Graph Attention Convolution for Point Cloud Segmentation

Anonymous CVPR submission

Paper ID 4649

Abstract

Standard convolution is inherently limited for semantic point cloud segmentation due to its isotropy about features. It neglects the structure of an object, and results in poor object delineation and small spurious regions in the segmentation result. We propose a novel graph attention convolution (GAC), whose convolution kernels can be dynamically carved into specific shapes to adapt to the structure of an object. Specifically, by assigning specific attentional weights to different neighboring points, GAC is designed to selectively focus on the most relevant part of them according to their features. The shape of the convolution kernel is then determined by the learned distribution of the attentional weights. Though simple, GAC can capture the structured features of point clouds for fine-grained segmentation and avoid feature contamination between objects. Theoretically, we provide a thorough analysis on the expressive capabilities of GAC to show how it can learn about the features of point clouds. Empirically, with the experiments on challenging indoor and outdoor datasets, our proposed GAC demonstrates state-of-the-art performance over existing deep learning methods on both datasets.

1. Introduction

Semantic segmentation of point clouds aims to assign a category label to each point, which is an important yet challenging task for 3D understanding. Recent approaches have attempted to generalize convolutional neural networks (CNNs) from grid domains (i.e., speech signals, images, and video data) to unorganized point clouds [34, 44, 33, 35, 43, 23, 26, 14]. However, due to the isotropy of their convolution kernels about the neighboring points' feature attributes, these works are inherently limited for the semantic point cloud segmentation. Intuitively, the learned features of two objects' intersecting points (i.e., point 1 in Figure 1) actually contain features of both objects rather than the object they truly belong to, which results in ambiguous label assigning.

In fact, standard convolution kernels work in a regular

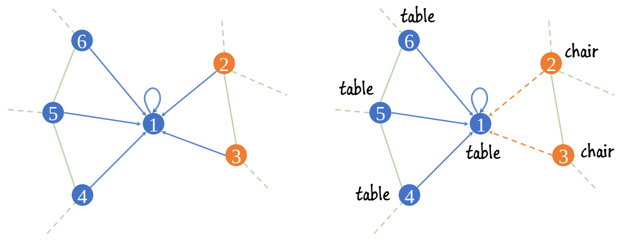

Figure 1. Illustration of the standard convolution and GAC on a subgraph of a point cloud. **Left**: The weights of standard convolution are determined by the neighbors' spatial positions, and the learned feature at point 1 characterizes all of its neighbors indistinguishably. **Right**: In GAC, the attentional weights on "chair" (the brown dotted arrows) are masked, so that the convolution kernel can focus on the points of "table".

receptive field for feature response, and the convolution weights are fixed at specific positions within the convolution window. This kind of position-determined weights results in the isotropy of the convolution kernel about the feature attributes neighboring points. For instance, in Figure 1, the learned feature at point 1 characterize its neighboring "table" and "chair" indistinguishably. This limitation of standard convolution neglects the structural connection between points belonging to the same object, and results in poor object delineation and small spurious regions in the segmentation result.

To address this problem, the key idea of this work is as follows: Based on the position-determined weights of the standard convolution, we learn to mask or weaken part of the convolution weights according to the neighbors' feature attributes, so that the actual receptive field of the convolution kernel for point clouds is no longer a regular 3D box but has its own shape to dynamically adapt to the structure of the objects.

In this paper, we realize this idea by proposing a novel GAC to selectively focus on the most relevant parts of the neighbors in the receptive field. Specifically, inspired by the idea of the attention mechanism [4, 13, 46], GAC is designed to dynamically assign specific attentional weights to different neighboring points by combining their spatial positions and features. The shape of the convolution kernel is then determined by the learned distribution of the attentional weights.

Finally, like the standard convolution in grid domain, our GAC can also be efficiently implemented on the graph representation of a point cloud. Referring to image segmentation network, we train an end-to-end graph attention convolution network (GACNet) with the proposed GAC for semantic point cloud segmentation.

Notably, postprocessing of CNNs' outputs using conditional random field (CRF) has practically become a de facto standard in semantic segmentation [44, 5, 9, 2]. However, by combining the spatial and feature constraints for attentional weights generating, GAC shares the same properties as CRF that encourages the label agreement between similar points. Thus, CRF is no longer needed in our GACNet.

Our contributions are as follows:

- We propose a novel graph attention convolution with learnable kernel shapes to dynamically adapt to the structure of the objects;
- We provide thorough theoretical and empirical analysis on the capability and effectiveness of our proposed graph attention convolution;
- We train an end-to-end graph attention convolution network for point cloud segmentation with the proposed GAC and experimentally demonstrate its effectiveness.

2. Related Works

In this section, we will discuss the related prior works within three main aspects: deep learning on point clouds, convolution on graphs, and CRF in deep learning.

Deep learning on point clouds. While deep learning has been successfully used in 2D images, there are still many challenges to exploring its feature learning power for 3D point clouds with irregular data structures. Recent researches on this issue can be mainly summarized as voxelization-based [9, 25, 48], multi-view-based [42, 24], graph-based [7, 50, 41] and set-based methods [33, 35].

The voxelization-based method [33, 24, 49, 30] aims to discretize the point cloud space into regular volumetric occupancy grids, so that the 3D convolution can be applied similarly as the image. These full-voxel-based methods inevitably lead to information loss, as well as memory and computational consumption as it increases cubically with respect to the voxel's resolution. To reduce the computational cost of these full-voxel-based methods, OctNet [37] and Kd-Net [20] were designed to resolve them by skipping the computations on empty voxels and focusing on informative voxels. The multi-view-based method [42, 24, 18] represents the point cloud as a set of images rendered from multiple views. It is still unclear how to determine the number and distribution of the views to cover the 3D objects while avoiding mutual occlusions.

