapeNet by our AGConv-based segmentation network.

References

[1] L. Jiang, J. Zhang, and B. Deng, "Robust RGB-D face recognition using attribute-aware loss," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 10, pp. 2552–2566, 2020.

[2] W. Feng, J. Li, H. Cai, X. Luo, and J. Zhang, "Neural points: Point cloud representation with neural fields," *CoRR*, vol. abs/2112.04148, 2021.

[3] C. Yi, D. Lu, Q. Xie, S. Liu, H. Li, M. Wei, and J. Wang, "Hierarchical tunnel modeling from 3d raw lidar point cloud," *Comput. Aided Des.*, vol. 114, pp. 143–154, 2019.

[4] Q. Wu, H. Yang, M. Wei, O. Remil, B. Wang, and J. Wang, "Automatic 3d reconstruction of electrical substation scene from lidar point cloud," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 143, pp. 57–71, 2018.

[5] M. Guo, J. Cai, Z. Liu, T. Mu, R. R. Martin, and S. Hu, "PCT: point cloud transformer," *Comput. Vis. Media*, vol. 7, no. 2, pp. 187–199, 2021.

[6] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, 2021.

[7] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3577–3586.

[8] H. Zhou, H. Chen, Y. Feng, Q. Wang, J. Qin, H. Xie, F. L. Wang, M. Wei, and J. Wang, "Geometry and learning co-supported normal estimation for unstructured point cloud," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 235–13 244.

[9] Z. Li, Y. Zhang, Y. Feng, X. Xie, Q. Wang, M. Wei, and P. Heng, "Normalf-net: Normal filtering neural network for feature-preserving mesh denoising," *Comput. Aided Des.*, vol. 127, p. 102861, 2020.

[10] H. Zhou, H. Chen, Y. Zhang, M. Wei, H. Xie, J. Wang, T. Lu, J. Qin, and X.-P. Zhang, "Refine-net: Normal refinement neural network for noisy point clouds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 00, no. 00, pp. 1–18, 2022.

[11] C. R. Qi, H. Su, K. Mo, and L. 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*, 2017, pp. 77–85.

[12] 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)*, vol. 38, no. 5, pp. 1–12, 2019.

[13] Z.-H. Lin, S.-Y. Huang, and Y.-C. F. 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*, 2020, pp. 1800–1809.

[14] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, "Graph attention convolution for point cloud semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 296–10 305.

[15] K. Fujiwara and T. Hashimoto, "Neural implicit embedding for point cloud analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 734–11 743.

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

[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[18] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin, "A convolutional encoder model for neural machine translation," *arXiv preprint arXiv:1611.02344*, 2016.

[19] H. Zhou, Y. Feng, M. Fang, M. Wei, J. Qin, and T. Lu, "Adaptive graph convolution for point cloud analysis," in *ICCV*, 2021, pp. 4965–4974.

[20] L. Pan, "Ecg: Edge-aware point cloud completion with graph convolution," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4392–4398, 2020.

[21] H. Chen, Z. Wei, Q. Xie, M. Wei, and J. Wang, "Method for extracting multiple circle primitives extraction of aircraft surface based on 3d point cloud deep learning," *JOURNAL OF MECHANICAL ENGINEERING*.

[22] K. Fu, S. Liu, X. Luo, and M. Wang, "Robust point cloud registration framework based on deep graph matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8893–8902.

[23] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 922–928.

[24] 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 the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[25] T. Le and Y. Duan, "Pointgrid: A deep network for 3d shape understanding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9204–9214.

[26] 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, pp. 1–11, 2017.

[27] R. Klokov and V. 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*, 2017, pp. 863–872.

[28] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri, "3d shape segmentation with projective convolutional networks," in *proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3779–3788.

[29] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.

[30] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[31] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3d segmentation of point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2626–2635.

[32] M. Gadelha, R. Wang, and S. Maji, "Multiresolution tree networks for 3d point cloud processing," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 103–118.

[33] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on χ-transformed points," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 828–838.

[34] Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, "Spidercnn: Deep learning on point sets with parameterized convolutional filters," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 87–102.

[35] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," *arXiv preprint arXiv:1803.10091*, 2018.

[36] 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 the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6411–6420.

[37] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8895–8904.

[38] Z. Zhang, B.-S. Hua, and S.-K. Yeung, "Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1607–1616.

[39] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun, "Point transformer," *arXiv preprint arXiv:2012.09164*, 2020.

[40] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *arXiv preprint arXiv:2012.09688*, 2020.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[42] F. Wu, A. Fan, A. Baevski, Y. N. Dauphin, and M. Auli, "Pay less attention with lightweight and dynamic convolutions," *arXiv preprint arXiv:1901.10430*, 2019.

[43] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[44] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4548–4557.

[45] B.-S. Hua, M.-K. Tran, and S.-K. Yeung, "Pointwise convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 984–993.

[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*, 2020.

[47] 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 the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5115–5124.

[48] 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, pp. 2598–2606.

[49] M. Lin and A. Feragen, "diffconv: Analyzing irregular point clouds with an irregular view," *CoRR*, vol. abs/2111.14658, 2021.

[50] F. Pistilli, G. Fracastoro, D. Valsesia, and E. Magli, "Learning graph-convolutional representations for point cloud denoising," in *European Conference on Computer Vision*. Springer, 2020, pp. 103–118.

