Graph Attention Networks for Point Cloud Processing

by

Sumesh Thakur

A Thesis Submitted to

Saint Mary's University, Halifax, Nova Scotia
in Partial Fulfilment of the Requirements for
the Degree of Master of Science in Applied Science

July, 2021, Halifax, Nova Scotia

Approved by:

Dr. Jiju Poovvancheri,
Supervisor

Dr. Othman Soufan,
External Examiner

Dr. Jason Rhinelander,
Examiner

Dr. Stavros Konstantinidis,
Examiner

Date: 31 July 2021

Acknowledgements

I would like to express my deep and heartfelt gratitude for my respected guide and mentor, Dr. Jiju Poovvancheri, for his evergreen blessings, valuable advice, support, good wishes and encouragement all along the journey of this work. Without his undisputed support, this work would not have been true in the light of daylight. I am also grateful to the members of my supervisory committee (Dr. Stavros Konstantinidis and Dr. Jason Rhinelander) for reviewing my work and for their valuable comments.

Place: Halifax
Date: 31 July 2021

Sumesh Thakur

Abstract

"Graph Attention Networks for Point Cloud Processing"

By Sumesh Thakur

Three-dimensional point cloud datasets are becoming ubiquitous due to the availability of consumer-grade 3D sensors such as Light Detection and Ranging (LIDAR), and RGB-D cameras. Recent advancements in 3D deep learning has dramatically improved the ability to recognize physical objects and interpret the indoor and outdoor scenes using point clouds acquired through different sensors. This thesis focuses on deep learning based techniques for point cloud processing. We propose novel architectures leveraging graph attention networks for point cloud-based object detection, classification, and segmentation. The proposed architectures work on point cloud scans directly by constructing a connected graph. For point cloud detection, we use the concatenation of relative geometric difference and feature difference between each pair of neighbouring points in the graph. To improve the performance of object detection, we introduce a distance-aware down-sampling scheme for object detection space. For point cloud segmentation and classification, we employ a global aware attention module using global, local, and self feature information. The experiments on different datasets (KITTI, ShapeNet, ModelNet, and Semantic3D) show that our methods yield comparable results for object detection, part segmentation, semantic segmentation, and classification.

Date: 31 July 2021

Contents

1 Introduction **1**
- 1.1 Introduction . . . 1
- 1.2 Challenges . . . 3
- 1.3 Graph Neural Networks . . . 5
- 1.4 Graph Attention Networks . . . 7
- 1.5 Contributions . . . 8
- 1.6 Thesis Organization . . . 9

2 Related Work **10**
- 2.1 Object Detection . . . 10
  - 2.1.1 Voxel based Methods . . . 11
  - 2.1.2 Point-Wise MLP based Method . . . 12
  - 2.1.3 Graph based Methods . . . 13
- 2.2 Classification . . . 13
  - 2.2.1 Voxel based Methods . . . 13
  - 2.2.2 Point-wise MLP based Methods . . . 14
  - 2.2.3 Graph Based Methods . . . 15
- 2.3 Segmentation . . . 16
  - 2.3.1 Point-wise MLP based methods . . . 16
  - 2.3.2 Graph based Methods . . . 16
  - 2.3.3 Graph Attention based Methods . . . 19

3 GAT3D - Graph Attention Network for 3D Object Detection **21**
- 3.1 Distance Aware Down Sampling . . . 22
- 3.2 Attention based Aggregation . . . 22
  - 3.2.1 Relationship with Prior Methods . . . 24
  - 3.2.2 Loss . . . 25
- 3.3 Implementation Details . . . 27
- 3.4 Experiments . . . 28
  - 3.4.1 Evaluation Metrics for Object Detection . . . 28

3.4.2 Qualitative Results 29
3.4.3 Quantitative Comparison 30
3.4.4 Ablation Studies 31

4 GAGAT - Global Aware Graph Attention Network for 3D Segmentation and Classification 34
4.1 Global Aware GAT Layer 34
4.2 Segmentation and Classification 37
4.3 Relationship with Prior Works 38
4.4 Loss 38
4.5 Implementation Details 39
4.6 Experiments 40
4.6.1 Classification 40
4.6.2 Part Segmentation 40
4.6.3 Indoor Scene Segmentation 43
4.6.4 Outdoor Scene Segmentation 44
4.7 Ablation Study and Differences between GAT3D and GAGAT 47

5 Conclusions and Future Direction 49
5.1 Conclusion 49
5.1.1 GAT3D - Graph Attention Network for 3D Object Detection . 49
5.1.2 GAGAT - Global Aware Graph Attention Network for Classification and Segmentation 50
5.2 Future Directions 51

Chapter 1
Introduction

1.1 Introduction

Deep learning methods have shown to be successful in tackling computer vision problems such as image classification [1], object detection [2, 3], and face recognition [4]. With the advancements 3D sensors hardware, there is a growing interest in understanding 3D data employing deep learning techniques. Many attempts have been made to expand the convolutional architecture, which has shown to be quite successful in interpreting 2D images, to 3D data. The output of 3D sensors, on the other hand, is frequently a collection of 3D points representing x, y and z coordinates and properties (reflectance value, RGB values etc.), combined referred to as point clouds. Point clouds have tons of applications in robotics, autonomous driving, and virtual reality based systems. Point clouds, unlike 2D images, do not have a regular grid and so lack the translational-invariant structure that enables convolution. Raw point clouds are frequently translated to another representation such as 3D voxel grids [5, 6] or rendered 2D images [7]. However, these conversions are often irreversible, and tear down the local geometric relationship between points in the original point cloud, which makes it difficult to use 2D processing methods on point cloud data. There exists many classic computer vision tasks, many of which have been successfully tackled using deep learning frameworks. In this thesis, we focus on a subset of these tasks for 3D point clouds as listed below.

Detection Object detection is a computer vision technique that deals with distinguishing between objects in an image, video or point cloud scan. Object detection consists of two parts i.e., object classification and localization of object using bounding box. Object classification deals with the process of predicting the correct label of object whereas localization deals with regressing the coordinates of bounding box. Figure 1.1 visualizes an example of object detection.

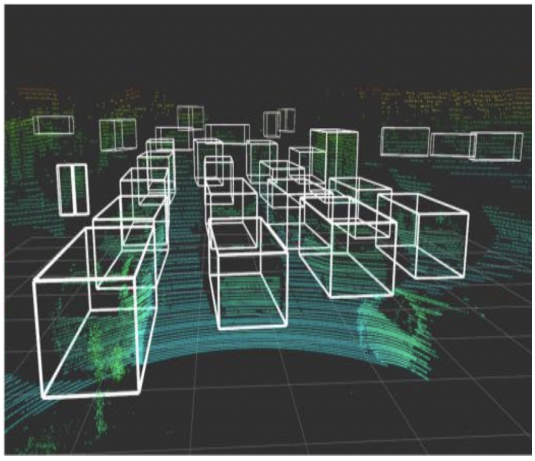


Figure 1.1: An example of object detection [5] on KITTI dataset.

Classification Object Classification refers to a type of labelling where an image, video or a point cloud scan is assigned certain concepts, with the goal of answering the question, "What is in this image or video or point cloud?". A point cloud scan can be classified into a number of categories based on the geometric representation of the scan. Figure 1.2 shows the label predictions by Pointnet [8] architecture.

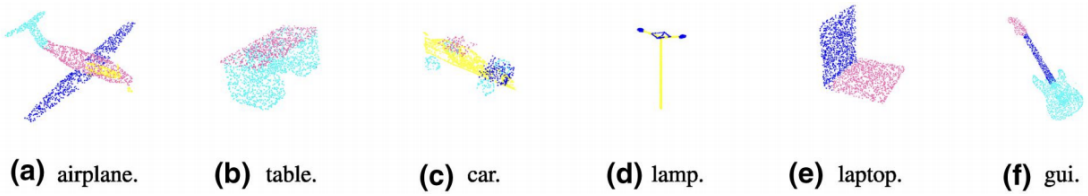


Figure 1.2: An example of object classification [8] on Modelnet-40 dataset.

Segmentation Segmentation is a type of labelling where each point in a point cloud scan is labelled with given concepts. Here, the whole scan is divided into groups which can then be labelled and classified, with the goal of simplifying a point cloud scan or changing how a point cloud is presented to the model, to make it easier to analyse. Segmentation models provides the exact outline of the object within a point cloud scan. That is, point by point details are provided for a given object, as opposed to classification models, where the model identifies what is in a point cloud scan, and detection models, which places a bounding box around specific objects. Figure 1.3 shows an example result from PointCNN [9]. 3D segmentation can be

further divided into two types, semantic segmentation and part segmentation. Semantic segmentation focuses on predicting labels of different objects in a surrounding, whereas part segmentation subdivides an object into different parts (i.e. wings of a plane, legs of a chair etc) and predicts their labels.

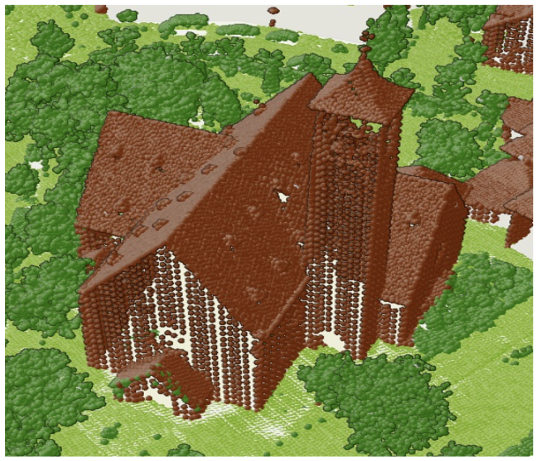


Figure 1.3: An example of object segmentation [9].

1.2 Challenges

Extraction of meaningful information from 3D scans is a fundamental challenge in the field of computer vision. Much like the two-dimensional (2D) image understanding, 3D understanding has greatly benefited from the current technological surge in the field of machine learning. However, 3D processing algorithms are not as precise as 2D processing algorithms. The main properties in point clouds that make it difficult to learn meaningful features are as follows:

1. **Unordered and lacks regular structure**: 3D data, unlike 2D image data, usually lacks a regular structure. A typical point cloud scan is an unorganized collection of points that is not localised in the same way that 2D data is, as visualized in Figure 1.4.
2. **Concept of relative neighbourhood**: Point clouds are $\mathbb{R}^3$ subspaces with a distance metric attached. A meaningful subset is formed by a collection of points, and feature representations should reflect this property.

Figure 1.4: An example of unordered point cloud scan [8].

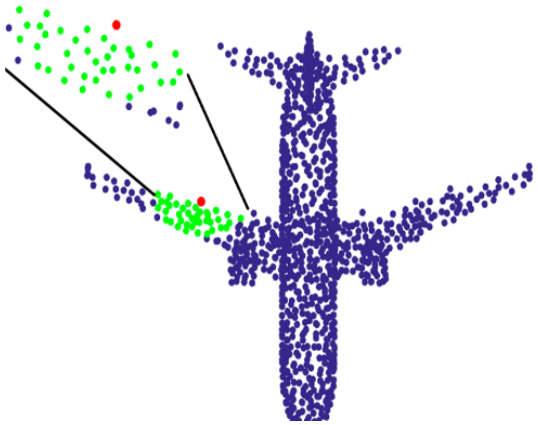


Figure 1.5: As evident in this visualization, the idea of relative neighbourhood can be very uncertain [10].

3. **Affine invariance**: Feature representations should be invariant to non-deforming rigid transformations (rotation and translation) of the whole point cloud as visualized in Figure 1.6.

4. **Sparsity**: When compared to 2D data, this is by far the most differentiating feature of point clouds. The point cloud is contained in a 3D volume that is not densely packed. The local density of point clouds is considerable, yet the overall volume occupied by the points is quite low. As a result, imposing a grid on a point cloud and utilising 3D convolutions is computationally expensive because this approach will be processing on empty voxels the majority of the time.

In recent years, researchers have demonstrated their work on various standard bench-

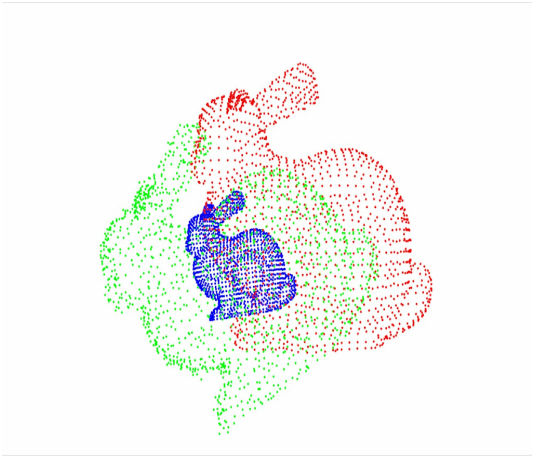


Figure 1.6: The inference model should be invariant to affine transformations. Image courtesy [11].

marks such as KITTI object detection benchmark [12] for 3D object detection, Shapenet benchmark [13] for 3D segmentation, Modelnet benchmark [14]for 3D classification. In this work, we explore the technique of converting point clouds to a representation suitable for deep learning, without destroying any geometric information. Specifically, we connect neighbouring points in a point cloud to form an undirected graph. We use underlying relationship between vertices and their neighbourhood to learn meaningful features for detection, segmentation and classification purposes. Graph represented leaning based approach has shown appreciable results [8, 15, 16, 17] on point cloud learning algorithms. The constructed graphs are processed by graph neural networks, which is discussed in section 1.3.

1.3 Graph Neural Networks

A graph neural network (GNN) is a form of neural network that directly works on graph structures. Node classification and prediction are some of the common applications of GNN. In GNN, each node is assigned with a label, which is used to predict the labels of the nodes without using ground-truth data. A graph is a tuple $G = (V, A, E)$ where V is the set of vertices, E is the set of edges and A is the adjacency matrix that associates each edge $e \in E$ with a weight. Each node is the graph $v_i \in V$ can be associated with a feature $f_i \in \mathbb{R}^D$. A graph convolutional neural network or GNN takes as input a graph G and learns corresponding features f_i for each vertex. One of the popular approaches in graph neural networks is spatial graph

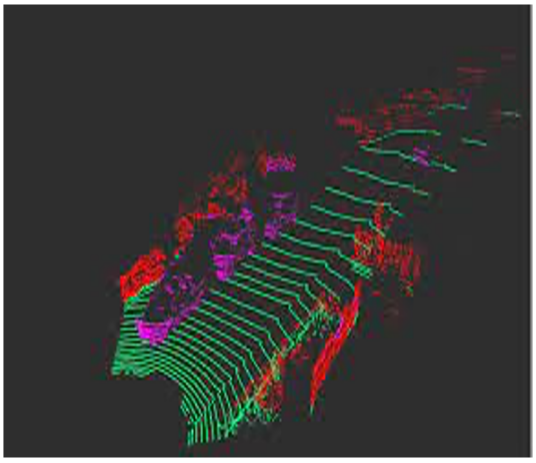


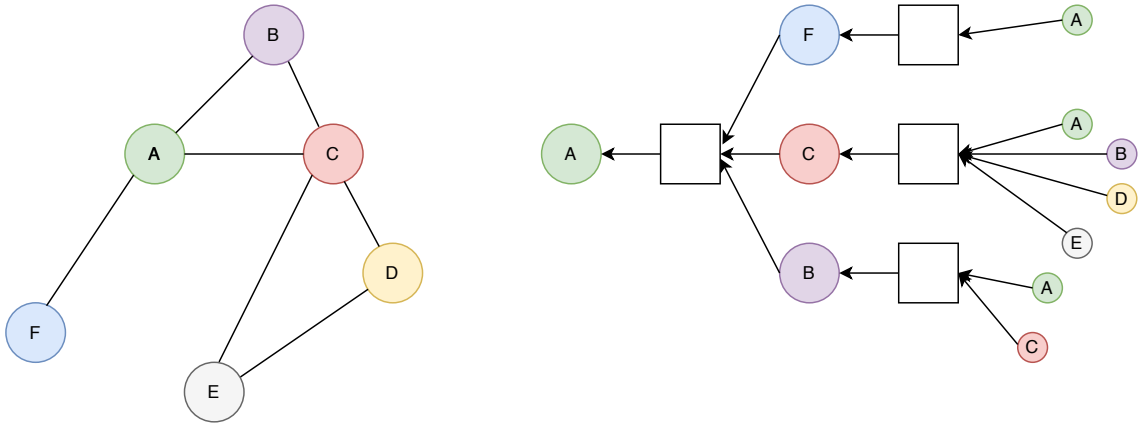
Figure 1.7: In this particular visualization, it is evident that majority of space in scan is just empty space [5].

convolution. Let $S = s_1, s_2, s_3, ...s_N \in \mathbb{R}^F$ be a set of input features, associated with vertex $u \in V$, at $(k-1)^{th}$ iteration. A single iteration of a graph neural network (GNN) aggregates features from k nodes in a neighbourhood $N(u)$ of a given node u such that the updated feature s_u^t of vertex u at t iteration is given by:

$$s_u^t = \sigma(W_t \phi_{uv}^{t-1}(s_v^{t-1}) \otimes B_t s_u^{t-1}), v \in N(u) \tag{1.1}$$

Here, W_t and B_t are trainable weight and bias matrices and σ is the activation function (e.g. ReLU) to introduce non-linearity. The $\otimes$ here refers to the concatenation function. The function ϕ_{uv}^{t-1} aggregates features along the edges. This function updates feature and repeats the process in every iteration. The function visualization has been described in Figure 1.8.