The graph-based method [7, 50, 41] first represents the point cloud as a graph according to their spatial neighbors, and then generalizes the standard CNNs to adapt to the graph-structural data. Shen et al. [39] defined a point-set kernel as a set of learnable 3D points that jointly respond to the neighboring points according to their geometric affinities measured by the kernel correlation. Recently, benefiting from the development of deep learning on sets [33, 51, 36], researchers constructed effective and simple architecture to directly learn on point sets by first computing individual point features from per-point multilayer perceptron (MLP) and then aggregating all the features as a global presentation of a point cloud [35, 23, 12]. The set-based method can be used directly on the point level and is robust to the rigid transformation. However, it neglects the spatial neighbor relation between points, which contains fine-grained structural information for semantic segmentation.

Convolution on Graphs. Related works about convolution on graphs can be categorized as spectral approaches and non-spectral approaches. Spectral approaches work with a spectral representation of graphs that relies on the eigen-decomposition of their Laplacian matrix [19, 10]. The corresponding eigenvectors can be regarded as the Fourier bases in the harmonic analysis of spectral graph theory, and then, the spectral convolution can be defined as the element-wise product of two signal's Fourier transform on the graph [8]. This spectral convolution does not guarantee the spatial localization of the filter and thus requires expensive computations [40, 17]. In addition, as spectral approaches are associated with their corresponding Laplacian matrix, a spectral CNN model learned on one graph cannot be transferred to another graph that has a different Laplacian matrix.

Non-spectral approaches aim to define convolutions directly on a graph with local neighbors in a spatial or manifold domain. The key to non-spectral approaches is to define a set of sharing weights applied to the neighbors of each vertex [3, 47]. Duvenaud et al. [11] computed a weight matrix for each vertex and multiplied it to its neighbors following a sum operation. Niepert et al. [32] proposed selecting and ordering the neighbors of each vertex in heuristically so that 1D CNNs can be used. Monti et al. [31] proposed a unified framework that allows the generalization of CNNs architectures to graph using fixed local polar pseudo-coordinates around each vertex. Hamilton et al. [16] introduced an inductive framework by applying a specific aggregator over the neighbors, such as the max/mean operator or a recurrent neural network (RNN). However, their convolution weights are mainly generated according to the predefined local coordinate system, while neglecting the structure

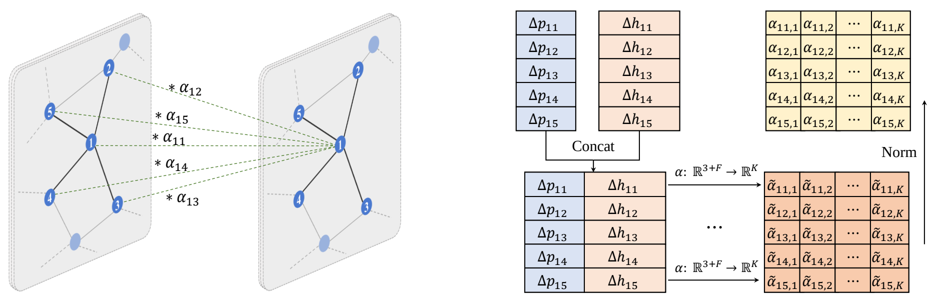
* α_{12}
* α_{15}
* α_{11}
* α_{14}
* α_{13}
Δp_{11} Δp_{12} Δp_{13} Δp_{14} Δp_{15}
Δh_{11} Δh_{12} Δh_{13} Δh_{14} Δh_{15}
Concat
$\alpha: \mathbb{R}^{3+F} \rightarrow \mathbb{R}^K$
Norm
$\alpha_{11,1}$ $\alpha_{11,2}$ ⋯ $\alpha_{11,K}$
$\tilde{\alpha}_{11,1}$ $\tilde{\alpha}_{11,2}$ ⋯ $\tilde{\alpha}_{11,K}$

Figure 2. **Left**: Illustration of GAC on a subgraph of a point cloud. The output is a weighted combination of the neighbors of point 1. **Right**: The attention mechanism employed in GAC for dynamically attentional weights generating. It receives the neighboring vertices' spatial positions and features as input, and then maps them to normalized attentional weights.

of the objects for semantic segmentation.

CRF in Deep Learning. CRF [22] possesses fine-grained probabilistic modeling capability, while CNNs have powerful feature representation capability. The combination of CRF and CNNs has been proposed in many image segmentation works [5, 9, 2, 29]. Recently, referring to the mean-field algorithm [21], the iteration of CRF inference was modeled as a stack of CNN layers [52, 28]. For 3D point cloud, following CRF-RNN [52], SegCloud [44] extends the implementation of CRF into 3D point clouds after a fully CNNs. However, as CRF is applied as an individual part following the CNNs, it is difficult to explore the power of the combination of CNNs and CRF.

3. Method

We propose a novel graph attention convolution (GAC) for structured feature learning of 3D point cloud and demonstrate its theoretical advantage (section 3.1). Afterwards, we construct an end-to-end point cloud segmentation framework (section 3.2) with our proposed GAC. The details of converting point cloud into our needed graph pyramid are provided in section 3.3.

