PointNet: Deep Learning on 3D Point Cloud for Object Classification, Semantic Segmentation and Part Segmentation

Studienarbeit

at University of Siegen
at the Department of Computer Vision

Principal Supervisor:	Prof. Dr. Michael Möller
Secondary Supervisor:	Dr. Zorah Lähner
Author:	Rishav Kumar Paramhans rishav.paramhans@student.uni-siegen.de 1607138
Submission:	September 2022

Abstract

PointNet is a unified Deep Neural Network to carry out object classification, semantic segmentation and part segmentation on 3D Point Clouds. This project aims to implement the PointNet architecture and conduct an in-depth study on it. The object classification, semantic segmentation and part segmentation networks were implemented in the first step. In the second step, different network configurations were implemented by introducing weight initialization and Dropout techniques to improve the model performance. Experiments on the ModelNet40 dataset for object classification, Stanford Indoor Data Scenes (S3DIS) for semantic segmentation and ShapeNet dataset for part segmentation showed improved performance of the model in most cases. Further, the object classification network was generalized on the ScanObjectNN dataset to emphasize the effect on model performance on real-life scans compared to Computer-Aided Design (CAD) generated point clouds. Part segmentation network was also generalized on the ShapeNet-C dataset to gain insight into the model's robustness to various noises. An ablation study on the PointNet part segmentation network shows that the modular network used in the PointNet architecture for input pose normalization has an insignificant effect on the model performance.

Contents

Chapter 1

Introduction

Computer vision (CV) is currently one of the largest areas of research in artificial intelligence, which aims to efficiently perform some of the tasks that the human optical system can do. It has applications in various fields, from civil engineering [1], autonomous driving [2], perception [3], industrial engineering [4] to medical technology [5] and many others. The field of computer vision captures, analyses and processes 2D data such as images and 3D data such as point clouds, meshes, etc. to perform various vision tasks such as object classification, segmentation, recognition, etc.

Point cloud has become one of the prevalent geometric data structures to represent 3D shapes, objects, real landscapes, and so on. Its popularity is based on its increasing simplicity and quick accessibility through low-cost solutions. These are sets of data points distributed in space. Each point in the point cloud is described by its x, y, z coordinates and some other features depending upon the application.

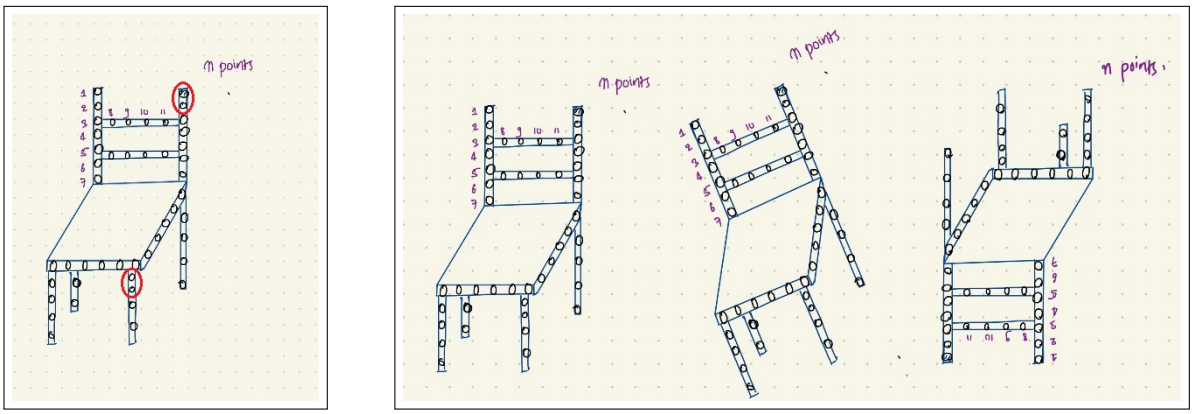
Although there are several Deep Learning architectures that can handle 3D data such as point clouds, meshes etc., typical CNN-based architectures require the input data to be in highly regular formats such as image grids or 3D voxels for weight sharing and other kernel optimisations. Transformation into a regular data format is computationally intensive, time-consuming and introduces quantisation artefacts that can obscure the natural invariance of the data.

PointNet [6] takes in the point cloud data directly and applies Deep Learning to perform object classification and part/semantic segmentation. The revolutionary change that PointNet brought was the elimination of the requirements for intermediate representation of the input data. PointNet architecture consists of two networks, one for object classification and another for part/semantic segmentation. The classification network takes the point cloud and outputs the K-score for the candidate K-classes, and the part/semantics segmentation assigns part/semantics labels to each of the points in the input point cloud on a point-by-point basis. Point clouds are essentially a set of points; therefore, the PointNet network must meet some of the requirements listed below:

1. Permutation invariance: The points in the point cloud must be stored in order to perform a meaningful operation. Storing points means ordering these points, which does not really contain useful information. The order does not have much importance as the object's shape would not be affected by the order of the stored points. The network must learn regardless of the order of the input points. Therefore, the network must be invariant to the permutation of the points in the input point cloud.

2. Point interactions: Points interact with each other in local neighbourhoods, and their interaction contains important information. Although the classification network works

with global features, only the segmentation network can benefit from local and global point features. As shown in Fig. 1a, the information conveyed by two circled points on the back-rest of the chair is not the same as the two points on the chair leg , and this information is useful for part segmentation or semantic segmentation of the input point cloud.



(a) (b)

Figure 1.1: (a) Interaction among the points in Point Cloud carries essential information for the segmentation task i.e. the interaction of encircled points on the back-rest of the chair will contain different information as compared to the encircled points on the chair leg which would be essential for the segmentation task, (b) Spatial Transformations of Point Cloud should not effect the result of the network

3. Transformation Invariance: The objects in the natural landscape can be positioned in different orientations. Therefore, the network should be able to perform the classification and segmentation task well even when the point clouds are transformed (rotated, translated, etc.). As shown in Fig. 1b, the input point cloud representing a chair could be spatially transformed. However, the classification result should be the same for all three instances shown and the segmentation result should also be unchanged for all transformed instances of the same object.

Having understood the PointNet architecture, I re-implemented it using PyTorch and present an in-depth analysis of the architecture. First, the classification and part/semantic segmentation networks were implemented. These networks were then experimented with different configuration settings to find the best performing model configuration for each task. In addition, I generalised the classification and partial segmentation networks to other datasets to gain deeper insights into the PointNet architecture. I also conducted an ablation study for the part segmentation network to investigate the contribution of the input transform T-Net on the model performance.

The rest of the report is organised as follows: Chapter 2 provides the background knowledge needed to understand the project, some work related to this project has been discussed in Chapter 3, Chapter 4 discussed the methodology used in carrying out this project, and finally the experimental results and conclusions were presented in Chapter 5 and Chapter 6, respectively.

Chapter 2

Background

2.1 3D Object classification and Segmentation

Classification and segmentation of 3D objects is one of the fundamental and challenging tasks in computer vision and graphics. The following is a brief summary of point-based 3D object classification and segmentation:

2.1.1 3D Object Classification

3D object classification is the task of identifying the objects by classifying them into one of the finite sets of classes. As shown in Fig. 2.1, classification generally involves comparing the measured characteristics of a new object with the already known object and determining whether the new object belongs to that particular object class.

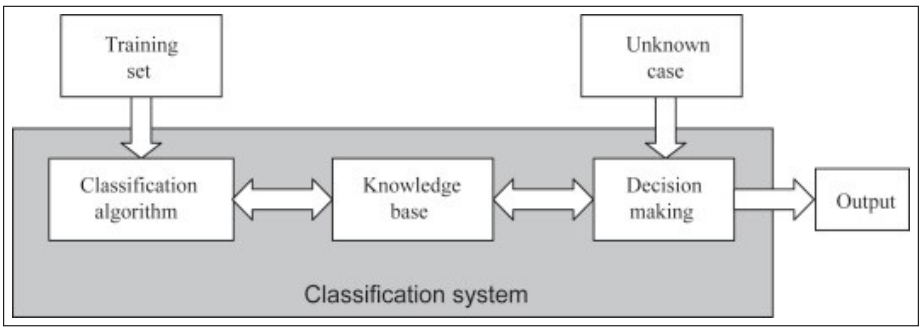


Figure 2.1: General configuration of classification network

2.1.2 3D Segmentation

3D segmentation aims to predict point-wise class labels in a 3D instance or scene. 3D segmentation can be broadly categorized into three types: semantic segmentation, instance segmentation and part segmentation. The goal of semantic segmentation is to predict pointwise object class assignments in a 3D scene, e.g. chair, table, bed, etc. Instance segmentation further seeks to distinguish between different instances of the same class label, e.g. chair one, chair two, etc., and partial segmentation aims to make a further subdivision indicating the components of a given instance, e.g. legs or backrest of a chair, etc.

Deep Learning methods have proven to be quite successful in accomplishing the above tasks. All these Deep Learning methods can be classified into five categories based on 3D data rep-

resentation, namely RGB-D image-based, voxel-based, projected image-based, point-based and based on other possible representations [7]. The point-based approach is briefly discussed here.

Point-based 3D object classification and segmentation: These point-based Deep Learning approaches for 3D object classification and segmentation can be divided into three broad categories:

1. Multilayer Perceptron (MLP) based method: In this method, the MLP is applied directly to the points to learn features.
2. Point convolution based methods: In this method, the convolution operation is applied directly to the points.
3. Graph convolution based methods: In this method, the convolution operation is applied to the points associated with a graph structure.

2.2 Overview of PointNet Classification Network

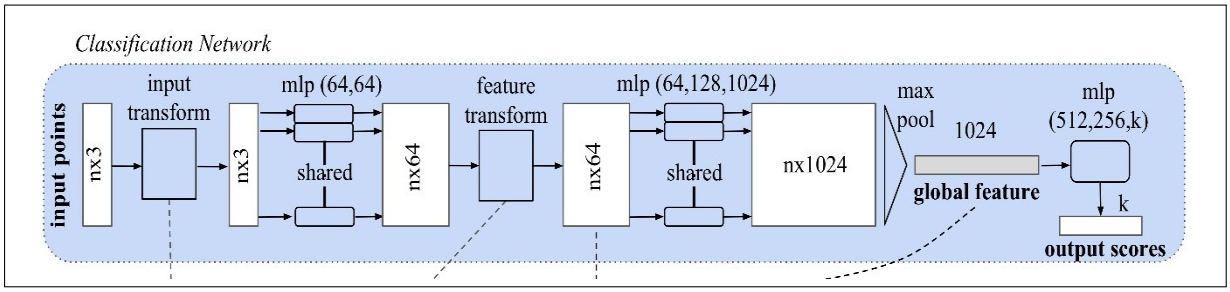


Figure 2.2: PointNet Object Classification network: It takes in the input point cloud and outputs the K-classification scores for K candidate object classes representing different shapes.

Point clouds are created using 3D laser scanners and LiDAR (light detection and ranging) sensors and various reconstruction algorithms. Each point in the point cloud can be represented by its x, y, z coordinates and various other features. Fig.2.2 shows the structure of PointNet [6] classification network. In the PointNet classification network, the point cloud is first stored in an n×3 matrix (for simplicity, only the x, y and z coordinates are considered features). The input passes through a modular network called T-Net, which regresses an input-dependent 3×3 transformation matrix for pose normalisation of the input point cloud. This transformation does not change the features of the points, i.e. the points undergo transformation such as rotation, scaling, etc., but the object shape remains the same. This modular network is called an input transform. The concept of pose normalisation is extended to a 64-dimensional embedding space called the feature transform. This network works in the same way as the input transform, but in a higher dimension. It regresses an input-dependent 64×64 transformation matrix for pose normalisation further down the network. Each point in the point cloud can be considered as a vector of dimension 3. All these output vectors of the input transform are then passed through a shared Multi-layer perceptron (MLP) network where the dimensionality of each vector is increased to 64. These output vectors are called feature vectors. There will be as many feature vectors as the number of points in the input point cloud. Each feature vector contains specific information about a point in the point cloud. Then these vectors go through a feature transform network. The output of the feature transform goes through another shared MLP network where the dimensionality is increased to 1024. In this crucial part of the network, the output of the last MLP goes through a maxpooling operation, resulting in a vector of dimension 1024 representing the entire point cloud. The features of this vector are called the Global Feature, and the vector

is called global feature vector [1]. The network gains its permutation invariance to the input because of this Maxpool operation at the end. These Global Feature Vectors are assumed to contain meaningful information required to classify the objects. At the end, the Global Feature Vector is passed through another MLP network that outputs K classification scores for K candidate classes.

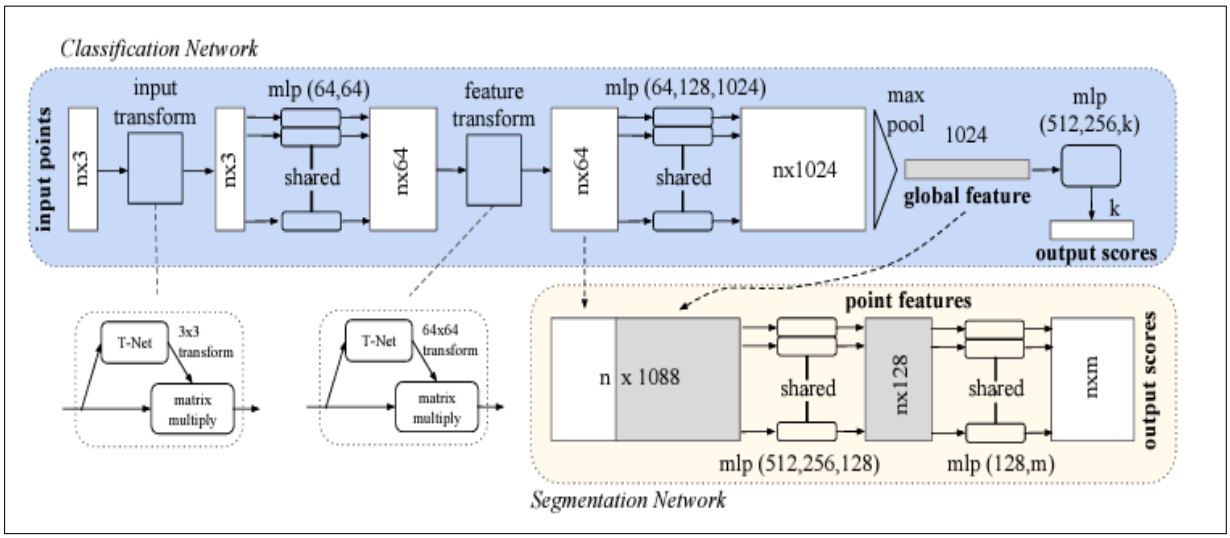


Figure 2.3: PointNet Semnatic Segmentaion network: It is an extension of the PointNet object classification network. The network takes in the input point cloud and outputs the per-point part/semantic segmentation label in the input point cloud

As shown in Fig. 2.3, the PointNet segmentation network is an extension of the PointNet classification network. Here, the feature vector of dimension 64 from the first part of the PointNet classification network is concatenated with the global feature vector to combine local (point-specific) and global (the entire point cloud) information. This concatenation results in n-number of vectors of dimension 1088. These vectors are passed through a shared MLP network where the dimensionality is reduced to 128. The output of this MLP is again fed into another MLP network where the dimensionality is further reduced to m, where m is the total number of parts of the objects or the number of semantic classes in the scenes.

2.3 Max-pooling

Max-pool is a mathematical operation that takes a discrete sample-based input and gives the maximum entry as output. It is used primarily in convolutional neural networks to capture the strongest input feature values between layers of a neural network. It is also used to downsample an input representation, reducing computational costs by decreasing the number of parameters [8].

Let's assume a vector input of dimension n with components a_i, where $i \in [0,\ n-1]$, then

Max-pool $(a_0, a_1, \ldots\ldots, a_n - 1) = a_p$, *where* a_p *is the greatest value among all entries*

It is a symmetric function, so any change in order of input doesnot affect the output of this operation.

Max-pool (a, b) = *Max-pool* (b, a) *where* a *and* b *are the inputs to the functions*

Furthermore, the operation can be easily extended to multi-dimensional inputs as well.

2.4 Spatial Transformer Network (STN)

Max et. al [9] have proposed a modular neural network that can be dropped into a convolutional neural network at any point and in any number into a foldable neural network to achieve pose normalisation of its input.

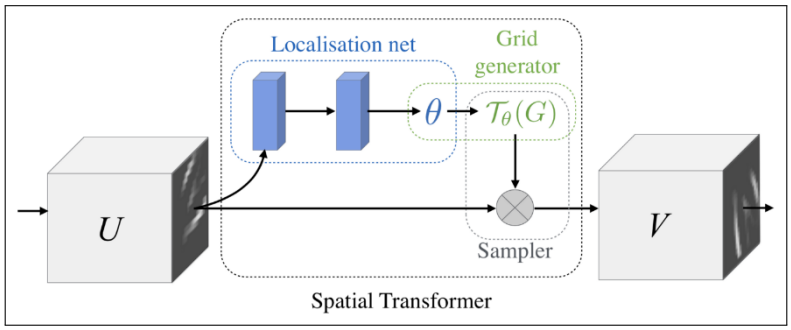


Figure 2.4: Spatial Transformer Network: The input feature map(U) goes through the Localisation net (colored in blue), Grid generator and a sampler to regress the transformation θ matrix.

As shown in Fig.2.4, the spatial transformer network consists of a localisation network, a grid generator and a sampler. First, the localisation network takes the input feature map U and predicts the transformation parameters θ. Then the grid generator uses the predicted transformation parameters to create a sample grid. Sample grid is set of points at which the input map is sampled to produce the transformed output. Finally, the sampler takes the feature map and the sampling grid as input and generates the output map, which is sampled from the input at the grid points.

2.5 Evaluation Metrics

Evaluation metrics establish the qualitative performance of methods and thus help to compare the performance of different methods. Some of the widely used evaluation metrics for classification of 3D objects, part and semantic segmentation are described below:

2.5.1 Overall Accuracy (OAcc)

It is the ratio of the truly classified samples to the total number of samples. The mathematical relationship is given by:

$$OAcc = \sum_{i=0}^{n} \frac{x_{ii}}{\sum_{j=0}^{n} x_{ij}} \tag{2.1}$$

2.5.2 Mean Intersection over Union (mIOU)

It is a widely used metric for segmentation tasks. It is calculated by the intersection ratio of ground truth and the predicted value, averaged over the total number of classes N. The mathematical relationship is given by:

$$mIOU = \frac{1}{N+1} \sum_{i=0}^{N} \frac{x_{ii}}{\sum_{j=0}^{N} x_{ij} + \sum_{i=0}^{N} (x_{ji} - x_{ii})} \tag{2.2}$$

2.5.3 Category Mean Intersection over Union (Category mIOU)

It is used in case of 3D part segmentation tasks. Suppose that $M_J, J \in [0,\ LI]$ are parts in each object instance and q_{ij} is the total number of points belonging to part i but derived from part j and q_{ii} stands for the true positives and q_{ji} and q_{ij} for the false negatives and false positives, respectively. Then the category Mean Intersection over Union over N classes is given by:

$$\text{CateogrymIOU } = \frac{1}{N+1} \sum_{I=0}^{N} \sum_{J=0}^{L_I} \sum_{i=0}^{M_J} \frac{q_{ii}}{\sum_{j=0}^{M_j} q_{ij} + \sum_{i=0}^{M_j} (q_{ji} - q_{ii})} \tag{2.3}$$

Chapter 3

Related Works

Earlier, features of point clouds were handcrafted for specific tasks [10][11]. Point features contain specific information about the points and are called local features, while the features that contain information about the entire point cloud are called global features [12][13]. It was very difficult to find an optimal combination of features as this was done manually [14].

3D data can be presented in different representations, which motivates the development of different methods for learning from 3D data based on different representations [15].

Volumetric convolutional neural networks: This type of network applies 3D convolutional neural networks to voxel-based shapes [16][17]. However, these representations are limited by its data sparsity and the high cost of computation for 3D convolution.

Vote3d [18] and FPNN [19] have presented some specific methods to solve the problem of data sparsity, however it is still difficult for these networks to process large point cloud data. Muzahid et.al [20] has proposed an economic 3D Volumetric CNN that prepares the input data by applying a memory-effective octree representation. It uses joint multi-scale hierarical and sub-volume supervised learning for classification of 3D objects.

