Neurocomputing, Volume 438, May 2021, pp. 122-132
DOI: 10.1016/j.neucom.2021.01.095

GAPointNet: Graph Attention based Point Neural Network for Exploiting Local Feature of Point Cloud

Can Chen, Luca Zanotti Fragonara, Antonios Tsourdos

Centre for Autonomous and Cyberphysical Systems, Cranfield University, UK, MK43 0AL

Luca Zanotti Fragonara[1,*]

Abstract

Exploiting fine-grained semantic features on point cloud data is still challenging because of its irregular and sparse structure in a non-Euclidean space. In order to represent the local feature for each central point that is helpful towards better contextual learning, a max pooling operation is often used to highlight the most important feature in the local region. However, all other geometric local correlations between each central point and corresponding neighbourhood are ignored during the max pooling operation. To this end, the attention mechanism is promising in capturing node representation on graph-based data by attending over all the neighbouring nodes. In this paper, we propose a novel neural network for point cloud analysis, GAPointNet, which is able to learn local geometric representations by embedding graph attention mechanism within stacked Multi-Layer-Perceptron (MLP) layers. Specifically, we highlight different attention weights on the neighbourhood of each center point to efficiently exploit local features. We also combine attention features with local signature features generated by our attention pooling to fully extract local geometric structures and enhance the network robustness. The proposed GAPointNet architecture is tested on various benchmark datasets (i.e. ModelNet40, ShapeNet part, S3DIS, KITTI) and achieves state-of-the-art performance in both the shape classification and segmentation tasks.

Keywords: Point Cloud, Graph Attention, Multiple Heads Mechanism, Attention Pooling, Semantic Segmentation, Shape Classification

1. Introduction

Point cloud data are increasingly popular for a wide range of applications such as autonomous vehicles perception [1, 2, 3, 4], robotic mapping, and navigation [5, 6] and 3D shape representation and modelling [7]. Thus, many researchers are drawing attention to shape analysis and understanding, especially once Convolutional Neural Networks (CNNs) revolutionized the computer vision field [8]. However, CNNs heavily rely on data having a regular grid structure, which leads to inefficient performance on irregular and unordered geometric data, such as the one typically found in a point cloud. As a result, the full exploitation of contextual information from point clouds remains a challenging problem.

In order to leverage the advantages of CNNs, some approaches [9, 10, 11] attempted to map the unstructured point cloud to a standard 3D grid hence allowing the application of CNN architectures. However, these volumetric representations are not efficient in terms of memory and computational efficiency due to the typical sparsity of point cloud structure. *PointNet* [12] pioneered a direct application of deep learning over irregular point cloud data, bypassing the mapping to a gridded point cloud. In particular, *PointNet* renders input point cloud invariant to permutations and exploits point-wise features by independently applying a Multi-Layer-Perceptron (MLP) network and a symmetric function on each point. However, it only captures the global feature without local information. *PointNet++* [13] extends the *PointNet* model by constructing a hierarchical neural network that recursively applies *PointNet* with

*Corresponding author
Email address: l.zanottifragonara@cranfield.ac.uk (Luca Zanotti Fragonara)

March 30, 2021

designed sampling and grouping layers in order to extract the local features. *DGCNN* [14] operates an edge convolution on points and corresponding edges to further exploit the local information. Adapted from the point cloud registration method, *KC-Net* [15] builds a kernel correlation layer to measure the geometric affinities between points. However, it is noteworthy that the max pooling operation is often used to efficiently capture the most important feature from the local region, but the relationship between each point and all the corresponding neighbouring points is ignored. Motivated by this problem, we employ an attention mechanism to highlight all the local geometric correlations, rather than the most important information between each central point and corresponding neighbourhood, to better represent local features.

Attention mechanisms have proved to be efficient in many areas, especially in machine translation [16, 17], computer vision [18], and graph analysis [19]. Inspired by graph attention networks [19], we wanted to leverage the advantages of both max pooling and graph attention to fully exploit the fine-grained local features for point cloud in an attention manner for the 3D shape classification and part segmentation tasks. The key contributions of our work can be summarized as follows:

- We propose a multi-head GAPLayer, which is able to capture contextual attention features by indicating different *importance* of neighbours for each point.
- We introduce self-attention and neighbouring-attention mechanisms allowing the GAPLayer to learn the attention coefficients by considering the self-geometric information and local correlations to the corresponding neighbours.
- An attention pooling layer over the neighbours is proposed to identify the most important features to obtain the local signature representation to enhance network robustness.
- To reduce model complexity and computation, we integrate a low-dimensional GAPLayer into the *PointNet* pipeline to efficiently extract the local features from an unordered point cloud.
- We evaluate our model also on the KITTI object detection benchmark [20], which contains large scale traffic scene point cloud representations of the real world. Our model achieves the best performance and outperforms several previous state-of-the-art models.

2. Related work

Learning features from volumetric grids.. Voxelization is an intuitive way of converting sparse and irregular point clouds to a standard 3-dimensional grid structure, after which standard CNNs can be applied for feature extraction. *Voxnet* [9] voxelizes the point cloud into a volumetric grid where a single cell is called voxel, followed by a 3D CNN over the occupied voxels to predict categories of objects. However, a 3-dimensional, dense, and sparsely-occupied volumetric grid leads to large memory and high computational costs to achieve high spatial resolution. As a result, some improvements were proposed to address the sparsity problem. For instance, *Kd-Net* [21] uses a kd-tree [22] to build an efficient 3D space partition structure and a deep architecture to learn representations of point cloud. Similarly, *OctNet* [11] applies 3D convolution on a hybrid grid-octree structure generated from a set of shallow octrees to achieve high resolution.

Learning features from multi-view models.. In order to apply the standard CNN operation, whilst avoiding large computational costs in volumetric-based methods, some researchers are interested in multi-view based approaches. For instance, Qi et al. [23] and Wang et al. [24] attempted to learn features of the point cloud in an indirect way by applying a typical 2D CNN architecture to multiple 2D image views that are generated by the multi-view projections over a 3D point cloud. However, these multi-view approaches are not capable of carrying out the semantic segmentation task for point cloud data, because 2D images lack the depth information, which leads to the fact that it is non-trivial to classify each point from images only.

Geometrical deep learning.. Geometrical deep learning is a modern terminology that stands for the application of deep learning methods to non-Euclidean structured data (e.g. 3D point cloud, social networks, or genetic networks) [25]. As aforementioned, the issue arising with non-Euclidean data is that the convolution operation cannot be applied directly. On the other hand, the direct application of deep learning to graphs-like data structures discloses the opportunity of extracting features and information from nodes and edges, which are naturally carrying

individual and local information. This is valid for any type of non-Euclidean data, hence also to point clouds. Geometrical deep learning can be broadly classified into two categories: GNNs (graph neural networks) either applied in the spatial domain or in the spectral domain.