One of the feature of GNNs is that it gives equal weithage to every node that is present in neighbourhood of query node. This might not be desirable in some cases (like social media interactions, citation models), since a neighbouring node may or may not posses relevant features to the query node. Velivckovic et al. [17] cited this limitation of GNNs and proposed Graph Attention Networks. GATs addresses the shortcomings and approximations of prior methods based on graph convolutions by leveraging masked self-attentional layers. GATs is explained in section 1.4.



(i) (ii)

Figure 1.8: Let node 'A' be the target node. In one iteration of GNN, model takes the features from all the immediate neighbours of target node and aggregate them with an invariant aggregation function.

1.4 Graph Attention Networks

Graph Attention Network (GAT), is an extension on graph neural networks. GATs uses masked attentional layers to solve the flaws and approximations of preceding approaches based on graph convolutions. The concept of attention in GATs is derived from a past work by Vaswani et al. [18]. The method allows specifying different weights to different nodes in a neighbourhood by stacking layers in which nodes are able to attend to their neighbourhoods' features, without requiring any kind of costly matrix operation (such as inversion) or relying on knowing the graph structure upfront. GAT tackles many limitations of spectral-based graph neural networks at the same time, allowing the model to be applied to both inductive and transductive applications. Analysing and visualising learnt attentional weights also results in a better interpretable model in terms of neighbour importance. Figure 1.9 explains how graph attention networks assign different weights to nodes in contrast to graph convolution networks. Figure 1.9 explains how graph attention networks assign different weights to nodes in contrast to graph convolution networks. GNN explicitly assigns non-parametric weight $\alpha = \frac{1}{\sqrt{deg(v_i)deg(v_j)}}$, via normalization function during neighbourhood aggregation. GAT implicitly captures the weight α_{ij}, via the attention mechanism, so that more important nodes receive higher weight during neighbourhood.

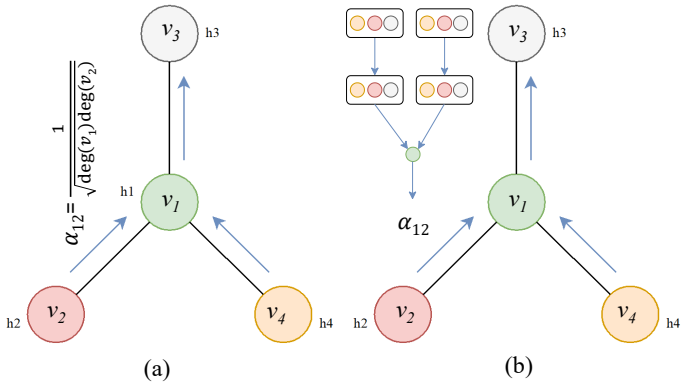


Figure 1.9: Comparison between GNNs and GAT [19]. Figure **(i)** represents the structure of GNN layer, whereas **(ii)** represents the structure of GAT.

1.5 Contributions

In this thesis, we investigate the possibility of employing graph attention networks for object detection, segmentation and classification from point clouds. We propose two architectures: one for object detection and the second, for point cloud classification and segmentation. Our object detection architecture formulation utilizes the relative coordinates along with the vertex features to produce relative edge weights. The key idea of our work is as follows. Based on the spatial positions and feature attributes of the neighbourhood vertices, we learn to strengthen or weaken the edge weightage accordingly. This approach allows our model to dynamically adapt to the structure of the objects. For segmentation and classification, we utilize global aware attention system, taking into account global, local and self features to enhance feature learning process. In object detection, to enhance the algorithmic performance, we introduce a new distance aware voxelization method, which downsamples the point cloud scans from LiDAR sensor without losing the relevant information required for graph generation. In summary, we make the following key contributions in this work.

- **Down sampling algorithm**: We introduce a distance aware downsampling algorithm for 3D object detection, that employs variable sized voxels depending on the distance of points from the sensor origin to sub-sample the point cloud. These downsampled points are used to construct a nearest neighbour graph. The algorithm maximizes geometric features of objects in the downsampled point cloud even if they lie far from the sensor origin.

- **Attention based feature aggregation for 3D object detection**: We design a single stage GNN based algorithm for object detection and localization. Feature aggregation is performed using a masked attention by assigning different weights to different neighbouring nodes.
- **Global aware graph attention network for point cloud processing**: We design an attention based graph neural networks which also leverages global features to perform classification and segmentation on point clouds. For learning features at different levels, we use self and local attention on each graph constructed in hierarchical manner, along with the global information.

1.6 Thesis Organization

This thesis is organized into following chapters.

- **Introduction**: This chapter (Chapter 1) consists of an overview of concepts and a brief introduction to the problem set. This chapter covers the problem introduction, challenges and a overview of concepts needed to understand this thesis.
- **Related Work**: This chapter (Chapter 2) consists of a literature review of state of art methods divided in three categories based on their use case i.e (object Detection, classification and segmentation). These categories are sub divided into different section based on the learning approach used by different methods.
- **GAT3D - Graph Attention Network for 3D Object Detection**: This chapter (Chapter 3) consists of an in-depth explanation of our 3D object detection architecture, along with implementation details. We also cover the performance of our architecture on KITTI benchmark dataset and a few ablation studies.
- **GAGAT - Global Aware Graph Attention Network for 3D Segmentation and Classification**: This chapter (Chapter 4) comprises explanation an an in-depth analysis of our classification and segmentation architecture. We also cover qualitative and quantitative performance of our architecture in this chapter.
- **Conclusion**: We conclude our thesis in the last chapter (Chapter 5). We also list potential future directions of our work in this chapter.

Chapter 2
Related Work

Point cloud learning has recently captivated a lot of attention by research community because of its numerous applications in fields like computer vision, autonomous driving, and robotics. Deep learning has been effectively used to solve numerous 2D vision problems as one of the major methodologies. However, due to the particular constraints of processing point clouds, deep learning in the 3D domain is still in its infancy. In general, 3D data is acquired in the form of a point cloud by LiDAR sensors, RGB-D cameras, or photogrammetry techniques. A point cloud is a set of data points in 3D space where every point has x, y, z values representing its geometric position along with the features associated with it, like reflectance value, intensity values or RGB values. In 2017, Qi et al. proposed PointNet [8] which demonstrated how to directly manipulate point cloud with neural networks. Since then, majority of the research on point cloud processing has switched to employing various 3D data representations using neural networks. In the past, the processing of point clouds for visual intelligence has been based on handcrafted features. Although handcrafted features do require a lot of feature engineering but they do not require large training data. However, with the increasing availability of acquisition devices, large, open and annotated public data such as KITTI [12], Shapent [13] etc. are making use of deep learning for their processing. In this chapter, we review various deep learning algorithms for object detection, point cloud classification and point cloud segmentation.

2.1 Object Detection

Following the trend of 2D object methods, the traditional methods for 3D proposal generation utilized hand-crafted features to generate a small set of candidate boxes that retrieve most of the objects in 3D space. 3DOP [20] is one of the significant algorithms in this direction. It uses hand-crafted geometric features from stereo point

clouds to score 3D sliding windows in an energy minimization framework. The top k-scoring windows are selected as region proposals, which are then fed to a modified Fast-RCNN to finally generate 3D bounding boxes. MONO3D [21] uses the same framework, but instead exploits plane prior and handcrafted features from semantic segmentation outputs to generate 3D proposals from monocular images.

2.1.1 Voxel based Methods

Voxel (also known as volumetric pixel or volume elements) is the smallest unit of volume when dividing 3D space into discrete, uniform regions. Voxel based object detection converts a point cloud data into symmetrical 3D grid and inputs it into convolution based layers. This approach extends the basic principle of 2D object detectors to 3D object detection. VeloFCN [22] projects a LIDAR point cloud to the front view, which is fed into a fully convolutional network to generate dense 3D bounding boxes. Zhou et al. [5] presents a voxel-based end-to-end trainable framework called VoxelNet. A point cloud is partitioned into equally spaced voxels and per voxel features are encoded into a 4D tensors. Each voxel is further passed through voxel feature extractor (VFE). A region proposal network [2] is then utilized to produce the detection results. Although detection results are highly accurate in this case, this method is very slow due to the sparsity of voxels and 3D convolutions. 3D-FCN [23] extends this concept by applying 3D convolutions on 3D voxel grid from LiDAR point clouds to generate better 3D bounding boxes. The limitation of both these methods is that they use very expensive 3D convolution layers, which make it difficult to be deployed in real-time driving scenarios.

Voxelnet VoxelNet [5] is an end-to-end network that combines feature extraction and bounding box prediction. This network works directly on 3D point cloud data. To begin, the 3D space is partitioned into voxels that are evenly spaced. The points are grouped according to the voxel they belong to. The VoxelNet's first layer is an encoding layer that converts a set of points within each voxel into a feature representation, a 4D tensor. The input tensor is then passed to a 3D convolution to aggregate voxel-wise characteristics. The RPN layer takes the output of the convolutional middle layer as its input. A probability score map and a regression map are generated by the RPN. The loss is the sum of the classification loss and the regression loss. The architecture has been explained in Figure 2.1.

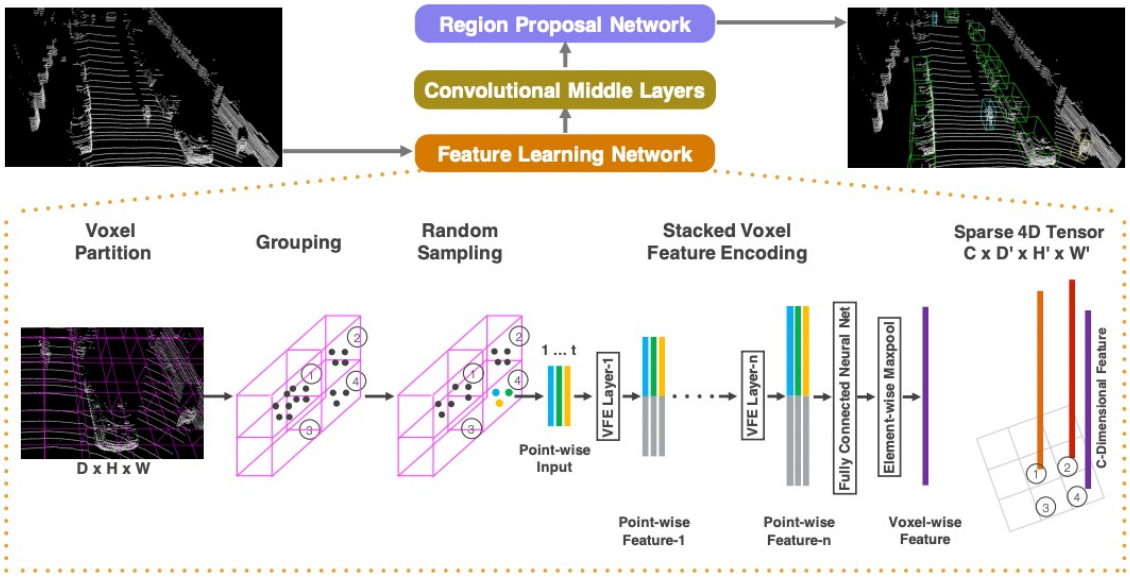


Figure 2.1: Voxelnet [5]: The feature learning network takes a raw point cloud as input, divides the space into voxels, then converts points within each voxel to a vector representation that represents the shape information.

2.1.2 Point-Wise MLP based Method

Point based methods [8] process point cloud data directly without representing or projecting it to other views such as front view or bird eye view. Lang et al. [24] proposed the PointPillars 3D object detector. This method uses PointNet to learn the features of point clouds grouped in vertical columns (referred to as Pillars) and then encodes the learnt features into a 2D pseudo image. The 3D bounding boxes are then predicted using a 2D object detection algorithm.. TA-Net [25] is one of the few deep learning based 3D detection technique that targets pedestrian detection. TA-Net contains a Triple Attention (TA) module, and a Coarse-to-Fine Regression (CFR) module. The TA module increases the target's relevant information while suppressing irrelevant information by combining channel-wise, point-wise, and voxel-wise attention. 3D SSD [26] is the latest approach in point based 3D detection. It uses fusion sampling strategy joined with anchor free regression head to regress bounding box co-ordinates. This approach achieves a good balance between accuracy and inference. Shi et al. [27] proposed PV-RCNN. This method integrates CNNs with Pointnet [8] based set abstraction to improve feature learning. Recently, Li et al. [28] improved on 3D proposals generated by pointnet architecture to produce

better 3D detection results.

2.1.3 Graph based Methods

Graphs are a type of non-Euclidean data structure that can be used to represent point clouds. In graph representation each node corresponds to an input point, and the edges represent the relationship between each point neighbors. Graph neural networks propagate the node states until equilibrium in an iterative manner. The graph CNNs operate on groups of spatially close neighbours and define convolutions directly on the graph in the spectral and non-spectral (spatial) domains. The benefit of graph-based models is that they allow investigating the geometric relationships between points and their neighbours. The grouped edge associations on each node are thus used to derive spatially local correlation characteristics. PointGNN [16] can be regarded as one of the first methods to use graph cnns for 3D detection. It designs a one-stage graph neural network to predict the category and shape of the object with an auto-registration mechanism. Chen et al. [29] proposed shape-attentive GConv (SA-GConv) to capture the local shape features, by modeling the relative geometric positions of points to describe object shapes. PointRGCN [30] leverages a graph representation for feature generation. This method uses a 2-stage object detection, where R-GCN is a residual graph CNN that classifies and regresses 3D proposals, and C-GCN a contextual GCN that further refines proposals by sharing contextual information between multiple proposals.

2.2 Classification

Classification refers to the idea of classifying a point cloud representation of an object. Classifictaion can also be considered as a sub problem in both segmentation and detection algorithms. Point cloud classification is gaining interest and becoming a very active field of research. Various classification techniques can be divided into different categories based on the learning representation used by them.

2.2.1 Voxel based Methods

Early methods usually applied 3D Convolution Neural Network (CNN) built upon the volumetric representation of 3D point clouds for point cloud classification. Daniel et al. [31] introduced VoxNet to achieve robust 3D object classification. This method used the idea of occupancy grids in point cloud scans for feature learning. Wu et al. [14] proposed a convolutional deep belief-based 3D ShapeNets to learn the

distribution of points from various 3D shapes. Le et al. [32] proposed a network called PointGrid. For point cloud learning, this technique combines the point and grid representations. Within each volumetric grid cell, a random number of points are subsampled, allowing the network to extract geometric characteristics using 3D convolutions.

