3-D Object Classification and Segmentation Using PointNet

A PROJECT REPORT

Submitted by

BL.EN.U4EIE18006	Ms. B Raajitha Sai
BL.EN.U4EIE18014	Ms. Greeshma Namana
BL.EN.U4EIE18016	Ms. Harshini Kadiyala
BL.EN.U4ECE18170	Ms. Akshitha Veeravelli

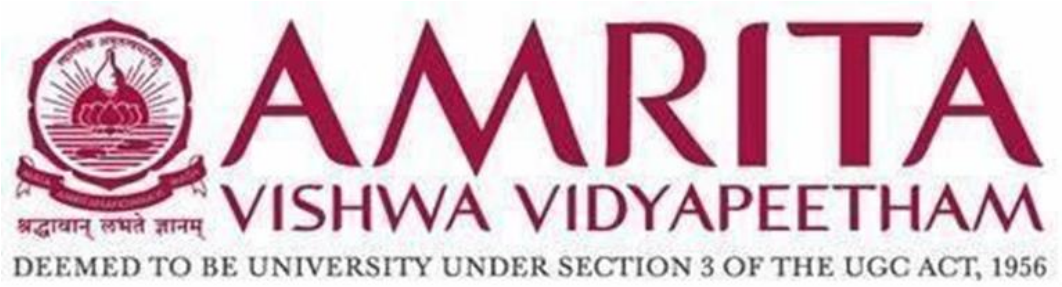
In Partial Fulfilment for the Award of the Degree

Of

BACHELOR OF TECHNOLOGY

IN

ECE, EIE



AMRITA SCHOOL OF ENGINEERING, BANGALORE
AMRITA VISHWA VIDYAPEETHAM
BANGALORE 560035

JUNE-2022

AMRITA VISHWA VIDYAPEETHAM

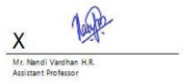
AMRITA SCHOOL OF ENGINEERING, BENGALURU – 560035

BONAFIDE CERTIFICATE

This is to certify that the project report titled **"3D objects classification and segmentation using PointNet"** submitted by

BL.EN.U4EIE18006	Ms. B Raajitha Sai
BL.EN.U4EIE18014	Ms. Greeshma Namana
BL.EN.U4EIE18016	Ms. Harshini Kadiyala
BL.EN.U4ECE18170	Ms. Akshitha Veeravelli

In partial fulfilment of the requirements for the award of the degree **Bachelor of Technology in Electronics and Communication Engineering** is a bonafide record of the work carried out under our guidance and supervision at Amrita School of Engineering, Bangalore



Mr. Nandi Vardhan H.R,
Assistant professor,
Department of ECE,
Amrita School of Engineering,
Bengaluru

Dr. Navin Kumar
Professor and chairperson,
Department of ECE,
Amrita School of Engineering,
Bengaluru.

This project report was evaluated by us on 17th June 2022.

Dr. Pooja Kenchetty P.
Asst. Professor (SG)

Ms. Jayasree M.Oli,
Asst. Professor (SG)

ACKNOWLEDGEMENT

We wish to express our sincere and heartfelt gratitude to those whose timely guidance has made us accomplish this project. First of all, our humble pranams at the holy feet of Amma who has been a driving force and a source of inspiration for all educational activities and extracurricular activities at Amrita School of Engineering, Bengaluru.

We would like to express our sincere gratitude to our director Br.Vishwamitra Chaitanya for the successful completion of the project. We would like to extend our sincere thanks to our principal Dr. Sriram Devanathan for inspiring us. We are highly indebted to Dr. Navin Kumar Chairman, Department of Electronics and Communications Engineering for motivating us towards the successful completion of the project.

We would like to express sincere thanks to our guide Mr. Nandi Vardhan Harlalli Rajendra Asst. Professor, Department of Electronics and Communications Engineering who has been very patient and helpful at all times and for pointing us in the right direction.

We would like to thank the panel members for their valuable suggestions for the improvement of this project. Finally, we would like to thank all the Project coordinators, Department of Electronics and Communications Engineering, Amrita School of Engineering, Bengaluru.

Also, we would like to thank our family, friends and colleagues whose constant support has helped us through tough times.

ABSTRACT

Point clouds are commonly used in 3D modelling, face recognition, and robotic missions to classify three-dimensional (3D) objects. However, because of the irregular data format of point clouds, processing raw point clouds directly is difficult for a typical convolutional network. In this project, we implement neural network that directly consumes point clouds, which well respects the permutation invariance of points in the input. This network, named PointNet, provides a unified architecture for applications ranging from object classification, part segmentation, to scene semantic parsing. Though simple, PointNet is highly efficient and effective.

The PointNet structure implemented in this project can analyse point cloud data without any pre-processing. To percept coordinate information of each point, a 2D convolutional layer is first implemented. To extract global features, multiple 2D convolutional layers and a global max pooling layer are implemented. Finally, fully connected layers predict object class labels based on the retrieved features. We tested the proposed PointNet structure on the ModelNet10 dataset. In comparison to existing approaches, the structure that we implemented achieved greater accuracy. Experiments using the ModelNet10 dataset show that the point number of point clouds does not have a major impact on the suggested PointNet structure.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Point Cloud

A point cloud [1] is the simplest form of 3D model data that can be generated by depth sensors like LIDARs and RGB-D cameras. It is a collection of individual points plotted in 3D space where each point contains several measurements including its coordinates along the (x,y,z) axis. The accurate 3D models from point clouds can be used in a variety of ways, including autonomous driving and augmented reality. Distinguishing vehicles from pedestrians, for example, is crucial for planning the course of an autonomous vehicle.[2]

It's a form of geometric data structure that's quite important. Most researchers convert such data to standard 3D voxel grids or collections of photos due to its irregular format. This makes data overly large and causes problems. However, due to the sparsity of data per item, object occlusions, and sensor noise, building strong classifiers with point cloud data is difficult. Many of these issues have been addressed using deep learning approaches, which learn strong feature representations straight from point cloud data.

1.2 PointNet

PointNet [5] is a deep neural network that directly consumes unordered point clouds and outputs class labels. The main feature of PointNet is its robustness towards input perturbation and corruption. It also takes care of the permutation invariance and transformation invariance of points in the point cloud. PointNet can implement object classification, part segmentation and semantic segmentation. The network's basic design is simple, as each point is treated identically and independently in the early stages. Each point is represented by only three coordinates in the basic setup (x, y, z). The use of a single symmetric function, max pooling, is the most important key to this method. The network effectively learns a set of optimization functions that selects informative points in the point cloud and encodes the reason for their selection. The network's last fully connected layers combine these learned optimal values into a global descriptor for the entire form (shape classification) or predict per point labels (shape segmentation) [1].

1.3 LIDAR

Due to the rapid advancement of 3D sensing technology, LIDAR [30] 3D scanners are becoming more widely available and affordable. This technique can accurately detect the distance between the sensor and nearby barriers while also delivering detailed geometric, shape, and scale data. Lidar sensors send out pulses of light to the surface of an object and measure how long it takes for each pulse to reflect back and hit the sensor. The distance between barriers in any given direction is determined by the timing information. The sensor data produce a Point Cloud (PCL), which is a collection of 3D points with corresponding reflectance values that describe the strength of the received pulses. Because active sensors do not require external illumination, more reliable detection can be obtained in severe weather and under extreme lighting circumstances (e.g., night-time or sun glare scenarios).

CHAPTER 2
MOTIVATION

The goal is to classify three-dimensional representation of images where we employ a data structure known as a point cloud, which is a collection of points that represent a three-dimensional shape or object. Our network, named PointNet, provides a unified architecture for applications ranging from object classification, part segmentation, to scene semantic parsing as seen in Figure 1. Our design for PointNet architecture is motivated by two challenges, the first challenge is to design a neural network for unordered inputs and the next challenge is to make our network more robust to input geometric transformations [1].

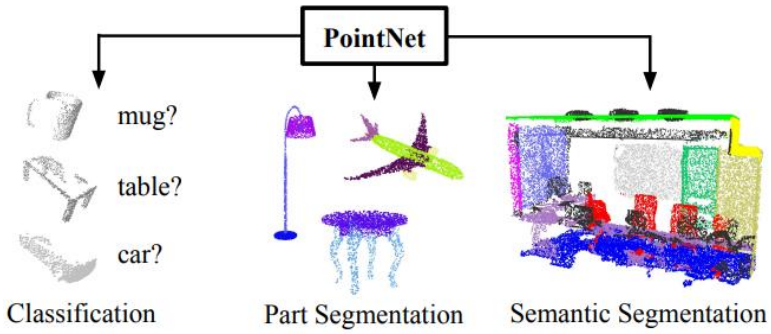


Figure 1 Applications of PointNet[1]

In this regard, PointNet has proven innovative, as it provides a learnable structured representation for point clouds. This procedure can be compared to "imaging" in that it produces a fixed dimensional output regardless of the number of samples or point order. This accomplishment has resulted in a variety of unique extensions and variants in point cloud object classification and segmentation [18].

CHAPTER 3
LITERATURE SURVEY

In order to select the best architecture that deals with Point clouds we compared the different ways in which point cloud data can be processed for implementation of classification and segmentation.
The different ways are:
3.1 Projection
3.2 Point net
3.3 Vox net

3.1 Projection

Dimensionality reduction methods, also known as projections, are often used to explore multidimensional data in machine learning, data science, and information visualization.

3.1.1 Advantages

Uses existing architectures for object identification on 2D pictures, with addition to regress 3D bounding boxes, to project point clouds onto a 2D image [9].

3.1.2 Limitations

Data loss is unavoidable when projecting point cloud data. It also avoids geographical information from being explicitly encoded in raw point cloud data [9].

3.2 VoxNet

A basic 3D CNN architecture that can be applied to create fast and accurate object class detectors for 3D point cloud data [31].

3.2.1 Advantages

Fully Convolutional Networks are used to predict object detections and generate a three-dimensional representation of the point cloud in a voxel structure. Shape information is explicitly encoded [31].

3.2.2 Limitations

Expensive 3D convolutions lengthen the time it takes to infer a model. The volumetric form is sparse and inefficient in terms of computing [31].

3.3 PointNet [1]

3.3.1 Advantages

To provide predictions on class and estimated bounding boxes, feed-forward networks consume raw 3D point clouds [9].

3.3.2 Limitations

Using the entire point cloud as input can lengthen the processing time. Creating region proposals using raw point inputs is difficult [9].