Spatial domain GNNs apply MLPs on individual and neighbouring points to extract local features. As previously mentioned, *PointNet* [12] introduced the direct application of deep learning on raw point cloud data. In more detail, a Multi-Layer-Perceptron (MLP) network and a symmetric function (e.g. max pooling) are applied on every individual point to extract global features. This approach provides an efficient way for unstructured point cloud understanding, but local features are not captured. In fact, the architecture only works on independent points without considering any relationship measurements between points in the local regions. To address this problem, PointNet++ [13] constructs a hierarchical neural network that recursively applies *PointNet* with a sampling layer and a grouping layer to exploit local representations. *DGCNN* [14] extends *PointNet* by having an edge convolution operation (EdgeConv) that is applied on edge features which aggregate each point and corresponding edges connecting to the neighbouring pairs. In order to leverage the advantages of standard CNN operation, *PointCNN* [26] attempts to learn a χ-convolutional operator to transform a given unordered point set to a latent canonical order, after which a typical CNN architecture is used to extract local features. *RS-CNN* [27] learns a CNN-like filter from relational neighbourhoods, which can be applied over neighbouring points to capture fine-grained local representations. In order to address the irregularity of point cloud data, *A-CNN* [28] processes the point cloud to ring-shaped structure and introduces an annular convolution to exploit the local geometric information. In this context, our model also introduces an attention mechanism and builds an efficient framework to capture the local features.

Spectral domain GNNs rely on the generalization of the Fourier transform operation to graph-like data. They typically define convolutions as spectral filtering applied on the eigenvectors of the graph Laplacian matrix [29] on graphs. Graph CNNs [29, 30, 31] show advantages of graph representation in many tasks for non-Euclidean data, as they can naturally deal with these irregular structures. *PointGCN* [32] builds a graph CNN architecture to capture the local structure and classify point cloud, which also proves the good potential for unordered point cloud analysis. On the other hand, this approach seems to be more susceptible to domain changes as eigenvectors tend to be inconsistent across domain [33].

3. GAPointNet architecture

In this section, we present and discuss the main features of our GAPointNet model aimed at learning local representations for unstructured point cloud data for the shape classification and part segmentation tasks. In the following sections, we discuss in detail the research gaps and how our model addresses those.

3.1. Problem statement

We start by defining a raw set of a point cloud, which is also the input of our model, as Eq. (1):

$$\mathbf{X} = \left\{\mathbf{x_i} \in \mathbb{R}^F, i = 1, 2, \ldots, N\right\} \tag{1}$$

where $\mathbf{x_i}$ is the feature vector of the i-th point in the point cloud with F-dimension. Typical features used are the 3D space coordinates (x_i, y_i, z_i), the colour, the intensity, the surface normal, etc. For the sake of simplicity, in this study we set $F = 3$, and we only use 3D coordinates as the input features. Finally, the total number of points in the point cloud is defined N.

As previously mentioned, the method is proposed for object category classification and semantic segmentation of point cloud data. For the object classification task, our aim is to find a non-linear mapping function f_c that is capable of estimating the probability distribution of all the categories $\mathbf{C}$ for the given point cloud $\mathbf{X}$:

$$\mathbf{C} = f_c(\mathbf{X}) \tag{2}$$

Similarly, for the point cloud segmentation task, we aim at inferring another non-linear function f_s, which is able to estimate the probability distribution of all the categories $\mathbf{S_i}$ for each point $\mathbf{x_i}$:

$$\mathbf{S_i} = f_s(\mathbf{x_i}), i = 1, 2, \ldots, N \tag{3}$$

However, it is challenging to design the non-linear functions f_c and f_s, for classification and segmentation respectively, due to the fact that: firstly, the point cloud is a set of irregular and unstructured points, so the functions should be permutation invariant; secondly, the points of a point cloud are

likely to be relatively rotated and translated in a real application. As a result, the designed functions should be invariant for certain transformation, such as rotation and translation; thirdly, local features should be considered to better represent relationship between each point and the corresponding local region.

Inspired by *PointNet* [12], we use a symmetric function (e.g. max pooling, weighted average pooling) to address the permutation invariance problem and leverage the high non-linear approximation ability of Multi-Layer-Perceptron (MLP) to extract useful features. Consequently, our model achieves the following approximation:

$$f(\{\mathbf{x_1}, \ldots, \mathbf{x_N}\}) \approx g(h(\mathbf{x_1}), \ldots, h(\mathbf{x_N})) \quad (4)$$

where g is a symmetric function and h indicates the MLP function. The symbol f can represent either the classification or the segmentation task.

3.2. Local structure representation

In order to efficiently represent the point cloud structure, we propose to convert it to a directed acyclic graph structure with nodes and edges. It is worth highlighting that the directed graph structure ignores the order of the nodes when we apply the neural network on it, and each node is propagated. The edges of the graph are a flexible way to represent different types of information about the connectivity (e.g. distance, concatenation) existing between a pair of nodes [34]. As a result, we convert each point to a node in the graph and define the edge as the dependency of information between two points. Generally, a directed graph with nodes $\mathbf{V}$ and edges $\mathbf{E}$ is constructed as Eq. (5):

$$\mathbf{G} = (\mathbf{V}, \mathbf{E}) \quad (5)$$

where $\mathbf{V} \subset \mathbb{R}^F$ are nodes for points with F dimension, $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ are edges connecting neighbouring pairs of points.

However, considering the fact that the number of samples in point cloud can be very large in real applications (e.g. autonomous vehicles, robotics), allowing every point to attend to all other points will lead to a high computational cost and the gradient vanishing problem due to very small weights being allocated on every other point for every point, so it is not a practical solution to involve all other points for each central point being considered. Besides, a graph structure with edges and nodes is a natural choice to represent a point cloud structure. As a result, we construct a directed k-nearest neighbour graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, as shown in part (b) of Fig. 1, to represent the local structure of the point cloud, then $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V_{nei}}$ where $\mathbf{V_{nei}}$ is the neighbourhood set of point $\mathbf{x_i}$. We group the neighbouring points by computing the feature space Euclidean distance between each central point and all other points, and clustering k-nearest points as neighbourhood. The edge features $\mathbf{e_i}$ of a raw point set $\mathbf{X}$ are defined as:

$$\begin{aligned}
&\mathbf{X} = \left\{\mathbf{x_i} \in \mathbb{R}^F, i = 1, 2, \ldots, N\right\} \\
&\mathbf{V} = \mathbf{X} \\
&\mathbf{y_{ij}} = \mathbf{x_i} - \mathbf{x_{ij}} \\
&\mathbf{e_i} = (\mathbf{y_{i1}}, \mathbf{y_{i2}}, \ldots, \mathbf{y_{iK}}), i = 1, 2, \ldots, N \\
&\mathbf{E} = \{\mathbf{e_i} | i = 1, 2, \ldots, N\}
\end{aligned} \quad (6)$$

where $\mathbf{x_{ij}}$ indicates a certain point $\mathbf{x_j}$ belonging to the i-th central point $\mathbf{x_i}$ neighbourhood, whilst $\mathbf{y_{ij}}$ is a directed and relative distance edge between the central point $\mathbf{x_i}$ and its neighbour $\mathbf{x_{ij}}$. The total number of elements in the neighbourhood for a certain central point is defined as K.

3.3. Attention-aware local feature extraction

In order to capture local features, we update the general function Eq. 4 to:

$$f(\{\mathbf{x_1}, \ldots, \mathbf{x_N}\}) \approx g(h(\mathbf{e_1}), \ldots, h(\mathbf{e_i})) \quad (7)$$

where g is a symmetric function and h indicates the MLP function that is applied on local regions. The function f can represent either the classification or the segmentation function.

3.3.1. GAPLayer

In this section we introduce our GAPLayer module, aimed at efficiently capture local features of the point cloud. To the benefit of the readers, we start by introducing a single-head GAPLayer that takes point cloud data as the input, jointly with a multi-head mechanism that concatenates all heads together over the feature channels in our network.