2.2.2 Point-wise MLP based Methods

Point-wise MLP methods model each point independently using several Multi-Layer Perceptrons (MLPs) and then aggregate a global feature using a symmetric function. Point-wise methods are able to achieve permutation invariance, which makes it possible to process unordered points. One limitation with these methods is that they can't utilize the geometric relationships among 3D points. Pointnet [8] can be considered as pioneer method in this category. Zaheer et al. [33] explained that the key to achieve permutation invariance is by joining each representations and applying nonlinear transformations. They also designed an architecture called DeepSets, for shape classification. Joseph et al. [34] proposed an architecture called Mo-Net. The main idea behind this architecture remains similar to Pointnet, but it takes a finite set of moments as the input of its network. SRINet [35] builds on top of Pointnet architecture. It first projects a point cloud to obtain rotation invariant representations, and then utilizes PointNet-based backbone to extract a global feature and graph-based aggregation to extract local features. PA-Conv [36] is another model for point-cloud classification, which uses adaptive dynamic kernels for predictions.

Pointnet PointNet represents a milestone in deep earning for 3D data. Pointnet unifies the architecture for a variety of applications, from object categorization to semantic segmentation. PointNet works by using a shared MLP as a feature learner for local representations, then a global pooling layer to learn a context vector, which is then concatenated with local features to create a global representation for each point. The concatenated features are then used to predict the labels for 3D semantic segmentation and classification. Two main modules of Pointnet are as follows: the max pooling layer which uses a symmetric function to collect information from all the points, a local and global information combination structure, and two joint alignment networks to align both input points and point characteristics. Figure 2.2 shows the architecture of Pointnet.

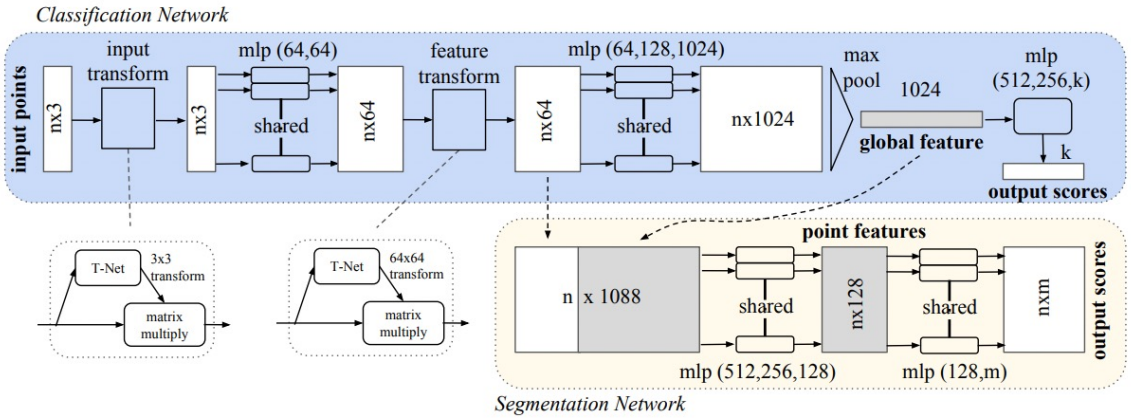


Figure 2.2: **PointNet Architecture.** [8] The classification network takes n points as input, applies input and feature transformations, and then aggregates point features by max pooling.

2.2.3 Graph Based Methods

Graph-based approaches specify operations in the spatial domain (for example, convolution and pooling). Pooling creates a new coarsened graph by aggregating information from each point's neighbours, while convolution is commonly accomplished by MLP over spatial neighbours. Co-ordinates, laser intensities, or colours are normally assigned to features at each vertex, while geometric properties between two connected points are usually assigned to features at each edge. Each point was treated as a vertex of the graph by Simonovsky et al. [37], and each vertex was connected to all of its neighbours by a directed edge. Then, utilising a filter generating network, Edge Conditioned Convolution (ECC) is proposed (e.g., MLP). To aggregate neighbourhood information, max pooling is used, and graph coarsening is done using the VoxelGrid [38] technique. Zhang et al. [39] extends the idea of CNNs to handle graph data. The architecture combines localized graph convolutions with two types of graph downsampling operations (also known as pooling). Zhai et al. [40] propose a Multi-scale Dynamic GCN model for point clouds classification. They first apply farthest point sampling method to sample points, to cover the entire point set, They use different scale k-NN group method to locate on k nearest neighborhood for each central node. They extract and aggregate local features between neighbour linked nodes and the central node using the edge convolution (EdgeConv) operation.

2.3 Segmentation

The technique of classifying point clouds into distinct categories in three dimensions is known as 3D point cloud segmentation. Because of the significant redundancy, uneven sample density, and lack of explicit organisation in point cloud data, 3D segmentation is a difficult task. A crucial stage in the processing of 3D point clouds is segmenting them into foreground and background. We look at a few prominent ways for segmenting point clouds.

2.3.1 Point-wise MLP based methods

Because of its high efficiency, these methods typically use shared MLP as the basic unit in their network, as mentioned in earlier sections. Point-wise features retrieved using shared MLP, on the other hand, are unable to capture the local geometry in point clouds as well as point-to-point interactions [8]. Several dedicated networks have been proposed to collect a broader environment for each point and learn deeper local structures. Pointnet++ [10] is a follow-up to Pointnet [8]. As seen in Figure 2.3, it clusters points hierarchically and gradually learns from bigger local regions.. It groups points hierarchically and progressively learns from larger local regions, as illustrated in Figure 2.3. Figure 2.3 visualizes the architecture of Pointnet++ architecture. Multi-scale grouping and multi-resolution grouping are also proposed to overcome the problems caused by non-uniformity and varying density of point clouds. In contrast to grouping techniques in Pointnet++, Engelmann et al. [41] utilizes K-means clustering and KNN to separately define two neighborhoods in the world space and feature space. A pairwise distance loss and a centroid loss are included to further regularise feature learning, based on the concept that points from the same class are predicted to be closer in feature space. For large-scale point cloud segmentation, Hu et al. [42] presented RandLA-Net, an efficient and lightweight network. To achieve a remarkable level of memory and processing efficiency, this network employs random point sampling. To capture and maintain geometric features, a local feature aggregation module is also provided.

2.3.2 Graph based Methods

Several methods use graph networks to capture the underlying forms and geometric features of 3D point clouds. Graph-based networks consider each point in a graph to be a vertex, and construct directed edges for the graph based on each point's neighbours. Then, a spectral or spatial domain based feature learning takes place.

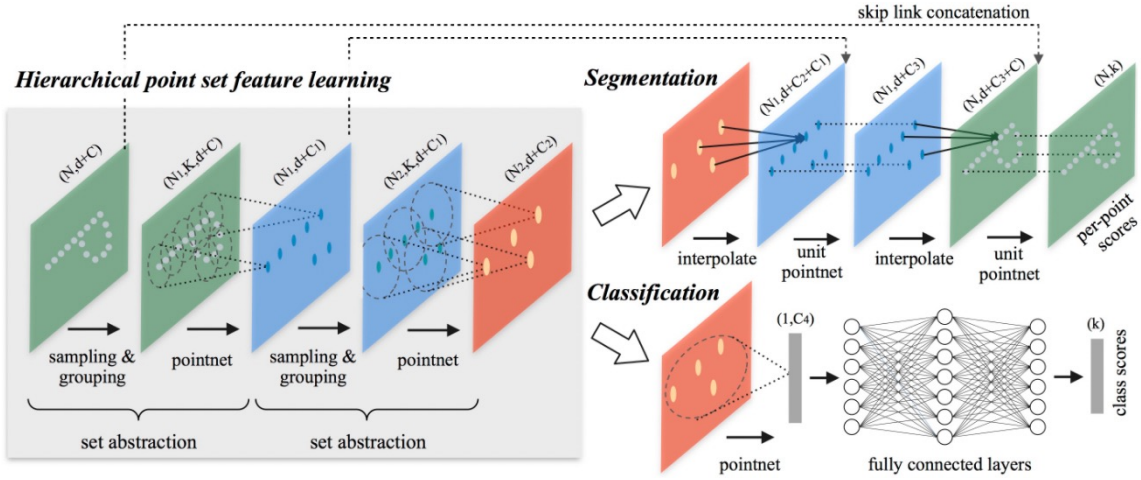


Figure 2.3: Pointnet++ partitions the set of points into overlapping local regions by the distance metric of the underlying space.

The process is illustrated in Figure 2.4.

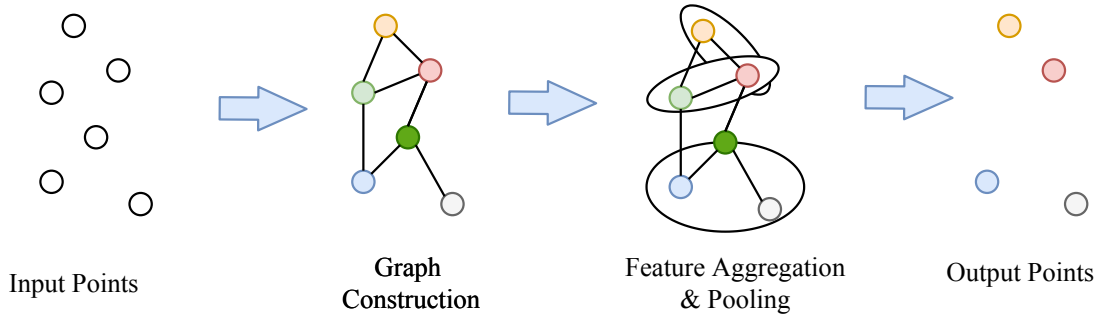


Figure 2.4: **An illustration of a graph-based network.**

Wang et al. [15] proposes EdgeConv operation to extract local features in a graph. Bi et al. [43] uses GNNs for 3D object classification. Guo et al. [44] proposes PCT (Point Cloud Transformer) framework. They try to use the permutation invariant properties from [18] for point cloud processing. Zhang et al. [45] propose a two stage network for 3D object detection. Their point cloud completion module recovers high-quality proposals of dense points to preserve entire views of original structures. Then a graph neural network module is used to capture relations among points. Lie et al. [46] have introduced spherical convolutions for efficient graph based learning. Zhang

et al. [47] propose a 3D point-based scene graph generation (SGGpoint) framework to effectively achieve scene understanding via three sequential stages, namely scene graph construction, reasoning, and inference. They introduce an edge-oriented Graph Convolutional Network (EdgeGCN) to exploit multi-dimensional edge features between neighbors. Fu et al. [48] develop a module based on deep graph matching to calculate the local geometry of each point, its structure and topology. Wang et al. [49] proposes a novel graph attention convolution (GAC), whose kernels can be dynamically carved into specific shapes to adapt to the structure of an object. It assigns attentional weights to different neighboring points. Grid-GCN [50] uses a novel data structuring strategy, Coverage-Aware Grid Query (CAGQ). By leveraging the efficiency of grid space, CAGQ improves spatial coverage while reducing the theoretical time complexity. ASAP-Net [51] further improves spatio-temporal point cloud feature learning with a flexible module called ASAP considering both attention and structure information across frames. RandLA-Net [42] introduced a novel local feature aggregation module to progressively increase the receptive field for each 3D point, thereby effectively preserving geometric details for semantic segmentation. Liu et al. [52] proposed a Dynamic Points Agglomeration Module (DPAM) based on graph convolution to simplify the process of points agglomeration (sampling, grouping and pooling) into a simple step, which is implemented through multiplication of the agglomeration matrix and points feature matrix. Based on the PointNet architecture, a hierarchical learning architecture is constructed by stacking multiple DPAMs. Compared to the hierarchy strategy of PointNet++, DPAM dynamically exploits the relation of points and agglomerates points in a semantic space.

EdgeConv EdgeConv extracts semantic features from point cloud by iteratively performing convolution on a dynamically updated neighborhood. This work extends on the PointNet architecture. EdgeConv addresses the same probelm that pointnet++ tried to solve, but instead of working on individual points it exploits the geometric structure by constructing a local neighborhood graph and applying convolution-like operation on the edge connecting the neighborhood pair of points. Instead of using farthest point sampling, EdgeConv uses k Nearest neighbor algorithm to construct a fully connected graph. EdgeConv appealing property is that it incorporates local neighborhood information as it can be stacked or recurrently applied to learn global shape properties. EdgeConv captures local geometric structure while maintaining permutation invariance. It generates edge features that describe the relationship between a point and its neighbors instead of generating points' feature directly from embedding. As visualized in Figure 2.5, Edgeconv applies channel wise symmetric aggregation function operation

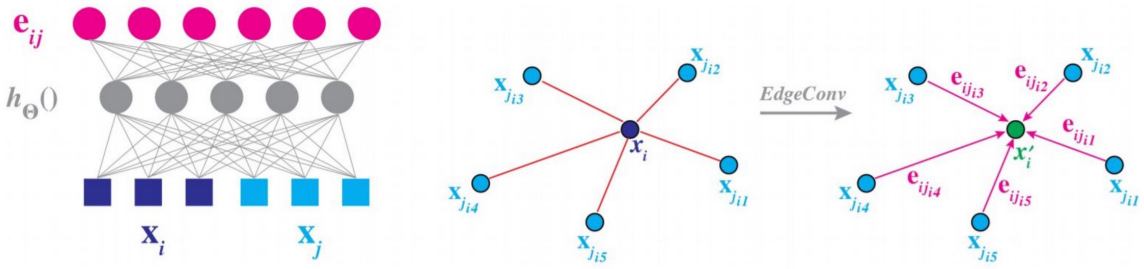


Figure 2.5: **EdgeConv Operation:** The output of EdgeConv is calculated by aggregating the edge features associated with edges from each connecting vertex.

$$\mathbf{x}'_i = \sum_{j \in \mathcal{N}(i)} h_{\mathbf{\Theta}}(\mathbf{x}_i \,\|\, \mathbf{x}_j - \mathbf{x}_i) \tag{2.1}$$

Here $h_{\mathbf{\Theta}}$ denotes a neural network, i.e. a MLP.

2.3.3 Graph Attention based Methods

One of the common problems in most of point based and projection based methods for point cloud processing is the fast growth of point sets size [42] and geometric information loss [53]. To alleviate these problems, attention mechanism is introduced to make neural networks to focus on the important parts of input data, helping to simplified point clouds and capture sufficient feature representations. Qingyong et al. [42] combine Local Spatial Encoding and Attentive Pooling modules to automatically learn important local feature. Tu et al. [54] present an online attention base spatial and temporal feature fusion method for high precision and real-time semantic segmentation. Liang et al. [55] introduce a graph neural network based on attention mechanism which can aggregate geometric and embedding information from neighbours.

GACNet Wang et al. [49] introduces GACNet (Graph Attention Convolution for Point Cloud Semantic Segmentation) for point cloud semantic segmentation. GACNet presents a new graph attention convolution (GAC) method in which the kernels can be dynamically carved into specific shapes to adapt to an object's structure. GAC carefully focuses on the most relevant component of each surrounding point by assigning correct attentional weights to them based on their dynamically learnt properties. The learnt distribution of attentional weights determines the form

of the convolution kernel. GAC can capture point cloud structured features for fine-grained segmentation while avoiding feature contamination across objects. The overview of structure is explained in Figure 2.6.

