Effective Neighborhood Feature Exploitation in Graph CNNs for Point Cloud Object-Part Segmentation

Pranav Megarajan

Publisher: Dean Prof. Dr. Wolfgang Heiden

**Hochschule Bonn-Rhein-Sieg – University of Applied Sciences,
Department of Computer Science**

Sankt Augustin, Germany

January 2022

Technical Report 01-2022

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

ISSN 1869-5272

ISBN 978-3-96043-099-5

Deutsches Zentrum
für Luft- und Raumfahrt
German Aerospace Center

This work was supervised by	Ulrich Hillenbrand Paul G. Plöger Rudolph Triebel

Digital Object Identifier https://doi.org/10.18418/978-3-96043-099-5

Abstract

Part segmentation is the task of semantic segmentation applied on objects and carries a wide range of applications from robotic manipulation to medical imaging. This work deals with the problem of part segmentation on raw, unordered point clouds of 3D objects. While pioneering works on deep learning for point clouds typically ignore taking advantage of local geometric structure around individual points, the subsequent methods proposed to extract features by exploiting local geometry have not yielded significant improvements either. In order to investigate further, a graph convolutional network (GCN) is used in this work in an attempt to increase the effectiveness of such neighborhood feature exploitation approaches. Most of the previous works also focus only on segmenting complete point cloud data. Considering the impracticality of such approaches, taking into consideration the real world scenarios where complete point clouds are scarcely available, this work proposes approaches to deal with partial point cloud segmentation.

In the attempt to better capture neighborhood features, this work proposes a novel method to learn regional part descriptors which guide and refine the segmentation predictions. The proposed approach helps the network achieve state-of-the-art performance of 86.4% mIoU on the ShapeNetPart dataset for methods which do not use any preprocessing techniques or voting strategies. In order to better deal with partial point clouds, this work also proposes new strategies to train and test on partial data. While achieving significant improvements compared to the baseline performance, the problem of partial point cloud segmentation is also viewed through an alternate lens of semantic shape completion.

Semantic shape completion networks not only help deal with partial point cloud segmentation but also enrich the information captured by the system by predicting complete point clouds with corresponding semantic labels for each point. To this end, a new network architecture for semantic shape completion is also proposed based on point completion network (PCN) which takes advantage of a graph convolution based hierarchical decoder for completion as well as segmentation. In addition to predicting complete point clouds, results indicate that the network is capable of reaching within a margin of 5% to the mIoU performance of dedicated segmentation networks for partial point cloud segmentation.

Acknowledgements

I would like to take the chance to pay my special regards to everyone who supported me, by any means, throughout the period of this thesis.

My heartfelt appreciation goes to my supervisor at DLR, Dr. Ulrich Hillenbrand, for his continuous support throughout the period of this thesis. I am extremely grateful for all the wonderful scientific discussions and invaluable inputs received which have enriched this work. I want to especially thank him for believing in me and providing me this great opportunity.

I would like to thank my supervisor Prof. Dr. Paul G. Plöger for his support throughout the period of this thesis and especially for his prompt responses whenever I needed his help. I also would like to thank PD Dr. habil. Rudolph Triebel for accepting my request to be the second supervisor for my thesis.

I want to express my deepest gratitude to my parents, sisters and brothers, who without them I would not be able to accomplish this work. I also wish to give special thanks to all my friends, especially Pradeep, Loganathan, Prashant, Harshit, Pooja Bhat, Arun and Supriya for their support throughout this period.

I also wish to thank my colleagues at DLR, Raja, Shreyas, Pooja Krishnan and Liu Siyuan for the amazing discussions and inputs.

Contents

List of Figures

List of Tables

1

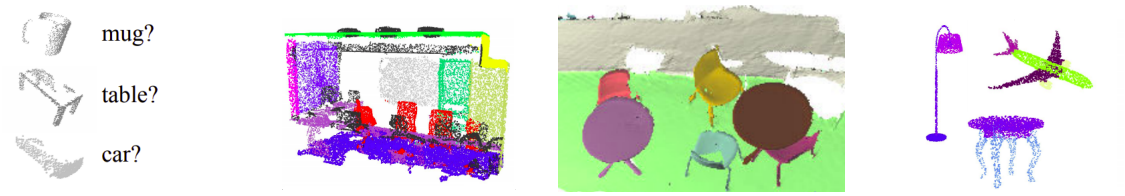
Introduction

With the increasing availability and affordability of 3D data acquisition technologies and the computing capabilities to process the same, there has been a rising interest in extending state-of-the-art 2D data processing techniques to 3D data. Compared to 2D data, 3D data provides rich geometric and shape information. The need to process them to extract meaningful information arises from the fact that they are extremely useful in understanding the surrounding environment and for intelligent decision making for machines. 3D data is useful across a wide spectrum of application areas such as robotics, autonomous driving, engineering design and reconstruction, remote sensing and medical treatment [10].

3D data can be represented in different formats such as point clouds, meshes, volumetric grids and depth images. The raw information that is directly obtained from 3D sensors are mostly in the form of point clouds. Compared to other forms of representation, point clouds are more compact and contain complete information without any loss due to discretization by preprocessing techniques like voxelization or meshing. These characteristics make point clouds a more suitable choice for processing and deployment in machines which are usually constrained by hardware resources.

Deep learning has evolved into the go-to solution for most computer vision problems these days. Neural networks have achieved the state-of-the-art in computer vision problems of relevance to robotics and autonomous systems. These problems typically include detection, classification, semantic and instance segmentation from images and more recently also from 3D point clouds. While CNNs (Convolutional Neural Networks) can be attributed with much of the success on processing images, there is yet an effective equivalent to emerge in the 3D domain. The unstructured and unordered nature of 3D point clouds has posed a major challenge to applying already available techniques. Although there has been a decent progress with deep learning for point clouds understanding since the past couple of years, there is still a lot of scope for improvement.

One approach to extracting meaningful information from point clouds is through semantic segmentation. It helps us with the information of points clustered according to their corresponding semantic classes, modeled as a per-point classification problem. Semantic segmentation is usually applied for scene or object analysis. When applied on whole scenes, it is to segment them into different object classes and when applied on individual objects, it is to segment different parts in the object as seen in Figure 1.1. This work specially focuses on object part segmentation, also widely known as 3D shape segmentation, the problem of semantic segmentation on point clouds of objects or shapes.



(a) Classification (b) Scene Segmentation (c) Instance Segmentation (d) Part Segmentation

Fig. 1.1. This work focuses on one of the common point cloud learning tasks which is known as object-part segmentation or part segmentation or 3D shape segmentation as seen in section (d) of the image adopted from [38]

1.1 Motivation

This thesis was carried out at the Institute of Robotics and Mechatronics of the German Aerospace Center (DLR), which works on autonomous robots, telerobots and humanoids for space, industrial, medical and service applications. In robotics, the information obtained through segmentation can be used by autonomous mobile robots for various decision making processes, sometimes to make a navigation decision by analysing the 3D scene surroundings or to aid in manipulation strategies to handle a certain object in the environment. In short, to navigate or to interact with the environment, semantic segmentation can play a crucial role.

There have been several deep learning based models proposed in recent times to tackle the problem of segmenting point clouds beginning with the pioneering work of PointNet [38]. PointNet only employs a few layers of shared MLPs (Multi-Layer Perceptrons) and predicts the segmentation labels for individual points from the learnt individual point features and a global feature obtained by max-pooling across all points. While it successfully overcomes the problem of permutation invariance caused due to the unordered point clouds, it completely ignores making use of the local geometric structure around individual points.

In order to overcome the problem, there have been different approaches proposed to better capture the local geometry around individual points. Projection based methods using multi-view images [46] or voxelization [32] have tried to extend the application of dominant techniques like 2D and 3D CNNs to better capture neighborhood information. Despite showing some good improvements, these methods suffer from computational complexities and memory issues. Other techniques proposed directly process point cloud data by building neighborhood graphs with the point of interest as the centroid. These techniques typically use graph convolutions [58], point convolutions [51] or specially defined convolution kernels.

1.1.1 Potential Applications

3D shape analysis and segmentation can be useful not only in robotics but also in other domains. Shape segmentation can be useful in robotic manipulation, navigation and also in 3D modeling and analysis.

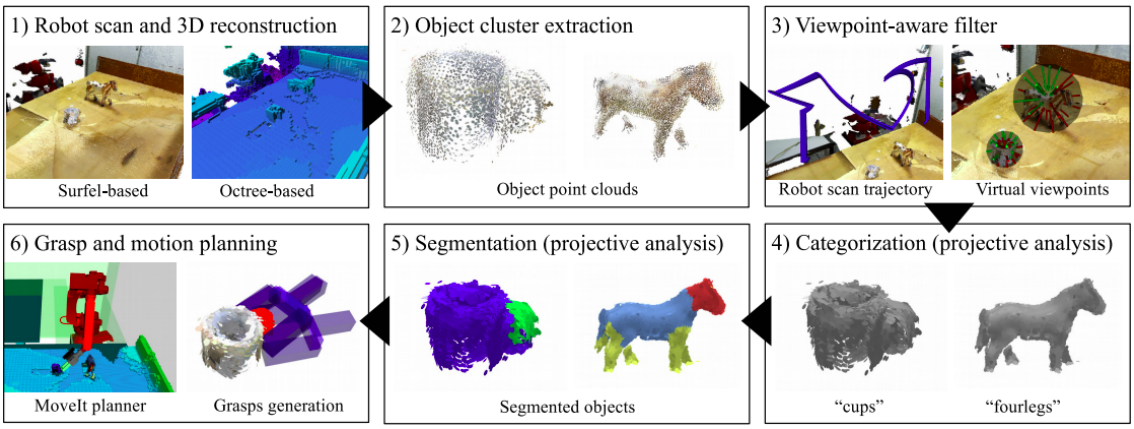


Fig. 1.2. Part-based grasp planning approach as seen in the image adopted from [34].

a) Robotic Tasks: In a common service robot related scenario involving the handling of everyday objects such as mugs, knowing the semantic information of where the mug's handle is located could make the work of a grasp planner much easier in order to generate a stable grasp hypothesis as seen in Figure 1.2. Not just in generating grasps, but the semantic breakdown of an object can help in other non-prehensile manipulations like pushing where knowing which part of the object to push could be critical. In the case of mobile manipulators or humanoids, the knowledge of a door's handle can be crucial in navigating through doors.

b) 3D Modeling: The capability to segment shapes could be used in various 3D design analysis tasks. One does not always need to have a known model-based approach to analyzing designs when provided with the ability of segmenting the given objects to different parts. Generative tasks which aim to produce novel designs and shapes[35] could also benefit from being able to breakdown shapes into parts used for further novel shape synthesis by generating the removed part or add parts to the given object all from the knowledge obtained through part segmentation.

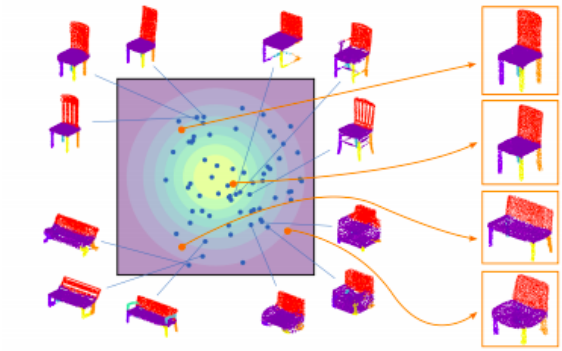


Fig. 1.3. Shape-VAE network learning to generate novel shapes. Samples from the learned embedding distribution used to generate novel shapes with part labels. Image adopted from [35].

c) Others: 3D shape analysis and segmentation in general could provide dense semantic information for a myriad of other applications like medical and dental imaging and computer aided manufacturing.

1.2 Challenges and Difficulties

The challenge with respect to point cloud segmentation is predominantly two-fold, arising from the sensors and the methods involved with respect to raw point cloud data collection and also due to the reason that state-of-the-art conventional image based deep learning techniques fail to generalize to point clouds as discussed below.

1.2.1 Point Cloud Data

Although point cloud data has it's advantages of providing rich geometric, scale and shape information, they tend to be very noisy, sparse and unordered. There can considerably sharp features which seem arbitrary. The density of points is also not uniform. This is caused mainly due to the method involved in point cloud collection with 3D sensors, because of the varying angular and linear rates of the scanners [36]. These inherent qualities of point cloud data call for robust processing capabilities of algorithms. Conversion of point clouds to other representations such as voxels and meshes has also not yielded expected improvements and have their own shortcomings. The various representations that are commonly dealt with alongside point clouds are shown in the below Figure 1.4 of the stanford bunny adopted from [24].

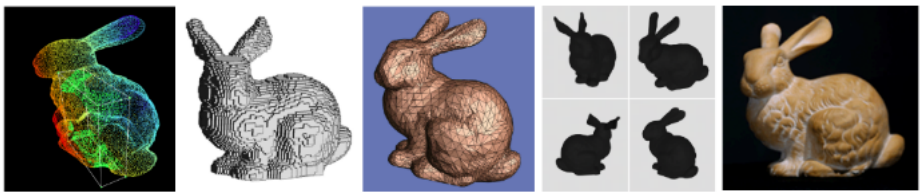


Fig. 1.4 (a) Point Clouds (b)Volumetric (c)Meshes (d)Multi-View (e)Depth Images

1.2.2 Deep Learning on Point Clouds

Due to the unordered and unstructured nature of point clouds, CNNs which dominate the 2D image segmentation techniques cannot be naturally extended to point clouds. Conventional CNNs are well suited for regular structured data like images and even texts(1D sequence). The application of 3D CNNs on voxelized point cloud data has led to improvements in various tasks but they do not scale well to large point clouds and also lose out on information due to quantization artifacts, especially when adopting binary occupancy grids to represent point clouds. Thus the non-euclidean nature of point clouds pose challenges to convolutional kernels used to extract information from images as seen in Figure 1.5. The unordered nature also calls for permutation invariant processing of point clouds which has not usually been on the agenda of image processing techniques.

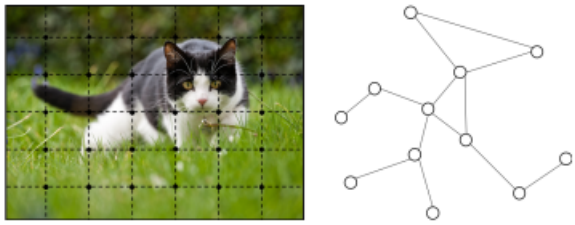


Fig. 1.5. Images in regular grids (Euclidean space), compared to graph data (Non-Euclidean space). Point clouds are usually modeled as the latter for segmentation tasks. Image adopted from [10].

1.3 Problem Statement

As discussed in the previous section, in spite of showing good potential, the methods proposed for directly processing point clouds by exploiting local features have not resulted in expected performance improvements on benchmark datasets like ShapeNet[2]. In this thesis, we aim to explore methods to improve this shortcoming by adapting methods from conventional deep learning on images and also formulating and testing new solutions in order to increase the effectiveness of exploiting neighborhood features beginning with a graph convolution based architecture, DGCNN (Dynamic Graph CNN) as a baseline network [58]. In particular, we explore techniques to enhance the learning capabilities through the already proposed graph convolutions which operate on point clouds modeled as graphs as seen in Figure 1.6.

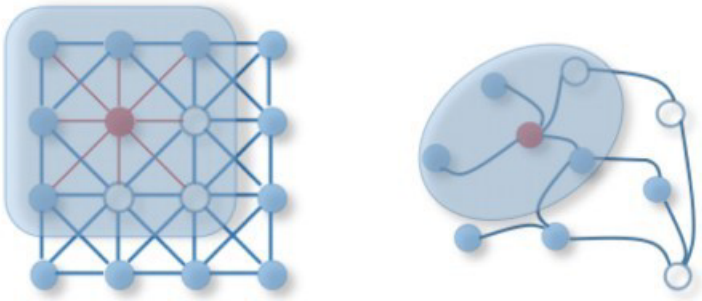


Fig. 1.6. **2D Convolution vs Graph Convolution:** A 3x3 convolution kernel applied on 2D images compared to a graph convolution which operates on the unordered and unstructured neighbors of the center point to learn its hidden representations. Image adopted from [59].

The problem of part segmentation is generally considered to be one of the toughest point cloud processing problems compared to the likes of scene segmentation. It is to be noted that rotation-invariance, translation-invariance, scale-invariance and partial point cloud segmentation are still major open problems and resemble the needs of a network to be deployed in a real world setting. Considering the ultimate aim of the work to help robots in real world to manipulate objects, we also propose and experiment with new solutions to tackle segmentation of partial point clouds. We consider this problem to be of more

importance owing to the fact that complete scans of point clouds of objects are costly to obtain and are sometimes infeasible to obtain for the network once deployed on the real system. It is also to be noted that, only oriented point clouds are considered for this problem as well. Even though translation and scale-invariance are a crucial part of solving this problem, we do not work on proposing solutions to solve the problem of translation or scale-invariance directly but only approach the problem with the sole aim of segmenting partial point clouds.

As with any deep learning based approach, one requires a wide and varied collection of data to train and experiment, hence the changes with proposed methods are tested against a popular benchmark dataset, ShapeNetPart. It consists of 16 object categories, 50 part categories and a total of 16,881 examples.

2

State of the Art

This chapter is dedicated to discussing the state-of-the-art covering all three major problem areas that we will be later dealing with in this thesis. These include point cloud segmentation, partial point cloud analysis and point cloud completion, also known as shape completion. Each section below expands on each of the problems.

2.1 Point Cloud Segmentation

Most of the work done previously can be broadly categorized into two different approaches, projection based and direct point processing approaches.

2.1.1 Projection Based Approaches

These approaches usually involve projecting the point clouds to regular grids and then using conventional CNN based models for learning. They can be further classified into two major types:

1. **Image-based:** Image or Pixel-based architectures typically make use of multi-view images of the 3D object collected from different viewpoints. View-based features are learnt and aggregated to form the global features for the entire shape. The pixel-wise features are then aggregated to make the final predictions on the original input point cloud. As an example, the output from the network proposed

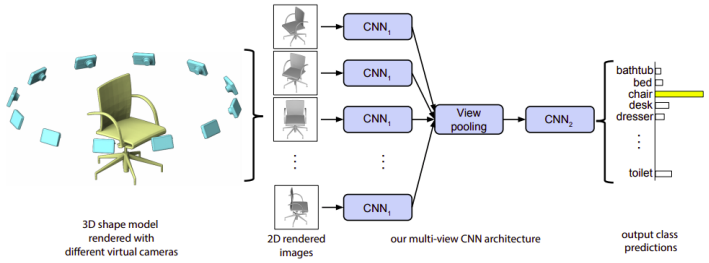


Figure 2.1: **Multi-View CNN:** The network in the image takes rendered 2D images from the 3D models and aggregates the features learnt from each of the images to obtain an object classification score. Image adopted from [46].

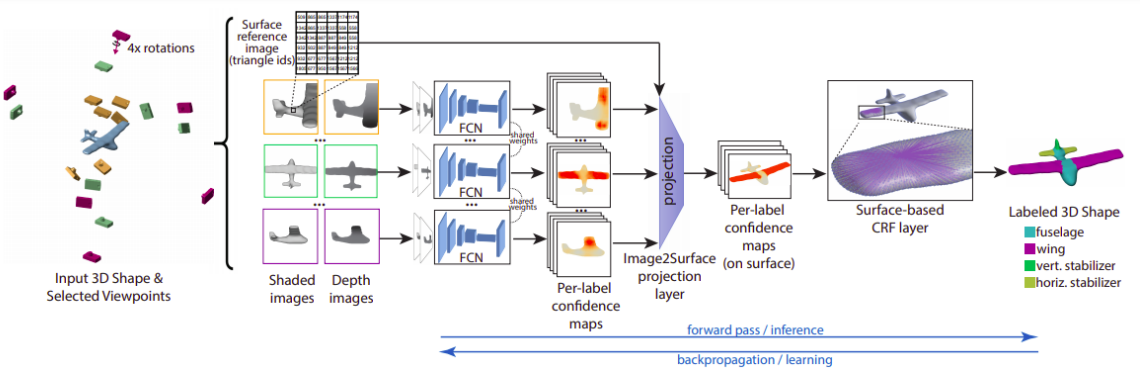


Figure 2.2: **Deep Projection:** The network in the image takes rendered 2D images from select viewpoints and processes them through a *shared Fully Convolutional layer (FCN) layers* to obtain confidence maps for individual parts which are projected back to the 3D model through a special projection layer. A *Conditional Random Field (CRF)* is then used for further processing for consistent labeling across the surface. Image adopted from [22].

in [46] and as seen in Figure 2.1 is only a classification score for the input shape and not a per pixel segmentation score. A spatial correlation of the learnt 2D features is required for segmentation purposes.

In order to overcome the problem of obtaining correspondences between images, a deep projective approach is proposed in [22]. Here they tackle the problem by projecting the predicted scores from each multi-view image onto the mesh model and then using a CRF layer to promote consistency of predictions. As seen in Figure 2.2, the network uses a FCN layer for each of the parts and is only able to handle a few number of parts, around 2-6 and hence has an inherent scaling issue. One can also notice that the input to the network is only the multi-view images and no part of the mesh or 3D data.

As an alternate approach to incorporate 3D data with the network inputs, a joint 3D and 2D based approach for segmentation was proposed in [3]. The network uses voxelized 3D data along with multi-view images and a backprojection layer to map the learned 2D features to 3D voxel grids to finally make voxel-wise segmentation predictions. As the voxel-wise segmentation scores are projected onto the point clouds, there is an inherent limitation caused due to the voxel grid resolution. All the above discussed approaches generally suffer due to self-occlusion related problems, required number of viewpoints and challenges in fusion and backprojection of final predictions from multiple views.

2. **Voxel-based:** The architectures under this category project the input point clouds to regular 3D voxel grids and make use of conventional 3D CNNs to transform features. Some methods process the projected point clouds as binary occupancy grids, which can be considered as a sparse representation of point clouds. The approaches proposed in [21, 49] use 3D Fully Convolutional Networks to segment voxelized point cloud data and project back the segmentations. These networks take the binary occupancy values along with other modality information as input. The discretization due to the binary grid coupled with the resolution of the grid turn out to be a hindrance for scaling and improvements.

Voxel based architectures like [32] have dominated the point cloud based object recognition tasks but

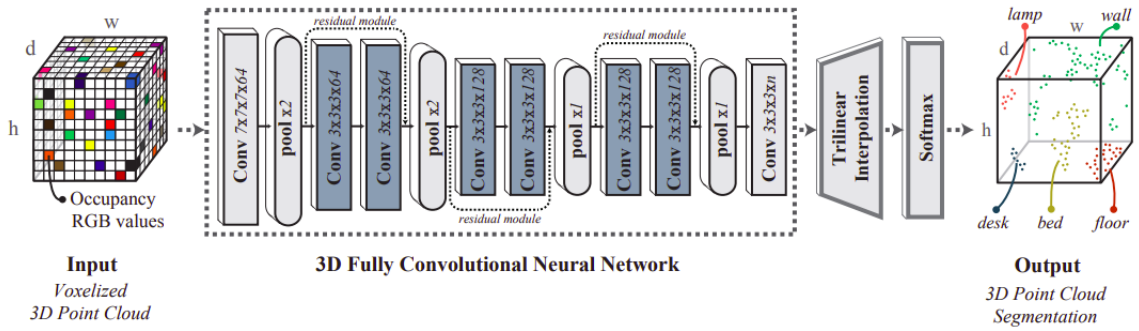


Figure 2.3: The network shown in the image uses *3D Fully Convolutional layer (FCN) layers* to obtain segmentation scores. A trilinear interpolation technique is used to map the voxel-wise scores to the point clouds. Image adopted from [49].

still suffer from discretization owing to its binary occupancy grid representation at low resolutions. In order to tackle the problem of discretization, a recent method implements kernels to extract voxel-wise continuous representations instead of processing binary voxels as in [33]. Here, they use radial basis function (RBF) kernels to learn representations as continuous values for each voxel and train them in a Variational Autoencoder (VAE) setting to find latent representations for each sub-voxel. These sub-voxel representations are further stacked to provide as input for the next layer. The authors then propose group convolutions for 3D voxel grid data to achieve rotation equivariance on defined groups. In spite of showing improvements in performance, these methods suffer from quantization artifacts, low spatial resolution and high computational demand.

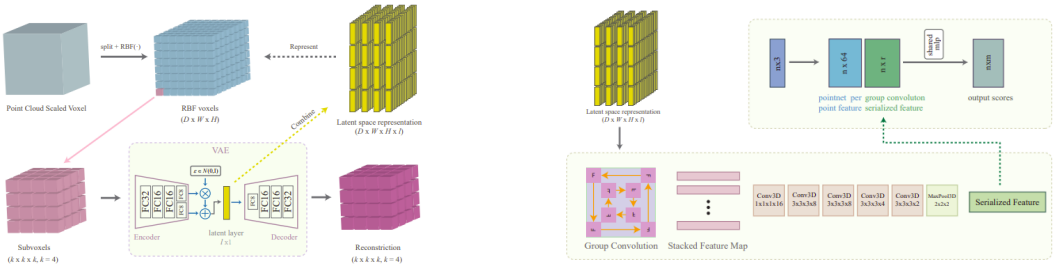


Figure 2.4: **Voxel-VAE Net:** Continuous representations are learned via *RBF kernels*. *Group Convolutions* are used to incorporate rotation equivariance on the *p4m* group of mirroring and 90-degree rotations. Image adopted from [33].

2.1.2 Direct Point Processing

As the title suggests, the methods in this category take point clouds without any preprocessing except for normalizing and rescaling values before feeding them into the network. These methods can be further

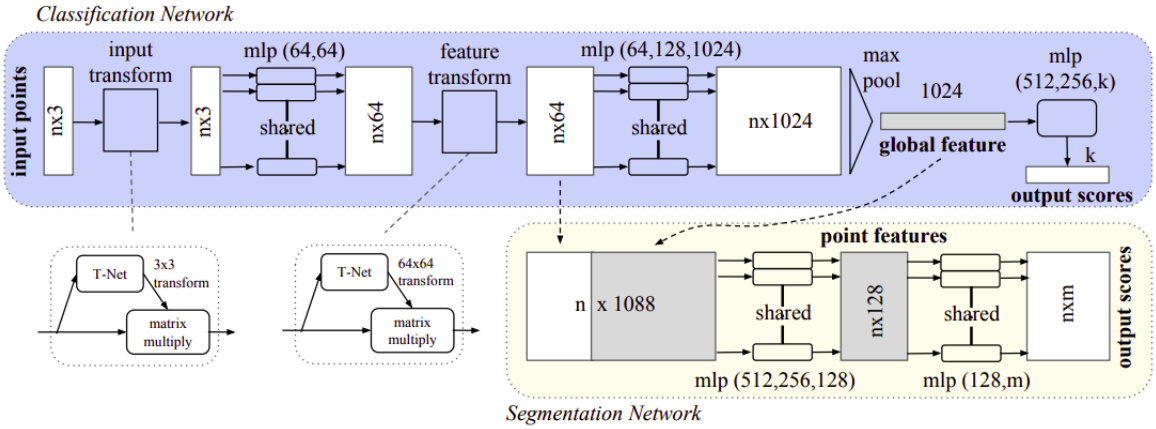


Figure 2.5: **PointNet**: The network in the image uses *shared MLPs* to extract features from individual points. A spatial transform net, *T-Net* is also added at the beginning of the network in order to cope with certain geometric transformations. Image adopted from [38].

divided into four major categories such as:

1. Pointwise MLPs: Pioneering methods in direct point processing typically used shared MLPs or what might be considered as 1x1 image convolutions or unary convolutions to process pointwise features. They were combined with a symmetric function like max or mean pooling across points to overcome the permutation invariance problem. For segmentation tasks, individual point features are concatenated along with the permutation-invariant global feature. PointNet [38] was the first network as such to successfully prove its performance for classification as well as segmentation tasks.

A spatial transformer network (STN) is also deployed in order to learn to project the input point clouds to a certain canonical orientation before feature extraction. A feature transform net is deployed as well, but with the additional regularization term as shown below which is added in order to stabilize training.

$$L_{reg} = \|I - AA^T\|^2 \tag{2.1}$$

where A is the feature transform matrix and I the identity matrix. The regularization loss term is only applied to the feature transform net and not to the STN at the input, hence doesn't penalize a non-rigid transformation at the input. Another network was proposed in DeepSets[67] where they formulate operating point clouds as a set function learning problem which is solved in a similar manner to PointNet but instead apply summation in place of max-pooling in order to obtain global features over non-linearly transformed point features. Both the methods discussed above only extract features for individual points and completely ignore any opportunity to make use of local geometric structure around each individual point. They completely rely on global structural information and pointwise features for predicting segmentation labels. In order to overcome this issue, the methods in the following sections describe different approaches that make use of local information.

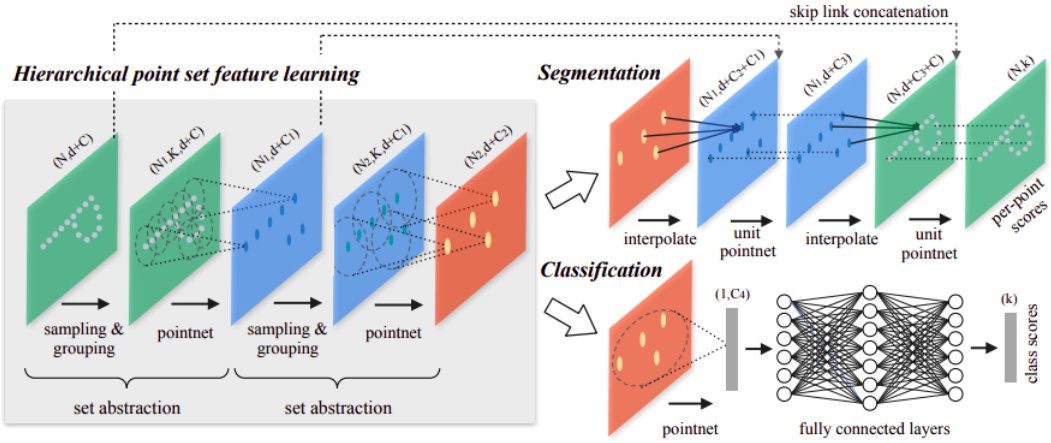


Figure 2.6: **PointNet++**: The network in the image uses *shared MLPs* as mini-PointNets to extract features from individual points and its neighborhood. The points are subsampled based on *farthest point sampling* (FPS). Image adopted from [39].

2. Hierarchical Approaches: PointNet++ (PN++) [39] was first introduced as a hierarchical approach which recursively applies mini-PointNets on a nested partitioning of the input set of points. The basic idea is to operate on a subsampled set of points after each layer with features assigned to a centroid point around each region, which in turn helps to learn features with increasing receptive field sizes.

For classification tasks, the global feature is alone used for inference. For dense prediction tasks, the points are upsampled to the original number forming an encoder-decoder based approach. Given a set of points as input, PN++ first applies a *Farthest Point Sampling* (FPS) technique in order to choose the centroids. A ball query neighborhood is then explored to collect the neighbors for each centroid. The k-neighbors are then represented relative to the centroid. Each region represented by the centroid and its neighbors are then transformed by a mini-PN with the centroid being assigned the aggregated features from its neighborhood. This forms a so called Set Abstraction (SA) layer. Multiple SA layers are applied consecutively leading to a downsampling of points and increased receptive field. In the case of segmentation, the upsampled points are assigned feature values based on inverse distance weighted average from nearest neighbors. Skip connections are established to concatenate features from the encoder SA layer with the same number of points before applying another set of shared MLPs. The subsampling of points not only enforce a hierarchical learning but also serves to be more computationally efficient compared to having the neighborhood feature aggregation with every individual point as a centroid.

The FPS algorithm is stochastic in the way it produces different set of points depending on the chosen first point. An alternate approach is proposed in [28] where the spatial distribution of the point cloud is modeled by a self-organizing map (SOM). The SOM generates the centroid points which serve as the anchors for feature extraction for each individual points based on a point-to-centroid kNN search based on the associations between the points and the centroids. Max-pooled features are then assigned to each centroid which may later be used for classification. For segmentation purposes, per-point features are

(a) Output of a grid-based subsampling in image adopted from [51]

(b) SOM based sampling. Image adopted from [28]

(c) Farthest point sampling. Image adopted from [70]

Figure 2.7: **Sampling Comparison:** Image shown in (b) has just one centroid point for each leg of the table. The SO-Map generated in [28] tends to avoid redundant centroid points in regions exhibiting similar spatial distribution compared to the centroid points generated by FPS in (c) or grid-based subsampling as seen in (a).

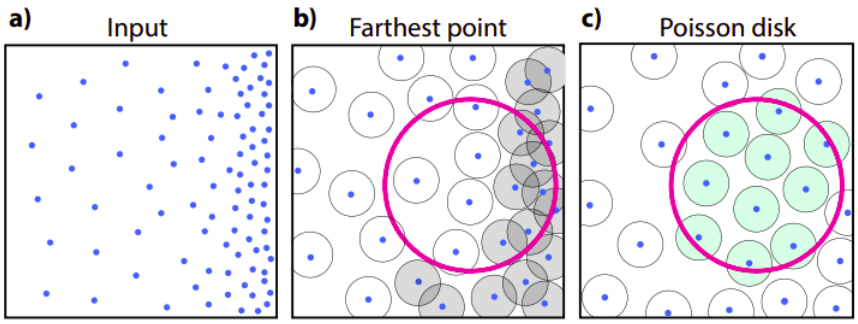


Figure 2.8: **Poisson Disk Sampling**: Given a minimum distance to maintain between points, there is a clear overlap in the regions around the sampled points as seen in (b) represented by overlapping gray circles. The neighborhood region around the center point is indicated by the pink circle. Image adopted from [13].

aggregated via average pooling on the point-to-centroid graph, before being passed to layers of shared MLPs to predict segmentation scores.

Another subsampling approach such as the *Inverse Density Importance Sampling* is attempted by the FlexConv [9] network. In spite of the computational efficiency due to the subsampling approaches as seen in Figure 2.7 and 2.8 in the previous networks, the very method used to identify the set of points for each subsampling step turns out be computationally intensive and not suitable for processing large point clouds. Hence a random subsampling approach is tried and tested with success by RandLA-Net [14] for processing large scale point clouds and speeds up computational time and makes it 200x faster.