Various methods such as Multiview Convolutional Neural Networks first attempted to convert the 3D point clouds into 2D images and then apply 2D convolutions to them to perform tasks such as shape classification and retrieval [21], and these also became quite popular due to the performance dominance of 2D convolutions. H. Su et al[22] presented a Multiview CNN approach where they made use of multiple views to create a 2D image descriptor for each view, and then processed all of these image descriptors using a CNN architecture which included a view pooling layer to combine the information from the different views. This CNN architecture creates a single dense descriptor that represents the entire 3D shape and was used for tasks such as shape recognition, etc. However, these methods could not be extended well for other tasks such as 3D scene understanding, shape completion and point classification.

Few other methods apply Spectral Convolutional Neural Network [23] to 3D meshes for learning process. Methods like Feature based Deep neural networks [24] first generate a vector from the 3D data representing the features of the input 3D data and this vector is then used for classification tasks by passing it through fully connected layers.

Point clouds are basically an unordered set of vectors. While all previous work in deep learning on 3D data was mainly based on regular 3D input representations, PointNet [6] serves as a pioneer work that ingests 3D point clouds directly to perform 3D objection classification, semantic segmentation and partial segmentation in an end-end fashion. However, it reaches its limits in exploring the geometric context of points when learning representations.

Since then, various PointNet-based methods have been proposed, such as PointNet++ [25], which solves the aforementioned limitation of PointNet by applying the max-pool operation hierarchically in the local regions. However, it should be noted that both of these methods rely on the max-pooling operation to collect context information without using convolution.

SO-Net proposed by Jiaxin et al [26], represents the spatial distribution of the input point cloud in self-organising maps and performs feature extraction at the nodes of the self-organising map and individual points hierarchically to produce a single feature vector using PointNet.

Recently, the demand for processing irregular data with convolutional network-like architectures has led to an upsurge in graph-based convolutional networks [27]. In general, in addition to convolutional networks, techniques that recurrently update vertex features to relay contextual information have been developed in the context of graph-based deep learning [28][29]. Graph CNNs can be further divided into spatial [30] and spectral [23] networks. The spatial networks apply convolution directly to the spatial vertices, while the spectral networks apply convolution to the spectral vertex signal generated from the Fourier transforms.A major bottleneck in using the Spectral Network is that it requires a fixed graph structure, which makes it difficult to apply to data with varying graph structures (e.g. point clouds).

ECC proposed by Martin et al [31] is an outstanding work for the analysis using spatial graph convolution. It generates dynamic convolution filters between connected vertices by adapting the MLPs. However, dynamic filter generation significantly increases the computational cost.

Datasets play a crucial role in training and testing Deep Learning models. Continuous work has been done in the field of creating benchmark datasets for computer vision tasks. The PointNet proposed by Charles et al [6] uses the ModelNet40 dataset [17] for object classification, the ShapeNet dataset [32] for the part segmentation and Stanford 3D Indoor Scenes [33] for semantic segmentation. ModelNet40 is a comprehensive, clean collection of 3D models CAD for objects and ShapeNet is a comprehensive repository of 3D models CAD with 16,881 shapes from 16 categories and a total of 50 part labels. Since these datasets are based on clean 3D CAD models of objects, they are free from disturbances such as object clutter, occlusions, brokeness etc., which are highly likely to be present in point clouds generated from real scans. The distortion of point clouds is inevitable due to the inaccuracy of processing, inaccuracy of sensors, complexity of scenes, etc. ScanObjectNN, proposed by M. A. Uy et al. [34] has created a benchmarking dataset for classifying 3D objects on a real world dataset. Therefore, ScanObjectNN was used in this project to investigate the robustness of the model performance to the above perturbations. Similarly, ShapeNet-C, proposed by A. X. Chang et al [35], is a test suite for the robustness of point cloud data in the presence of distortions. It consists of point cloud instances annotated with part labels for each point with seven different types of corruptions such as jitter, rotation, scaling, local and global dropping of points with five severity levels each.

In this project, however, the focus is on the PointNet architecture, its re-implementation and its in-depth analysis.

Chapter 4

Methodology

The invention of PointNet architecture by Charles et. al [6] has proved to be a pioneering work in the field of computer vision to perform 3D object classification and segmentation directly on point cloud data. I implemented the PointNet network and performed an in-depth analysis of the network to improve the performance of the model. In addition, the generalisation of the classification and segmentation network to different datasets was investigated to determine the impact of noise, real scans, etc. on model performance. An ablation study was also conducted to show the effect of spatial transformations of the input data on model performance.

The project was conducted in three different segments corresponding to the tasks, namely 3D object classification (see section 5.1), 3D semantic segmentation (see section 5.2) and 3D object part segmentation (see section 5.3). All Deep Learning models depend heavily on the datasets used to train and test the model. Therefore, a selection of suitable datasets was made before implementing the network in the project.

4.1 Datasets

Datasets play a critical role in training and testing deep learning models. However, collecting and annotating datasets privately is very tedious and requires a high level of expertise and high-quality equipment. Therefore, building on public datasets proves to be an ideal way to reduce costs. The following datasets were used in this project for training and testing models:

4.1.1 3D Object Classification

ModelNet40

It is a comprehensive, clean collection of 3D CAD models for objects. It contains 12,311 pre-adjusted shapes from 40 categories [17]. The split between training and testing is 80%-20% with a split of 9843 objects for training and 2468 objects for testing. This dataset was used to create a comparative study to the study conducted by Charles et. al [6].

ScanObjectNN

It is a benchmarking dataset for 3D object classification on a real-world dataset comprising 2902 objects in 15 categories [34]. Various Deep Learning models have shown improved performance in classifying 3D objects over the years, which raises the question whether the problem of classifying 3D objects is almost solved. It has been noted that most of these models have shown such high performance on datasets with CAD models of the objects. However, real-life scans often have noises such as object clutter, occlusions, brokenness and backgrounds, etc., which

make the low-level geometry look quite different and learning more difficult. This real-world dataset was used to investigate the robustness of the model performance to the above mentioned noises.

4.1.2 3D Semantic Segmentation

Stanford 3D Indoor Scene Dataset (S3DIS)

The dataset contains six large-scale indoor scenes with 271 rooms. Each point in the scene point cloud data is annotated with one of the thirteen semantic categories [33]. This dataset was used to create a comparative study to the one conducted by Charles et. al [6].

4.1.3 3D Part Segmentation

Shapenet

This is an extensive repository of 3D CAD models containing 16,881 shapes from 16 categories, with a total of 50 part labels [32]. This dataset was used to conduct a comparative study with the PointNet architecture proposed by Charles et. al [6].

Shapenet-C

This dataset is a test suite for the robustness of point cloud data in the presence of corruption. Point cloud corruption is inevitable due to imprecision in processing, sensor inaccuracy, complexity of scenes etc. It consists of point cloud instances annotated with part labels for each point with seven different types of corruptions, each with five levels of severity [35],. The types of corruptions are:

1. Jitter : Gaussian noise $\epsilon \in N(0,\ \sigma^2)$ was added to each of the point's coordinates, where $\sigma \in \{0.01,\ 0.02,\ 0.03,\ 0.04,\ 0.05\}$ for the five levels

2. Scale: Scaling was applied randomly to each of the axes. The coefficient of scaling was independently sampled as $K \sim U(1/K,\ K)$, where $K \in \{1.6,\ 1.7,\ 1.8,\ 1.9,\ 2.0\}$ for the five levels. Point clouds are re-normalized to a unit sphere after the scaling operation.

3. Rotate: Random rotation was applied described x, y, z Euler angles $(\alpha,\ \beta,\ \gamma)$, such that $\alpha,\ \beta,\ \gamma\ \sim U(-\theta,\ \theta)$ and $\theta\ \in \{\pi/30,\ \pi/15,\ \pi/10,\ \pi/7.5,\ \pi/6\}$ for the five levels.

4. Drop Global: All the points were randomly shuffled and the last N* ρ points were dropped, where N = 1024 and $\rho\ \in \{0.25,\ 0.375,\ 0.5,\ 0.675,\ 0.75\}$ for all five levels.

5. Drop Local: K points in total were dropped where $K \in \{100,\ 200,\ 300,\ 400,\ 500\}$ for the five levels. Points are randomly shuffled to select the number of local parts to be dropped $C \in U\{1,\ 8\}$. Then, a cluster size N_i was assigned to i-th local part. The following steps are repeated C times: Selecting a point as i-th local centre randomly and then dropping its N_i-nearest neighbour points from the point cloud.

6. Add Global: K points are sampled uniformly inside a unit sphere and are added to the point cloud, where K $\in \{10,\ 20,\ 30,\ 40,\ 50\}$ for the five levels.

7. Add Local : K points in total were added where K $\in \{100,\ 200,\ 300,\ 400,\ 500\}$ for the five levels. Points are randomly shuffled to select the first $C \in U(1,\ 8)$ points as the local centres. Then, a cluster size N_i was assigned to i-th local part. Neighbouring point's coordinates were generated from a Normal Distribution$N(\mu_i,\ \sigma_i^2 I)$, where μ_i is the i-th local centre's X-Y-Z coordinate and $\sigma_i \in U(0.075,\ 0.125)$. Then each local part was added to the point cloud one after another.

This dataset was used to illustrate the impact of corruptions on the performance of the model.

Evaluation metrics can explain the predominance of one method over others. Overall Accuracy was used for 3D object classification and Mean Intersection over Union (mIOU) for 3D semantic segmentation and part segmentation as evaluation metrics in this project. Category Mean Intersection over Union (Category mIOU) was also used to demonstrate class-wise performance in part segmentation. All these evaluation metrics were described in detail in section 2.5.

All three sections of this project follow a similar structure, starting with implementing each network, experimenting with the network to find the best performing configurations, conducting an in-depth study to shed light on other aspects of the network, and finally generalising the network to other datasets. However, each section is described in detail below:

4.2 3D Object Classification

First, an attempt was made to replicate the exact configuration of the PointNet architecture proposed by Charles et.al [6]. This configuration was to be used as the base configuration for the PointNet architecture. The implementation of the network involved a separate modular network creation for different functional blocks of the PointNet. The input transform and feature transform blocks were implemented through their respective networks, namely STN3d and STNkd, where 'k' represents the dimensionality of the feature transform. Furthermore, the entire 3D object classification network was implemented by two separate networks, namely PointNetFeature and PointNetClassifier, where the PointNetFeature network takes in the input point cloud and outputs the global feature vectors for the object classification task or the array of vectors formed by concatenating feature vectors with the global feature vector in the case of the segmentation task and PointNetClassifier takes in the global feature vector as input and gives out the prediction on object classes. The object classification network was trained and tested with the ModelNet40 dataset. The optimiser Adam [36] was used with fixed initial hyperparameters such as learning rate(l_r) = 0.001 and betas(β_1, β_2) = (0.9, 0.999). However, to further smooth the optimisation process, a learning rate scheduler was implemented with learning rate decay period (step_size) = 20 and learning rate decay multiplicative factor (γ) = 0.5.

The dropout technique has proven to be an effective way to improve the generalisation capability of neural networks. The baseline configuration already had a dropout layer (p = 0.3) in the PointNetClassifier network. In this project, several other configurations of PointNet have been proposed, e.g. configurations with varying dropout probability, varying positioning of dropout layers in the network, introducing multiple dropout layers in different parts of the network, etc., in the hope of improving the generalisation capability of the PointNet object classification network.

Moreover, Deep Neural Networks are very sensitive to the initialisation of weights [37][38]. Improper initialisation of the weights can lead to exploding or vanishing outputs in the deeper parts of the network, which in turn significantly affects the training of the network. However, batch normalisation layers are available in the base configuration, which solve the problem of vanishing and exploding outputs very well and smoothen the training process of the network.

Batch normalisation and weight initialisation, although completely different techniques, have a similar effect in keeping the weights of the layers within a certain range. Since both techniques have a similar effect on training the network, various weight initialisation techniques such as the weight initialisation technique proposed by Xavier et al, the weight initialisation technique proposed by Kaiming He et al.[39], etc, have been implemented in the hope of achieving small improvements in model performance. In addition, the batch normalisation layers were omitted

to test the sole effect of the weight initialisation technique on the performance of the network.

Although many studies[40][41] suggest the usefulness of batch normalisation and dropout in neural network training, there is still much debate about the concurrent use of these two techniques and their ordering in the network when used simultaneously. It has been shown that different configurations of use formed by combining these two techniques often lead to different performance results. There is also evidence in the literature that the simultaneous use of both techniques leads to a worse result [42]. The reason for this may be that dropout shifts the variance of a given neural layer as the state of the network changes from training to testing, whereas batch normalisation retains the statistical variance accumulated throughout the learning process in the testing phase. This leads to inconsistency of variances in both techniques, resulting in unstable numerical behaviour during inference and thus inaccurate predictions.

An interesting study by Guangyong et. al [43] proposed a novel training procedure combining dropout and batch normalisation (collectively referred to as Independent Component (IC) layer) before each weighting layer to make the network inputs more independent and achieve fast convergence. Therefore, this project also explored a different configuration of PointNet by combining these two techniques in the way proposed above..

Furthermore, changing the dimensions of the layers in the neural network significantly affects the flow of information through the network, which in turn affects the performance of the model [44]. Since it was assumed that the global feature vector contains all the essential information required for object classification decision making, the performance of the model was investigated by changing the dimensions of the global feature vector layer and some other layers in the network [6].

The same dropout probability in different layers of the network has a different impact on the generalisation ability of the model. Therefore, in this project, another strategy of using multiple dropout layers with reducing the probability of retaining the neural unit 'p' while going deeper into the network was implemented in the hope of improving the generalisation capability of the network.

Once the best performing configuration was found, the network was generalised to the ScanObjectNN dataset to investigate the impact of real scans, which may contain various other noises like addition of background, object clutter, occlusion , brokenness etc.compared to a point cloud generated from the clean CAD models on model performance.

4.3 3D Semantic Segmentation

The PointNet 3D Semantic Segmentation Network is basically an extension of the 3D Object Classification Network discussed earlier. In this project, a similar approach as in case of 3D Object classification network was used to perform the full study of the 3D semantic segmentation network. The implementation of the 3D semantic segmentation network uses both the already defined modular networks, namely PointNetFeature, and a newly defined PointNetSemanticSegmenter network which takes in the array of concatenated feature vectors and the global feature vector as input and outputs the per-point semantic segmentation labels as output. First, an attempt was made to replicate the exact implementation of the PointNet 3D semantic segmentation network by Charles et. al [6]. This configuration was to be used as the base configuration for the PointNet Semantic Segmentation network architecture. The Stanford 3D Indoor Scene Dataset (S3DIS) was used to train and test the network. Same optimiser setup as in case 3D object classification was used for training. Dropout and weight initialisation schemes were not part of the base configuration of the network. A lot of learning acquired during the study of the object classification network have been used in this segment of the project to avoid unnec-

essary manual work. Since the first part of the 3D semantic segmentation network is identical to the object classification network, some of the configuration settings of the network, such as the introduction and positioning of dropouts in the network, ordering of batch normalisation and dropout layers, etc., were directly adapted to the best performing configurations in the 3D object classification case.

The impact of weight initialisation schemes was discussed earlier in this project. So different weight initialisation schemes were tried in the hope of improving the performance of the model. In addition,configuration with reducing the probability of retaining the neural unit 'p' while going deeper into the network with appropriate dropout probabilities were tested out to find the best performing configuration for the 3D semantic segmentation task.

4.4 3D Part Segmentation

The PointNet 3D part segmentation network is essentially the same network as the 3D semantic segmentation with changes to the final Softmax layer. A new Softmax layer has been introduced to the network to output probabilities for classes, where classes represent the parts of objects. The best performing model configuration from the 3D Semantic Segmentation task with the above mentioned changes to the Softmax layer was used for the 3D Part Segmentation task. The model was trained and tested with the Shapenet dataset.

As described in Section 4.1.3, the Shapnet-C dataset consists of similar object classes to the Shapenet dataset, but here there are object instances to which certain types of noise have been added. The model trained with the Shapenet dataset was used directly to study the effect of noise such as jitter, rotation, dropping off points locally and globally and scaling on model performance. In addition, an ablation study was performed for the part segmentation network by omitting the modular STN3D network to investigate the robustness of the model performance to noises such as rotation, jitter and scaling.

New models were created to handle object instances with noise, like added noisy points locally and globally in the point cloud, consisting of an additional class representing the global and local noisy points by using transfer learning. A suitable network configuration was determined by training and testing the model by varying the number of trainable layers in the pre-trained model to investigate the effects of adding global and local noisy points to the point cloud.

Chapter 5

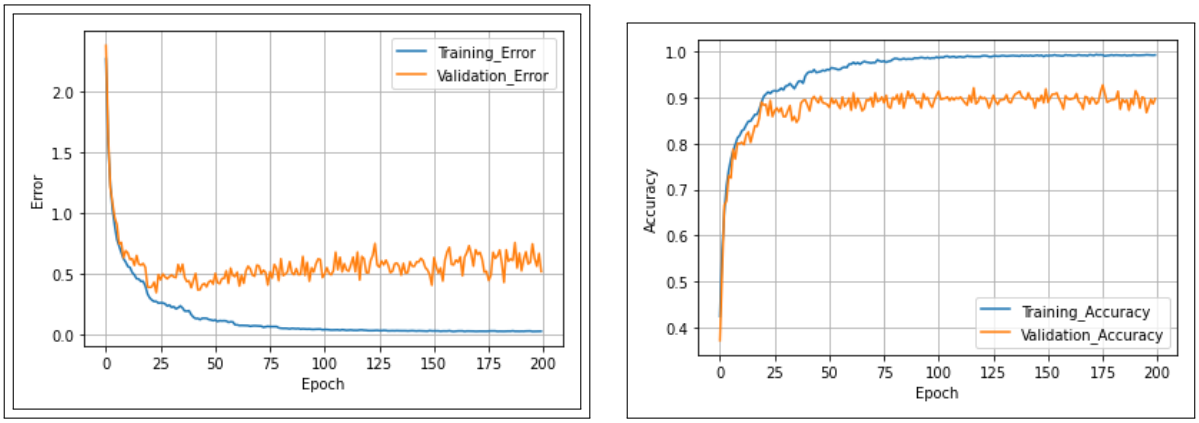
Experimentation

A summary of the quantitative results of 3D object classification, 3D Semantic Segmentation and 3D part segmentation has been presented below:

5.1 Results of 3D object classification

Trial 1: Random weight initialisation, Dropout(p=0.3) in PointNetClassifier network and Batch normalisation layer after every convolution and linear layers

First, the implemented network was trained with a similar network configuration as proposed by Charles et. al [6], i.e. with random weight initialisation and with a fixed dropout of p=0.3 in the PointNetClassifier network on the ModelNet-40 dataset. Batch normalisation was applied after each linear and convolutional layer in the network. The performance with this base configuration would serve as a baseline result for the further experiments aimed at improving the performance of the network.



(a) (b)

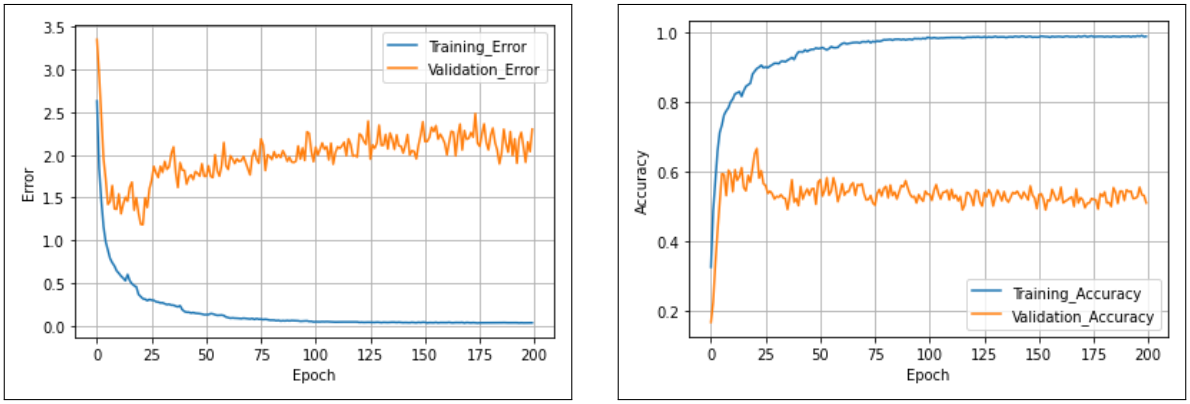
Figure 5.1: (a) Trial 1: Training and Validation loss, (b) Trial 1: Training and Validation Accuracy. Training and Validation losses show a decreasing trend and the respective accuracies shows an increasing trend over the coarse of training.