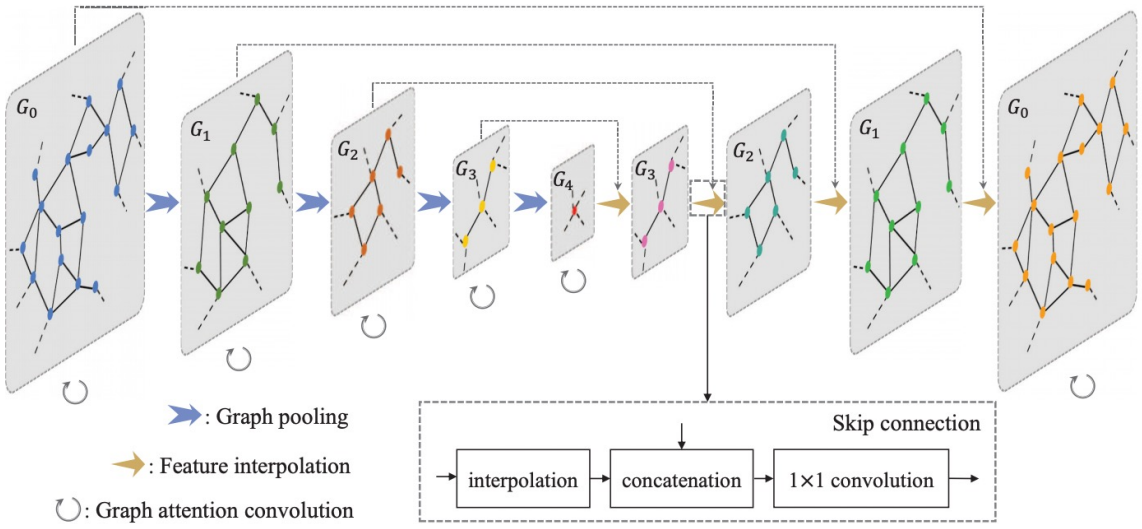


Figure 2.6: **GACNet architecture**. GACNet is constructed on the graph pyramid of a point cloud. On each scale of the graph pyramid, the GAC is applied for local feature learning, followed by the graph pooling for resolution reducing in each feature channel. After that, the learned features are interpolated back to the finest scale layer by layer for point-wise label assignment

A limitation of current graph based methods is that feature learning is biased towards local receptive fields. Receptive fields are defined portion of space or spatial construct containing units that provide input to a set of units within a corresponding layer. Given the nature of aggregation based methods, global information does not pass through each neuron of the model. Our architecture for classification and segmentation uses global features to overcome this limitation of graph based methods, so that along with local features model can also focus on global features.

Chapter 3

GAT3D - Graph Attention Network for 3D Object Detection

In this chapter, we discuss the proposed architecture for 3D object detection called GAT3D. We will also explain our new downsampling technique for point cloud scans, which improves the computational performance of our architecture. The outline of this chapter is as follows:

1. **Distance Aware Downsampling**: Section 3.1 explains our distance aware downsampling technique. We cover the algorithmic performance of our technique in Section 3.4.4.4.
2. **Attention based Aggregation**: In Section 3.2 we explain the structure and aggregation process of our proposed architecture. In section 3.2.2, we provide the definition of loss functions used in our architecture.
3. **Implementation Details**: In Section 3.3, we discuss the technical details of our implementation. We also discuss the data augmentation techniques in and model training details in this subsection.
4. **Experiments**: In Section 3.4, we discuss the qualitative and quantitative performance and comparison of our approach with other state-of-the-art methods.

3D object detection is a fundamental challenge for automated driving and robotic navigation systems. A good detection system must work in real time to match demands of real life driving scenario. To meet the demands of real time processing we propose a distance aware downsampling, that enhances the algorithmic speed of our algorithm.

3.1 Distance Aware Down Sampling

A point cloud P consists of N points in D dimensions such that $P = \{p_i | i = 1, 2, 3....n\} \in \mathbb{R}^D$, where a point p_i is a vector consisting of its coordinates (x, y, z) values, and state values, i.e., reflectance values or encoded features of neighbourhood vertices. A single point cloud scan in autonomous scenarios commonly comprises tens of thousands of points. It is computationally exorbitant to construct a graph with such a large number of points. Along with proposed architectures, we also introduce a distance-aware downsampling scheme to downsample points P without losing the relevant information in the original point cloud scan. A simple voxel downsampling uses a regular voxel grid to create a uniformly downsampled point cloud from an input point cloud. The objects located near the centre of scan have dense construction whereas the objects located far from centre are poorly defined. As apparent in Figure 3.1, the points located far from the ego vehicle are not well defined in a scan. We employ variable voxel sizes depending on the location of the objects from the origin. The points that are located far from the origin uses a smaller voxel size so that the downsampled point cloud do not lose geometrical information from original point cloud scans, since smaller voxel size tends to downsample less number of points than bigger voxel. We have discussed in-depth analysis and qualitative comparison of our approach in experiments section. We have visualized the output scan produced by our technique in Figure 3.1 along with the visualization of original scan and uniformly downsampled scan.

The downsampled point cloud P_D is used to construct a k-nearest neighbour graph $G = \{(V, E)\}$, where $V = \{p_1, p_2, p_3...p_N\}$ are the points and E consists of edges between point p_i to its neighbour vertices within a fixed radius.

3.2 Attention based Aggregation

Let $S = s_1, s_2, s_3, ...s_N \in \mathbb{R}^N$ be a set of input features, associated with vertex $u \in V$, at $t-1$ layer. A single iteration of a Graph Neural Network (GNN) aggregates features from k nodes in a neighbourhood $N(u)$ of a given node u such that the updated feature s of vertex u at layer t is given by:

$$s_u^t = \sigma(W_t \phi_{uv}^{t-1}(s_v^{t-1}) \otimes B_t s_u^{t-1}), v \in N(u) \tag{3.1}$$

Here, W_t and B_t are trainable hyper parameters and σ is the activation function (we use LeakyReLU) to introduce non-linearity. The $\otimes$ here refers to the concatenation

(a) (b) (c)

Figure 3.1: This figure visualize the comparison between (a) original point cloud scan, (b) uniform down sampled scan and (c) a downsampled scan, using our downsampled technique.

function. The function ϕ_{uv}^{t-1} aggregates features along the edges. ϕ could be weighted average of neighbours, element wise mean or max. It can also be an LSTM layer. This function updates feature and repeats the process in every iteration. Unlike [56], [57], GATs [17] leverages self node features and neighbour features to train a model. Inspired by the same idea, we propose an attention based aggregation method to refine neighbourhood vertex states using weights. The proposed method can handle unordered point cloud sets and size-fluctuating neighbour relationships. The proposed architecture has been visualized in Figure 3.2. Let α be the weighting factor (importance) of node v's message to node u. In a standard GNN, $\alpha = \frac{1}{|N(u)|}$. In GAT, α is computed as the by-product of an attention mechanism a, which computes the attention coefficients e_{uv} across all the pairs of u in V and $v \in N(u)$. Therefore, e_{uv} in GAT is defined as in Equation 3.2.

$$e_{uv}^{t} = a(W_t s_u^{t-1} \otimes W_t s_v^{t-1}), v \in N(u) \tag{3.2}$$

We present coordinate difference in a single representation x_{uv}. To capture the local structure of objects and dynamically adapt the weights of edges to the similar neighbours, we define e_{uv} as:

$$e_{uv} = a(\delta x_{uv}, \delta s_{uv}), v \in N(u) \tag{3.3}$$

where $\delta x_{uv} = x_v$ - x_u, the difference between relative coordinates and $\delta s_{uv} = s_v$ - s_u is the difference between the features of every node. For initial layer, s is the input

features and for consecutive layers, it is the learned features. The relative coordinate difference between vertices learns the spatial relationship between u and neighbour v. The feature difference between vertices pairs, assigns more weight to the similar neighbours. a is the feature mapping function i.e MLP (Multi Layer Perceptron) here. Both these terms are concatenated and implemented using a multi-layer neural network. Therefore we rewrite e_{uv} as:

$$e_{uv} = MLP(\delta x_{uv} \otimes \delta s_{uv}) \tag{3.4}$$

After handling the different sized vertices from the neighbourhood of u, we normalize the e_{uv} coefficients using a softmax function to compare the importance of vertices across different neighbours and calculate α_{uv} such that

$$\alpha_{uv}^{t} = \frac{\exp(e_{uv}^{t})}{\sum_{v \in N(u)} \exp(e_{uv}^{t})} \tag{3.5}$$

where α_{uv}^{t} is the attentional weight of vertex v to vertex u at the t^{th} iteration. Therefore, we formulate one iteration of our GNN as:

$$s_{u}^{t} = \sigma(\sum_{v \in N(u)} \alpha_{uv}^{t} W_{t} s_{v}^{t-1}) + s_{u}^{t-1} \tag{3.6}$$

We use this final vertex feature to predict both the class and the oriented bounding box of the object. Contrary to various methods [16, 15] that consider only the relationship between the coordinates of two vertices, we also consider the feature difference along with relative coordinates to give higher weightages to similar vertices during feature aggregation.

3.2.1 Relationship with Prior Methods

After the introduction of *EdgeConv* operator by Wang et. al. [15], research community started exploring graph neural networks as a new technique to improve on feature learning for point cloud processing. Our method has been inspired from DGCNN [15], Graph Attention Networks [58] and GACNet [49]. However, our method has its unique characteristics that makes it different from prior proposed methods. Both [49] and [15] work on point cloud processing and generates labels for classification and segmentation, whereas GAT3D generates point labels along with bounding boxes for object detection. Our attention aggregation method shares some similarities with GACNet. GACNet utilizes farthest point sampling to sample n

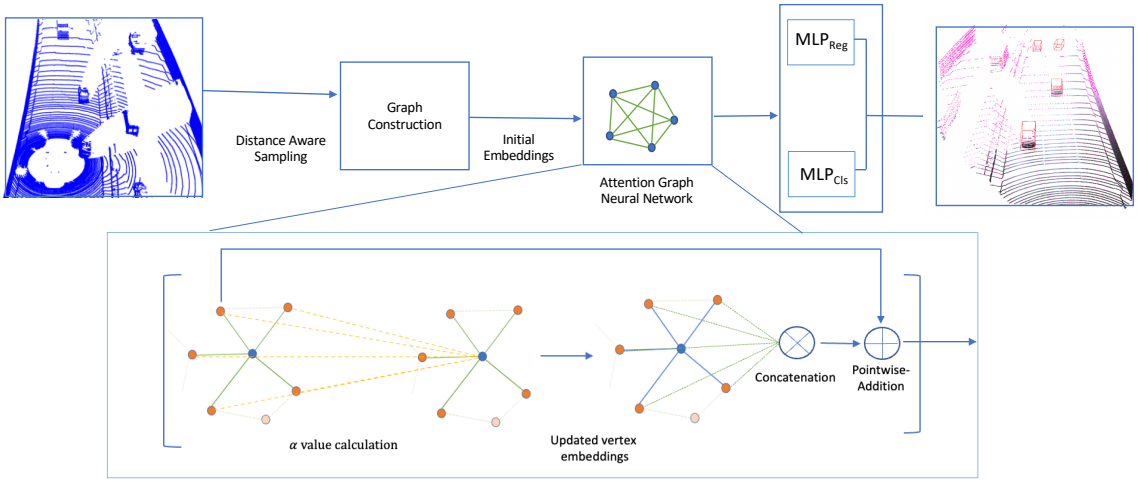


Figure 3.2: The architecture of GAT3D approach. The blocks in square brackets constitute various steps in one iteration of proposed GNN.

points from the point cloud scans, whereas we utilize our novel distance aware downsampling to downsample the point cloud scans. We generate static graphs before feature propagation, whereas GACNet generates graphs and uses graph pooling at every level of their feature propagation process. The other big difference is the architecture design of GACNet and GAT3D. Since GACNet targets point cloud segmentation, it uses a U-Net [59] based pyramid like structure. GAT3D uses hierarchical top-down architecture for 3D detection. The other significant difference between both the architectures is use of graph pooling and skip layers. Since GACNet is designed in a pyramid kind of architecture, it utilizes pooling layers to match the size of different layer levels. GAT3D doesn't use any such skip connection or pooling layers.

3.2.2 Loss

Our final loss function is composed of three main components, a classification loss, a regression loss and a localization loss.

For regression loss λ_{reg}, we parametrize a 3D ground truth value of a bounding box in seven degrees-of-freedom, such that $b_{gt} = (x_{gt}, y_{gt}, z_{gt}, l_{gt}, w_{gt}, h_{gt}, \theta_{gt})$. Similarly the prior anchor box coordinates are encoded as $b_a = (x_a, y_a, z_a, l_a, w_a, h_a, \theta_a)$. Therefore, the residual difference between the predicted bounding boxes and ground truth boxes is given by:

$$\delta x = \frac{x_{gt} - x_a}{\delta d}, \delta y = \frac{y_{gt} - y_a}{\delta d}, \delta z = \frac{z_{gt} - z_a}{\delta d}$$
$$\delta l = \log(\frac{l_{gt}}{l_a}), \delta w = \log(\frac{w_{gt}}{w_a}), \delta h = \log(\frac{h_{gt}}{h_a})$$
$$\delta\theta = \sin(\theta_{gt} - \theta_a) \quad (3.7)$$

where d_a is $\sqrt{(w_a)^2 + (l_a)^2}$. For classification loss λ_{cls}, we use average cross entropy loss which is given by:

$$\lambda_{cls} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_l^i \log(p_l^i) \quad (3.8)$$

where y_l^i is the class label and p_l^i is the predicted probability.

Similar to [16], we use Huber loss to localize objects belonging to a class we are predicting. All the irrelevant classes are localized as background classes. After that, we average the localization loss of all relevant class objects. The localization loss λ_{loc} is given by:

$$\lambda_{loc} = -\frac{1}{N}\sum_{i=1}^{N}(v \in b_a) \sum_{\lambda \in \lambda_{gt}} \lambda_{huber}(\lambda b_a - \lambda b_{gt}) \quad (3.9)$$

Therefore the total loss λ_{total} is given by:

$$\lambda_{total} = \alpha\lambda_{reg} + \beta\lambda_{cls} + \gamma\lambda_{loc} \quad (3.10)$$

The weighting parameters α, β and γ are used to adjust the relative weights of each loss.

$$\rho_{\mathrm{t}} = \begin{cases} \rho_{\mathrm{t}} & \text{if } p = 1 \\ \rho_{\mathrm{t}} - 1 & \text{if } p = -1 \end{cases} \quad (3.11)$$

where p can be calculated with p = sigmoid(x). The binary cross entropy (BCE) loss can be formulated as:

$$\epsilon_{\mathrm{BCE}}(\rho_{\mathrm{t}}) = -\log(\rho_{\mathrm{t}}) \quad (3.12)$$

As stated in these equations, when the network is trained with BCE loss, its gradient will be dominated by vast easily classified negative samples. if a huge foreground-background imbalance exists. Focal loss can be considered as a dynamically scaled cross entropy loss, which is defined as:

$$\epsilon_{\mathrm{FL}}(\rho_{\mathrm{t}}) = -(1 - \rho_{\mathrm{t}})^{\gamma}\log(\rho_{\mathrm{t}}) \quad (3.13)$$

Therefore the contribution from the well classified samples to the loss is down-weighed. The hyper parameter of the focal loss can be used to tune the weight of different samples. As γ increases, fewer easily classified samples contribute to the training loss. When reaches 0, the focal loss degrades to become the BCE loss. We have discussed and demonstrated the performance of our architecture in experiments section.

3.3 Implementation Details

We have developed our architecture in Python 3.6. We use Tensorflow 1.5 [60] for designing the model architecture and training scripts. For point cloud processing and downsampling we use Open3D library [61]. For other mathematical computations and calculations we use Sklearn [62], Numpy [63] libraries. In following sections we have discussed specimens of our implementation.