The network in [31] adopts a hierarchical approach with dense blocks inspired from [17] which set a new benchmark for image processing. Several other methods exploit the hierarchical approach along with certain modifications to local feature aggregation [20]. In [60], the global feature is used to predict a confidence vector representing the objects present in the scene. This confidence is further used to refine the segmentation predictions of individual points. In addition to this, they also propose a region-similarity loss in order to pull closer the distinguishing features and get rid of any ambiguous features in the

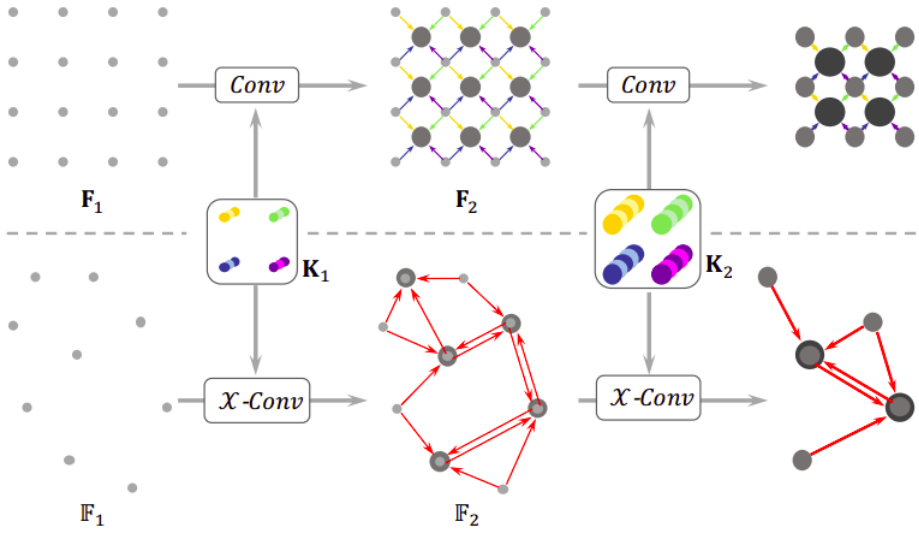


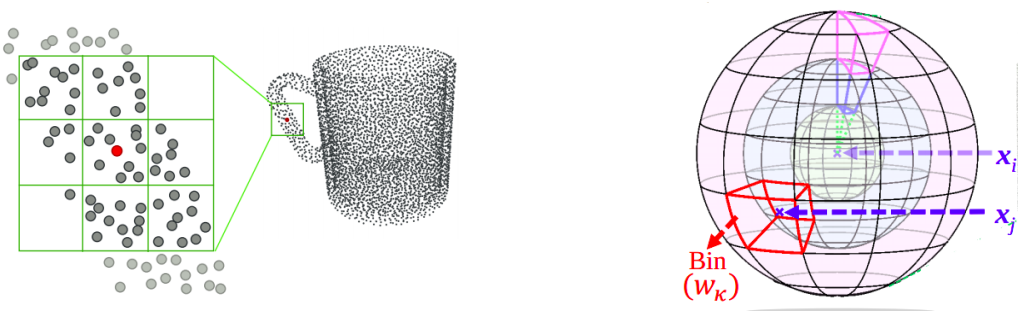
Figure 2.9: **PointCNN**: An illustration of 2D convolutions and PointCNN method of using typical convolutions. Before applying the convolution operator K_1 that leverages spatially-local correlation, χ-Conv operator weighs and permutes the centroid point along with its neighbor to some latent canonical order. Image adopted from [29].

neighborhood.

In [29], the network tries to learn a χ - transformation to weigh and permute points before applying a typical convolution operator like in images as seen in Figure 2.9. The network uses FPS for subsampling of points. The authors claim that the χ-Transformation is able to realize equivariance even though its not been shown explicitly. Here, the network captures the permutation at the input and produces a χ-transformation matrix depending on the input order of points and then uses it to permute the input set of points in each neighborhood. In spite of the authors' unproven claims, the matrix is able to achieve some transformation that is able to place the points in some latent and canonical order useful for learning through typical convolution operations.

3. Point Convolutions: As described in the previous section, 3D convolution kernels are not easy to design considering the sparsity and unstructured nature of point clouds. Nevertheless, convolution kernels have been adopted for processing point clouds and defined in different ways. The proposed methods under this approach can be broadly classified into discrete and continuous convolution methods.

a) Discrete Methods: These methods typically involve pointwise convolution operations with kernel weights convolved according to the binning of points in the neighborhood or assigning weightage through some kernel functions. In [15], the authors propose a discrete convolution approach to bin points into respective kernel cells and convolve the points with the respective kernel weights. Here the grid cells are arranged in a typical grid like structure as seen in Figure 2.10a. The kernel is typically centered at each of the input points and the neighboring points covered by the kernel will eventually contribute to the feature



(a) Regular grid convolution. Image taken from [16]

(b) Spherical kernel. Image taken from [27]

Figure 2.10: **Discrete Convolutions:** Illustration of different discrete convolution kernels adopted for learning on point clouds.

ascribed to the center point. Each sub-domain of the kernel has its own weights and the end product of the convolution is the summation of the convolutions across each sub-domain or each cell. Each cell can be altered with respect to a certain radius or width for good enough receptive field at each layer of the network.

As an alternate approach, spherical kernels were proposed in [27]. Just as in pointwise convolutional networks, the spherical kernels are centered along each point and the neighboring points are divided into the respective bins as seen in Figure 2.10b. In addition to deploying spherical kernels, this network also adopts a hierarchical approach. the hierarchical approach also demands a change in size of the kernel at each layer to ensure sufficient coverage across neighboring points.

Another approach proposed in [25] centers the kernel on each input point and divides the region around each point into octants. Each edge between the center point and the neighboring point is then convolved with the respective kernel weights. Here the kernel weights are not placed at the center of each octant and are rather placed along the direction of the individual axes spanning the octant. The features from each of the kernel transformation are then aggregated based on the angle between the edge and corresponding base axis as seen in Figure 2.11.

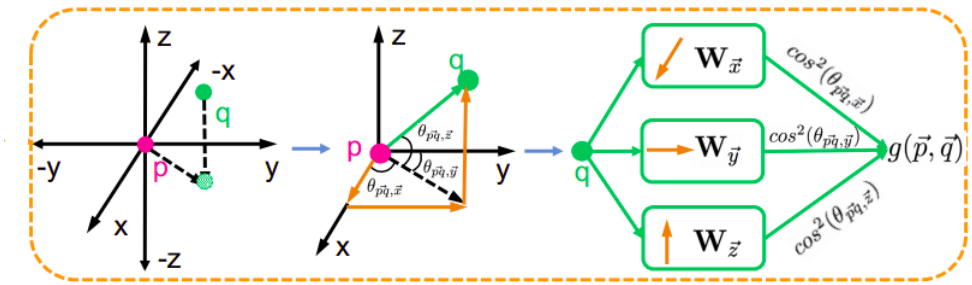


Figure 2.11: **Geo-Conv**: Edge feature learning using Geo-conv operation. Based on the angle θ between the edge and each of the axis, features for each edge are finally aggregated as a weighted sum. Image taken from [25].

In all of the above approaches and other related methods, while some try to better capture the spatial distribution within each convolution kernel via the edges between each center point and its neighbors, others do not consider it. Hence, unlike in [25], other methods don't explicitly make use of the edge features and completely rely on the input absolute or relative features with respect to the center, of its neighbors. Few works argue that an MLP as a kernel function can span the continuous vector space and hence propose an approximated continuous convolution operators which is discussed in the following section.

b) Continuous Methods: The network proposed in [13] tries to deal with non-uniformly sampled point clouds using continuous Monte Carlo convolutions. The authors point out the difference in performance exhibited by previous convolution based methods when dealing with non-uniform sampling densities, and propose a convolution which can be stated as a Monte Carlo estimate by utilizing the samples's density function. The convolution at each point is defined based on *Monte Carlo Integration* which computes the integral only using a subset of randomly picked samples.

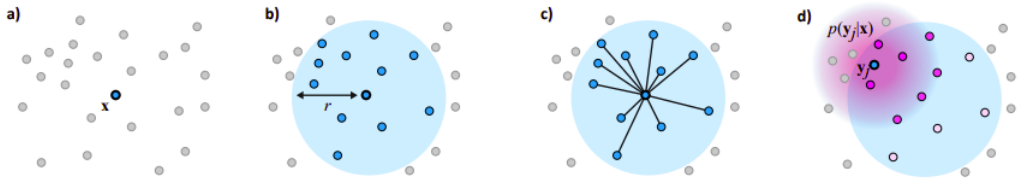


Figure 2.12: **MC-Convolution**: Steps involved in the proposed convolution operation. The PDF of each neighboring point y_j obtained via KDE is controlled using a bandwidth parameter visualized as a pink disk in the image taken from [13].

The convolution with each point x as center and its neighborhood points is defined here as:

$$(f * g)(x) \approx \frac{1}{|N(x)|} \sum_{j \epsilon N(x)} \frac{f(y_j) g(\frac{x - y_j}{r})}{p(y_j|x)} \tag{2.2}$$

where f is the function to be convolved, g represents the kernel, N(x) represents the neighborhood of x, defined by a ball query of radius r and the main part of the approach, $p(y_j|x)$ is the *probability density function* (PDF) at point y_j with a fixed point x. This PDF is in turn learnt through a *kernel Density Estimation* (KDE) process, with a *Gaussian* kernel as the density estimation kernel.

In short, the continuous kernel function implemented here through an MLP is weighed based on the density function and hence making up for the non-uniform sampling inherently present in most point clouds and which tends to affect the results of convolution operation proposed in previous methods.The network in [15] also employs a similar convolution method. In addition to deploying the MC-Convolution, the network in [13] follows a hierarchical approach and also uses a different sampling approach when subsampling for its consecutive layers. Here, they deploy a Poisson Disk (PD) sampling technique in order to afford scalability to process larger point clouds. The PD sampling approach also helps in fixing the

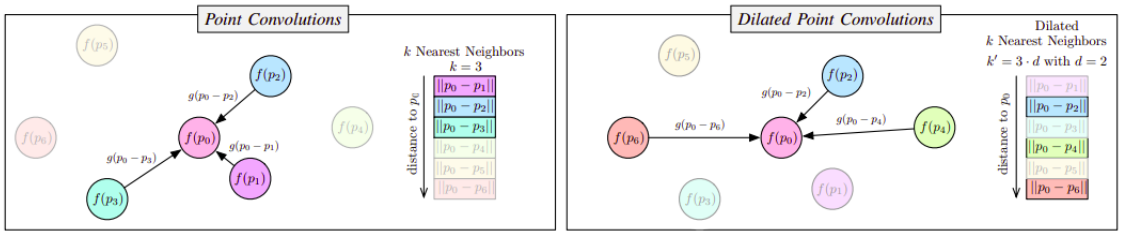


Figure 2.13: **Dilated Point-Convolution**: The network as seen in the image applies a small tweak by picking alternate points arranged according to their distances, in choosing the neighbors given a points larger k' set of neighbors compared to just a set of k neighbors. Image adopted from [8].

number of points within the defined receptive fields surrounding each center point. The effects of this can be seen as shown in the Figure 2.8.

In [56], the authors propose parameterized continuous convolutions as a new learnable operator for non-euclidean data and formulate their approach based on MC-Integration [1]. The work in [8] shows the improvements achieved by increasing receptive field sizes using dilated point convolutions. Here, they make a slight modification to pick the neighbor points for convolution from its computed k-nearest neighbors as shown in Figure 2.13. The network achieves better results especially on large scale point clouds and outperforms previous approaches on multiple datasets.

KP-Conv [51] uses a kernel represented by points distributed uniformly along the surface of a sphere but in contrast to the discrete spherical convolution method does not use the binning approach as in [27]. The convolution on each point is driven by the distance from the point to the kernel points. The convolution in KP-Conv is defined as:

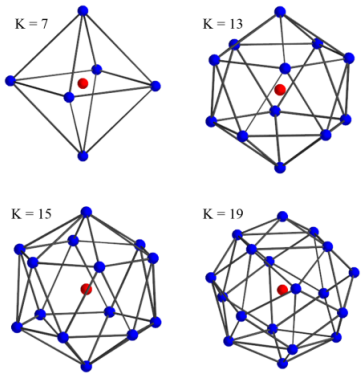
$$(f * g)(x) = \sum_{x_i \epsilon N_x} g(x_i - x) f_i \tag{2.3}$$

where f is the input features to be convolved, g represents the kernel function, x_i the neighboring points coordinates and f_i their corresponding features. The kernel function is assigned as below:

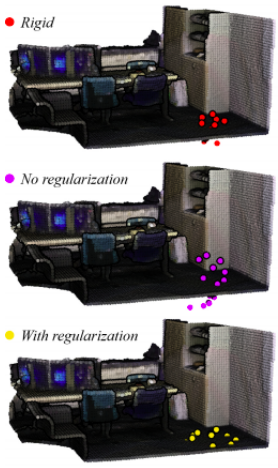
$$g(y_i) = \sum_{k<K} h(y_i, x_k) W_k \tag{2.4}$$

where y_i represents the relative coordinates of the neighboring point x_i w.r.t x, x_k the kernel weights position coordinates, K is the total number of kernel points and h represents the linear correlation between y_i and x_k and is greater than 0 when y_i within the region of influence of the kernel weight and gets higher if it's closer. The kernel points are placed in a way to give maximum coverage within a sphere which in turn is solved as an optimization problem and added as a additional regularization loss term during training. The stable ideal dispositions are shown in Figure 2.14a, which indicate positions that give maximum coverage along the surface of a sphere.

A deformable version of the kernels is also proposed as Kernel Point-Flexible CNN (KP-FCNN) where the kernel point shifts for each convolution operations is learnt based on the input points. As seen in



(a) KP-Conv Kernel Points: Stable dispositions for varying number of kernel points. The points are supposed to provide maximum coverage.



(b) Kernel Points Deformations: The learned deformations for the kernel points as in KP-FCNN. Compared to the rigid points, the deformed points seem to represent the underlying structure.

Figure 2.14: **Kernel Point-Convolution**: Kernel point dispositions and deformations. Images adopted from [51].

Figure 2.14b, the deformations are learnt by utilizing the input set of points' coordinates and hence end up capturing the surface structure and brings into play all the kernel weights. It is also shown in the results that the flexible version of the kernel in KP-FCNN is more effective in capturing features and outperforms the rigid version on most of the benchmark datasets.

While most of the continuous convolution approaches learn the operation through a single shared MLP, the difference that sets KP-Conv apart from th rest is the fact that it has a defined K number of weight matrices as chosen for the specific task and places them in a way to better capture the underlying structure than as done in discrete methods where a dedicated MLP operates on points in each bin. In contrast, a single point is not convolved by a dedicated MLP in KP-Conv and rather a weighted sum of the neighboring kernel weights is applied. Hence, KP-Conv in a way reaps the benefits of both discrete and continuous convolutions. It is also to be noted that this method set a new benchmark on the ShapeNet-Part dataset.

4. Graph Convolutions: Graph-based approaches typically involve building local neighborhood graphs with each point as a vertex and exploiting edge features with neighbors. Graph convolutions can be divided into *spectral* and *spatial convolutions*. In the spectral domain, the network in Regularized Graph CNN [50] learns to update the graph laplacian in each layer. The graph here is each vertex point connected to every other point in the point cloud. In order to exploit local structural information, LocalSpecGCN [55] was proposed to work on a local graph constructed by the kNN neighborhood. These convolutional networks can be typically trained for node prediction which classify each point or link prediction tasks which classify each edge. Graph convolution in the spatial domain for point cloud processing was first introduced by [44] where a filter generating network was used to generate dynamic convolution filters based

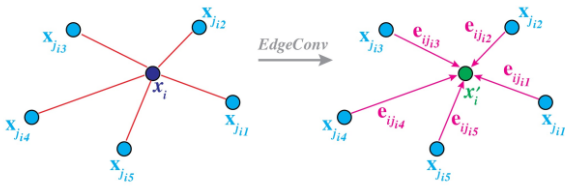


Figure 2.15: EdgeConv from DGCNN. As seen in the figure, the learned edge features e are aggregated via max-pooling and assigned to the center point x_i. Image adopted from [58].

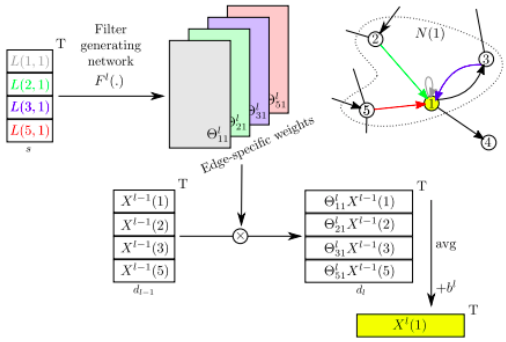


Figure 2.16: Edge Conditioned Filters for Graph Convolution. A filter generating network generates the weights for convolution based on the edge attributes. Image adopted from [43].

on edge attributes and coarsen the graph after every layer. DGCNN [58] introduced the dynamic graph approach with EdgeConv layers, formed of an MLP and a pooling operation, and set a new state-of-the-art for part segmentation.

The graph in DGCNN is updated after each EdgeConv layer based on the feature space neighbors of each vertex without any graph coarsening. The network showed an improved performance than the previous networks on several benchmark datasets including ShapeNet-Part. Since DGCNN has been chosen as the baseline network for this thesis, we will discuss about it in detail in the following chapters. Another network which tries to avoid the simple diffusion strategy proposes an edge-condition filter generation network for learning on graphs as seen in Figure 2.16. As a direct improvement to DGCNN, Linked-DGCNN (LDGCNN) [69] tried to also:

- Reduce the model size of DGCNN by removing the transformer network
- Link hierarchical features from previous layers as seen in Figure 2.17.

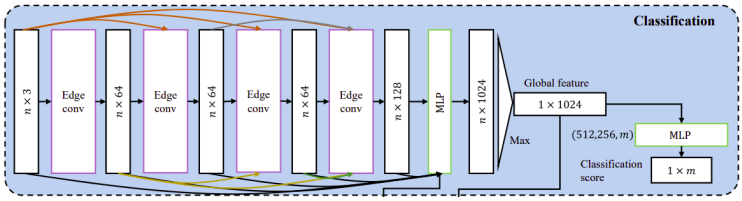


Figure 2.17: Classification Network in LDGCNN. Image taken from [69].

The above changes help the network to extract new edge features based on the features of the previous layers and in turn enhancing the learning capability. This also avoids the vanishing gradient problem since as the network goes deeper, the distance between edges in feature space neighbors is almost close to zero and having features from previous layers increases the distance and in turn helps in alleviating the

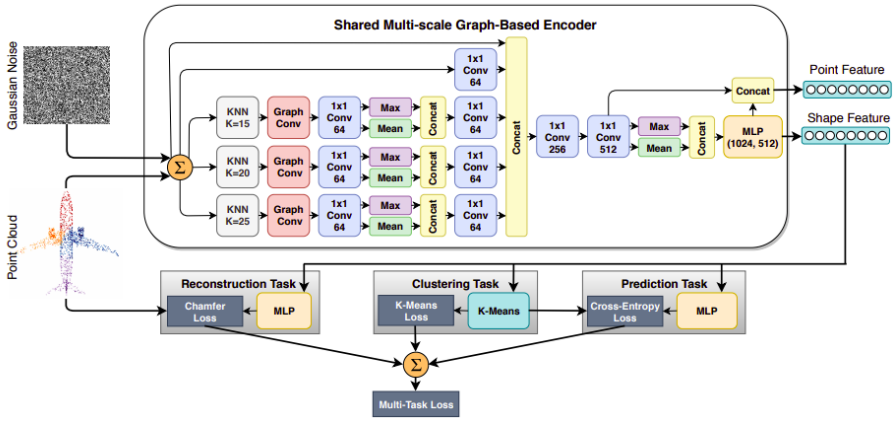


Figure 2.18: Multi-scale graph convolution based encoder in the image adopted from [11]. The multi-task loss is used to pre-train the network.

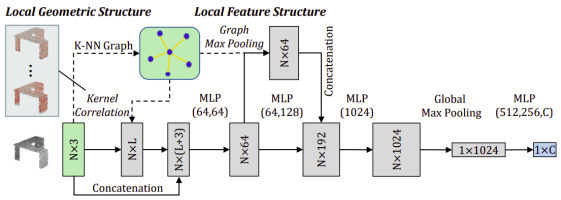
gradient issue.

In [26], the authors try to capture the organization of points efficiently through superpoint graphs, which are formed by geometrically homogeneous elements by anlaysing values like planarity and linearity. The superpoint graphs are then processed using regular graph convolutions.

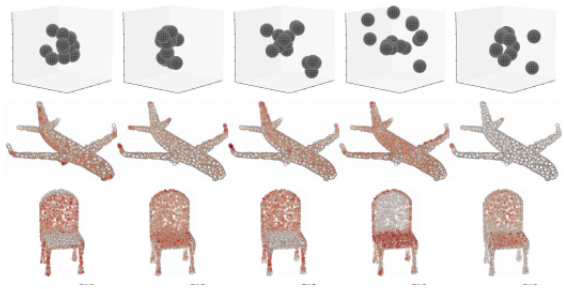
Taking cue from [47], the network in [11] propose unsupervised multi-task feature learning using a shared multi-scale graph encoder as seen in Figure 2.18. They use specialized loss functions for each task with *clustering, classification and reconstruction* losses. The training process alternates between clustering the latent variables and in turn produces labels which are used in a *self-supervised* manner for simultaneously predicting the class as well as reconstructing the input. This multi-task learning is typically used for pre-training process and they train on close to 57,000 models from 55 object categories comprising of the wider ShapeNet dataset. A *transfer-learning* approach without any fine-tuning yields an impressive 89.5% accuracy on the ModelNet40 dataset for shape classification and around 77.7% mIoU on ShapeNet-Part dataset for part segmentation.

In order to extract local geometric features from the kNN graph, the authors in [42] propose to learn the kernel point positions to extract kernel correlation information to be added as another modality to the input. The kernel correlation is a function of the pairwise point distance. The network takes inspiration from regular image convolution where convolutions are intuitively seen as a measure of similarity between the input and the kernel values. Such methods have also been adopted for point cloud registration where one of the point cloud is iteratively transformed in order to maximize the kernel correlation value with respect to the desired reference frame. Thus, convolution here is used to measure the geometric affinity between a learned set of points and the input and used as additional features for succeeding layers as seen in Figure 2.19.

5. Other methods: Different data-structuring schemes have been explored in order to overcome the memory inefficiency problem, especially of voxelized point clouds. In [23], the authors propose to use a



(a) KC-Net Architecture: kNN graph is used for for estimating kernel correlations as well for max-pooling operations to aggregate local features.



(b) Learned kernel point visualizations and the corresponding filter responses in the figures below.

Figure 2.19: **Kernel Correlation Network**: Network architecture and kernel points visualization. Images taken from [42].

kd-tree data structure for convolution operations, where points belonging to the same split type within the kd-tree share parameters for convolution as seen in Figure 2.20. Basically, the features for each parent node is computed from its children nodes. The authors theorize that the usage of kd-tree will help in:

- Deciding which leaf representations are getting grouped together and
- Exploiting the underlying geometric information of the point cloud encoded by the structure of the kd-tree itself

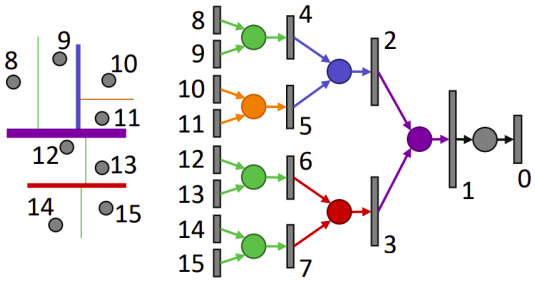
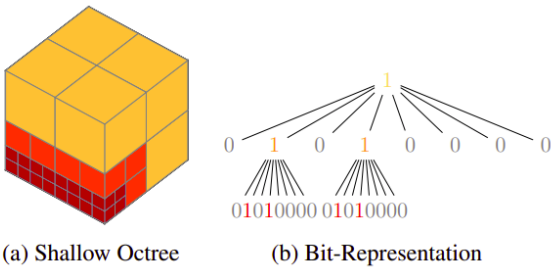


Figure 2.20: **Kd-Net**: Kd-tree for a group of points and the corresponding kd-net. Similar color circles represent shared parameters. Image taken from [23].



(a) Shallow Octree (b) Bit-Representation

Figure 2.21: **OctNet**: Shallow octrees and corresponding bit representation used in octnet . The depth of split is indicated by the color of the bit representations with a bit of value 1 indicating a presence of a point in that octant. Image taken from [40].

In [40], the authors propose a hybrid grid-octree data structure. Oct-tree representation recursively sub-divide 3D space into octants wherever relevant information is present. The authors propose a shallow

oct-tree as an effective alternative for voxel grid representations. The shallow octrees also alleviate the problem of inefficient memory access with respect to conventional octrees. Here the representation of a particular octant can be accessed just by simple arithmetic indexing compared to complex data structures required to implement complete octrees, as can be seen in Figure 2.21.

In order to increase flexibility of networks in segementation, PartNet [64] was developed as a model for *recursive part decomposition* for point clouds using recursive neural networks. *RNN-based* approaches have been proposed as well, like [7] which tries to extend PointNets to incorporate larger scale spatial contexts. [19] proposes a network to exploit the interaction between point and edge features in parallel branches of the network.

2.2 Partial Point Cloud Analysis

Analysing partial point clouds is a ubiquitous need which arises from the manner in which 3D data is generally captured. Most real life scenarios make it nearly impossible to have capture complete 3D point clouds and hence it is of critical importance that we develop tools to analyse them. Deep learning on partial point cloud of objects is still an evolving domain. The current works proposed are mostly focused on learning to predict the missing information via shape completion techniques. Partial point cloud segmentation on the other hand has received very little attention thanks to the many unsolved issues even with partial point cloud classification, a comparatively lower level task. In this section we will look into concurrent works that were developed to deal with partial point clouds.

2.2.1 Shape Completion

Shape completion is the task of predicting a complete point cloud given a partial point cloud of an object or shape. The network architectures that are used for shape completion are inspired from auto-encoder architectures. The critical part of such networks is the decoder architecture which deals with predicting points from the latent space. There have been several decoder architectures proposed for point cloud auto-encoding and completion.

One straightforward but computationally expensive way is to use *Fully Connected* (FC) layers to predict the final points from the latent space. The authors in [61] propose to decode shapes by deforming a canonical 2D grid. This network is called as the folding net for the way it learns to morph the grid structure onto the surface of the underlying shape in an iterative manner. The network has graph based encoder and the proposed folding based decoder as seen in Figure 2.22. This network has considerably lesser number of parameters than the FC decoder since it uses shared MLPs to predict each point. The reconstruction loss used to train the network is *Chamfer Distance* loss which does not require the number of predicted points and of ground truth points to be the equal.

In spite of being first proposed as an auto-encoding architecture, the idea of folding net has been adopted by an important pioneer network for point cloud completion. Hence, as it is the above seen model of folding net is not designed exactly for point cloud completion purposes. Even though the authors claim that the 2D grid can generate shapes that are not topologically similar to it, the network apparently

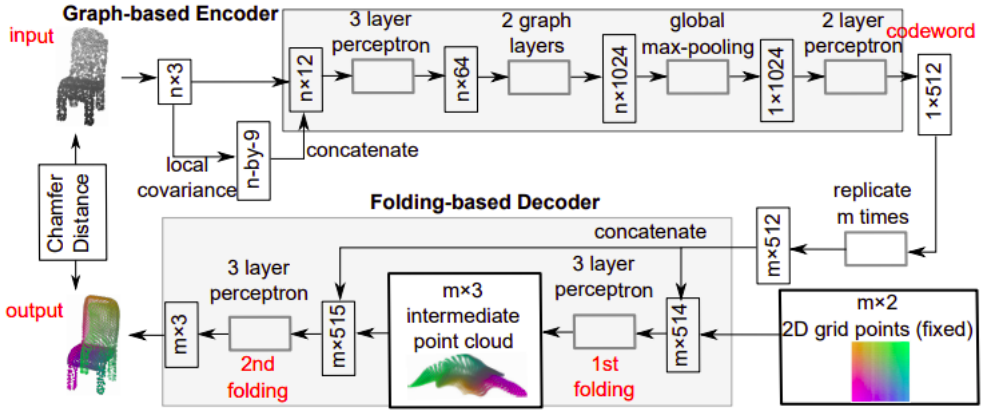


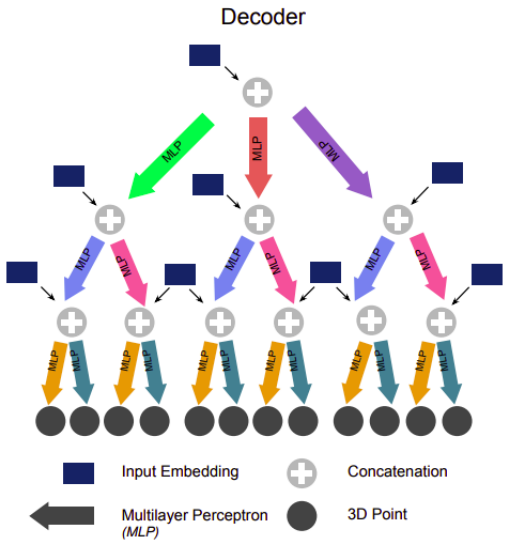
Figure 2.22: **FoldingNet**: The network learns to predict points by learning to fold the 2D grid to represent the underlying surface, in a two-stage fashion using the latent space feature as shown in the image taken from [61].

suffers from it.

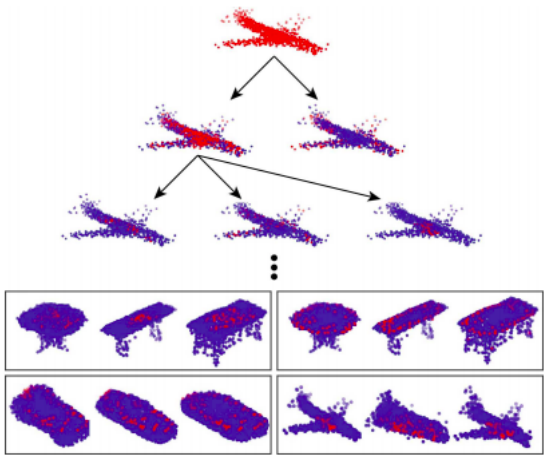
As an alternative, the authors in [66] propose to use the folding mechanism only after predicting a set of points which coarsely represent the geometric shape of the complete object. Thus, this network utilizes a fully connected decoder followed by a folding net style decoder for each local region around a point. This particular architecture is adopted due to observation that the folding net was better at approximating the smoothness of the underlying surface and not ideally suited to represent the entire geometric structure of the object, which is predicted by the fully connected layer driven coarse shape prediction layer. The network utilizes two different losses for the reconstruction task. The Chamfer Distance (CD) loss is used along with the Earth Mover's Distance (EMD) loss for the coarse prediction layer. The final prediction is used for calculating only the CD loss. The CD loss used here is the symmetric version. The EMD loss requires both the number of predicted points and the GT points to be equal and is also computationally expensive due to the bijection mapping involved between the predicted and GT points.

In addition to proposing an improved decoding mechanism, the authors also provide proof for their selection of a PointNet based encoder. The encoder consists of a two stage encoder where the global feature from the first stage is concatenated along with the input features to obtain the latent space feature. This way the authors propagate the necessary global contextual information while extracting features from individual points. This method seems to outdo even the graph based encoder approach or hierarchical approaches similar to PointNet++. Thus, indicating that local feature aggregation may not in fact be as vital when it comes to point cloud completion compared to other point cloud processing tasks. The network is tested extensively on synthetic data from ShapeNet and real world scans KITTI and formulate new metrics to evaluate the results. The proposed hybrid network considerably outperforms the plain folding based decoder for point completion tasks.

Taking cue from the smoothness constraint that is inherently present while deforming a 2D grid into



(a) TopNet Architecture: Inverted tree architecture. The arrows indicate MLPs and similar color indicates the shared MLPs.



(b) Points indicated in red represent the points predicted by traversing through the node to the last level and collecting all its leaf points.

Figure 2.23: **TopNet**: Network architecture and hierarchical decoding visualization. Image taken from [48].

3D shapes, the authors in [48] propose a hierarchical decoding mechanism which outperforms the previous methods. The core idea of the network is to not enforce any topology or structure in all of the levels of decoding. The authors prove that, given the flexibility of with the number of parameters and allowing for redundancies, the network learns any topology. Thus, not enforcing or assuming a particular topology turns out to be the biggest contribution by the proposal.

The network takes the encoded feature from an encoder same as the one in [66]. The encoded feature is then passed through a tree-structured decoder to generate the required number of points as seen in 2.23. The decoder is not constrained in any other way than the reconstruction loss at the output. The authors provide results indicating that the network is able to learn an underlying topology with outputs at the each level showing some semantics underlying it.

The network achieves this without enforcing the *2-manifold or surface topology* to learn mappings to produce completed object structures. The network goes by the definition of a topological space on discrete point sets as being a non-empty collection of subsets of a given set S. The network tree has the following three characteristics:

- The number of children nodes for all nodes at a level are equal
- Every node has only one parent
- All leaves belong to the same hierarchical level

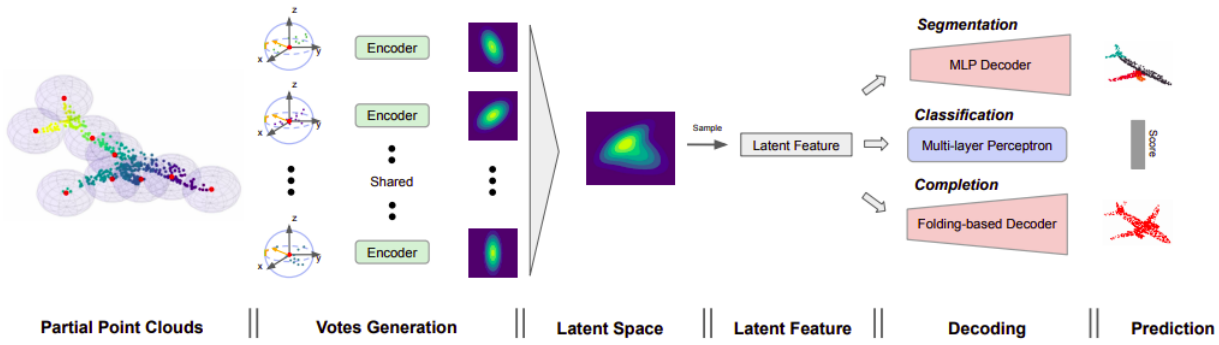


Figure 2.24: **Point Set Voting**: The network treats the input points as a number of partial subsets to learn the latent space feature, which is obtained by a random voting strategy amongst the subsets trained. Image taken from [68].

Unlike the previous works, the network lacks in any usage of intermediate geometric reconstruction supervision and completely relies on the final outputs.

2.2.2 Partial Point Cloud Segmentation

Partial point cloud segmentation is the problem of predicting point-wise semantic labels for a given incomplete point cloud. Partial point cloud processing is of utmost importance owing to the fact that complete point clouds are rarely available in real world scenarios. As seen in previous works, all the network heavily rely on the global information in order to predict the point-wise semantic labels, indicated by the number feature channels typically dedicated to the global feature vector. Hence, when the input point cloud is corrupted and the network is presented with occluded point clouds, one would easily guess the network to degrade in performance by a drastic margin. The work on segmenting partial point clouds has not gotten traction yet except for one recent method which aims to tackle the same.
The authors in [68] point out that the previous networks have not been designed in a way to handle partial point clouds and hence propose their new approach of Point set voting. The network takes inspiration from conventional image processing technique such as *Hough Transform* and another deep learning based object detection method VoteNet [37].

Given an input point cloud, the network divides them into a fixed number of partial subsets. The network is trained to generate votes from each of the partial subsets. The votes generated by each of the subsets is not a deterministic feature vector but rather a distribution in the latent space. In order to model the distribution, the authors follow the *Conditional Variational Autoencoder* (CVAE) framework. A direct application of CVAE is made possible thanks to division of the input point clouds into partial subsets.