3.4 Comparison between PointNet and VoxNet

- On the basis of resilience to missing data in the input point cloud, we compare PointNet with VoxNet[32]. Both networks are trained using the same train test split with a total of 1024 points. We voxelize the point cloud to 32× 32 ×32 occupancy grids for VoxNet and enrich the training data with random rotation and jittering around the up-axis. At test time, a specific percentage of input points are dropped out at random. VoxNet's prediction is based on average scores from 12 different perspectives of a point cloud because it is sensitive to rotations. As shown in Figure 2, our PointNet is significantly more resistant to missing points. When half of the input points are missing, VoxNet's accuracy declines substantially, from 86.3 percent to 46.0 percent, a 40.3 percent difference, whereas PointNet's performance drops only 3.7 percent. This is explained by our PointNet's theoretical analysis and explanation: it learns to describe a form using a selection of essential points, making it exceptionally robust to missing data [1].

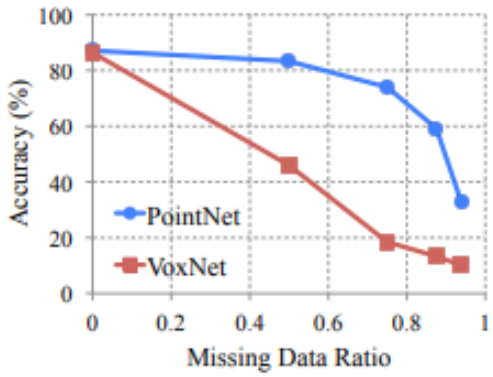


Figure 2. PointNet v.s. VoxNet on incomplete input data [1]

The overall classification accuracy on the ModelNet40 test set is the metric. Note that VoxNet averages 12 perspectives whereas PointNet only uses one view of the point cloud. PointNet appears to be considerably more robust when it comes to missing points.

- A VoxelNet and a Point Net are two new types of structures for point cloud 3D detection. This concept may be used to capture 3D data based on shape directly on sparse 3D points. The author has also presented a well-organised implementation of Voxel Net scarcity utilising a Voxel grid and parallel computation. By a wide margin, this KITTI presentation on the automobile detection test surpasses the other techniques.

When working on a more difficult task, such as 3D identification of bikes and pedestrians, net Point generates more useful data that indicates the most efficient technique of recognising 3D objects. Authors can build on this work by using the most up-to-date current ways to recognise extremely tiny 3D objects and improve the success rate. The Table 1 below illustrates the results of the PointNet and VoxelNet algorithms.

Net	Benchmark	Easy	Moderate	Hard
VoxelNet	Car (3D Detection), Car (Bird's Eye View)	78.48, 80.36	66.12, 80.27	58.74, 78.40
	Pedestrian (3D Detection), Pedestrian (Bird's Eye View)	40.49, 47.14	34.70, 41.75	32.52, 39.12
	Cyclist (3D Detection), Cyclist (Bird's Eye View)	62.23, 67.71	49.37, 55.77	44.38, 51.56
PointNet	Car (3D Detection), Car (Bird's Eye View)	82.21, 89.71	71.40, 85.01	63.20, 76.34
	Pedestrian (3D Detection), Pedestrian (Bird's Eye View)	52.22, 59.10	45.90, 51.23	41.24, 48.21
	Cyclist (3D Detection), Cyclist (Bird's Eye View)	72.97, 76.39	57.78, 62.97	51.40, 55.69

Table 1 The set of values obtained in detecting the 3D objects on KITTI by using VoxelNet and Point Net [16]

KITTI 3D data set is used in Table 1 which includes training images and testing images that have all the three classes of pedestrian, car and cyclist. For individual categories, the outputs of the detection are assessed based on three different levels: easy, moderate, and hard, which are determined according to the object size, occlusion state, and truncation level.

When used with small-scale point cloud segmentation, the identification of Net Point is excellent. In Voxelnet, scans are performed directly on raw cloud points, which are then manipulated on patterns. The KITTI car detection technology was used to arrive at the above result. To identify threats, the KITTI employs a bird's eye view. In this KITTI technique, we compare two different approaches: LiDAR and RGB-D. We conclude that PointNet is useful and effective when dealing with small scenarios, but Voxelnet is advantageous and effective when dealing with large scenarios.

3.5 Robustness Test for PointNet

Figure 3: Missing Data [1]

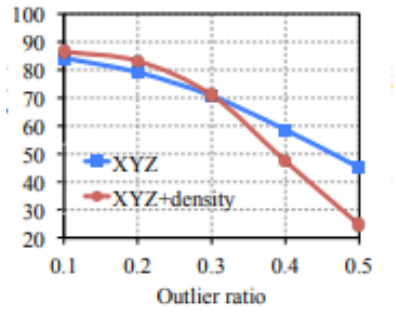


Figure 4: Outlier points [1]

Though PointNet is simple and effective, it is robust to various kinds of data corruptions and this can be observed from the above two graphs in Figure 3 and Figure 4.

Missing points: At test time when points are randomly dropped and classification accuracy is calculated, remarkably even after dropping 50% of the points, the accuracy only drops by 2.4% and 3.8% w.r.t. furthest and random input sampling which can be observed in Figure 3.

Outlier points: Pointnet is robust to outliers which can be observed in Figure 4.

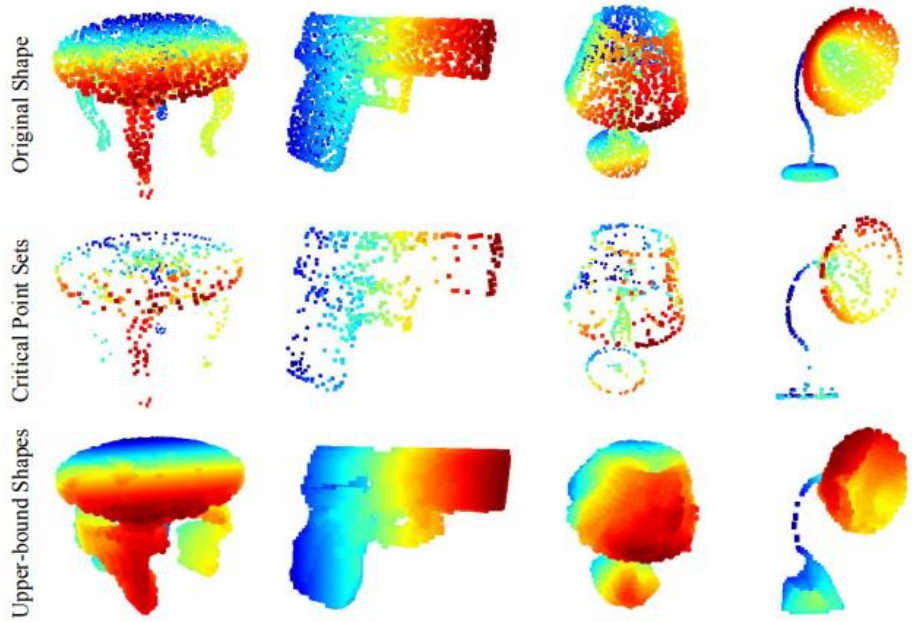


Figure 5 Critical point set and upper bound shape [1]

This robustness is achieved by pointnets due to max pooling that is being performed to extract global features. When max pooling is performed, only a set of points in the input contribute to the global features. This subset of contributing points is called the critical point set[1].

In Figure 5 if we see the critical point set, it can be observed that critical point sets are capturing the skeleton of the objects. The third row shows the set of points that won't affect the point cloud's global feature because this set has an upper bound shape. By definition any point that falls between critical set and upper bound set will result in the same global feature vector. Therefore, this explains why pointnet is so robust to various data corruptions such as missing data or point perturbations.

3.6 3D Object Part Segmentation Comparison:

Here we compare our segmentation version PointNet with two traditional methods:

- 3D Shape Co-segmentation with Graph Convolutional Networks [35]

- Framework for Region Annotation in 3D Shape Collections [36]

where, both make use of point-wise geometry features and shape correspondences, as well as the 3D CNN baseline that we implemented in this project. After comparison we see that the 3D fully convolutional network baseline executed in this project and the PointNet method achieved the state-of-the-art in mIoU.

The IoU is a number between 0 and 100, with a higher number indicating more precise segmentation. The mIoU is then the average value across all of the dataset's classes.

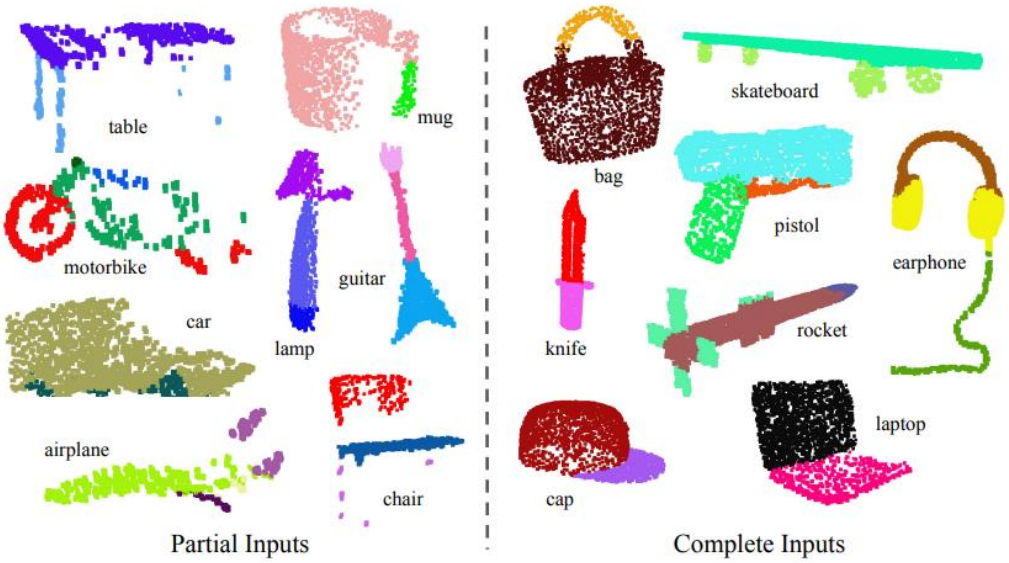


Figure 6 Qualitative results for part segmentation [1]

In the above Figure 6, we observe qualitative results on both complete and partial data. We observe that though partial data is fairly challenging, our predictions are reasonable.

3.7 Network Architecture and Training Details

3.7.1 3D Object Classification

The purpose of 3D object classification is to identify objects in a 3D point cloud or to label a distinct point cloud with a semantic object label. In robotics, virtual reality, and urban planning, it has a wide range of applications. Two datasets for 3D object categorization in the point cloud are now available: ModelNet40 and TU-Berlin.[18] Three characteristics of data-related classification problems include missing data, noise, and rotation invariance.

- Data loss: Scanned models are frequently blocked, resulting in data loss.
- Noise is present in all sensors. Point perturbations and outliers are examples of different types of noise. This means that even a point has that likelihood in a certain radius of the sampled location, or it could appear at random points in space (outliers).