In order to pay different attention to different neighbours, we propose a self-attention mechanism and a neighbouring-attention mechanism to capture the attention coefficients for each point to its neighbourhood as illustrated in part (c) of Fig. 1. In more detail, the self-attention mechanism learns the self-coefficients by considering the self-geometric information for each individual point (the central point

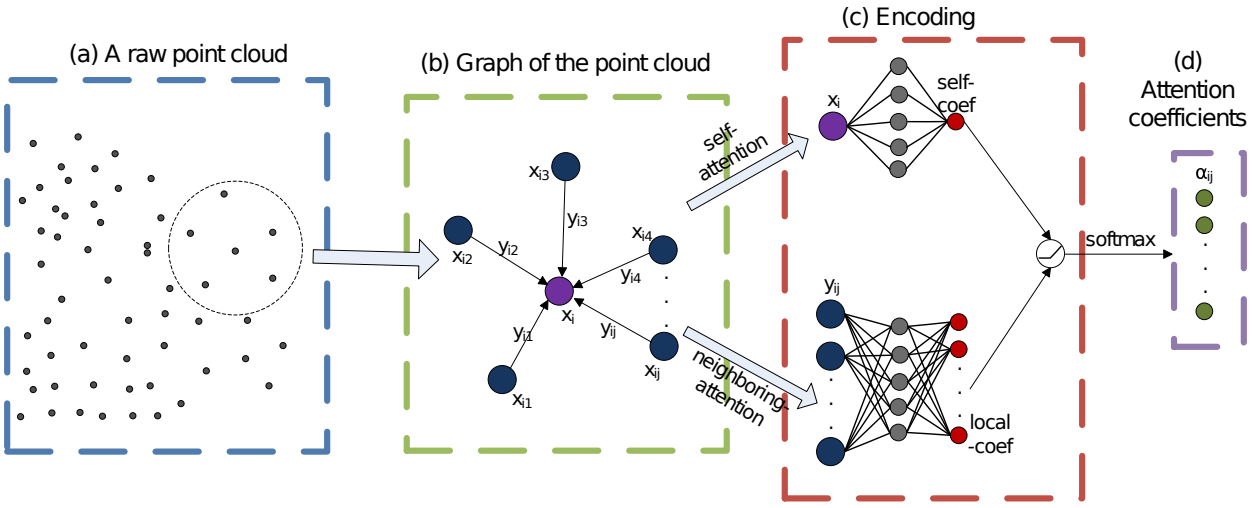


Figure 1: **Local attention coefficients mechanism.** A raw point cloud is represented in (a), the construction of a locally directed acyclic graph is represented in (b). x_i and x_{ij} denote the i-th point and its corresponding neighbours, respectively, whilst y_{ij} are the corresponding edges. Self-coefficients (self-coef for short) and local-coefficients (local-coef for short) are encoded by a Multi-Layer-Perceptron (MLP) layer and fused by a leaky RELU activation function in (c), and normalized by a softmax function to generate the attention coefficients for neighbouring pairs in (d).

of each neighbourhood), whilst the neighbouring-attention mechanism focuses on local-coefficients by considering the neighbourhood.

As an initial step, we encode the nodes and edges of the point cloud with respect to the high-level features with output dimension F' as defined by Eq. (8) and Eq. (9).

$$\mathbf{x'_i} = h(\mathbf{x_i}, \theta) \tag{8}$$

$$\mathbf{y'_{ij}} = h(\mathbf{y_{ij}}, \theta) \tag{9}$$

where $h(.)$ is any parametric non-linear function, assumed to be a single-layer MLP neural network in our experiment, and θ is the set of learnable parameters of the filter.

We obtain the attention coefficients by fusing the self-coef $h(\mathbf{x'_i}, \theta)$ and the local-coef $h(\mathbf{y'_{ij}}, \theta)$ as defined by Eq. (10). In particular, due to the fact that the self-coef and the local-coef are features with just 1 dimension, we simply fuse them by a sum operation, and then use the non-linear activation function leaky RELU [35]:

$$c_{ij} = LeakyReLU(h(\mathbf{x'_i}, \theta) + h(\mathbf{y'_{ij}}, \theta)) \tag{10}$$

where $h(\mathbf{x'_i}, \theta)$ and $h(\mathbf{y'_{ij}}, \theta)$ are single-layer neural network with a 1-dimensional output and $LeakyReLU(.)$ denotes the non-linear activation function leaky RELU.

In order to align the comparison of the attention coefficients across neighbours for different points, we use the *softmax* function to normalize the coefficients of all the neighbours to every point as follows:

$$\alpha_{ij} = \frac{\exp(c_{ij})}{\sum_{k \in \{1,2,\ldots,K\}} \exp(c_{ik})} \tag{11}$$

where K is the total number of points in the neighbourhood.

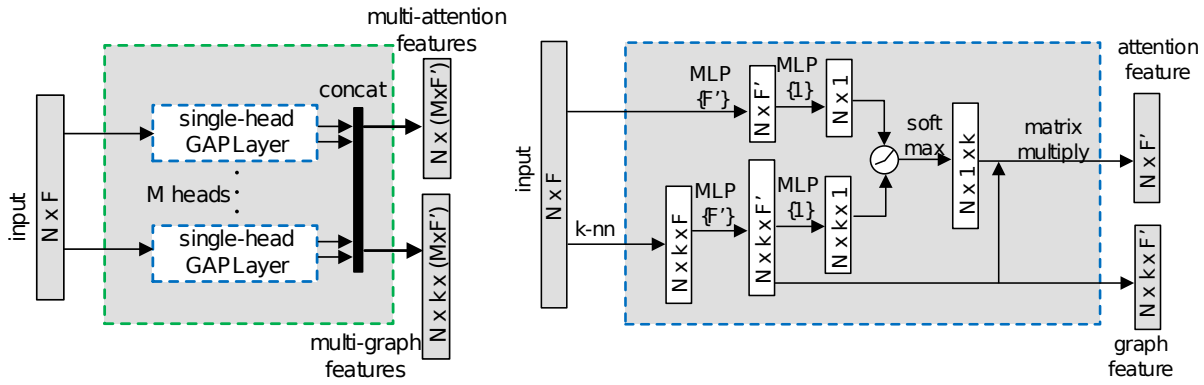
The goal of each single-head GAPLayer is to compute the contextual attention feature for every point. For this, we utilize the obtained normalized coefficients to compute a linear combination as shown in Eq. (12), and we successively apply to it a non-linear activation function to achieve a better extraction ability:

$$\mathbf{\hat{x}_i} = f(\sum_{j \in \{1,2,\ldots,K\}} \alpha_{ij} \mathbf{y'_{ij}}) \tag{12}$$

where K is the total number of the neighbouring points, and $f(.)$ is a non-linear activation function, chosen to be a RELU [35] in our experiment. As shown in Fig. 2(b), the outputs of the single-head GAPLayer are both the attention feature $\mathbf{\hat{x}_i} \in \mathbb{R}^{F'}$ and the graph feature $\mathbf{y'_{ij}}$ encoded from the graph edges.