Data Preprocessing Due to the presence of variable environment scenes in KITTI dataset [12], data augmentation is crucial to get better results. We applied data augmentation techniques to prevent over-fitting and make predictions robust. We individually process all the ground truth boxes. Each box is uniformly rotated in [$-\pi/20$, $\pi/20$] and translated along x, y, and z axes independently from (0, 0.25) to further enrich the training set. Our distance-aware voxel downsampling induces vertex jitter during the graph construction.

Training Details We train the proposed network end-to-end with a batch size of 2. The loss weighting parameters α, β, and γ are used to balance the relative importance of different parts and their value is set to α= 0.1, β= 10, γ= 0.0005. The β is given the highest value so that model can perform better on classification task. For car, we use an initial learning rate of 0.125 and a decay rate of 0.1 every 400K steps. We trained the network for 1400K steps. For pedestrian, we used a learning rate of 0.25 and a decay rate of 0.25 every 400K steps and trained for 1000K steps. Similarly for cyclist, we trained for 1000K steps with a learning rate of 0.32 and a decay rate of 0.25 after every 400K steps. For cars, we use the anchor box size of (1.6, 3.9, 1.5) meters covering width, length and height respectively with two rotations 0 and 90 degrees. We set r to 1.8 m. The anchor box sizes for pedestrian and cyclist objects were set to (0.6, 0.8, 1.73) meters and (0.6, 1.76, 1.73) meters respectively. We used variable voxel sizes during the sampling, i.e., we used voxel sizes of 0.5 and 0.8 for points greater than 40 m and less than 20 m respectively

from the sensor along the z axis. To reduce redundancy, we apply IoU threshold of 0.7 for NMS for car category and 0.6 threshold for pedestrian and cyclist category. The network was trained in an end-to-end manner on a single TITAN V GPU. We employed the ADAM optimizer to train our network.

3.4 Experiments

This section covers the various qualitative and quantitative experiments that we performed on our architecture. We have compared our architecture with various state of art methods also. The results are covered and documented below.

We evaluate our architecture on the widely used KITTI object detection benchmark[64] which contains 7481 training samples and 7518 testing samples. Each sample has a point cloud scan, a respective image and calibration data. Since the dataset only annotates objects that are visible within the image, we process the point cloud only within the field of view of the image. Due to the scale differences (different point density and size), we trained the network separately on car, pedestrian and cyclist data.

3.4.1 Evaluation Metrics for Object Detection

The section discusses the evaluation metrics that will be used to evaluate this architecture on KITTI dataset.

Bounding Box Classification For a bounding box to be considered a correct detection, the area of overlap a_o between the predicted bounding box B_p and ground truth bounding box B_{gt} must exceed a certain threshold. A common threshold for 2D is 0.7 (70%), obtained by the formula :

$$IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{3.14}$$

This formula was first introduced in VOC object detection challenge [65] . This formula can be extended to 3D to measure how well the 3D boxes overlap by considering the depth axis also.

Precision Recall For a given task and class, the precision-recall curve is computed from a method's ranked output. Recall is defined as the proportion of all positive

examples ranked above a given rank. Precision is the proportion of all examples above the rank which are from the positive class

$$Precision = \frac{TP}{TP + FP} \tag{3.15}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.16}$$

where TP is acronym for true positive, which indicates a correct detection with existing corresponding ground-truth. FP is false positive and indicates that an object was detected but it does not have a corresponding ground-truth, i.e. false detection. FN is false negative, and indicates that an object was in the ground-truth but not detected by the

Average Precision The Average Precision (AP) summarizes the shape of the precision-recall curve. In KITTI dataset benchmark is defined as the mean precision at a set of 11 equally spaced recall thresholds such as:

$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, ...1\}} p_{interp}(r) \tag{3.17}$$

The precision at every level r is calculated by considering the maximum precision measured for which the corresponding recall exceeds r, such that

$$p_{interp}(r) = max(p(\hat{r})) \tag{3.18}$$

where $p(\hat{r})$ is the measured precision at recall $\hat{r}$

3.4.2 Qualitative Results

We illustrate our prediction results on KITTI test data in Figures 3.3-3.4. We visualize 3D bounding boxes in LiDAR scan and 2D bounding boxes on RGB images. From the figures, we can observe that the proposed architecture can estimate accurate 3D bounding boxes in a variety of scenes. The architecture can predict the correct positions in poor lighting conditions and occlusions. As shown in Figure 3.4, one can see that the model is capable of predicting the pedestrian positions even when they are not clearly visible in RGB images.

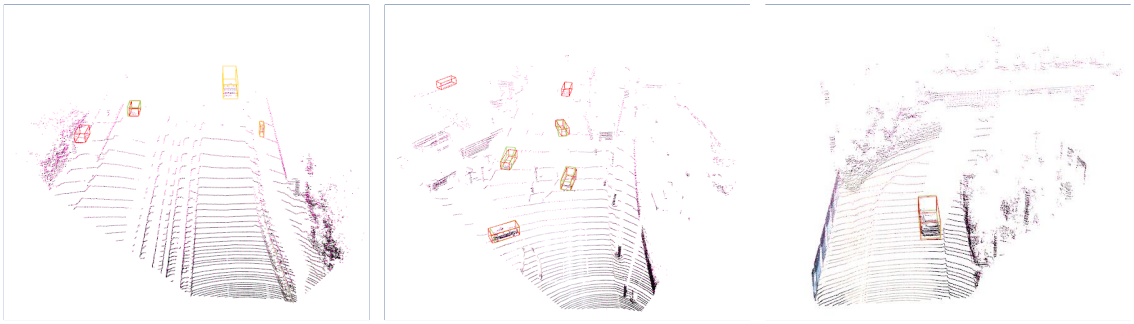


Figure 3.3: Qualitative 3D detection results of our architecture on the KITTI validation set. The detected objects are shown with red 3D bounding boxes and green 3D bounding boxes reflect the ground truth bounding boxes.

3.4.3 Quantitative Comparison

We validate the proposed approach using a set of experiments carried out on the KITTI object detection benchmark. The KITTI dataset evaluates the average precision (AP) on three difficulty levels: easy, moderate, and hard. We have presented the performance of our method in Table 3.1.The 3D and BEV detection results obtained by our proposed approach are comparable to the ones provided by the other state-of-the-art methods. Our approach detects all three classes in KITTI dataset reasonably well. We have compared the accuracy and inference speed of our method with [7, 66, 5], [16, 67, 55, 68, 69]. We have presented the results in a scatter plot in Figure 3.5. The proposed model outperforms fusion based MV3D in car category, and AVOD in pedestrian category.

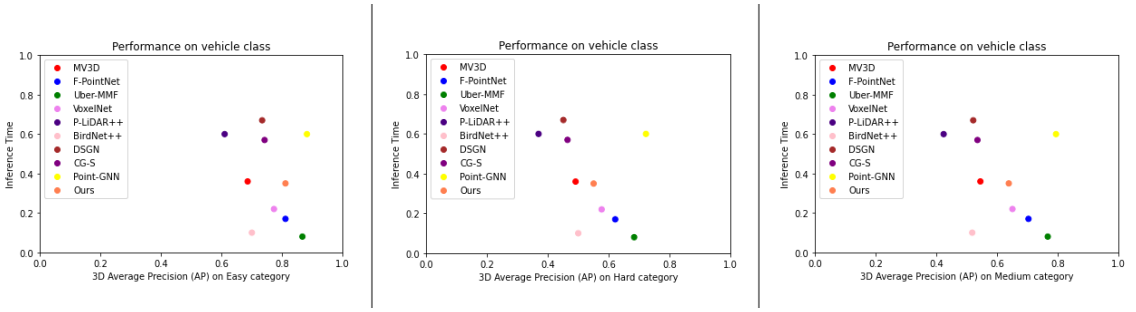


Figure 3.5: Comparison of our method across different 3D detection methods in Vehicle category

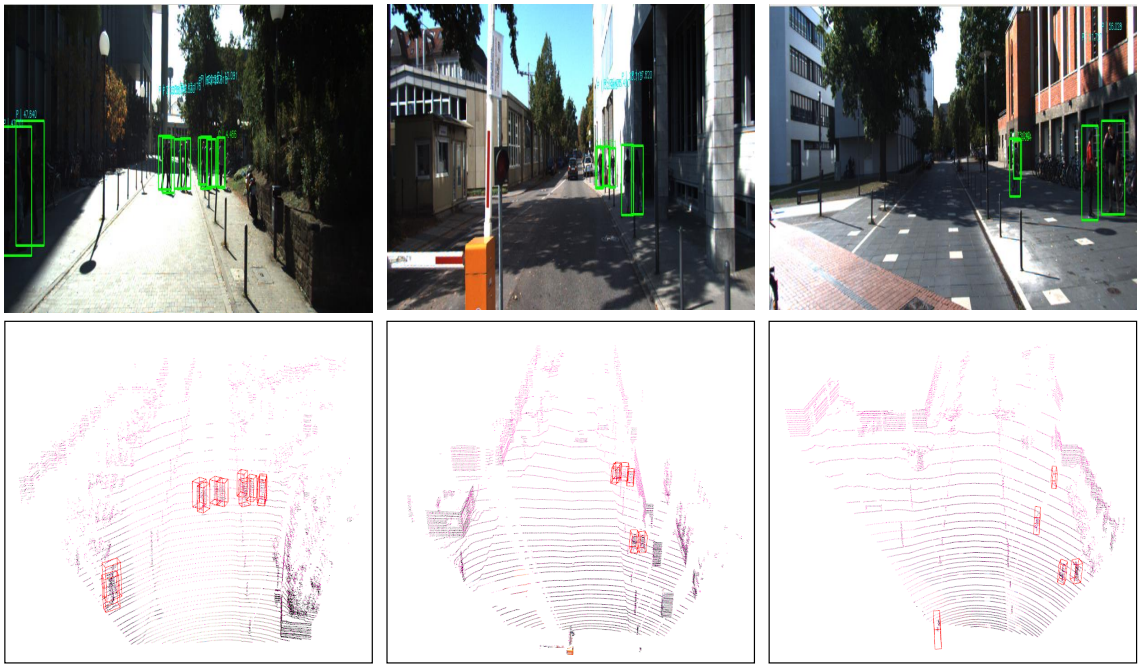


Figure 3.4: Qualitative 3D detection results of our architecture on the Pedestrian category of KITTI test set. The detected objects are shown with red 3D bounding boxes and green 2D bounding boxes. The upper row shows the 2D bounding box in the RGB images and the bottom row shows the results in the corresponding point clouds.

	Vehicle			Pedestrian			Cyclist		
	Easy	Medium	Hard	Easy	Medium	Hard	Easy	Medium	Hard
3D AP %	75.67	63.90	55.09	43.62	34.56	31.34	58.44	41.81	36.69
BEV AP%	87.95	80.65	70.97	48.26	39.41	35.90	66.56	47.80	41.82

Table 3.1: The average precision(AP) result on both 3D and Bird's eye view on KITTI test dataset.

3.4.4 Ablation Studies

3.4.4.1 Results on KITTI Validation Dataset

We test our method on the car category of KITTI validation dataset and compare our results with state-of-art methods. The performance of our detection system on

KITTI validation datset is presented in Table 3.2. For car category, the proposed method achieves results comparable to state-of-the-art methods on all the difficulty levels of the KITTI dataset. We visualize our results on KITTI validation split (refer to Figure 3.3).

Method	Vehicle 3D AP%			Vehicle BEV AP%		
	Easy	Medium	Hard	Easy	Medium	Hard
MV3D	71.29	62.68	56.65	86.55	78.10	76.67
AVOD	84.41	74.44	68.65	<na>	<na>	<na>
F-PointNet	83.76	70.92	63.95	88.16	84.02	76.44
DSGN	72.31	54.27	47.71	83.24	63.91	57.83
VoxelNet	81.97	65.46	62.85	89.60	84.81	78.57
PointGNN	87.89	78.34	77.38	89.82	88.31	87.16
Ours	83.54	74.47	63.84	90.12	87.05	75.48

Table 3.2: Average precision (AP) comparison of both 3D and Bird Eye View on KITTI Validation dataset.

3.4.4.2 Effects of Different Number of Layers

In our architecture, we stack n number of GNN layers to extract aggregated features. To demonstrate the influence of changing the value of n, we train our network with n varying from 1 to 4. We demonstrate our results in Table 3.3. Table 3.3 indicates there's a slight increase in the accuracy when n is changed from 1 to 3, which can be attributed to the fact that the neighborhood features are being aggregated to the vertex itself. Our model performance continues to increase as we increase the value of n. There is a slight decrease in accuracy at $n = 4$, which indicates that our neural network might be over-trained.

Number of layers (n)	Vehicle (3D AP%)		
	Easy	Moderate	Hard
1	80.24	72.27	62.78
2	82.73	73.65	63.14
3	83.54	74.47	63.84
4	83.14	74.21	63.46

Table 3.3: The 3D Average Precision (AP) comparison when changing the number of layers of our proposed Network.

3.4.4.3 Inference Time Analysis

For any algorithm to be deployed in autonomous vehicle scenario, the inference speed plays a crucial role. The algorithm must be able to predict the oncoming object's position in real-time. The performance of an algorithm is subject to the hardware and code-optimization. Our architecture is written in Python, and implemented in Tensorflow for GPU computation. We measure our inference time on a machine with Intel i7-8700k CPU, 32GB RAM and Nvidia Titan V GPU. The dataset reading and preprocessing takes 45 ms. The nearest neighbor graph construction consumes 112 ms. A single iteration of our GNN takes 270 ms and it takes 15 ms for final bounding box predictions.

3.4.4.4 Downsampling

Our downsampling technique significantly reduces the complexity of graph construction significantly. Our technique is able to reduce the total number of points to 25% without loosing relevant information in our scans. Compared to uniform downsampling , our technique is able to keep more points without taking significant time for graph construction. We compare the graph construction time on a sample from KITTI dataset in the Table 3.4. In our experiments, we found that our distance aware downsampling provides a significant accuracy enhancement when compared to the uniform downsampling. Our experimental results are presented in Table 3.5.

Technique	Points	Time (ms)
None	115094	362 ms
Uniform Downsampling	20753	110ms
Distance Aware Downsampling	22826	112ms

Table 3.4: The graph construction time comparison on one sample in KITTI dataset

Techniques	Accuracy (3D AP%)		
	Easy	Medium	Hard
Uniform	75.34	62.32	53.54
Distance-Aware	75.67	63.90	55.09

Table 3.5: We compare results of our downsampling technique with uniform downsampling on Vehicle category of KITTI dataset

Chapter 4

GAGAT - Global Aware Graph Attention Network for 3D Segmentation and Classification

In this chapter, we discuss our architecture to learn better local representations for unstructured point cloud in the context of shape classification and segmentation tasks. As an addition to GAT3D architecture, we propose GAGAT(Global Aware Graph Attention Network) for 3D pointcloud processing. GAGAT mainly consists of three parts, a self attention module, local attention module and global feature module. In detail, the self-attention module is used to learn self-geometric features for every single point, while local-attention module focuses on local geometric relationships in neighbourhood of the point. Along with focusing on local features, global features are also taken into account for learning. We learn relevant global feature embeddings using shared MLP layers, that symmetrically learn features over whole point cloud. We concatenate these features with the output of dynamically learned local features from our local and self attention layer. This approach creates a layerwise representation for point cloud points consisting of both global and local features.

4.1 Global Aware GAT Layer

A sample point cloud scan captured from real life scenarios (e.g., autonomous vehicle and geospatial scans) consists of a large number of points/samples. Processing every individual point leads to a high computation cost and causes vanishing gradient problem because only small weights are allocated to every point. As a result, we first sample points from point cloud scan using farthest point sampling technique [10].