- Invariance of rotation and translation points: Translational and rotational points must have no effect on classification.

When evaluating a classification model, accuracy is frequently used. In general, accuracy refers to the proportion of the time that the model correctly predicts the outcome. Formally, accuracy is defined as equation (1). As for the error rate, it is the misclassification rate and equal to one minus accuracy, as defined in equation (2).

Accuracy = (TP+TN)/(TP+TN+FP+FN) ……. (1)

Error rate = 1-accuracy ……. (2)

True positive, true negative, false-positive, and false-negative cases are represented as TP, TN, FP, and FN, respectively. PointNet, PointNet++, SO-Net, Dynamic Chart CNN, PointCNN, Kd-Network, 3DContextNet, Multi-Resolution Tree Network, SPLATNet, FoldingNet, and NeuralSampler are some of the categorization techniques available. ModelNet 10 and ModelNet 40 are the most extensively used datasets for accessing the performance of various models, despite the fact that there are many more accessible. The accuracies for class and instance, respectively, are class and instance accuracy.

3.7.2 3D Object Shape Segmentation

The technique of segmenting 3D point clouds into many homogenous regions is known as 3D point cloud segmentation. Points in the same region will have the same attributes. Because of the significant redundancy, inconsistent sample density, and lack of explicit organisation in point cloud data, segmentation is difficult.This topic has a wide range of robotics applications, including intelligent vehicles, autonomous mapping, and navigation. Various techniques and algorithms have been introduced by a number of authors. A crucial stage in the processing of 3D point clouds is segmenting them into foreground and background.The goal of the segmentation procedure is to group points with similar qualities into homogeneous regions given a set of point clouds[37]. These sparsely populated areas should be significant. The segmentation technique could be useful for analysing the scene in a variety of ways, including item recognition, classification, and extraction of features.

3.7.3 3D Object Part Segmentation

Part segmentation is a difficult task for fine-grained 3D recognition. The aim is to assign a part category label (e.g. chair leg, cup handle) to each point or face in a 3D scan or mesh model. Part segmentation is formulated as a per-point classification problem. Evaluation metric is mIoU on points.To calculate the shape's mIoU for each shape S in category C, compute the IoU between ground truth and prediction for each part type in category C. Count part IoU as 1 if the union of ground truth and forecast points is empty. Then, to get mIoU for that shape, we average IoUs for all part types in category C. We take the average of mIoUs for all shapes in the category to determine mIoU for the category.

3.7.4 PointNet Classification Network

A mini-PointNet is the first transformation network, which accepts a basic point cloud as input and regresses it to a three-dimensional matrix. On each point, it has a shared MLP(64, 128, 1024) network and two totally connected layers with output sizes 512 and 256. The output matrix is started with the identity matrix. All layers except the last incorporate ReLU and batch normalisation. The second transformation network produces a 64 × 64 matrix with the same layout as the first. Furthermore, the matrix is configured as an identity. A regularisation loss is applied to the softmax classification loss to get the matrix closer to orthogonality. On the last fully connected layer with a size of 256, we employ dropout with a keep ratio of 0.7 before class score prediction. The decay rate for batch normalisation starts at 0.5 and gradually increases to 0.99. The Adam optimizer is used with an initial learning rate of 0.001, a momentum of 0.9, and a batch size of 32. Every 20 epochs, the learning rate is doubled. ModelNet training takes 3-6 hours when using TensorFlow and a GTX1080 GPU. The comparison of the model with previous works as well as the baseline using MLP on traditional features extracted from point cloud in the below table. It is only possible with fully connected layers and max pooling, that net gains of a strong lead in inference speed and can be easily parallelized in CPU as well. Classification results on ModelNet40. Our net achieves state-of-the-art among deep nets on 3D input as shown in Figure 7.

	input	#views	accuracy avg. class	accuracy overall
SPH [11]	mesh	-	68.2	-
3DShapeNets [28]	volume	1	77.3	84.7
VoxNet [17]	volume	12	83.0	85.9
Subvolume [18]	volume	20	86.0	**89.2**
LFD [28]	image	10	75.5	-
MVCNN [23]	image	80	**90.1**	-
Ours baseline	point	-	72.6	77.4
Ours PointNet	point	1	86.2	**89.2**

Figure 7. Classification results on ModelNet40 [1]

3.7.5 PointNet Segmentation Network

The segmentation network is an extension of the PointNet classification system. For each point, the local point features (the output of the second transformation network) and the global feature (the result of the maximum pooling) are concatenated. For the segmentation network, there is no dropout. The classification network's training parameters are the same as the classification networks. In order to get the greatest performance for the task of shape part segmentation, a few changes to the fundamental segmentation network design are made, as show in Figure 8.

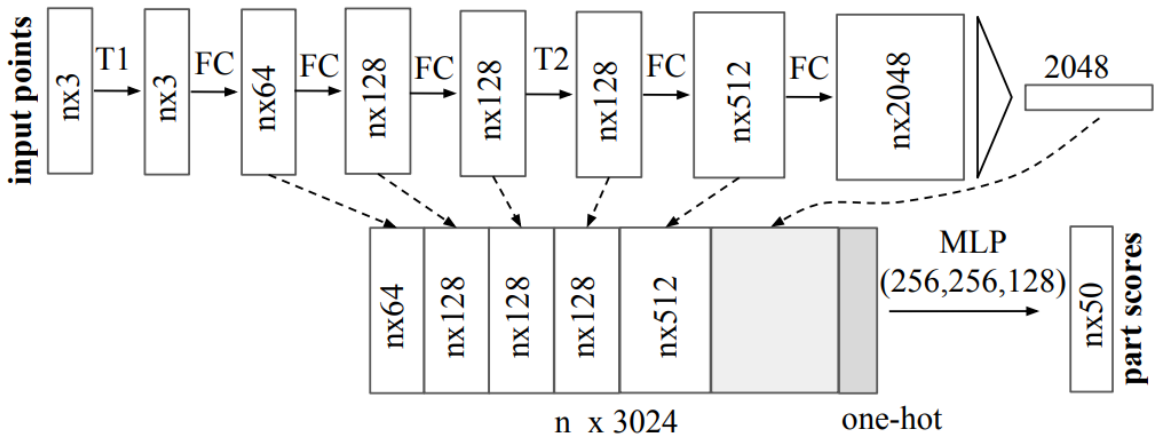


Figure 8. Network architecture for part segmentation [1]

We concatenate the output of the max pooling layer with a one-hot vector signifying the input's class. We additionally boost neurons in some levels and add skip connections to collect local point features from several layers and concatenate them to generate point feature input to the segmentation network. On the ShapeNet component dataset, training takes about six to twelve hours, while on the Stanford semantic parsing dataset, it takes about half a day.

3.8 Layers of architecture

In this project we are trying to implement a classification network which consists of various segments. The point cloud models are given as nx3 input points as shown in the Figure 9. The T-Net is employed twice and then a feature vector is formed. There are a series of MPLs in the layers of architecture. These MLPs are fully connected layers which means that the CNN model has activation functions, dense layers and max pooling. The global feature plays an important role in the structure since it comprises the highlight features of the inputs that are fed into the model.

A detailed description of each process involved in each layer is mentioned below.

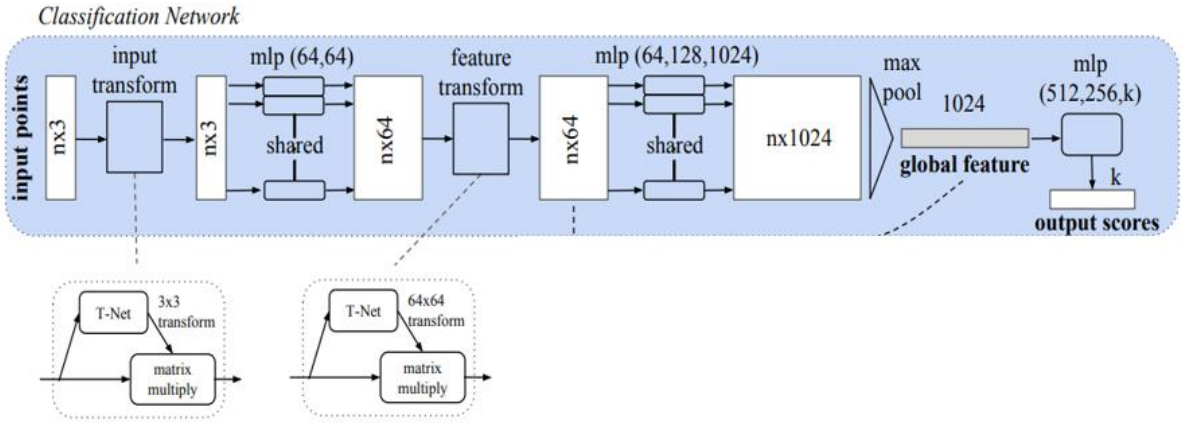


Figure 9 PointNet Classification Network Architecture [1]

3.8.1 Orthogonal Regularisation

Regularisation is the reduation of the complexity of the neural network by reducing the complexity of the weights, since complicated weights lead to overfitting. Orthogonal regularisation is an approach for regularising convolutional neural networks. Orthogonality is considered as a useful property of ConvNet filters. This is partly because multiplying an orthogonal matrix does not change the norm of the original matrix. This feature is useful in deep or recurrent networks where multiple matrix multiplication can cause signal fading or explosion. Orthogonal regularisation encourages weights to be orthogonal by trying to maintain orthogonality throughout training by pushing them to the nearest orthogonal manifold. the equation (4) given below is

$$L_{reg} = \|I - AA^T\|^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Where W is a filter bank, and I is the identity matrix.

3.8.2 Convolution

Convolution [20] is the technique of altering an image by applying a kernel to each pixel and its local neighbours over the entire picture in image processing. The kernel is a value matrix whose size and values affect the convolution process' transformation impact.

These are the steps in the Convolution Process. (1) It lays the Kernel Matrix over each pixel of the picture (ensuring that the whole Kernel is included inside the image) and multiplies each Kernel value with the pixel it is over. (2) After that, add the multiplied values together and return the result as the new value of the centre pixel. (3) This procedure is repeated throughout the picture.

The essential principle in Convolutional Neural Networks is convolution. Deep Neural Networks of the type Convolutional Neural Networks are a subset of Deep Neural Networks.

Convolutional Layer, Pooling Layer, and Fully-Connected Layer are the three layers that make up a CNN. A CNN uses a Kernel Matrix that it calibrates during training to apply convolution to its inputs at the Convolution layer. As a result, CNNs are suitable for matching features in pictures and object classification. A collection of learnable kernels make up the convolution layer parameters. Each kernel is a small matrix that covers the entire depth of the input volume. We convolve each kernel across the width and height of the input picture during the forward pass and compute dot products between the pixel values of the source and kernel at corresponding places during the forward pass.