3.3.2. Multi-head mechanism

In order to obtain sufficient structural information and to stabilize the network, we concatenate



(a) GAPLayer (b) Single-head GAPLayer

Figure 2: **GAPLayer structure.** The GAPLayer with M heads, as represented in 2(a), takes N points with F dimensions as input and concatenates the attention and graph features from all heads to generate multi-attention and multi-graph features as output. As shown in 2(b), the single-head GAPLayer learns self-attention and neighbouring-attention features in parallel that are then fused together by a non-linear activation function (i.e. leaky ReLU) to obtain the attention coefficients, which are further normalized by a *softmax* function. MLP{} denotes a multi-layer perceptron operation and the number within braces is the size of a set of filters.

M independent single-head GAPLayers to generate multi-attention features with $M \times F'$ channels (see Eq. (13)). Fig. 2(a) shows how the outputs of the multi-head GAPLayer (GAPLayer for short), i.e. the multi-attention features and multi-graph features $\hat{\mathbf{x}}'_{\mathbf{i}}$, are concatenated attention feature and graph feature from the corresponding head. The concatenation is defined as follows:

$$\hat{\mathbf{x}}'_{\mathbf{i}} = \Big\Vert_{m}^{M} \hat{\mathbf{x}}_{\mathbf{i}}^{(m)} \tag{13}$$

such that $\hat{\mathbf{x}}_{\mathbf{i}}^{(m)}$ is the attention feature of the m-th head, M is the total number of heads, and $\|$ is hereinafter used as the concatenation operation over all feature channels.

3.3.3. Attention pooling layer

To enhance the network robustness and improve the performance, we define an attention pooling layer on the neighbouring channel of multi-graph features. We use max pooling as our attention pooling operation, which identifies the most important feature across all heads to capture the local signature representation $\mathbf{Y_i}$ (see Eq. (14)).

$$\mathbf{Y_i} = \Big\Vert_{m}^{M} \max_{j \in \{1,2,\ldots,K\}} \mathbf{y'_{ij}}^{(m)} \tag{14}$$

where $\mathbf{y'_{ij}}$ are the edge features, M and K are the total number of heads and the neighbouring points of the central point, respectively. The local signature is connected to the intermediate layer for capturing global features. We know that the max pooling operation is a symmetric function, hence is invariant to the ordering of the points in the point cloud.

3.3.4. Feature aggregation

The attention-aware feature $\hat{\mathbf{x}}'_{\mathbf{i}}$, extracted as in Eq. (13) from the GAPLayer, and the corresponding local signature feature $\mathbf{Y_i}$ (Eq. (14)) originated from our attention pooling layer are aggregated by relying on a concatenation operation as shown in Eq. (15) to generate a contextual local feature. Finally, the classification function $\mathbf{C}$ in Eq. (2) and the segmentation function $\mathbf{S_i}$ in Eq. (3) can be easily applied on $\mathbf{l_i}$ in the further fully-connected layers neural network.

$$\mathbf{l_i} = [h(\hat{\mathbf{x}}'_{\mathbf{i}}), \mathbf{Y_i}] \tag{15}$$

where $h(.)$ indicates a MLP neural network.

3.3.5. Spatial transform network

Inspired by the *PointNet* [12] architecture, we also design a mini attention-aware spatial transformation network applied to the 3D coordinates of the input points to address the transformation invariance problem. To this aim, we use a single head GAPLayer module to learn an affine transformation matrix for the prediction of geometric transformations (e.g. rotations and translations). The detailed architecture of the attention-aware spatial transform network is shown in Fig. 3.

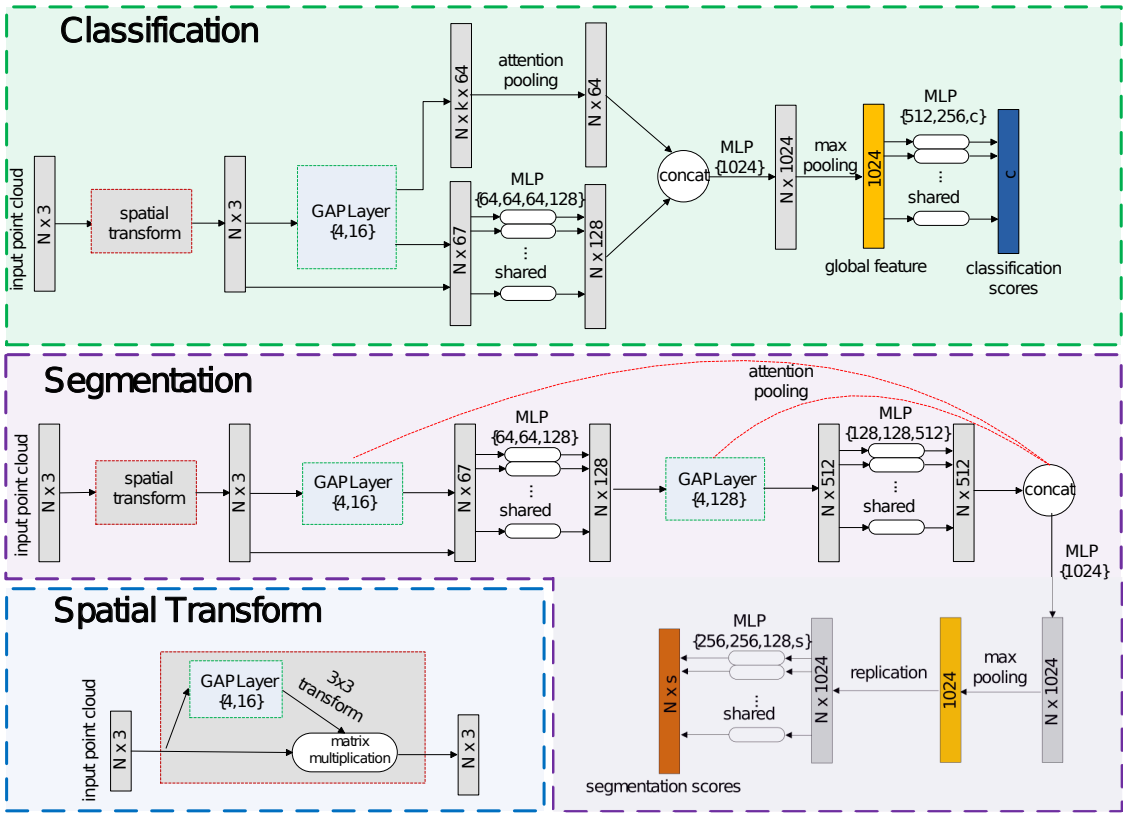


Figure 3: **GAPointNet architecture:** The architecture contains two parts: classification (top branch) and semantic part segmentation (bottom branch). The classification model takes N points as input and applies one GAPLayer to obtain the multi-attention features and the multi-graph features followed by shared MLP layers and an attention pooling layer respectively. Finally, a shared fully-connected layer is used to form a global feature that is used to obtain the classification scores for c categories. The semantic segmentation model (bottom branch) extends the classification model by a second GAPLayer with MLP layers to obtain a certain part category for each point from s semantic labels. The two red arrows represent the attention pooling operation from the corresponding GAPLayers used to generate a local signature that is concatenated to the intermediate layer for the global feature generation. Besides, GAPLayer{4,16} denotes a GAPLayer with 4 heads and 16 channels of encoding feature. **Spatial Transform:** The spatial transform network is used to make point cloud invariant to certain transformations. The model learns a 3×3 matrix for affine transformation from a single-head GAPLayer with 16 channels.