3.1. Graph attention convolution

Consider a graph $G(V, E)$ constructed from a given point cloud $P = \{p_1, p_2, ..., p_N\} \in \mathbb{R}^3$ according to their spatial neighbors, where $V = \{1, 2, ..., N\}$ and $E \subseteq |V| \times |V|$ represent the set of vertices and edges respectively and N is the number of vertices (points). Denote $\mathcal{N}(i) = \{j : (i,j) \in E\} \cup \{i\}$ (including itself) as the neighbor set of vertex i. Let $\boldsymbol{h} = \{h_1, h_2, ..., h_N\}$ be a set of input vertex features, each feature $h_i \in \mathbb{R}^F$ is associated with a corresponding graph vertex $i \in V$, where F is the feature dimension of each vertex.

Our GAC is designed to learn a function $g : \mathbb{R}^F \rightarrow \mathbb{R}^K$, which maps the input features $\boldsymbol{h}$ to a new set of vertex features $\boldsymbol{h}' = \{h_1', h_2', ..., h_N'\}$ with $h_i' \in \mathbb{R}^K$, while maintaining the structural connection between these output features. Meanwhile, unlike the relatively fixed neighboring relation in image domain, the proposed GAC should also handle the unordered and size-varying neighbors and retain the weight sharing property.

To this end, we construct a sharing attention mechanism $\alpha : \mathbb{R}^{3+F} \rightarrow \mathbb{R}^K$ to focus on the most relevant parts of the neighbors, so that the convolution kernel of GAC can dynamically adapt to the structure of the objects. Specifically, the attentional weight of each neighboring vertex is computed as follows:

$$\tilde{\alpha}_{ij} = \alpha(\Delta p_{ij}, \Delta h_{ij}), j \in \mathcal{N}(i) \qquad (1)$$

where $\Delta p_{ij} = p_j - p_i$, and $\Delta h_{ij} = M_g(h_j) - M_g(h_i)$, where $M_g : \mathbb{R}^F \rightarrow \mathbb{R}^K$ is a feature mapping function applied on each vertex, i.e., M_g is a multilayer perceptron. $\tilde{\alpha}_{ij} = [\tilde{\alpha}_{ij,1}, \tilde{\alpha}_{ij,2}, ..., \tilde{\alpha}_{ij,K}] \in \mathbb{R}^K$ indicates the attentional weight vector corresponding to the feature vector $M_g(h_j)$. The first term of α indicates the spatial relations of the neighboring vertices, which helps to span the unordered neighbors to meaningful patches. The second term measures the feature distance between vertex pairs, which guides to assign more attention to the similar neighbors. The sharing attention mechanism can be implemented with any differentiable architecture, we use the multilayer perceptron in this work (as shown in Figure 2), which can be formulated as follows:

$$\alpha(\Delta p_{ij}, \Delta h_{ij}) = MLP([\Delta p_{ij} || \Delta h_{ij}]) \qquad (2)$$

where $||$ is the concatenation operation.

In addition, to handle the size-varying neighbors across different vertices and spatial scales, the attentional weights are normalized across all the neighbors of vertex i as follows:

$$\alpha_{ij,k} = \frac{exp(\tilde{\alpha}_{ij,k})}{\sum_{l \in \mathcal{N}(i)} exp(\tilde{\alpha}_{il,k})} \qquad (3)$$

where $\tilde{\alpha}_{ij,k}$ is the attentional weight of vertex j to vertex i at the k-th feature channel.

Therefore, the final output of our proposed GAC can be

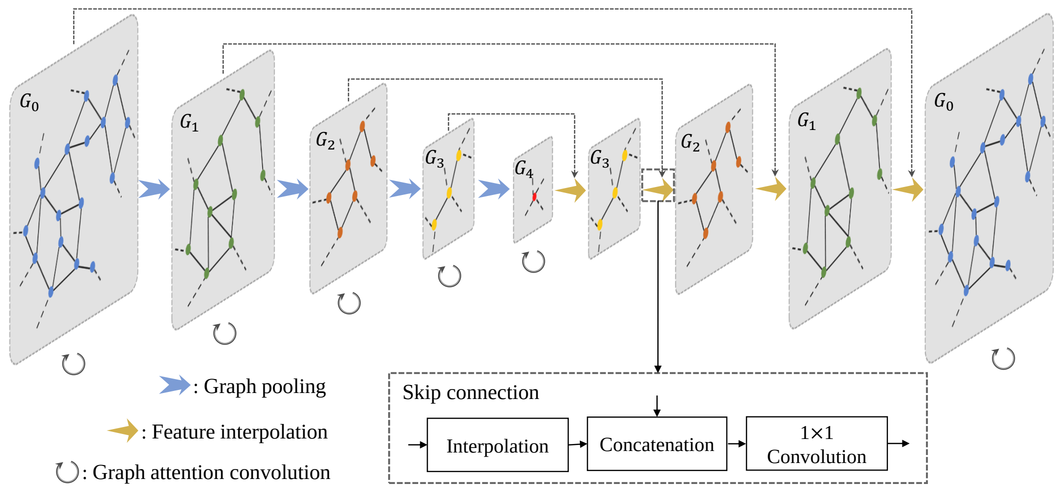

Figure 3. Flow chart of our GACNet constructed on the graph pyramid of a point cloud. GAC is applied on each scale of the graph pyramid for local feature learning, followed by the graph pooling which reduce the resolution of point clouds in each feature channel. After that, the learned features are interpolated back to the finest scale layer by layer.

formulated as follows:

$$h_i^{'} = \sum_{j \in \mathcal{N}(i)} \alpha_{ij} * M_g(h_j) + b_i \tag{4}$$

where * represents the Hadamard product, which produces the element-wise production of two vectors, and $b_i \in \mathbb{R}^K$ is a learnable bias.