3.8.3 T-Net

T-Net is a 3D point cloud-based transformer network based on PointNet. T-Net, on the other hand, produces an affine transformation that is not limited. This might result in unwanted shearing and scaling, resulting in the object losing its form. T-Net [15] is also only assessed on inputs with 2D rotations. Due to the invariance under geometric transformations; point cloud rotations should not alter the classification of the final machine learning model. Basic idea is to align the point cloud to a canonical space. This element is achieved by applying high fine transformation to input point coordinates. The transformation is predicted by a mini- point net which we call a T-net.

3.8.4 MLP (Multi-Layer Perceptron)

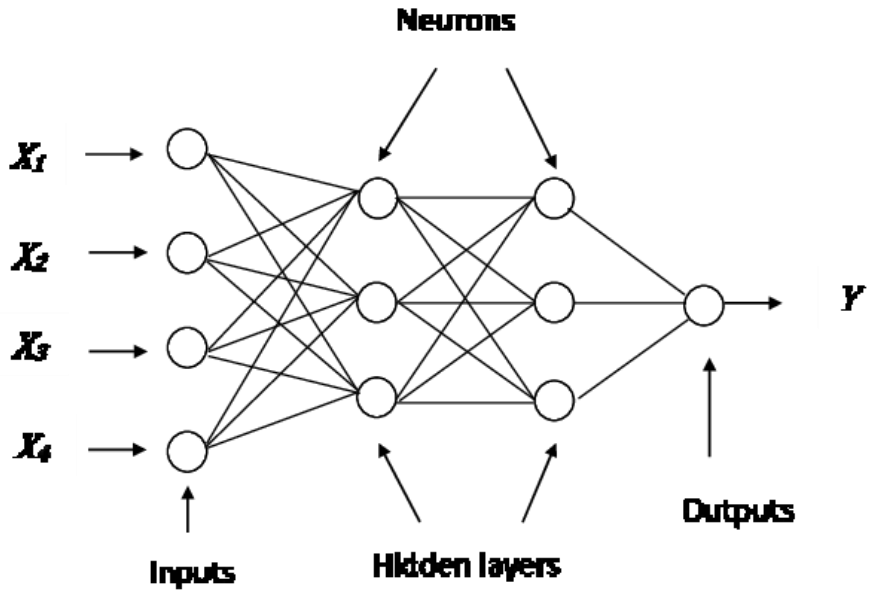


Figure 10. Types of layers in MLP [13]

The multi-layer perceptron [13] is a feedforward artificial neural network model that maps a set of input data onto a set of appropriate output. A MLP consists of multiple layers with each layer fully connected to the next one. The input layer, output layer, and hidden layer are the three types of layers represented in Figure 10. The input signal to be processed is received by the input layer. The output layer is responsible for tasks such as prediction and categorization. The real computational engine of the MLP is an arbitrary number of hidden

layers inserted between the input and output layers.There is no loop, the output of each neuron does not affect the neuron itself. This architecture is called feed-forward.The back propagation learning technique is used to train the neurons in the MLP. Pattern categorization, identification, prediction, and approximation are some of MLP's most common applications.

3.8.5 Activation Function

Activation function decides whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The purpose of the activation function is to introduce non-linearity into the output of a neuron. It's termed an activation function because it controls the intensity of the output signal and the threshold at which the neuron is activated. We know that neurons in a neural network work in accordance with their weight, bias, and activation function [21]. We would change the weights and biases of the neurons in a neural network based on the output error. Back-propagation is the term for this procedure. As the gradients are given simultaneously with the error to update the weights and biases, activation functions enable back-propagation.

3.8.6 Drop out

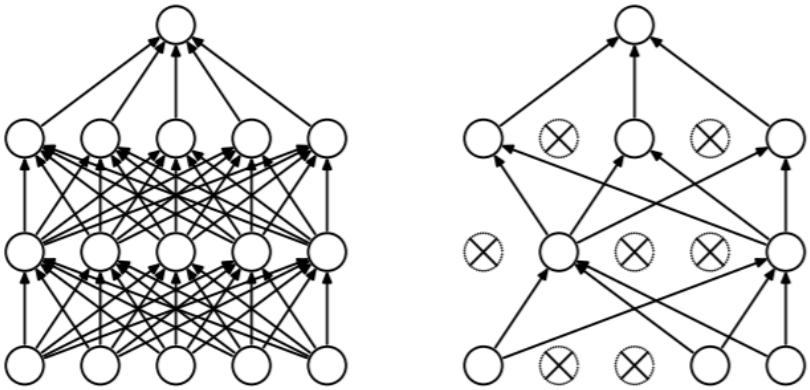


Figure 11 Dropout Neural Net Model [22]

Whenever we have an artificial neural network that is very deep, we tend to have many weights and biases which leads the artificial neural network to overfit.One of the ways to fix overfitting is dropout. Dropout [22] is a regularisation approach for simulating concurrent training of a large number of neural networks with varied topologies.To implement dropout we choose a drop out ratio and select a subset of features in every layer in the neural network as shown in Figure 11.

Most types of layers, including dense fully connected layers, convolutional layers, and recurrent layers like the long short-term memory network layer, may be used with it. Dropout can be used on any or all of the network's hidden layers, as well as the visible or input layer. It isn’t used on the output layer

3.8.7 Dense

Each neuron in a dense layer receives input from all neurons of its previous layer. Dense Layer performs a matrix-vector multiplication, and the values used in the matrix are parameters that can be trained and updated with the help of backpropagation [33]. Backpropagation can also update the values beneath the matrix, which are the learned parameters of the preceding layers. The row vector of the output from the preceding layers is identical to the column vector of the dense layer in matrix vector multiplication. Dense layer [24] is a deeply linked neural network. It is the most popular and widely used layer, and it operates by performing matrix vector multiplication to change the output dimension.

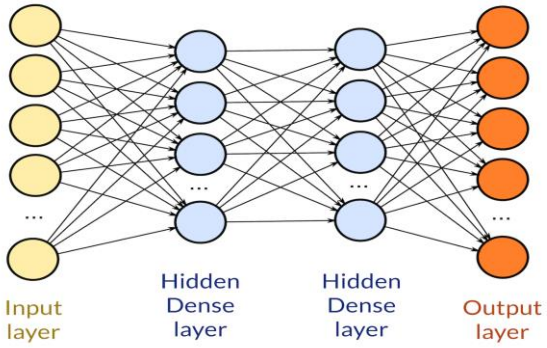


Figure 12 Dense layer in Neural network [24]

The Figure 12 represents the neural network with two hidden dense layers. If we consider the hidden layer as the dense layer the image can represent the neural network with multiple dense layers.

3.8.8 Max Pooling

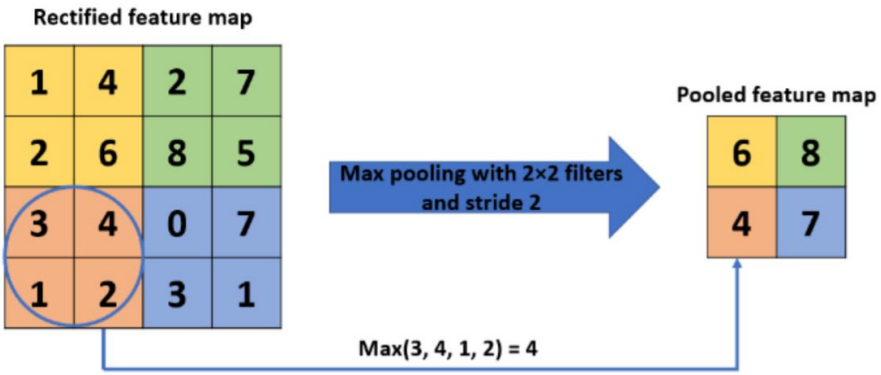


Figure 13 Example of Max-Pooling Operation [34]

A convolution technique in which the Kernel recovers the maximum value of the region it convolves is known as Max pooling as shown in Figure 13. When max pooling [23]

is used in a model, it simply tells the Convolutional Neural Network that only that information will be carried forward if it is the largest information available in terms of amplitude. When max pooling is used, it reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer. As we are convolving using a 2*2 Kernel, we are max-pooling on a 4*4 channel utilising a 2*2 kernels and a stride of 2. The channel has four numbers 8,3,4,7 if we look at the first 2*2 set on which the kernel is focused. Max-Pooling selects the highest value in the set, which is "8."

3.8.9 Data Augmentation

Data augmentation can help improve the performance and results of machine learning models by producing new and varied cases to train datasets. With the help of Data augmentation, we can transform a particular image to different images. By different images we mean that the output remains the same but some transformations are made like flipping, horizontal shifting, vertical shifting, rotation etc. When the dataset is large and sufficient, a machine learning model performs better and is more accurate. Data augmentation techniques, which produce deviations that the model could meet in the real world, can help make machine learning models more robust. In this project we have used data augmentation for improving the accuracy of the model and also to limit the risk of developing an overfitted model. This is an important part of the process because the data that is provided by the CAD models from the ModelNet10 data set is highly ideal and augmenting the data will train the model for the real-world scenarios.

3.8.10 Optimizer

Optimizers are the type of algorithms that are used to change the characteristics of the neural network model such as the weights and bias to reduce the losses. In this model we use ADAM (Adaptive Moment Estimation). This method has the fastest convergence. Adam is a best optimizer [25], it is more efficient than other available optimizers Adam is a combination of two other optimizers.
The weight is initialised using various initialization procedures and updated according to the update equation 3 given below, for each epoch.

$$w_{t+1} = w_t - m_t \left(\alpha/(v_t)^{1/2} + e\right) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

The update equation is the one that is used to update the weights in order to get the most accurate result. Using optimization procedures or algorithms known as optimizers, the best outcome can be achieved.

3.8.11 Epoch

An epoch [17] is a unit of time used to train a neural network using all of the training data for a single cycle. Epoch is a hyperparameter, and it defines how many times the learning algorithm works through the entire training process. From the Figure 14 it can be noted that

the process is a loop which runs depending on the number given by the machine learning model designer and within this loop the input values are iterated. Loss function is estimated from the target values by finding the derivative and then the weights of the neuron are updated.