In Fig. 5.1, the decreasing trend of training and validation loss and the increasing trend of accuracy shows that the model learns to approximate the function required to perform 3D object classification. The training and validation loss and accuracy show that the model performs well

on the training data but not so well on the validation data, i.e. the generalisation of the model is not good.

Trial 2: Random weight initialisation, Dropout(p=0.3) in PointNetClassifier network, Dropout (p=0.2) in PointFeature network and Batch normalisation layer after every convolution and linear layers

The dropout technique has proven to be an effective way to improve the generalisation capability of neural networks. Therefore, an additional dropout of p= 0.2 was introduced in the PointNetFeature network of the PointNet base configuration. However, this configuration resulted in poorer model performance, as can be seen in Fig. 5.2. Therefore, other variants with dropouts were implemented.

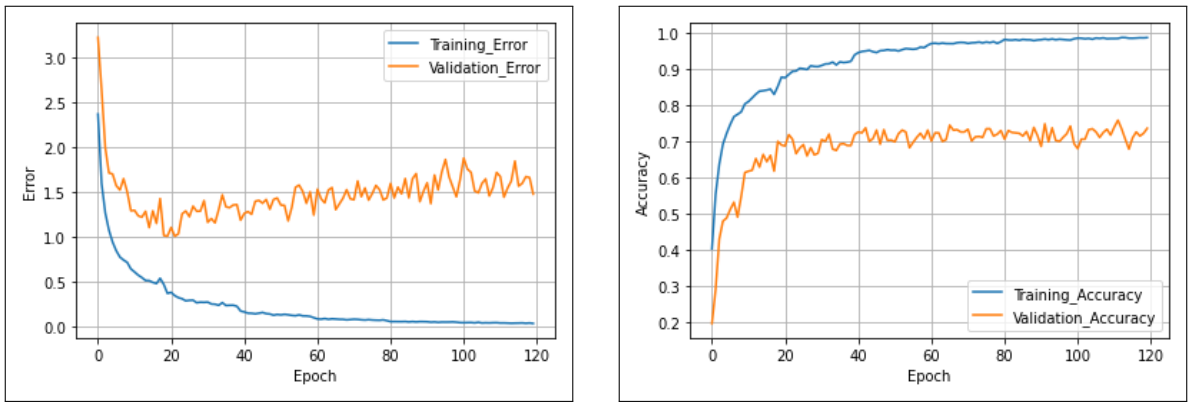


(a) (b)

Figure 5.2: (a) Trial 2: Training and Validation loss, (b) Trial 2: Training and Validation Accuracy. An additional dropout (p=0.2) in the PointNetFeature network results in higher training and validation losses and lower accuracies as compared to previous configuration.

Trial 3: Random weight initialisation, Dropout (p=0.2) in PointFeature network and Batch normalisation layer after every convolution and linear layers

The dropout layers in the previous configurations were replaced by a dropout layer of p=0.3 only in the PointNetFeature network. This resulted in even worse model performance compared to the previous configuration, as can be seen in Fig. 5.3. Furthermore, the number of training epochs was reduced to 120, as the training loss in all previous experiments reached a plateau after 120 epochs, which basically means that the network learns little or nothing after this epoch.



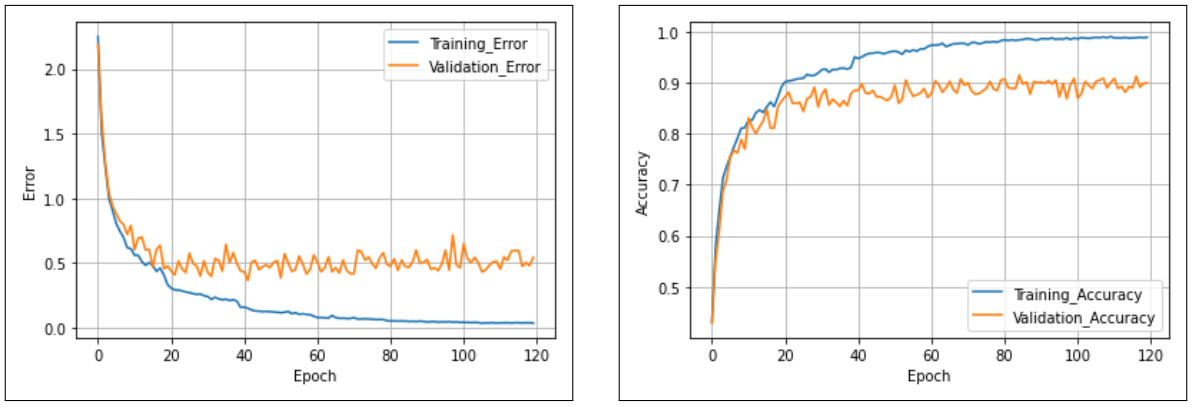
(a) (b)

Figure 5.3: (a) Trial 3: Training and Validation loss, (b) Trial 3: Training and Validation Accuracy. The number of training epochs has been reduced to 120 in this network configuration. The usage of dropout in PointNetFeature network results in higher training and validation losses and lower accuracies respectively.

This confirms the fact that dropouts can lead to poorer model performance when applied to convolutional layers compared to fully-connected layers. One possible explanation for the lower model performance could be that convolutional layers generate a much smaller number of parameters compared to fully-connected layers, so that when a certain number of neural nodes in a convolutional layer are dropped out, essential information is more likely to be lost than in fully-connected layers. Since the network configuration with dropout p=0.3 in the PointNetClassifier gave the best performance among the previous experiments, this configuration was used for the further experiments.

Trial 4: Random weight initialisation, Dropout(p=0.2) in PointNetClassifier network and Batch normalisation layer after every convolution and linear layers

Another variant of the network was implemented by varying the dropout probability in the PointNetClassifier network to 0.2. The model performed moderately worse than the previous configurations as shown in Fig.5.4.

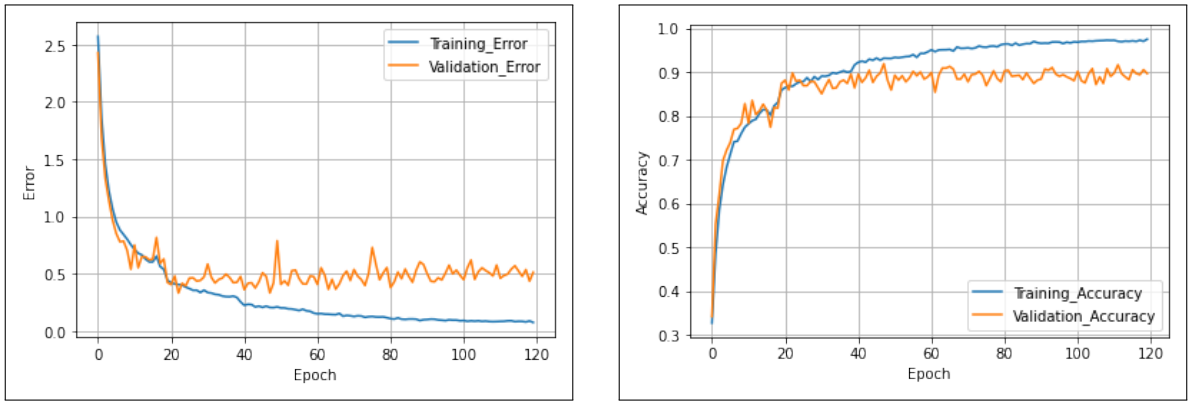


(a) (b)

Figure 5.4: (a) Trial 4: Training and Validation loss, (b) Trial 4: Training and Validation Accuracy. Dropout in PointNetClassifier network results in lower training and validation losses hence higher accuracies

Trial 5: Random weight initialisation, Dropout(p=0.4) in PointNetClassifier network and Batch normalisation layer after every convolution and linear layers

Another network configuration, in which the dropout probability in the PointNetClassifier network was increased to 0.4, was tried out to determine the optimal dropout probability for the 3D object classification network. However this configuration of the network performed inferior as compared to the previous configurations, as shown in Fig.5.5.



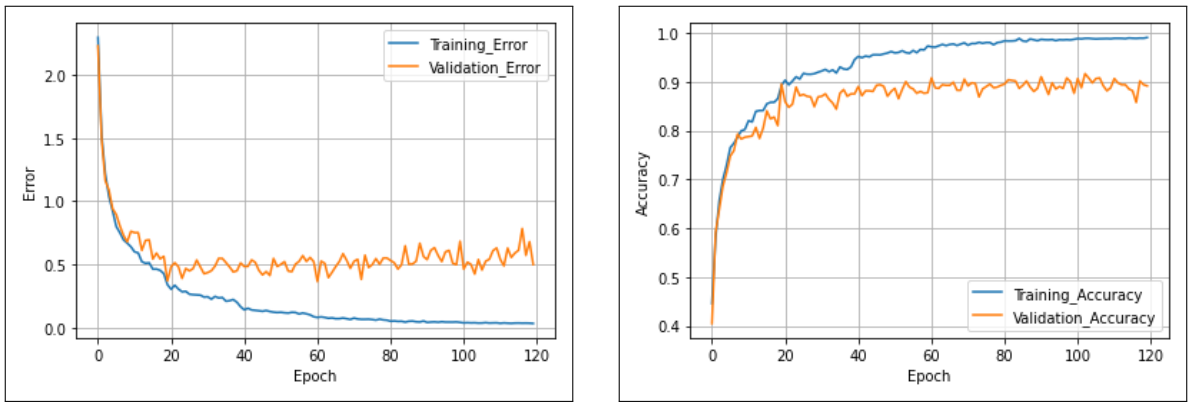
(a) (b)

Figure 5.5: (a) Trial 5: Training and Validation loss, (b) Trial 5: Training and Validation Accuracy. Dropout(p=0.5) in the PointNetClassifier network shows higher training and validation losses and lower accuracies as compared to the network configuration with dropout(p=0.3) in the PointNetClassifier network

Thus, it was found that the optimal positioning of the dropout layer and its dropout probability in the PointNet object classifier network was the PointNetClassifier network with a dropout probability of p= 0.3.

Trial 6: Kaiming Uniform weight initialisation, Dropout(p=0.3) in PointNetClassifier network and Batch normalisation layer after every convolution and linear layers

The performance of the model is very sensitive to weight initialisation techniques, as already explained in section 4.2. The Kaiming uniform weight initialisation technique was implemented for the model with the best performance obtained from the previous experiments in this project.



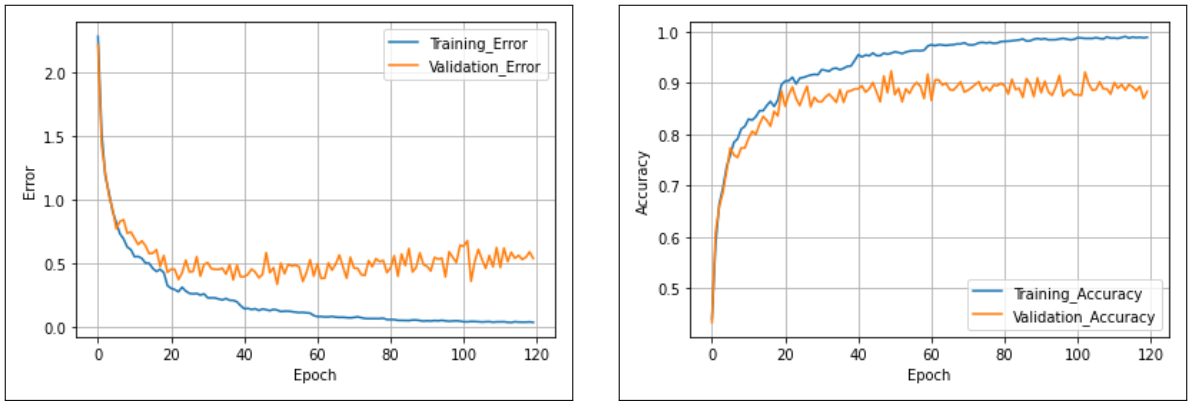
(a) (b)

Figure 5.6: (a) Trial 6: Training and Validation loss, (b) Trial 6: Training and Validation Accuracy. Kaiming He weight initialisation has a positive effect on the training and validation losses and accuracies.

The model showed superior performance among all network configurations tested so far in the project, as shown in Fig.5.6. However, performance increased slightly compared to the network configuration with dropout p=0.3 in the PointNetClassifier network with random weight initialization. The reason for the slight performance increase may be that batch normalization and weight initialization affect the network in a similar way by limiting the weights of the network in a certain range, as described in Section 4.2.

Trial 7: Kaiming Normal weight initialisation, Dropout(p=0.3) in PointNetClassifier network and Batch normalisation layer after every convolution and linear layers

Kaiming Normal weight initialisation technique was implemented instead of Kaiming uniform weight initialisation to investigate the effects of the different weight initialization technique on model performance. However, the experiment with this configuration showed that the model performed better with Kaiming Uniform Weight Initialisation than with Kaiming Normal Weight Initialisation.

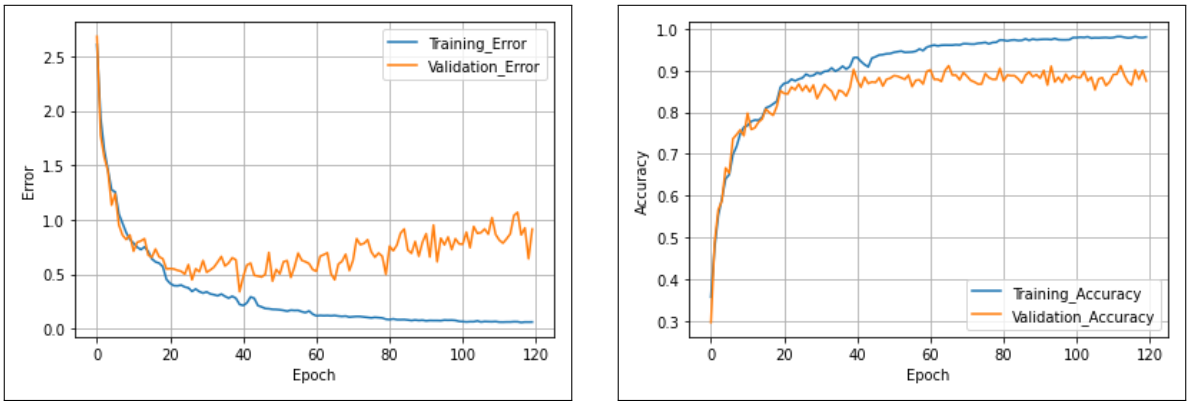


(a) (b)

Figure 5.7: (a) Trial 7: Training and Validation loss, (b) Trial 7: Training and Validation Accuracy. Kaiming He normal weight initialisation resulted in higher training and validation losses and lower accuracies as compared to the network configuration with Kaiming He uniform weight initialisation.

Trial 8: Kaiming uniform weight initialisation, Dropout(p=0.3) in PointNet-Classifier network and no Batch normalisation layer

To investigate the sole effect of weight initialisation without the contributions of the batch normalisation layers, another variant of the network with Kaiming uniform weight initialisation with dropout (p = 0.3) was implemented in the PointNetClassifier network without the use of batch normalisation. It turned out that the network with weight initialisation and batch normalisation performed better than the network that only used weight initialisation.

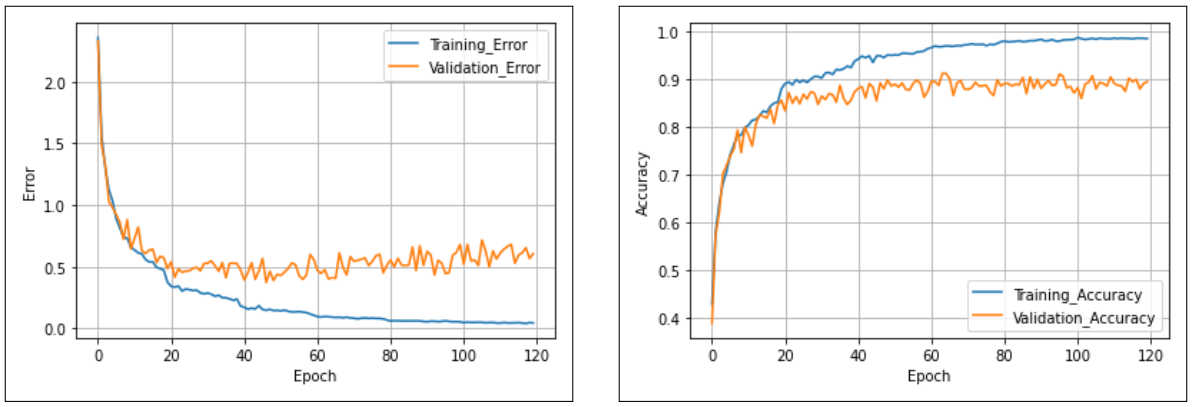


(a) (b)

Figure 5.8: (a) Trial 8: Training and Validation loss, (b) Trial 8: Training and Validation Accuracy. Omitting the batch normalisation layer resulted in increased training and validation losses and reduced accuracy further in the training.

Trial 9: Kaiming uniform weight initialisation, Dropout(p=0.3) in PointNet-Classifier network and Batch normalisation layer after every convolution and linear layer with Batch-norm layer applied before Dropout layer

The simultaneous use of batch normalisation and dropout layers and their placement in the network has implications for the performance of the model. Many studies suggest that different arrangement of batch normalisation and dropout layers leads to different model performance, as explained in Section 4.2. However, the differences in performance are subjective depending on the network. All network configurations experimented with so far in this project used the order - dropout and then batch normalisation for the implementation of the PointNet object classification network. Therefore, a new network configuration was tried in which the order of these two layers was reversed. The result of the experiment showed that the previous configuration worked well in the case of the PointNet object classification network.

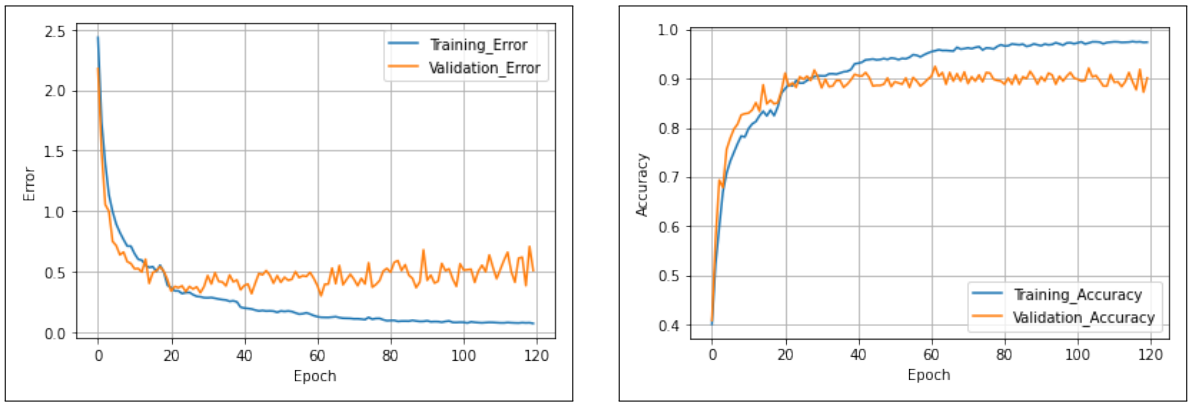


(a) (b)

Figure 5.9: (a) Trial 9: Training and Validation loss, (b) Trial 9: Training and Validation Accuracy. Placing the batch-normalisation layer before the dropout layer resulted in poorer performance of the model.