In the CVAE setting, the authors assume a Gaussian distribution with independent variables for the latent space features which is modeled by the recognition network $q_\phi(z|x_i)$, where z is the latent space distribution and the x_i the input sample. During training, the network collects *votes* by computing the

optimal latent feature z_{opt} with the highest probability amongst the randomly picked subset distributions. This is done by maximizing $q_\phi(z|x)$ with respect to z and achieved trivially through a closed form solution.

$$z_{opt} = argmax \prod_{i=1}^{n} q_\phi(z|x_i) \tag{2.5}$$

Also, since the latent feature is computed directly through a closed form solution, the reparameterization trick used for VAEs can be avoided. Note, that the network obtains its robustness to partial point clouds owing to its training strategy:

- Pick randomly a number of subset partial point clouds' latent space distribution
- Compute the latent feature z_{opt} as the one that maximizes the probability of the picked distributions

One of the main advantages of this approach is there is no requirement for annotated partial point clouds at training stage. The authors propose a testing scheme to test the network on benchmarks like ShapeNet and achieve impressive results of 79.0% and 78.1% mIoU on complete and partial point clouds respectively. Although it achieves high performance on partial inputs, the network works under the assumption of knowing the exact boundaries and scale of the complete point cloud which is not accurately known is most real world scenarios, as witnessed in concurrent *amodal object detection* techniques [6].

2.2.3 Semantic Scene Completion

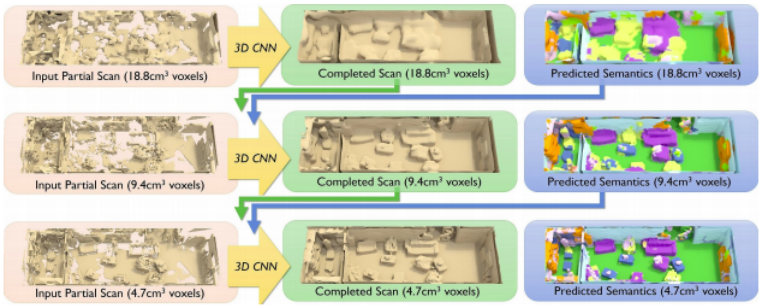


Figure 2.25: **ScanComplete Net**: Given an input large scale partial scan, the network learns to complete it with semantic labels in a corse-to-fine hierarchical manner. Image taken from [4].

There have been few works proposed to tackle the problem of partial input segmentation when operating on voxel grids. These have usually been for large scale semantic scene completion which combine the tasks of scene completion along with predicting a label for the completed scene.

In [4], the authors propose a hierarchical decoder with semantic and geometric supervision at every level for semantic scene completion as seen in Figure 2.25. The network operates on voxel grids with input samples as truncated signed distance field (TSDF) with truncation upto 3x the voxel grid size. The input partial scan in TSDF representation is divided into eight voxel groups and the network trains in

an autoregressive manner. The training for each group occurs parallely with input from the previous predictions of semantics and geometry. For geometric reconstruction, the network is trained to predict a target truncated unsigned distance field(TDF). The usage of implicit functions such as TSDF and TDF is due to its aid in training to produce better gradients compared to discrete occupancy grids. The signed fields presented as input help to distinguishing between voxels in front or behind the observed surface, which encodes the points in space within and out of reach from hte cameras visibility. The output is TDF instead of TSDF since it's mostly infeasible to obtain ground truth TSDF for training due to scarcity of watertight models when using synthetic 3D data.

In [41], the network is also trained for large scale semantic scene completion. Unlike the previous works, this network works on predicting a continuous representation rather than voxel grids. The authors make use of local deep implicit functions (LDIF) for large scale scene completion. They propose a non-unified loss for semantics and geometric reconstruction, with a cross-entropy loss for the output semantic classification vector and a binary cross-entropy on the same output classification vector. The encoder creates a bird's eye-view feature map of the point cloud to be processed. The latent representations are made up of the feature maps that are encoded in three different resolutions. A query point in 3D space is used along with the latent feature to make a final prediction of not only occupancy but also of classification. It is to be noted that the input to the encoder which is a fixed size feature map ideal for usage with 2D convolutions as in images but the output generation, the completed point cloud is in the $\mathbb{R}^3$ domain. Although LDIFs were first introduced only for shapes, this work extends its usage for large scale point clouds and scenes.

The works discussed so far do not take as input or predict raw point clouds without any additional transformation. In addition to this, such networks have only been studied for the task of large scale scene completion and hence need further investigation into its adaptation and application for the task of semantic shape completion.

3

Background

In this chapter, we look into how we systematically approach the problem of point cloud segmentation by analyzing a chosen baseline network architecture, a benchmark dataset and thereby the chosen corresponding metrics. Along with analyzing the baseline network, a new strategy for testing networks with partial point clouds is also proposed in the following sections.

3.1 DGCNN

The baseline network chosen for the work on complete point cloud segmentation is the one proposed in [58], **dynamic graph convolutional neural network (DGCNN)**. The network was proposed concurrently to PointNet++ as an effective approach for utilizing local geometric features for point cloud analysis after the initial approaches which only made use of pointwise features or used different representations other than raw point clouds. The network turned out to be a strong candidate for thus work for the following reasons:

- State-of-the-art performance on ShapeNet-Part dataset at the time of start of this project
- Consumes raw point clouds as inputs and makes point-wise prediction for segmentation tasks
- Offers scope for adopting proven deep learning techniques
- Offers scope for additional improvements on existing network architecture modules

The network architecture consists of an input input spatial transform net inspired by PointNet, a series of EdgeConv layers to perform graph convolutions, skip connections to concatenate all the features form previous EdgeConv layer outputs to be transformed through a shared MLP and aggregated by a symmetric function, such as a max-pooling in order to produce the global feature as well as to handle permutation invariance, which is directly used for classification, or which is later concatenated along with the previous EdgeConv layer outputs which is used for final predictions by a series of shared MLPs for segmentation tasks as seen in Figure 3.1. We shall discuss the network modules and their functions in the following section.

Spatial Transformer Network (STN): First introduced for 2D image processing, this module was proposed for point cloud processing tasks in [38] and has been adapted by various future works

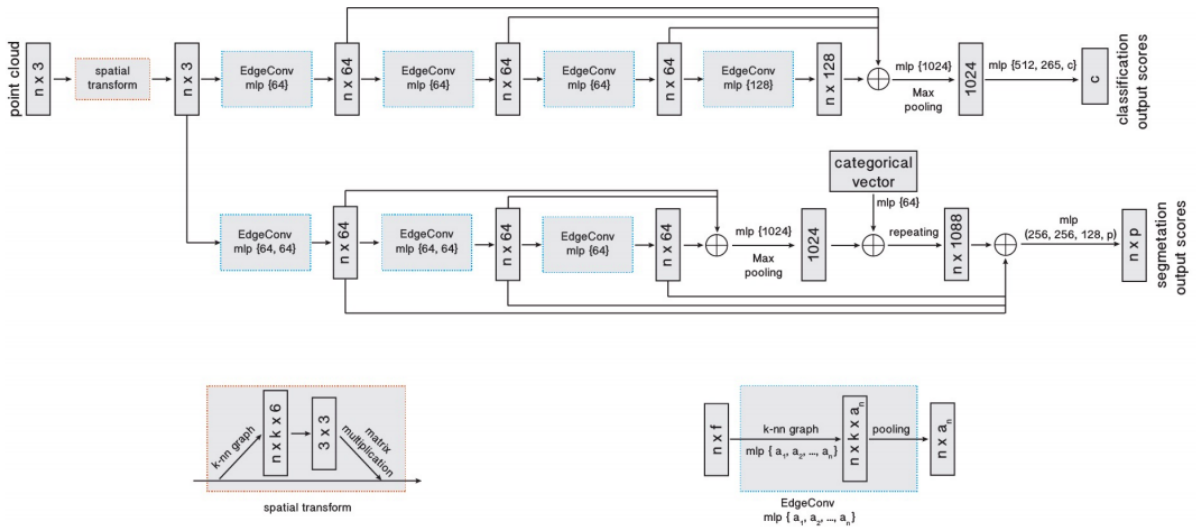


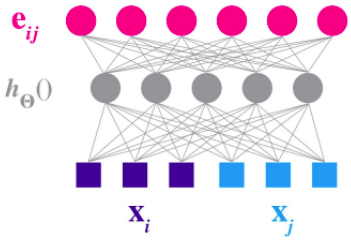
Figure 3.1: **DGCNN**: Network Architecture. Image adopted from [58].

succeeding it. The module performs the task of learning a kxk matrix in order to perform geometric transformations on the input point cloud. The authors hope that this module helps in improving the performance by learning to project the input point cloud into certain canonical space which make it easier for the network to learn and more importantly the module helps in achieving invariance to certain geometric transformations like rotations. This module can be a useful addition ot the network used for training them for various downstream tasks. If trained along with a regularization loss, the network can be trained to produce a rigid transformation like rotation or the network tends to produce affine transformations which in addition to rotation can also produced some undesired scaling. In this network, the STN is not trained with any regularization loss. The input to the module unlike in PointNet, is a kNN graph of the neighborhood around each point and the output is a 3x3 square matrix which is later used to transform the input.

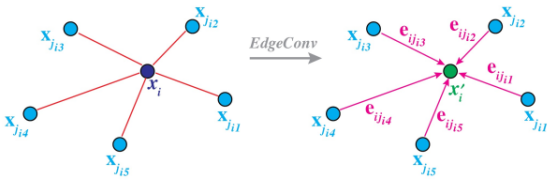
Edge Convolution: The network consists of three EdgeConv layers stacked sequentially. The EdgeConv layers perform the core function of graph convolution as proposed by the authors. Given a F-dimensional point cloud with n points, $X = x_1, x_2....x_n \subseteq \mathbb{R}^F$, where each input layer features are given by the 3D coordinates. The local point cloud structure is captured by constructing a directed , $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, .., n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ are the defined vertices and edges. In this network, a kNN graph of X in $\mathbb{R}^F$ is considered. The edge features are defined by:

$$e_{ij} = h_\theta(x_i, x_j - x_i) \tag{3.1}$$

where θ is the set of learnable parameters of the non-linear function h, which transforms the features to F' dimensions. In order to aggregate the edge features from each of the edges in the kNN graph, a



(a) Edge feature learning when absolute coordinates of neighbors x_j are used. Note that only relative coordinates $(xj - x_i)$ are used in the network.



(b) Edge feature aggregation: The edge features are aggregated via max-pooling

Figure 3.2: **Edge Convolution**: The proposed local feature learning and aggregation module. Images taken from [58].

symmetric function such as a max pooling is used:

$$x'_i = \max_{j:(i,j)\epsilon\mathcal{E}} h_\theta(x_i, x_j - x_i) \tag{3.2}$$

where x'_i is the new features assigned to each point in the point cloud aggregated from its edge features, (x_i, x_j) are the pair of absolute features of the query point and features of the neighboring point. This module is implemented by using a couple of shared MLP layers within each EdgeConv block along with the above discussed aggregation strategy.

Dynamic Graph Update: One of the main contributions of this work is the dynamic graph reconstruction after each edge convolution layer, which the authors suggest makes the receptive field as big as the size or diameter of the point cloud. After each EdgeConv operation, there is a different graph recomputed based on the feature space kNN as illustrated in Figure 3.3. At each layer, a new graph $\mathcal{G}^l = (\mathcal{V}^l, \mathcal{E}^l)$ is recomputed based on the feature space distance. Basically, the network is expected to learn to how to construct the graph at each layer.

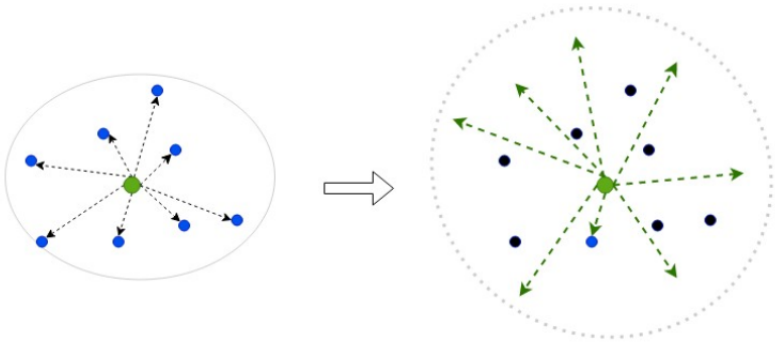


Figure 3.3: **Dynamic graph update**: Spatial neighborhood to feature space neighbors.

Translation-Invariance: As the authors suggest, the network can be made to have full translation if we only consider the relative features for edge convolution operation. The network is described to have *partial translation invariance* in the proposed version. The network is completely translation invariant when the EdgeConv layer at the input takes in only relative features as input as below:

$$x_i' = \max_{j:(i,j)\epsilon\mathcal{E}} h_\theta(x_j - x_i) \tag{3.3}$$

Feature Learning: The image in Figure 3.4 shows the feature transformation form the input layer to the last EdgeConv layer of the shape segmentation network. As the network goes deeper, the feature space distances between points belonging to the same part category tends to get pulled closer as indicated by the yellow points taking the red point as a reference. As indicated in [59], the deeper the graph convolution network with static graphs, the closer the features of the center point and its geometric neighbors tend to get, which makes it difficult to distinguish between parts belong to different category but are close together in the input $\mathbb{R}^3$ space. The dynamic graph update pursues to learn to build its own graph in each layer and hence minimizes the threat of static graphs as mentioned earlier.

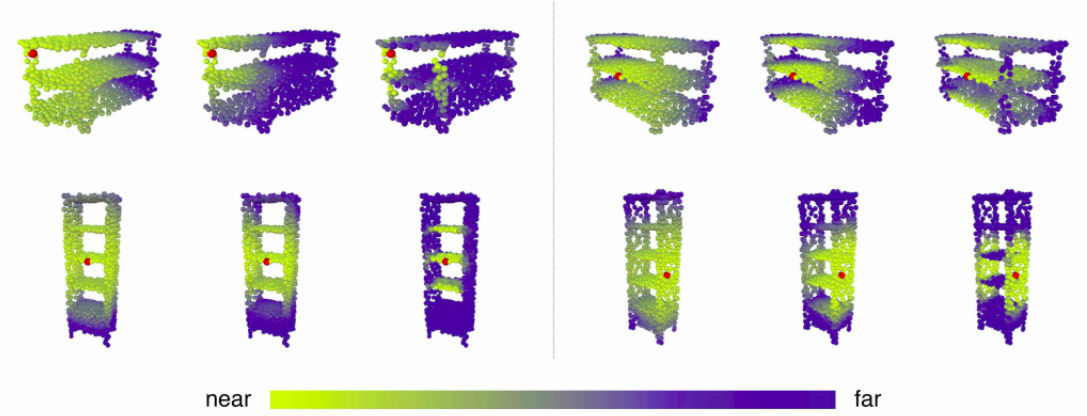


Figure 3.4: **Feature Learning:** Yellow points indicate closeness to the selected red point. The first column represents the euclidean space distance in $\mathbb{R}^3$, the middle indicates the distances after the spatial transform layer and the final column indicates the distances after the final EdgeConv layer. Image taken from [58].

Segmentation Head: The segmentation head of the network consists of a series of shared MLPs with the final predictions given by only a *single head* which output the softmax predictions for the possible set of part categories. The head takes as input the global features tiled and concatenated along with the local features for each point obtained from the previous EdgeConv layers and utilizes both the global and local information in order to make the final predictions for individual points.

Object Categories	No. of Parts	No. of Samples
Airplane	4	2690
Bag	2	76
Cap	2	55
Car	4	898
Chair	4	3758
Earphone	3	69
Guitar	3	787
Knife	2	392
Lamp	4	1547
Laptop	2	451
Motorbike	6	202
Mug	2	184
Pistol	3	283
Rocket	3	66
Skateboard	3	152
Table	3	5271
Total	50	16,881

Table 3.1: ShapeNet-Part Samples

3.2 Dataset

The dataset selected for this work is the popular part segmentation benchmark dataset, *ShapeNet-Part*. This is a subset of the collection proposed in [63], which in turn is a subset of the wider *ShapeNetCore* which consists of more than 30,000 samples. The ShapeNet-Part dataset consists of:

- 16,881 examples
- 16 object categories
- 50 part categories
- Each object is associated with 2-6 parts
- Objects belonging to the same category might contain different part combinations

The samples in the dataset are generated by *uniformly sampling* points from the annotated 3D CAD models. Each point is described by its 3D coordinates and its associated normal vector described by its x,y and z components. In this thesis, we only aim to take into consideration the 3D coordinates of individual points and do not take as input the normal vectors. The dataset is a mix of common household objects like mugs, tables, chairs etc., to other objects like airplanes and motorbikes.

In this thesis, we will not only be using the samples obtained directly from the dataset which are in the form complete, oriented and normalized point clouds but we will also need to generate partial point clouds to train and test our architectures. The following section provides some examples for both these cases and the partial point cloud generation method used in this work.

The model architectures proposed so far for the task of point cloud segmentation predominantly use

complete point clouds of 3D objects for training and testing. Typically, the points are also *oriented* and *normalized* to fit into a *unit sphere* to be fed into the network. In addition to the task of part segmentation, the network has to deal with additional challenges posed by the dataset itself due to the imbalance between number of samples for training with respect to object categories, part categories and part combinations.

3.3 Evaluation

This section is divided into two parts where we discuss all the evaluation metrics used at various points of this thesis and the evaluation strategies used for testing the networks and in particular we propose a new strategy for generating and testing partial point clouds in the second subsection.

3.3.1 Metrics

This section introduces the evaluation metrics used to evaluate the networks performance throughout this work. Metrics discussed here will be used when necessary and are not always presented for all the results.

Intersection Over Union (IoU): The IoU is a metric based on the Jaccard similarity coefficient [18] and is used to measure the similarity between two finite sets of elements. In our case, we use this to measure the similarity between the ground truth set of points and a predicted set of points in the point cloud assigned to a particular part category.

If G is the ground truth set of points belonging to a part category, it can be represented as:

$$A = \{x | x \in G, G \subset \mathbb{R}^3\} \tag{3.4}$$

If P is the predicted set of points assigned to a particular part category, it can be represented as:

$$B = \{x | x \in P, P \subset \mathbb{R}^3\} \tag{3.5}$$

Given A and B, the Intersection over Union can be calculated as below,

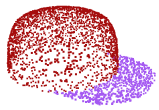
$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{3.6}$$

Here, the intersection of A and B represents the number of points correctly predicted as belonging to a particular part category and the union of A and B represents the total number of points belonging to both the sets.

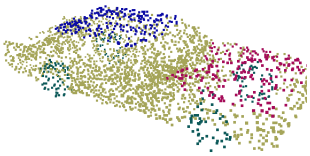
The IoU as you can guess is calculated for only one part. Since each object has a number of parts in it, and the entire dataset has a number of object categories in it, we look into the actual metrics considered for this thesis:

(a) Airplane

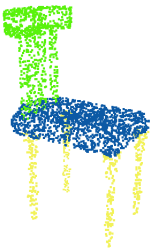
(b) Bag



(c) Cap



(d) Car



(e) Chair

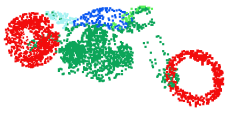
(f) Earphone

(g) Guitar

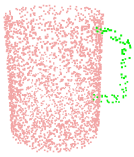
(h) Knife

(i) Lamp

(j) Laptop

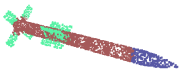


(k) Motorbike



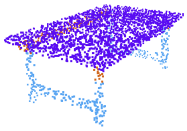
(l) Mug

(m) Pistol



(n) Rocket

(o) Skateboard



(p) table

Figure 3.5: **ShapeNet-Part** - Complete Objects

- mean per instance IoU (miIoU or mIoU): The mIoU of a single object is the mean of all the IoUs of individual parts in an object. Since not all the parts belonging to a particular object category is always there, we set the IoU to 1.0 when both the ground truth and predicted set of points is zero. The average of mIoUs of all the objects in the dataset is actually the mean per instance IoU(miIoU), but we still stick to the convention of earlier works and denote it as mIoU rather than miIoU.
- mean per category IoU (mcIoU): This is the average of all mIoUs calculated separately for each object category.

Accuracy: Accuracy of a segmented object is the ratio of correct number of predictions to the total number of points in the given instance. Note that a high accuracy does not always correspond to a high segmentation performance or IoU. For this reason, accuracy is not the primary metric used in this thesis.

Precision: Precision can be defined in terms of relevance, where predicting a certain part in an object might be of relevance. Also, known as the positive predictive value it is the fraction of correct predictions of a particular part among all its positive predictions for that particular part.

$$Precision = \frac{TP}{TP + FP} \tag{3.7}$$

where TP and FP are True Positives and False Positives respectively.

Recall: Also known as the sensitivity, it gives the fraction of correct predictions made for the part with respect to all the ground truth part instances in the data.

$$Recall = \frac{TP}{TP + FN} \tag{3.8}$$

where TP and FN are True Positives and False Negatives respectively. The illustration of the two metrics can be observed in Figure 3.6.

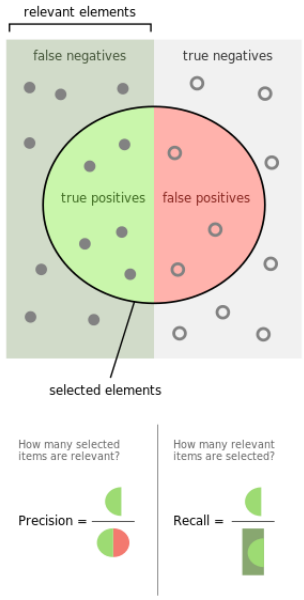


Figure 3.6: Precision and recall. Image adopted from [54].

3.3.2 Evaluation Strategies

In this section, we will look into evaluation strategies for testing the trained model. We can divide the section into parts, one for testing on complete point clouds and another for testing on partial point clouds where we elucidate our newly proposed strategy.

Complete Point Clouds

For testing segmentation performance on complete point clouds, we do not need to consider any different strategy other than the one already proposed in [58]. It can be worth noting that DGCNN does not involve any voting strategy owing no stochastic component in the architecture modules. On the other hand, networks like PN++ and KP-Conv utilize some sort of voting strategy thanks to the stochasticity with respect to the subsampling procedures. In [31], the network makes use of a voting strategy based on scale augmentations on the test set. In the coming sections, we do not make use of any *voting strategy* for the tests with complete point clouds unless otherwise explicitly indicated. The evaluation metrics that are presented are:

- $mIoU$ - mean Intersection over Union
- $mcIoU$ - mean categorical Intersection over Union

Partial Point Cloud

In this section, a new strategy for *testing* the architectures for robustness to partial point clouds is proposed. A testing strategy for partial data is not straightforward as it seems, hence we will discuss in detail the steps involved in generating partial data to create the test set. As done in [58], the main idea behind this strategy is to test the network for different percentages of points given a complete point cloud. It can be noted that a couple of other methods to generate part segmented partial point clouds have been adopted in [65] and [68]. The one in [68], where occlusions are generated by choosing points lying to one side of a randomly oriented plane passing through the objects center, is comparable to the proposed method. There is no clear consensus yet on testing on partial data. We hope that by following the proposed method, *viewpoint based occlusions* such as *self-occlusions* and occlusions due to external factors will be covered.

Part Segmented Partial Point Cloud Generation: The approach can be roughly viewed as slicing a point cloud for defined proportions by moving a plane perpendicular to the chosen main diagonal of a cube enclosing the unit sphere in which the object is centered at. Given an oriented, normalized point cloud as input, the approach consists of three main components:

- **Slicing Plane Direction**: The slicing plane is chosen based on the directions of one of the eight corners of a cube. The plane is oriented normally to the direction of the corner with respect to the center of the cube.
- **Occlusion Percentage**: The input points are projected along the normal direction of the plane or the direction of the chosen diagonal and depending the selected *occlusion quantile*, an equivalent number of points are dropped from the point cloud, which can be interpreted as points belonging to one side of the cutting plane being omitted. An occlusion quantile 0.0 indicates no points being

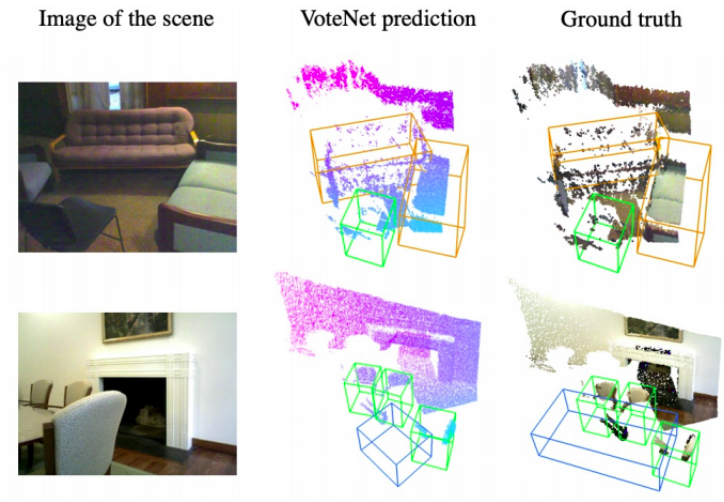


Figure 3.7: **Amodal Object Detection**: One could notice the inaccurate scale of the orange(long sofa) bounding box in the first row and the hugely inaccurate prediction for the blue box(table) in the second row, in the image adopted from [37]

dropped and an occlusion quantile of 0.9 indicates 90% of points being dropped from the given point cloud.

- **Recentering and Renormalizing Partial Object**: The occluded object is then recentered and then normalized back to the unit sphere before being fed into the network. This is needed in order to mimic the real world scenario where the center and the bounding region representing scale of the detected occluded object is most often not well known.

Reason behind Recentering and Renormalizing: One may question the need for recentering and renormalizing the occluded point cloud since other methods like in [68] describe the performance of their networks without this procedure. We ascribe the reason to recentering and renormalizing the occluded point cloud to *inaccurate bounding box predictions* from concurrent state-of-the-art 3D object detectors. In the real world setting, it can be expected that a 3D object detector would provide the segmentation network its input. When detecting occluded objects, the problem of predicting the instance location and bounding box for the complete object including the portion that is not directly perceived by the sensors is called as *Amodal object detection*. The performance of the amodal object detection module directly influences our decision on recentering and renormalizing. The typical evaluation metric for such networks is mAP(mean Average Precision). The following reasons add to our decision:

- mAP is calculated based on a mIoU threshold for qualifying bounding box predictions to be accepted or not for the evaluation.

- The mIoU is an indicator of how well the bounding box fits the underlying object

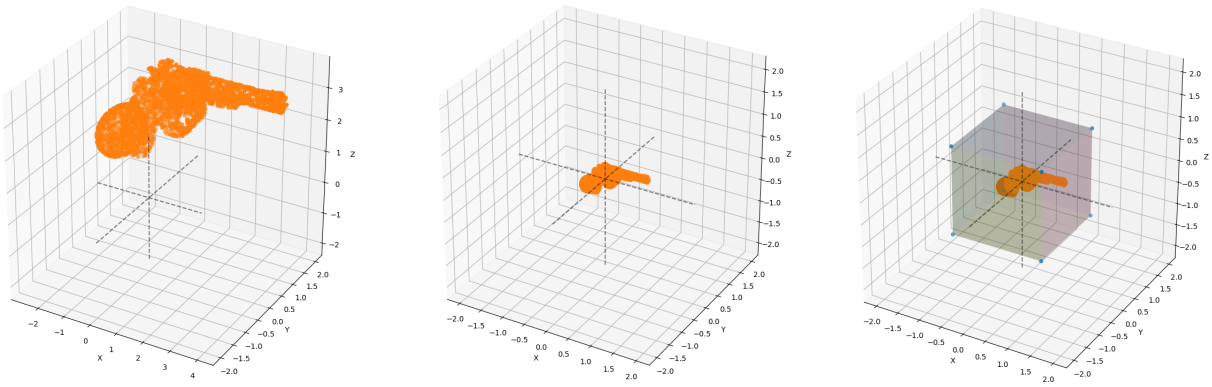
- The widely accepted mIoU threshold has so far been too low for the benchmark tests, such as 25% in [37].
- In addition to the above, the localization accuracy which helps with the centroid information is also deemed to be poor.
- Owing to the very low threshold for mIoU, we cannot be confident enough to assume that the object extent is fully known and in turn also the centroid information.

Hence, if the object detector provided us with accurate information regarding the bounding box, one could renormalize the partial clouds with respect to it but now we are pushed to renormalizing and recentering only with the partial cloud information.

Generation Sequence

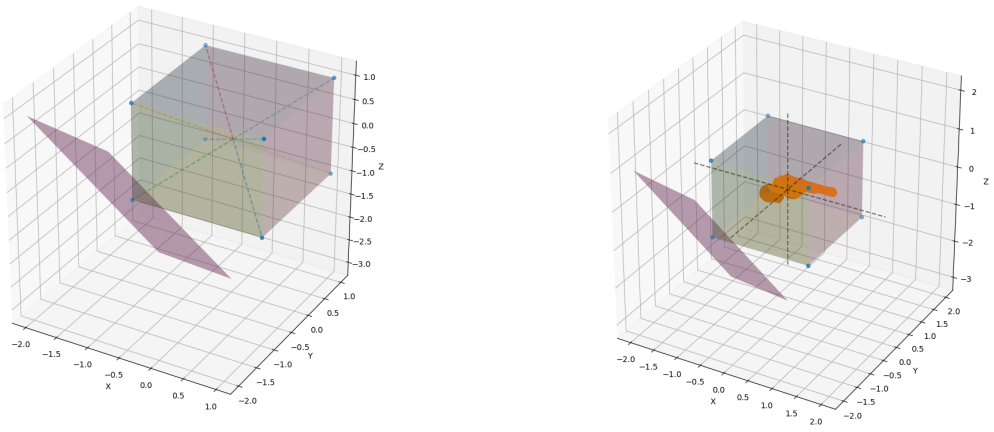
- Raw Input: The raw input is typically not normalized or centered at zero as shown in Figure 3.8a.
- Normalized and Centered Input: The input point cloud is normalized and centered at mean zero. As shown in the Figure 3.8b, the scaling and alignment along with the reference coordinate frame can be seen with respect to the raw input.
- Bounding Cube: The cube enclosing the unit sphere to which the normalized point cloud is fitted into can be seen in Figure 3.8c.
- Slicing Plane Selection: The normal direction of the slicing plane is chosen along each of the eight corners of the cube, creating eight samples for each sample. The slicing plane along [-1,-1,-1] is seen in Figure 3.9.
- Slicing Point Cloud: The point cloud is sliced according to the desire occlusion quantile as seen Figure 3.10a.
- Renormalizing and Recentering: The final output of the generation sequence where the partial point cloud is recentered and renormalized to the unit sphere as seen in Figure 3.11.

The example partial point clouds can be seen in Figure A.1 and A.2.



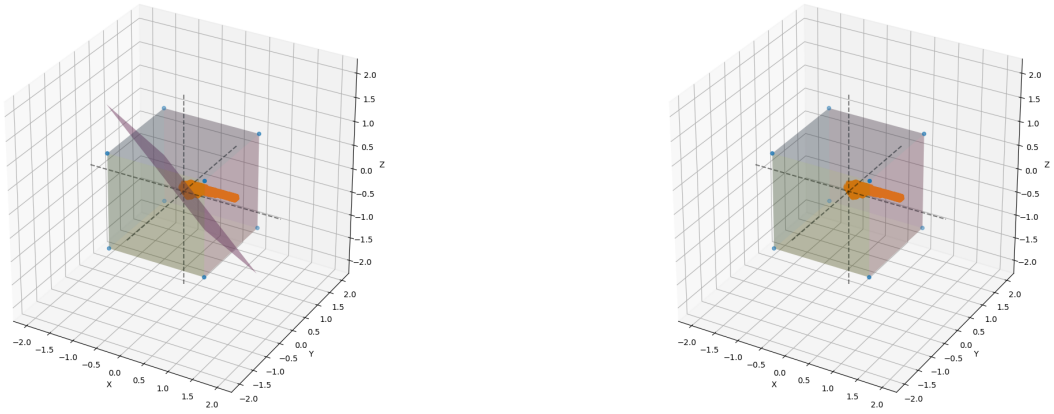
(a) **Raw Input** (b) **Standardization** (c) **Bounding Cube**

Figure 3.8: **Input Standardization:** Normalizing input to unit sphere and bounding cube visualization.



(a) **Selecting Main Diagonal Corner at [-1,-1,-1]** (b) **Slicing Plane Orientation**

Figure 3.9: **Slicing Plane Selection**: One of the main diagonal corners is chosen for deciding on the slicing plane orientation.



(a) **Slicing for 50% occlusion** (b) **Partial Point Cloud**

Figure 3.10: **Slicing Operation**: According to the chosen occlusion quantile, the slicing plane is positioned along the chosen main diagonal to provide the partial object.

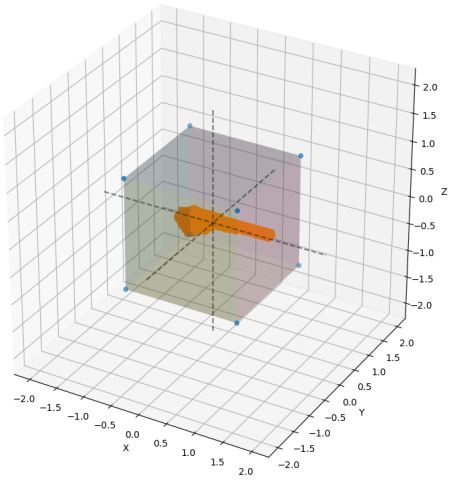


Figure 3.11: **Partial Data:** Final standardized partial object with 50% of original points. Notice the scaling and translation of the partial object due to the centering and normalizing standardization steps.

Occlusion Metrics

The evaluation metrics used is $mIoU$ with adaptation for different occlusion quantiles as below:

$$mIoU_{qc} = \frac{1}{N_p} \sum_{i=0}^{N_p} mIoU_{qci} \tag{3.9}$$

where $mIoU_{qc}$ is the mIoU w.r.t a particular quantile for an object category c, N_p is the number of cutting planes which is 8, and $mIoU_{qci}$ is the mIoU for a particular occlusion quantile q generated by a particular cutting plane i for an object category c.

$$mIoU_q = \frac{1}{N} \sum_{c=0}^{N_c} W_c * mIoU_{qc} \tag{3.10}$$

where $mIoU_q$ is the mIoU for a particular occlusion quantile for the entire dataset, N_c is the number of categories, W_c is the number of samples of each category and N is the total number of samples.
Even though mcIoU is not typically considered a reliable measure, and not presented for all experiments in this work, considering the imbalance within the dataset and is not given the same importance as mIoU, for certain comparison purposes it might come in handy to know its measure. mcIoU is the average across all categories:

$$mcIoU_q = \frac{1}{N_c} \sum_{c=0}^{N_c} mIoU_{qc} \tag{3.11}$$

where $mcIou_q$ is the average mIoU for a particular quantile q across all object categories, and N_c is the number of object categories which is 16 in this case.

3.4 Baseline Network Performance

In this section, we will analyze the performance of the chosen baseline network in order to propose hypothesis and test them in the following chapters. In order to analyze the network, we will have to decide on the training and evaluation setup.