3.4. GAPointNet architecture

Our GAPointNet model shown in Fig. 3 considers both shape classification and semantic part segmentation for point cloud. The backbone architecture is similar to *PointNet* [12]. However, there are three main differences between the two architectures:

1. an attention-aware spatial transform network to make the point cloud invariant to certain transformations is used;
2. local features are exploited by a GAPLayer before the stacked MLP layers instead of only processing individual points as in *PointNet*;
3. an attention pooling layer is used to obtain the local signature that is connected to the intermediate layer for capturing a global descriptor.

Classification structure. The classification model is presented in Fig. 3 top branch. As previously mentioned, in order to make the input points invariant to some geometric transformations, such as scale, rotation or translation, we firstly apply the attention-aware spatial transformer network to align the point cloud to a canonical space. The spatial transformer employs a single-head GAPLayer with 16 channels to capture attention features, followed by three shared MLP layers (64, 128, 1024) to output neurons with sizes 64, 128, 1024 respectively, then a max pooling operation and two full-connected layers (512, 256) are used to finally generate a transformation matrix.

A multi-head GAPLayer is then applied to generate multi-attention features with $M \times F'$ channels,

where the number of heads is set as $M = 4$, and the number of encoding channels is set as $F' = 16$. Our multi-attention features aggregate the coordinate features to obtain a contextual attention feature with the number of channels $3 + M \times F'$, which is then used to extract fine-grained features by four shared MLP layers (64, 64, 64, 128). The skip-connection method is employed to connect the local signature and these intermediate layers, followed by a shared fully-connected layer (1024) and a max pooling operation over the feature channels to obtain a global feature for the entire point cloud. We finally apply three shared MLP layers (512, 256, 40) and the dropout operation with a keep probability of 0.5 to map the global feature to 40 categories. Besides, the activation function ReLU with batch normalization is used in each layer, and the number of neighbours k is set to 20.

Segmentation structure. Our segmentation model, as shown in Fig. 3 (bottom branch), aims to predict a part category label for each point in the point cloud. The same spatial transformer network and GAPLayer as described in Section 4.1 are applied, followed by shared MLP layers (64, 64, 128). Then the second GAPLayer with 4 heads and 128 encoding channels is applied, followed by shared MLP layers (128, 128, 512) to obtain representations with 512 channels, which are concatenated with the local signature generated from corresponding attention pooling layer of GAPLayer. The aggregated feature is input to a shared full-connected layer (1024) and a max pooling operation is carried out to obtain a global feature. This is then duplicated 2048 times and finally applied to four shared full-connected layers (256,256,128,50) with a dropout probability 0.6 to transform the global feature to 50 part categories.

3.5. Comparison to existing models

We further theoretically compare our model to other state-of-the-art methods. As we know that a local feature is helpful to fully represent the geometrical information for the local region. As a result, the performance heavily depends on how to efficiently extract the local feature. We define the local feature function with respect to the central point $\mathbf{x_i}$ as $h(\mathbf{x_i}, \mathbf{x_{ij}}, \theta)$, where $h(.)$ is a non-linear function, $\mathbf{x_i}$, $\mathbf{x_{ij}}$ are a certain central point and corresponding neighbouring point respectively. θ denotes a set of learnable parameters in the non-linear function.

Then we discuss that *PointNet* [12] has no local feature extraction, then the local feature function is $h(\mathbf{x_i}, \mathbf{x_{ij}}, \theta) = h(\mathbf{x_i}, \theta)$. *PointNet++* [13] considers the local feature by concatenating each central point $\mathbf{x_i}$ and corresponding neighbouring point $\mathbf{x_{ij}}$. As a result, the local feature function in *PointNet++* is $h(\mathbf{x_i}, \mathbf{x_{ij}}, \theta) = h(\mathbf{x_i}, \mathbf{x_{ij}}, \theta)$. The results also show that the performance is improved significantly when the local feature is involved. *DGCNN* [14] introduces an edge feature and the local feature function is defined as $h(\mathbf{x_i}, \mathbf{x_{ij}}, \theta) = h(\mathbf{x_i}, \mathbf{x_i} - \mathbf{x_{ij}}, \theta)$. Compare to these methods, the local feature function used by our model is $h(\mathbf{x_i}, \mathbf{x_{ij}}, \theta) = h(\theta_\mathbf{m} \cdot \mathbf{x_i}, \theta_\mathbf{n} \cdot (\mathbf{x_i} - \mathbf{x_{ij}}), \theta_\mathbf{p} \cdot (\mathbf{x_i} - \mathbf{x_{ij}}))$, where $\theta_\mathbf{m}$ and $\theta_\mathbf{n}$ are learnable attention-aware parameters for our self-attention and local-attention modules respectively. $\theta_\mathbf{p}$ indicates a set of learnable edge-aware parameters. It shows that our model aggregates the self-attention feature, the local-attention feature and the edge feature, which can better exploit the local feature.

4. Experiments

In this section, we evaluate our GAPointNet model for both classification and segmentation of 3D point cloud data, and then we compare its performance with respect to several state-of-the-art methods. Furthermore, we perform an ablation study to investigate different design variations and hyper-parameter settings.

4.1. Classification

Dataset. The effectiveness of our classification model is firstly demonstrated on the ModelNet40 benchmark [36] for shape classification. The ModelNet40 dataset contains 12,311 meshed CAD models that are classified to 40 man-made categories. We separated the dataset into 9,843 models for training and 2,468 models for testing. Then we normalize the models in the unit sphere and uniformly sample 1,024 points over the model surface. Furthermore, we augmented the training dataset by randomly rotating, scaling the point cloud and jittering the location of every point by means of Gaussian noise with zero mean and 0.01 standard deviation for all models.

Training details. We used Adam [37] as an optimization algorithm, with momentum set to 0.9. The batch size was set to 32 and the learning rate started from 0.005 and was then divided by 2 every 20 epochs down to 0.00001. The decay rate for batch

normalization was initially set to 0.7 and gradually increased to 0.99. Our model has been trained on an NVIDIA GTX1080Ti GPU and TensorFlow v1.6.

Results. It is possible to compare our results and complexity with several state-of-the-art works (see Table 1), and our model achieves competitive performance on the ModelNet40 benchmark. The metrics used for comparison of the accuracy performance are: mean per-class accuracy (mA %) and overall accuracy (OA %). Further to this, we estimated the model complexity by measuring: the number of parameters (Million), the floating point operations (FLOPs) (Billion), and the forward propagation time. We also evaluated and listed in Table 1 the same metrics for all the available models in the same experimental environment. It shows that our model achieves a very competitive trade-off between accuracy and complexity.

It is also worth to mention that our model significantly improves *PointNet* [12] and *PointNet++* [13] by 3.8% and 2.3% accuracy, respectively. With respect to GAPointNet, *PointNet* [12] yields a graph without any local information, and *PointNet++* [13] downsamples the number of points and then applies Multi-Layer-Perceptron (MLP) on features of the neighbouring points for each central point but without using the edge features. Our model has been compared also with *DGCNN* [14] and outperforms it by 0.1% accuracy with approximately half of its computational cost. It is easy to observe that *DGCNN* [14] uses the sample max pooling operation as the local region aggregation method, which convincingly verifies that our attention-aware aggregation operation performs better than the max pooling in terms of accuracy and efficiency.