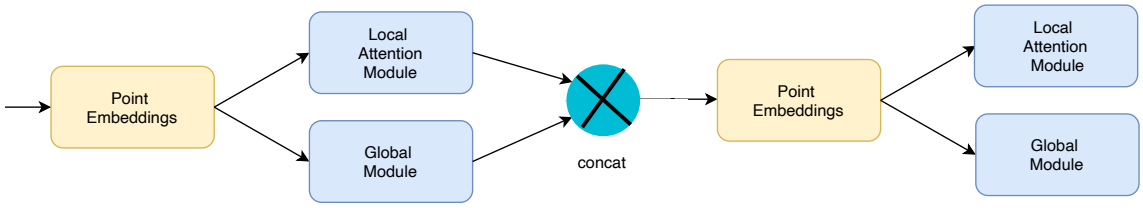


Figure 4.1: This figure describes how information propagates in our network between two global aware GAT layers. Every single feature embedding is passed through two separate networks focusing on local features and contextual features.

Contrary to autonomous vehicles, the point cloud scans are captured in a stationary setting with a fixed field of view, hence farthest point sampling represents a better approach in this context. We construct a k-nearest neighbour graph $G = (V, E)$ from these sample points to represent the geometric structure of the point cloud, here $V = \{p_1, p_2, p_3.....p_n\}$ are nodes for points, $E \subseteq V \times N_i$ are edges connecting neighbourhood pairs of points, and N_i is a neighbourhood set of point p_i. Like GAT3D, we define the edge feature as geometric difference between nodes, i.e., $x_{ij} = (p_i - p_j)$ where $p_i \in V, p_j \in N_i$, and x_{ij} indicates the coordinates difference between of neighbouring point p_j to point p_i. We pass the sampled points parallelly to shared global module and local attention module.

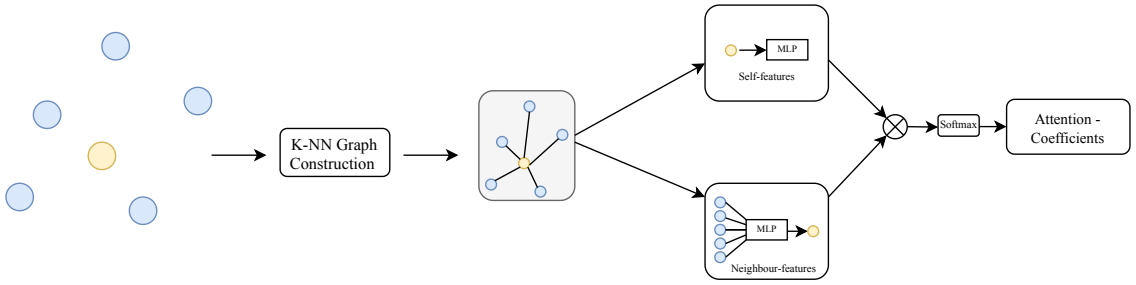


Figure 4.2: Local attention module of a global aware GAT layer.

Local Attention Module We construct a k-nearest neighbour graph $G = (V, E)$ from these sample points to represent the geometric structure of the point cloud, here $V = \{p_1, p_2, p_3.....p_n\}$ are nodes for points, $E \subseteq V \times N_i$ are edges connecting neighbourhood pairs of points, and N_i is the neighbourhood set of point p_i. We define the edge feature as geometric difference between nodes, i.e., $x_{ij} = (p_i - p_j)$ where $p_i \in V, p_j \in N_i$, and x_{ij} indicates the coordinates difference between neighbouring

point p_j to point p_i. In order to pay different attentions to different neighbouring points, we propose a dual attention mechanism using self and local attentions (see Figure 4.2). This combined attention mechanism assigns attention coefficients to each point in neighbourhood of the query point. We first map input features to a higher-level feature representation by using a non-linear mapping function M.

$$x_i^{'} = M(p_i, \theta) \tag{4.1}$$

$$x_{ij}^{'} = M(x_{ij}, \theta) \tag{4.2}$$

Here, the mapping function M, is selected as a multi layer perceptron (MLP) with θ as learnable hyper-parameters.

We combine self attention and local features, to obtain a combined feature embedding c_{ij} as shown in Equation 4.3.

$$c_{ij} = (x_i^{'} \otimes x_{ij}^{'}) \tag{4.3}$$

To align comparison of the attention coefficients across neighbours for different points, we use softmax function to normalize coefficients for all the neighbours to every point as follows (Equation 4.4).

$$\alpha_{ij} = \frac{\exp(c_{ij})}{\sum_{k \in (1,.....|N_i|)} \exp(c_{ik})} \tag{4.4}$$

Therefore, we formulate one iteration of our local attention module as below (Equation 4.5).

$$\hat{x}_{ij}loc = \sigma(\alpha_{ij} x_{ij}^{'}) \tag{4.5}$$

where σ is non-linear activation function. We used sigmoid as activation function here.

Global Module: As mentioned above, we pass the sampled point cloud parallelly through local attention module and global module. For sampled points $P_D = \{p_1, p_2.....p_n\}$, where $p_i \in \mathbb{R}^D$, D dimensional feature vector consists of input features (RGB value, reflectance value etc.) or transformed feature embeddings. Inspired from [8], we use a k-shared MLP to apply feature transformation on each point, such that

$$M(P_D) = h(p_1), h(p_2).....h(p_n) \tag{4.6}$$

In this equation, $h : \mathbb{R}^D \rightarrow \mathbb{R}^M$ which is realized using a shared MLP layer. Thus we can compile output of our global module as follows (Equation 4.7).

$$\hat{x}_{ij}glob = \sigma(M(P_D)) \tag{4.7}$$

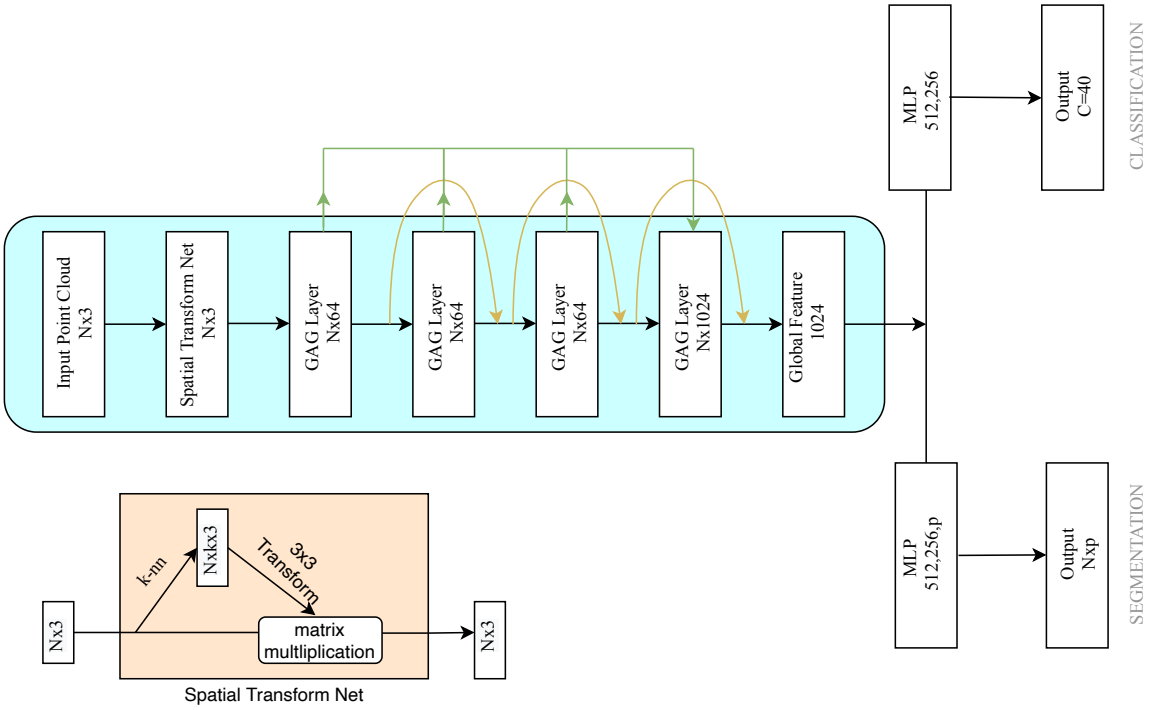


Figure 4.3: The architecture for classification and segmentation. The top branch illustrates the structure of our classification and the segmentation network. To make point clouds invariant to transformation, we use spatial transform net [70].

After obtaining the local features and global features, we concatenate both modules and pass them to the next layers as input to get the feature embedding to predict point wise segmentation score and classification score. So one iteration of a GAG layer can be defined as in Equation 4.8.

$$\hat{x}_{ij} = \sigma(\hat{x}_{ij}glob \otimes \hat{x}_{ij}loc) \tag{4.8}$$

where $\otimes$ represents a concatenation function.

4.2 Segmentation and Classification

Our architecture shown in Figure 4.3 captures both shape classification and segmentation for point cloud. Our model architecture is inspired from *Pointnet* [8]. The main difference between our architecture and Pointnet is that, contrary to processing every point features we exploit local features using graph based feature aggregation

along with masked attention. To leverage global features, we we use shared MLP layers point embeddings. We have illustrated the structure of both our segmentation and classification architecture in Figure 4.3. The classification and segmentation share a common architecture except the use of different MLP output units in the last layer. The green arrows refer that information from every level are added to the last GAG layer. The curved brown arrows refer to skip connections between different layers. The classification model generates a probability score across n classes, whereas the segmentation architecture generates segmentation score for single point.

4.3 Relationship with Prior Works

Our proposed architecture GAGAT shares similarities with several works proposed prior. Mainly our architecture share common characteristics with Pointnet [8], GACnet [49] and DGCNN [15]. The biggest similarity between all these methods is the idea of aggregating neighbourhood features. Although we inspire some ideas from above mentioned methods, our method is different with respect to the following points.

1. Compared to GACNet [49] architecture, we do not follow a pyramid structure but simple hierarchical top-down architecture for feature learning. GACNet takes into account the spatial relationship (i.e, geometrical difference) and feature difference between two vertices, we consider geometric difference and self features of each vertex and concatenate that with global information. GACNet does not use global information in their feature propagation.
2. DGCNN [15] is one of the first methods to use idea of graph convolutions on point cloud processing. Compared to DGCNN, we do not assign equal importance to every vertices in neighbourhood, but use an attention mechanism to assign different importance to different vertices. Given, the nature of convolution filters, DGCNN also suffers from the limitation of local receptive fields. We try to improve on this limitation by using global information during our feature propagation process.

4.4 Loss

The classification can be considered as a special case of classification. In our architecture, we use a modified version of cross entropy loss called softmax cross entropy

loss. Cross entropy loss [60] has been regarded as one of the common loss functions used for classification tasks. We have explained cross entropy loss in Section 3.2.2.

SCE loss measures the probability error in discrete classification tasks in which the classes are mutually exclusive (each entry is in exactly one class). For segmentation, we use the sparse version of the same loss function. The reason behind which is that we can assign discreetly different labels to points, so that a point only belongs to a single class.

4.5 Implementation Details

Similar to our previous architecture, we use Tensorflow 1.5 [60] for designing the model architecture and training scripts. For point cloud processing and down-sampling, we use Open3D library [61]. For other mathematical computations and calculations, we use Sklearn [62]and Numpy [63] libraries.

Data Preprocessing We evaluate and train our model on the ModelNet40 [14] for classification task. The dataset contains 12,311 meshed CAD models from 40 categories. Following the official and mostly used convention we use 9843 models training and 2468 models for testing. For classification, we follow similar data augmentation setting as of [8]. For each model, we uniformly sample 1,024 points from the mesh faces; the point cloud is the rescaled to fit into a unit sphere. During the training procedure, we perform data augmentation by randomly scaling objects and interchanging the object and point locations.

For segmentation, we train and test our architecture on three datasets Shapenet [13], S3DIS (Stanford Large-Scale 3D Indoor Spaces Dataset) [71] and Semantic3D [72]. For part segmentation, we follow the same train/test split as [13]. For indoor scene segmentation, similar to Wang et al. [8], we split each scan into blocks with area 1m $\times$ 1m, and each point is represented as a 9D vector (XYZ, RGB, and normalized spatial coordinates). We sample 4,096 points for each block during training process, and all points are used for testing.

4.6 Experiments

4.6.1 Classification

We test our model's performance on Modelnet 40 dataset. Modelnet consists of 40 man made models in different types and orientations. We divide the dataset in 9483 samples for training and 2468 samples for testing. We use the same data split for our experiments and report the results on the test set. Our model is able to produce good results when compared to SOTA methods. We obtained 89.6 % in overall accuracy and 90.6 % in mean accuracy.

4.6.1.1 Qualitative Comparison

We have provided the label predictions of our network on Modelnet dataset in Figure 4.4. As visualized, our model is able to predict the correct labels of objects irrespective of their different types and shapes. The proposed GAGAT model scores a commending overall accuracy of 90.6 %.

4.6.1.2 Quantitative Comparison

We evaluated and listed our model comparison in Table 4.1 on two famously used metrics i.e Overall accuracy (OA) and Mean accuracy (mAcc). Overall accuracy measures total number of correctly predicted items divided by total number of items to predict. Mean accuracy calculates the average accuracy per class. Our model performs reasonably well in mAcc metrics. Our model does fairly well on overall accuracy metrics and managed to surpass the overall accuracy of Pointnet [8], PointGCN [39] and SO-Net [73]. Our model falls behind PointConv [74], SpiderCNN [75] and DGCNN [15] by a small margin. One reason for it could be that both PointConv and SpiderCNN use the normals information along with specific refinement modules to improve on final predictions. KPConv [76] uses a deformable kernel that can change it's shape based on the local structure of points. This technique gives it an advantage on classification task. Our model is able to score a score of 89.6% in mean accuracy.

4.6.2 Part Segmentation

For part segmentation purpose, we use the Shapenet dataset [13]. The main objective of part segmentation is to segment a point cloud scan into sub parts (ex. legs and arms of chair, wings of air plane etc.). The shapenet dataset consists of 16,881

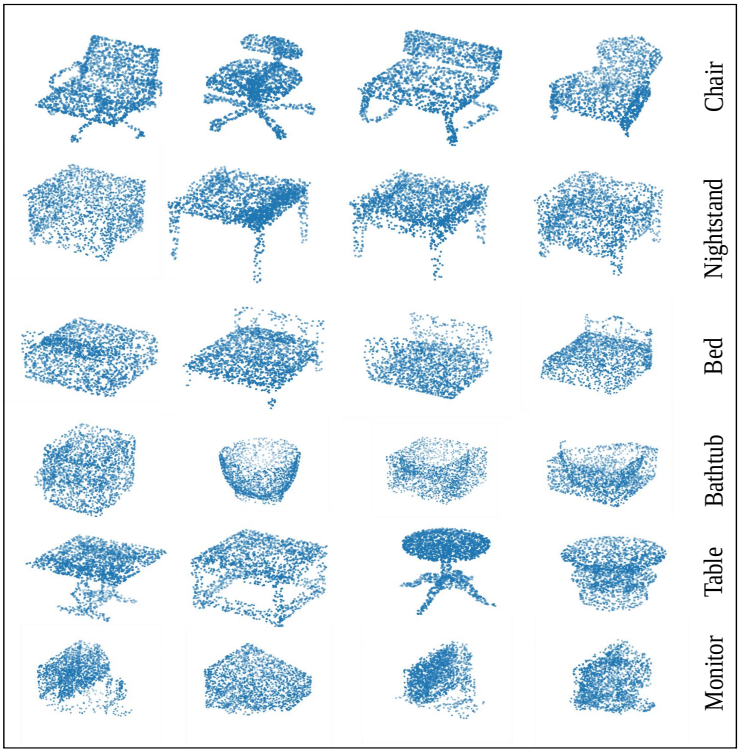


Figure 4.4: Visualization of various objects in Modelnet [14] dataset and predicted labels by our model.

3D models of 16 object categories with 50 part segmentation ground truths. The evaluation metric that we use for this task is the mean intersection over union (mIoU) 3.4.1 for all the shapes in a particular category. We use the official train/val/test split for consistency with other results.