Trial 10: Kaiming uniform weight initialisation with Independent Component layer in PointNetClassifier network and Batch normalisation layer applied after every convolution and linear layers

An interesting study by Guangyong et. al [43] proposed a novel training procedure combining dropout and batch normalisation (collectively referred to as Independent Component (IC) layer) before each weight layer to make the network inputs more independent and achieve fast convergence (see Section 4.2). Therefore, in this study, another variant with an Independent Component layer (consisting of a batch normalisation layer with a dropout of 0.3 before the weight layer) was implemented in the PointNetClassifier network. This configuration performed marginally inferior as compared to the other configurations tested so far.

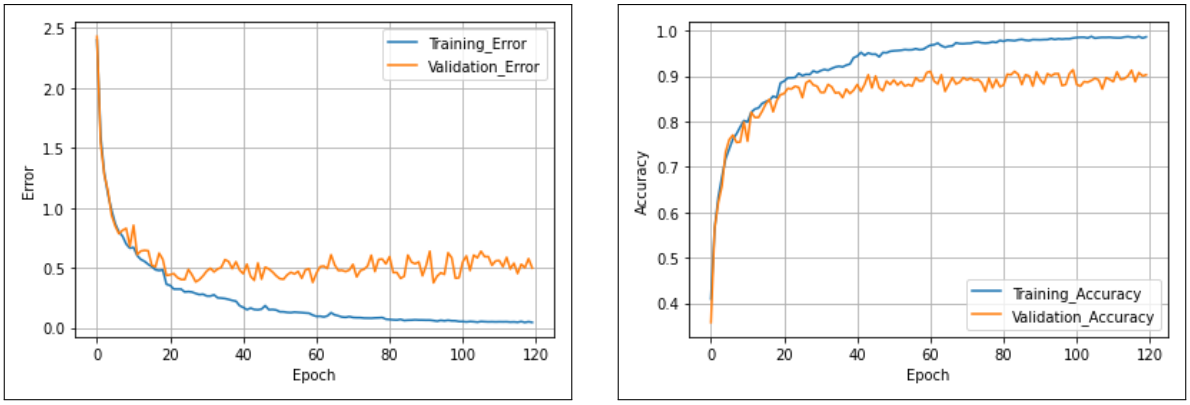


(a) (b)

Figure 5.10: (a) Trial 10: Training and Validation loss, (b) Trial 10: Training and Validation Accuracy. Arranging the dropout and batch normalisation layers as independent component layers shows positive effects on training and validation loss and accuracy, but is still worse than the best configuration found in the previous experiments.

Trial 11: Kaiming Uniform weight initialisation, Dropout (p=0.3) in the PointNetClassifier network, global feature vector dimension changed to 2048 and Batch normalisation layer applied after every convolution and linear layers

Different dimensions of layers in neural networks affect their performance, as explained in Section 4.2. A new variant of the object classification network, in which the dimension of the global feature vector has been changed from 1024 to 2048, has been implemented in this section.



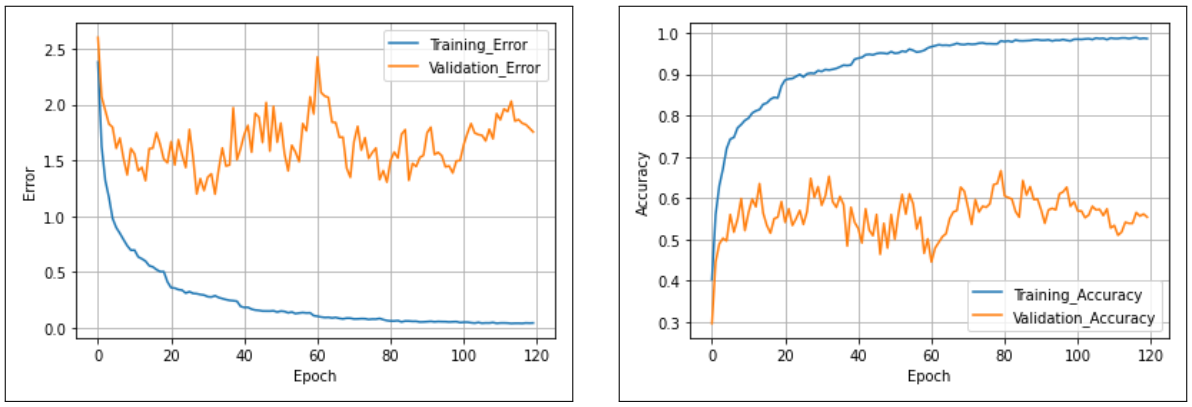
(a) (b)

Figure 5.11: (a) Trial 11: Training and Validation loss, (b) Trial 11: Training and Validation Accuracy. Increasing the dimension of the global feature vector to 2048 resulted in lower training and validation losses and lower accuracy.

This configuration performed slightly worse than the best performing configuration identified in the previously tested configurations. This underlines the fact that increasing the dimensions of the layer does not guarantee improved performance in all cases. A possible explanation for this could be that increasing the dimensions of the layers leads to an increase in the number of parameters to be trained, which in turn increases the complexity of training the network and thus also leads to poorer model performance.

Trial 12: Kaiming Uniform weight initialisation, two subsequent Dropouts (p=0.3,0.2) in PointNetClassifier network and Batch normalisation layer applied after every convolution and linear layers

A different network configuration with two dropout layers with decreasing dropout probability of 0.3 and 0.2 respectively was implemented in the hope of further improving the generalisation capability of the network.



(a) (b)

Figure 5.12: (a) Trial 12: Training and Validation loss, (b) Trial 12: Training and Validation Accuracy. The introduction of an additional dropout (p=0.2) into the PointNetClassifier network resulted in a drastic increase in training and validation losses and very low accuracy.

The model performed drastically worse compared to other network configurations, which can be attributed to the fact that two consecutive dropout layers with p=0.3 and 0.2, respectively, may have resulted in a significant loss of essential information for performing 3D object classification.

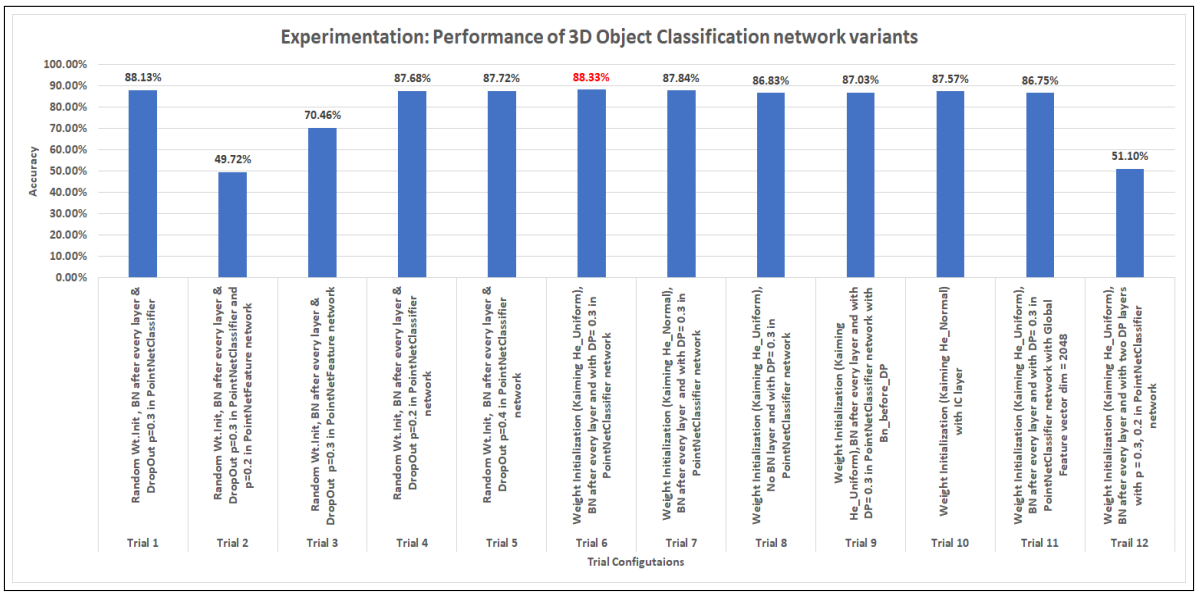


Figure 5.13: Performaces of different trial configurations of PointNet object classification network. The network configuration with Kaiming He uniform weight initialisation and with Dropout(p=0.3) in the PointNetClassifier network is the best performing configuration of all.

The performances of all network configurations are summarised in Fig. 5.13. The best performing configuration of the PointNet 3D object classification network is the one Kaiming Uniform weight initialisation and dropout of dropout probability p = 0.3 in the PointNetClassifier network.

Reproducibility test was carried out on the best performing network configuration. The results of the test are summarised in Table 5.1.

The overall accuracy of the PointNet 3D object classification network was found out to be $Accuracy = 87.914 \pm 0.297\%$

S. No	Accuacy
Trial A	88.01%
Trial B	87.76%
Trial C	87.53%
Trial D	87.94%
Trial E	88.33%

Table 5.1: Reproducibility test result on best performing 3D Object Classification network. The performance of the best performing confiration is highly reproducible.

Furthermore the best performing network (trained on ModelNet-40 dataset) was generalised on ScanObjectNN dataset to catch the attention on difference in model performance when the model is generalised on real life scans of the object as compared to input point clouds generated on CAD models of the object (see Table 5.2).

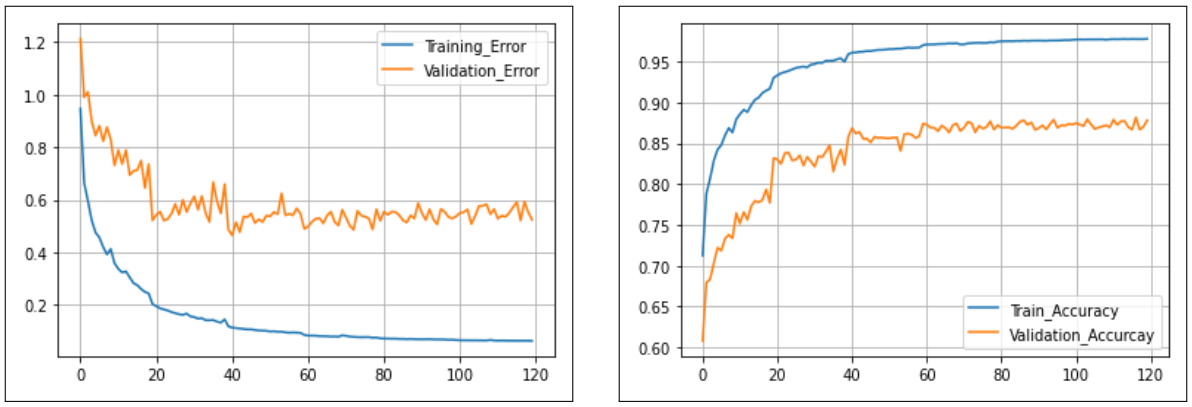
ScanObjectNN dataset varaint	Overall Accuracy
With noise and Background	72.34%
With noise but without Background	74.76%

Table 5.2: Performance of 3D Object classification network on real-life scan. The PointNet object classification network shows poorer model performance when exposed to real scan input than when exposed to clean CAD model-based input point clouds.

5.2 Results of 3D semantic segmentation

Trial 1: Random weight initialisation, no Dropout and batch normalisation applied after every convolution and linear layers

First, the implemented network was trained with a similar network configuration as proposed by Charles et. al [6], i.e. with batch normalisation after each linear and convolutional layer all configurations of the network and with random weight initialisation and no dropouts on the Stanford 3D Indoor Scene Dataset (S3DIS). The optimiser Adam has been used to train the model. The model performance with this base configuration would serve as a baseline result for further experiments aimed at improving the performance of the network.



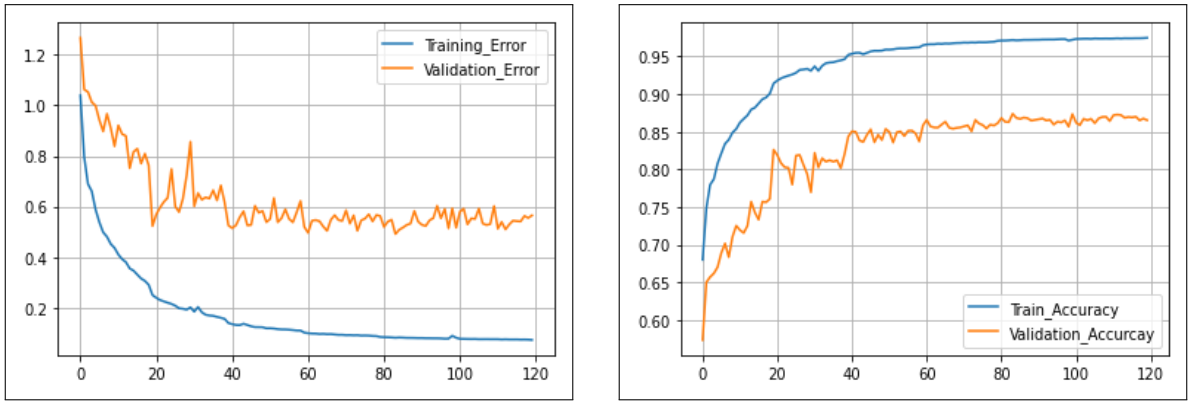
(a) (b)

Figure 5.14: (a) Semantic Segmentation Trial 1: Training and Validation loss, (b) Training and Validation Accuracy. Training and Validation losses show a decreasing trend and the respective accuracies shows an increasing trend over the coarse of training.

The decreasing trend of training and validation loss and the increasing trend of accuracy shows that the model learns to approximate the function required to perform the semantic segmentation task. Various techniques such as weight initialisation, dropouts, etc. have proven to be an efficient way to improve model performance.

Trial 2: Xavier uniform weight initialisation, no Dropout and batch normalisation applied after every convolution and linear layers

Weight initialisation techniques are widely used to solve the problem of vanishing and exploding gradients in deep neural networks. There are several popular weight initialisation schemes that have been shown to be useful in improving the performance of deep neural networks. In this section, Xavier Uniform weight initialisation scheme was introduced into the model.



(a) (b)

Figure 5.15: (a) Semantic Segmentation Trial 2: Training and Validation loss, (b) Training and Validation Accuracy. Xavier uniform weight initialisation resulted in lower training and validation losses and higher accuracies.

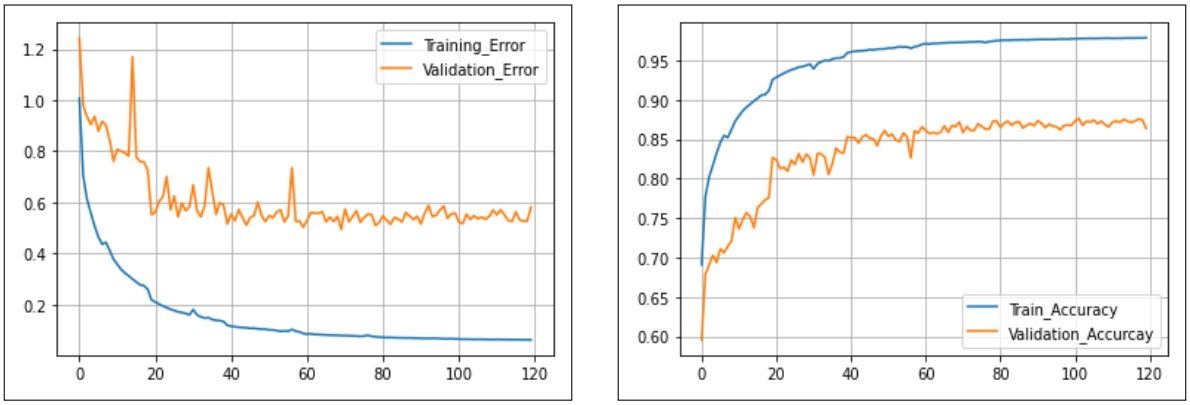
However, it was found that the implementation of Xavier weight initialisation did not significantly affect the performance of the model. One possible explanation for the insignificant change in the model's performance is that although weight initialisation and batch normalisation are completely different techniques, they nevertheless attempt to influence the model in a similar

way by keeping the parameters within a certain range to prevent the gradients from exploding or decreasing.

Other weight initialisation schemes were tested in further trials.

Trial 3: Xavier normal weight initialisation, no Dropout and batch normalisation applied after every convolution and linear layers

Xavier normal weight initialisation was included in the model instead of Xavier uniform weight initialisation to check the effects of other weight initialisation schemes.



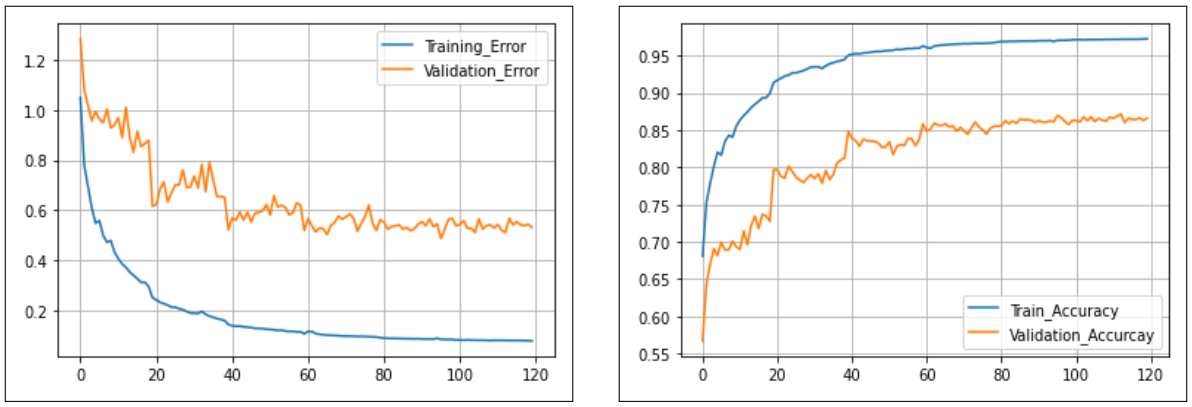
(a) (b)

Figure 5.16: (a) Semantic Segmentation Trial 3: Training and Validation loss, (b) Training and Validation Accuracy.Initialisation with Xavier normal weight resulted in lower training and validation losses and higher accuracy compared to initialisation with Xavier uniform weight.

The performance of the model has improved slightly compared to the configuration with Xavier Uniform weight initialisation scheme.

Trial 4: Kaiming He uniform weight initialisation, no Dropout and batch normalisation applied after every convolution and linear layers

In this trail Kaiming He Uniform weight initialisation scheme was introduced instead of Xavier Normal. The model showed a slight degradation in performance compared to the previous configuration.

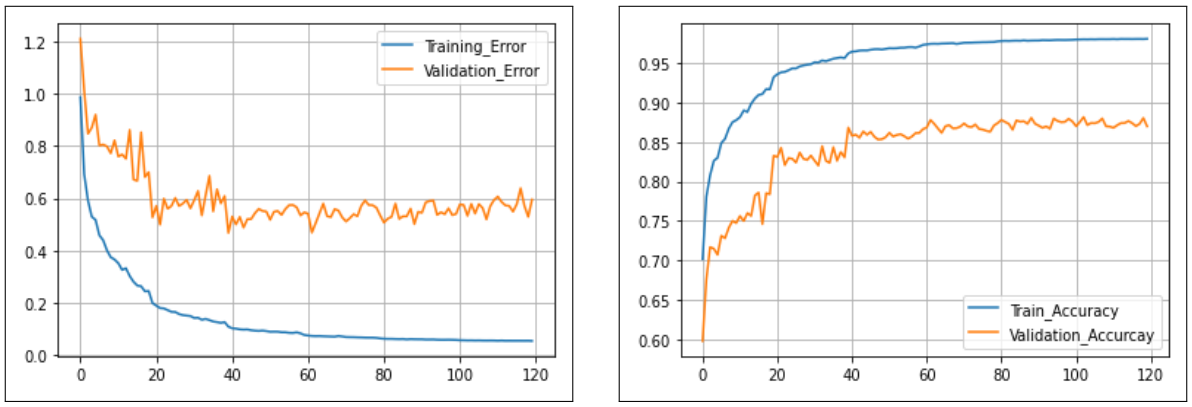


(a) (b)

Figure 5.17: (a) Semantic Segmentation Trial 4: Training and Validation loss, (b) Training and Validation Accuracy. Initialisation with Kaiming He uniform weight resulted in slightly higher training and validation losses and lower accuracy compared to initialisation with Xavier normal weight.