Discussion. The good performance of our model arises from a contribution of different factors. Firstly, the GAPLayer is efficient in highlighting the different importance of neighbouring points. Specifically, we use a better attention-aware local feature function (see Section 3.5) to represent a high-level relation expression for the local region of the point clouds. Besides, compared to the max pooling operation, the aggregation of features from neighbouring points with their learnable attention coefficients better exploits the local geometric correlations between each central point and its corresponding neighbourhood. In fact, the max pooling operation only extracts the most relevant feature from the neighbourhood. On the other hand, an attention-weighted sum of neighbourhood takes all the local region features into account. Secondly, the multi-head mechanism is beneficial to capturing useful information with greater width, as each head is likely to contribute to learning different kinds of features, such as geometric shape information or intrinsic features. Thirdly, our attention pooling layer captures local signatures from the neighbourhood, which is also the important feature extracted from all the neighbouring points for each central point. Then the attention-aware feature and the important feature are aggregated to efficiently boost the performance. As a result, our deep-wide and attention-aware model is efficient in extracting fine-grained local representations for point cloud data.

On the other hand, our model just adopts one multi-head GAPLayer module to *PointNet* [12] pipeline but remarkably improves the accuracy and reduces the model complexity, which shows the effectiveness of our attention structure and also the great feasibility for real-time applications.

Ablation study. We also tested our classification model with different hyper-parameters settings on the ModelNet40 benchmark [36]. As can be seen in Table 2, we carried out a comprehensive ablation study by involving four factors: points number, spatial transformer, number of neighbours, and grouping method and their influence on the mean accuracy. The model complexity was also evaluated resorting to the number of parameters, FLOPs, and forward time.

Specifically, our attention-aware spatial trans-

Table 1: Classification results on the ModelNet40 dataset. mA, OA and FLOPs are mean per-class accuracy, overall accuracy and floating point operations, respectively.

	mA (%)	OA (%)	params	FLOPs	time
VoxNet [9]	83.0	85.9	-	-	-
PointNet [12]	86.0	89.2	3.48M	**957M**	14.7ms
PointNet++ [13]	-	90.7	1.99M	3136M	32.0ms
KC-Net [15]	-	91.0	-	-	-
SpecGCN [38]	-	91.5	2.05M	1112M	11254ms
KD-Net [21]	-	91.8	-	-	-
PCCN [39]	-	92.3	8.1M	-	80ms
PointCNN [26]	88.8	92.5	**0.6M**	1681M	**12.0ms**
A-CNN [28]	**90.3**	92.6	-	-	-
DGCNN [14]	90.2	92.9	1.84M	2768M	52.0ms
OURS	**90.3**	93.0	1.91M	1228M	26.0ms
RS-CNN [27]	-	**93.6**	-	-	-

Table 2: Ablation study for more hyper-parameters.

model	points	spatial transformer	neighbours	grouping	Params (Million)	FLOPs (Billion)	Time (MS)	Accuracy (%)
A	1k	attention-aware	20	knn	1.91	1.23	27	93.0
B	1k	local-aware [14]	20	knn	1.89	1.55	30	92.9
C	1k	-	20	knn	1.09	0.89	17	91.8
D	2k	attention-aware	20	knn	1.91	2.44	57	93.2
E	1k	attention-aware	10	knn	1.91	1.22	23	92.0
F	1k	attention-aware	40	knn	1.91	1.26	38	92.5
G	1k	attention-aware	20	ball query (r=0.1)	1.91	1.22	24	92.0

Table 3: Effectiveness of GAPLayer and attention pooling.

model	GAPLayer	Attention Pooling	OA (%)	time
H	×	×	89.2	14.7ms
I	✓	×	92.6	24.0ms
J	×	✓	92.4	18.9ms
K	✓	✓	93.0	26.0ms

Table 4: Effectiveness of different numbers of heads and encoding channels.

model	Heads	Encoding Channels F'	OA (%)	time
L	1	8	91.6	20.2ms
M	4	8	92.2	24.1ms
N	4	16	93.0	26.0ms
O	8	16	93.0	36.1ms
P	16	16	91.8	51.8ms
Q	4	32	92.9	30.9ms
R	8	32	92.5	41.3ms
S	16	32	91.1	61.3ms

former (model A) is slightly better than the local-aware method [14] (model B) and significantly better than model C, having no-spatial transformation at all. We discuss that the spatial transformer is capable of boosting the performance. Besides, benefiting from the advantages of the GAPLayer, the attention-aware spatial transformer could better canonicalize the point clouds before we process GAPointNet. For what concerns the number of the points, model D shows that when the number of points becomes much larger, the accuracy only slightly improves, but other performance metric significantly degenerated. We also note that too many points of the point clouds are likely to lead to over-fitting, which is harmful to the neural networks. Furthermore, setting an inappropriate number of neighbouring points (model E and model F) greatly deteriorates the performance of the model. In fact, the number of neighbouring points is equivalent to the receptive field of the network, and the model is unlikely to capture sufficient useful local features from a small receptive field (i.e. 10 points in our study). In contrast, using a large receptive field makes for the model more difficult to learn useful information. Hence, this hyper-parameter show some potential for fine-tuning operations. We also compare different grouping methods, specifically the k-nn method (model A) and the ball query method (model G). We discuss the reason that k-nn grouping method is more efficient when the layout of point clouds can be more or less treated as uniform distribution. However, when the dataset is extremely non-uniform, we believe that the ball query method is a better choice.

Furthermore, the effectiveness of our GAPLayer and attention pooling layer have been evaluated in Table 3 in terms of accuracy and forward-propagation time. Our GAPLayer module and attention pooling layer show a considerable increase in accuracy: the 3.4% and 3.2% respectively. The application of both components leads to a total accuracy improvement of 3.8%. It is worth highlighting that, when both components are removed, our model becomes *PointNet*, which leads to 89.2% accuracy. We discuss that the main advantage of the GAPLayer is that it could better represent the relative geometric relation expression between each center point and all its neighbouring points. On the other hand, the attention pooling layer is capable of capturing the most important feature in the local region. As a result, we combine both of them together to fully exploit the local features for each center point and boost the overall performance.

For what concerns the impact of the different numbers of heads M and encoding channels F', the results are presented in Table 4. It shows that an appropriate number of heads and channels is beneficial to local feature extraction, however the accuracy drops significantly and the computation cost

Table 5: Semantic part segmentation results on ShapeNet part dataset.

	avg	air.	bag	cap	car	cha.	ear.	gui.	kni.	lam.	lap.	mot.	mug	pis.	roc.	ska.	tab.
Kd-Net [21]	82.3	82.3	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
Kc-Net [15]	83.7	82.8	81.5	86.4	77.6	90.3	76.8	91.0	87.2	**84.5**	95.5	69.2	94.4	81.6	60.1	75.2	81.3
PointNet [12]	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
3DmFV [40]	84.3	82.0	84.3	86.0	76.9	89.9	73.9	90.8	85.7	82.6	95.2	66.0	94.0	82.6	51.5	73.5	81.8
RSNet [41]	84.9	82.7	**86.4**	84.1	78.2	90.4	69.3	91.4	87.0	83.5	95.4	66.0	92.6	81.8	56.1	75.8	82.2
PointNet++ [13]	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
DGCNN [14]	85.1	**84.2**	83.7	84.4	77.1	**90.9**	**78.5**	**91.5**	**87.3**	82.9	**96.0**	67.8	93.3	82.6	59.7	75.5	82.0
SGPN [42]	**85.8**	80.4	78.6	78.8	71.5	88.6	78.0	90.9	83.0	78.8	95.8	**77.8**	93.8	**87.4**	60.1	**92.3**	**89.4**
OURS	84.9	84.0	86.2	**88.8**	**78.3**	90.7	70.4	91.3	**87.3**	82.8	**96.0**	68.7	**95.1**	82.0	**63.0**	74.8	81.4

increases remarkably when the number becomes too large. For example, model S with 16 heads and 32 encoding channels achieves the worst performance in both accuracy and forward time. It is assumed that model S is learning too much information and leads quickly to over-fitting. We also discuss that too many heads tend to aggregate a very high-dimensional but low-level feature, making the overall model performance even worse. That is why a higher dimension of features is normally used to represent higher-level of abstraction in the deeper layers of a model. In contrast, smaller numbers (model L with 1 head and 8 encoding channels) lead to the fact that the model is unlikely to extract sufficient useful information.