Relationship to standard convolution. The convolution weights of a standard convolution in the grid domain are determined by the neighbors' local spatial positions. In our GAC, the attentional weights are generated according to not only the neighbors' spatial positions, but also their learned features. Additionally, as GAC is designed on the spatial neighbors of points, our GAC also retains the key properties of the standard convolution in grid domain: weight sharing and locality.

Relationship to prior works. Our proposed GAC is related to several prior works, mainly including GAT [46] and PointNet [33].

Although we are inspired by the idea of attention mechanism from GAT [46], our GAC is different: 1) GAC assigns specific attentional weights to not only different neighboring points but also features at different channels, as the features at different channels are hopefully independent; 2) Compared to GAT, GAC incorporates the local spatial relationship between neighboring points, which plays an important role in 3D shape analysis; 3) We generate the attentional weights based on the feature differences rather than the concatenation of two neighboring features, which is more efficient and explicit to characterize the feature relation.

PointNet [33] and its variations [35] have achieved promising results for point cloud analysis by directly learning on point sets. The key to PointNet is the use of the max operator (including an MLP). Actually, the max operator can be seen as an extreme case of GAC as "max attention", which aggregates the neighboring features by taking the max value at each feature channel. Thus, the max operator tends to capture the most "special" features, which damages the structural connections between the points of an object and becomes sensitive to noise. Comparatively, the proposed GAC aggregates the neighboring features by assigning them specific attentional weights, so that to maintain the structure of the objects for fine-grained point cloud segmentation.

Theoretical analysis. In this section, we explore the expressive capabilities of our GAC to further understand how GAC can efficiently learn the features of point clouds. Specifically, we consider whether GAC can learn to precisely represent the neighboring features of each vertex.

Suppose the input vertex features $\boldsymbol{h}$ are bounded, i.e., $\boldsymbol{h} \subseteq [a, b]^F$, where a and b indicate the lower and upper bound respectively. In fact, we can show that the proposed GAC is capable of aggregating the entire neighbor information of any vertex to an arbitrary precision:

Theorem 1. *Let $\mathcal{X} = \{S : S \subseteq [a, b]^F$ and S is finite$\}$, $f : \mathcal{X} \rightarrow \mathbb{R}$ is a continuous set function w.r.t Hausdorff distance $d_H(\cdot, \cdot)$. Denote $S_i = \{h_j : j \in \mathcal{N}(i) \in \mathcal{X}$ as the set of neighbor points of vertex $i \in V$ with arbitrary order. $\forall \epsilon > 0$, $\exists K \in \mathbb{Z}$, and parameter θ, such that for any $i \in V$,*

$$|f(S_i) - \gamma(g_\theta(S_i))| < \epsilon \tag{5}$$

where γ is a continuous function, and $g_\theta(S_i) \in \mathbb{R}^K$ is the output of our GAC.

The full proof is provided in the Appendix. Similar to PointNet, in the worst case, our GAC can learn to divide the point cloud into a volumetric representation. In PointNet, the representation capability is limited by the output dimension K. However, as the attention mechanism in our GAC actually acts as a feature encoder, GAC is capable of approximating the set function f even though K is not sufficiently large.

3.2. Graph attention convolution network

In principle, our GAC is applicable to both point cloud classification and semantic segmentation tasks, in this paper we mainly focus on the latter one. We follow the common image segmentation architecture to organize our network, coined graph attention convolution network (GACNet). The difference is that, our GACNet is implemented on the graph pyramid of point cloud, as shown in Figure 3. At each scale of the graph pyramid, GAC is applied for local feature learning. Then a graph pooling operation is followed for resolution reducing in each feature channel. After that, the learned features are interpolated back to the finest scale layer by layer. Inspired by [27], features at the same scale are skip-connected. Finally, considering the loss of feature fidelity caused by the multiple graph pooling and feature interpolation layers, an additional GAC layer is applied at the finest scale for feature refinement.

Graph pooling. Graph pooling aims to output the aggregated features on the vertices of a coarsened graph. Denote $\boldsymbol{h}_l^{'}$ as the output feature set at the l-th scale of the graph pyramid, the input feature set $\boldsymbol{h}_{l+1}$ of the $l+1$-th scale is calculated as follows:

$$h_v = pooling\{h_j^{'} : j \in \mathcal{N}_l(v)\} \tag{6}$$

where $h_v \in \boldsymbol{h}_{l+1}$ and $\mathcal{N}_l(v)$ indicates the neighbors of vertex v at the l-th scale. The $pooling$ function can be a max or mean function, which corresponds to the max and mean pooling, respectively [41].

Feature interpolation. To finally obtain the feature map that has the same number of points as the original input, we must interpolate the learned features from the coarsest scale to the original scale step by step. Let $\boldsymbol{h}_l^{'}$ be the learned feature set at the l-th scale of the graph pyramid, P_l and P_{l-1} are the spatial coordinates set of the l-th and l-1-th scales, respectively. To obtain the features of the l-1-th scale, we simply find the three nearest neighbors of P_{l-1} in P_l and calculate the weighted sum of their features. The combination weights are acquired according to the neighbors' normalized spatial distances [35].