Trial 5: Kaiming He normal weight initialisation, no Dropout and batch normalisation applied after every convolution and linear layers

Kaiming Normal Weight initialisation scheme was introduced in place of Kaiming Uniform weight initialisation scheme in the trail. This model configuration showed better performance compared to the previously tested configurations.



(a) (b)

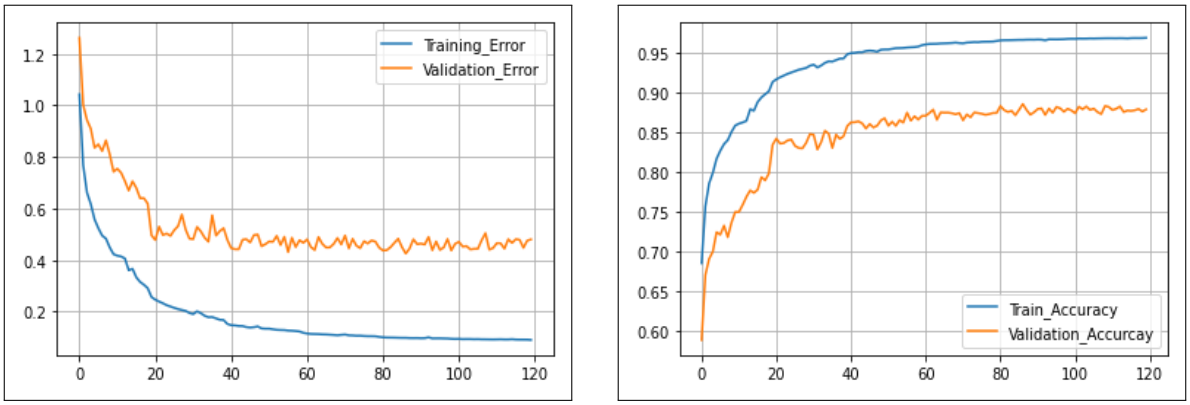
Figure 5.18: (a) Semantic Segmentation Trial 5: Training and Validation loss, (b) Training and Validation Accuracy. Initialisation with Kaiming He normal weights showed the lowest training and validation losses compared to the other experimental initialisation techniques for weights in the previous trials.

It was found that model configurations with normal weight initialisation performed better in both weight initialisation schemes proposed by Xavier et al. and Kaiming He. et al. [39]

Fig. 5.18 shows that the model performs well in the training data set, but not so well in the validation data set. Therefore, different variants with dropout techniques were implemented in the upcoming experiments.

Trial 6: Kaiming He normal weight initialisation, two subsequent Dropout (p=0.3,0.3) in PointNetSegmenter network and batch normalisation applied after every convolution and linear layers

The use of multiple dropout layers affects the flow of information in the network, as described in section 4.3. In the PointNetSegmenter network, two dropout layers were implemented with a fixed dropout probability p= 0.3 and 0.3 respectively.



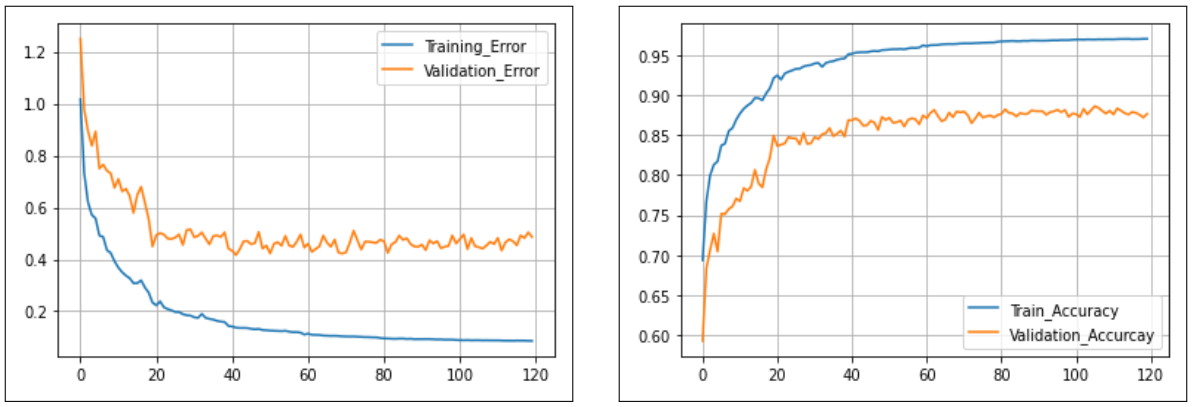
(a) (b)

Figure 5.19: (a) Semantic Segmentation Trial 6: Training and Validation loss, (b) Training and Validation Accuracy. The introduction of two dropouts (p=0.3,0.3) into the PointNetSegmenter network resulted in lower validation and training losses compared to the previous configuration attempts.

This network configuration showed significantly improved performance compared to the networks without dropout layers.

Trial 7: Kaiming He normal weight initialisation, two subsequent Dropout (p=0.3,0.2)in PointNetSegmenter network and batch normalisation applied after every convolution and linear layers

In this trail, two dropout layers were implemented in the PointNetSegmenter network, reducing the dropout probability p= 0.3 and 0.2, respectively, as one went deeper into the network. The performance of the model has further improved compared to the previous trial configuration.

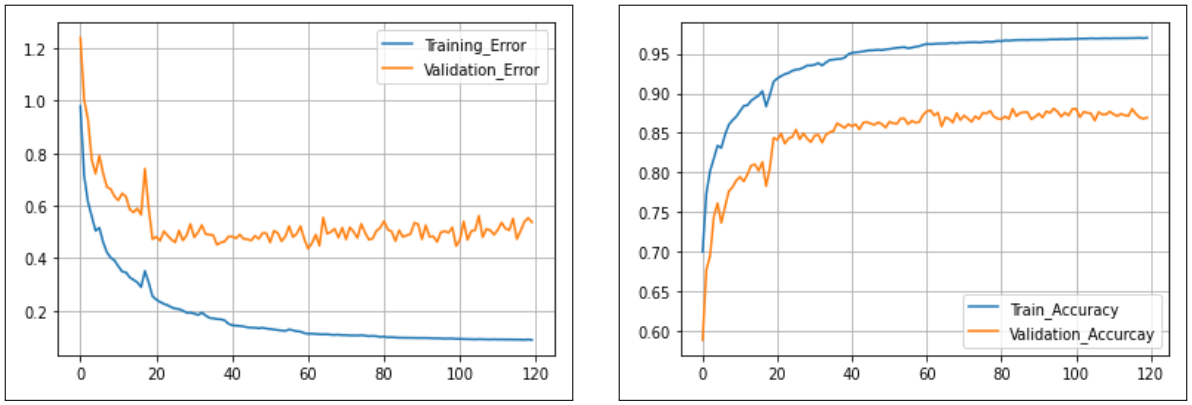


(a) (b)

Figure 5.20: (a) Semantic Segmentation Trial 7: Training and Validation loss, (b) Training and Validation Accuracy. Reducing the second dropout probability from 0.3 to 0.2 resulted in lower training and validation losses compared to the previous trial configuration.

Trial 8: Kaiming He normal weight initialisation, two subsequent Dropout (p=0.4,0.3)in PointNetSegmenter network and batch normalisation applied after every convolution and linear layers

As the performance of the previous trail configuration dropped, another network variant was implemented by increasing the dropout probability to 0.4 and 0.3. respectively.



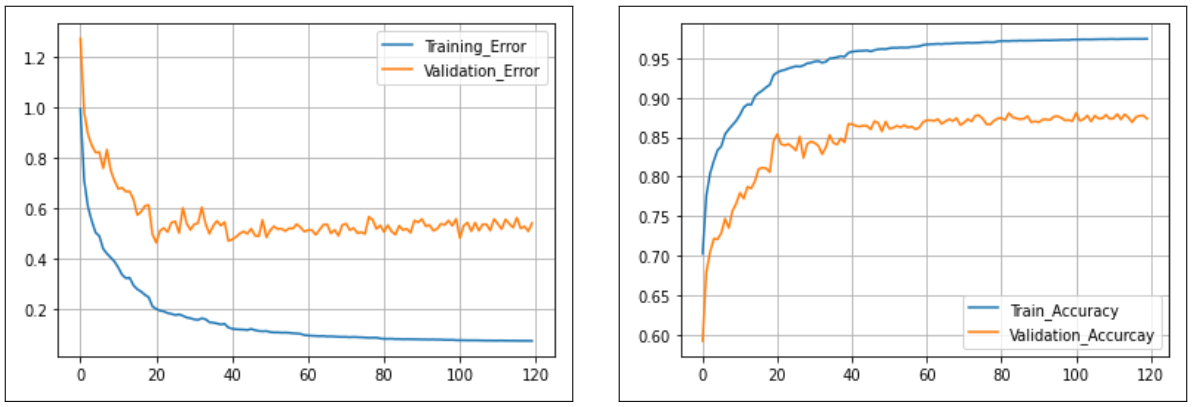
(a) (b)

Figure 5.21: (a) Semantic Segmentation Trial 8: Training and Validation loss, (b) Training and Validation Accuracy. This network configuration with dropout(p=0.4,0.3) into the PointNetSegmenter network showed the lowest training and validation losses.

This network configuration performed best compared to all previously tested network configurations of the 3D semantic segmentation network.

Trial 9: Kaiming He normal weight initialisation, two subsequent Dropout (p=0.2,0.1)in PointNetSegmenter network and batch normalisation applied after every convolution and linear layers

Another network was implemented with a similar configuration, further reducing the dropout probability of the layers to 0.2 and 0.1 respectively. However, this configuration performed worse than the previous network configuration.

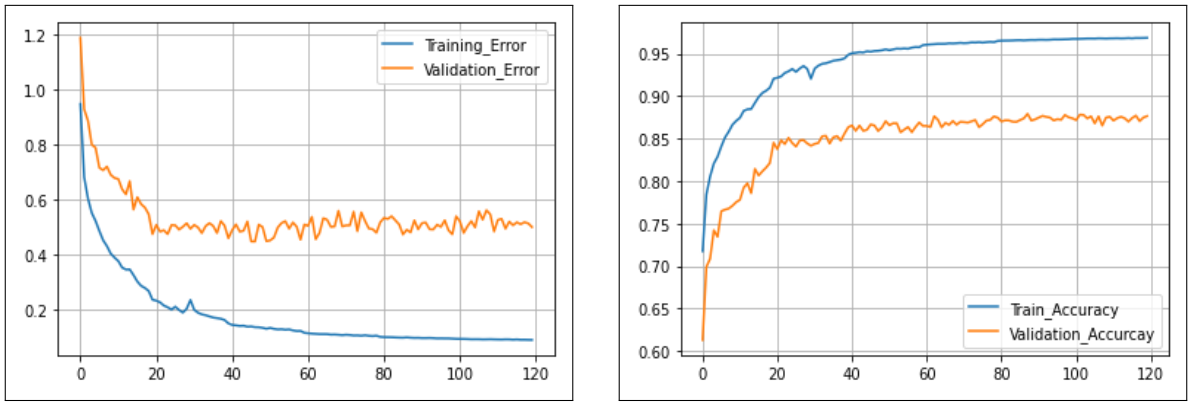


(a) (b)

Figure 5.22: (a) Semantic Segmentation Trial 9: Training and Validation loss, (b) Training and Validation Accuracy. This network configuration with dropout(p=0.2,0.1) showed higher training and validation losses as compared to the network configuration with dropout(p=0.4,0.3).

Trial 10: Kaiming He normal weight initialisation, two subsequent Dropout (p=0.5,0.2)in PointNetSegmenter network and batch normalisation applied after every convolution and linear layers

Another network variant was implemented by further increasing the dropout probability to 0.5 and 0.2 respectively. However, the performance of the network decreased significantly when the dropout probability was further increased, as can be seen in Fig.5.23.



(a) (b)

Figure 5.23: (a) Semantic Segmentation Trial 10: Training and Validation loss, (b) Training and Validation Accuracy.This network configuration with dropout(p=0.5,0.2) showed higher training and validation losses as compared to the network configuration with dropout(p=0.4,0.3)

The mIOUs of trail configurations of 3D semantic segmentation network are shown in Fig. 5.24.

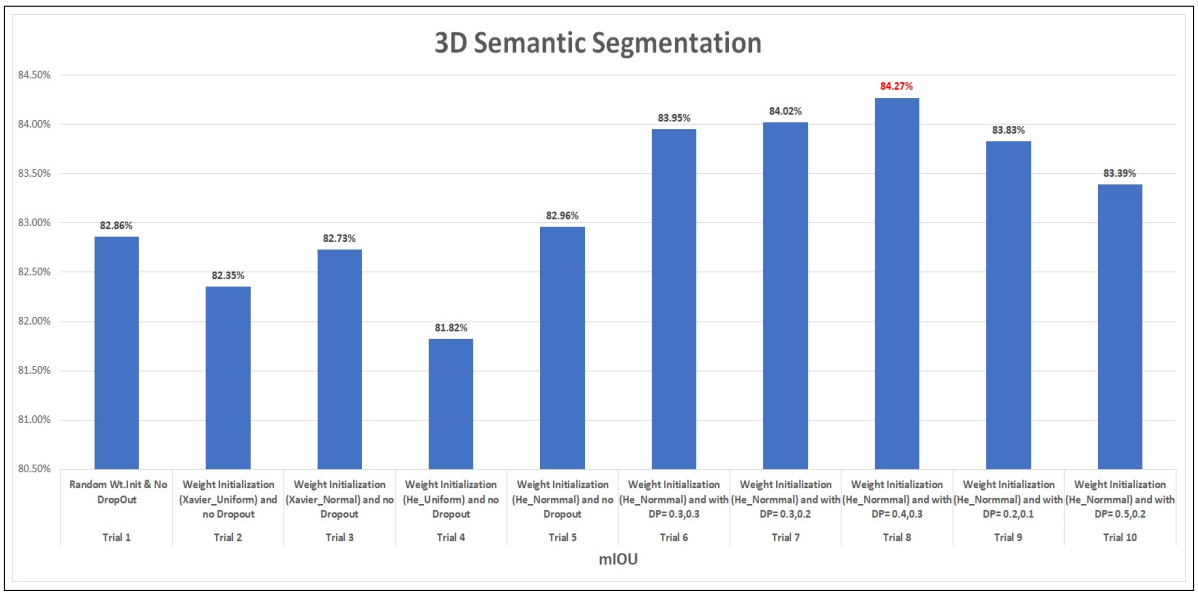


Figure 5.24: PointNet 3D Semnatic Segmentaion Network Performance. The network configuration with Kaiming He normal weight initialisation and with dropout(p=0.4,0.3) in the PointNetSegmenter network showed the best performance with mIOU = 84.27%

Thus, the best performing configuration of the PointNet 3D semantic segmentation network is the one with the Kaiming normal weight initialisation with two dropout layers in the PointNetSegmentaer network with dropout probability p =0.4 and 0.3, respectively.

In addition, the reproducibility test was conducted with the best performing configuration to check the consistency of the network's performance. The results of the reproducibility test are shown in Table 5.3.

S. No	**mIOU**
Trial A	84.09%
Trial B	83.67%
Trial C	84.27%
Trial D	83.47%
Trial E	84.06%

Table 5.3: Reproducibility test result on best performing 3D Semantic Segmentation network. The best performing PointNet semantic segmentation network shows highly reproducible performance.

Overall Performance of the PointNet 3D Semantic Segmentation Network:

$mIOU = 83.912 \pm 0.33\%$

The results of the 3D semantic segmentation network were visualised to provide a perceivable understanding of the model performance.

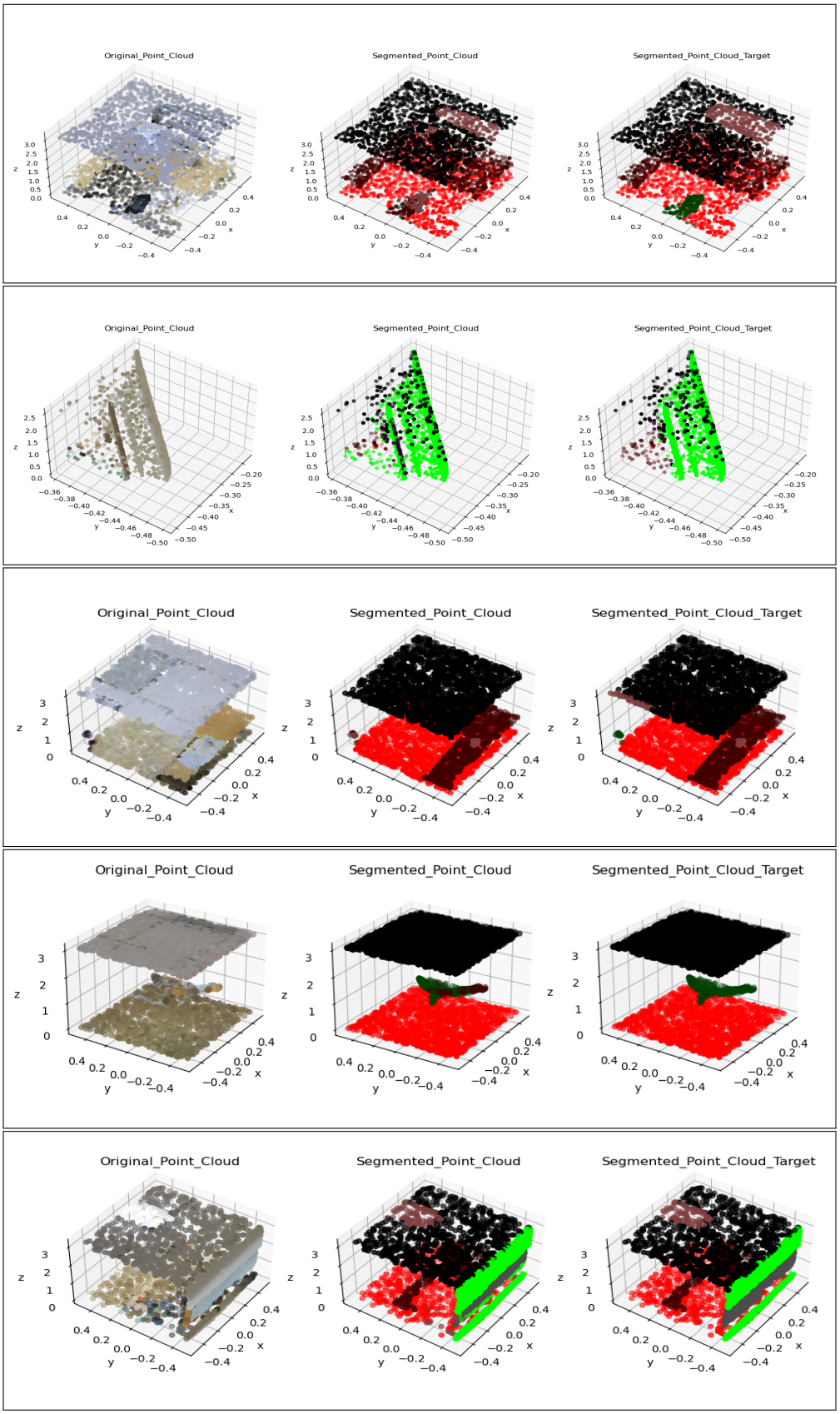
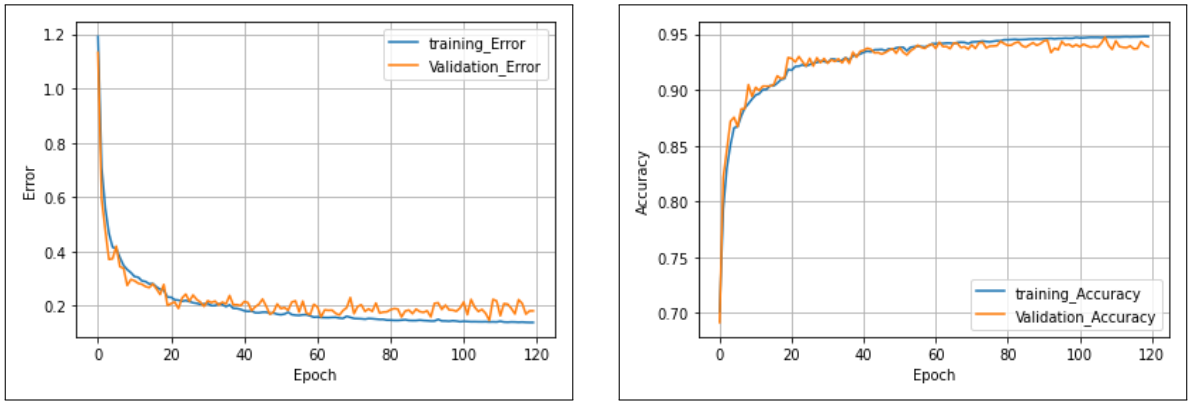


Figure 5.25: Result Visualisation of PointNet 3D Semantic Segmentation network. The point cloud on the left shows the input point cloud, the one in the middle shows the output of the PointNet semantic segmentation network and the one on the right shows the target point cloud. Different colours in the point cloud indicate different segmented semantic scenes in space.