4.6.2.1 Qualitative Segmentation

We visualize our results from selected categories of Shapenet in Figure 4.5. For visualization, we use the tool developed by Xu Yan [78]. We illustrate different segmented parts with different colours. The quantitative results show that our model accurately segments various parts of different models. Our model works very well on easy categories, i.e., laptop, headphone, guitar and table. In our visualizations, our model shows reasonable results on difficult classes (motorcycle, vehicle and airplane) too.

Method	Input	Overall Accuracy (OA)	Mean Accuracy (mAcc)
Pointnet [8]	Coordinates	89.2%	86.2%
Pointnet++ [10]	Coordinates	90.7%	-
PointConv [74]	Coordinates+Normal	92.5%	-
SpiderCNN [75]	Coordinates+Normal	92.4%	-
KPConv [76]	Coordinates	**92.9**%	
DGCNN [15]	Coordinates	92.2%	**90.2**%
PointGCN [39]	Coordinates	89.5%	86.1%
SO-Net [73]	Coordinates	90.9%	87.3%
3DmFV-Net [77]	Coordinates	91.6%	-
Ours	Coordinates	90.6%	89.6%

Table 4.1: Comparison of our method on Modelnet-40 Dataset

4.6.2.2 Quantitative Comparison

In this paragraph, we present a qualitative analysis for the predictions obtained by the 3D part segmentation on Shapenet dataset. Similar to classification, our model performed very well for the segmentation task. We achieved a score of 84.2% of average mIOU. The model produced the highest score of 92.8% in Mug class. The lowest mIOU score is 60.5% in Rocket class. We have compared our method with popular part segmentation methods. For this analysis, we have compared our method with Pointnet [8], Pointnet++ [10], KD-Net [79], 3D-GCN [80], KPConv [76], PointCNN [9], DGCNN [15] and Spider CNN [75]. Our architecture performs fairly well in comparison with other methods as demonstrated in Table 4.3. We have also presented class wise mIOU in Table 4.2. We manage to surpass the performance of [8], [10], [79], [80], and [75] by a huge margin. Our model lies behind [9], [15] and [76] in terms of accuracy. Similar to classification, one reason behind it could be that use of dynamic filters by [9] and [76], thus giving a edge to these methods.

Method	Mean	Aero	Bag	Cap	Car	Chair	Headphone	Guitar	Knife	Lamp	Laptop	Motorcycle	Mug	Pistol	Rocket	Skateboard	Table
PointNet	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
PointCNN	**86.1**	**84.1**	**86.4**	86.0	**80.8**	90.6	**79.7**	**92.3**	88.4	**85.3**	**96.1**	**77.2**	**95.3**	**84.2**	**64.2**	**80.0**	**83.0**
DGCNN	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
SpiderCNN	85.3	83.5	81.0	**87.2**	77.5	**90.7**	76.8	91.1	87.3	83.3	95.8	70.2	93.5	82.7	59.7	75.8	82.1
Ours	84.2	82.9	81.2	86.8	76.3	89.7	75.3	91.3	**88.7**	82.1	95.6	69.4	92.8	79.8	60.5	76.2	82.7

Table 4.2: Classwise Accuracy on ShapeNet [13] Dataset.

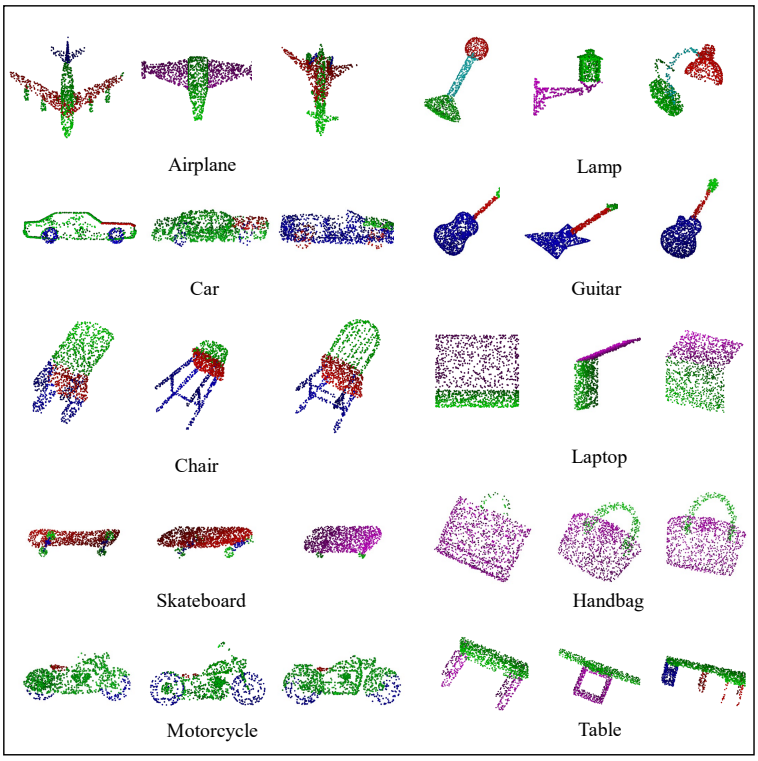


Figure 4.5: Visualizations of the part segmentations by GAGAT on various models from Shapenet [13].

4.6.3 Indoor Scene Segmentation

For indoor scene segmentation, we train our model on S3DIS (Stanford 3-Dimensional Indoor Scene) [71] dataset. Our this experiment shows how our segmentation architecture generalizes to real indoor scenes. S3DIS covers six large-scale indoor areas from three different buildings for a total of 273 million points annotated with 13 classes. Following common research convention [76], we use the point cloud scans from Area-5 as test scene to better measure the generalization ability of our method.

4.6.3.1 Qualitative Comparison

We present a qualitative analysis for the predictions obtained by the 3D segmentation architecture on S3DIS dataset. We have selected and demonstrated various scenes in Figures 4.6. In Figure 4.6(a), our model well captures the chair shapes and tables, however, fails to capture fine objects such as board on the back wall. The model is

Method	Class Average IoU
PointNet	80.4
PointNet++	81.9
KD-Net	77.4
3D-GCN	82.1
KPConv	**86.4**
PointCNN	86.14
DGCNN	85.2
Spider CNN	82.4
Ours	84.2

Table 4.3: Comparison of our method on ShapeNet [13] dataset.

able to segment the floor and wall correctly in Figure 4.6(b). The model struggles a little with the fine area of door casing, but for the majority part of door casing it successfully differentiates it from the wall. In Figure 4.6(c) segments most of the objects in scene accurately. The model predicts the labels and areas for most of the objects correctly in Figure 4.6(d) except a clutter and a few parts of the floor.

4.6.3.2 Quantitative Comparison

Our method works well on all the scenes available in S3DIS dataset. We compare the performance of our method with other methods. We report the results in 4.4. In our comparison, model gives an acceptable accuracy when compared to other methods. Our model is able to score better accuracy then both [8] and [10]. The accuracy is slightly below [76], [52], [81] and [49]. The KPConv [76] is able to score highest score in this category.

4.6.4 Outdoor Scene Segmentation

The Semantic3D [72] dataset is a repository of with over 4 billion points from a variety of urban and rural scenes. In this dataset, along with geometric information every point has RGB and intensity values. The dataset has been labelled with one of 8 categories: man-made terrain, natural terrain, high vegetation, low vegetation, buildings, hard scape, scanning artefacts, and cars. To adapt to the size of objects, we follow the same data preparation technique by [49]. Since Semantic3D consists of a relative scan of large outdoor buildings and environments, we sample 4096 points from 4×4 boxes across whole scan.

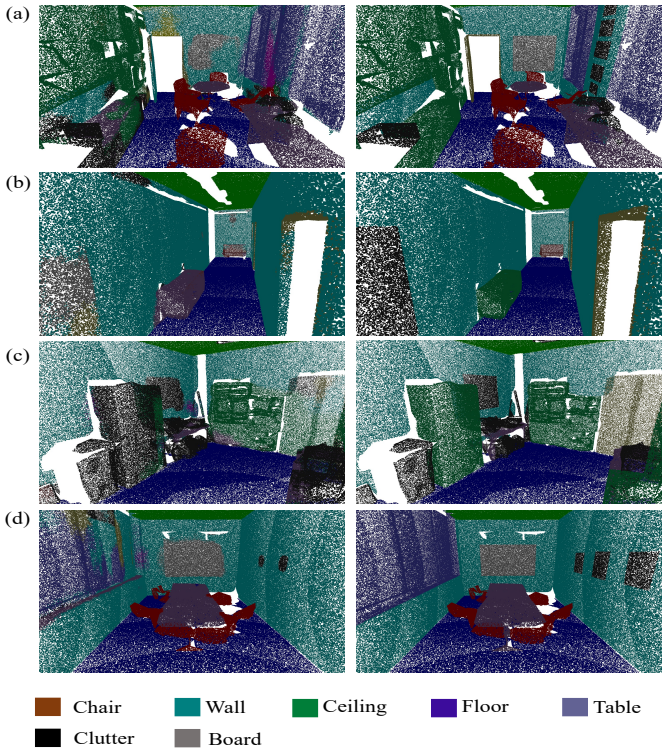


Figure 4.6: Indoor Segmentation results of Room 5 in S3DIS dataset. **Left:** segmentations by GAGAT and **Right:** the ground truths.

We state the overall accuracy and mean IOU of our architecture compared to other state of art methods.

4.6.4.1 Qualitative Comparison

In this section, we present a qualitative analysis for the predictions obtained by the 3D segmentation architecture on Semantic3D dataset.The Semantic3D dataset point clouds have been segmented into 9 categories, i.e man-made terrain, natural terrain, high vegetation,low vegetation, buildings, hard scape, scanning artefacts, cars. Our method produces an acceptable score of 88.6 % on overall accuracy and 63.4% on mean accuracy.

We have demonstrated our results in Figure 4.7(a)-4.7(c). From these figures, it is quite evident that in most of the cases our model is able to segment different parts of scenes correctly.Similar to previous examples, our model performs well on this scan

Method	Mean IOU	Overall Accuracy
PointNet [8]	41.1	-
PointNet++ [10]	53.4	-
PointCNN [9]	57.3	85.9
KPConv [76]	**67.1**	-
GACNet [49]	62.9	**87.8**
SPH3D-GCN [81]	59.5	87.7
DPAM [52]	60.0	86.1
Ours	58.5	86.2

Table 4.4: Comparative semantic segmentation results on the S3DIS (comparison on Area5).

too. We have produced the visualization using CloudCompare [82].

4.6.4.2 Quantitative Comparison

We present the quantitative comparison of our method in this section. Our performance is on par with other competitive methods on Semantic3D dataset. We present the overall accuracy (OA) and mIOU of our method and compare it's performance in Table 4.5. In our comparison, our model doesn't perform very well on Semantic 3D large scale scenes, when compared to other methods. We need to do an in-depth analysis to find the reason behind this. In comparison to our method, Randla-Net [42] performs the best out of all the methods mentioned, along with FGNet [83]. One reason for such a excellent results by both these methods could be because is that they have been specifically designed to process large scale outdoor data.

Method	Mean IOU	Overall Accuracy
SnapNet [84]	-	41.09
GACNet [49]	70.8	91.9
RANDLA-Net [42]	**77.4**	**94.8**
FG-Net [83]	77.2	94.4
KPConv [76]	74.6	92.6
SPG [85]	62.85	87.79
Ours	63.4	87.65

Table 4.5: Comparative outdoor segmentation results on the Semantic3D (reduced-8 challenge).

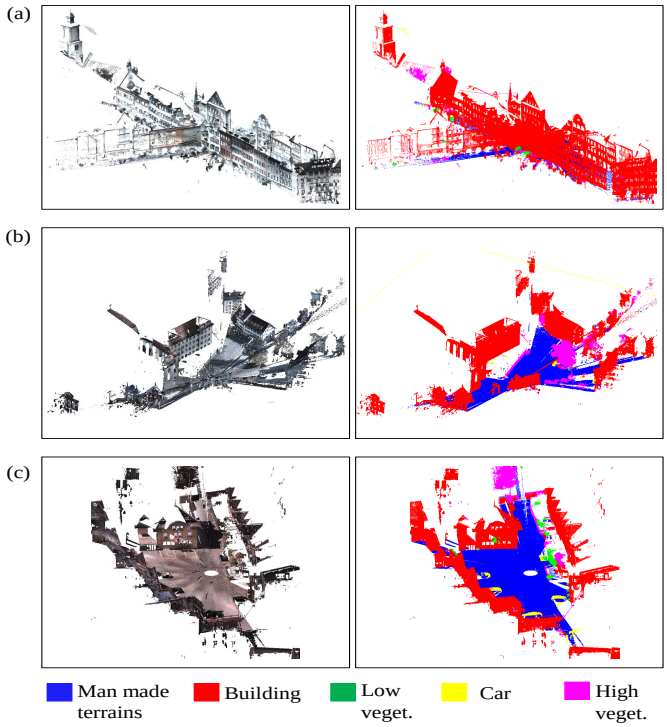


Figure 4.7: Examples of outdoor segmentation by GAGAT on Semantic3D datasets. (a) Cathedral in St. Gallen (b). Market square and (c). Town square.

4.7 Ablation Study and Differences between GAT3D and GAGAT

There are some differences between GAT3D and GAGAT architectures, we have listed all the differences below.

1. **Applications:** GAT3D model has been designed to output oriented bounding boxes along the object classes whereas GAGAT works on both classification and segmentation labels.

2. **Absence of global shared MLP:** The GAT3D architecture doesn't uses a shared MLP layer to capture global information. We tested our model using global shared branch on 3D detection. In our experiments we found that using a shared MLPs on GAT3D affects the overall accuracy of model by a small margin. We have compared accuracies in Table 4.6.

Technique	3D Average Precision (AP)		
	Easy	Medium	Hard
Using global MLP	81.42	73.24	59.14
Without using global MLP	83.54	74.47	63.83

Table 4.6: We compare accuracies of GAT3D architecture with and without shared global MLP on KITTI validation dataset on car category.

Efficiency of Global Aware Module: To test the efficiency of our global aware module, we train our model with and without global module on Modelnet40 dataset. We test the efficiency of our global module and compare results of both models in Table 4.7. In our experiments we found that using global module in our feature

Technique	Mean Accuracy(%)
Without global module	90.2%
With global module	90.6 %

Table 4.7: Ablation tests on the architecture with and without using global module. Results are the values for the overall accuracy on the ModelNet40 dataset.

aggregation process gives us an improvement of 0.4 % accuracy. Even without using global module our model is able to score a good accuracy score of 90.2 % on classification task. Although this increase in overall accuracy comes with increased computational complexity and inference speeds. The use of global module is chosen so that final predictions are not aligned alone towards the local features and increase in the overall performance of model.

Chapter 5
Conclusions and Future Direction

5.1 Conclusion

In this thesis, we presented an attention based feature aggregation method for 3D object detection and a global aware attention network for 3D segmentation and classification. Both the architectures have potential applications in robot perception systems, AR/VR systems and terrain classification.

5.1.1 GAT3D - Graph Attention Network for 3D Object Detection

We propose GAT3D architecture, an attention-based neighbour feature aggregation technique for detecting objects in point cloud scan. Along with attention based feature aggregation method, we use distance aware downsampling to enhance the algorithmic performance of our model. For distance aware downsampling, we employ variable voxel sizes for downsampling depending on the distance of point from the origin. We trained and tested our model on KITTI dataset. In our experiments, we also demonstrate the benefit of our distance aware downsampling technique that not only enhances the performance of our algorithm but also increases the accuracy compared to uniform down sampling technique. We were able to achieve good results on the vehicle class under easy and medium categories. Similar to majority of the other methods, our model is only able to achieve moderate results on the pedestrian class. The reason behind which is the less number of training instances belonging to pedestrians class in KITTI dataset in comparison to vehicle class. Also the point density of pedestrian scan is very sparse in comparison to other classes. We are able to achieve acceptable results on cyclist class. As demonstrated in Figure 3.3, we show that our model is able to present a good trade-off between accuracy and inference speed. In comparison to other graph representation based methods like PointGNN

[16], our model lies behind in terms of accuracy. PointGNN uses auto-registration and improved NMS refinement techniques to achieve these results. The addition of similar kind of refinement modules and techniques should help our model to achieve better accuracy results as well.