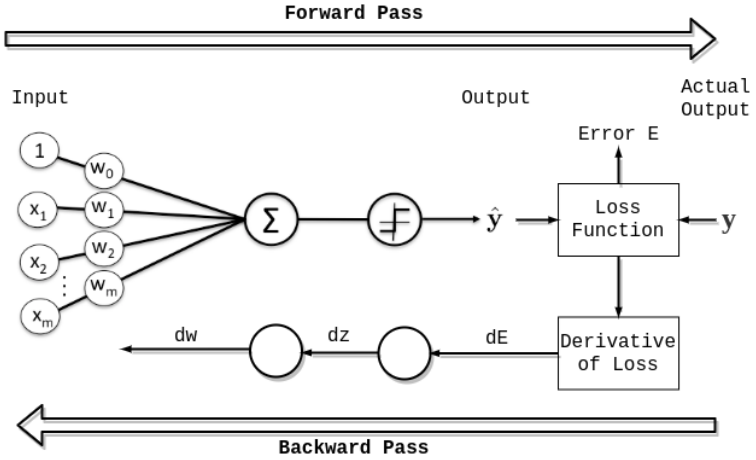


Figure 14. One pass (forward pass and backward pass) [17]

In an epoch, the entire dataset is passed forward and backward through the neural network only once. A forward pass and a backward pass together are counted as one pass: An epoch is made up of one or more batches, where we use a part of the dataset to train the neural network.

3.8.12 Bias & Variance

When calculating the output of a node, the inputs are multiplied by weights, and a bias value is added to the result. The bias value allows the activation function to be shifted to the left or right, to better fit the data. Bias has less data, so it won't give precise output. Since it predicts from available data itself, it is called biased. High biased means underfitting. Variance is where the data will fit exactly what is already there but won't predict new data. High variance means overfitting. To reduce overfitting, we do regularisation.

3.8.13 Normalisation

Normalisation [26] is a data pre-processing technique for converting numerical data to a common scale without changing the form of the data. When we feed data into a machine learning or deep learning system, we usually modify the numbers to a balanced scale. It is done in part to guarantee that our model can generalise correctly. Normalising the data to get a mean near to 0 is one of the best practises for training a Neural Network. In general, normalising data speeds up learning and leads to faster convergence. Also, because the tanh function appears to be strictly better, the sigmoid function is rarely utilised as an activation function in hidden units of Neural Networks. While it may not be immediately apparent, there

are some very comparable causes behind this. The logistic sigmoid and the tanh function are quite similar. The primary distinction is that the tanh function returns values ranging from -1 to 1, but the sigmoid function returns values ranging from 0 to 1, therefore they are always positive.

3.8.14 Metrics

Building machine learning models is based on the notion of constructive feedback. We create a model, gather data from measurements, make adjustments, and repeat until we attain the desired accuracy. The performance of a model is explained by its evaluation metrics [27]. The capacity of assessment metrics to recognise between model outcomes is a key feature. The choice of measure is entirely dependent on the type of model and the model's implementation strategy.

Given below are the types of metrics:

1. Classification Accuracy: It is a metric that gives a brief summary regarding the performance of a classification model. The number of true positive predictions is divided by the total number of predictions.
2. Logarithmic Loss: It is one of the important metrics which is based on probabilities. It is very useful for comparing the different machine learning models. For a given problem lower the logarithmic loss better the prediction.
3. Confusion Matrix: It is a table used to define the performance of a classifier model on a set of data for which the true values are known. Parameters such as Accuracy, Precision, Recall can be calculated from the matrix.
4. Area under Curve: It tells us how much the model is capable of distinguishing between classes. Higher the value of AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1.
5. F1 Score: conveys the balance between precision and recall.
6. Mean Absolute Error: MAE takes the average of absolute errors for a group of predictions or observations as a measurement of the magnitude of errors for the entire group. MAE helps users to change learning problems into optimization problems.
7. Mean Squared Error: The MSE is useful for ensuring that our trained model does not contain any outlier predictions with significant mistakes, as the squaring element of the function gives these errors more weight.

3.8.15 Learning rate:

The learning rate is a hyperparameter that governs how much the model changes each time the model weights are changed in response to the predicted error. A value that is too small can lead to a long, stopped training process, whereas a value that is too large can lead to learning a sub-optimal set of weights too rapidly or an unstable training process.

When constructing a neural network, the learning rate may be the most crucial hyperparameter. As a result, it's critical to understand how to examine the effects of the learning rate on model performance and to develop an intuition for the learning rate's dynamics on model behaviour. The step size, often known as the "learning rate," is the amount by which the weights are changed during training.

The learning rate is an adjustable hyperparameter that has a modest positive value, usually between 0.0 and 1.0, and is used in the training of neural networks. Given the smaller changes to the weights each update, lesser learning rates necessitate more training epochs, whereas greater learning rates necessitate fewer training epochs.

CHAPTER 4
PROBLEM STATEMENT

Recently it is observed that many emerging applications have been there that require a perception of 3d environments or require interacting with 3d objects. To serve those applications we see a strong need for deep learning technology that is tailored for 3d data. However, different from images that have the dominant representation as 2d pixel arrays, 3d has many popular representations. Point cloud is probably closest to 3d representation to raw sensor data. Point cloud is in canonical form which means that it is easy to convert the point cloud to another form and the other forms to point cloud.

However, there is barely any work on point cloud feature learning, most existing features are local to certain specific tasks. There have been few works that process point clouds with deep neural networks, however nearly all of them convert point clouds to other representations and then use existing neural network architecture. So, the problem we are facing is to see if we can achieve effective feature learning directly on point clouds. So, in our project, point nets will do end to end learning for scattered and unordered point data.

CHAPTER 5
METHODOLOGY

5.1 About PointNet

PointNet is the first deep learning system to use raw point clouds as input. It's been used to recognise and conceptually segment 3D objects. To deal with unstructured point cloud data, the invariance of the input point cloud arrangement was proposed. Transformation networks (T-Net) and symmetric functions are the two primary building pieces. The T-net is used to fit the model with the source and gather data at each location. It uses a spatial transformation network (STN) [29] to solve the rotation problem. STN in the computer vision community was proposed to deal with spatial invariance of objects. STN was introduced in the computer vision community to deal with object spatial invariance. STN learns the transformation matrix that is best for network classification or division by learning the point cloud's attitude information. It also makes use of STN twice. Adjusting the point cloud in the space is the initial input conversion.

The PointNet spins the item away from an angle that is more favourable to categorising or segmentation, such as to the front. The second feature transformation is to alter the point cloud at the feature level to align the retrieved 64-dimensional features. The symmetric function for analysing the point cloud is max pooling. It collects each point's high-dimensional local characteristics, which are learnt using a multi-layer perceptron (MLP)[13]. It may deal with the disorder issue as well as invariance under transformations. This is because max-pooling may extract the global characteristics of the whole point cloud. As shown in Figure 15 Max Pooling [23] achieves the best performance by a large winning margin

	accuracy
MLP (unsorted input)	24.2
MLP (sorted input)	45.0
LSTM	78.5
Attention sum	83.0
Average pooling	83.8
Max pooling	**87.1**

Figure 15. Three approaches to achieve order invariance [1]

5.2 Input Data

Raw point cloud data, which is generally obtained from a lidar or radar sensor, is fed into PointNet. Point clouds, unlike 2D pixel arrays and 3D voxel arrays, feature an unstructured form in which the data is essentially a collection of points [1]. Many practitioners discretize a point cloud by taking multi-view projections into 2D space or quantizing it to 3D voxels in order to utilise latest approaches based on 2D and 3D convolutions. In the process of converting 2D to 3D there are chances of completely altering the data of the original data set which leads the machine to learn wrong data and therefore predicts incorrect outputs. For the sake of simplicity, a point in a point cloud will be considered to be completely defined by its (x, y, z) coordinates. Other parameters, such as surface normal and intensity can be utilised whenever required.

5.3 Architecture

As PointNet uses raw point cloud data, it was important to create an architecture that strongly sticks to point sets inherent attributes [1]. Among these:

- Combination (Order): Since The point cloud data are versatile in nature, there are N! possibilities in a scan of N points. The data processing that follows for any order of the arguments the function value should be the same.
- Transformation Invariance: If the object undergoes specific transformations, such as rotation and translation, the segmentation and recognition outputs should remain unaltered.
- Interactions between neighbouring points: the interaction between neighbouring points frequently contains significant information (i.e., a single point should not be treated in isolation). Whereas classification requires simply the use of global features, segmentation requires the ability to employ both local and global point characteristics.

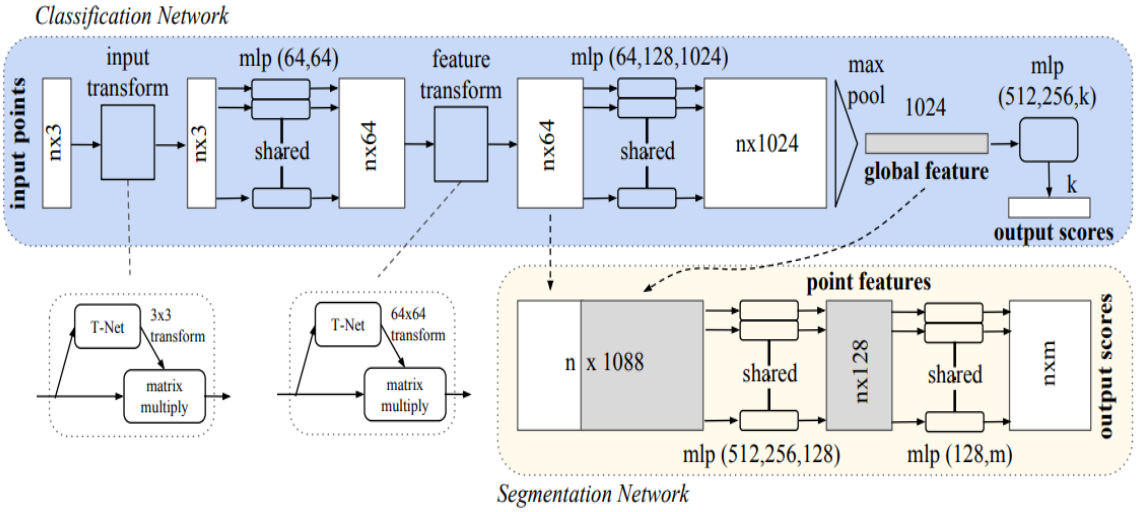


Figure 16. PointNet Architecture [1]

The classification network maps each of the n points in three dimensions (x, y, and z) to 64 dimensions using a shared multi-layer perceptron (MLP). It is important to note that each of the n points is served by a separate multi-layer perceptron. We repeat the method to transfer the n locations from 64 dimensions to 1024 dimensions as seen in Figure 16. In R1024, max pooling is used to construct a global feature vector from points in a greater embedding space. Finally, the global feature vector is mapped to k output classification scores using a three-layer fully-connected network.

The segmentation network is an extension to the classification net. It combines global and local characteristics and generates point scores. The letters mlp stand for multi-layer perceptron, while the numbers in brackets represent the layer sizes of the perceptron. With ReLU, Batchnorm is used for all layers. In a classification net, dropout layers are employed for the last mlp.