Network Architecture

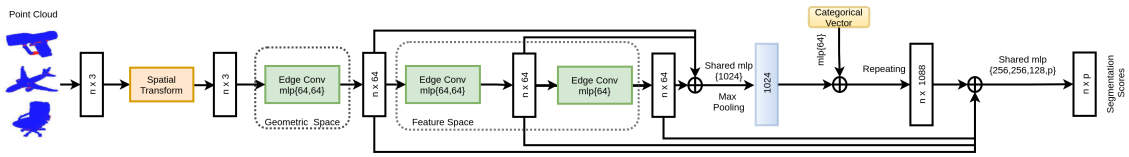


Figure 3.12: **DGCNN Segmentation Architecture**

Experimental Setup 1	
Batch Size	24
Learning rate	0.001
Optimizer	Adam Optimizer
LR Decay rate	0.5
Decay Steps	200000
LR Clip	1e-5
BN Decay	0.5
BN Decay Rate	0.5
BN Decay Steps	400000
CPU	Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz
GPU	2x NVIDIA Titan RTX (Turing)
RAM	24GB
DL Framework	TensorFlow 1.13(GPU)

Table 3.2: DGCNN Baseline - Training Hyperparameters and Setup

Training and Evaluation Setup

The below setup is maintained the same for all complete point cloud segmentation training experiments unless and until otherwise mentioned.

Segmentation Experimental Setup - 1

The details discussed below will be used in order to train and test the network, hence any metric used such as inference time should be considered with respect to the provided information below.

- Data Split: Train-Validation-Test split is the same as in [58]. We use the validation data to avoid over-fitting and carry out early stopping when necessary.

- Data Augmentation: During training and testing, no data augmentation strategy is applied.

- Network Hyperparameters: The network has one hyperparameter k, which decides the kNN of the EdgeConv layers. Here, k is set to 20.

- Input points: The number of input points is set to 2048 with each point carrying only the x,y, and z coordinate information and does not use normal information.

- Train Setup: The following table 3.2 lists the relevant training parameters like Learning Rate(LR) and Batch Normalization(BN) decay rates.

Testing on Complete Point Clouds

As seen in the Figure 3.14, the network yields a performance of:

- mIoU = 85.2%
- mcIoU = 82.2%

The performance measure of mIoUs of the baseline network reveals the following details:

- The mIoUs across object categories varies significantly from a maximum of 95.7% for laptop category to 63.5% for rockets.
- With the same number of parts, the categories with lesser number of examples lag behind in performance(i.e lamp compared to chair).
- Object categories with lesser number of parts generally perform better than the rest.
- Object categories with the least(2) number of parts don't necessarily perform better than objects with two or more parts more than it, as in the case of bags and airplanes.
- Objects with fewer number of parts and large number of examples still lag behind in performance due to varying part combinations from one object instance to another, as in the case of tables.
- Rocket category has the least performance, which has all the disadvantages discussed above, more than two parts, different part combinations and very few samples.

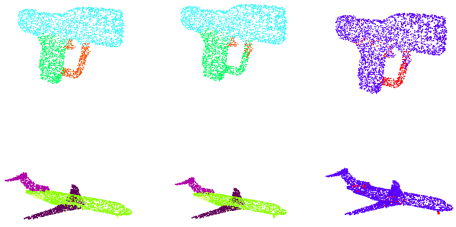
Failure Modes - Complete Point Clouds: As shown in the Figure 3.13, the failure modes can be broadly classified into five categories. Some problems like boundary distortion is also common among other segmentation techniques, whereas problems with temporary parts is something more specific to the challenge in place here. Objects with very few samples suffer heavily which is revealed quite obviously looking at the qualitative results as shown in the Figure 3.13.

Testing on Partial Point Clouds

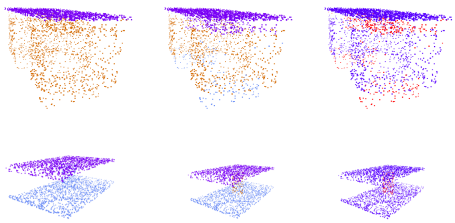
While training on complete point clouds and testing on partial point clouds, the network achieves the performance as shown in Figure 3.15. The results reveal the following:

- Steep decline in performance as the occlusion quantiles increases and reduces to a minimum of 36.5% at 90% occlusion.
- The decline in performance is almost linear to the amount of occlusion present in the objects. This makes for a completely undesirable performance by the current network architecture.

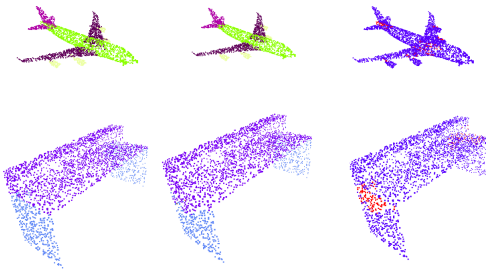
The Figure in 3.16 reveals the poor performance of the network. The network virtually breaks down when having to deal with partial data even at 50% occlusion and there is no case of discussing any *failure modes* here. In some cases, as with the earphones, the entire set of point s are misclassified to a part which is not present in the object taken as input. In some case as with the airplanes,the entire engine is misclassified as belonging to the wings which usually doesn't happen when dealing with complete point clouds. Such predictions which make no semantic sense is the results of the design of the network and training procedures which do not take into account the real world scenario of having to deal with partial point clouds in a frequent manner.



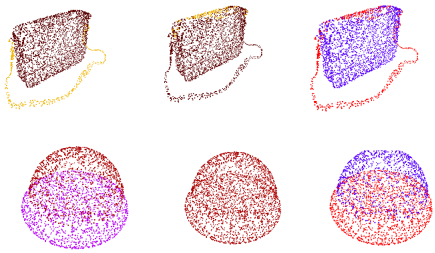
(a) Small parts with fewer points being overwritten by large parts. Notice front wheel in airplane.



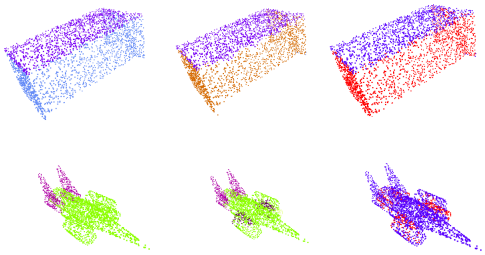
(b) Spurious predictions of absent parts and temporary parts being completely omitted in predictions.



(c) Boundary points misclassified.

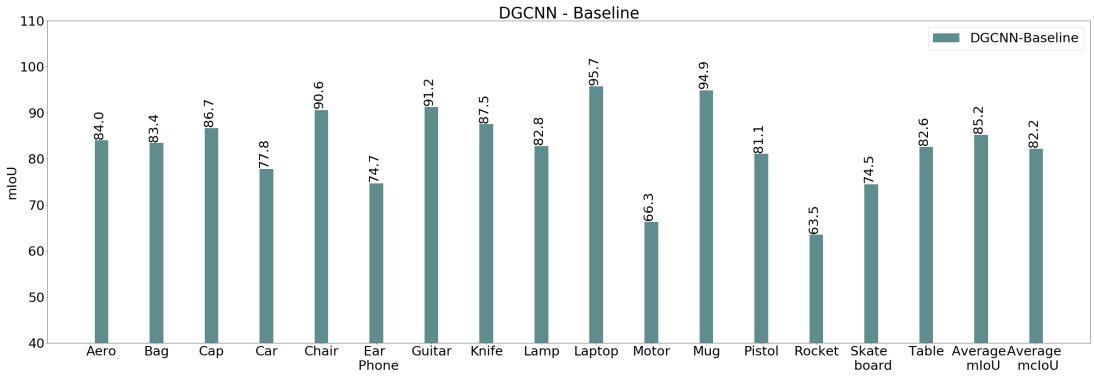


(d) Poor segmentation of objects with very few training samples.



(e) Ambiguous parts which even humans tend to misclassify.

Figure 3.13: **Failure Modes**: Classification of popular failure modes exhibited by the baseline network. Each row within images is arranged in the following order from left to right: Ground truth, Prediction and Misclassification(red points).



	Aero	Bag	Cap	Car	Chair	Ear Phone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate board	Table	mIoU	mcIoU
# Samples	2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271	16881	16881
DGCNN-Baseline	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	85.2	82.2

Figure 3.14: DGCNN Baseline Performance on Complete Point Cloud Segmentation.

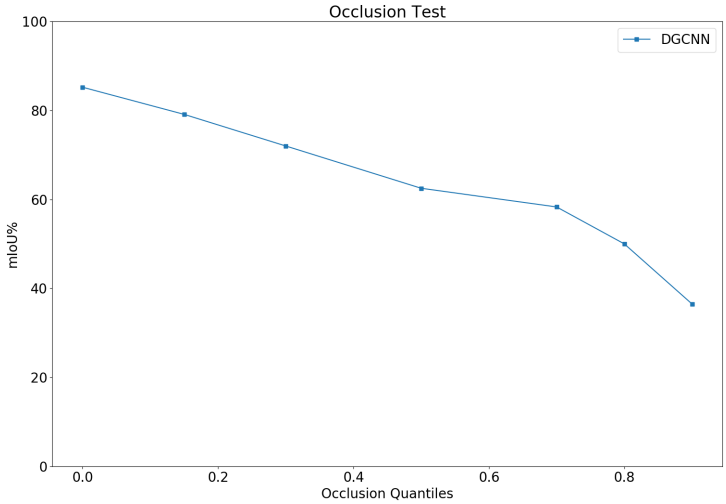


Figure 3.15: DGCNN Baseline Performance on Partial Point Cloud Segmentation.

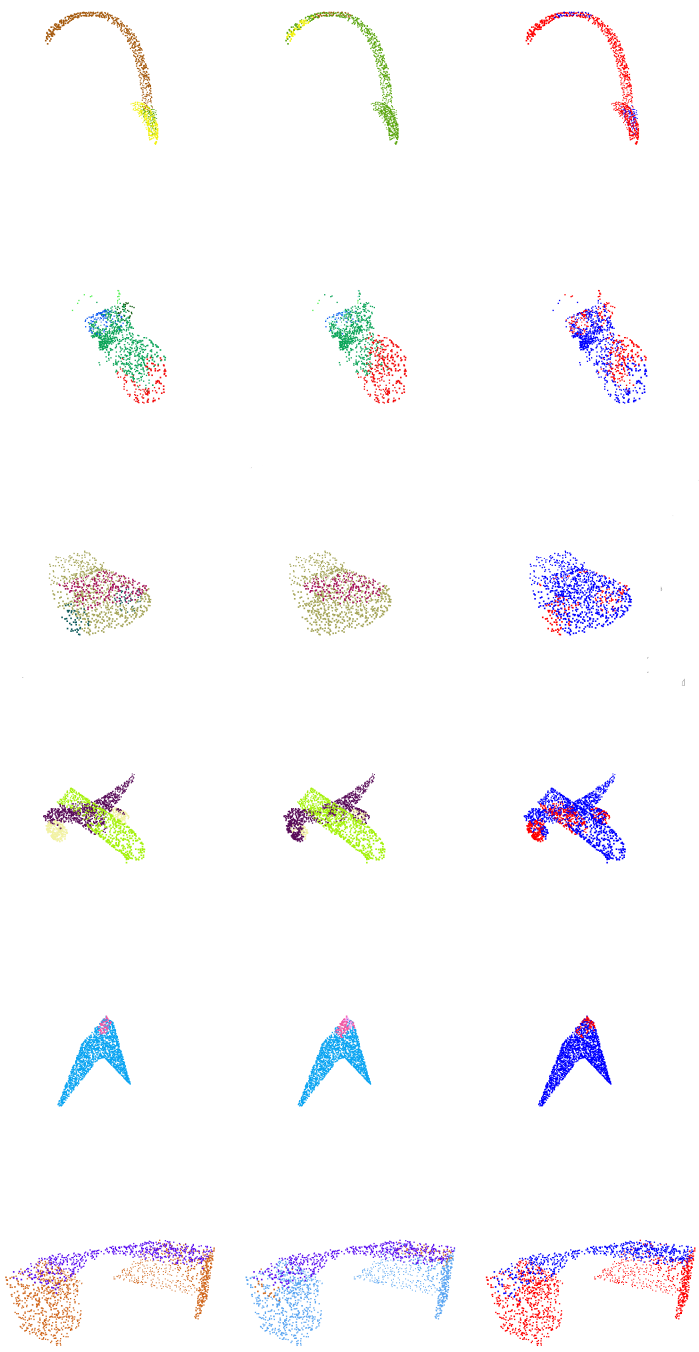


Figure 3.16: Qualitative results: DGCNN Baseline Performance on Partial Point Cloud Segmentation. GT, Prediction and Misclassification(red points) of samples presented in columns from left to right.

4

Methodology

This chapter elaborates on the various ideas proposed by this work. Not only do we propose solutions to counter the exposed failures of the baseline network form the previous chapter, we will also look into what changes can be made to improve the overall performance in other ways as well. The chapter is broadly divided into three sections each for complete point cloud segmentation, partial point cloud segmentation and semantic shape completion. Within each section, we define the hypotheses behind the proposed solutions which will eventually be put to test in the following chapter. To the best of our knowledge, semantic shape completion on 3D object point clouds is the first work of its kind.

4.1 Complete Point Cloud Segmentation

In this section we will discuss in detail what changes have been proposed to improve the segmentation performance on complete point clouds. The solutions that are proposed deal with architecture modifications at almost every module level that constitutes the network. We will look into the specifics of what fundamental operational blocks of the network can be maneuvered in order to garner gain improvement of overall performance and to counter the failure modes of the network.

4.1.1 Going Deeper with Graph Convolutions

The vanilla version of DGCNN consists of a spatial transform network which takes in the input and transforms it, before it is passed through three EdgeConv layers and the segmentation head. One of the main characteristic of this architecture compared to other state-of-the-art architecture proposed during the same time, is that this is not a hierarchical approach. Most hierarchical architectures have been inspired by PointNet++ which was the pioneer for such designs for processing point clouds. Hierarchical architectures involve subsampling of the input points until it reaches a limited number of points or to a single global point representing the entire point cloud. Even though hierarchical approaches have their own advantages of computational efficiency due to subsampling, it inherently limits the possibility to extend state-of-the-art CNN based deep architectures proposed for image processing, such as ResNet[12] and DenseNet[17]. On the contrary, networks like DGCNN which do not use any kind of subsampling offers the scope for extending proven techniques to make the networks substantially deeper.

It is natural to assume that, deeper the network better the performance, since more learnable parameters

are involved, but its not that straightforward as it seems. The main motivation behind ResNet and DenseNet is to counter the problem of degradation of performance in CNNs as the depth of such networks is increased beyond a certain number of layers. In ResNet, they redefine the convolutional layers as *residual learning functions*. This makes the optimization process easier for the network. In DenseNet, the network alleviates the problem by encouraging *features reuse* where each layer utilizes the features from all the previous layers. In this way, they also alleviate the vanishing gradient problem and provide diverse inputs to each layer which, as the authors claim, increases efficiency. It is clear that both the solutions not only enable the network to be trained with substantially deep layers, but also add onto other indirect benefits which increases the efficacy of the convolution operation, hence we can expect the EdgeConv operation to be more effective this way.

The success of the deeper networks with different methods such as residual learning and feature reuse, makes for a promising direction to test our network with the same incorporated principles. Thus, this work considers modifying the EdgeConv layer with both the ways in which we can enable the network to go deeper as shown in Figure 4.1.

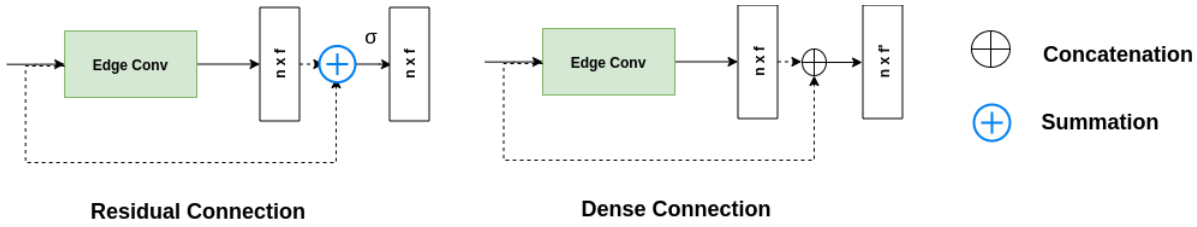


Figure 4.1: Proposed modifications to the EdgeConv layer output in order to facilitate addition of more layers for a deeper network.

More formally, the features assigned to each point at the output of each Edge Conv layer can be defined as:

$$\hat{x}_i^l = \max_{j:(i,j)\epsilon\mathcal{E}} h_\theta(x_i^{l-1}, x_j^{l-1} - x_i^{l-1}) \tag{4.1}$$

with *dense connection* the operation can be redefined as:

$$\hat{x}_i^l = \max_{j:(i,j)\epsilon\mathcal{E}} h_\theta(x_i^{l-1}, x_j^{l-1} - x_i^{l-1})) \oplus x_i^{l-1} \tag{4.2}$$

and with *residual connection* the operation can be redefined as:

$$\hat{x}_i^l = \sigma(\max_{j:(i,j)\epsilon\mathcal{E}} h_\theta(x_i^{l-1}, x_j^{l-1} - x_i^{l-1}) + x_i^{l-1}) \tag{4.3}$$

where $\hat{x}_i^l$ is the output features at the end of an EdgeConv layer with input x_i^{l-1} belonging to the same point and x_j^{l-1} the features of its neighborhood points. The symbol $\oplus$ and $+$ indicates concatenation and summation respectively. σ represents the activation function used and h_θ, a non-linear function with trainable parameters θ. The amount of impact that such a modification to the network could have is

difficult to hypothesize but the choice of connection to adopt will be rightly made after evaluating the proposed solutions. Thus, as discussed above this brings us to the first hypothesis this work puts forth in this section, Hypothesis H_1.

Hypothesis H_1: If deeper networks lead to performance improvements with convolutions on images, they should also aid in improving performance with convolutions on point clouds.

4.1.2 Edge Classification as Auxiliary Supervision

Typical graph neural networks are trained to perform node prediction and also link prediction depending on the task at hand. Deploying graph CNNs for point cloud processing provides with the opportunity to not only predict nodes, in our case the individual points but also make predictions on edges between point-pairs. The vanilla version of DGCNN as we know it only makes predictions on individual points which are taken as the segmentation scores with probabilities of the point being classified into each of the part categories. This begs the question of using edge prediction to enhance the networks learning capabilities.

The work in [19] proposes a hierarchical edge prediction branch parallel to a hierarchical point prediction branch. Both the edge prediction and point prediction also share features in order to enhance message passing. The edge prediction branch is basically trained to predict whether two points making up the edge belong to the same class or not. The authors hypothesize that this enhances the discriminative capability of the network by steering the point prediction branch to produce features that are distinct from one class to another. This kind of learning aims to make features of points belonging to the same class more similar than to points from other classes and in turn produce distinguishing features for different classes. In another work in [60], the authors propose a *region similarity loss* with the same goal of increasing similarity between features of points belonging to the same class. Here, distinguishing feature points are first identified after the prediction layer and then the kNN features of points with the same class labels neighboring the point carrying the distinguishing feature is use as input to an auxiliary loss function which penalizes based on a cosine similarity loss. The authors claim that such supervision can also help with boundary points labeling distortions which is one of the common failure modes with segmentation networks. Such methods in a way can also be viewed as a means to mimic *contrastive learning*.

Inspired by the previous works, this work proposes a new strategy to help steer the network to learn distinguishing features. The features from the point prediction branch of the network are sampled randomly and edges are constructed between the features representing the individual points and a series of shared MLP layers are used to predict whether the points carrying features belong to the same class or not. To explain in detail, the following steps make up the strategy being proposed:

- *Random point-pair sampling*: Given an input point cloud, random point pairs are sampled. Since there is a possibility that the selected point pairs do not represent all possible combination between

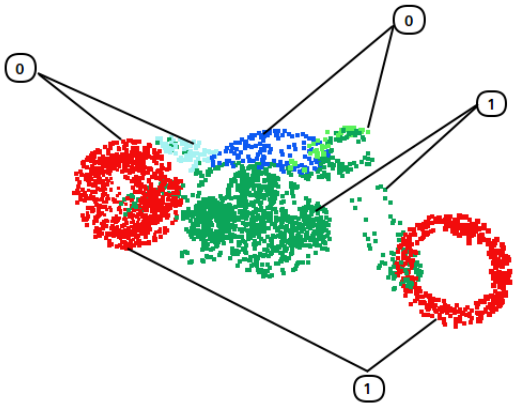


Figure 4.2: Edge labeling for edges constructed between randomly selected point-pairs where 1 represents similar parts and 0 indicates dissimilar parts.

points representing different parts and also might result in an imbalance in samples between number of point-pairs representing different combinations of parts, it is recommended that the number of sampled point pairs for each part combination is maintained the same. In this, even smaller parts with fewer number of points get equal representation in the training process.

- *Picking Features*: The features that are used for the prediction task are obtained from the last EdgeConv layer before the global max-pooling inspired from the strategy proposed in [19]. This forms a parallel branch of layers alongside the pointwise segmentation head of the network and also forces the EdgeConv layers to produce discriminative features.

- *Constructing Edges*: From the given set of point-pair indices and features from the above step, the edges are then constructed by concatenating the feature channels.

- *Edge Labels*: The edges built of points belonging to the same part categories are labeled as 1 and other edges with different combinations are labeled as 0.

- *Point-pair similarity loss (L_{PPS})*: The edge classification network is trained with a *binary cross entropy* (BCE) loss. The total loss of the network is a weighted sum of the PPS loss and the segmentation loss.

More formally, we can look into the loss modification needed to train this network.
Loss Function: The total loss of the network used during training would be:

$$L = \lambda_1 L_{Point-seg} + \lambda_2 L_{PPS} \tag{4.4}$$

where $L_{Point-seg}$ is the loss for the segmentation head and is implemented as a softmax cross entropy loss. λ_1 and λ_2 are the corresponding weights assigned during training. Given the final set of edges E, the label for each edge $e_{i,j} = (p_i, p_j) \in E$ is given as:

$$g^e_{i,j} = \begin{cases} 1 & \text{if } g^p_i = g^p_j \\ 0 & \text{if } g^p_i \neq g^p_j \end{cases} \tag{4.5}$$

where g^p_i and g^p_j are the ground truth part labels for point i and j constituting the edge. The PPS loss is then defined as:

$$L_{PPS} = -\frac{1}{|E|} \sum_{e_{i,j} \in E} (g^e_{i,j} log(pred^e_{i,j}) + (1 - g^e_{i,j}) log(1 - pred^e_{i,j})) \tag{4.6}$$

where $|E|$ is the number point-pairs or edges considered, $pred^e_{i,j}$ is the predicted label for the edge constituting points i and j. This loss is expected to provide auxiliary supervision to the overall point segmentation task by steering the network modules such as the EdgeConv layers to learn distinguishing features. Thus, the above discussion lead us to the second hypothesis we consider in this work.

Hypothesis H_2: Learning to classify edges between points of same parts and different parts leads to learning discriminative features and hence improves the segmentation performance of the network.

4.1.3 Attention Based Neighborhood Feature Aggregation

The EdgeConv module in DGCNN is based on learning parameters for edge feature transformation and aggregating neighborhood edge features through a predefined aggregation operation such as max-pooling across feature channels. This max-pooling aggregation operation retains only the maximum features across neighborhood edges for each of the feature channels and discards the rest of the features. This may result in information loss. In this section we look into how this max-pooling aggregation operation can give way to better techniques.

Attention mechanism has become a dominant and successful technique in current deep learning architectures which deal with sequential data such as for natural language processing (NLP). Typically, an attention mechanism does not need a fixed size input and as the name suggests, is capable of attending to the most important or vital part of the input in order to make decisions. This makes it a very powerful mechanism and has gradually evolved from being used as an improvement aid to conventional methods like *recursive neural networks* (RNNs) to being the basis for entire models as proposed in [52]. As deep learning for arbitrarily structure data developed more interest, graph neural networks have become one of the go to solution for handling such non-euclidean data and which is the basis for DGCNN as well. In [53], the authors propose a graph neural network with attention mechanism for node classification tasks. The

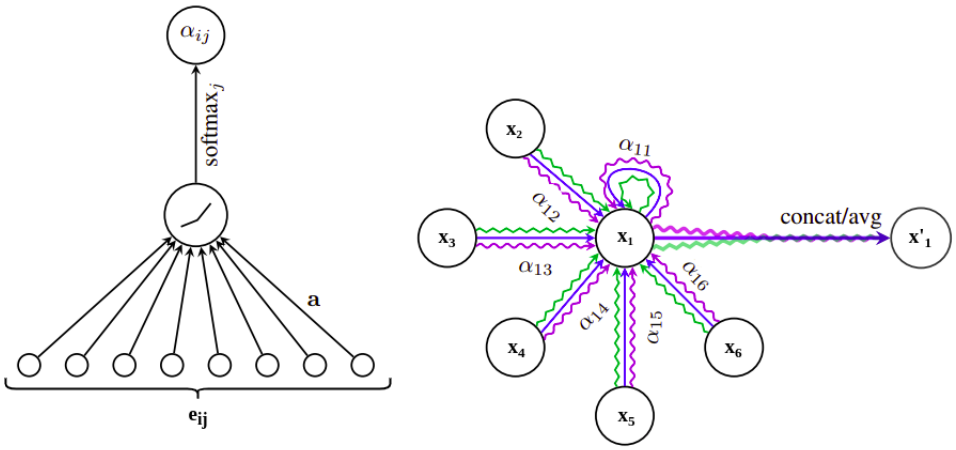


Figure 4.3: **Left:** The attention mechanism parametrized by the weight matrix a, followed by the activation function and softmax across all neighbors j. **Right:** Multi-head attention with 3 heads represented by different colors. Image adapted from [53].

idea here is similar to the original idea of *self-attention*. The graph attention (GAT) mechanism proposed in the paper computes point feature representations by attending over its neighbors.

In this work, we adopt the attention strategy used in [53] into the EdgeConv module of the network. Note that the proposed solution here can be thought of as an instance of the GAT mechanism. The difference here is that we attend over the neighborhood edge features constructed as defined in equation 3.1 rather than the absolute features of the neighbors. The attention mechanism produces a weighted sum over the neighborhood features. This enables the network to assign different importances to certain neighbors over others and in this way also may increase the interpretability of the learning module. More formally, we can introduce the proposed changes as follows:

$$w_{ij} = a^T(e_{ij}) \tag{4.7}$$

where w_{ij} is the corresponding weight for each edge produced by the attention head without using any activation function, e_{ij} is the learned edge feature representing an edge between point x_j in the neighborhood of point x_i.

$$\alpha_{ij} = softmax(w_{ij}) = \frac{exp(w_{ij})}{\sum_{k \in N_i} exp(w_{ik})} \tag{4.8}$$

where α_{ij} is the final output of attention weights of the attention head with a softmax applied across the weights w_{ij} of all neighbors x_j. To put it in more detailed manner:

$$\alpha_{ij} = \frac{exp(LeakyRELU(a^T e_{ij}))}{\sum_{k \in N_i} exp(LeakyRELU(a^T e_{ik}))} \tag{4.9}$$

where LeakyRELU is the activation applied on the transformed edge features using the transposed attention head weights a^T which produces w_{ij}. Finally the attention weights are used for the weighted sum as below:

$$x'_i = \sigma\Big(\sum_{j \in N_i} \alpha_{ij} e_{ij}\Big) \tag{4.10}$$

where σ is the chosen activation function applied on the weighted sum features that is assigned to point x_i. In some cases, in order to stabilize the attention learning mechanism, *multiple heads* make a difference. Thus *multi-head attention* can be formally denoted as:

$$x'_i = \Big\Vert_{k=1}^{K} \sigma\Big(\sum_{j \in N_i} \alpha^k_{ij} e^k_{ij}\Big) \tag{4.11}$$

where $\|$ denotes the concatenation of features from the the k attention heads, which is adopted by this work in the case of multiple attention heads. In addition to the benefits of the attention mechanism discussed above, it does not add much to the computational requirement and has a time complexity just as much as the regular GCNs. Thus, this brings us to the third hypothesis put forwad by this work, Hypothesis H_3.

Hypothesis H_3: Deploying a learning based neighborhood feature aggregation strategy, such as an attention mechanism, with EdgeConv is better than a max-pooling aggregation operation and hence will result in overall performance improvement.

4.1.4 Regional Part Descriptors

One of the popular failure modes of the network as discussed in section 3.4 is the handling of temporary parts and the noisy and spurious segmentation predictions of parts without any context to the region in which the point is present in. This is especially worsened when dealing with boundary points which poses a tough challenge for segmentation and in some case even difficult for a human to predict. Spurious predictions of temporary parts even when all the neighborhood points are predicted to belong to a different part makes it difficult to derive any understanding of the networks decision. In addition to that, the number of spurious prediction of a particular part and the average number of points typically representing the part are generally significantly different. As common with other deep learning networks, such predictions are also made with extreme confidence.

In [60], while working on scene segmentation tasks, the authors propose a strategy in order to refine the final predictions based on the global scene descriptor learnt by the network. This global scene descriptor is produced by parallely transforming the global feature form the segmentation network. The global scene descriptor is basically trained to encode the different objects that make up the given scene. In this way, it can be used to refine the final predictions made by the segmentation network and avoid misclassification of

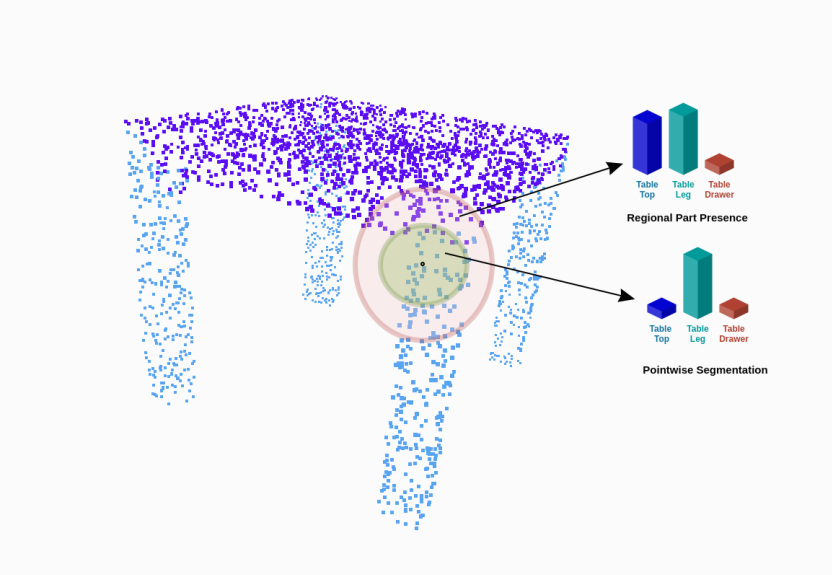


Figure 4.4: Illustration of the pointwise segmentation predictions and regional part presence predictions. The two ellipses indicate the rough neighborhood size of the kNN neighborhood used for the EdgeConv blocks in the input $\mathbb{R}^3$ space. Notice that the inner region has points belonging to table top too.

points to categories that are not present in the scene. The global scene descriptor used for the refinement procedure is learnt during training by a loss applied on the final refined predictions.

In this work, in order to deal with the problem of noisy predictions and to enhance the local feature learning capabilities, a new strategy is proposed where, in addition to predicting segmentation scores for each point, a parallel branch of the network is trained to predict the part composition surrounding the corresponding point's neighborhood and whose features are used to provide better local context to the segmentation branch. In its simplest setting, this part composition is barely limited to predicting the part presence in the neighborhood of a point. Considering that the network only operates on a small neighborhood size in order to make segmentation predictions, the part composition prediction is made for a larger neighborhood on a fixed graph unlike the dynamic graph reconstruction adopted for the segmentation branch. The final regional part presence (RPP) predictions can be used to further refine the pointwise segmentation predictions. In addition to this, the features from RPP branch can be fed to the segmentation head for *message passing* and providing better regional contextual information for predictions.

The proposed solution involves the following key considerations:

- *Parallel branches:* Parallel branches of the network are trained, where one branch predicts the segmentation scores and the other branch predicts the regional part composition/presence and features are shared between the RPP branch to the segmentation branch.
- *kNN neighborhood:* Considering a kNN neighborhood for each point in the EdgeConv operation, where k_1 and k_2 represent the k value for the two scales, k_1 which is used for the segmentation

branch of the network is less than k_2 (i.e. $k_1 < k_2$).

- *Graph update:* The segmentation network adopts the dynamic graph update while the RPP branch has a fixed graph of the spatial neighborhood.

- *RPP Labeling:* RPP prediction is a *multi-label* problem where the ground truth labels are in the form of a binary vector (multi-hot label) where parts present in the neighborhood of the point are represented by 1. The minimum number of points required in order for the part to be considered as present is a hyperparameter represented as n_min. More formally, the ground truth RPP label R for a point x:

$$R = (r^0, r^1, .., r^{s-1}) \in \{0, 1\}^s \tag{4.12}$$

where r_i is the ground truth label representing the presence of a part in the neighborhood and s is the total number of parts.

$$r_i = \begin{cases} 1 & \text{if } n_i \geq n_{min} \\ 0 & \text{if } n_i < n_{min}, \end{cases} \tag{4.13}$$

where n_i is the total number of points representing a part j within the kNN neighborhood.

- *$Loss_{RPP}$:* Since RPP prediction is a multi-label problem with a binary vector as a ground truth, the loss for each prediction can be treated as a sum of the binary cross entropy (BCE) or sigmoid cross entropy (SCE) losses corresponding to each vector component. Since there is typically a huge imbalance in the number of points representing a part, *focal loss* (FL) can be used to obtain the final RPP loss. More formally:

$$Loss_{RPP} = -\sum_{j=1}^{s} \alpha_j^t (1 - p_j^t)^\gamma log(p_j^t) \tag{4.14}$$

where j denotes each part category, α_j^t is the weight set as a hyperparameter or learnt for tackling class imbalance, γ is the focusing parameter as explained in [30] and p_j is the part presence prediction output for class j. α_j^t is defines similar to p_j^t which is defined as below:

$$p_j^t = \begin{cases} p_j & \text{if } r_j = 1 \\ 1 - p_j & \text{otherwise} \end{cases} \tag{4.15}$$

and finally the total loss is given by:

$$Loss_{Total} = \lambda_1 * Loss_{Point-seg} + \lambda_2 * Loss_{RPP} \tag{4.16}$$

where λ_1 and λ_2 are the weights for the corresponding losses.

- *Segmentation prediction refinement:* In addition to sharing features the RPP prediction branch's output is also used to refine the final predictions of the segmentation branch during test time. The

refinement happens by first obtaining a binary mask from the prediction of the RPP branch. The binary mask defines the presence of the parts in a neighborhood or the entire object when obtained from the *global part presence* prediction which is obtained by max-pooling over the individual RPP predictions. In order to decide whether to accept the prediction confidence indicates a part being present or not, one has to decide the threshold confidence values. For this purpose, this work follows the SCut score-based threshold selection procedure. For each part j, the threshold value T_j is obtained as in [62]:

$$T_j(c_j) = \arg\min_{t_j} \; M(t_j; c_j) \tag{4.17}$$

where c_j represents part j and t_j its corresponding threshold. M is the function of choice to reduce. Here, we consider the negative of overall mIoU of the refined segmentation network as the function M. For each part j, a corresponding t_j is chosen which results in increased mIoU of the segmentation network after refinement.

$$P_{RPP} = (p_0, p_1, ..., p_{s-1}) \in [0, 1]^s \tag{4.18}$$

where P_{RPP} is the output of the RPP branch with confidence scores for each part and the binary mask is obtained by:

$$P_{RPP-Binary\ mask} = (p'_0, p'_1, ..., p'_{s-1}) \in \{0, 1\}^s \tag{4.19}$$

where p'_i is obtained using the threshold operation using the values T_j for each class separately,

$$p'_i = \begin{cases} 1 & \text{if } p_i \geq T_j \\ 0 & \text{if } p_i < T_j, \end{cases} \tag{4.20}$$

$$P_{Final} = P_{Point-seg} * P_{RPP} * P_{RPP-Binary\ mask} \tag{4.21}$$

where $P_{Point-seg}$ is the output of the segmentation network. Thus the final prediction for a single point in the network is presented after utilizing the confidence scores of the RPP branch as well as the binary mask obtained from it. Note that not only the binary mask is expected to mask out noisy predcitions occuring out of regional context but the confidence scores from RPP prediction is utilized to refine the final output of the segmentation branch as well. Thus, this bring us to the next hypothesis considered in this work, Hypothesis H_4.