5.1.2 GAGAT - Global Aware Graph Attention Network for Classification and Segmentation

We design GAGAT architecture for point cloud classification and segmentation. GAGAT can be utilized to learn relevant local representations for unstructured point cloud for shape classification and part segmentation tasks. As an addition to GAT3D architecture, we use a self-attention mechanism along with a neighbouring-attention mechanism to capture attention coefficients for a point to its neighbourhood. To capture contextual information, we employ global feature module. We trained and tested the presented architecture on Modelnet dataset for classification, Shapenet for part segmentation, S3DIS (Stanford 3D Indoor Scenes) dataset for indoor segmentation and Semantic3D for outdoor segmentation. As demonstrated in Table 4.1 the architecture performed fairly well on classification task and scores 90.6% in overall accuracy. The presented architecture demonstrates similar performance on part segmentation task as presented in Table 4.3. The model performs fairly well on all the classes and is also able to surpass all other mentioned methods in knife class as shown in Table 4.2. We also test the architecture on complicated indoor scenes in S3DIS dataset as well. As visualized in Figure 4.6, GAGAT is able to handle semantic labels of complicated scenes fairly. Lastly, this architecture is able to score moderate results on outdoor scene segmentation task as shown in Table 4.5. In our experiments we also presented the advantage of the global aware module in Table 4.7. Most importantly, the new global aware module aids in avoiding the bias of graph based feature learning towards local receptive fields. We show that our architecture can show acceptable results across distinctive applications and scenarios. Our model struggles to surpass the accuracies of some state-of-the-arts methods. GAGAT lacks the use of targetted techniques to achieve high accuracies across different applications contrary to mentioned methods. For example graph pyramid structure used by GACNet [49] for better segmentation results and point feature encoding augmentation in RandLANet [42] for better large scene segmentation. We believe stacking refinement modules and using additional techniques to target specific applications should give an accuracy boost to our architecture as well.

In conclusion, we have presented an alternate approach of graph feature aggregation

using masked attention and graph representation of point cloud data in this thesis. Our experiments show good results across different datasets and applications. We present this work as a common architecture for various applications on point cloud data. Our architecture can be joined with refinement modules to target a specific application. Finally, our proposed architectures produce results in low inference times, which can be a great advantage for autonomous vehicles and robots.

5.2 Future Directions

A potential future direction would be to explore the possibility of using multi-head attention and weighted coordinate difference to improve the detection results. There exists further scope for improving the current prediction results and code-optimization to improve better inference speeds. The optimization of current architecture for geospatial data or aerial data is also one direction that may be explored. We would also look into the direction of addition of refinement modules or techniques targetting specific tasks. We would also explore different graph representation (i.e Delaunay Graphs) for point cloud data.

Bibliography

[1] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. 'Imagenet classification with deep convolutional neural networks'. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[2] Shaoqing Ren et al. 'Faster r-cnn: Towards real-time object detection with region proposal networks'. In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[3] Wei Liu et al. 'Ssd: Single shot multibox detector'. In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[4] Steve Lawrence et al. 'Face recognition: A convolutional neural-network approach'. In: *IEEE transactions on neural networks* 8.1 (1997), pp. 98–113.

[5] Yin Zhou and Oncel 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.

[6] Yan Yan, Yuxing Mao and Bo Li. 'Second: Sparsely embedded convolutional detection'. In: *Sensors* 18.10 (2018), p. 3337.

[7] Alejandro Barrera et al. *BirdNet+: End-to-End 3D Object Detection in LiDAR Bird's Eye View*. 2020. URL: `http://arxiv.org/abs/2003.04188`.

[8] Charles R Qi et al. 'Pointnet: Deep learning on point sets for 3d classification and segmentation'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660.

[9] Yangyan Li et al. 'Pointcnn: Convolution on x-transformed points'. In: *Advances in neural information processing systems* 31 (2018), pp. 820–830.

[10] Charles Ruizhongtai Qi et al. 'Pointnet++: Deep hierarchical feature learning on point sets in a metric space'. In: *Advances in neural information processing systems*. 2017, pp. 5099–5108.

[11] Artyom Makovetskii et al. 'Affine registration of point clouds based on point-to-plane approach'. In: *Procedia Engineering* 201 (2017), pp. 322–330.

[12] Andreas Geiger et al. 'Vision meets robotics: The KITTI dataset'. In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1231–1237.

[13] Angel X Chang et al. 'Shapenet: An information-rich 3d model repository'. In: *arXiv preprint arXiv:1512.03012* (2015).

[14] Zhirong Wu et al. '3d shapenets: A deep representation for volumetric shapes'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920.

[15] Yue Wang et al. 'Dynamic graph cnn for learning on point clouds'. In: *Acm Transactions On Graphics (tog)* 38.5 (2019), pp. 1–12.

[16] Weijing Shi, Ragunathan and Rajkumar. *Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud*. 2020. arXiv: 2003.01251 [cs.CV].

[17] Petar Veličković et al. 'Graph attention networks'. In: *arXiv preprint arXiv:1710.10903* (2017).

[18] Ashish Vaswani et al. 'Attention is all you need'. In: *arXiv preprint arXiv:1706.03762* (2017).

[19] DGL Team. *Graph attention network*. 2020. URL: https://docs.dgl.ai/en/0.4.x/tutorials/models/1_gnn/9_gat.html.

[20] Xiaozhi Chen et al. '3d object proposals for accurate object class detection'. In: *Advances in Neural Information Processing Systems*. 2015, pp. 424–432.

[21] Xinshuo Weng and Kris Kitani. 'Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud'. In: *arXiv preprint arXiv:1903.09847* (2019).

[22] Bo Li. '3d fully convolutional network for vehicle detection in point cloud'. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1513–1518.

[23] Bo Li, Tianlei Zhang and Tian Xia. 'Vehicle Detection from 3D Lidar Using Fully Convolutional Network'. In: *CoRR* abs/1608.07916 (2016). arXiv: 1608.07916. URL: http://arxiv.org/abs/1608.07916.

[24] Alex H. Lang et al. 'PointPillars: Fast Encoders for Object Detection from Point Clouds'. In: *CoRR* abs/1812.05784 (2018). arXiv: 1812.05784. URL: http://arxiv.org/abs/1812.05784.

[25] Zhe Liu et al. 'TANet: Robust 3D Object Detection from Point Clouds with Triple Attention'. In: *arXiv preprint arXiv:1912.05163* (2019).

[26] Zetong Yang et al. '3DSSD: Point-based 3D Single Stage Object Detector'. In: *arXiv preprint arXiv:2002.10187* (2020).

[27] Shaoshuai Shi et al. 'Pv-rcnn: Point-voxel feature set abstraction for 3d object detection'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10529–10538.

[28] Zhichao Li, Feng Wang and Naiyan Wang. 'LiDAR R-CNN: An Efficient and Universal 3D Object Detector'. In: *arXiv preprint arXiv:2103.15297* (2021).

[29] Jintai Chen et al. 'A Hierarchical Graph Network for 3D Object Detection on Point Clouds'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 392–401.

[30] Jesus Zarzar, Silvio Giancola and Bernard Ghanem. *PointRGCN: Graph Convolution Networks for 3D Vehicles Detection Refinement*. 2019. arXiv: 1911.12236 [cs.CV].

[31] Daniel Maturana and Sebastian Scherer. 'Voxnet: A 3d convolutional neural network for real-time object recognition'. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 922–928.

[32] Truc Le and Ye Duan. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018).

[33] Manzil Zaheer et al. 'Deep sets'. In: *arXiv preprint arXiv:1703.06114* (2017).

[34] Mor Joseph-Rivlin, Alon Zvirin and Ron Kimmel. 'Momen (e) t: Flavor the moments in learning to classify shapes'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[35] Xiao Sun, Zhouhui Lian and Jianguo Xiao. 'Srinet: Learning strictly rotation-invariant representations for point cloud classification and segmentation'. In: *Proceedings of the 27th ACM International Conference on Multimedia*. 2019, pp. 980–988.

[36] Mutian Xu et al. 'PAConv: Position Adaptive Convolution with Dynamic Kernel Assembling on Point Clouds'. In: *arXiv preprint arXiv:2103.14635* (2021).

[37] Martin Simonovsky and Nikos Komodakis. 'Dynamic edge-conditioned filters in convolutional neural networks on graphs'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3693–3702.

[38] Radu Bogdan Rusu and Steve Cousins. '3d is here: Point cloud library (pcl)'. In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 1–4.

[39] Yingxue Zhang and Michael 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.

[40] Zhengli Zhai, Xin Zhang and Luyao Yao. 'Multi-Scale Dynamic Graph Convolution Network for Point Clouds Classification'. In: *IEEE Access* 8 (2020), pp. 65591–65598. DOI: 10.1109/ACCESS.2020.2985279.

[41] Francis Engelmann et al. 'Know what your neighbors do: 3D semantic segmentation of point clouds'. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 2018, pp. 0–0.

[42] Qingyong Hu et al. 'Randla-net: Efficient semantic segmentation of large-scale point clouds'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11108–11117.

[43] Yin Bi et al. 'Graph-based object classification for neuromorphic vision sensing'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 491–501.

[44] Meng-Hao Guo et al. 'PCT: Point Cloud Transformer'. In: *arXiv preprint arXiv:2012.09688* (2020).

[45] Yanan Zhang, Di Huang and Yunhong Wang. 'PC-RGNN: Point Cloud Completion and Graph Neural Network for 3D Object Detection'. In: *arXiv preprint arXiv:2012.10412* (2020).

[46] Huan Lei, Naveed Akhtar and Ajmal Mian. 'Spherical kernel for efficient graph convolution on 3d point clouds'. In: *IEEE transactions on pattern analysis and machine intelligence* (2020).

[47] Chaoyi Zhang et al. 'Exploiting Edge-Oriented Reasoning for 3D Point-based Scene Graph Analysis'. In: *arXiv preprint arXiv:2103.05558* (2021).

[48] Kexue Fu et al. 'Robust Point Cloud Registration Framework Based on Deep Graph Matching'. In: *arXiv preprint arXiv:2103.04256* (2021).

[49] Lei Wang et al. 'Graph attention convolution for point cloud semantic segmentation'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10296–10305.

[50] Qiangeng Xu et al. 'Grid-gcn for fast and scalable point cloud learning'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 5661–5670.

[51] Hanwen Cao et al. 'ASAP-Net: Attention and Structure Aware Point Cloud Sequence Segmentation'. In: *arXiv preprint arXiv:2008.05149* (2020).

[52] Jinxian Liu et al. 'Dynamic points agglomeration for hierarchical point sets learning'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 7546–7555.

[53] Can Qin et al. 'Pointdan: A multi-scale 3d domain adaption network for point cloud representation'. In: *arXiv preprint arXiv:1911.02744* (2019).

[54] Xinyuan Tu et al. 'Reconstruction of High-Precision Semantic Map'. In: *Sensors* 20.21 (2020), p. 6264.

[55] Ming Liang* et al. 'Multi-Task Multi-Sensor Fusion for 3D Object Detection'. In: *CVPR*. 2019.

[56] Thomas N Kipf and Max Welling. 'Semi-supervised classification with graph convolutional networks'. In: *arXiv preprint arXiv:1609.02907* (2016).

[57] William L. Hamilton, Rex Ying and Jure Leskovec. 'Inductive Representation Learning on Large Graphs'. In: *NIPS*. 2017.

[58] Petar Veličković et al. 'Graph Attention Networks'. In: *International Conference on Learning Representations* (2018). URL: https://openreview.net/forum?id=rJXMpikCZ.

[59] Olaf Ronneberger, Philipp Fischer and Thomas Brox. 'U-net: Convolutional networks for biomedical image segmentation'. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[60] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[61] Qian-Yi Zhou, Jaesik Park and Vladlen Koltun. 'Open3D: A Modern Library for 3D Data Processing'. In: *arXiv:1801.09847* (2018).

[62] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[63] Charles R. Harris et al. 'Array programming with NumPy'. In: *Nature* 585 (2020), 357–362. DOI: 10.1038/s41586-020-2649-2.

[64] Andreas Geiger, Philip Lenz and Raquel Urtasun. 'Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite'. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[65] Mark Everingham et al. 'The pascal visual object classes (voc) challenge'. In: *International journal of computer vision* 88.2 (2010), pp. 303–338.

[66] Xiaozhi Chen et al. 'Multi-view 3d object detection network for autonomous driving'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1907–1915.

[67] Charles R Qi et al. 'Frustum PointNets for 3D Object Detection from RGB-D Data'. In: *arXiv preprint arXiv:1711.08488* (2017).

[68] Yurong You et al. 'Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving'. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=BJedHRVtPB.

[69] Chengyao Li, Jason Ku and Steven L Waslander. 'Confidence Guided Stereo 3D Object Detection with Split Depth Estimation'. In: *IROS* (2020).

[70] Max Jaderberg et al. 'Spatial transformer networks'. In: *arXiv preprint arXiv:1506.02025* (2015).

[71] I. Armeni et al. 'Joint 2D-3D-Semantic Data for Indoor Scene Understanding'. In: *ArXiv e-prints* (Feb. 2017). arXiv: 1702.01105 [cs.CV].

[72] Timo Hackel et al. 'SEMANTIC3D.NET: A new large-scale point cloud classification benchmark'. In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. IV-1-W1. 2017, pp. 91–98.

[73] Jiaxin Li, Ben M Chen and Gim Hee Lee. 'So-net: Self-organizing network for point cloud analysis'. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9397–9406.

[74] Wenxuan Wu, Zhongang Qi and Li Fuxin. 'Pointconv: Deep convolutional networks on 3d point clouds'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9621–9630.

[75] Yifan Xu et al. 'Spidercnn: Deep learning on point sets with parameterized convolutional filters'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 87–102.

[76] Hugues Thomas et al. 'Kpconv: Flexible and deformable convolution for point clouds'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6411–6420.

[77] Yizhak Ben-Shabat, Michael Lindenbaum and Anath Fischer. '3DmFV: Three-Dimensional Point Cloud Classification in Real-Time Using Convolutional Neural Networks'. In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3145–3152.

[78] Xu Yan. 'Pointnet Pytorch'. In: (2019).

[79] Roman Klokov and Victor Lempitsky. 'Escape from cells: Deep kd-networks for the recognition of 3d point cloud models'. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 863–872.

[80] Zhi-Hao Lin, Sheng-Yu Huang and Yu-Chiang Frank Wang. 'Convolution in the cloud: Learning deformable kernels in 3D graph convolution networks for point cloud analysis'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1800–1809.

[81] Huan Lei, Naveed Akhtar and Ajmal Mian. 'Octree guided cnn with spherical kernels for 3d point clouds'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9631–9640.

[82] Daniel Girardeau-Montaut. 'CloudCompare'. In: *Retrieved from CloudCompare: https://www. danielgm. net/cc* (2016).

[83] Kangcheng Liu et al. 'FG-Net: Fast Large-Scale LiDAR Point CloudsUnderstanding Network Leveraging CorrelatedFeature Mining and Geometric-Aware Modelling'. In: *arXiv preprint arXiv:2012.09439* (2020).

[84] Alexandre Boulch et al. 'SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks'. In: *Computers & Graphics* 71 (2018), pp. 189–198.

[85] Loic Landrieu and Martin Simonovsky. 'Large-scale point cloud semantic segmentation with superpoint graphs'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4558–4567.