5.3.1 Permutation Invariance

Originally the point cloud data is unstructured data, point clouds are portrayed as numerical sets. There are N! permutations for a set of N data points.
We use a 2D array to represent a point cloud. Each point is a 2D vector simplest case a point is like x,y,z coordinate but we can also include RGB or laser intensity or service normals etc. since point cloud is a set its elements are order less, it means that with raw permutation the new array will represent the exact same set, therefore our neural network needs to be invariant to N! permutations.[1]It's also regarded as the commutative property for binary operators.
Common examples include:

- sum (a, b) = sum (b, a)
- average (a, b) = average (b, a)
- max (a, b) = max (b, a)

Once the n input points have been transferred to higher-dimensional space, we employ the symmetric function, those whose value given *n* arguments is the same regardless of the order of the arguments [6] The outcome is a global image vector that attempts to capture the n input points combined features as shown in Figure 17. The dimensionality of the global feature vector is, of course, related to the number of important features the machine wants to learn. The global feature vector is utilised for classification and segmentation, and it is combined with local point features. The global feature vector has an important role to play here because it contains the crisp features which define the object that needs to be predicted. Even if sometimes the input point cloud does not have the points it will not compromise the model's ability to predict correctly because the global features which have been extracted will have the outline of the object that needs to be predicted.

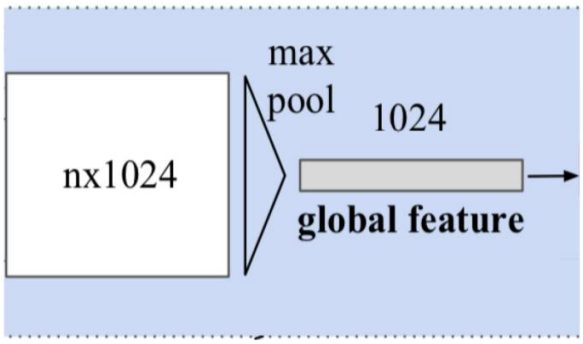


Figure 17. Max Pooling [1]

5.3.2 Transformation Invariance

The classification of an object should be invariant to certain geometric transformations (e.g., rotation). Here we use augmentation to perform the transformation invariance.

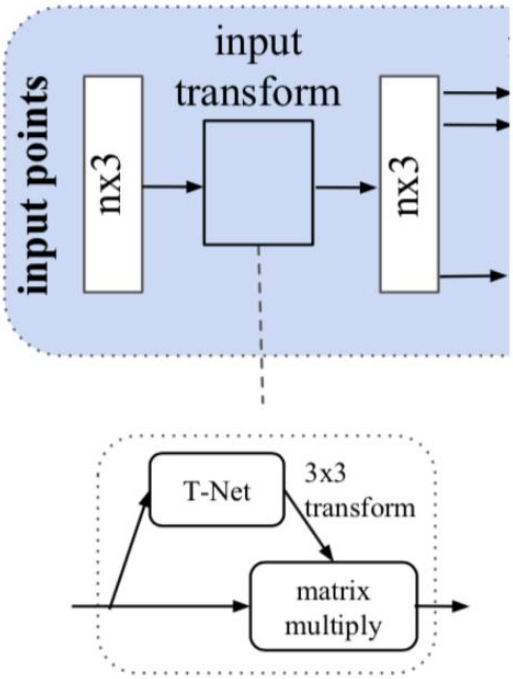


Figure 18. T-net (Transformer Network) [1]

T-Net [15] is a regression network charged with predicting a 3-by-3 transformation matrix that is then matrix multiplied with the n-by-3 input as shown in Figure 18.

The basic idea is to a line input point cloud canonical space, this alignment is achieved by applying a transformation to input point coordinates. The transformation is predicted by a mini point net which we call T-Net [15], which is n to n trained with the rest of the network. This is inherently similar to spatial transformer networks [29]. However, what's different is that for point clouds a geometric transformation is just matrix multiplication, it's very simple and efficient.

Pose normalisation [26] is extended to a 64-dimensional embedding space. Except for the dimensions of the trainable weights and biases, which become 256-by-4096 and 4096, respectively, resulting in a 64-by-64 transformation matrix, the corresponding T-Net is virtually identical to that in the preceding figure. As the increasing number of adjustable parameters increases the risk of overfitting and volatility during training, the loss function is modified to include a regularisation component. The regularisation term is depicted below, and it promotes the 64-by-64 transformation matrix in equation (4) to resemble an orthogonal transformation.

$$L_{reg} = \|I - AA^T\|^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Avoiding overfitting is one of the most important components of training the machine learning model. If the model is overfitting, it will have a low accuracy. This occurs because the model is attempting to capture the noise in the training dataset far too hard. The term "noise" refers to data points that do not accurately represent the true qualities of our data, but rather represent random chance. [1] The model becomes more flexible as a result of learning such data points, but at the risk of overfitting.

Understanding the phenomena of overfitting is aided by the idea of balancing bias and variance.

CHAPTER 6
EXPERIMENTAL SETUP

6.1 Experimental Setup of Classification Network

6.1.1 About Dataset:

The use of CNN for 3D model classification had been limited due to unavailability of large-scale training sets like ImageNet. The release of ModelNet[28], which has 662 object types, 127, 915 CAD models, and subsets like ModelNet40 and ModelNet10, which are used to benchmark classification algorithms for 3D models, partially solves this problem.

ModelNet10 is a subset of ModelNet40, and it contains 4,899 pre-aligned shapes from ten different categories. There are 3,991 shapes for training (80%) and 908 shapes for testing (20%) as shown in Figure 19. Object File Format is used for the CAD models (OFF). Princeton Vision Toolkit includes Matlab routines for reading and visualising OFF files (PVT).

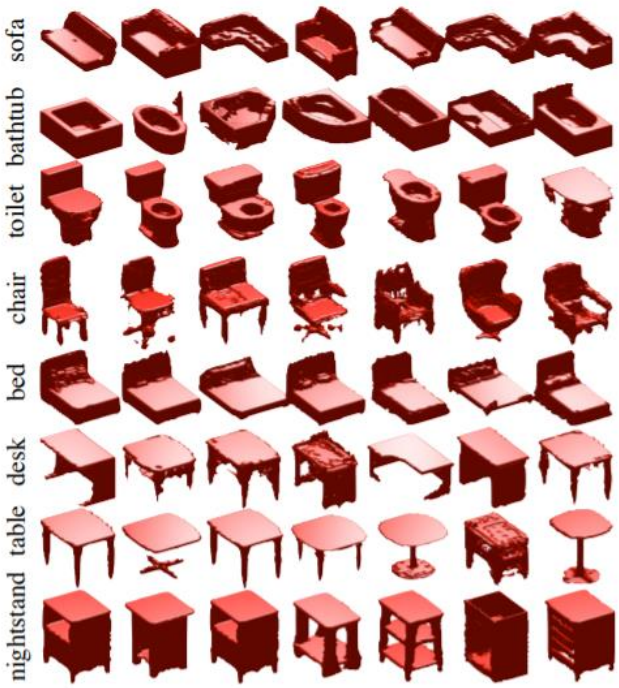


Figure 19 Example shapes generated by sampling our ModelNet 10 dataset for some categories [28]

6.1.2 Load Dataset

In this project, we have used the ModelNet10 model dataset, which is a scaled-down version of the vast ModelNet40 dataset with 10 classes, that include bathtub, desk, monitor, sofa, chair, dresser, table, bed, nightstand, toilet in which chair class is of 20%, sofa class is of 16% and rest comprise of 64%.
The data is downloaded and then we have to first sample points on the mesh surface and then implement random sampling in order to convert a mesh file to a point cloud. The number of points to be sampled can be varied and in this project, we sampled Toilet at 2048 locations as shown in Figure 20, sampled at 1024 locations as shown in Figure 21 and sampled at 500 locations as shown Figure 22. Here we can conclude that the number of samples can reproduce the original shape of the structure.

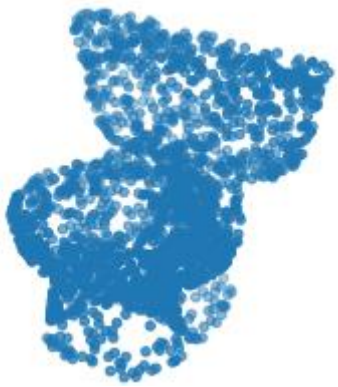


Figure 20. Sampled at 2048 locations

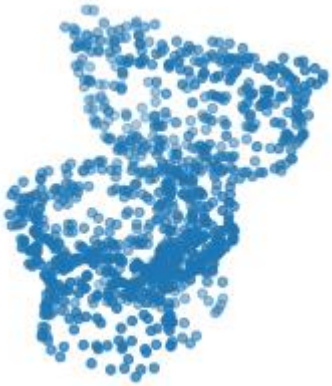


Figure 21. Sampled at 1024 locations

Figure 22. Sampled at 500 locations

To create a dataset object, we must first parse through the ModelNet data files. The initial data set is in the form of CAD models. These CAD models are turned into mesh as shown in Figure 23 where each mesh is turned into a point cloud and sampled, after this they are added to a standard Python list and converted to a numpy array. The current enumeration index value is likewise saved as the object label, and we use a dictionary to retrieve it later

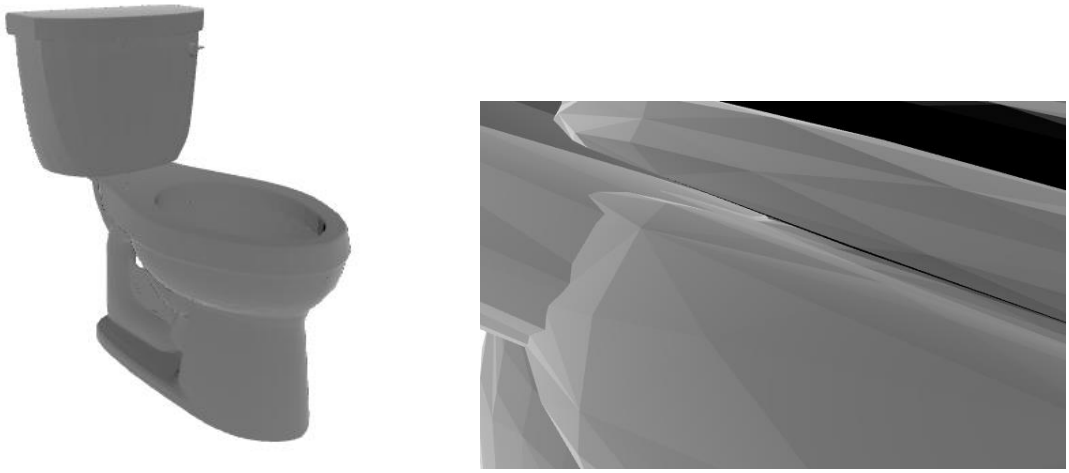


Figure 23 Mesh Model of Toilet

Our data can now be read into the dataset object. The shuffle buffer size was set to the total size of the dataset because the data was previously organised by class. When working with point cloud data, data augmentation is important. To jitter and shuffle the train dataset, we develop an augmentation function.