Hypothesis H_4: Learning to predict the presence of other parts in a point's neighborhood as an auxiliary task can help in providing more regional contextual information for pointwise segmentation predictions and hence improve its performance.

4.1.5 Loss Modifications

In addition to proposing solutions to enhance the networks performance by modifying the fundamental blocks of the network architecture, this work also considers potential loss functions to enable the network to learn in an improvised manner. The dataset plays as much as important role as the network architecture itself. It is important for the dataset to represent the different classes in a balanced manner. In some case, it is unavoidable that some categories might receive less representation than the other(i.e. number of points representing an airplane tire is always going to less than the number of points representing the airplane body in spite of both parts being present in all the samples). This can in a way be considered as part of the broader problem to solve for the network but imbalance should be avoided whenever possible. In unavoidable cases of dealing with imbalanced data, loss modifications have helped in various previous works [57].

Class Balancing

The dataset as discussed in section 3.2 is highly imbalanced with respect to the number of samples representing each object category as well as with respect to the number of points representing each part category. In this work, we consider to tackle the imbalance problem by adopting the strategy of *median frequency balancing*. The frequency of each part j is determined by the number of points of class j divided by the total number points in samples which carries part j. The median of all the part frequencies is the median frequency. The total loss is computed as the weighted sum of all the individual point's softmax loss.

$$\alpha_j = median\_frequency/freq_j \tag{4.22}$$

where $freq_j$ is given by,

$$freq_j = n_j/N_j \tag{4.23}$$

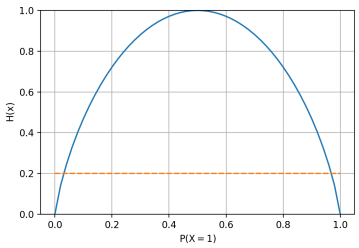
where n_j is the total number of points in the training dataset representing part j and N_j is the total number of points in all samples containing part c. Then the total loss for each sample point cloud with M points is computed by:

$$Loss_{Total} = \frac{1}{M}\sum_{i=0}^{M-1} \alpha * Loss_{softmax}(x_i) \tag{4.24}$$

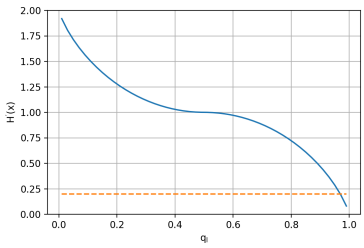
where α is set to α_j if the ground truth label for the point is j. Thus this brings us to the fifth hypothesis put forward by this work, Hypothesis H_5.

Hypothesis H_5: The drastic imbalance in the portion of dataset samples for each part degrades the performance of the network and can be compensated by a weighted loss function.

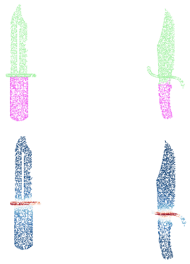
Boundary Loss



(a) Binary Entropy



(b) Loss weightage based on entropy and percentage of points belonging to the same class as the centroid.



(c) GT vs Weightage: Blue-white-red transition corresponds to min-medium-max entropy.

Figure 4.5: Illustration of boundary loss based on entropy of part combinations in the point's neighborhood with two parts and can be similarly extended to multiple parts. The orange line indicates the minimum weightage.

As discussed in the section 3.4, one of the most popular failure modes is the noisy boundary predictions or boundary distortions. Across different methods of semantic segmentation, handling boundary points is one of the most challenging situations. A boundary point lies in the neighborhood of more than one part. The simplest case being at the intersection of two parts. In this work, we consider proposing a new strategy for a loss term that takes into account the awareness needed to segment points present at the intersection of two or multiple parts.

Here, the proposed loss focuses on points which possess a neighborhood in the input $\mathbb{R}^3$ space, containing points from other parts. In essence, these points can be considered as boundary points dotting the region of separation between different parts. In addition to identifying the points to focus on while training, a second requirement is to identify the criteria on which to assign the magnitude of weights for the proposed loss term. For this, one approach would be to quantify the amount of *confusion or uncertainty* that the network might have to deal with while segmenting the point. This uncertainty can be given by considering the amount of part mixture in the point's neighborhood.

The difficulty to deal with boundary points is considered to be at the minimum when the point comprises of a neighborhood of points only belonging to the same category as itself, while it is at a maximum when the point is the only point in it's neighborhood belonging to the same category as itself. More formally, the entropy for the point x can be defined as below:

$$H(x) = max\left(h_{min}, \frac{-\sum_{i=1}^{n} q_l * log(q_l)}{log(n)}\right) \tag{4.25}$$

where $H(x)$ is the normalized entropy value with a minimum value h_{min}, n is the number of parts in the

neighborhood, q_l the portion of points belonging to the same part as x.

$$q_l = \frac{n_l}{k} \tag{4.26}$$

where n_l is the number of points as the same part as x and k is the total number of neighborhood points. Since entropy returns a lower value even in the case when q_l is not the majority portion in its own neighborhood, we can do the following adjustment as below:

$$H'(x) = \begin{cases} H(x) & \text{if } q_l > t \\ 2 - H(x) & \text{if } q_l \leq t, \end{cases} \tag{4.27}$$

where $H'(x)$ is the final entropy value assigned to the point x, and t is the threshold beyond which the entropy value is to above 1, which can be set to $1/n$. Overall, the entropy is below 1 when x is surrounded by majority of points as the same class as itself and is more than 1 if otherwise. The final loss function can be given by multiplying the entropy based weights of each point with the individual points segmentation cross entropy loss with softmax, similar to equation 4.24. Thus, this brings us to the sixth hypothesis put forward by this work, hypothesis H_6.

Hypothesis H_6: Boundary regions are one of the most difficult to segment and a loss function which focuses on the boundary points will help in improving overall performance.

4.2 Partial Point Cloud Segmentation

In this section, we will look into the proposed approaches to help improve the segmentation performance of the network when dealing with partial point clouds. The approaches are discussed in detailed along with the hypothesis on which it stands on, which will later be tested in the following chapter.

4.2.1 Training on Partial Data

The state-of-the-art networks proposed so far have been only designed to face the challenge of segmenting complete point clouds . When faced with the task of segmenting partial point clouds, they are bound to produce a degraded performance or might completely break down. Most of the approaches are not *translation-invariant* (TI), rotation-invariant (RI) and scale-invariant (SI) except for a few networks which are invariant to one or couple of them. Very few works have actually worked on improving the performance on partial point clouds. Even such networks assume prior knowledge of the complete point cloud which is not usually available in real world scenarios. One of the reasons for no progress could be stated as the inflexibility of the building blocks of the networks which have served as the baseline for various works dealing with point cloud processing. Network architecture modifications pose a major challenge in dealing with partial point clouds.

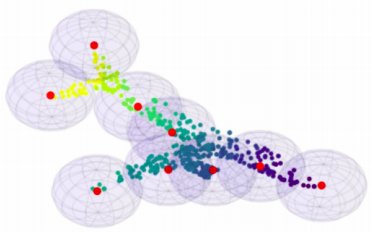


Figure 4.6: Emulating partial inputs by partitioning input point clouds which are used as voting candidates to obtain the global feature, as seen in the image adopted from [68].

Partial point clouds pose multiple challenges to the current state-of-the-art models. The major problems could be summarized as follows:

- **False Centroid**: The true centroid of the partial point cloud is typically not known and hence the input to the network is the partial point cloud centered with respect to the incomplete set of points. This results in the undesirable effect of the coordinates of the input points being corrupt and the network perceives the input shape as a completely new shape rather than only a partial shape.
- **Scaling**: Typically the input points to the network are normalized to a unit sphere. When partial point clouds are normalized, it tends to have an undesired scaling effect. This poses a challenge to the network and especially for the convolution operation which now perceives a neighborhood scaled to a different level than what was expected for a given object category.
- **Incompleteness**: Finally, the obvious challenge of having to deal with incomplete information regarding the object. Compared to the centroid information which corrupts the absolute coordinates and the undesired scaling effect of the normalizing process, this is a challenge that one has to deal even when the other two problems have been tackled possibly by architecture modifications.

Among the three problems discussed above, the first two can be possibly dealt with using TI networks combined along with SI convolutions. The third problem poses the major challenge compared to the other two. In one approach, the authors deal with partial data by training with complete point clouds and the network emulates a partial input by a random voting strategy from different regions of the point cloud in order to determine the global feature. The partitioned regions are as seen in Figure 4.6. The network performance improves, but is still significantly below the performance levels as compared to being trained and tested on complete data. In this work, owing to the reasons discussed above, and considering the successes of data augmentation strategies used for other deep learning based tasks, training with partial data is considered as a simple and effective strategy for combating partial point cloud segmentation.

Data Augmentation: The partial data generated for training are produced in a similar manner in which the occlusion test point clouds are generated in section 3.3.2 from the previous chapter. The key difference here is:

- Random sampling of normals for describing the slicing plane orientation.
- Random sampling of the occlusion quantile levels in order to describe the percentage of dropped points. The range of occlusions are from 0 to 90%, where 0 corresponds to the complete point cloud.
- Each complete point cloud is used to generate 8 occluded samples using the randomly generated slicing plane orientations and occlusion quantiles.

Thus, as discussed above, training on only complete data makes it infeasible for the network to deal with partial inputs given the limitations of the network architecture designs. Instead of emulating the partial point clouds form complete point clouds, this work considers training on partial data as an effective strategy. This brings us to the next hypothesis put forward in this work, hypothesis H_7.

Hypothesis H_7: Training on partial point clouds will improve the overall performance of the network for partial point cloud segmentation.

4.2.2 Multi-Task Learning

The networks that have been dealt in this work so far, take a complete or partial point cloud as input and aim to produce segmentation scores for each input point. When partial point clouds are presented to the network, the network has apparently struggled to deal with it in the same way it deals with complete point clouds and which is obvious from the performance measures. The aim of the network is to learn one task, which is segmentation. This begs the question whether there is any other task which can help the network in learning features that can ultimately lead to improvements on task of partial point cloud segmentation.

It can be learnt from several previous works that auxiliary tasks can help steer the network to learn features to better suit our end goal. Multi-task learning aims at leveraging commonalities between different tasks in order to help improve the performance of the network for the desired tasks. The features that are learnt are shared between the tasks and hence need to be capture enough information to be able achieve multiple goals. Multi-task learning has been already proven to be effective in image segmentation tasks [5], where a cascaded architecture helps improve on one task from the features learnt during the completion of another task. This approach adds more awareness to the subsequent tasks in the cascaded architecture. In [11], a graph convolution based multi-task feature learning architecture is propose. The key challenge in such approaches is to identify the closely related tasks which can benefit from each other.

In this work, we consider the usage of shape completion as an additional task which can help with segmentation. The next challenge is to identify how to effectively modify the architecture to reap the benefits of shape completion. As already discussed above, the incompleteness characteristic of the input point cloud leads to major distortions in the predicted global feature. This serves as a good starting point to propose solutions based on a multi-task loss involving shape completion. One approach is to

Figure 4.7: **Left:** A shared encoder for both the completion and segmentation. **Right:** A dedicated segmentation encoder with a shared completion encoder setup.

consider using the encoder's output, the global feature, to be used for both completion and segmentation. This can be expected to force the network to learn similar features for different occlusion samples of the same object owing to the completion networks requirement to output the same shape irrespective of the differing occlusion samples. In order to achieve our goal, the key modifications considered are:

- **Shape Completion Decoder**: The decoder for the proposed architecture is based on the FoldingNet [61] decoder. The global feature is concatenated along with fixed set of grid point indices, which are later transformed by shared MLPs to produce the final coordinates of the completed point cloud.

- **Reconstruction Loss**: The loss considered for shape completion is the symmetric version of the chamfer distance loss (CD Loss),

$$Loss_{CD} = \frac{1}{|P|} \sum_{x \in G} \min_{y \in P} \|x - y\|_2 + \frac{1}{|G|} \sum_{y \in P} \min_{x \in G} \|y - x\|_2 \tag{4.28}$$

  where P and G are the predicted and ground truth point clouds respectively. there is no constraint on the number of points in the completed point cloud to be the same as the one in the ground truth point cloud. The first term forces the predicted points to be as close to the ground truth points as possible and the second term forces the predicted points to be spread across to provide enough coverage of the ground truth object. Finally, the total loss is given by a weighed combination of the segmentation loss and the chamfer distance loss as below:

$$Loss_{Total} = \lambda_1 * Loss_{Point-seg} + \lambda_2 * Loss_{CD} \tag{4.29}$$

With the above two modifications, the network should be able to segment as well complete partial inputs with a common encoder. Thus, the next hypothesis proposed by this work is hypothesis H_8.

Hypothesis H_8: A multi-task learning framework with a shared encoder to leverage the commonalities between shape completion and semantic segmentation will help improve the overall performance on partial point cloud segmentation.

Another way of exploiting shape completion for segmentation tasks would be to implement a dedicated segmentation encoder and a segmentation decoder which utilizes the features from the encoder trained for the shape completion task. In this way, not only can we better deal with the distortions to the global feature due to incompleteness of input data, but also have a dedicated encoder for segmentation which may not have the added responsibility of learning robust global features irrespective of the partial inputs of a given object sample. The shape completion encoder however trains to produce latent representations for multiple tasks. In order to achieve this, another modification is proposed.

- **Shape Completion Encoder**: A shape completion encoder is added to the network based on the two-stage pointnet encoder in point completion network (PCN) [66]. The encoder features are then concatenated along with the feature from the dedicated segmentation encoder and further used for the segmentation task.

The two different proposed approaches can be seen as in the Figure 4.7. Thus, the dedicated segmentation encoder is hypothesized to be more effective in the multi-task learning framework. This brings us to the next hyopthesis put forward by this work, hypothesis H_9

Hypothesis H_9: A multi-task learning framework with a dedicated segmentation encoder and a shared shape completion encoder to leverage the commonalities between shape completion and semantic segmentation will help improve the overall performance on partial point cloud segmentation.

4.3 Semantic Shape Completion

In this section, we will look into the newly proposed approach of semantic shape completion. Inspired by semantic scene completion, semantic shape completion on point clouds is the task of jointly predicting complete point clouds of objects along with the corresponding part labels for each point. This section discusses in detail the key differences between the previously discussed approaches of using shape completion to enhance segmentation to intertwining the tasks together as semantic shape completion. The techniques and modifications to achieve the desired goal are also discussed below.

Even though the previous ideas proposed consider shape completion and semantic segmentation to be closely related tasks, the architectures do not fully intertwine both the tasks. The architectures proposed earlier have dedicated segmentation or completion encoders and decoders which splits the task between them. The segmentation labels produced are for the input partial point cloud and not for the completed point cloud. This separation of tasks and the resulting network architectures makes us reconsider a few points regarding our approach. A few undesirable conditions due to the earlier proposed approach are discussed below:

- The parallel branches for separate decoder and encoder architectures increase the computational and memory cost of the network.

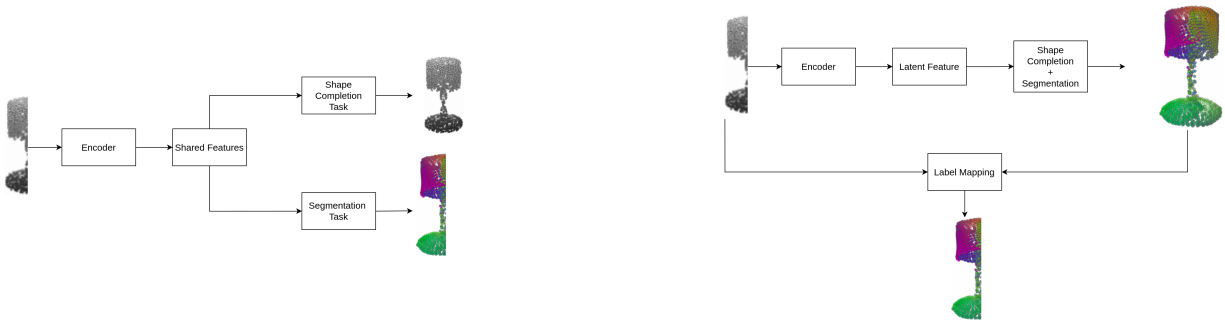


Figure 4.8: An Illustration of the proposed method of considering a joint approach for segmentation and completion. **Left:** Shape completion and segmentation considered as separate tasks . **Right:** Joint predictions of complete point clouds and corresponding semantic labels.

- Since we don not consider prior knowledge regarding bounding box parameters, the completed point clouds are typically rendered useless. The completed point clouds are not in the same coordinate frame as the input partial point cloud. The completion network typically is trained to map the input partial point cloud to the complete point cloud fit inside a unit sphere which makes it not straightforward to interpret the scale and position of the partial input point cloud.
- The completed point clouds do not carry semantic labels and are also typically too noisy to be used by a segmentation network to extract semantic labels.

Thus, in a real world scenario, knowing the complete point clouds with the same scale and in the same coordinate frame of the input object is very much desirable. In addition, knowing the semantic labels of individual points makes it an even more attractive option to be considered for various tasks which have to tackle partial point clouds in a frequent manner.

The network proposed in [45] for semantic scene completion argue that the tasks of completion and segmentation are deeply intertwined and can be considered as single task. The authors of the shape completion on point clouds in [48] also indicate that their hierarchical decoder learns to predict points which are semantically related as seen in Figure 4.9.

For the above discussed reasons, it is compelling to train networks for semantic shape completion while dealing with partial point cloud segmentation.

4.3.1 PCN Based Semantic Shape Completion

The network chosen for the task of semantic shape completion is the point completion network (PCN). The simplicity of the network architecture and its impressive results matching other state-of-the-art methods for shape completion makes it an ideal candidate to be chosen as a baseline network for semantic shape completion.

The illustration shown in Figure 4.10 shows the approach of the network used for the shape completion task. The network can be broken down for further understanding into the following:

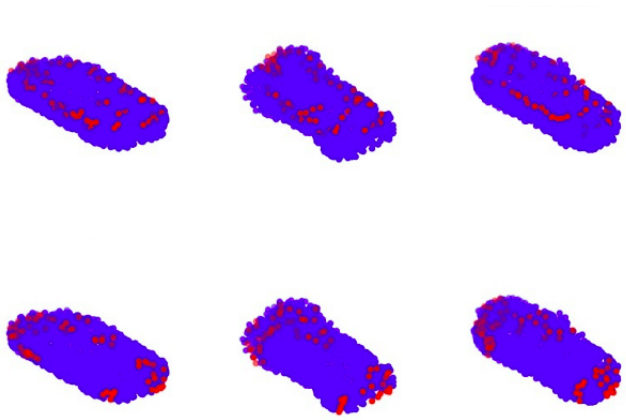


Figure 4.9: TopNet Results: Example results showing different parent nodes(nodes 2 and 3) leading to children node predictions(red points) which are semantically related. The first row shows points predicted to cover the top of a car and the second row shows predictions which cover the front and rear of the car. Image adopted from [48].

- **Encoder**: The network takes in the partial input and encodes it using a two stage *pointnet* based encoder. Series of shared MLPs are used to extract features and a global feature is obtained in both stages by a pointwise maxpool operation. The global feature from the first stage is concatenated along with the pointwise features from the previous layers in order to extract features for the second stage. This two stage encoding process helps the network capture individual point features with global context.

- **Intermediate predictions**: The encoded feature is then used by a fully connected network to predict the coarse set of points for the complete point cloud. This is part of the first stage in the decoding process. A stage-wise loss is considered where the intermediate set of points are used to provide additional supervision. This stage plays a major role in predicting the global shape structure.

- **Folding based decoder**: The predicted intermediate points are concatenated along with 2D grid indices and the global feature from the encoders output. Each point receives a small scale grid compared to the one in FoldingNet. The grid concatenated features are then transformed, or folded, to produce fine predictions which better represent the local regions around each point.

- **Loss**: The network uses a stage wise loss in the decoder. The coarse set of points are used to calculate the chamfer distance (CD) loss as well as the earth mover's (EMD) distance loss. Both the losses are calculated based on the subsampled GT points in order to match the requirement of the earth mover's distance loss which requires the number of predicted and ground truth points to be equal. The final set of predicted points are used to compute the chamfer distance loss. The EMD

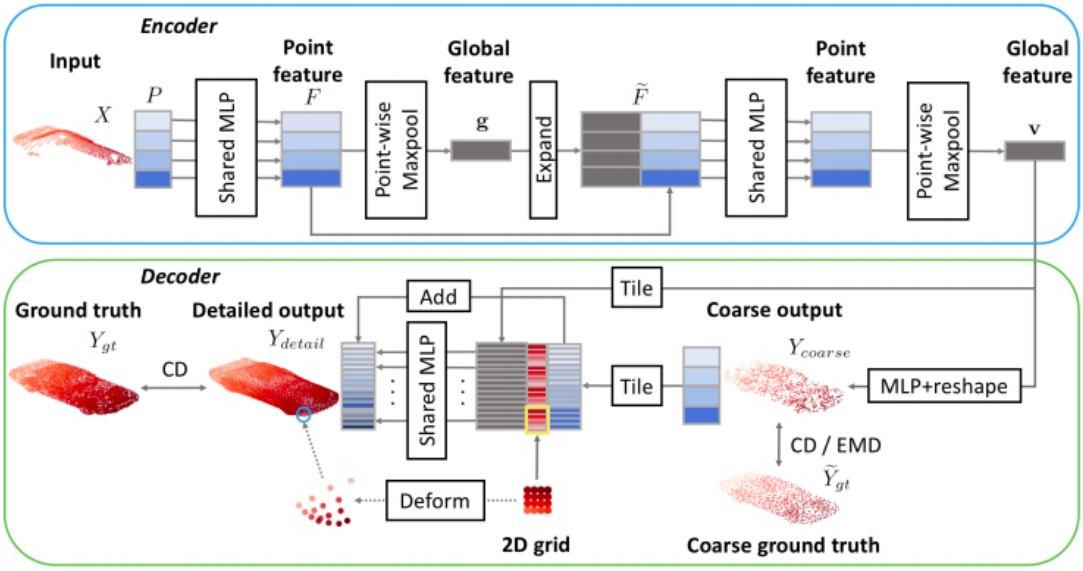


Figure 4.10: **PCN**: The network combines the benefits of using a FC decoder, used for the intermediate coarse prediction, and a folding based decoder in a two stage decoding process to produce the final fine predictions. Image taken from [66].

loss is not used in the final layer predictions owing to the computational complexity rendered by the large number of final set of points compared to the intermediate predictions.

The PCN approach makes use of the advantages of both the FC based decoding as well as the folding based decoding. The folding based decoder inherently impose the smoothness constraint by using the manifold based decoder and PCN avoids the adverse effects of it by not using it for the global shape prediction via the intermediate points, but uses the smoothness constraint in order to learn the shapes of local patches surrounding each point. The multi-stage predictions also provide more control over the networks learning by usage of the loss for the intermediate predictions.
In order to utilize the network for our desired goal, the following key considerations are attended to:

- *Architecture Modifications*: The shape completion network is only concerned with predicting the x,y and z coordinates of the final set of points. To match our goal, the final prediction layer is made to predict the semantic label along with the point coordinates, hence the final layer of MLP learns 3 + p features for each point instead of just 3, where p is the total number of points.

- *Training Data*: Unlike the previous proposed approaches, here the training data contains complete point clouds adjusted to the original scale and centered on the true centroid of the partial point cloud. This forces the networks to predict complete point clouds along the same coordinate frame as the original complete point cloud matching the same scale as it.

- *Loss*: The loss consists of the weighted combination of the segmentation and the reconstruction loss

at the intermediate and final layers. Chamfer distance is used for the reconstruction loss.

$$Loss_{Total} = \lambda_1 * Loss_{Point-seg} + \lambda_2 * Loss_{CD-Coarse} + \lambda_3 * Loss_{CD-Fine} \tag{4.30}$$

where $Loss_{CD-Coarse}$ is the loss on the intermediate coarse point cloud prediction from the global feature and $Loss_{CD-Fine}$ is the final output point cloud's chamfer distance loss.

- *Label Mapping*: Given the ultimate goal of partial point cloud segmentation, the obtained labels from semantic shape completion output has to be transferred back to the partial point cloud. Give that the predicted point cloud is in the same scale and coordinate frame as that of the original point cloud, the labels can be transfered by a simple *nearest neighbor* (NN) strategy. For every point in the input partial point cloud, the label of the nearest neighbor in the predicted completed point cloud is assigned as its own label.

With the above mentioned changes, the network can be expected to learn to map a partial point cloud to a completed point cloud with semantic labeling of individual points without the assumption of any prior knowledge other than the object category it belongs to. This lead us to the next hypothesis proposed by this work, hypothesis H_{10}.

Hypothesis H_{10}: Shape completion and semantic segmentation are deeply intertwined tasks and hence a joint estimation of complete point clouds and semantic labels can be learnt using a network architecture primarily designed for shape completion.

4.3.2 Integrating Graph Convolutions

As discussed before, the PCN architecture deploys a fully connected layer to predict a corase point cloud from the global feature and then deploys a 2-manifold based folding decoder strategy to learn fine point clouds. This strategy of using 2D-grids inherently enforces a smoothness constraint [48]. In order to overcome this constraint, a hierarchical approach was proposed in TopNet. This architecture uses a shared MLP to predict children node features from the given parent node. The final layer then predicts the complete set of points, and there is no intermediate predictions as in PCN. Since our goal is to primarily improve on the segmentation performance of the network, a new strategy is proposed here which takes advantage of a powerful module like the EdgeConv, a graph convolution module used in our previous works for segmentation.

A good starting point to integrate the EdgeConv block is to take advantage of the intermediate coarse predictions in the PCN netowrk. The coarse predictions are fed to the edge convolution block in order to extract the neighborhood information of individual points and therefore enrich the semantic information ultimately helping with its semantic labeling task. In order to apply this in a multi-stage manner, a hierarchical decoder is adopted instead of a folding based decoder in PCN. Thus, the edge convolution

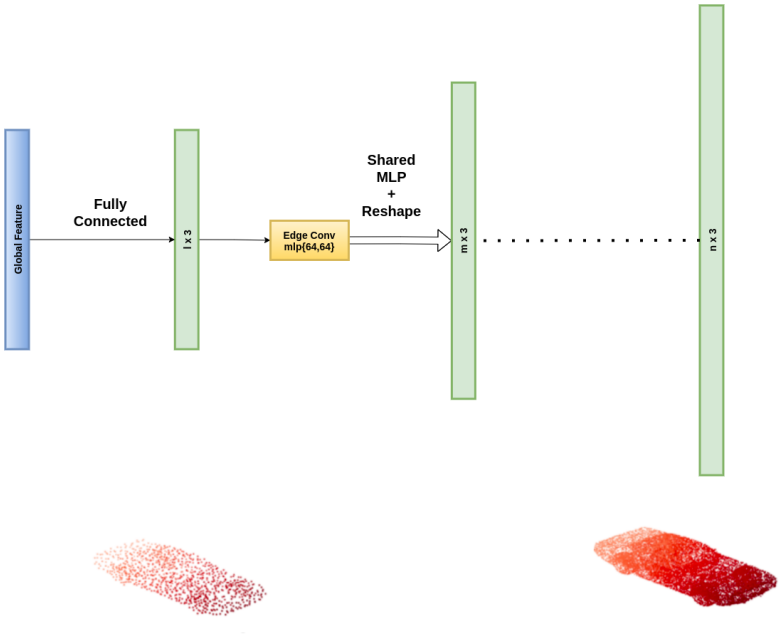


Figure 4.11: An illustration of the proposed approach for integrating graph convolutions. A graph convolution based hierarchical decoder is used starting form the coarse predictions by the FC layer in PCN.

module is applied at every stage learning the neighborhood information of individual points and a shared MLP is used to predict the children points from the parent node feature. These children nodes are further passed through another edge convolution block and a shared MLP to produce its own children nodes until the desired number of nodes or points is reached as seen in the Figure 4.11.

In order to enable the edge convolution to be more effective with the intermediate predictions, earth mover's distance loss (EMD) is used here instead of the chamfer loss used for the previous approach. The EMD loss is more computationally expensive and it requires the number of GT points and the predicted points to be the same. The loss is implemented along with an approximation scheme for the mapping.

$$Loss_{EMD-Coarse} = \min_{\phi:P\to G} \frac{1}{|P|} \sum_{x\in P} \|x - \phi(x)\|_2 \tag{4.31}$$

where a bijection mapping ϕ between the predicted points P and GT points G is used to calculate the overall loss, given by the distance between the mapped points. The final loss is given as:

$$Loss_{Total} = \lambda_1 * Loss_{Point-seg} + \lambda_2 * Loss_{EMD-Coarse} + \lambda_3 * Loss_{CD-Fine} \tag{4.32}$$

In addition to the above, a *spatial transform network* is also used to learn the translation and scaling for the final output point cloud. Thus, with all the details discussed above, it brings us to the final hypothesis proposed by this work, hypothesis H_{11}.

Hypothesis H_{11}: A graph convolution based hierarchical decoder can better capture semantic information than a folding based decoder for improving the segmentation performance of the PCN based semantic shape completion network.

5

Experiments and Results

This chapter focuses on testing the proposed ideas and hypotheses in the previous chapter. The chapter is divided into three sections for complete point cloud segmentation, partial point cloud segmentation and semantic shape completion. The network architectures and experimental setup used are included within each experiments. Key observations are also listed in the discussion subsections in experiments.

5.1 Complete Point Cloud Segmentation

This section provides the detailed information on the experiments and results for complete point cloud segmentation.

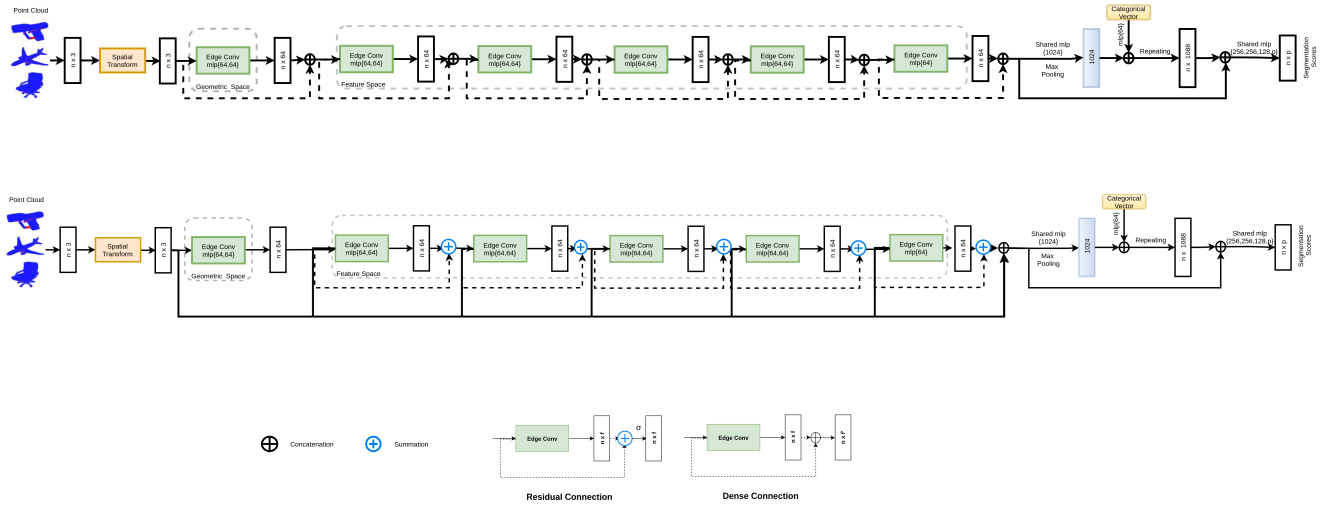
Experiment 1

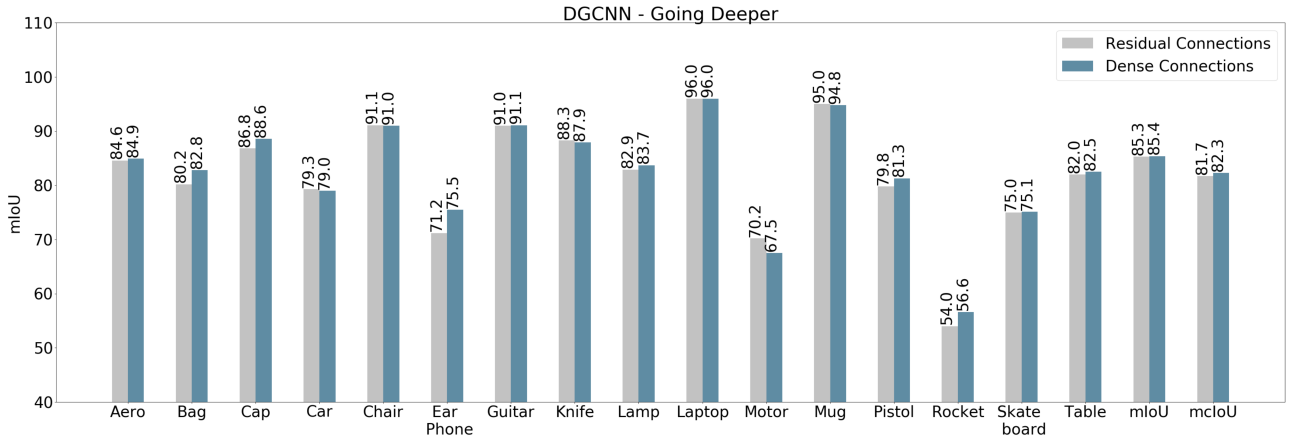
- **Objective**: The objective of this experiment is as follows:
  - Test the effect of stacking more EdgeConv layers to the network and henceforth verify the impact of deeper GCNs for part segmentation.
  - Difference in impact between using dense and residual connections for the deep network.
- **Experimental setup**: The data split and other network hyperparameters not mentioned in the table are the same as in section 3.14 used to train the baseline network, **Experimental Setup - 1**.

Experimental Setup - 2	
Batch Size	16
Learning rate	0.0003
Optimizer	Adam Optimizer
LR Decay rate	0.5
Decay Steps	200000
LR Clip	1e-5
BN Decay	0.5
BN Decay Rate	0.5
BN Decay Steps	400000
CPU	Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz
GPU	2x NVIDIA Titan RTX (Turing)
RAM	24GB
DL Framework	TensorFlow 1.13(GPU)

Table 5.1: Deep DGCNN - Training Hyperparameters and Setup

- **Results**: The results obtained are shown in the Figure 5.1.