We investigated the robustness and stability on variant attention components (see Table 3) by randomly reducing the number of points, and as illustrated in Fig. 4, it is possible to observe that the sparser the points, the worse the performance. Our classification model is still considerably stable and robust even when the number of points is halved. However, the performance degenerates dramatically when the number of points are reduced to less than 256.

The efficiency of the non-linear activation function when merging the local-coefficients and the self-coefficients using a sum operation has also been studied. When the non-linear activation function leaky ReLU was removed, the performance dropped from 93.0% to 92.7%.

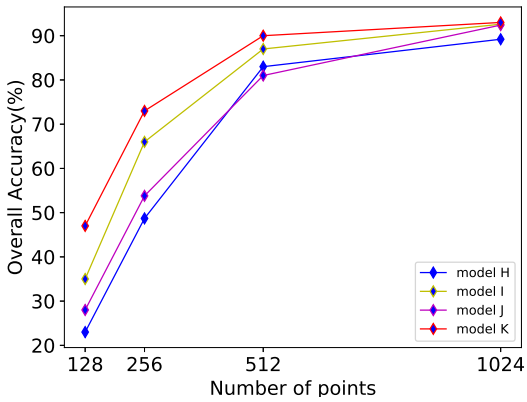


Figure 4: Random point dropout in point cloud.

4.2. Semantic segmentation

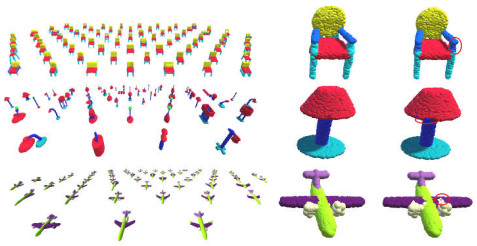
The segmentation model is evaluated on standard datasets such as the ShapeNet part [43], the Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS) [44], and, in order to test realistic application scenarios, the KITTI object detection benchmark [20] dataset that includes more than 100K points in a single frame. Hereinafter, a short description of the used dataset is presented.

ShapeNet part dataset. The dataset consists of 16881 CAD shapes of 16 categories and each point from a model is annotated with one of 50 part classes. Besides, each shape model is labelled with several but less than 6 parts. We follow the same sampling strategy as described in Section 4.1 to sample 2048 points uniformly, and split dataset into 14007 models for training and 2874 models for testing in our experiment. The task is to classify the part category for each point from a mesh model.

S3DIS dataset. S3DIS is a large-scale point cloud dataset that contains 3D geometric shape information (XYZ) and color feature (RGB). It is collected from 271 rooms in 6 areas using a Matterport scanner. All the rooms are sliced into blocks of 1 meter by 1 meter. We sampled 4096 points from every individual block and converted the feature to 9D-dimensions (XYZ, RGB, and normalized spatial

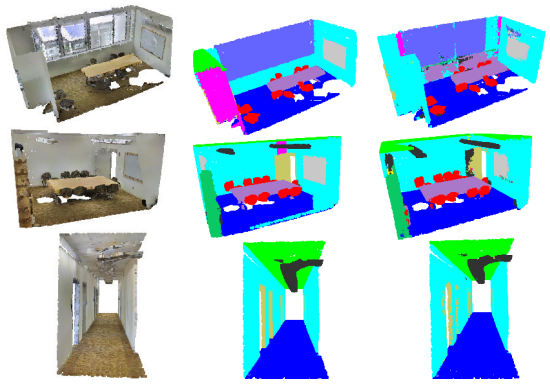
coordinate). Our model was tested on area 5, whilst all other areas are used for training.

KITTI dataset. KITTI Object Detection Benchmark [20] is a real traffic scene dataset collected by a Velodyne HDL-64E Laserscanner, and each frame contains more than 100000 points. This poses a serious challenge to our model, as it cannot be applied directly on all points. Therefore, we firstly filtered out the points outside the image view and then the rest of the points are selected by random sampling of 11469 points within 40 meters and 4915 points for the rest. Hence, with this strategy we effectively downsampled the point cloud from around 100000 to 16384 points. The KITTI dataset was then split into 7481 frames for training and 191 frames for testing.



(a) Visualization (b) Comparison

Figure 5: Visualization of semantic part segmentation results. Figure (a) visualizes some samples: chair (top), lamp (middle), and airplane (bottom). While Figure (b) visualizes the difference between ground truth (left) and prediction (right).



(a) RGB points (b) Ground Truth (c) Prediction

Figure 6: Example visualization of semantic segmentation on S3DIS dataset.

Training details. The training settings were very much similar to the settings used for the classification task, except for the batch size that was set to 8, the number of neighbours k is set to 30, and that we distributed the task to two NVIDIA TESLA V100 GPUs.

Results. We use the mean Intersection over Union (mIoU) [12] as our evaluation scheme to align the evaluation metrics. The IoU of each shape is calculated by averaging IoUs for all parts that fall into the same category, and then the mIoU is the mean IoUs for all shapes from testing dataset.

We can observe Table 5 that our model achieves competitive results on the ShapeNet part dataset [43]. In fact, GAPointNet and *DGCNN* [14] win most of categories for part segmentation. *PointNet++* has no best categories, although the average accuracy slightly outperforms than our model. We also notice that *SGPN* [42] achieves the best performance due to the fact that it takes the advantages from the instance segmentation task, which employs additional losses (e.g. losses for similarity matrix and confidence) to precisely distinguish different instances with the same category. As a result, it also boosts the performance for the semantic segmentation task. In addition to some shape examples from our results that are represented in Fig. 5(a)), we also visualize the difference between the ground-truth and our prediction results, and highlight some of the errors (red circle) as shown in Fig. 5(b), where the left shapes are the ground truth and right shapes show the prediction of our model.

Finally, we present the results of our model when applied on the KITTI and S3DIS dataset in Table 6 and Table 7 respectively. As shown in Table 7, our segmentation model performs better overall accuracy than *DGCNN* [14], and *SPGraph* [47] achieves the best performance. In fact, *SPGraph* organizes the point clouds into different object parts (e.g. legs and surface for the table). As a result, it becomes much easier to identify these relative bigger parts, rather than classifying individual points. We visualise the prediction and compare our results with the ground truth for S3DIS dataset in Fig. 6. Hence, our model shows promising effectiveness on large scale and real world point clouds. Finally, our model achieves the best performance over the KITTI dataset, outmatching any other available model, which shows its great potential for industrialization in the context of autonomous driving applications.