6.1.3 Build a Model

The most important part of each convolution and fully-connected layer (with exception for end layers) consists of Convolution / Dense which is followed by-> Batch Normalisation -> and finally followed by ReLU Activation.

PointNet consists of two important components. The transformer network (T-net) and the primary MLP network. The T-net aims to learn an affine transformation matrix by its own mini network. The T-net is used twice in the model. When the input data is sent to the T-net for the first time the input features (n, 3) are transformed into a canonical representation. The second is an affine transformation for feature space alignment (n, 3). We confine the transformation to be near to an orthogonal matrix (i.e., ||X*XT - I|| = 0) as in the original paper.

After that, we might have to construct a general function for creating T-net layers. The main network can then be implemented in the same way that the t-net mini models were dropped into the graph's layers. We use the smaller 10 class ModelNet dataset and duplicate the network architecture presented in the original research, but with half the number of weights at each layer.

6.1.4 Train model

Once the model is defined it can be trained like any other standard classification model and can be visualised

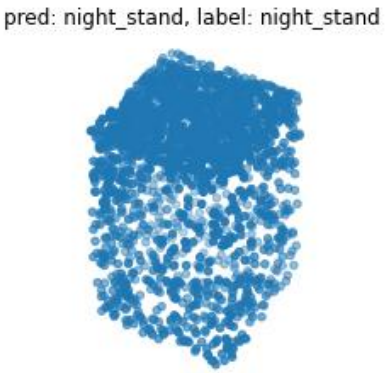
```
Model: "pointnet"
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_1 (InputLayer)         [(None, 2048, 3)]         0
_________________________________________________________________________________________________
conv1d (Conv1D)              (None, 2048, 32)          128         input_1[0][0]
_________________________________________________________________________________________________
batch_normalization (BatchNorma (None, 2048, 32)       128         conv1d[0][0]
_________________________________________________________________________________________________
activation (Activation)      (None, 2048, 32)          0           batch_normalization[0][0]
_________________________________________________________________________________________________
conv1d_1 (Conv1D)            (None, 2048, 64)          2112        activation[0][0]
_________________________________________________________________________________________________
batch_normalization_1 (BatchNor (None, 2048, 64)       256         conv1d_1[0][0]
_________________________________________________________________________________________________
activation_1 (Activation)    (None, 2048, 64)          0           batch_normalization_1[0][0]
_________________________________________________________________________________________________
conv1d_2 (Conv1D)            (None, 2048, 512)         33280       activation_1[0][0]
```

Figure 24: Summary of the model

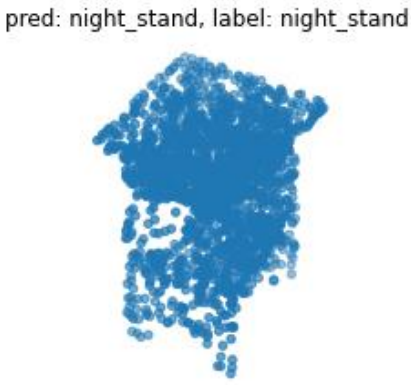
From the above Figure 24 summary we can see that initially the network is fed with the input layer with nx3 dimensionality having (x,y,z) coordinates as end points of the point cloud. The input points are made permutation invariant with the use of an augmentation function that jitter and shuffle the train dataset. Then, with the help of a convolution layer the dimensionality is increased to nx32, followed by batch normalisation and activation function. Next, the dimensionality is increased to nx64 and the process continues until the dimensionality is increased to nx512.Two T-nets are implemented with convolution layers and a global max pooling layer. Finally, global max pooling is implemented in order to extract the global vector which is updated to final 10 scores for 10 classes of the dataset that help in class labelling the output which is nothing but object classification.

6.1.5 Result

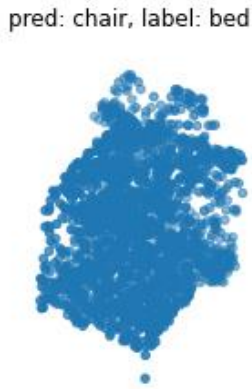
Output 1



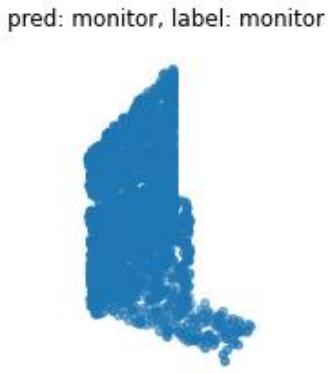
Output 2



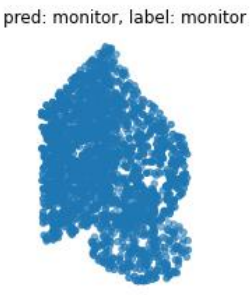
Output 3



Output 4



Output 5



Output 6

Output 7

Output 8

Output 9

Therefore, from our results it has been observed that our model has correctly predicted the classes for Output 1, Output 2, Output 3, Output 5, Output 6 and Output 7 and incorrectly predicted the classes for Output 4, Output 8 and Output 9.

6.1.6 Inference

```
Epoch 10/20
125/125 [==============================] - 442s 4s/step - loss: 2.0362 - sparse_categorical_accuracy: 0.7066 -
Epoch 11/20
125/125 [==============================] - 440s 4s/step - loss: 1.9554 - sparse_categorical_accuracy: 0.7294 -
Epoch 12/20
125/125 [==============================] - 440s 4s/step - loss: 1.8722 - sparse_categorical_accuracy: 0.7529 -
Epoch 13/20
125/125 [==============================] - 441s 4s/step - loss: 1.8607 - sparse_categorical_accuracy: 0.7590 -
Epoch 14/20
125/125 [==============================] - 445s 4s/step - loss: 1.8280 - sparse_categorical_accuracy: 0.7690 -
Epoch 15/20
125/125 [==============================] - 449s 4s/step - loss: 1.7892 - sparse_categorical_accuracy: 0.7795 -
Epoch 16/20
125/125 [==============================] - 463s 4s/step - loss: 1.7946 - sparse_categorical_accuracy: 0.7873 -
Epoch 17/20
125/125 [==============================] - 450s 4s/step - loss: 1.7493 - sparse_categorical_accuracy: 0.7905 -
Epoch 18/20
125/125 [==============================] - 447s 4s/step - loss: 1.6777 - sparse_categorical_accuracy: 0.8128 -
Epoch 19/20
125/125 [==============================] - 442s 4s/step - loss: 1.6979 - sparse_categorical_accuracy: 0.8118 -
Epoch 20/20
125/125 [==============================] - 440s 4s/step - loss: 1.6585 - sparse_categorical_accuracy: 0.8203 -
<keras.callbacks.History at 0x7f084663ca50>
```

Figure 25. Training of data points

Our model is approximately 82% accurate as shown in Figure 25, which is the reason why for some of the objects our model is predicting incorrectly.
Objects with similar shapes, particularly those in the bed, monitor, and sofa, were confused by this model. We assume this happened as a result of using global features for object categorization and data loss in the point cloud. We will consider implementing local features for categorization in the future to increase accuracy.

```
Epoch 1/5
125/125 [==============================] - 116s 875ms/step - loss: 3.4388 - sparse_categorical_accuracy: 0.3049
Epoch 2/5
125/125 [==============================] - 109s 875ms/step - loss: 3.0262 - sparse_categorical_accuracy: 0.3899
Epoch 3/5
125/125 [==============================] - 111s 892ms/step - loss: 2.7768 - sparse_categorical_accuracy: 0.4781
Epoch 4/5
125/125 [==============================] - 110s 881ms/step - loss: 2.5279 - sparse_categorical_accuracy: 0.5380
Epoch 5/5
125/125 [==============================] - 110s 883ms/step - loss: 2.4778 - sparse_categorical_accuracy: 0.5733
<keras.callbacks.History at 0x7f8d029377d0>
```

Figure 26. 500 samples

```
Epoch 1/5
125/125 [==============================] - 498s 4s/step - loss: 3.4517 - sparse_categorical_accuracy: 0.3200
Epoch 2/5
125/125 [==============================] - 476s 4s/step - loss: 2.8509 - sparse_categorical_accuracy: 0.4638
Epoch 3/5
125/125 [==============================] - 484s 4s/step - loss: 2.6416 - sparse_categorical_accuracy: 0.5322
Epoch 4/5
125/125 [==============================] - 493s 4s/step - loss: 2.3912 - sparse_categorical_accuracy: 0.6109
Epoch 5/5
125/125 [==============================] - 503s 4s/step - loss: 2.2319 - sparse_categorical_accuracy: 0.6522
<keras.callbacks.History at 0x7f7dfa279190>
```

Figure 27. 2048 samples

By comparing Figure 26 and Figure 27, we observe that for 500 samples and 5 epochs in Figure 26, the accuracy that we achieved is 57% whereas for 2048 samples and 5 epochs in Figure 27, the accuracy is around 65%, from which we can infer that, increase in the number of samples increases accuracy.

6.2 Experimental Setup for Segmentation Network:

6.2.1 About Dataset:

Dataset used for segmentation network: ShapeNet dataset

ShapeNet consists of several subsets:

ShapeNetCore

ShapeNetCore is a subset of the complete ShapeNet collection, made up of single clean 3D models with manually verified category and alignment annotations. It offers about 51,300 different 3D models that cover 55 different object categories. ShapeNetCore is a computer vision 3D benchmark dataset that covers all 12 object categories in PASCAL 3D+ as shown in Figure 28 and Table 2.

ShapeNetSem

ShapeNetSem is a smaller, more heavily annotated subset of the ShapeNet dataset, with 12,000 models spread across 270 categories. In addition to personally validated category labels and consistent alignments, these models are annotated with real-world dimensions, estimates of their material composition at the category level, and estimates of their overall volume and weight.

6.2.2 Loading the dataset

In segmentation network, we have used one of the 12 object categories of PASCAL 3D+, included as part of the ShapenetCore dataset.
For visualisation, we parse the dataset metadata to easily assign model categories to their appropriate categories and segmentation classes to colours.

Here, we train PointNet to segment the parts of a Chair model.

We give validation split as 0.2 which means 80% of the data is given for training and 20% of the data is given for testing. The number of sample points are 1024. The overall data is split into batches with each batch containing the size of 32. The initial learning rate is taken as 1e^-3. The learning rate tells us how fast the model learns. Learning rate can be increased or decreased based on the desired output. Epochs are the number of iterations that model performs. Accuracy of the model can be increased by increasing the epochs. The number of epochs is determined not only by learning rate but also by the amount of data involved.