5.3 Results of 3D part segmentation

As explained in Section 4.4, the 3D part segmentation network is essentially the same network as the 3D semantic segmentation network. Therefore, the best performing configuration of the 3D semantic segmentation network was used directly for the part segmentation task with a new Softmax layer that outputs class scores over 50 classes, where these classes represent the different object parts in the dataset. The 3D part segmentation network was trained and tested with the Shapenet dataset. The network configuration used for part segmentation has Kaiming normal weight initialisation and two dropout layers in the PointNetSegmenter network with a dropout probability p = 0.4 and 0.3 respectively.



(a) (b)

Figure 5.26: (a) Part Segmentation Trial: Training and Validation loss, (b) Training and Validation Accuracy. This network configuration had very low training and validation loss and the distance between the validation and training loss and accuracy curves is also less.

Fig. 5.26 shows that the network performed very well on both the training and validation data. The model performance was mIOU = 98.64%.

A reproducibility test was performed on this network configuration to verify the consistency of the network's performance. The results of the reproducibility test are shown in Table 5.4.

S. No	mIOU
Trial A	98.64%
Trial B	98.59%
Trial C	98.64%
Trial D	98.58%
Trial E	98.62%

Table 5.4: Reproducibility test result on best performing 3D Part Segmentation network. This network configuration shows highly reproducible performance.

Overall performance of the PointNet 3D Part Segmentation network:

$mIOU = 98.614 \pm 0.0279\%$

However, it was noticed that the Shapenet dataset has quite an unbalanced number of instances belonging to different objects. Hence, a category wise analysis of the model performance was done to investigate the model performance for each object category.

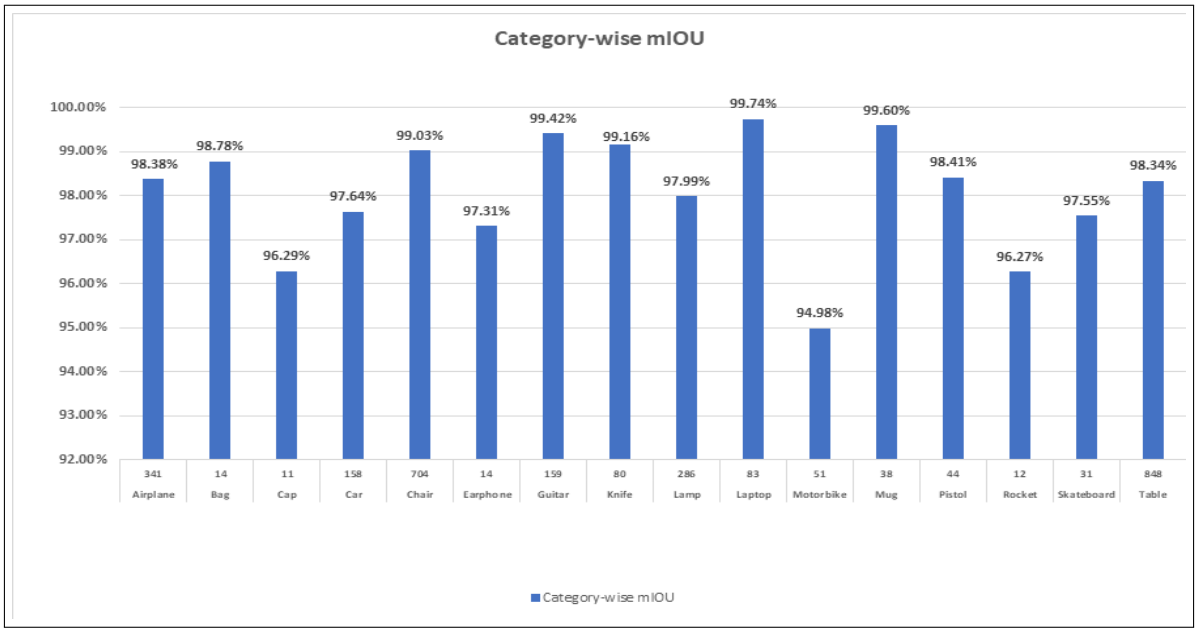


Figure 5.27: Category-wise Performance of PointNet 3D Part Segmentaion network. The number of category instances is indicated on the lateral axis together with the category name. The model performs very well on simpler shapes such as laptops, chairs, etc., but not so well on complex shapes such as motorbikes, cars, etc.

Fig. 5.27 shows the performance of the network for 3D part segmentation in each category. It can be seen that the model performs very well for objects with simpler shapes such as chair, table, cup, knife, bag, etc., but relatively poorly for complex shaped objects such as motorbike, rocket, etc.

The results of the 3D part segmentation network were visualised to provide a perceivable understanding of the model performance.

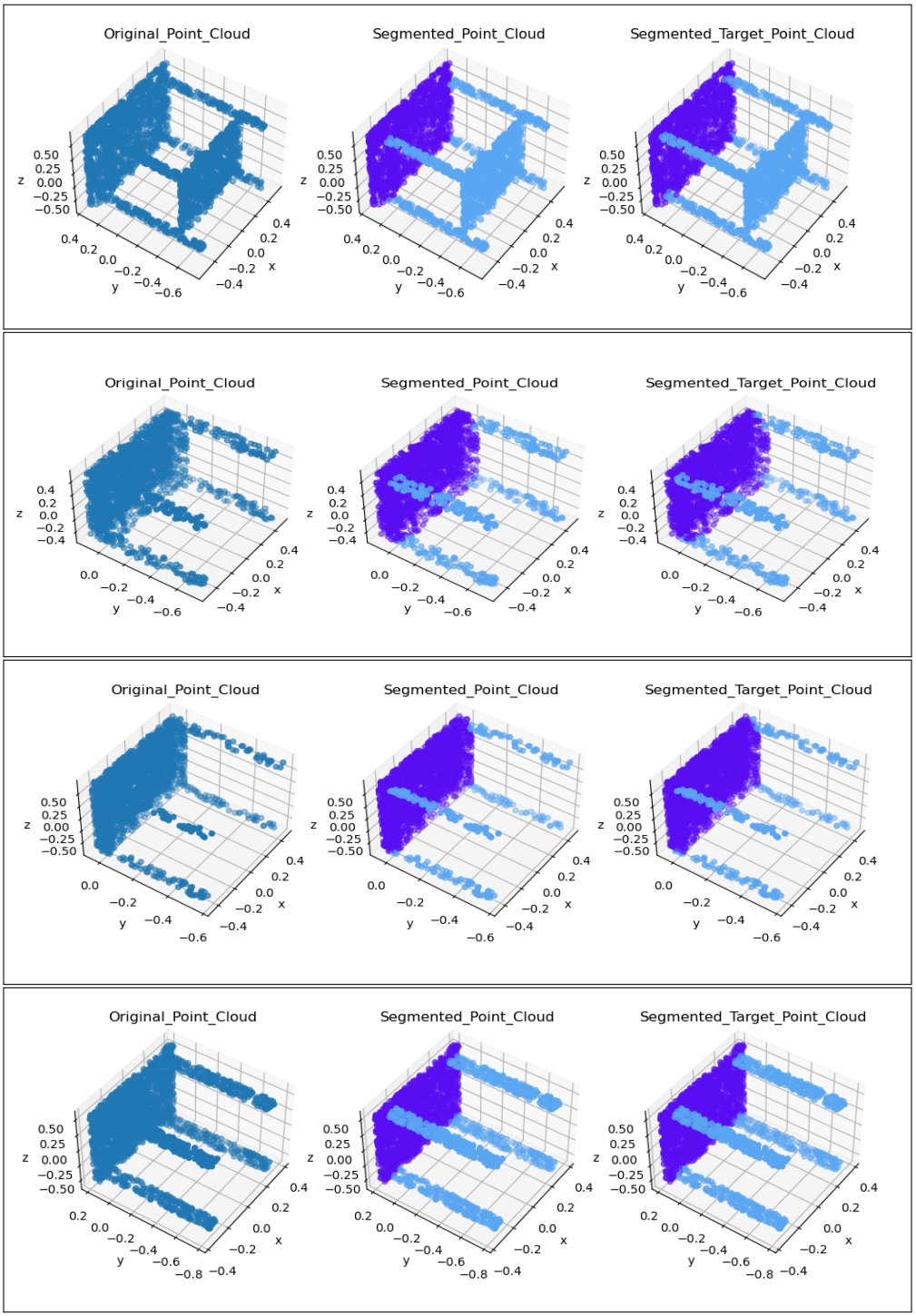


Figure 5.28: Result Visualisation of PointNet 3D Part Segmentation network. The point cloud on the left shows the input point cloud, the one in the middle shows the output of the PointNet part segmentation network and the one on the right shows the target point cloud. Different colours in the point cloud indicate different segmented parts in the object instance.

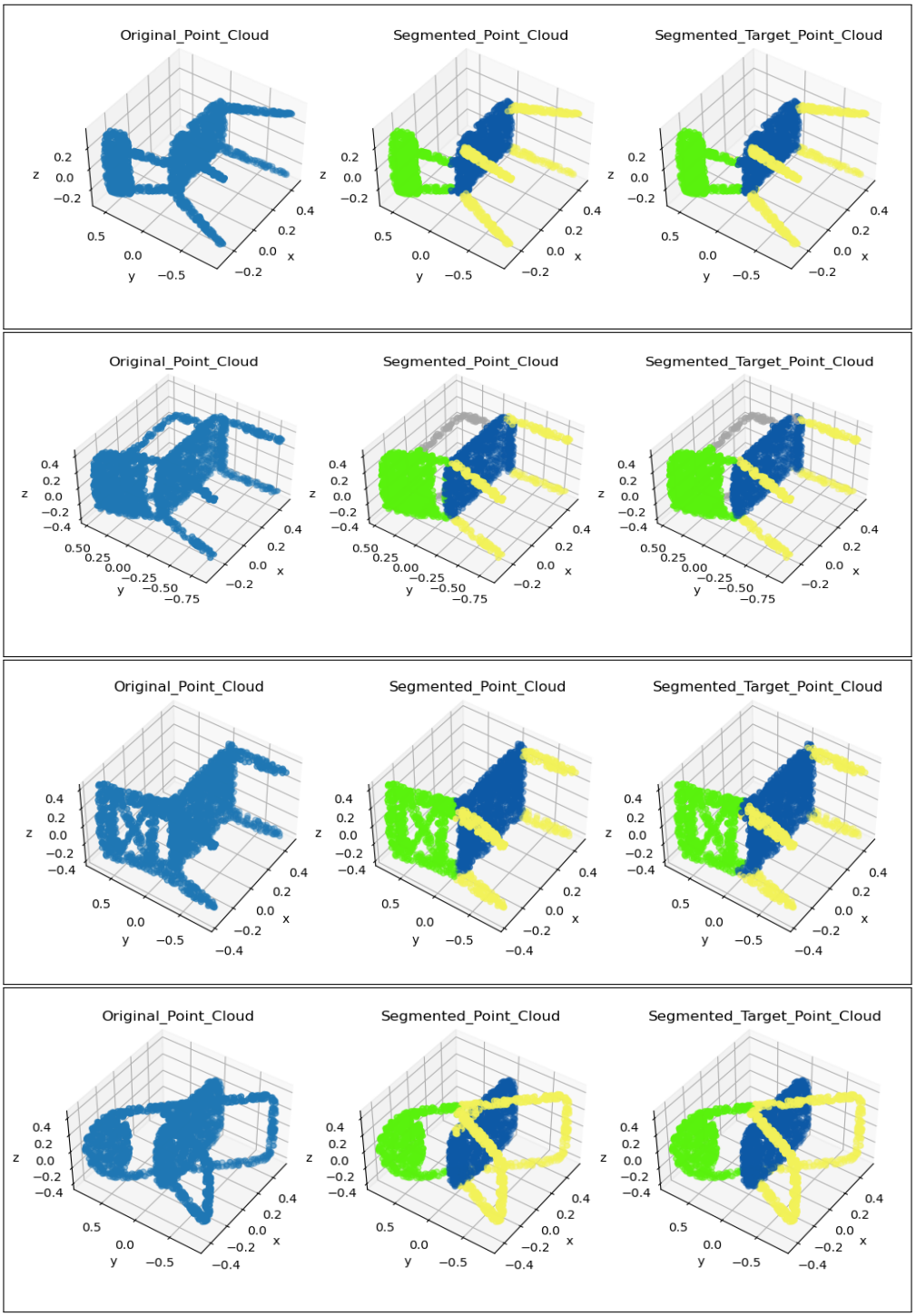


Figure 5.29: Result Visualisation of PointNet 3D Part Segmentation network.The point cloud on the left shows the input point cloud, the one in the middle shows the output of the PointNet part segmentation network and the one on the right shows the target point cloud. Different colours in the point cloud indicate different segmented parts in the object instance.

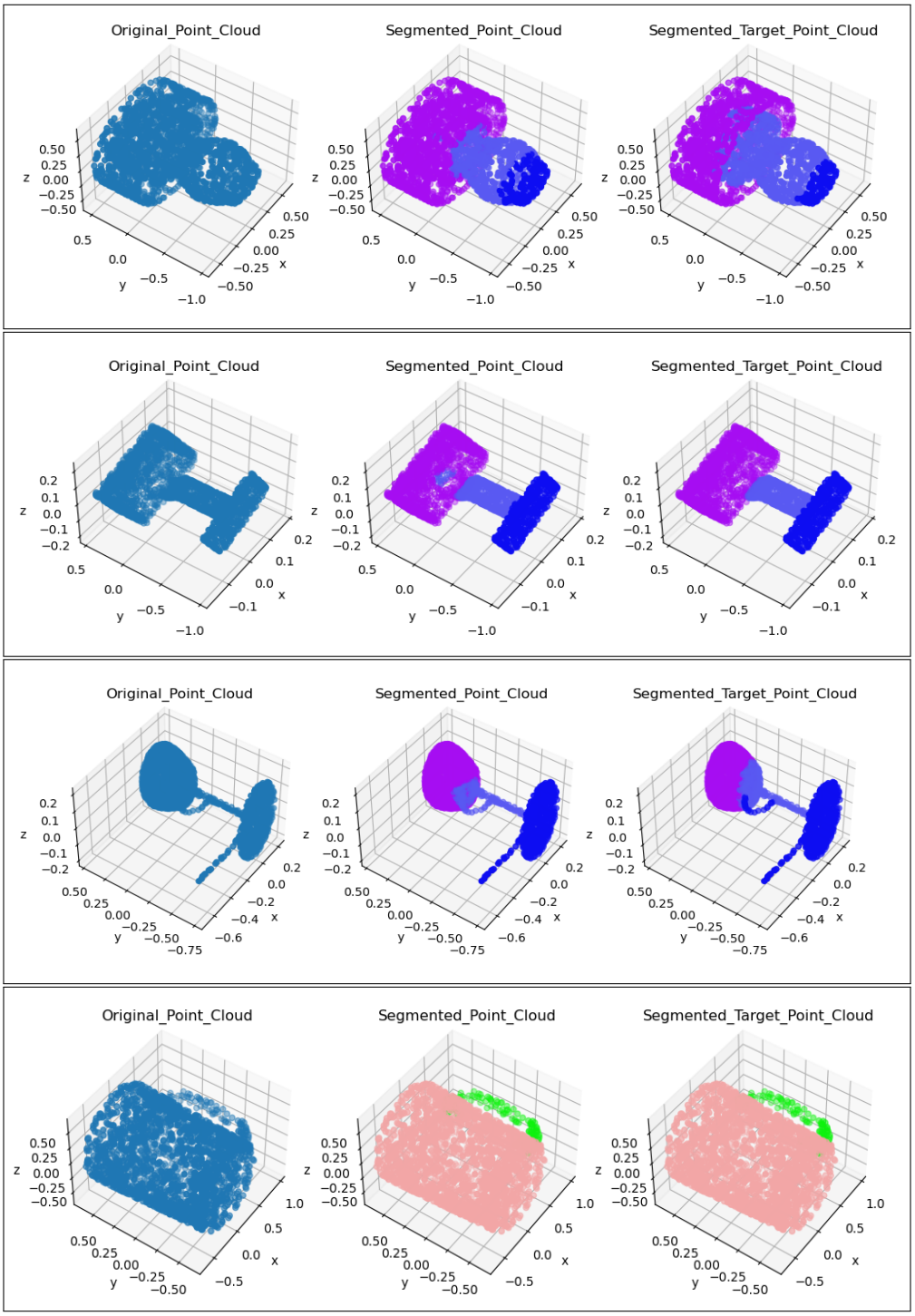


Figure 5.30: Result Visualisation of PointNet 3D Part Segmentation network.The point cloud on the left shows the input point cloud, the one in the middle shows the output of the PointNet part segmentation network and the one on the right shows the target point cloud. Different colours in the point cloud indicate different segmented parts in the object instance

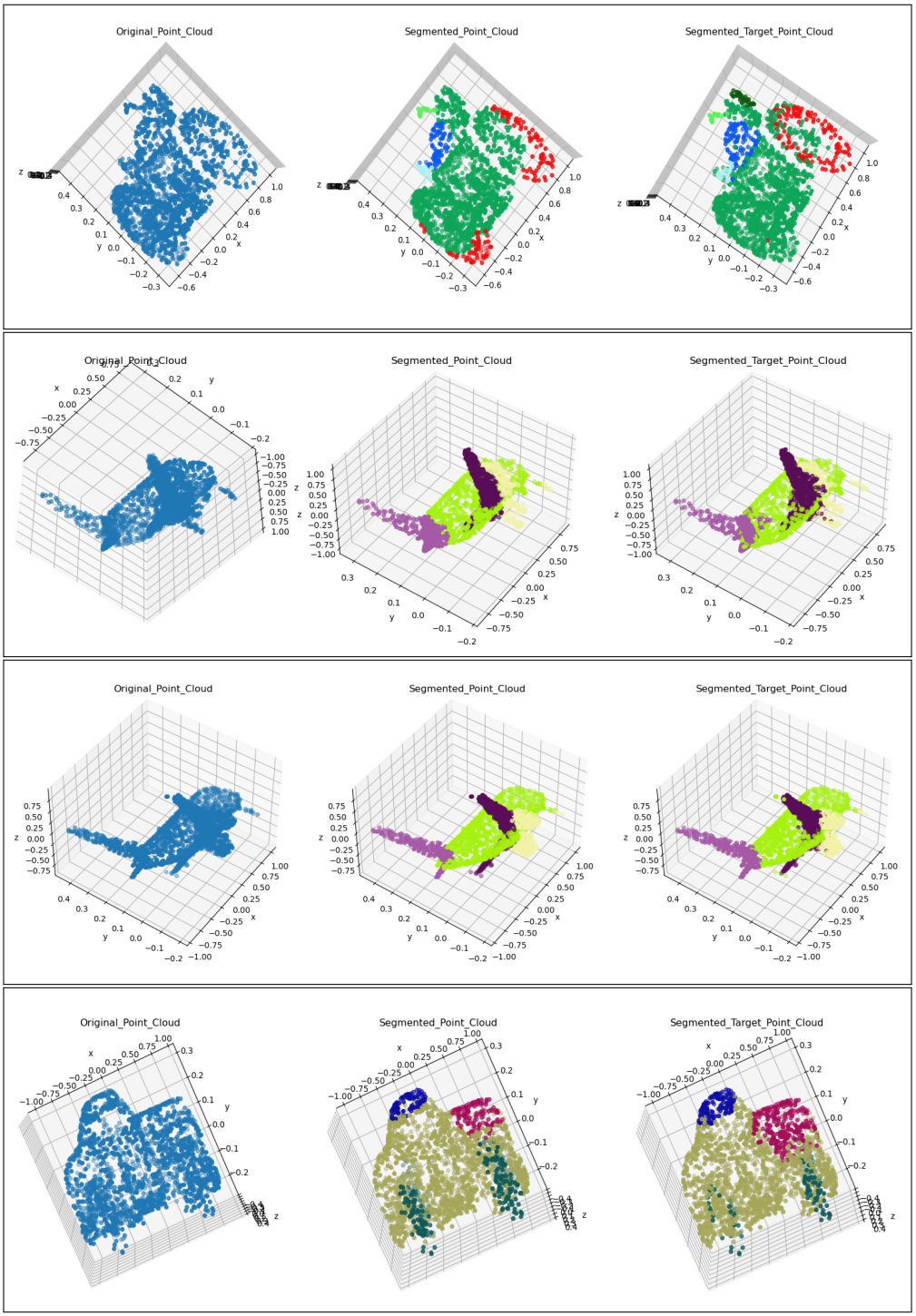


Figure 5.31: Result Visualisation of PointNet 3D Part Segmentation network.The point cloud on the left shows the input point cloud, the one in the middle shows the output of the PointNet part segmentation network and the one on the right shows the target point cloud. Different colours in the point cloud indicate different segmented parts in the object instance

In addition, the model was generalised to the Shapenet-C dataset to show the effects of perturbations such as scaling, rotation, jitter, addition and dropping of local and global points on the performance of the model.