Table 6: Segmentation results on KITTI.

model	mIoU	accuracy	vehicle	bicyclist	pedestrian	background
PointNet [12]	38.1%	92%	76.7%	2.9%	6.6%	89.8%
PCCN [45]	58.1%	95.5%	**91.8%**	40.2%	47.7%	89.3%
OURS	**74.8%**	**98.9%**	85.3%	**51.7%**	**63.2%**	**98.8%**

Table 7: Segmentation results on S3DIS Area 5.

model	mIoU	overall accuracy
SegCloud [46]	48.9%	-
PointNet [12]	47.6%	78.5%
DGCNN [14]	56.1%	84.1%
OURS	51.2%	85.0%
SPGraph [47]	**58.0%**	**86.4%**

5. Conclusions

In this paper, we proposed a graph attention based point neural network, named GAPointNet, which is able to learn shape representations for point cloud data. Typically, solely the max pooling operation is used to capture the local feature for each central point in the point cloud. Conversely, we leveraged graph attention with a multi-head mechanism to capture attention-aware local features, and after aggregating with local signature features captured from our attention pooling layer, we finally obtained a contextual and fine-grained local feature for each point in the point cloud. Experiments show, for GAPointNet, a state-of-the-art performance for both the shape classification and semantic segmentation tasks on various datasets. In particular, GAPointNet shows a remarkable 98.9% accuracy on the segmentation task of the KITTI dataset, which shows great potential for real-time applications, such as autonomous driving. The success of our model also verifies the fact that graph attention network shows efficiency not only in the similarity computation for graph nodes, but also for geometric relationship understanding.

In the future, we plan to explore several research avenues. In fact, some applications, such as autonomous vehicle, normally need to process very large-scale point cloud data. As a result, how to efficiently and robustly deal with large-scale data, without the need of initial downsampling, would be a promising work. Furthermore, it would be interesting to develop an efficient *CNN*-like operation for unstructured data analysis.

References

[1] Y. Zhou, O. Tuzel, Voxelnet: End-to-end learning for point cloud based 3d object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4490–4499.

[2] C. R. Qi, W. Liu, C. Wu, H. Su, L. J. Guibas, Frustum pointnets for 3d object detection from rgb-d data, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 918–927.

[3] J. Ku, M. Mozifian, J. Lee, A. Harakeh, S. L. Waslander, Joint 3d proposal generation and object detection from view aggregation, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 1–8.

[4] Z. Liu, H. Chen, H. Di, Y. Tao, J. Gong, G. Xiong, J. Qi, Real-time 6d lidar slam in large scale natural terrains for ugv, in: 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2018, pp. 662–667.

[5] J. Biswas, M. Veloso, Depth camera based indoor mobile robot localization and navigation, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, 2012, pp. 1697–1702.

[6] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, A. Farhadi, Target-driven visual navigation in indoor scenes using deep reinforcement learning, in: Robotics and Automation (ICRA), 2017 IEEE International Conference on, IEEE, 2017, pp. 3357–3364.

[7] A. Golovinskiy, V. G. Kim, T. Funkhouser, Shape-based recognition of 3d point clouds in urban environments, in: Computer Vision, 2009 IEEE 12th International Conference on, IEEE, 2009, pp. 2154–2161.

[8] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[9] D. Maturana, 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.

[10] D. Z. Wang, I. Posner, Voting for voting in online point cloud object detection., in: Robotics: Science and Systems, Vol. 1, 2015, pp. 10–15607.

[11] G. Riegler, A. Osman Ulusoy, 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.

[12] C. R. Qi, H. Su, K. Mo, L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, Proc. Computer Vision and Pattern Recognition (CVPR), IEEE 1 (2) (2017) 4.

[13] C. R. Qi, L. Yi, H. Su, L. J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric

space, in: Advances in Neural Information Processing Systems, 2017, pp. 5099–5108.

[14] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, J. M. Solomon, Dynamic graph cnn for learning on point clouds, arXiv preprint arXiv:1801.07829.

[15] Y. Shen, C. Feng, Y. Yang, 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.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

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

[18] V. Mnih, N. Heess, A. Graves, et al., Recurrent models of visual attention, in: Advances in neural information processing systems, 2014, pp. 2204–2212.

[19] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903.

[20] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The kitti dataset, The International Journal of Robotics Research 32 (11) (2013) 1231–1237.

[21] R. Klokov, 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.

[22] J. L. Bentley, Multidimensional binary search trees used for associative searching, Communications of the ACM 18 (9) (1975) 509–517.

[23] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, 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.

[24] C. Wang, M. Pelillo, K. Siddiqi, Dominant set clustering and pooling for multi-view 3d object recognition, in: Proceedings of British Machine Vision Conference (BMVC), Vol. 12, 2017.

[25] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond euclidean data, IEEE Signal Processing Magazine 34 (4) (2017) 18–42.

[26] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, B. Chen, Pointcnn: Convolution on x-transformed points, in: Advances in Neural Information Processing Systems, 2018, pp. 828–838.

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

[28] A. Komarichev, Z. Zhong, J. Hua, A-cnn: Annularly convolutional neural networks on point clouds, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7421–7430.

[29] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, arXiv preprint arXiv:1312.6203.

[30] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in neural information processing systems, 2016, pp. 3844–3852.

[31] Y. Zhang, M. Rabbat, A graph-cnn for 3d point cloud classification, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 6279–6283.

[32] Y. Zhang, M. Rabbat, A graph-cnn for 3d point cloud classification, in: International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, Canada, 2018.

[33] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, 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.

[34] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, arXiv preprint arXiv:1812.08434.

[35] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, arXiv preprint arXiv:1505.00853.

[36] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, 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.

[37] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[38] C. Wang, B. Samari, 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.

[39] M. Atzmon, H. Maron, Y. Lipman, Point convolutional neural networks by extension operators, arXiv preprint arXiv:1803.10091.

[40] Y. Ben-Shabat, M. Lindenbaum, A. Fischer, 3d point cloud classification and segmentation using 3d modified fisher vector representation for convolutional neural networks, arXiv preprint arXiv:1711.08241.

[41] Q. Huang, W. Wang, 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.

[42] W. Wang, R. Yu, Q. Huang, U. Neumann, Sgpn: Similarity group proposal network for 3d point cloud instance segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2569–2578.

[43] L. Yi, V. G. Kim, D. Ceylan, I. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, L. Guibas, et al., A scalable active framework for region annotation in 3d shape collections, ACM Transactions on Graphics (TOG) 35 (6) (2016) 210.

[44] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, 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.

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

[46] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, S. Savarese, Segcloud: Semantic segmentation of 3d point clouds, in: 2017 International Conference on 3D Vision (3DV), IEEE, 2017, pp. 537–547.

[47] L. Landrieu, 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.

GAPointNet: Graph attention based point neural network for exploiting local feature of point cloud

Chen, Can

2021-01-26

Chen C, Zanotti Fragonara L, Tsourdos A. (2021) GAPointNet: Graph attention based point neural network for exploiting local feature of point cloud. Neurocomputing, Volume 438, May 2021, pp.122-132

https://doi.org/10.1016/j.neucom.2021.01.095