GACNet vs. CRF. CRF has practically become a de facto standard as the postprocessing of the CNNs' outputs in semantic segmentation tasks. The key idea of CRF is to encourage similar points to share consistent labels. Intuitively, spatially close and appearance-similar points are encouraged to be assigned the same label.

Actually, our proposed GAC shares the same characteristics as the CRF model. Specifically, GAC assigns neighbors specific attentional weights according to both their spatial positions and feature attributes. The spatial positions term encourages the spatially close points to share similar features. While, the feature attributes term aims at leading the information propagating between points with similar attributes (i.e., low-level local features or high-level semantic labels). Therefore, the CRF model is no longer needed in our GACNet.

Additionally, compared to formulating the CRF model as a recurrent network [52], our GACNet has several compelling advantages. First, rather than using CRF for a post-processing which is independent of the CNNs, GACNet is equivalent to flattening the recurrent network of CRF into each layer of our network, which directly guides the learned features to maintain the structural connections for object segmentation. Second, compared to the simple message passing and compatibility transform in the class-probability space of CRF [21, 52], GAC also has the capability to map the input signal into a hidden feature space for further feature extraction. We experimentally evaluate these claims in section 4.3.

3.3. Graph pyramid construction on a point cloud

In this section, we describe how we construct the graph pyramid on point clouds according to their neighbors. Specifically, we search the spatial neighbors for all points and link them as a graph. The graph pyramid with different spatial scales is constructed by alternately applying graph construction and coarsening techniques. Notably, the covariance matrix of each point's neighbors at the finest scale are recorded during the graph construction process, and its eigenvalues are used as local geo-feature. The initial feature vector of a point is composed of height, RGB, and geo-feature.

Graph construction on a point cloud. For given point cloud P, which records the spatial coordinates of the points, we construct a directed graph $G(V, E)$. Here, each vertex is associated with a point, and the edges are added between the point and its K_G neighbors. In our experiments, the K_G neighbors are randomly sampled within radius ρ, which shows better performance than finding their K_G nearest neighbors as it is unrelated to the density of the point cloud.

Graph coarsening. Similar to pyramid construction in the image domain, we subsample the input point cloud P with a set of ratios with the furthest point sampling algorithm [35]. Denote the subsampled point clouds as $P = P_0, P_1, , P_L$, where L is the number of scales for subsampling and $P_0 = P$. For each $P_l, l = 0, , L$, a corresponding graph $G_l(V_l, E_l)$ can be constructed as described above.

Method	OA	mIoU	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
PointNet [33]	–	41.09	88.80	97.33	69.80	**0.05**	3.92	46.26	10.76	52.61	58.93	40.28	5.85	26.38	33.22
SegCloud [44]	–	48.92	90.06	96.05	69.86	0.00	18.37	38.35	23.12	75.89	70.40	58.42	40.88	12.96	41.60
SPG [23]	86.38	58.04	89.35	96.87	78.12	0.00	**42.81**	48.93	**61.58**	**84.66**	75.41	**69.84**	52.60	2.10	52.22
GACNet(ours)	**87.79**	**62.85**	**92.28**	**98.27**	**81.90**	0.00	20.35	**59.07**	40.85	78.54	**85.80**	61.70	**70.75**	**74.66**	**52.82**

Table 1. Results on the S3DIS dataset (testing on Area 5 of the S3DIS dataset and training on the rest).

Method	OA	mIoU	man-made terrain	natural terrain	high vegetation	low vegetation	buildings	hard scape	scanning artefacts	cars
SnapNet [6]	88.6	59.1	82.0	77.3	79.7	22.9	91.1	18.4	37.3	64.4
SegCloud [44]	88.1	61.3	83.9	66.0	86.0	40.5	91.1	30.9	27.5	64.3
RF_MSSF [45]	90.3	62.7	87.6	80.3	81.8	36.4	92.2	24.1	42.6	56.6
MSDeepVoxNet [38]	88.4	65.3	83.0	67.2	83.8	36.7	92.4	31.3	50.0	**78.2**
SPG [23]	**94.0**	**73.2**	**97.4**	**92.6**	87.9	44.0	93.2	31.0	**63.5**	76.2
GACNet(ours)	91.9	70.8	86.4	77.7	**88.5**	**60.6**	**94.2**	**37.3**	43.5	77.8

Table 2. Results on the Semantic3D dataset (reduce-8 challenge).

4. Experiments

In this section, we evaluate our proposed GACNet on various 3D point cloud segmentation benchmarks, including the Stanford Large-Scale 3D Indoor Spaces (S3DIS) [1] dataset and the Semantic3D [15] dataset. Three metrics were used to quantitatively evaluate the performance of the proposed method, including per-class intersection over union (IoU), mean IoU of each class (mIoU), and overall accuracy (OA). In addition, the performance of several key components of GAC is further analyzed. Our code will be made publicly available to encourage further studies.

4.1. Indoor segmentation on the S3DIS dataset

The S3DIS dataset contains 3D RGB point clouds from six indoor areas that originate from three different buildings. Each point is annotated with one of the semantic labels from 13 categories. For a principled evaluation, we follow [44, 33, 23] to choose Area 5 as our testing set and train our GACNet on the rest to ensure that the training model does not see any part of the testing area. Notably, Area 5 is not in the same building as other areas, and there exist some differences between the objects in Area 5 and other areas. This across-building experimental setup is better for measuring the model's generalizability, while also brings challenges to the segmentation task.