[51] R. Hanocka, A. Hertz, N. Fish, R. Giryes, and D. Cohen-Or, "Meshcnn: A network with an edge," *ACM Transactions on Graphics*, vol. 38, pp. 1–12, 2018.

[52] S. Hu, Z. Liu, M. Guo, J. Cai, J. Huang, T. Mu, and R. R. Martin, "Subdivision-based mesh convolution networks," *arXiv:2106.02285v1*, pp. 1–15, 2021.

[53] Y. Feng, Y. Feng, H. You, X. Zhao, and G. Yue, "Meshnet: Mesh neural network for 3d shape representation," p. 8279–8286, 2018.

[54] S. Gong, L. Chen, M. Bronstein, and S. Zafeiriou, "Spiralnet++: A fast and highly efficient mesh convolution operator," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2020, p. 4141–4148.

[55] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3693–3702.

[56] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9621–9630.

[57] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, "Dynamic filter networks," *Advances in neural information processing systems*, vol. 29, pp. 667–675, 2016.

[58] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *arXiv preprint arXiv:1506.02025*, 2015.

[59] 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 the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5648–5656.

[60] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 52–66.

[61] J. Li, B. M. Chen, and G. H. Lee, "So-net: Self-organizing network for point cloud analysis," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9397–9406.

[62] X. Yan, C. Zheng, Z. Li, S. Wang, and S. 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*, 2020, pp. 5589–5598.

[63] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[64] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[65] 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," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 6, pp. 1–12, 2016.

[66] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "Segcloud: Semantic segmentation of 3d point clouds," in *2017 international conference on 3D vision (3DV)*. IEEE, 2017, pp. 537–547.

[67] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, "Deep parametric continuous convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2589–2597.

[68] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "Pointweb: Enhancing local neighborhood features for point cloud processing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5565–5573.

[69] L. Jiang, H. Zhao, S. Liu, X. Shen, C.-W. Fu, and J. Jia, "Hierarchical point-edge interaction network for point cloud semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 10 433–10 441.

[70] H. Thomas, F. Goulette, J.-E. Deschaud, B. Marcotegui, and Y. LeGall, "Semantic classification of 3d point clouds with multiscale spherical neighborhoods," in *2018 International conference on 3D vision (3DV)*. IEEE, 2018, pp. 390–398.

[71] Z. Liang, M. Yang, L. Deng, C. Wang, and B. Wang, "Hierarchical depthwise graph convolutional neural network for 3d semantic segmentation of point clouds," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8152–8158.

[72] X. Roynard, J.-E. Deschaud, and F. Goulette, "Classification of point cloud for road scene understanding with multiscale voxel deep network," in *10th workshop on Planning, Perception and Navigation for Intelligent Vehicules PPNIV'2018*, 2018.

[73] H. Thomas, F. Goulette, J.-E. Deschaud, B. Marcotegui, and Y. LeGall, "Semantic classification of 3d point clouds with multiscale spherical neighborhoods," in *2018 International conference on 3D vision (3DV)*. IEEE, 2018, pp. 390–398.

[74] A. Boulch, "Convpoint: Continuous convolutions for point cloud processing," *Computers & Graphics*, vol. 88, pp. 24–34, 2020.

[75] 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 the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1534–1543.

[76] L. Landrieu and M. Simonovsky, "Large-scale point cloud semantic segmentation with superpoint graphs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4558–4567.

[77] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 728–737.

[78] C.-H. Lin, C. Kong, and S. Lucey, "Learning efficient point cloud generation for dense 3d object reconstruction," in *proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[79] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3d point capsule networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1009–1018.

[80] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese, "Topnet: Structural point cloud decoder," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 383–392.

[81] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, "Morphing and sampling network for dense point cloud completion," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 11 596–11 603.

[82] Z. Huang, Y. Yu, J. Xu, F. Ni, and X. Le, "Pf-net: Point fractal network for 3d point cloud completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7662–7670.

[83] D. Zhang, X. Lu, H. Qin, and Y. He, "Pointfilter: Point cloud filtering via encoder-decoder modeling," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 3, pp. 2015–2027, 2021.

[84] R. Li, X. Li, C. W. Fu, D. Cohen-Or, and P. A. Heng, "Pu-gan: A point cloud upsampling adversarial network," *The Chinese University of Hong Kong; Tel Aviv University; The Chinese Univsersity of Hong Kong*.

[85] S. Qiu, S. Anwar, and N. Barnes, "Pu-transformer: Point cloud upsampling transformer," *arXiv e-prints*, 2021.

[86] W. Shi, J. Caballero, F. Huszár, J. Totz, and Z. Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," *IEEE*, 2016.

[87] G. Qian, A. Abualshour, G. Li, A. Thabet, and B. Ghanem, "Pu-gcn: Point cloud upsampling using graph convolutional networks," 2019.

[88] L. Yu, X. Li, C. W. Fu, D. Cohen-Or, and P. A. Heng, "Pu-net: Point cloud upsampling network," *IEEE*, 2018.

[89] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Ec-net: an edge-aware point set consolidation network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 386–402.

[90] X. Wang, Y. Xu, K. Xu, A. Tagliasacchi, B. Zhou, A. Mahdavi-Amiri, and H. Zhang, "Pie-net: Parametric inference of point cloud edges," *arXiv preprint arXiv:2007.04883*, 2020.

[91] Z. J. Yew and G. H. Lee, "Rpm-net: Robust point matching using learned features," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 824–11 833.