	Aero	Bag	Cap	Car	Chair	Ear Phone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate board	Table	mIoU	mcIoU
# Samples	2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271	16881	16881
Residual Connections	84.6	80.2	86.8	79.3	91.1	71.2	91.0	88.3	82.9	96.0	70.2	95.0	79.8	54.0	75.0	82.0	85.3	81.7
Dense Connections	84.9	82.8	88.6	79.0	91.0	75.5	91.1	87.9	83.7	96.0	67.5	94.8	81.3	56.6	75.1	82.5	85.4	82.3

Figure 5.1: Dense-DGCNN vs Residual-DGCNN: The networks are shown above the bar plot, where the topmost network is the one with dense connection.

- **Discussion**: The deeper network has four more edge convolution layers than the vanilla version of DGCNN and within each layer there are two layers of shared MLPs and hence an overall increase in eight layers of additional MLPs. The network is also trained with a batch size lesser than the vanilla version owing to its large size. Considering the difference in performance between previous state-of-the-art networks and the current state-of-the-art models, an increase in mIoU value by 0.2% of 85.4% from 85.2% can still be seen as an overall improvement. The mcIoU value has also shown a similar improvement.
  Dense vs Residual: The network with dense connections seems to outdo the network with residual connections. The mcIoU comparison shows the dense network, with 0.6% higher value, has a significant advantage over the residual counterpart. One explanation could be possibly due to the dynamic graph update scenario where learning the residual might not sit right due to new graphs being formed anew in each layer. The dense connections which carry concatenated features from its previous layers might convey more information about its neighborhood in its previous layers and hence be more effective. Thus, this brings us to the conclusion about the first hypothesis H_1 as being True.

Hypothesis H_1: If deeper networks lead to performance improvements with convolutions on images, they should also aid in improving performance with convolutions on point clouds.

Experiment 2

- **Objective**: The objective of this experiment is to verify the effect of point-pair similarity loss (PPS or PP Loss) as proposed in the previous chapter. It is also verify the impact of the loss for various weightage when trained along with the softmax cross entropy loss for the segmentation network.

- **Experimental setup**: The train and test setup is the same as in **Experimental Setup - 2.**

  - In addition, here the weightage for the loss value is:

$$L = \lambda_1 L_{Point-seg} + \lambda_2 L_{PPS} \tag{5.1}$$

    where $\lambda_1 = 1.0$ and λ_2 is set to 0.3,0.6 and 0.9 for experimental runs.
  - The total number of point pairs sampled is 2000 and is equally shared by the different point pair combinations.

- **Results**: The architecture used and the corresponding bar plots for different runs of lambda values are shown in the Figure 5.2.

- **Discussion**: The results show that the point-pair loss seems to have an adverse effect on the networks performance. The mIoU drops by almost 0.5% and the mcIoU drops by almost 1.5% compared to the baseline network. This could be due to various reasons:
  - Improper weightage of loss values
  - Sampling of the points used for edge prediction could be crucial for steering the segmentation network.
  - The desired effect of pulling the point features closer may not be better for segmentation than letting the network learn by itself.
  - The edge segmentation network itself might not be the best suited and might need more modifications than what is presented.

Thus these points brings us to the conclusion about the second hypothesis H_2 being False.

Hypothesis H_2: Learning to classify edges between points of same parts and different parts leads to learning discriminative features and hence improves the segmentation performance of the network.

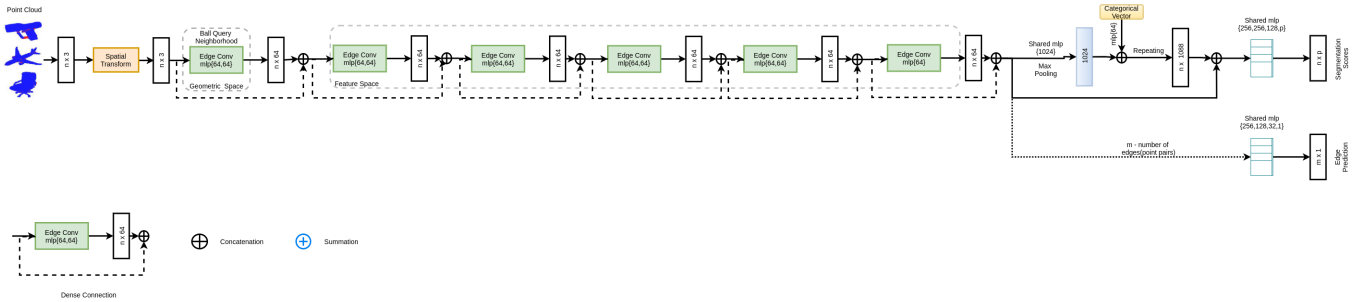
Experiment 3

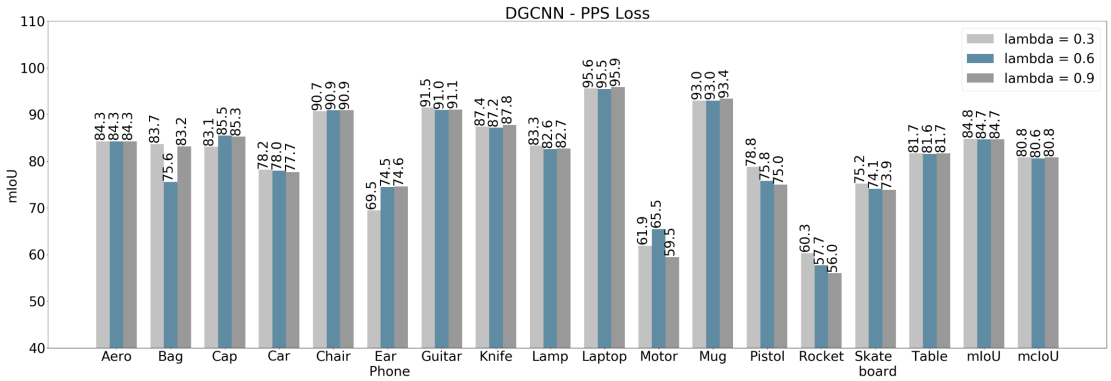
- **Objective**:The objective of this experiment is to verify the effect of adopting the attention mechanism within the edge convolution module.

- **Experimental Setup**: The experimental setup is the same as in **Experimental Setup - 2.** In addition to that, the attention mechanism can have multiple heads and this experiment uses the dense network with multiple head of attention in the EdgeConv layer operating in the input $\mathbb{R}^3$ space.
  - *Attention type:* Multi-Headed Attention
  - *No. of heads*: 3

- **Results**:The results are shown in Figure 5.3 where the architecture used and the bar plot of comparison w.r.t the dense network without the edge attention mechanism can also be viewed at the bottom of the image.

- **Discussion**: The network has shown a similar improvement in performance like the vanilla dense network did w.r.t the vanilla DGCNN version. The overall improvement in mIoU is by 0.4% compared to the baseline network and the improvement over the dense version is by 0.2%. The





	Aero	Bag	Cap	Car	Chair	Ear Phone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate board	Table	mIoU	mcIoU
# Samples	2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271	16881	16881
lambda = 0.3	84.3	83.7	83.1	78.2	90.7	69.5	91.5	87.4	83.3	95.6	61.9	93.0	78.8	60.3	75.2	81.7	84.8	80.8
lambda = 0.6	84.3	75.6	85.5	78.0	90.9	74.5	91.0	87.2	82.6	95.5	65.5	93.0	75.8	57.7	74.1	81.6	84.7	80.6
lambda = 0.9	84.3	83.2	85.3	77.7	90.9	74.6	91.1	87.8	82.7	95.9	59.5	93.4	75.0	56.0	73.9	81.7	84.7	80.8

Figure 5.2: DGCNN PPS Loss: The network architecture used is presented above.

mcIoU has improved by 0.2% compared to the baseline network. The network was also tested with the attention mechanism in all the EdgeConv layers but resulted in a degradation of performance to 84.9% overall mIoU which is 0.7% lesser than when the attention mechanism is only applied to the input layers. Other key observations also from additional experimental runs:

- The performance improvement achieved compared to the *computational cost* increase is impressive when compared to the dense model's improvement over the vanilla version.
- The multiple attention heads doesn't increase the performance compared to a single head attention mechanism.
- The application of attention to the feature space neighbors in the subsequent layers might also be due to the same reason as discussed before, the dynamic graph update, since the neighborhood is the only major difference compared to the input layer. This is yet to be verified by further experimentation.

The overall improvement in mIoU and mcIoU compared to the baseline network brings us to the conclusion about the third hypothesis H_3 as being True.

Hypothesis H_3: Deploying a learning based neighborhood feature aggregation strategy, such as an attention mechanism, with EdgeConv is better than a max-pooling aggregation operation and hence will result in overall performance improvement.

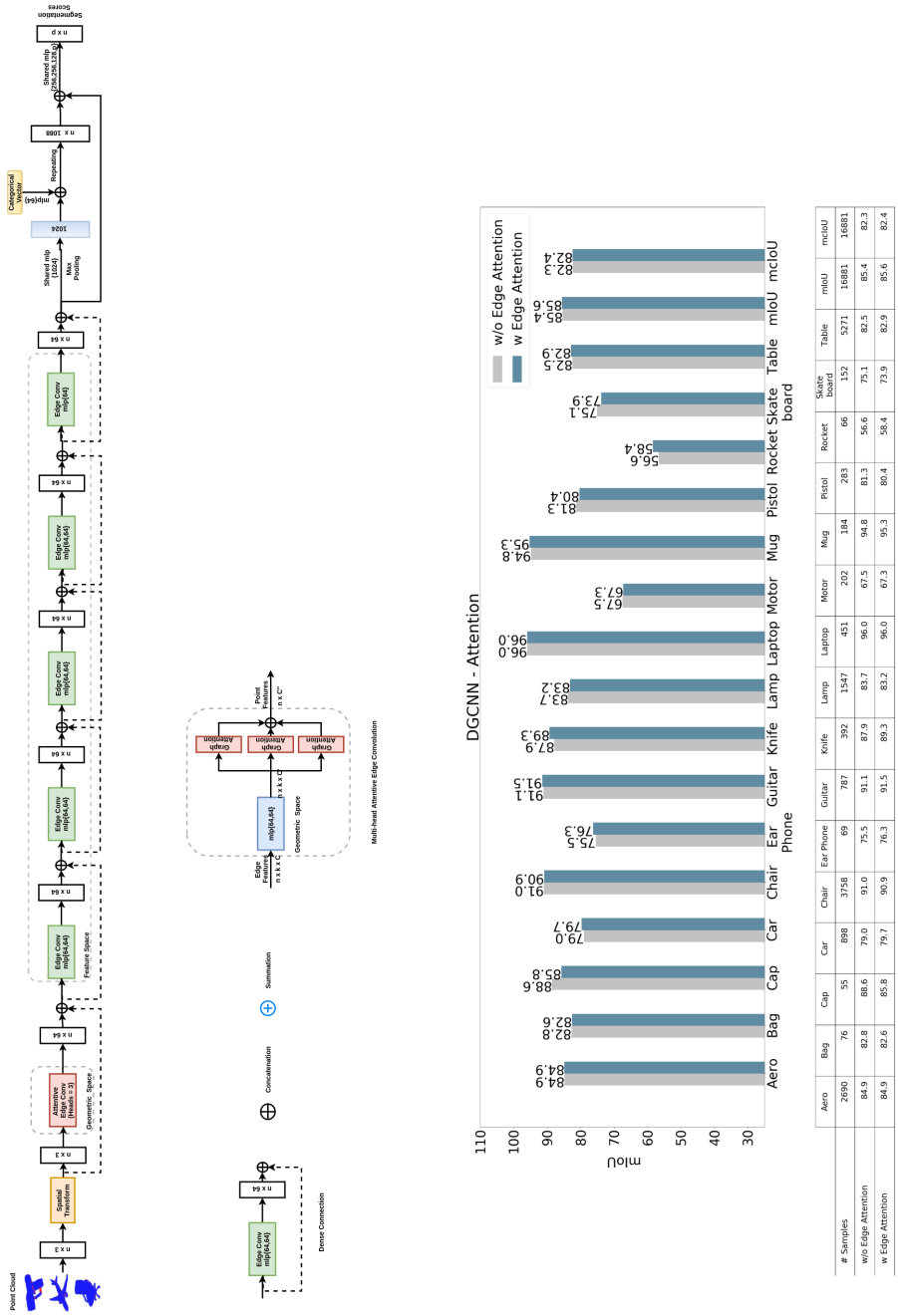
Experiment 4

- **Objective**: The objective of this experiment is to test the idea of regional part presence in order to learn regional part descriptors to aid in segmentation.

- **Experimental Setup**: The train and test setup is the same as in **Experimental Setup - 2** with only the following additional changes and inputs as in Table 5.2. The values for the RPP loss parameters and neighborhood sizes can be found in the table.

- **Results**:The results and the network architecture are shown as in Figure 5.8. The RPP network is combined to the previous best version of the network which has only one segmentation branch with attention mechanism in the input layer. The threshold values for each part is obtained using the SCut method with the validation set.

- **Discussion**: The network outperforms all the other versions that have been experimented with until this point. The network achieves a best overall mIoU of **86.4%** and an mcIoU of **83.6%**. This



	Aero	Bag	Cap	Car	Chair	Ear Phone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate board	Table	mIoU	mcIoU
# Samples	2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271	16681	16681
w/o Edge Attention	84.9	82.8	88.6	79.0	91.0	75.5	91.1	87.9	83.7	96.0	67.5	94.8	81.3	56.6	75.1	82.5	85.4	82.3
w Edge Attention	84.9	82.6	85.8	79.7	90.9	76.3	91.5	89.3	83.2	96.0	67.3	95.3	80.4	58.4	73.9	82.9	85.6	82.4

Figure 5.3: DGCNN Multi-Head Attention.

Experimental Setup - 3	
Batch Size	12
Learning rate	0.0003
α	1.0
γ	2.0
k_1	20
k_2	50
n_{min}	3
λ_1	0.75
λ_2	0.25

Table 5.2: DGCNN RPP - Training Hyperparameters and Setup

is an overall improvement of **1.2%** of mIoU and **1.4%** of mcIoU compared to the baseline vanilla version of the network. The following key observations can be made:

- The RPP network outperforms the previous best version of the network in 12 of the 16 categories.
- The network performs exceedingly well in the categories which contain temporary parts, such as Airplane,Table and Chairs.
- None of the categories in which the network doesn't exceed in performance seem to not do well, indicating that the network is at least as good as the network without the parallel RPP network.
- The threshold values obtained from the validation set seem to generalize well to the test set resulting in the overall increase in performance.
- The loss weightage works out well when it is in favor of the segmentation network(i.e. 0.75 vs 0.25) rather than equal weightage.
- Notice that the network is not the deeper version and only contains 6 EdgeConv layers overall similar to the dense and attention version and hence doesn't add much to the *computational cost*.
- While multi-scale approaches as tried in PN++ have failed to result in a significant increase in performance, the RPP approach which trains the convolutions on the larger scale to predict the part composition instead of the segmentation might be the right way to exploit multi-scale features.
- The predictions made by the RPP network and the final refined predictions where the network is able to tackle noisy points is seen in the figures 5.4 and 5.5.
- Just as in the case with individual category mIoUs, the individual part IoUs also seem to be at least as good as the unrefined predictions as shown in the Figures 5.6 and 5.7

Thus, as discussed above, the RPP network has produced exceedingly better results and overall mIoUs and mcIoUs than the previous approaches. This is further verified by detailed quantitative and qualitative

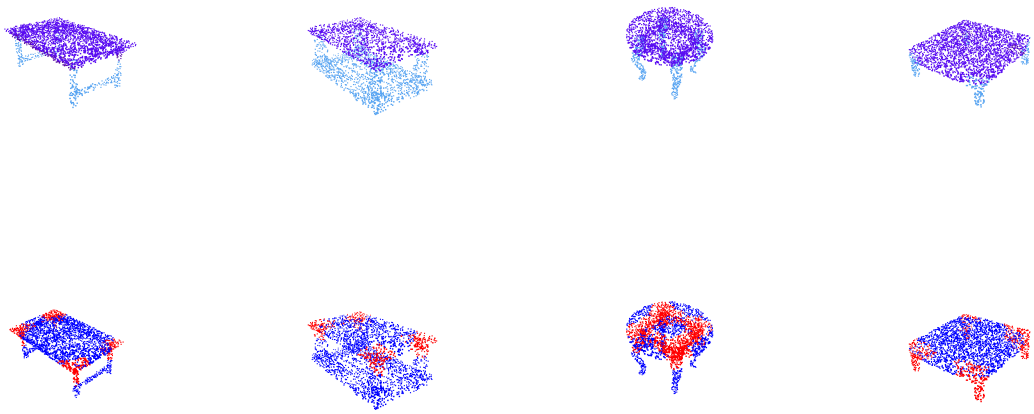


Figure 5.4: GT vs RPP : Points predicting the presence of two or more parts are marked red.

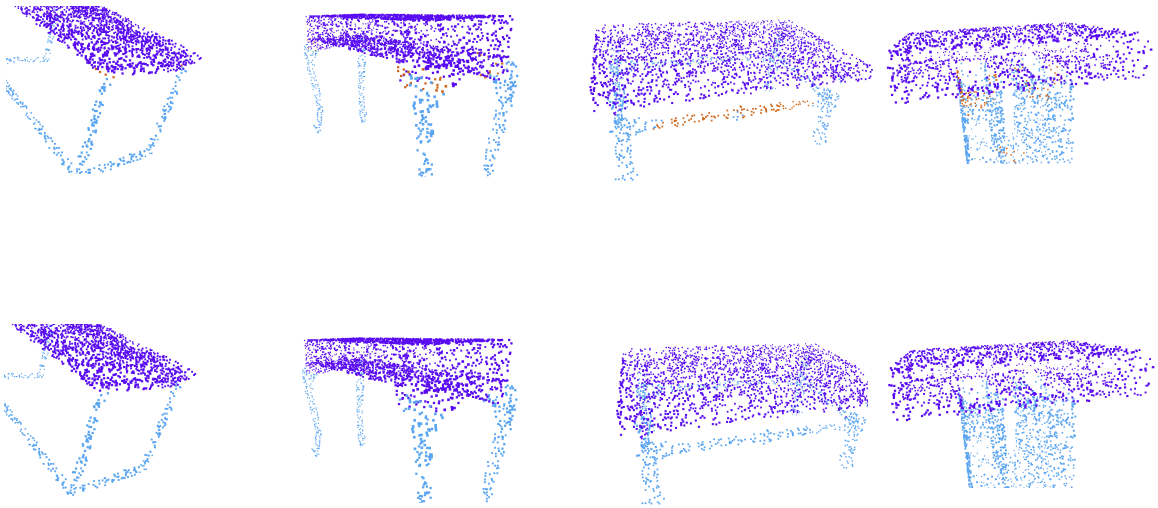


Figure 5.5: Unrefined vs Refined predictions.The noisy predictions of table drawer(brown points) are refined in the final refined predictions.(**Best viewed zoomed in**)

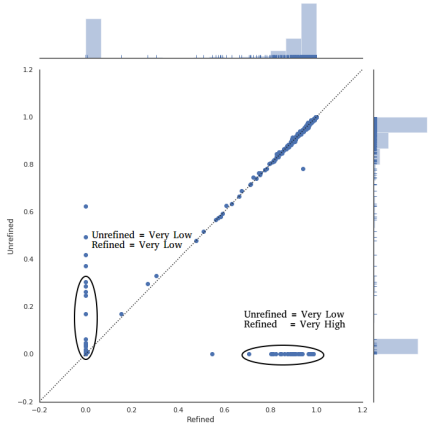


Figure 5.6: Table drawers IoU: The refined predictions are at least as good as the unrefined predictions.

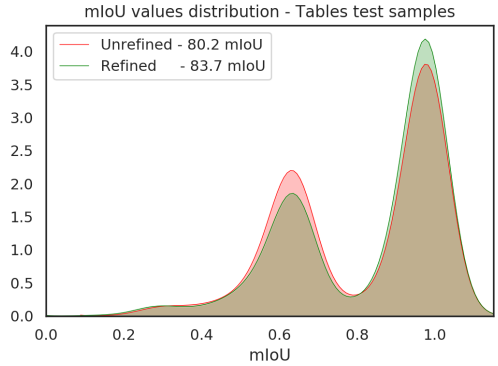


Figure 5.7: Table mIoUs: The distribution shows the large number of improved sample mIoUs for tables resulting in much higher mIoUs.

analysis of the network predictions. This brings us to the conclusion regarding the fourth hypothesis H_4 as being True.

Hypothesis H_4: Learning to predict the presence of other parts in a point's neighborhood as an auxiliary task can help in providing more regional contextual information for pointwise segmentation predictions and hence improve its performance.

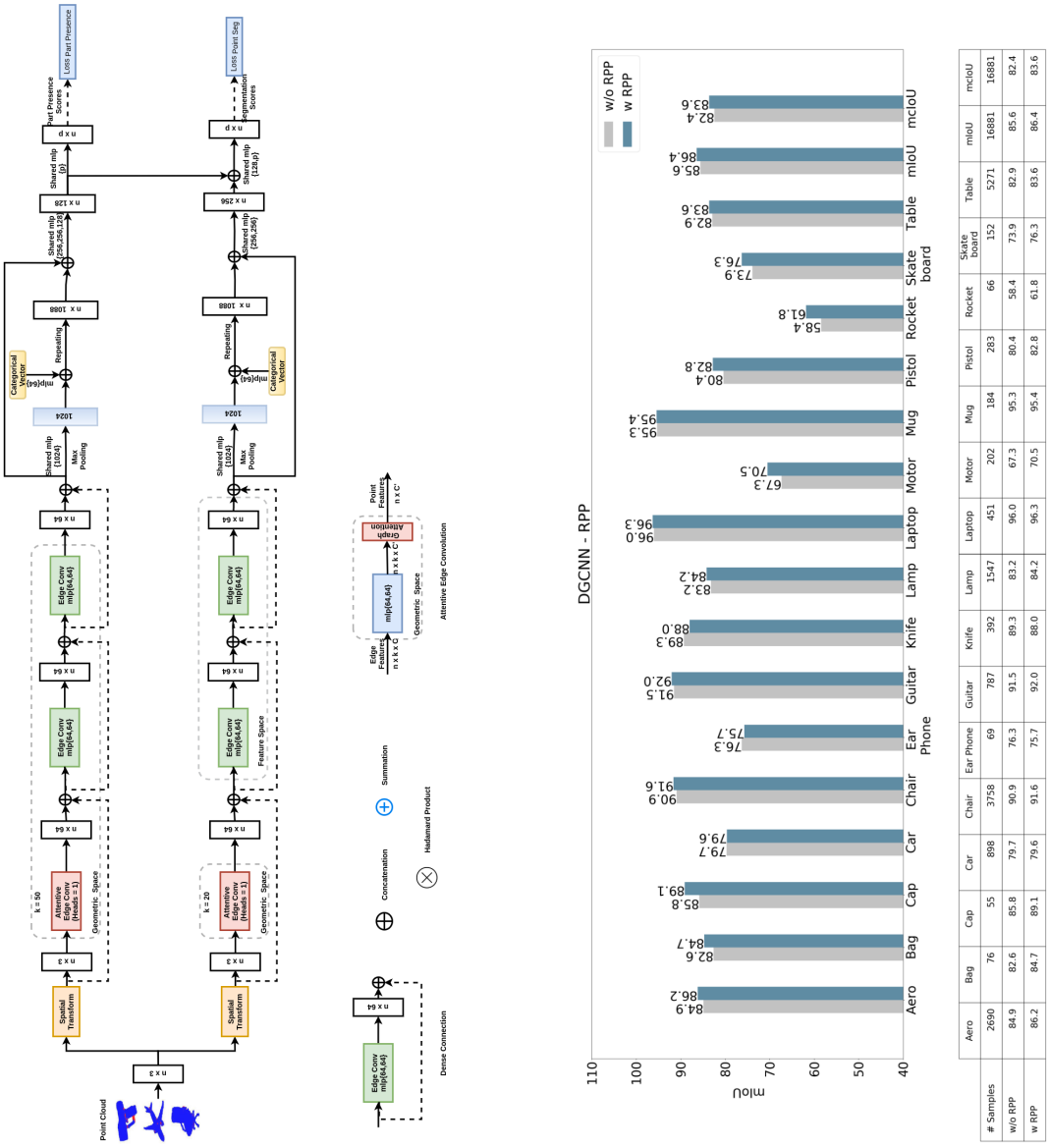
Experiment 5

- **Objective**: The objective of this experiment is to test the network with the proposed class balancing loss to tackle the imbalance posed by the dataset.

- **Experimental Setup**: The train and test setup is the same as in **Experimental Setup -2**.

- **Results**: The results obtained are shown in the Figure 5.9. The architecture used for this test was the deep and dense network with attention in the input EdgeConv layer. The weights for the class balancing are obtained depending on the portion of samples of various object categories in the train set.

- **Discussion**: The following key observations can be made from the results:



	Aero	Bag	Cap	Car	Chair	Ear Phone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate board	Table	mIoU	mcIoU
# Samples	2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271	16881	16881
w/o RPP	84.9	82.6	85.8	79.7	90.9	76.3	91.5	89.3	83.2	96.0	67.3	95.3	80.4	58.4	73.9	82.9	85.6	82.4
w RPP	86.2	84.7	89.1	79.6	91.6	75.7	92.0	88.0	84.2	96.3	70.5	95.4	82.8	61.8	76.3	83.6	86.4	83.6

Figure 5.8: DGCNN - Regional Part Presence.

- The network with class balancing outperforms the other network without any class balancing in 10 out of the 16 categories.
- In spite of outperforming in most categories, the network performs surprisingly bad in the table category where there is a huge imbalance between different parts. This could be partly attributed too the extremely high and low weights for the corresponding parts within the table category which might have had a negative effect. This might also suggest that the weights might have to be within a set limits in order for the balancing to have a desired effect.
- The most difficult category of rockets has seen a drastic improvement by almost **3.6%**.
- Other difficult categories with four parts with large number of examples like Aero, Car and lamp have improved significantly.
- The overall mcIoU has increased by 0.4% while the overall mIoU has decreased by 0.2%.

As listed above, the network has shown desired improvements in certain aspects but the balancing loss has also affected the network with some undesired results. The mcIoU has increased while the mIoU has decreased which makes it tough to decide on the hypothesis H_5. Due to the above discussed reasons, the test on hypothesis H_5 still remains inconclusive.

Hypothesis H_5: The drastic imbalance in the portion of dataset samples for each part degrades the performance of the network and can be compensated by a weighted loss function.

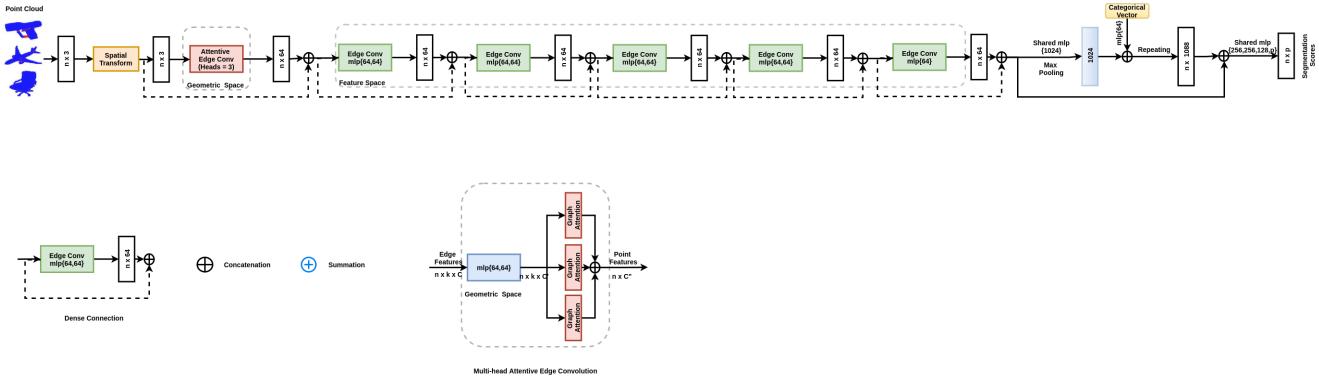
Experiment 6

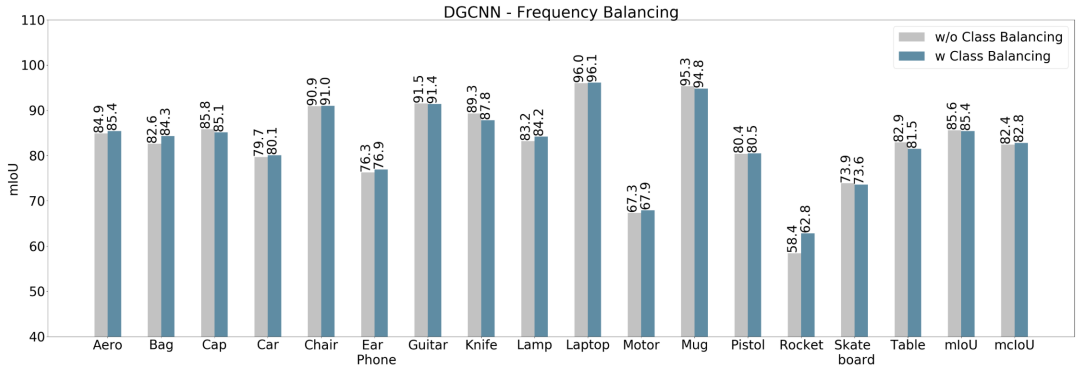
- **Objective**: The objective of this experiment is to verify the impact of the boundary loss proposed by this work.

- **Experimental Setup**: The experimental setup is the same as in **Experimental Setup - 3** used to train the RPP network.

- **Results**: The results obtained are shown in the Figure 5.10. The RPP architecture is used for this experiment and the loss weightage is applied for both the network predictions.

- **Discussion**: The following key observations can be made from the obtained results:
  - The network achieves the best mcIoU value so far of **83.9%**.
  - The category of airplanes reaches its highest mIoU level of **86.5%** and the category of cars reaches its highest mIoU of **81.1%**.





	Aero	Bag	Cap	Car	Chair	Ear Phone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate board	Table	mIoU	mcIoU
# Samples	2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271	16881	16881
w/o Class Balancing	84.9	82.6	85.8	79.7	90.9	76.3	91.5	89.3	83.2	96.0	67.3	95.3	80.4	58.4	73.9	82.9	85.6	82.4
w Class Balancing	85.4	84.3	85.1	80.1	91.0	76.9	91.4	87.8	84.2	96.1	67.9	94.8	80.5	62.8	73.6	81.5	85.4	82.8

Figure 5.9: Class Balancing loss: The architecture used is shown above the plot.

- The category of earphone improves by **2.7%** and skateboards by **2.0%**.
- A similar pattern of improvements is observed across objects with lesser number of samples and lesser number of parts.
- The performance for objects with larger number of samples is almost as good as the network without the boundary loss indicating no loss in performance.
- The increase in mcIoU is not reflected in the mIoU due to the categories associated with the increase in performance contribute less to the mIoU value owing to fewer number of samples.
- Categories like knife, bag and cap with only two parts mostly suffer from boundary point errors and the improvements in their mIoU values is a clear indicator that the loss is having the intended effect.

With an overall mcIoU of 83.9% and mIoU value of 86.4%, this network trained with boundary loss seems to be the best solution proposed so far for the problem of segmentation on complete point clouds. Given the overall best performance and the results indicating that the loss is having the intended effect of bettering the boundary region predictions, the hypothesis H_6 can be considered to be as True.

Hypothesis H_6: Boundary regions are one of the most difficult to segment and a loss function which focuses on the boundary points will help in improving overall performance.

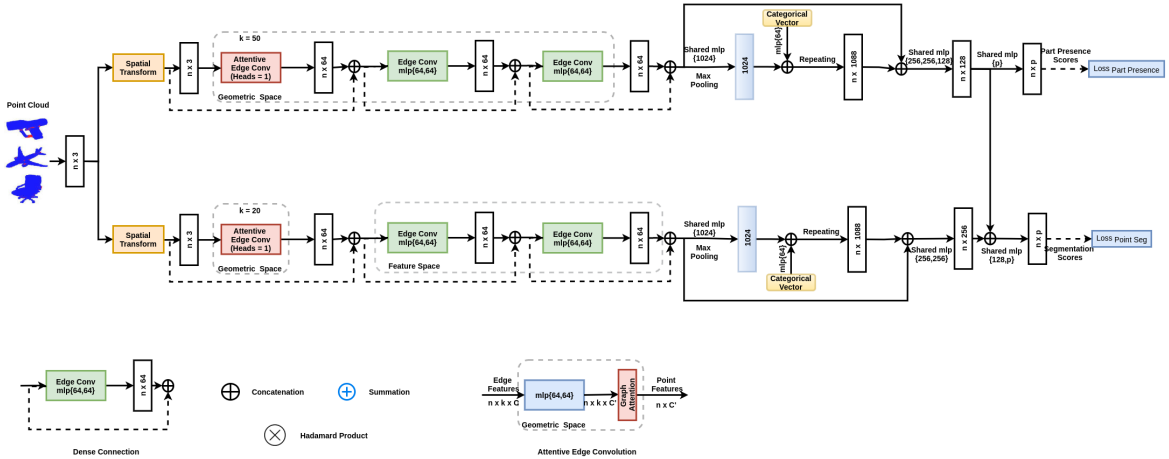
5.2 Partial Point Cloud Segmentation

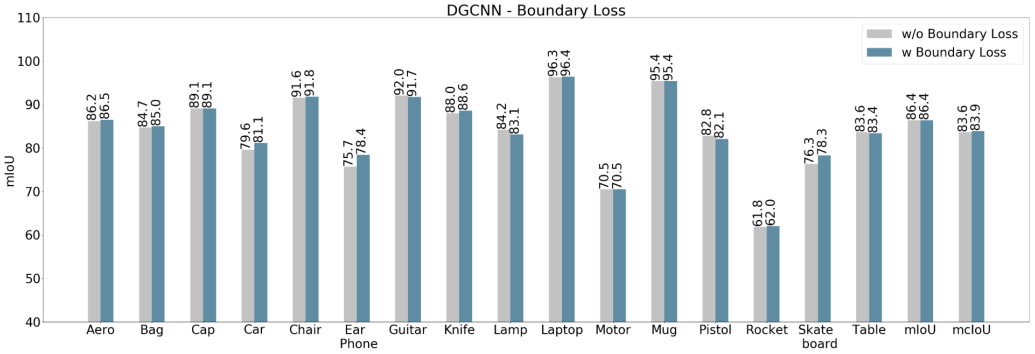
In this section, we will look into the experimental evaluations for the methods proposed to tackle partial point cloud segmentation. The results are presented and key observations are discussed followed by a judgment on the hypothesis to be tested.

Experiment 7

- **Objective**: The objective of this experiment is to verify the impact of training on partial data for the task of partial point cloud segmentation. The partial data is generated according to the method proposed in this work and tested based on the occlusion metrics discussed earlier which are based on mIoUs for different occlusion quantiles.