To prepare our training data, we first split the dataset room by room and then sample them into 1.2m by 1.2m blocks with a 0.1m buffer area on each side, points lying in the buffer area are regarded as the contextual information and are not linked to the loss function for model training or class prediction. In addition, for training convenience, the points in each block are sampled into a uniform number of 4096 points. During the testing phase, blocks can be any size depending on the memory of the computing device. In this experiment, we slice our test room into 3.6m by 3.6m blocks with a maximum of 4096$\times$9 points. Each block is individually constructed as a graph pyramid according to section 3.3 for training or testing.

The quantitative evaluations of the experimental results are provided in Table 1. We can see that our GACNet performs better than other competitive methods in most classes. In particular, we achieve considerable gains in window, table, sofa, and board. In the S3DIS dataset, the board and window are pasted onto the wall and difficult to delineate geometrically, but our proposed GACNet can still segment them out according to their color features. As the convolution weights of GAC are assigned according to not only the spatial positions but also the feature attributes of the neighboring points, the proposed GACNet is able to capture the discriminative feature of point clouds even though the spatial geometry is lost or weak.

4.2. Outdoor segmentation on the Semantic3D dataset

The Semantic3D dataset is currently the largest available LiDAR dataset, with over 4 billion points from a variety of urban and rural scenes. Each point has RGB and intensity values and is labeled with one of 8 categories: man-made terrain, natural terrain, high vegetation, low vegetation, buildings, hard scape, scanning artefacts, and cars. Different from the S3DIS dataset, the Semantic3D dataset contains outdoor scenes that have relatively larger objects. To adapt to the size of objects, the sampled blocks for the Semantic3D dataset is set to be 4m by 4m while maintaining the same maximum number of 4096 points. We provide the evaluation results on the reduced-8 challenge of the benchmark in Table 2.

Additionally, we list the overall accuracy and mean IoU of our GACNet compared to other state-of-the-art algo-

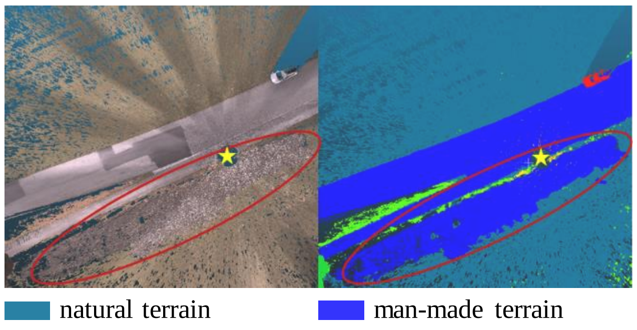

Figure 4. Illustration of the easily-confused area (in the red circle) which is close to the scanning station (yellow star) and similar to natural terrain in color and geometry but is actually man-made terrain. However, this kind of area does not appear in our training set and is difficult to segment.

rithms. In general, our performance is on par with or better than other competitive algorithms for many classes. Notably, in the semantic3D dataset, most objects, such as car, hard scape, building, and low/high-vegetation, are fragmented and incomplete due to the mutual occlusion among points. However, our GACNet can still learn to capture their discriminative features for segmentation owing to the powerful structured feature learning capability of GAC. Meanwhile, we also notice that the man-made terrain and the natural terrain are relatively difficult to split for GACNet in this experiment, as there are a large number of points in an easily-confused area (as shown in Figure 4) and that does not appear in the training set.

4.3. Ablation studies and analysis

To better understand the influence of various design choices made in our framework, we further conduct several ablation studies to demonstrate the effectiveness of GAC, explore the effect of spatial positions and feature attributes in GAC, compare GAC with CRF-RNN [52], and investigate the influence of initial features of points.

Effectiveness of GAC. To further understand the effectiveness of our proposed GAC, we compare it with the max operator (including an MLP) from PointNet [33], which has achieved promising results by directly learning on point sets. Specifically, we only replace the attention mechanism in GAC with the max operator while keeping the rest unchanged in our GACNet. The testing results on the S3DIS dataset are provided in Table 3. We can see that the mean IoU of our GAC is 4.43% higher than the max operator, which shows that our GAC has more advantages in discriminative feature learning than the max operator. Actually, the max operator in PointNet [33] acts as a ”max attention” mechanism that tends to characterize the contour of point sets in the feature space while damaging the structural connections between the points of an object. It results in the max operator being good at the object classification task but poor at segmentation where the border of the object needs to be finely delineated.

Ablation studies	OA	mIoU
Max operator	85.47	58.42
Spatial positions only	87.44	60.41
Feature attributes only	87.28	60.25
CRF-RNN (1 iteration)	87.12	61.70
CRF-RNN (3 iteration)	87.86	61.97
CRF-RNN (5 iteration)	87.46	61.83
No RGB	86.06	60.16
No geo-feature	86.17	60.37
Height only	83.56	58.96
GACNet	87.79	62.85

Table 3. Ablation studies on the S3DIS test set.

Spatial positions and feature attributes. In our GAC, the neighboring points' spatial positions and feature attributes serve as spatial and feature guides to dynamically generate their attentional weights. To explore their respective roles, we designed two other variations of GAC that only use the spatial positions and the feature attributes. Their testing results on the S3DIS dataset are reported in Table 3 for comparing convenience. The experimental results show that, both spatial positions and feature attributes have played important roles in GAC for semantic point cloud segmentation. The spatial positions span the unordered neighboring points to meaningful object surfaces, while the feature attributes further guide GAC to adapt to the structure of an object by assigning specific weights to different neighbors. Without the constraint of the spatial positions, points will only exchange information with neighbors with similar initial features, which causes the final features to be piecemeal and difficult to form meaningful objects. Without the guidance of the feature attributes, convolution kernels can hardly distinguish where the object's border is, and the current points are easily contaminated by the neighboring objects (as shown in Figure 5).