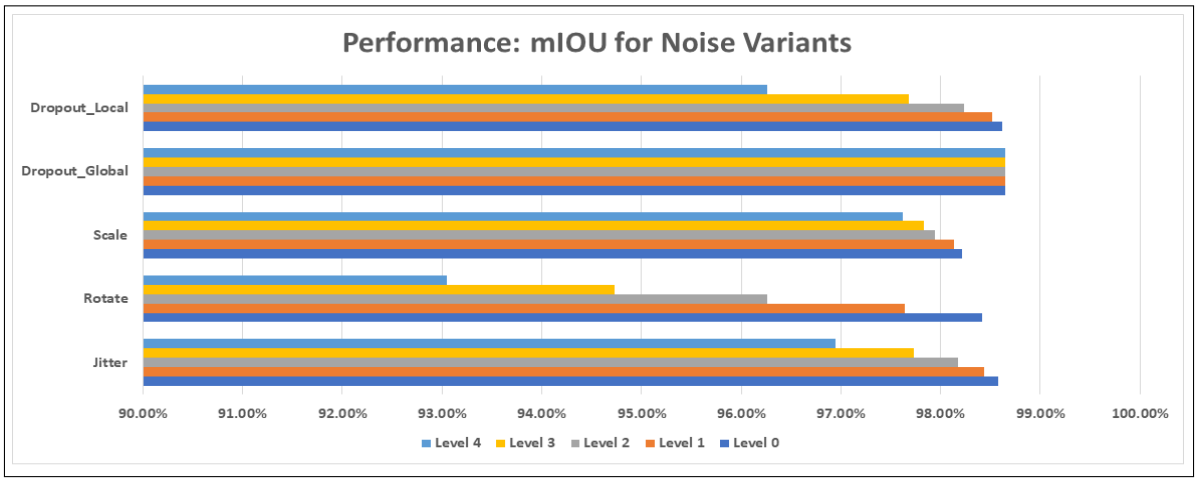


Figure 5.32: Performance of PointNet 3D Part Segmentation network with Noisy input. The performance of the moel deteriorated as the intensity of the noise increased. The degradation was more severe when the model was exposed to Rotation noise.

Figure 5.32 shows the performance of the 3D part segmentation model when the input point cloud is subjected to noises such as Dropout_Local, Dropout_Global, Scaling, Rotation and Jitter. All these noises were described in detail in section 4.1.3. It can be seen from Fig. 5.32 that the model performs quite robustly for inputs exposed to Dropout_Local, Dropout_Global, Scale and Jitter, but not so robustly for inputs exposed to Rotation noise. It was expected that the network would perform robustly in the presence of noise such as rotation, which is essentially an affine transformation of the input point cloud, since the network consists of a modular network, namely the T-net, which is used to pose normalisation of the input point clouds.

Therefore, an ablation study of the 3D part segmentation network was performed by omitting the STN3d network corresponding to the input transform functional block in the network. Using the Shapenet dataset, this network configuration was trained and tested with the Shapenet-C dataset. This revealed an interesting aspect of the PointNet 3D part segmentation network, namely that the modular network T-Net has no significant effect on model performance when the model is exposed to an input point cloud generated by affine transformation of the input such as scaling and rotation, as shown in Fig. 5.33.

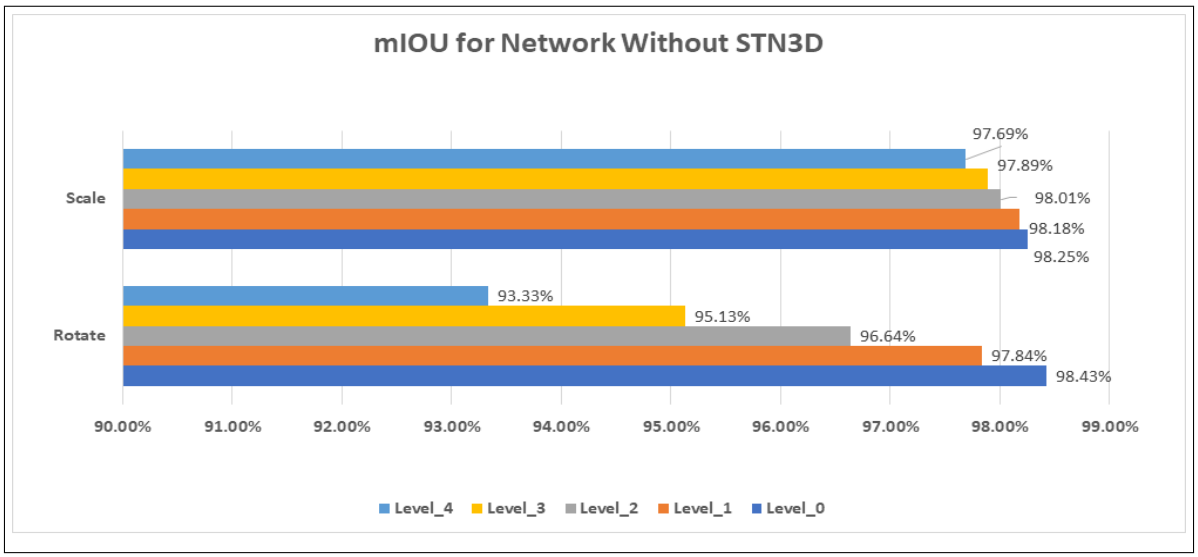


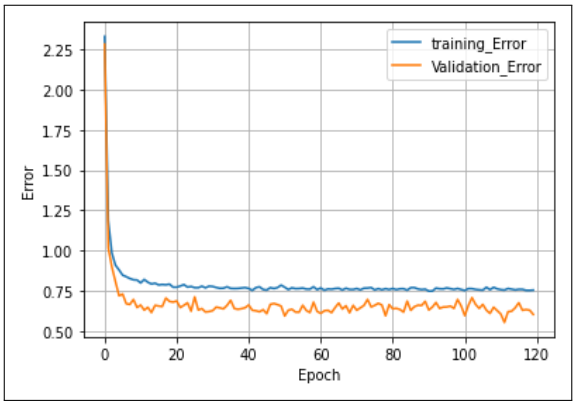
Figure 5.33: Performance of PointNet 3D Part Segmentation network without STN3D network. It was found that the degradation of model performance is the same for noises such as scaling and rotation in the network with and without input transform.

In addition, the generalisation of the 3D part segmentation network to the part of the Shapnet-C dataset consisting of noise, such as the addition of local and global noisy points, required the implementation of a new softmax layer in the network to generate an additional class score corresponding to the added noisy points. Transfer learning was used to transfer the learning from the 3D part segmentation model trained on the Shapenet dataset.

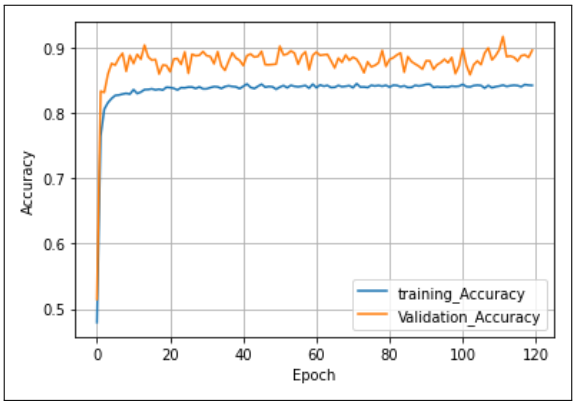
Two variants of the network were implemented using Transfer Learning. Both variants of the network were trained and tested separately on Add_Local and Add_Global noise data.

Variant 1: All parameters in the 3D part segmentation network were frozen until the last convolutional layer of the network, so that only the last convolutional layer of the network participated in the training process.

Initially, variant 1 was trained on input data exposed to Add_Local noisy data of minimal severity, i.e. level 0. Fig.5.34 shows the training and validation curves of variant 1 on Add_Local data.



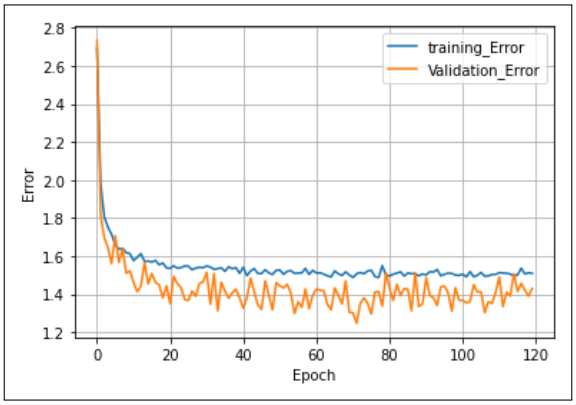
(a)



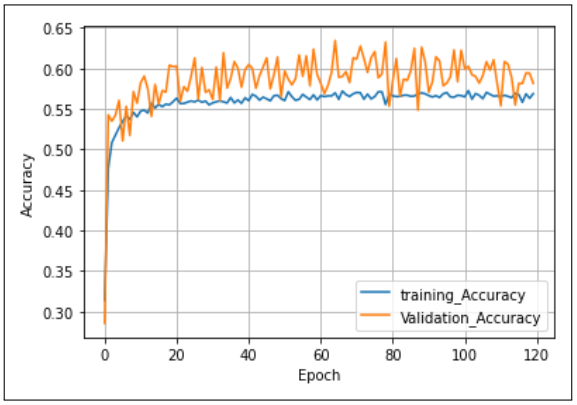
(b)

Figure 5.34: (a) Variant 1: Training and Validation loss on Add_Local Noisy input, (b) Training and Validation Accuracy on Add_Local Noisy input. The distance between training and validation curves is small. The validation loss curve is above the training loss curve.

Then, network variant 1 was trained separately on input data exposed to Add_Global noisy data of the lowest severity, i.e. level 0. Fig.5.35 shows the training and validation curves of variant 1 on Add_Global noisy data.



(a)



(b)

Figure 5.35: (a) Variant 1: Training and Validation loss on Add_Global Noisy input, (b) Training and Validation Accuracy on Add_Global Noisy input. The gap between the training and validation curves is small. The validation loss curve is above the training loss curve, as in the case of the model trained with the noisy Add_Local Level 0 data.

It can be seen that the training loss in both cases is lower than the validation loss in variant 1 and that the curves are more or less constantly apart. A possible explanation for such behaviour could be the fact that dropout penalises the variance of the model by randomly dropping the neurons in a layer only during the training of the model and therefore only affects the training loss.

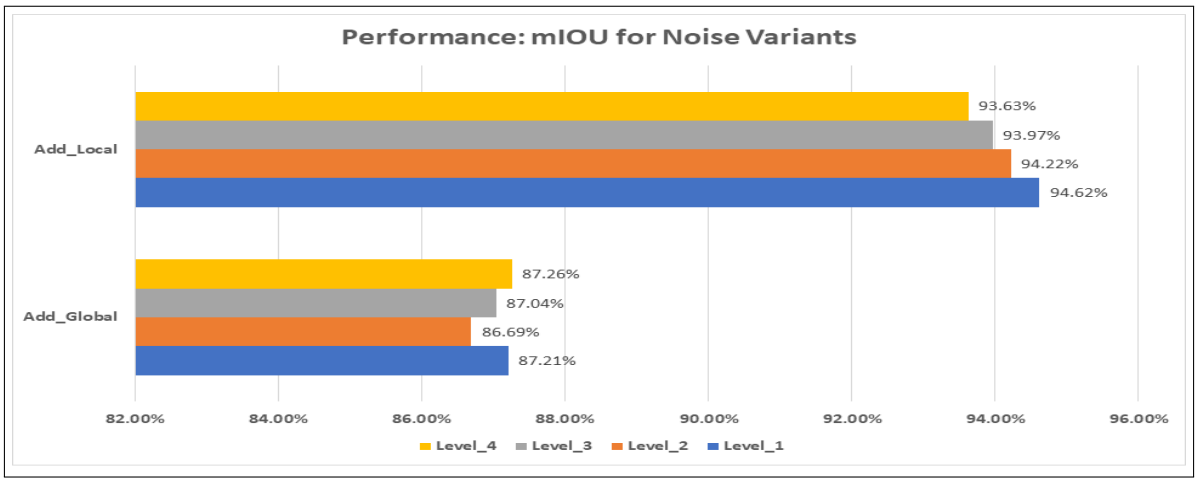


Figure 5.36: Performance of PointNet 3D Part Segmentation network Variant 1 with Noisy input. Model performance shows a decreasing trend with increasing noise strength in the case of Add_local noise. However, no plausible trend in model performance was observed in the case of Add_Global noise

Fig. 5.36 shows the performance of network variant 1 when subjected to Add_Local and Add_Global noise. As expected, the performance of the model decreased with increasing the severity of the noise in the case of Add_Local noise, but not in the case of Add_Local noise. The model did not show a plausible pattern in model performance when subjected to Add_Global noise of increasing severity.

The results of the network variant 1 were visualized to gain a perceivable understanding of the model performance.

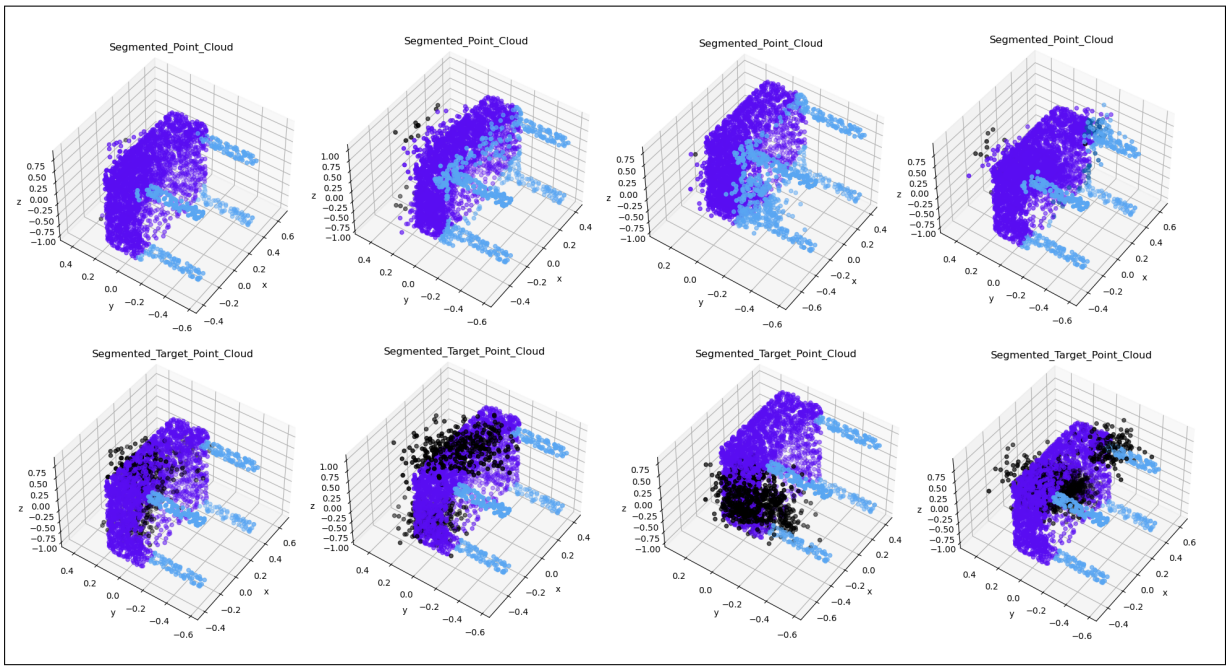


Figure 5.37: Result visualisation of Variant 1 on Add_Local noisy input. The point clouds in the first row show the output of the network and the point clouds at the bottom show the respective target point clouds. The visualisation of the result shows that this network configuration does not respond well to the Add_Local noise

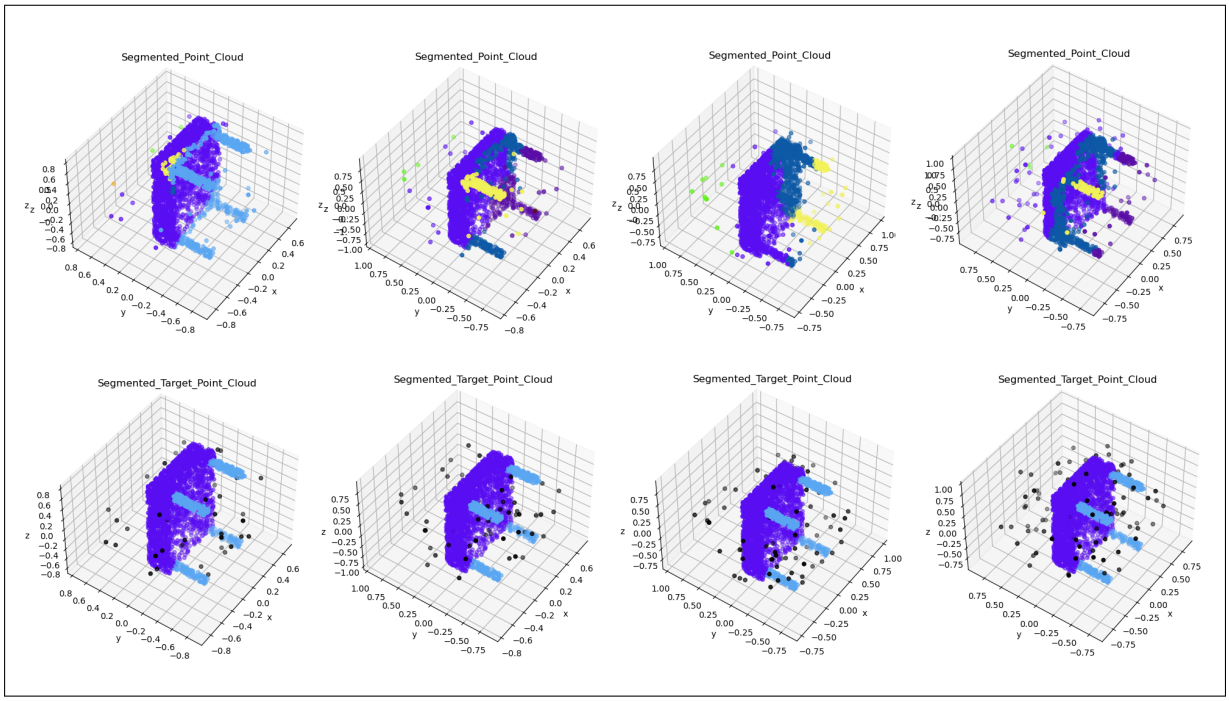
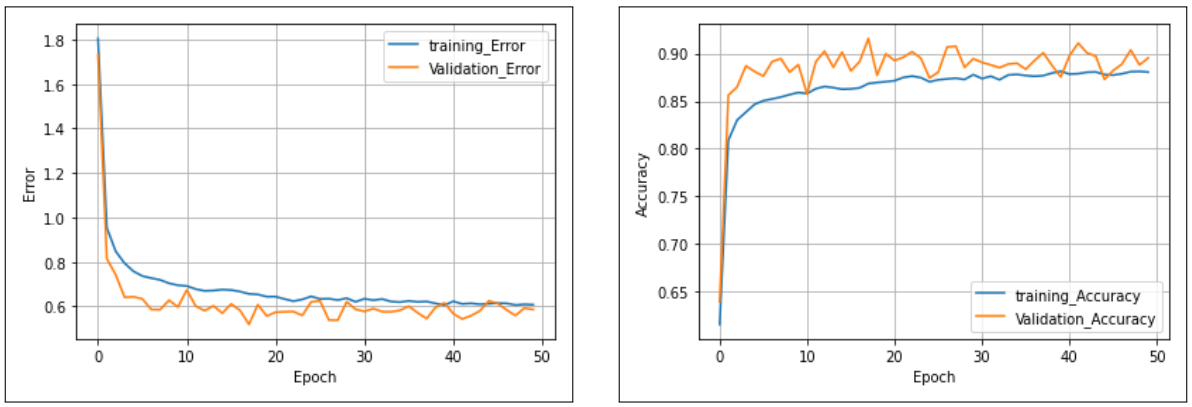


Figure 5.38: Result visualisation of Variant 1 on Add_Global noisy input.The point clouds in the first row show the output of the network and the point clouds at the bottom show the respective target point clouds. The visualisation of the result shows that this network configuration does not respond well to the Add_Global noise

Variant 2: All parameters in the 3D part segmentation network were frozen until the last three convolutional layers of the network, so that only the last three convolutional layers of the network participated in the training process.