- **Experimental Setup**: The experimental setup details are provided in the table 5.3. The network architecture used for this experiment is the vanilla version of DGCNN.
- **Results**: The occlusion test results are shown in the Figure 5.11. The mIoU calculated with respect to different levels of occlusion can be found in the plot.
- **Discussion**: The following key observations were made during the experimental runs:





	Aero	Bag	Cap	Car	Chair	Ear Phone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate board	Table	mIoU	mcIoU
# Samples	2690	76	55	898	3758	69	787	392	1547	451	202	184	283	66	152	5271	16881	16881
w/o Boundary Loss	86.2	84.7	89.1	79.6	91.6	75.7	92.0	88.0	84.2	96.3	70.5	95.4	82.8	61.8	76.3	83.6	86.4	83.6
w Boundary Loss	86.5	85.0	89.1	81.1	91.8	78.4	91.7	88.6	83.1	96.4	70.5	95.4	82.1	62.0	78.3	83.4	86.4	83.9

Figure 5.10: Boundary Loss: The boundary loss is tested on the RPP network shown above the plots.

Experimental Setup - 4	
Batch Size	24
Learning rate	0.0001
Optimizer	Adam Optimizer
LR Decay rate	0.5
Decay Steps	400000
LR Clip	1e-6
BN Decay	0.5
BN Decay Rate	0.5
BN Decay Steps	800000
CPU	Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz
GPU	2x NVIDIA Titan RTX (Turing)
RAM	24GB
DL Framework	TensorFlow 1.13(GPU)

Table 5.3: DGCNN - Hyperparameters and Setup for Training on Partial Data

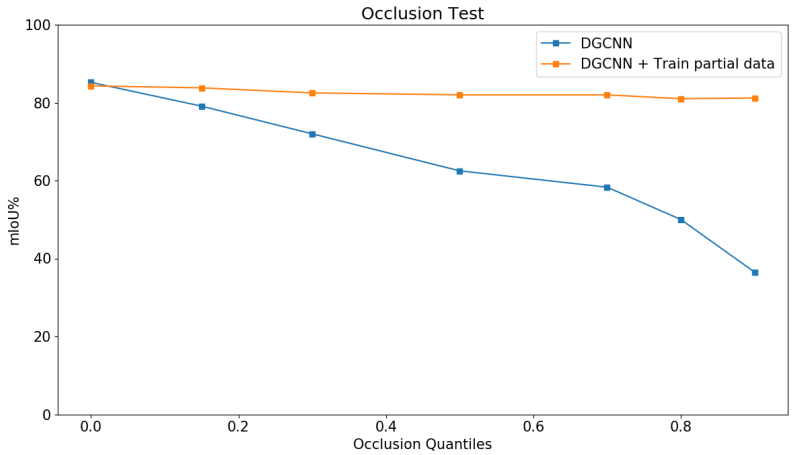


Figure 5.11: Performance of DGCNN trained on partial data.

- The network trained on augmented partial data considerably outperforms the baseline network's performance when trained without partial data.
- The network achieves an mIoU of **81.2 %** at the most extreme case of occlusion where 90% of points are dropped. This is an increase by around **44.7%** compared to the baseline network.
- The network maintains almost the same level of performance over all occlusion quantiles.
- The performance on complete data of **84.3%** has declined by a small margin compared to the network trained only on complete point clouds which yields **85.2%**. This decline was something that was expected and can be seen as a minor trade-off.
- It was also observed that the network shows similar performance when trained with only 4 occlusion samples for each point cloud in the train data. For an increase of dataset size by only 4×, the performance increment is significantly large.

Thus, the above discussed points gives us a clear picture of the benefits of training with partial data

although only with minor trade-offs. This brings us to the conclusion about the hypothesis proposed earlier, hypothesis H_7 as being True.

Hypothesis H_7: Training on partial point clouds will improve the overall performance of the network for partial point cloud segmentation.

Experiment 8

- **Objective**: The objective of this experiment is to verify the impact of the multi-task learning framework with shape completion as an additional task alongside segmentation.
- **Experimental Setup**:
  The experimental setup for this experiment is the same as that of **Experimental Setup - 4** with the additional inputs and modifications as shown in Table 5.4. The final number of predicted points is represented as n_{final}. The loss weightages are listed in the table. The architecture used can be seen in Figure 5.13.

Experimental Setup - 5	
Batch Size	12
Learning rate	0.001
λ_1	1.0
λ_2	20.0
n_{final}	1024

Table 5.4: Multi-task learning with shared encoder - Hyperparameters

- **Results**: The results are shown as in the Figure 5.12. A comparison with the previous two approaches can be seen.

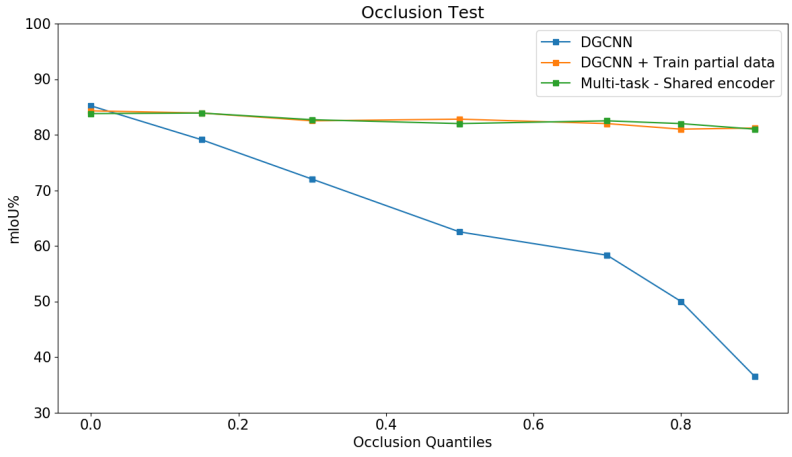


Figure 5.12: Performance of proposed multi-task learning with a shared encoder.

- **Discussion**: The following key observations were made with the experimental runs:

- The network does not show any improvement in performance compared to the vanilla DGCNN trained on partial data.
- There is no negative impact of the shared encoder as seen from the results.
- One reason for this performance could be the *loss weightage*, since finding the right loss weightage between the two tasks can have a significant impact on the overall performance. This additional *hyperparameter* has to be tested with various combination in order to make a final conclusion regarding the network.
- Another reason could be that the shared encoder is not powerful enough and not able to capture substantially more information than when being used only for one task.

For the above said reasons, the network either needs to be made more powerful by opting more layers or different architectures or at least the loss weightage should be tuned to enable better learning. This brings us to the conclusion on the hypothesis H_8 as being False.

Hypothesis H_8: A multi-task learning framework with a shared encoder to leverage the commonalities between shape completion and semantic segmentation will help improve the overall performance on partial point cloud segmentation.

Experiment 9

- **Objective**: The objective of this experiment is to test the impact of having a dedicated segmentation encoder for in the multi-task learning framework with a shared shape completion encoder.
- **Experimental Setup**: The experimental setup is the same as in the previous experiment **Experimental Setup - 5** as shown in Table 5.4. The network architecture used can be seen if Figure 5.15.
- **Results**: The results are as shown in the Figure 5.14. The comparison between all the previous approaches can be seen in the plot.
- **Discussion**: The following key observations were made while training and testing the models for several experimental runs:
  - The network shows a marginal improvement in performance for low levels of occlusion and gradually improves as the occlusion levels rise.
  - The network achieves an overall mIoU of **83.4%** at the most extreme occlusion level compared to the previous best of **81.2%**.
  - There mIoU values of categories of objects which contain only two parts was noted to be significantly higher, almost touching a 100%. This could partly be due to the relative ease with which the network can classify all the points as belonging to just one category in most cases

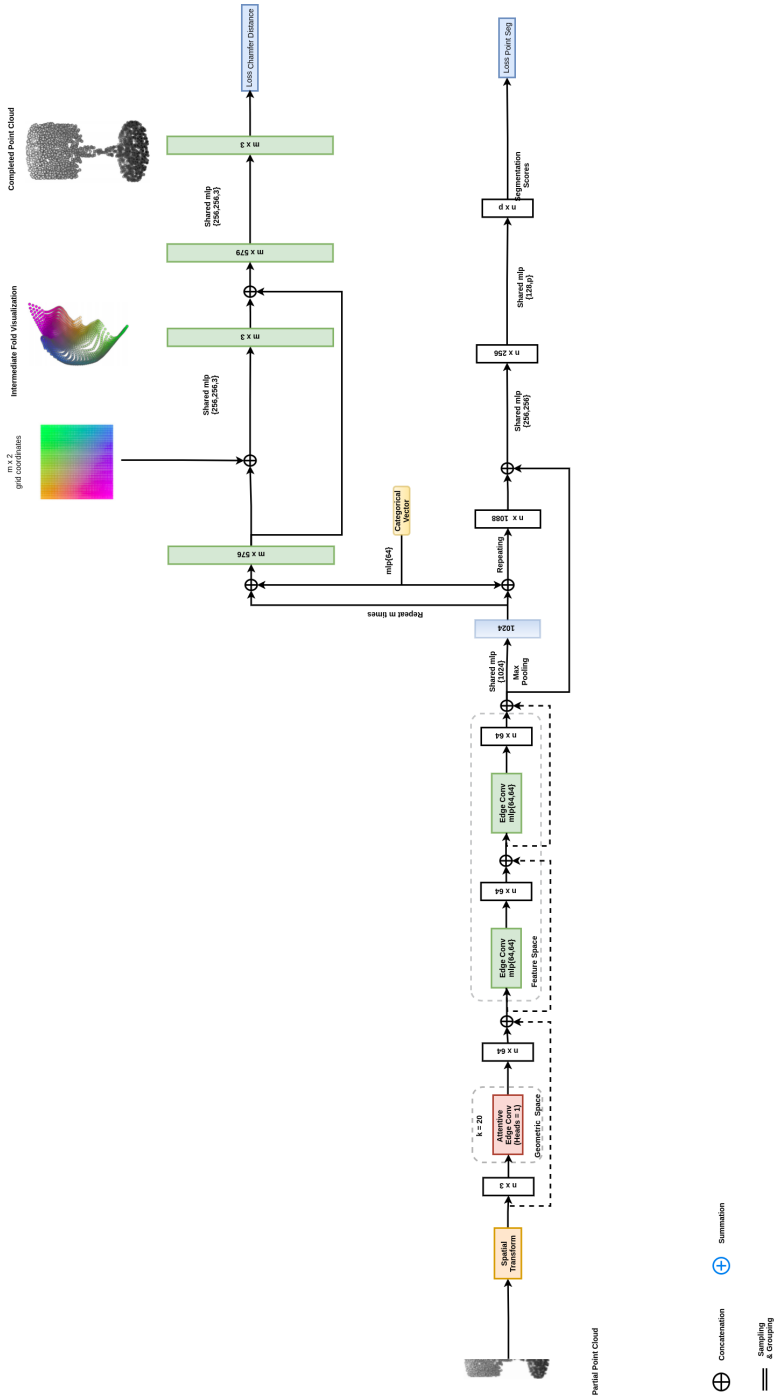


Figure 5.13: Shared Encoder architecture for multi-task learning for partial point cloud segmentation.

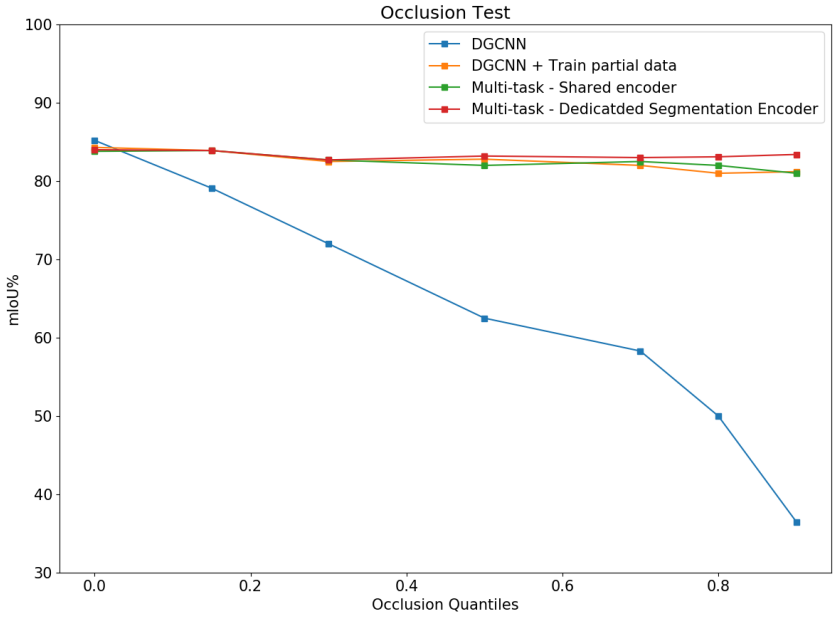


Figure 5.14: Performance of multi-task learning with dedicated segmentation encoder.

due to the other part being occluded. There is no loss performance due to boundary regions or noisy predictions in such cases and hence the high levels of performance.

From the above discussions, it can be said that the usage of a dedicated segmentation decoder has a positive impact on the overall performance of the network. Considering the already impressive performance of the previous networks, the improvements achieved can be considered as a success. There is a clear difference in performance as the occlusion levels increase and hence has turned out to be a desired result. This brings us to the conclusion on the hypothesis H_9 as being True.

Hypothesis H_9: A multi-task learning framework with a dedicated segmentation encoder and a shared shape completion encoder to leverage the commonalities between shape completion and semantic segmentation will help improve the overall performance on partial point cloud segmentation.

5.3 Semantic Shape Completion

Experiment 10

- **Objective**: The objective of this experiment is to test the network architecture proposed for semantic shape completion.

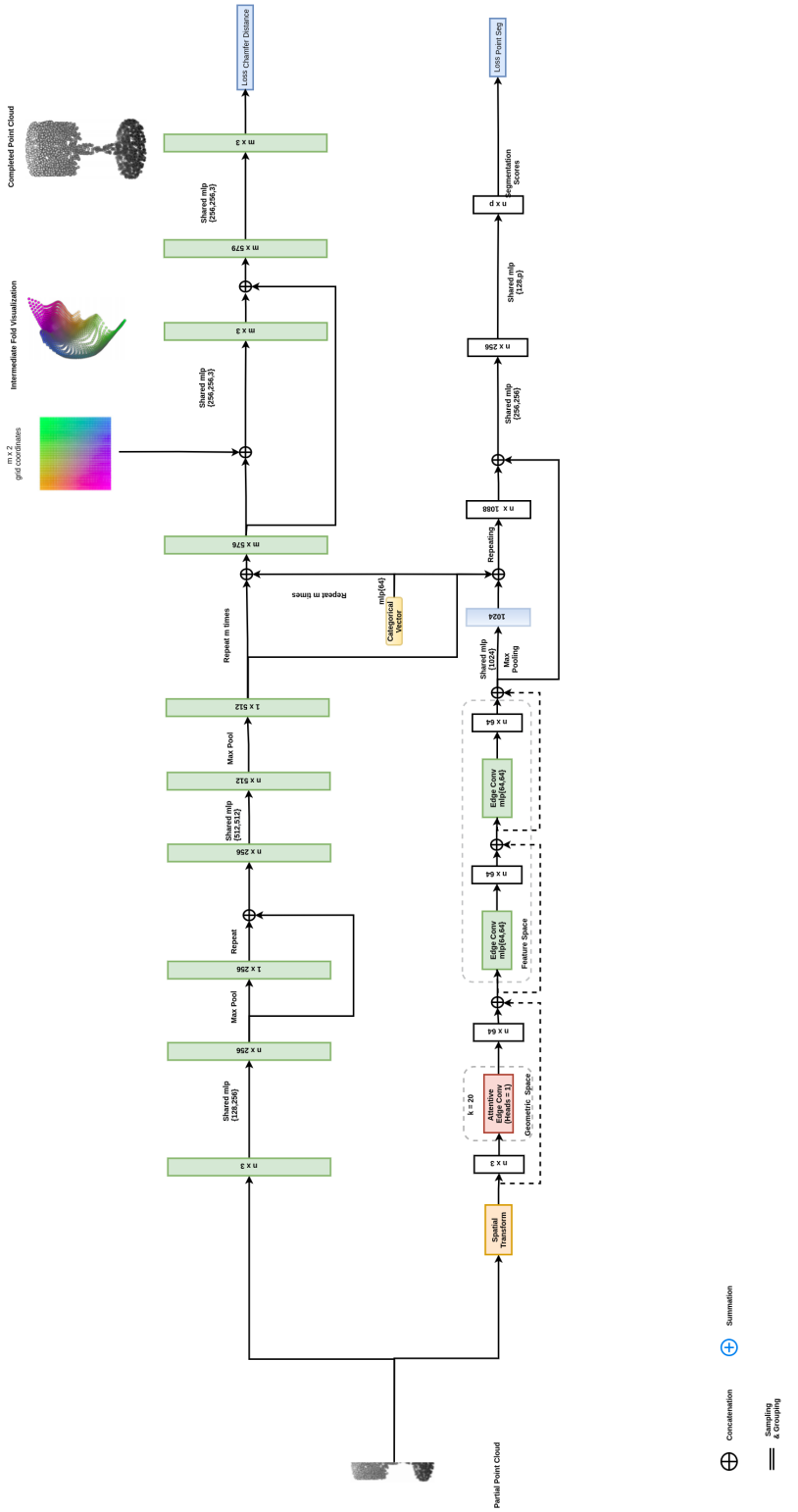


Figure 5.15: Dedicated Segmentation Encoder architecture for multi-task learning for partial point cloud segmentation.

- **Experimental Setup**: The experimental setup is as shown in Table 5.5. The architecture used can be seen in Figure 5.17. In order to measure the mIoU values, a simple nearest neighbor labeling approach is used to map the labels onto the partial point cloud from the predicted complete point cloud.

Experimental Setup - 6	
Batch Size	32
Learning rate	0.0003
Optimizer	Adam Optimizer
LR Decay rate	0.5
Decay Steps	200000
LR Clip	1e-6
BN Decay	0.5
BN Decay Rate	0.5
BN Decay Steps	400000
λ_1	1.0
λ_2	0.20
λ_3	0.20

Table 5.5: PCN Based Semantic Shape Completion - Training Hyperparameters and Setup

- **Results**: The results are seen as shown in Figure 5.16. The comparison to previous approaches can be seen in the plot.
- **Discussion**: The following key observations were made during the experimental runs:
  - The semantic shape completion network performs better than the vanilla DGCNN network but still significantly lags behind the previous approaches proposed for training on partial point clouds.
  - The network maintain almost the same level of performance through all occlusion quantiles with some increase seen as the occlusion levels increase.
  - The increase in performance with occlusion can be attribute to the same reason as discussed before of the objects with just two parts.
  - Notice that a similar increase in performance with occlusion levels is observed with the previous approach using a shared completion encoder.
  - The network produces a performance of around **73%** at the minimum as well as the maximum occlusion levels.
  - Even though the overall mIoU might be lower than the previous approaches, it can be considered as a trade-off for the additional semantic information obtained through the network which can be critical in real world scenarios.

Thus, considering the network didn't breakdown like the vanilla DGCNN version and also considering the enriched information obtained through the semantic shape completion network. This can be considered as

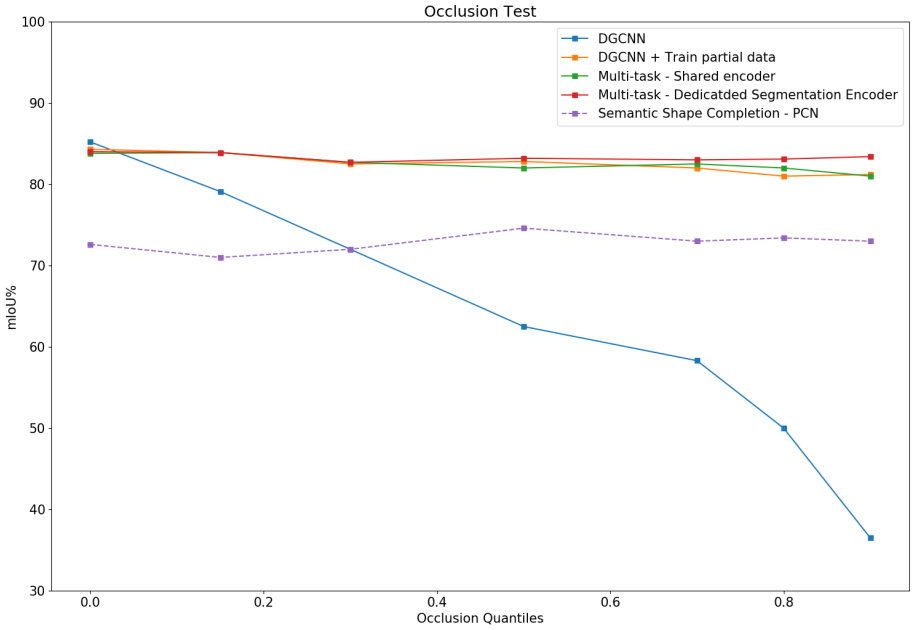


Figure 5.16: Performance of PCN based semnatic shape completion.

as a success.
This brings us to the conclusion on the hypothesis H_{10} as being True.

Hypothesis H_{10}: Shape completion and semantic segmentation are deeply intertwined tasks and hence a joint estimation of complete point clouds and semantic labels can be learnt using a network architecture primarily designed for shape completion.

Experiment 11

- **Objective**: The objective of this experiment is to verify the proposed approach of graph convolution based hierarchical decoder for the semantic shape completion network .
- **Experimental Setup**: The experimental setup is as shown in Table 5.6. The architecture used can be seen in Figure 5.19. In order to measure the mIoU values, a nearest neighbor labeling approach is used as in the previous experiment to map the labels onto the partial point cloud from the predicted complete point cloud.
- **Results**: The results are seen as shown in Figure 5.18. The comparison to previous approaches can

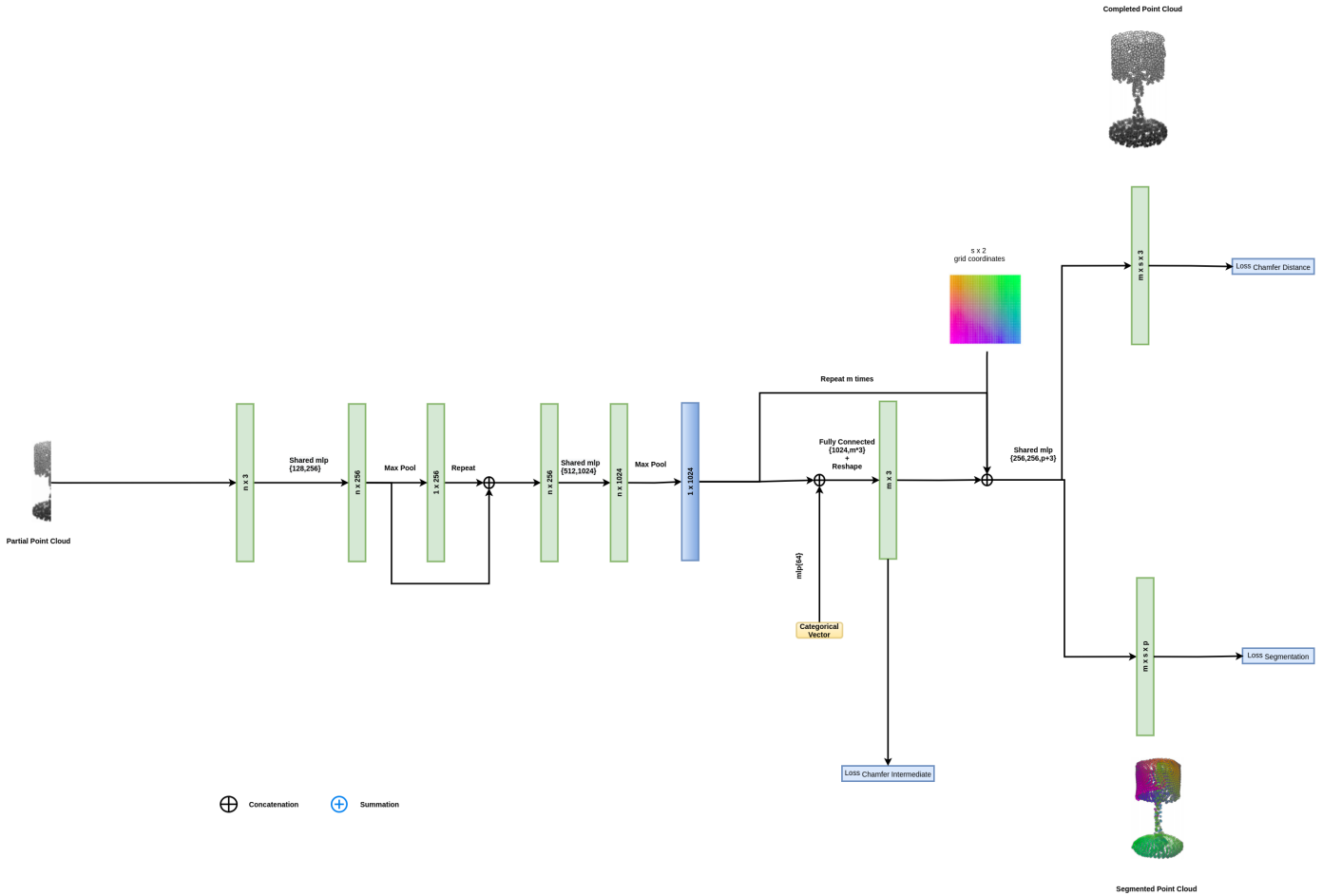


Figure 5.17: PCN based Semantic Shape Completion Network: The architecture modifications at the final layer can be noticed to predict a complete point cloud(gray-scale lamp) and its corresponding semantic labels(colored lamp) compared to the original PCN architecture.

Experimental Setup - 6	
Batch Size	24
Learning rate	0.0003
Optimizer	Adam Optimizer
LR Decay rate	0.5
Decay Steps	200000
LR Clip	1e-6
BN Decay	0.5
BN Decay Rate	0.5
BN Decay Steps	400000
λ_1	10.0
λ_2	0.25
λ_3	0.25

Table 5.6: Graph Convolution Based Hierarchical Decoder for PCN-Semantic Shape Completion Network - Training Hyperparameters and Setup

be seen in the plot.

- **Discussion**: The following key observations were made during the experimental runs:
  - The graph convolution based hierarchical decoder performs significantly better than the folding based decoder in the previous approach.
  - The network shows a performance of around **79%** mIoU for complete point clouds and at minimum levels of occlusion. This is an increase by around **7%** compared to the previous approach.
  - The network shows improvement in performance at all occlusion quantiles compared to the previous approach. At the extreme case, the network shows a performance around **76%**.
  - It was also observed that the object categories with fewer parts, three or two, benefited the most with the integrated approach.
  - The network lags in performance at extreme occlusions by 5% compared to the best results produced so far. This again can be viewed as a trade-off for the additional critical information obtained regarding the semantic labeling of the completed point cloud.

As discussed above, the network performs exceedingly well compared to the previous semantic shape completion network and is not far behind the best result achieved so far on partial point cloud segmentation. Qualitative results are found in the appendix chapter B. This brings us to the conclusion on the hypothesis H_{11} as being True.

Hypothesis H_{11}: A graph convolution based hierarchical decoder can better capture semantic information than a folding based decoder for improving the segmentation performance of the PCN based semantic shape completion network.

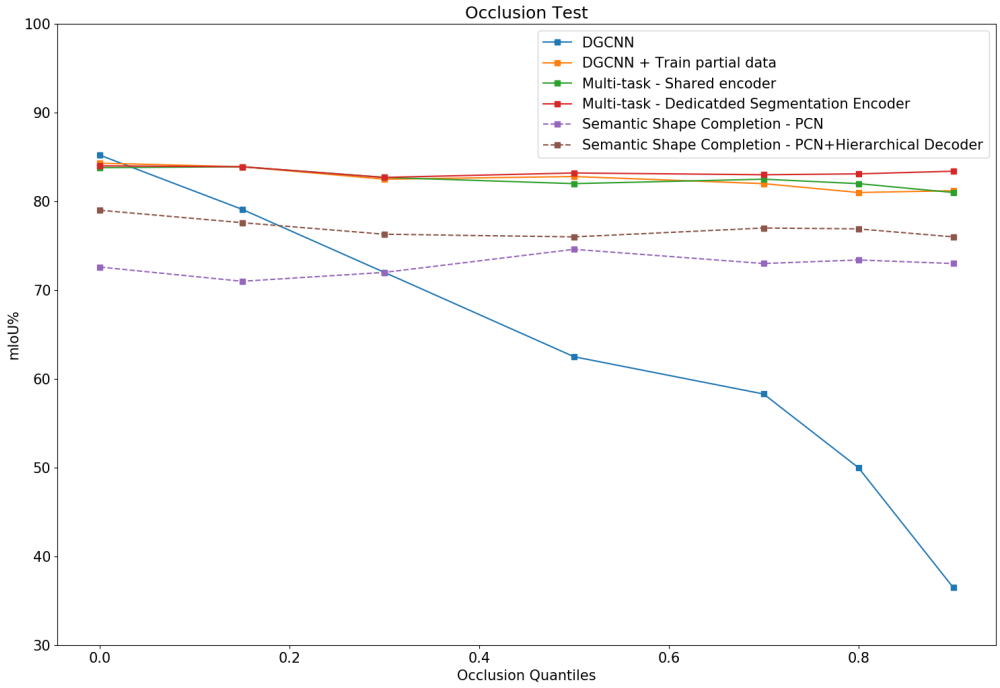


Figure 5.18: Performance of graph convolution based hierarchical decoder for semantic shape completion.

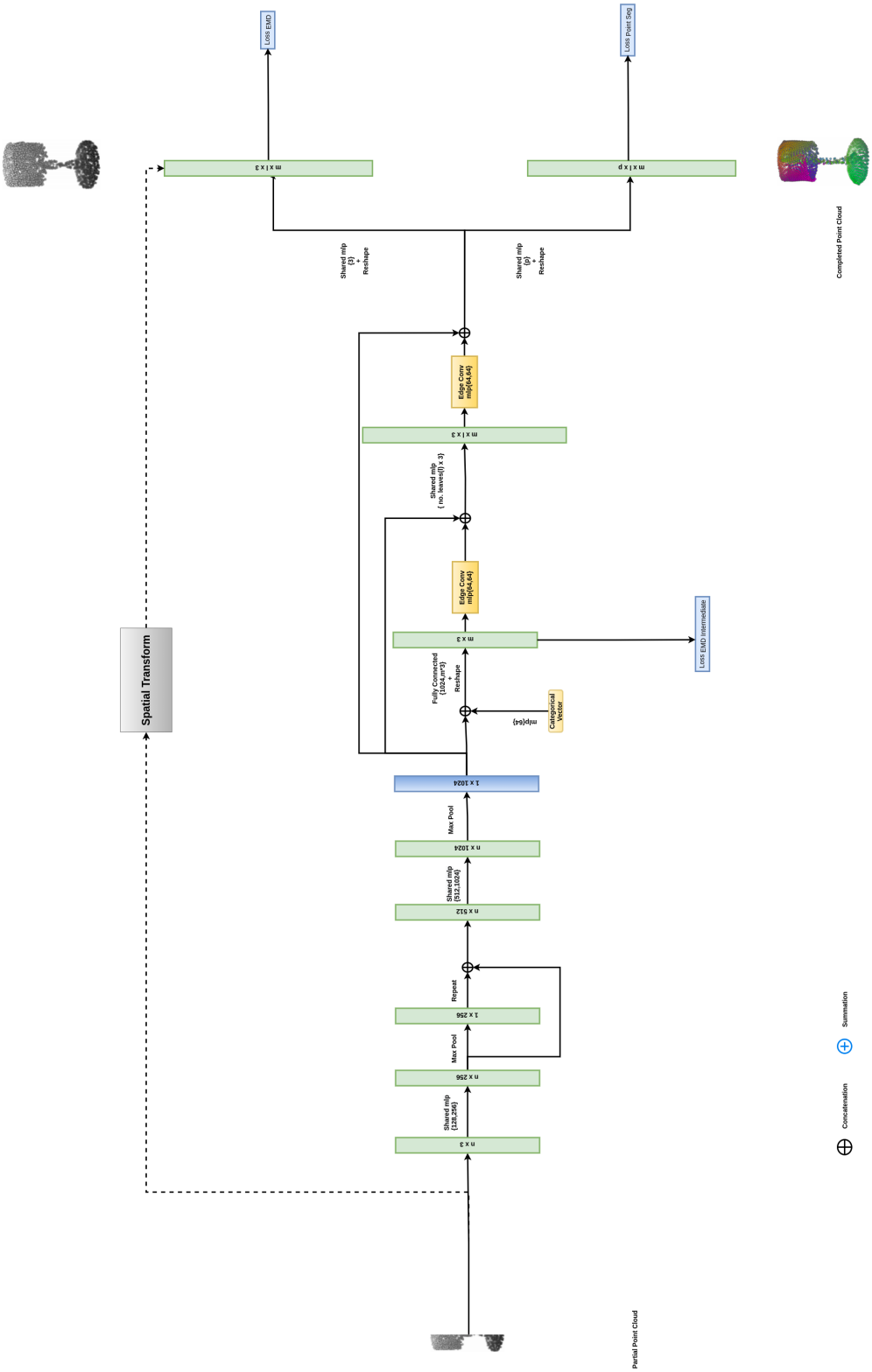


Figure 5.19: Graph convolution based hierarchical decoder: The EdgeConv blocks integrated into the hierarchical framework helps to enrich the semantic information learnt by the network leading to an improvement in the task of partial point cloud segmentation. The spatial transformer's output is used to translate and scale the final layers's prediction of complete point cloud.

6

Conclusions

In this work, the problem of point cloud segmentation was dealt with not only for *complete point clouds* but also for *partial point clouds* taking into consideration the real world scenarios. A benchmark dataset was chosen for complete point clouds and a new strategy to train and test on partial point clouds was proposed and implemented. A graph convolutions based baseline network was chosen along with the benchmark dataset and various techniques were proposed and integrated into the network's learning scheme resulting in significant improvements for both complete and partial point cloud segmentation tasks. While starting out with an aim to effectively exploit local neighborhood features to enrich the semantic information for segmentation tasks, the goal was realized by integrating and successfully testing various ideas like the attention mechanism and the regional part presence scheme for complete point cloud segmentation and also by integrating a graph convolution module into the semantic shape completion network for partial point cloud segmentation. Realizing the importance of processing partial point clouds, a novel partial data generation strategy and a robust testing scheme were proposed and discussed in detail. This work also explored the problem of partial point cloud segmentation through an alternate lens of *semantic shape completion*, which carried with it the benefits of providing complete point clouds with corresponding semantic labels which could be easily mapped on to the partial point clouds thus enriching the overall information held by the system. Along the way, the following milestones were achieved:

- An overall mIoU of **86.4%** on the ShapeNetPart dataset. Until the time of writing this thesis, there is yet no other method to surpass this performance without any preprocessing techniques or voting strategies.
- An overall mIoU of greater than **82%** for all occlusion quantiles on partial point cloud segmentation. Since the task of partial point cloud segmentation is a nascent one, there is yet a consensus to be reached on the ideal metric and benchmark dataset which makes it difficult to compare with other approaches.
- A semantic shape completion network which aids in partial point cloud segmentation and achieves an overall mIoU of over **75%** for all occlusion quantiles.

6.1 Contributions

On a methodological level, the following notable contributions were made by this work:

1. For complete point cloud segmentation, a regional part presence scheme which for the first time has lead to a significant improvement in performance in networks with a multi-scale convolutions approach in the input $\mathbb{R}^3$ layer.
2. A novel training and testing scheme for partial point cloud segmentation.
3. A multi-task learning framework which involves shape completion for aiding in partial point cloud segmentation.
4. A semantic shape completion network trained and tested for 3D object point clouds.
5. A semantic shape completion based partial point cloud segmentation without assuming any prior knowledge of the bounding box parameters of the input partial object.