CRF-RNN. As described in section 3.2, our GACNet actually shares the same characteristics with the CRF model, which encourages feature and label agreement between similar points. To experimentally verify this claim, we remove the last GAC layer in our GACNet and replace it with the CRF-RNN [52] using different iterations. Specifically, we use the Gaussian kernels from [21] for the pairwise potentials of CRF. Their testing results on the S3DIS dataset are also provided in Table 3 for comparing convenience. We can see that, with one iteration, the CRF-RNN has basically converged, and more iterations do not result in considerably increased accuracy. Since our GACNet has shared the same characteristics of CRF in each layer of the network (section 3.2), the recurrence of CRF is no longer needed.

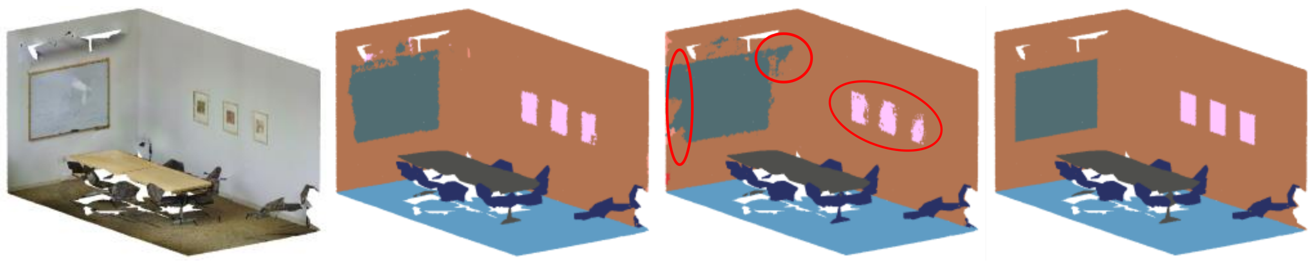

Figure 5. Illustration of the role of feature attributes in our GAC. Figures from left to right are the input point cloud, the predicted result by GACNet, the predicted result by GACNet without the feature attributes, and the ground truth. We can see that, with the guidance of the the feature attributes, the objects are more clearly delineated and more regular.

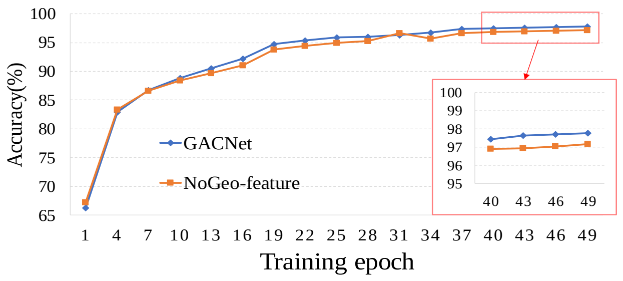

Figure 6. Training accuracy with or without geo-feature.

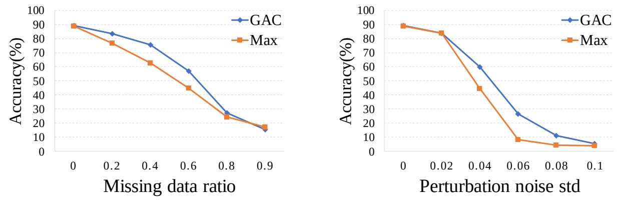

Figure 7. Robustness and stress test. GAC and Max indicate that we use graph attention convolution and the max operator in the classification network respectively.

Influence of initial features. In the above experiments on the S3DIS dataset, the initial feature vector of each point is composed of height, RGB, and geo-feature. In this section, we provide additional ablation studies to further understand the performance of our GACNet with different initial input features. We design three comparison experiments where we remove the RGB information, the geo-feature, and both of them, respectively. Their testing results on the S3DIS dataset are provided in Table 3. By comparison, their mIoU drop by 2.69%, 2.48%, and 3.89% respectively. However, compared to the relatively large accuracy difference in the testing phase, we also notice that the training accuracy without geo-feature actually shows little difference from our standard GACNet (as shown in Figure 6). The initial geo-feature serves as the low-level universal features and is designed according to priori knowledge, which is useful to improve the generalization ability of the network.

4.4. Robustness test and stress test

We compare our GAC with the max operator [33] on robustness against random Gaussian noise, and resistance on missing data. However, as additional noise will change the class attribute of the point in the segmentation task, we turn to a classification task for our robustness and stress test. We realize this work on the ModelNet40 [49] shape classification benchmark. There are 12,311 CAD models from 40 man-made object categories, splitting them into 9,843 for training and 2,468 for testing. We uniformly sample 1024 points on their mesh and normalize them into a unit sphere as inputs for our framework. Our classification framework is built by simply replacing the feature interpolation layers in GACNet with a global pooling layer, and the input of the network is just the height information of each point. All models are trained without data augmentation. During the robust test, input points are added with Gaussian noise with a series of standard deviations and zero means. For the stress test, a series of ratios of input points are randomly dropped out. From Figure 7, we can see that the max operator is more sensitive to noise because it actually tends to capture the most "special" feature (probably noise), while GAC is robust to noise due to its spatial and feature constraints. For missing data, the accuracy of GAC drops by 13.66% when the missing ratio is 40%, while the max operator drops by 26.48%.