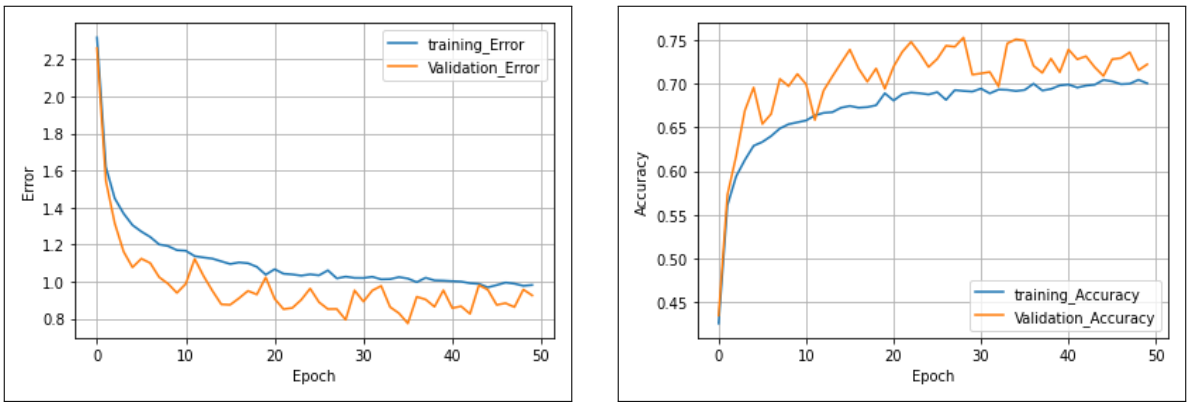
A similar approach to the previous variant was taken to investigate the effects of the noisy inputs of Add_Local and Add_Global on the performance of this variant.

The variant 2 was trained on input data exposed to Add_Local noisy data of minimum severity i.e Level 0 at first and then on Add_Global noisy data of minimum severity. Fig.5.39 and 5.40 show the training and validation curves of Variant 2 on Add_Local and Add_Global noisy data respectively.



(a) (b)

Figure 5.39: (a) Variant 2: Training and Validation loss on Add_Local Noisy input, (b) Training and Validation Accuracy on Add_Local Noisy input.The distance between training and validation curves is small. The validation loss curve lies above the training loss curve, and the distance between the training and validation curves decreases as training progresses.



(a) (b)

Figure 5.40: (a) Variant 2: Training and Validation loss on Add_Global Noisy input, (b) Training and Validation Accuracy on Add_Global Noisy input. The distance between training and validation curves is small. The validation loss curve lies above the training loss curve, and the distance between the training and validation curves decreases as training progresses.

Figs. 5.39 and 5.40 show that the training and validation curves show similar behaviour to the previous variant (see Figs. 5.34 and 5.35). The validation loss is still consistently lower than the training loss and the possible reason for this has already been explained in the prior case.

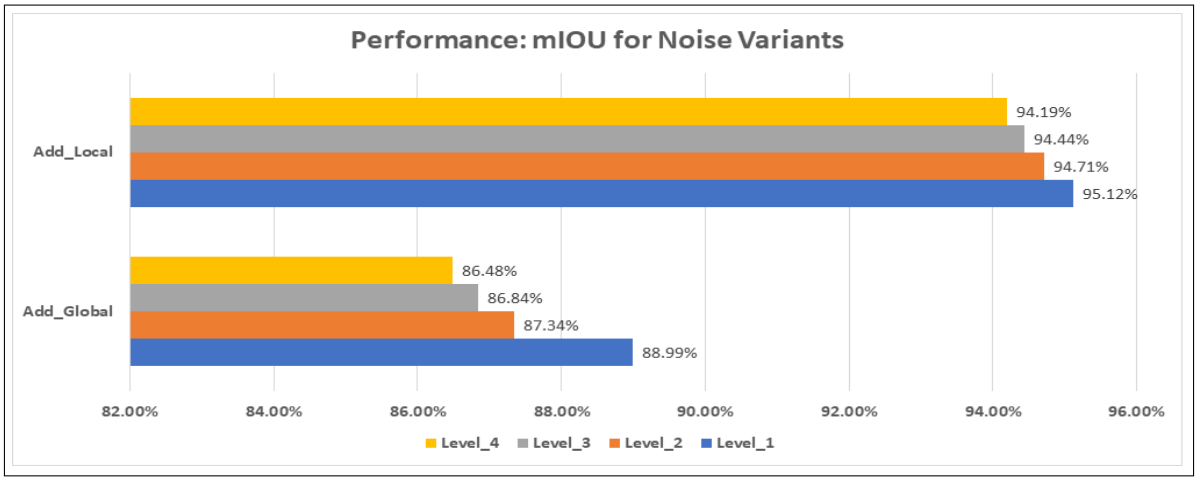


Figure 5.41: Performance of PointNet 3D Part Segmentation network Variant 2 with Noisy input. The performance of the model shows a decreasing trend with increasing strength of noise for both Add_local and Add_Global noise.

Now, however, there is a plausible trend that the performance of the model deteriorates as the severity of noise increases, both for the noisy inputs of Add_Local and Add_Global (see Fig. 5.41).

The results of the network variant 2 were visualized to gain a perceivable understanding of the model performance.

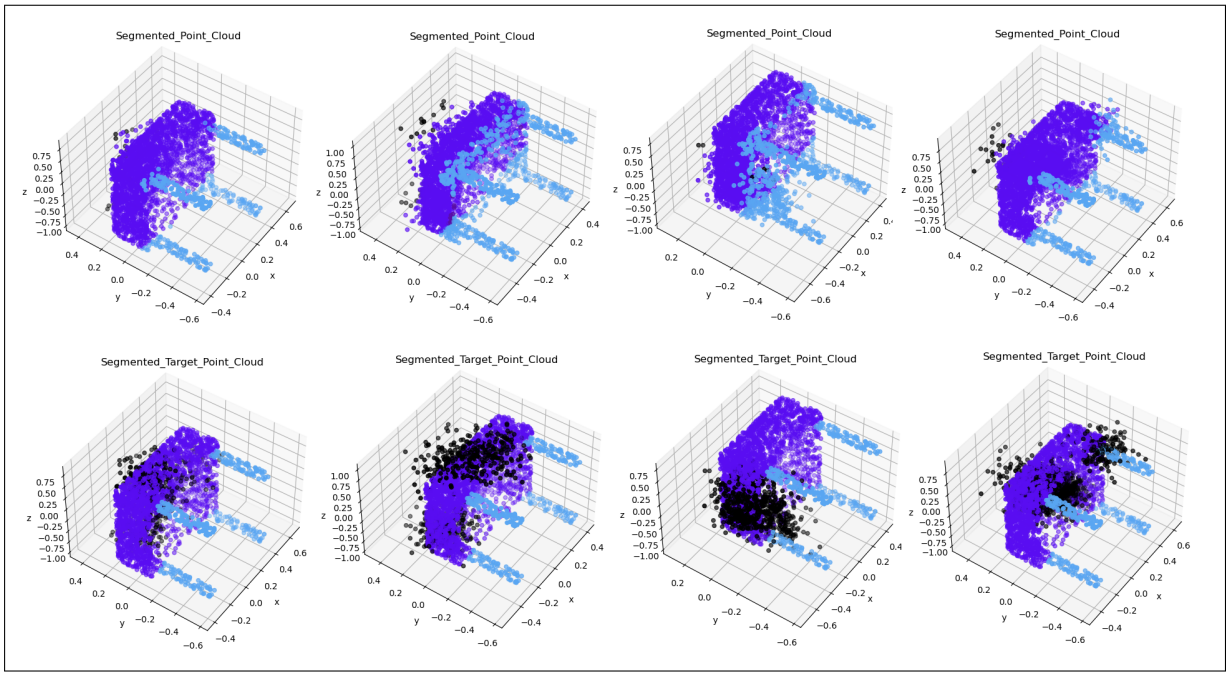


Figure 5.42: Result visualisation of Variant 2 on Add_Local noisy input.The point clouds in the first row show the output of the network and the point clouds at the bottom show the respective target point clouds. The visualisation of the result shows that this network configuration does not respond well to the Add_Local noise

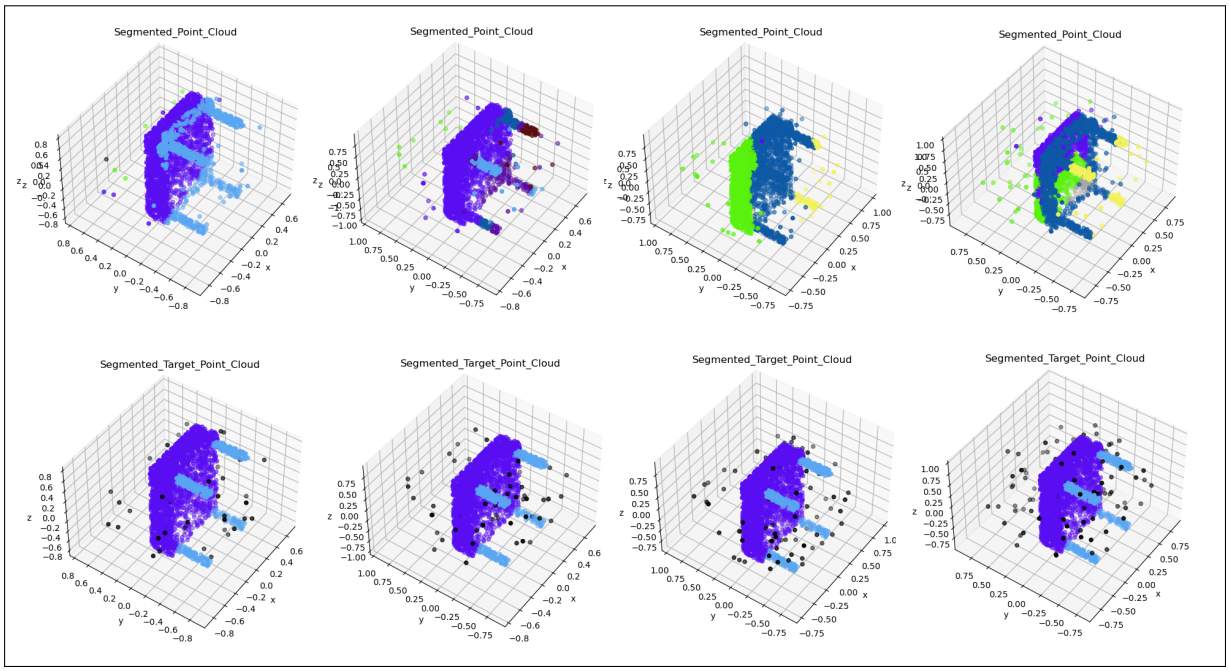


Figure 5.43: Result visualisation of Variant 2 on Add_Global noisy input.The point clouds in the first row show the output of the network and the point clouds at the bottom show the respective target point clouds. The visualisation of the result shows that this network configuration does not respond well to the Add_Global noise

From Figs. 5.37 ,5.38, 5.42 and 5.43 it can be seen that the performance of both variants of the 3D part segmentation network implemented with Transfer Learning to handle the additional class representing the local and global noisy points was not good enough.

Chapter 6

Conclusion

This project presents the implementation and in-depth analysis of the PointNet architecture to perform 3D object classification, 3D semantic segmentation and 3D part segmentation on 3D point clouds. It started with the implementation of the PointNet architecture for each of the above tasks. Then a detailed study of each of the networks was carried out by applying different techniques such as dropouts, batch normalisation, weight initialisations, etc. to the networks. These studies not only served the purpose of finding out the best performing configurations of the implemented network, but also highlighted the effects of different techniques on the behaviour of the model when applied in different possible combinations and positions in the network. It was found that the 3D object classification network performed best in the configuration with batch normalisation and Kaiming uniform weight initialisation with a dropout of p=0.3 in the classification network. This model performed almost as well as the PointNet classification network proposed by Charles et. al [6], suggesting that the PointNet classification network has reached a potential peak performance plateau in terms of achievable performance. However, the best performing configuration of the network for part/semantic segmentation network found in this project outperformed the model proposed by Charles et. al [6]. The best performing configuration for the segmentation task has batch normalisation and Kaming normal weight initialisation with two dropout layers in the segmentation network with dropout probability p = 0.4 and 0.3, respectively. Furthermore, the robustness of these models to various perturbations was tested by synthetically evolved perturbations to simulate the possible perturbations in point clouds, such as jitter, scale, rotation, brokenness as well as by direct scans from the real world. It was found that the PointNet architecture did not perform well as expected against noise generated by affine transformations of the input, such as rotation. Therefore, an ablation study was conducted in which the modular T-net (Input transform used for pose normalisation) was removed. This showed that the modular T-net had no significant effect on the performance of the model and could be considered a redundant part of the network. Visualisation of the results to understand the network behaviour was also covered in this project.

References

[1] S. Xu, J. Wang, W. Shou, T. Ngo, A.-M. Sadick, and X. Wang, "Computer vision techniques in construction: a critical review," *Archives of Computational Methods in Engineering*, vol. 28, no. 5, pp. 3383–3397, 2021.

[2] Y. Li, L. Ma, Z. Zhong, F. Liu, M. A. Chapman, D. Cao, and J. Li, "Deep learning for lidar point clouds in autonomous driving: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3412–3432, 2020.

[3] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang, "Deep learning on point clouds and its application: A survey," *Sensors*, vol. 19, no. 19, p. 4188, 2019.

[4] C. Yeo, S. Kim, H. Kim, S. Kim, and D. Mun, "Deep learning applications in an industrial process plant: repository of segmented point clouds for pipework components," *JMST Advances*, vol. 2, no. 1, pp. 15–24, 2020.

[5] A. Esteva, K. Chou, S. Yeung, N. Naik, A. Madani, A. Mottaghi, Y. Liu, E. Topol, J. Dean, and R. Socher, "Deep learning-enabled medical computer vision," *NPJ digital medicine*, vol. 4, no. 1, pp. 1–9, 2021.

[6] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[7] Y. He, H. Yu, X. Liu, Z. Yang, W. Sun, Y. Wang, Q. Fu, Y. Zou, and A. Mian, "Deep learning based 3d segmentation: A survey," *arXiv preprint arXiv:2103.05423*, 2021.

[8] H. Wu and X. Gu, "Max-pooling dropout for regularization of convolutional neural networks," in *International Conference on Neural Information Processing*. Springer, 2015, pp. 46–54.

[9] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[10] N. El-Ashmawy and A. Shaker, "Raster vs. point cloud lidar data classification," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, no. 7, p. 79, 2014.

[11] M. Saleh, S. Dehghani, B. Busam, N. Navab, and F. Tombari, "Graphite: Graph-induced feature extraction for point cloud registration," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 241–251.

[12] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, "On visual similarity based 3d model retrieval," in *Computer graphics forum*, vol. 22, no. 3. Wiley Online Library, 2003, pp. 223–232.

[13] M. M. Bronstein and I. Kokkinos, "Scale-invariant heat kernel signatures for non-rigid shape recognition," in *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE, 2010, pp. 1704–1711.

[14] P.-H. Hsu and Z.-Y. Zhuang, "Incorporating handcrafted features into deep learning for point cloud classification," *Remote Sensing*, vol. 12, no. 22, p. 3713, 2020.

[15] E. Ahmed, A. Saint, A. E. R. Shabayek, K. Cherenkova, R. Das, G. Gusev, D. Aouada, and B. Ottersten, "A survey on deep learning advances on different 3d data representations," *arXiv preprint arXiv:1808.01462*, 2018.

[16] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 922–928.

[17] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[18] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection." in *Robotics: Science and Systems*, vol. 1, no. 3. Rome, Italy, 2015, pp. 10–15.

[19] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas, "Fpnn: Field probing neural networks for 3d data," *Advances in neural information processing systems*, vol. 29, 2016.

[20] A. Muzahid, W. Wan, and L. Hou, "A new volumetric cnn for 3d object classification based on joint multiscale feature and subvolume supervised learning approaches," *Computational Intelligence and Neuroscience*, vol. 2020, 2020.

[21] M. Savva, F. Yu, H. Su, M. Aono, B. Chen, D. Cohen-Or, W. Deng, H. Su, S. Bai, X. Bai *et al.*, "Shrec16 track: largescale 3d shape retrieval from shapenet core55," in *Proceedings of the eurographics workshop on 3D object retrieval*, vol. 10, 2016.

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

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

[24] Y. Fang, J. Xie, G. Dai, M. Wang, F. Zhu, T. Xu, and E. Wong, "3d deep shape descriptor," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2319–2328.

[25] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

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

[27] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[28] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[29] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[30] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

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

[32] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[33] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-Semantic Data for Indoor Scene Understanding," *ArXiv e-prints*, Feb. 2017.

[34] M. A. Uy, Q.-H. Pham, B.-S. Hua, D. T. Nguyen, and S.-K. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *International Conference on Computer Vision (ICCV)*, 2019.

[35] J. Ren, L. Kong, L. Pan, and Z. Liu, "Benchmarking and analyzing point cloud robustness under corruptions," *arXiv:220x.xxxxx*, 2022.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[37] S. K. Kumar, "On weight initialization in deep neural networks," *arXiv preprint arXiv:1704.08863*, 2017.

[38] M. V. Narkhede, P. P. Bartakke, and M. S. Sutaone, "A review on weight initialization strategies for neural networks," *Artificial intelligence review*, vol. 55, no. 1, pp. 291–322, 2022.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[40] N. Chakraborty, A. Dan, A. Chakraborty, and S. Neogy, "Effect of dropout and batch normalization in siamese network for face recognition," in *International conference on innovative computing and communications*. Springer, 2020, pp. 21–37.

[41] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimedia Tools and Applications*, vol. 79, no. 19, pp. 12 777–12 815, 2020.

[42] X. Li, S. Chen, X. Hu, and J. Yang, "Understanding the disharmony between dropout and batch normalization by variance shift," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 2682–2690.

[43] G. Chen, P. Chen, Y. Shi, C.-Y. Hsieh, B. Liao, and S. Zhang, "Rethinking the usage of batch normalization and dropout in the training of deep neural networks," *arXiv preprint arXiv:1905.05928*, 2019.

[44] L. Chen, H. Wang, J. Zhao, D. Papailiopoulos, and P. Koutris, "The effect of network width on the performance of large-batch training," *Advances in Neural Information Processing Systems*, vol. 31, 2018.