6.2 Lessons learned

While tackling the problems of segmentation on complete and partial point clouds and aiming to predict semantic labels while learning to complete partial point clouds, the following were the major lessons learnt:

1. In order to effectively evaluate the proposed approaches, more than anything, it is of utmost importance that the dataset is of the highest quality possible. The ShapeNetPart dataset is highly imbalanced with some categories having more than 5000 samples while others having close to a 100 samples only. This not only affects the learning of the network but is also a huge hindrance to compare and evaluate different strategies.
2. With graph neural networks typically requiring large chunks of memory to train, it is of highest importance to be able to tackle the problems by techniques such as gradient checkpointing. Only a subject knowledge of the problem at hand is insufficient while carrying out such projects which requires a lot of additional efforts.
3. In a lot of cases, architectures designed for a defined task can be easily generalized to other tasks which are closely related with each other. In our case, it was between shape completion and semantic segmentation.
4. Finally, it is not always good to head in directions of ideas which give a desired result with respect to particular metric for a complicated task like semantic segmentation.

6.3 Future work

Of all the approaches discussed in this work, semantic shape completion is the most attractive one given its direct practical applications. For this cause, it will be interesting to consider the following future works:

- **Adversarial training**: A graph patch generative adversarial network (GAN) based scheme to denoise and improve the quality of completion predictions will come in handy for segmentation tasks and semantic shape completion. Here, the discriminator is trained to differentiate between local patches of ground truth point clouds and noisy predicted point clouds which can lead to improvement with respect to the noisy predictions.
- **Deep Implicit Functions**: Another attractive option to tackle the problem of noisy point cloud predictions is to lean towards more powerful and smooth representations. A 3D shape representation using deep implicit functions which have been proven to produce accurate reconstructions of shapes is definitely another strategy to look forward to.
- **Part-level Semantic Scene Completion**: In addition to semantically completing shapes in isolation, a more challenging setting would be to integrate the proposed approach to complete partial scans of entire scenes, typically observed by robots, with part-level semantic labeling and not just object instance level semantic scene completion as done in previous works.

A

Dataset - Partial Objects Visualizations

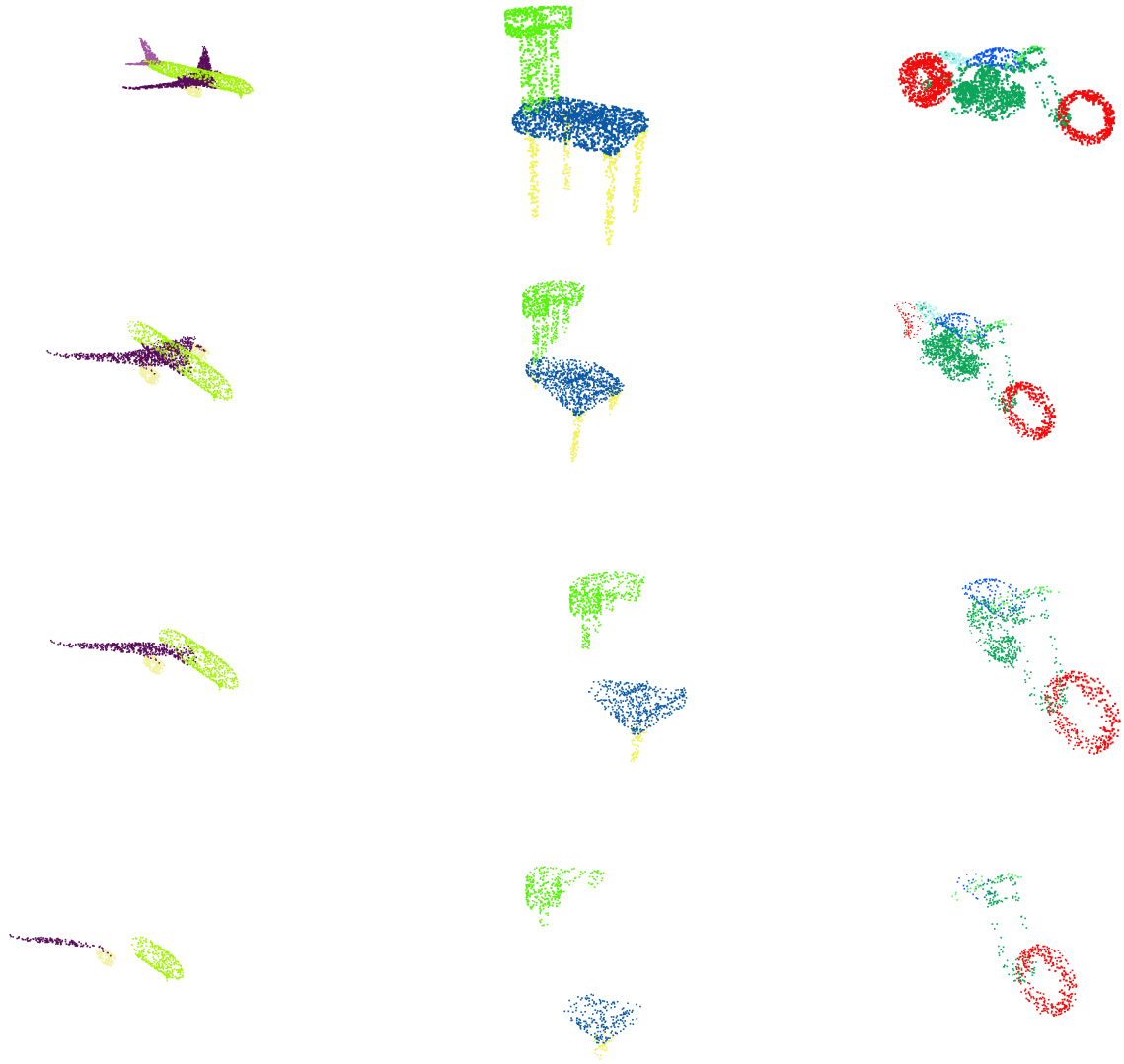


Figure A.1: **Partial Dataset**: Generated Partial Point Cloud Examples. Occlusion percentages of 30%, 60% and 90% for the same slicing plane orientation shown along each row below starting with the first row of complete point clouds.

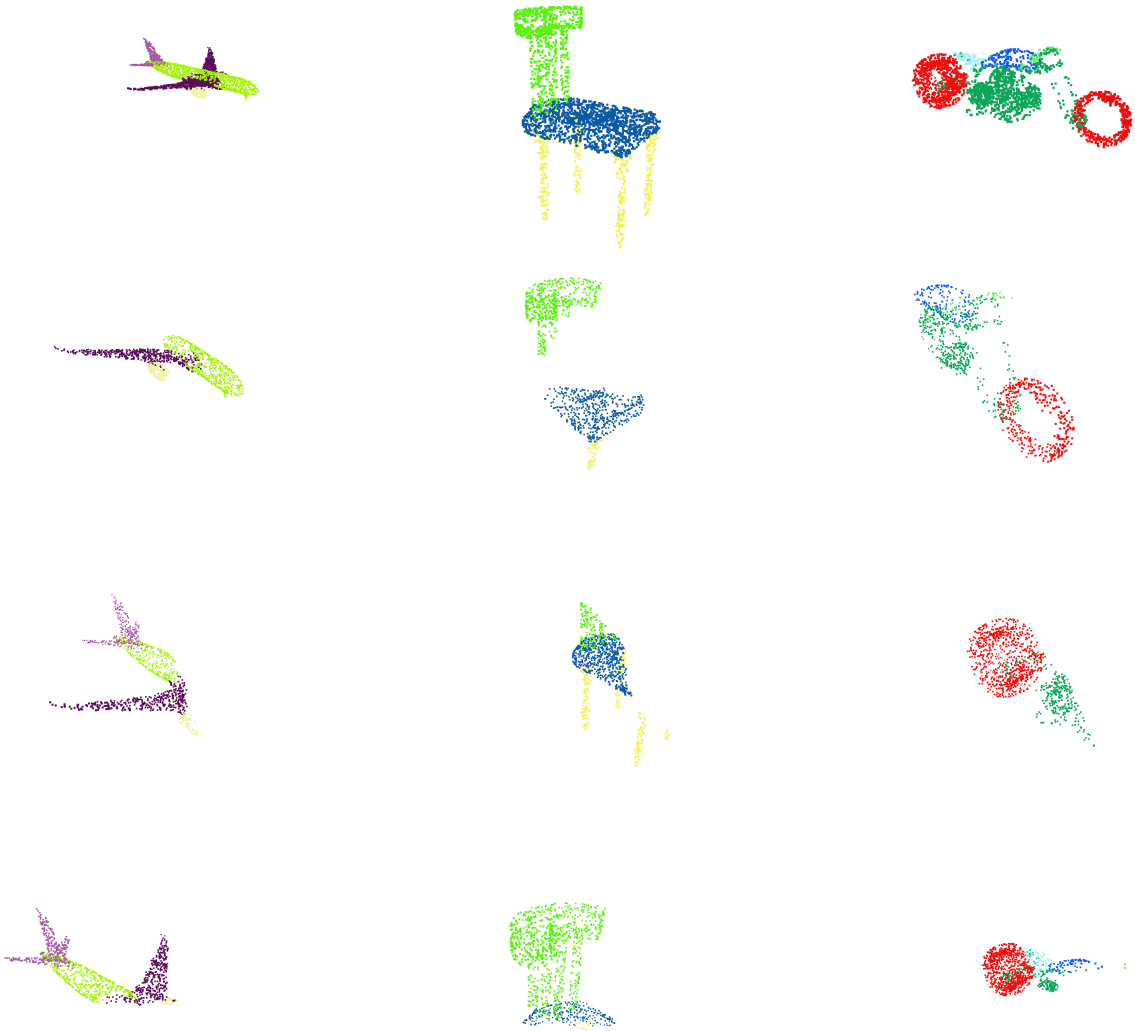


Figure A.2: **Partial Dataset**: Generated Partial Point Cloud Examples. Occlusion percentages of 50% with different slicing plane orientations shown along each row below starting with the first row of complete point clouds.

B

Qualitative Results

This chapter is dedicated to qualitative results for semantic shape completion.

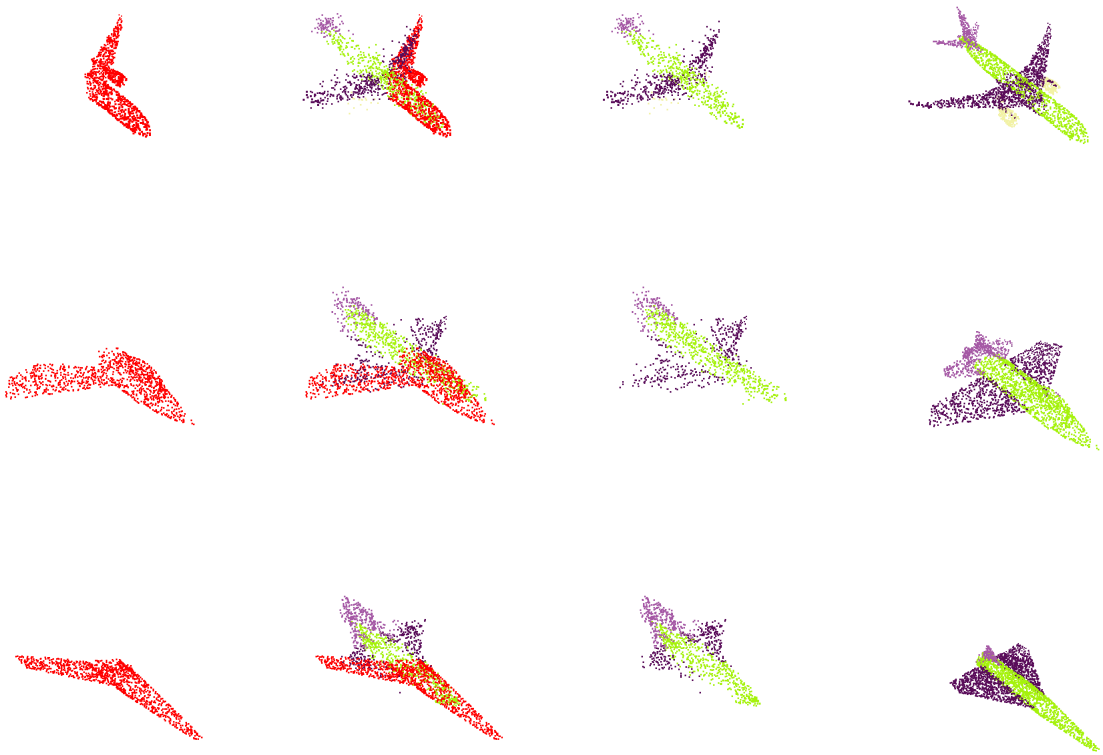


Figure B.1: Semantic Shape Completion - Qualitative Results. From left column to right: Input, Prediction plus input merged, Prediction, GT

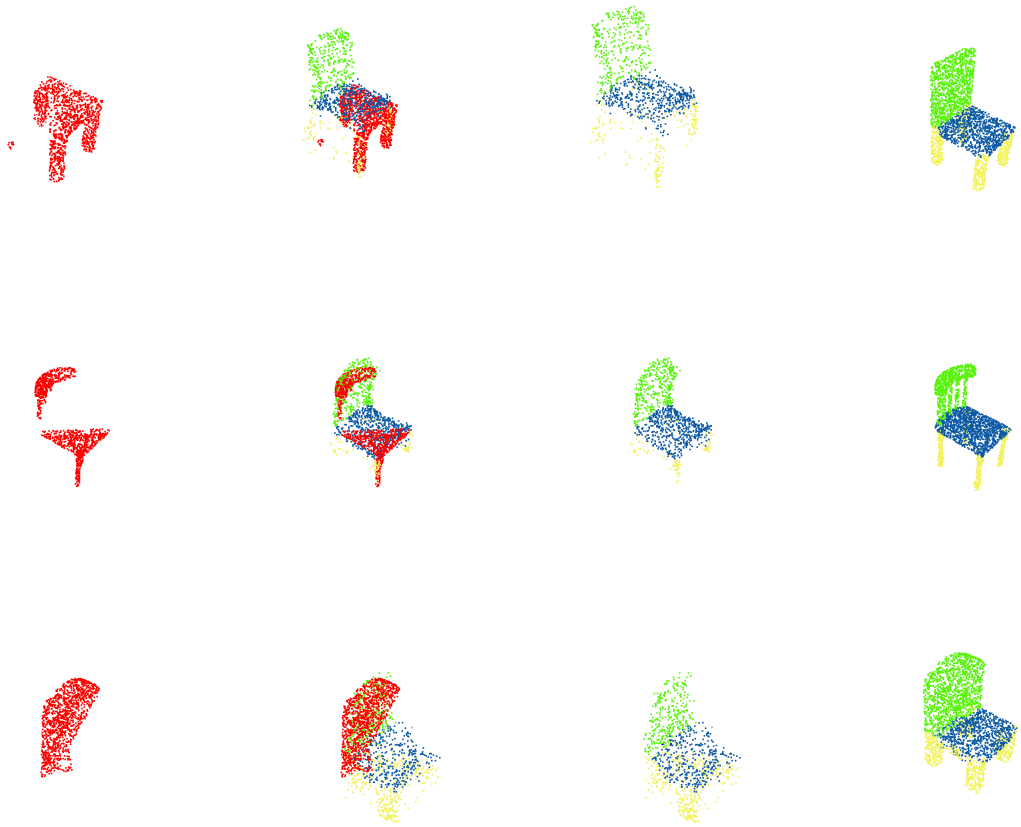


Figure B.2: Semantic Shape Completion - Qualitative Results. From left column to right: Input, Prediction plus input merged, Prediction, GT

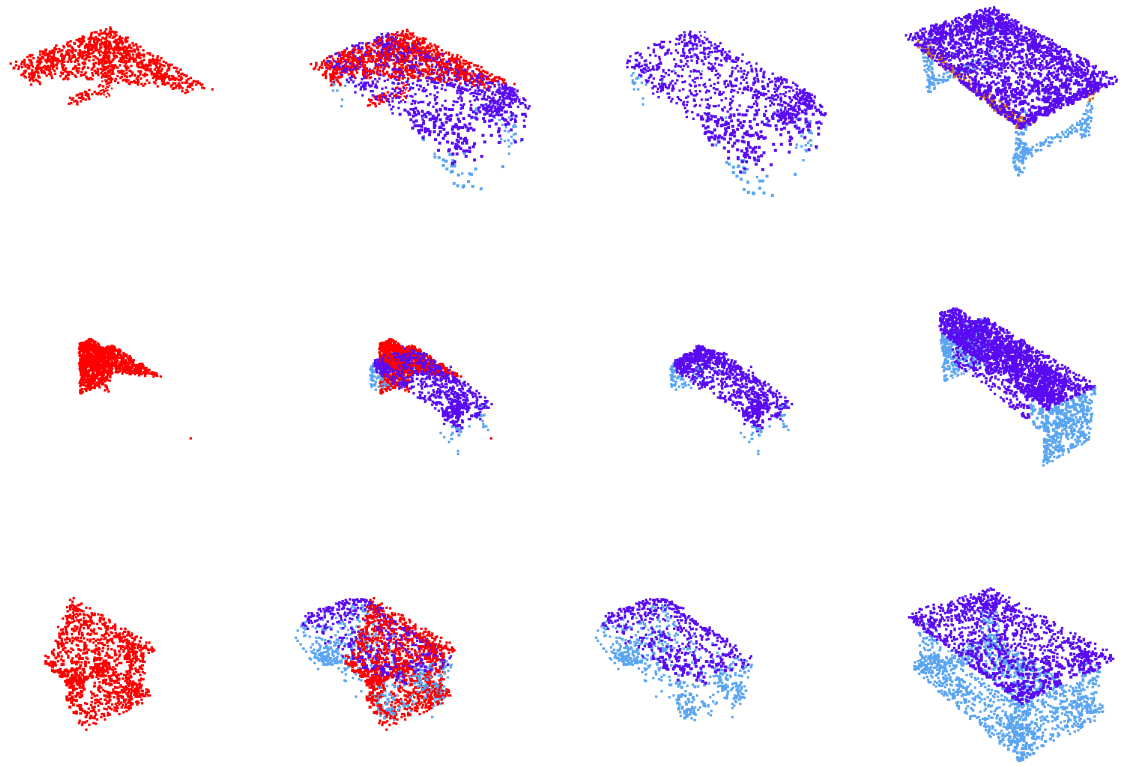


Figure B.3: Semantic Shape Completion - Qualitative Results. From left column to right: Input, Prediction plus input merged, Prediction, GT

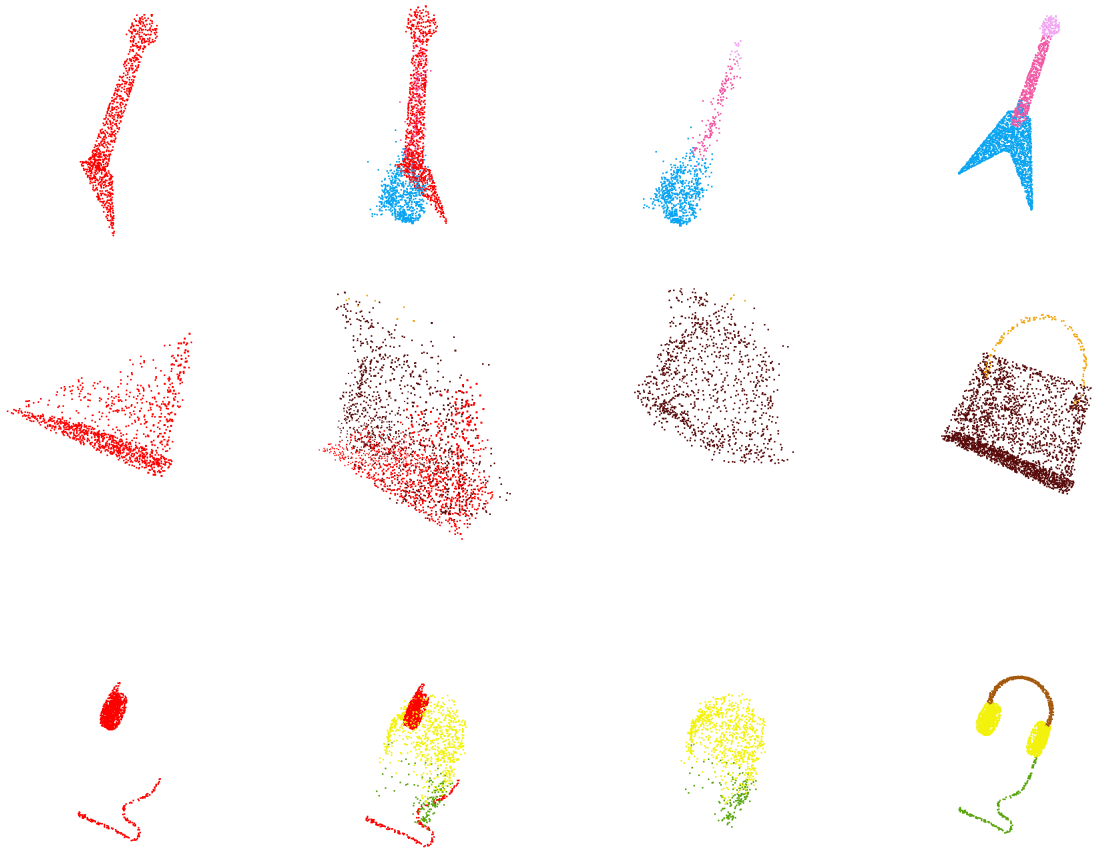


Figure B.4: **Failures**: Semantic Shape Completion. From left column to right: Input, Prediction plus input merged, Prediction, GT

C

Ablation Study

This chapter is dedicated to ablation study on a couple of networks for complete and partial point cloud segmentation. The best performing networks are scrutinized through an ablation study in the sections below.

C.1 Complete Point Cloud Segmentation

The best performing network for complete point cloud segmentation is the DGCNN-RPP net with a performance of 86.4% mIoU. The table below shows the results of the ablation study conducted.
The main modules of the method are:

- Parallel Branch: A branch that operates on the static graph on a larger neighborhood.
- RPP Loss: The multi-label loss to train the RPP branch to predict the regional part descriptors.
- SCut Threshold: The threshold technique used to find the right threshold value for the multi-label problem in order to decide whether a part is present or absent.

Network	mIoU %
w/o Parallel branch	85.4
w Parallel branch, w/o RPP Loss	85.4
w Parallel branch and RPP Loss, w/o SCut Threshold	85.9
w Parallel branch and RPP Loss, w SCut Threshold	**86.4**

Table C.1: DGCNN RPP - Ablation Study

The ablation study is conducted in a step-by-step fashion where the whole parallel branch is removed from the network and each component is added one by one. This results in four conditions, which are evaluated and the results are presented in the Table C.1. The results indicate very clearly the effectiveness of the RPP Loss. Even with the addition of the RPP branch with multiple layers of EdgeConv, the performance doesn't improve. This shows that the parallel branch only when working along with the RPP loss can produce the desired results. The overall comparison of the network to other state-of-the-art works is seen in Table C.2.

Network	mcIoU%	mIoU%
kd-Net	77.4	82.3
SO-Net	81.0	84.9
DGCNN*	82.2	85.2
SPLATNet	83.7	85.4
SubSparseCNN	83.3	86.0
PointCNN	84.6	86.1
KP-Conv	**85.1**	**86.4**
DGCNN-RPP	83.9	**86.4**

Table C.2: Comparison - DGCNN RPP on ShapeNetPart

C.2 Partial Point Cloud Segmentation

The best performing network for partial point cloud segmentation is the one with the dedicated segmentation encoder. This network is put to test through a simple ablation study to verify the role of the shape completion branch. The main modules of the network to be tested are:

- The shape completion decoder is the one that guides the network to learn features that leverages the commonalities between the two tasks. Testing the network by removing this will give a good idea on the actual impact of the decoder on the network's performance.

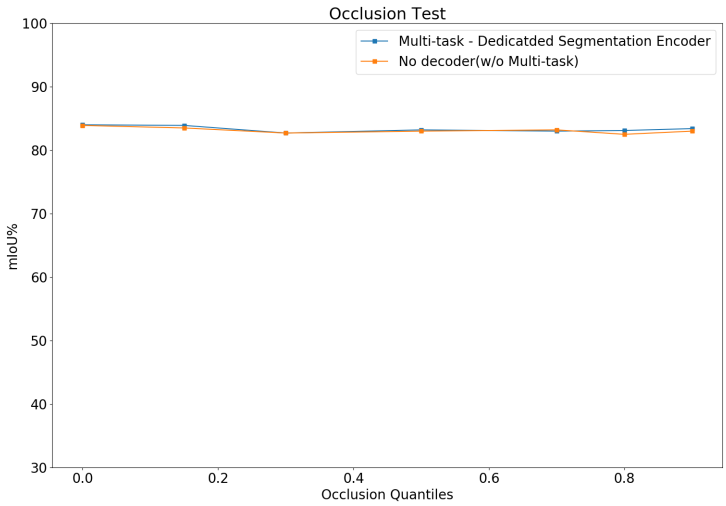


Figure C.1: Performance of the network with the decoder removed.

The results can be seen in Figure C.1 where the network performs almost the same as before. This makes us question two things:

- The decoder might not actually have the intended effect on the learning process

- The two-stage encoder could actually be the difference when compared to the network without the multi-task learning framework.

The results from this study reveals that the decoder actually might not have the intended effect which is contrary to our earlier assumption.

References

[1] Russel E. Caflisch. Monte carlo and quasi-monte carlo methods. *Acta Numerica*, 7:1–49, 1998. doi: 10.1017/S0962492900002804.

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[3] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. *CoRR*, abs/1803.10409, 2018. URL `http://arxiv.org/abs/1803.10409`.

[4] Angela Dai, Daniel Ritchie, Martin Bokeloh, Scott Reed, Jürgen Sturm, and Matthias Nießner. Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans. *CoRR*, abs/1712.10215, 2017. URL `http://arxiv.org/abs/1712.10215`.

[5] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. *CoRR*, abs/1512.04412, 2015. URL `http://arxiv.org/abs/1512.04412`.

[6] Z. Deng and L. J. Latecki. Amodal detection of 3d objects: Inferring 3d bounding boxes from 2d ones in rgb-depth images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 398–406, 2017. doi: 10.1109/CVPR.2017.50.

[7] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. Exploring spatial context for 3d semantic segmentation of point clouds. *CoRR*, abs/1802.01500, 2018. URL `http://arxiv.org/abs/1802.01500`.

[8] Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. Dilated point convolutions: On the receptive field of point convolutions. *CoRR*, abs/1907.12046, 2019. URL `http://arxiv.org/abs/1907.12046`.

[9] Fabian Groh, Patrick Wieschollek, and Hendrik P. A. Lensch. Flex-convolution (million-scale point-cloud learning beyond grid-worlds). In *Asian Conference on Computer Vision (ACCV)*, Dezember 2018.

[10] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey, 2019.

[11] Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds. *CoRR*, abs/1910.08207, 2019. URL `http://arxiv.org/abs/1910.08207`.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

[13] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua, and Timo Ropinski. Monte carlo convolution for learning on non-uniformly sampled point clouds. *CoRR*, abs/1806.01759, 2018. URL `http://arxiv.org/abs/1806.01759`.

[14] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds, 2019.

[15] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Point-wise convolutional neural network. *CoRR*, abs/1712.05245, 2017. URL `http://arxiv.org/abs/1712.05245`.

[16] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Point-wise convolutional neural network. *CoRR*, abs/1712.05245, 2017. URL `http://arxiv.org/abs/1712.05245`.

[17] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL `http://arxiv.org/abs/1608.06993`.

[18] Paul Jaccard. Etude de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547–579, 01 1901. doi: 10.5169/seals-266450.

[19] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu, and Jiaya Jia. Hierarchical point-edge interaction network for point cloud semantic segmentation. *CoRR*, abs/1909.10469, 2019. URL `http://arxiv.org/abs/1909.10469`.

[20] Mingyang Jiang, Yiran Wu, and Cewu Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *CoRR*, abs/1807.00652, 2018. URL `http://arxiv.org/abs/1807.00652`.

[21] Jing Huang and Suya You. Point cloud labeling using 3d convolutional neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2670–2675, 2016.

[22] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. *CoRR*, abs/1612.02808, 2016. URL `http://arxiv.org/abs/1612.02808`.

[23] Roman Klokov and Victor S. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. *CoRR*, abs/1704.01222, 2017. URL `http://arxiv.org/abs/1704.01222`.

[24] S. U. C. G. Laboratory. Stanford bunny. 1993.

[25] Shiyi Lan, Ruichi Yu, Gang Yu, and Larry S. Davis. Modeling local geometric structure of 3d point clouds using geo-cnn. *CoRR*, abs/1811.07782, 2018. URL `http://arxiv.org/abs/1811.07782`.

[26] Loïc Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. *CoRR*, abs/1711.09869, 2017. URL `http://arxiv.org/abs/1711.09869`.

[27] Huan Lei, Naveed Akhtar, and Ajmal Mian. Spherical kernel for efficient graph convolution on 3d point clouds, 2019.

[28] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. *CoRR*, abs/1803.04249, 2018. URL `http://arxiv.org/abs/1803.04249`.

[29] Yangyan Li, Rui Bu, Mingchao Sun, and Baoquan Chen. Pointcnn. *CoRR*, abs/1801.07791, 2018. URL `http://arxiv.org/abs/1801.07791`.

[30] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. URL `http://arxiv.org/abs/1708.02002`.

[31] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing, 2019.

[32] 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, sep 2015. doi: 10.1109/iros.2015.7353481. URL `https://doi.org/10.1109%2Firos.2015.7353481`.

[33] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. Vv-net: Voxel VAE net with group convolutions for point cloud segmentation. *CoRR*, abs/1811.04337, 2018. URL `http://arxiv.org/abs/1811.04337`.

[34] R. Monica and J. Aleotti. Point cloud projective analysis for part-based grasp planning. *IEEE Robotics and Automation Letters*, 5(3):4695–4702, 2020. doi: 10.1109/LRA.2020.3003883.

[35] C. Nash and C. Williams. The shape variational autoencoder: A deep generative model of part-segmented 3d objects. *Computer Graphics Forum*, 36:1–12, 08 2017. doi: 10.1111/cgf.13240.

[36] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. pages 225–230, 11 2013. ISBN 978-1-4799-1201-8. doi: 10.1109/RAM.2013.6758588.

[37] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. *CoRR*, abs/1904.09664, 2019. URL `http://arxiv.org/abs/1904.09664`.

[38] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs / 1612.00593, 2016. URL `http://arxiv.org/abs/1612.00593`.

[39] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet ++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs / 1706.02413, 2017. URL `http://arxiv.org/abs/1706.02413`.

[40] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. *CoRR*, abs/1611.05009, 2016. URL `http://arxiv.org/abs/1611.05009`.

[41] Christoph B. Rist, David Emmerichs, Markus Enzweiler, and Dariu M. Gavrila. Semantic scene completion using local deep implicit functions on lidar data, 2020.

[42] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Neighbors do help: Deeply exploiting local structures of point clouds. *CoRR*, abs/1712.06760, 2017. URL `http://arxiv.org/abs/1712.06760`.

[43] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR*, abs/1704.02901, 2017. URL `http://arxiv.org/abs/1704.02901`.

[44] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR*, abs/1704.02901, 2017. URL `http://arxiv.org/abs/1704.02901`.

[45] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[46] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. *CoRR*, abs/1505.00880, 2015. URL `http://arxiv.org/abs/1505.00880`.

[47] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL `http://arxiv.org/abs/1409.4842`.

[48] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese. Topnet: Structural point cloud decoder. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 383–392, 2019. doi: 10.1109/CVPR.2019.00047.

[49] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. *CoRR*, abs/1710.07563, 2017. URL `http://arxiv.org/abs/1710.07563`.

[50] Gusi Te, Wei Hu, Zongming Guo, and Amin Zheng. RGCNN: regularized graph CNN for point cloud segmentation. *CoRR*, abs/1806.02952, 2018. URL `http://arxiv.org/abs/1806.02952`.

[51] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *CoRR*, abs/1904.08889, 2019. URL `http://arxiv.org/abs/1904.08889`.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

[53] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rJXMpikCZ`. accepted as poster.

[54] Walber. Precision and recall, 2014. URL https://commons.wikimedia.org/wiki/File:Precisionrecall.svg. [Online; accessed 22-Mar-2021].

[55] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. *CoRR*, abs/1803.05827, 2018. URL http://arxiv.org/abs/1803.05827.

[56] S. Wang, S. Suo, W. Ma, A. Pokrovsky, and R. Urtasun. Deep parametric continuous convolutional neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018.

[57] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. SGPN: similarity group proposal network for 3d point cloud instance segmentation. *CoRR*, abs/1711.08588, 2017. URL http://arxiv.org/abs/1711.08588.

[58] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018. URL http://arxiv.org/abs/1801.07829.

[59] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, Jan 2021. ISSN 2162-2388. doi: 10.1109/tnnls.2020.2978386. URL http://dx.doi.org/10.1109/TNNLS.2020.2978386.

[60] Jiachen Xu, Jingyu Gong, Jie Zhou, Xin Tan, Yuan Xie, and Lizhuang Ma. Sceneencoder: Scene-aware semantic segmentation of point clouds with a learnable scene descriptor, 2020.

[61] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Interpretable unsupervised learning on 3d point clouds. *CoRR*, abs/1712.07262, 2017. URL http://arxiv.org/abs/1712.07262.

[62] Yiming Yang. A study of thresholding strategies for text categorization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, page 137–145, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133316. doi: 10.1145/383952.383975. URL https://doi.org/10.1145/383952.383975.

[63] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *SIGGRAPH Asia*, 2016.

[64] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. *CoRR*, abs/1903.00709, 2019. URL http://arxiv.org/abs/1903.00709.

[65] Wentao Yuan, David Held, Christoph Mertz, and Martial Hebert. Iterative transformer network for 3d point cloud. *CoRR*, abs/1811.11209, 2018. URL http://arxiv.org/abs/1811.11209.

[66] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: point completion network. *CoRR*, abs/1808.00671, 2018. URL http://arxiv.org/abs/1808.00671.

[67] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. *CoRR*, abs/1703.06114, 2017. URL http://arxiv.org/abs/1703.06114.

[68] Junming Zhang, Weijia Chen, Yuping Wang, Ram Vasudevan, and Matthew Johnson-Roberson. Point set voting for partial point cloud analysis. *IEEE Robotics and Automation Letters*, 6(2):596–603, Apr 2021. ISSN 2377-3774. doi: 10.1109/lra.2020.3048658. URL http://dx.doi.org/10.1109/LRA.2020.3048658.

[69] Kuangen Zhang, Ming Hao, Jing Wang, Clarence W. de Silva, and Chenglong Fu. Linked dynamic graph CNN: learning on point cloud via linking hierarchical features. *CoRR*, abs/1904.10014, 2019. URL http://arxiv.org/abs/1904.10014.

[70] Zhiyuan Zhang, Binh-Son Hua, David W. Rosen, and Sai-Kit Yeung. Rotation invariant convolutions for 3d point clouds deep learning. *CoRR*, abs/1908.06297, 2019. URL http://arxiv.org/abs/1908.06297.