5. Conclusion

We propose a novel graph attention convolution (GAC) with learnable kernel shapes for structured feature learning of 3D point cloud. Our GAC is a universal and simple module maintaining the weight sharing property of the standard convolution and can be efficiently implemented on graph data. We have applied our GAC to train an end-to-end network for semantic point cloud segmentation. Both theoretical analysis and empirical experiments have shown the effectiveness and advantage of our proposed GAC.

References

[1] 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 *CVPR*, pages 1534–1543, 2016. 6

[2] A. Arnab, S. Jayasumana, S. Zheng, and P. H. S. Torr. Higher order conditional random fields in deep neural networks. In *ECCV*, 2016. 2, 3

[3] J. Atwood and D. Towsley. Diffusion-convolutional neural networks. In *NIPS*, 2015. 2

[4] D. Baddanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. 1

[5] L. Bao, Y. Song, Q. Yang, and N. Ahuja. An edge-preserving filtering framework for visibility restoration. In *ICPR*, 2012. 2, 3

[6] A. Boulch, J. Guerry, B. Le Saux, and N. Audebert. Snapnet: 3d point cloud semantic labeling with 2d deep segmentation networks. *Computers and Graphics (Pergamon)*, pages 189–198, 2018. 6

[7] M. M. Bronstein, J. Bruna, Y. Lecun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017. 2

[8] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv:1312.6203*, 2013. 2

[9] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848, 2018. 2, 3

[10] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016. 2

[11] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gmez-Bombarelli, T. Hirzel, and A. Aspuru-Guzik. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015. 2

[12] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe. Exploring spatial context for 3d semantic segmentation of point clouds. In *ICCV Workshop*, pages 716–724, 2017. 2

[13] J. Gehring, M. Auli, D. Grangier, and Y. N. Dauphin. A convolutional encoder model for neural machine translation. *arXiv:1611.02344*, 2016. 1

[14] B. Graham, M. Engelcke, and L. van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 1

[15] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys. Semantic3d.net: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2017. 6

[16] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017. 2

[17] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv:1506.05163*, 2015. 2

[18] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 3d shape segmentation with projective convolutional networks. In *CVPR*, pages 6630–6639, 2017. 2

[19] T. N. Kipf and M. Welling. Semi-supervised calssification with graph-concoluational neural networks. In *ICML*, 2017. 2

[20] R. Klokov and V. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *CVPR*, pages 863–872, 2017. 2

[21] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, pages 109–117, 2012. 3, 5, 7

[22] J. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001. 3

[23] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018. 1, 2, 6

[24] T. Le, G. Bui, and Y. Duan. A multi-view recurrent neural network for 3d mesh segmentation. 2017. 2

[25] T. Le and Y. Duan. Pointgrid: A deep network for 3d shape understandings. In *CVPR*, 2018. 2

[26] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas. Fpnn: Field probing neural networks for 3d data. In *NIPS*, 2016. 1

[27] T.-Y. Lin, P. Doll, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 936–944, 2017. 5

[28] Z. Liu, X. Li, P. Luo, C. C. Loy, and X. Tang. Deep learning markov random field for semantic segmentation. *TPAMI*, 40(8):1814–1828, 2018. 3

[29] P. Luo, X. Wang, and X. Tang. Pedestrian parsing via deep decompositional network. In *ICCV*, pages 2648–2655, 2013. 3

[30] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. *Intelligent Robots and Systems (IROS)*, pages 922–928, 2015. 2

[31] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, pages 5425–5434, 2017. 2

[32] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016. 2

[33] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1, 2, 4, 6, 7, 8

[34] 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 *CVPR*, pages 5648–5656, 2016. 1

[35] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 1, 2, 4, 5

[36] S. Ravanbakhsh, J. Schneider, and B. Poczos. Deep learning with sets and point clouds. *arXiv:1611.04500*, 2016. 2

[37] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, pages 6620–6629, 2017. 2

[38] X. Roynard, J.-E. Deschaud, and F. Goulette. Classification of point cloud scenes with multiscale voxel deep network. In *3DV*, 2018. 6

[39] Y. Shen, C. Feng, Y. Yang, and D. Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*, 2018. 2

[40] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2012. 2

[41] M. Simonovsky and N. Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 2017. 2, 5

[42] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015. 2

[43] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou. Tangent convolutions for dense prediction in 3d. In *CVPR*, 2018. 1

[44] L. P. Tchapmi, C. B. Choy, I. Armeni, J. Gwak, and S. Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *3DV*, 2017. 1, 2, 3, 6

[45] H. Thomas, J.-E. Deschaud, B. Marcotegui, F. Goulette, and Y. L. Gall. Semantic classification of 3d point clouds with multiscale spherical neighborhoods. *arXiv:1808.00495*, 2018. 6

[46] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018. 1, 4

[47] N. Verma, E. Boyer, and J. Verbeek. Dynamic filters in graph convolutional networks. *arXiv:1706.05206*, 2017. 2

[48] 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*, 36(4), 2017. 2

[49] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, and X. Tang. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015. 2, 8

[50] L. Yi, H. Su, X. Guo, and L. Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *CVPR*, 2017. 2

[51] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola. Deep sets. In *NIPS*, 2017. 2

[52] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, pages 1529 – 1537, 2015. 3, 5, 7