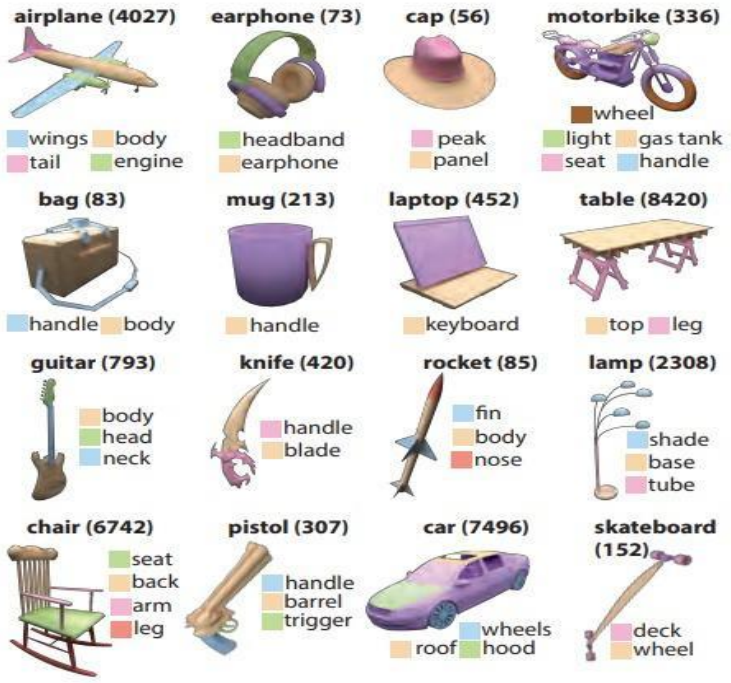


Figure 28 ShapeNet Data Set [36]

Category	N	L	Ann	Ver	T	FMF
Guitar	793	3	270	2380	2.8	7
Knife	420	2	149	828	1.5	4.6
Pistol	307	3	176	893	1.7	4.5
Lamp	2308	3	696	6783	7.4	7.6
Chair	6742	4	2112	26152	26.2	8.3
Table	8420	2	999	16022	14.6	9.1
Mug	213	1	33	207	0.4	3.9
Airplane	4027	4	2142	15757	22.6	5.8
Bag	83	2	18	160	0.2	7.7
Cap	56	2	18	109	0.2	5.4
Earphone	73	2	25	143	0.2	5
Laptop	452	1	18	444	0.2	14.8
Skateboard	152	2	24	304	0.3	8.6
Rocket	85	3	35	240	0.5	4.2
Motorbike	336	5	264	1568	2.9	4.5
Car	7496	3	2215	21635	27.8	6.5
Total	31963	42	9194	93625	109.6	7.1

Table 2 Number of Models for each Category

The Figure 28 contains the images of the models that were utilised in the ShapeNet data set. Each of the objects depicted in the figure is segmented according to its respective parts, for example from figure 28 the chair has seat, back, arm, and leg. Similarly, the rest of the objects have their labels shown.

6.2.3 Structuring the dataset

We generate the following in-memory data structures from the Chair point clouds and their labels:

- point clouds is a list of np.array objects that represent the x, y, and z coordinates of the point cloud data. The number of points in the point cloud is represented by axis 0, while the coordinates are shown by axis 1. all labels are a list that represents each coordinate label as a string (needed mainly for visualisation purposes).
- The format of test point clouds is the same as that of point clouds, but it doesn't have the point cloud labels.
- all labels is a list of np.array objects that correspond to the point clouds list and represent the point cloud labels for each coordinate.
- point cloud labels is a list of np.array objects that correspond to the point clouds list and represent the point cloud labels for each coordinate in one-hot encoded form.

```
point_clouds[454].shape: (2714, 3)
point_cloud_labels[454].shape: (2714, 5)
all_labels[454][0]: arm         point_cloud_labels[454][0]: [0. 1. 0. 0. 0.]

all_labels[454][1]: back        point_cloud_labels[454][1]: [0. 0. 1. 0. 0.]

all_labels[454][2]: arm         point_cloud_labels[454][2]: [0. 1. 0. 0. 0.]

all_labels[454][3]: back        point_cloud_labels[454][3]: [0. 0. 1. 0. 0.]

all_labels[454][4]: back        point_cloud_labels[454][4]: [0. 0. 1. 0. 0.]

point_clouds[5379].shape: (2714, 3)
point_cloud_labels[5379].shape: (2714, 5)
all_labels[5379][0]: seat       point_cloud_labels[5379][0]: [0. 0. 0. 1. 0.]

all_labels[5379][1]: none       point_cloud_labels[5379][1]: [0. 0. 0. 0. 1.]

all_labels[5379][2]: seat       point_cloud_labels[5379][2]: [0. 0. 0. 1. 0.]

all_labels[5379][3]: seat       point_cloud_labels[5379][3]: [0. 0. 0. 1. 0.]

all_labels[5379][4]: seat       point_cloud_labels[5379][4]: [0. 0. 0. 1. 0.]
```

Figure 29. One-hot encoding.

In Figure 29 we can see the labels of the corresponding point clouds which in turn are represented in the form of one-hot encoding.

6.2.4 Visualising data

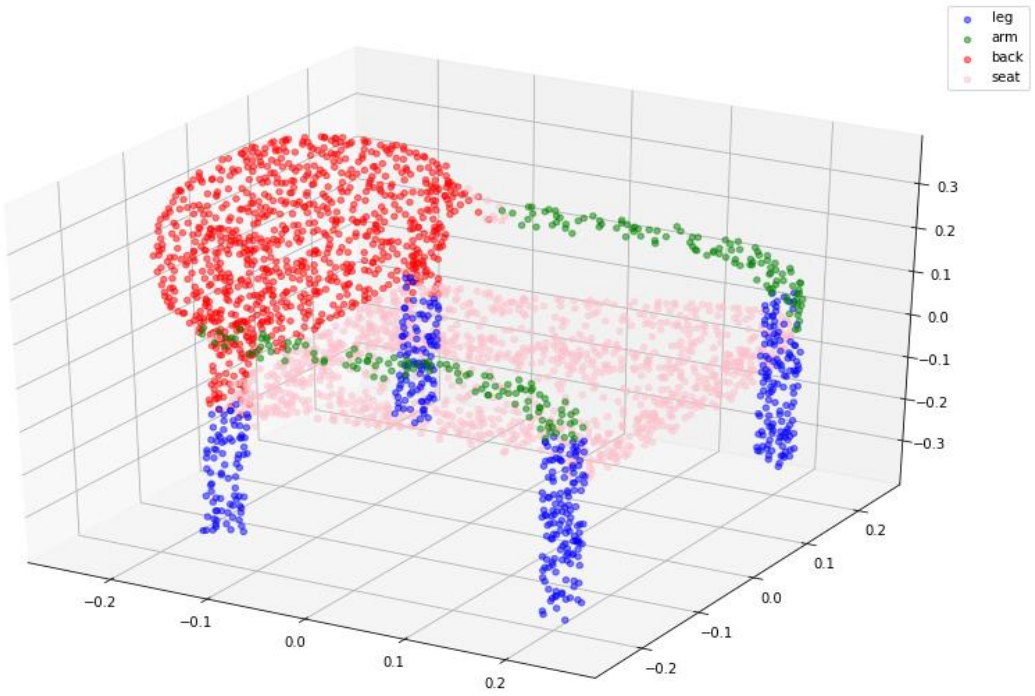


Figure 30 Visualising data of model chair

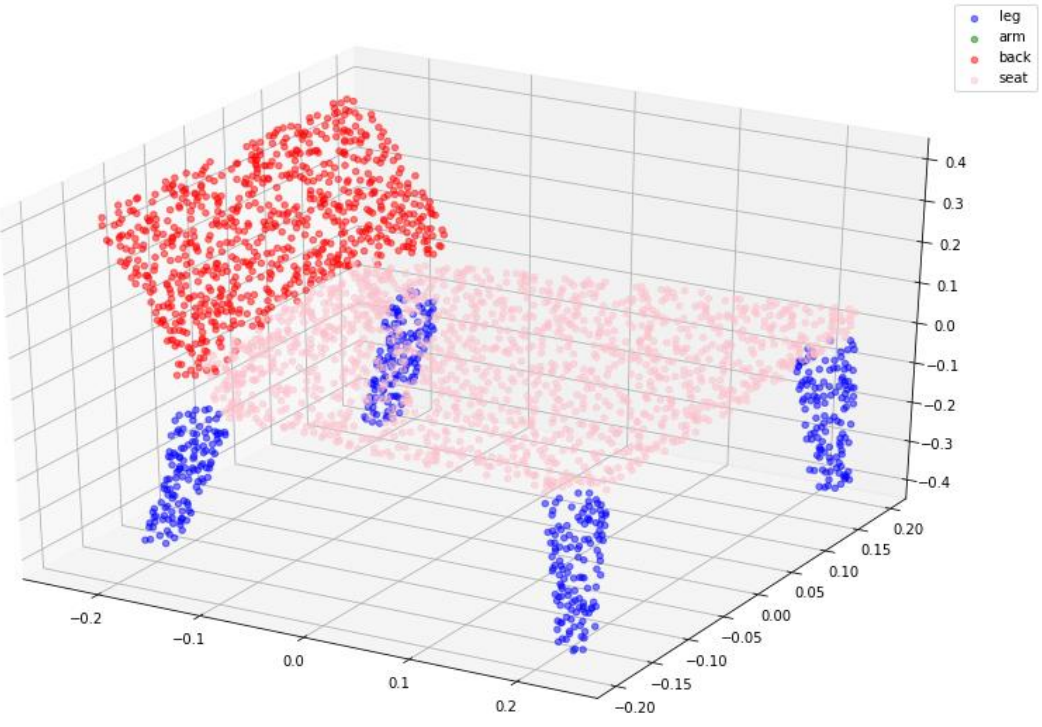


Figure 31 Visualising data of model chair

The above models (Figure 30 and Figure 31) are the point clouds of Chair before pre-processing. After that, we'll look at some examples from the in-memory arrays we just created: Now, let's visualise some of the point clouds along with their labels.

6.2.5 Pre-processing

It's important to note that each of the point clouds we imported has a different number of points, making it difficult to batch them together. To get around this, we take a predefined number of points from each point cloud and sample them at random. In order for the data to be scale-invariant, we also normalise the point clouds as shown in Figure 32 and Figure 33.
Let's visualise the sampled and normalised point clouds along with their corresponding labels.

Visualising the sampled and normalized point clouds along with their corresponding labels

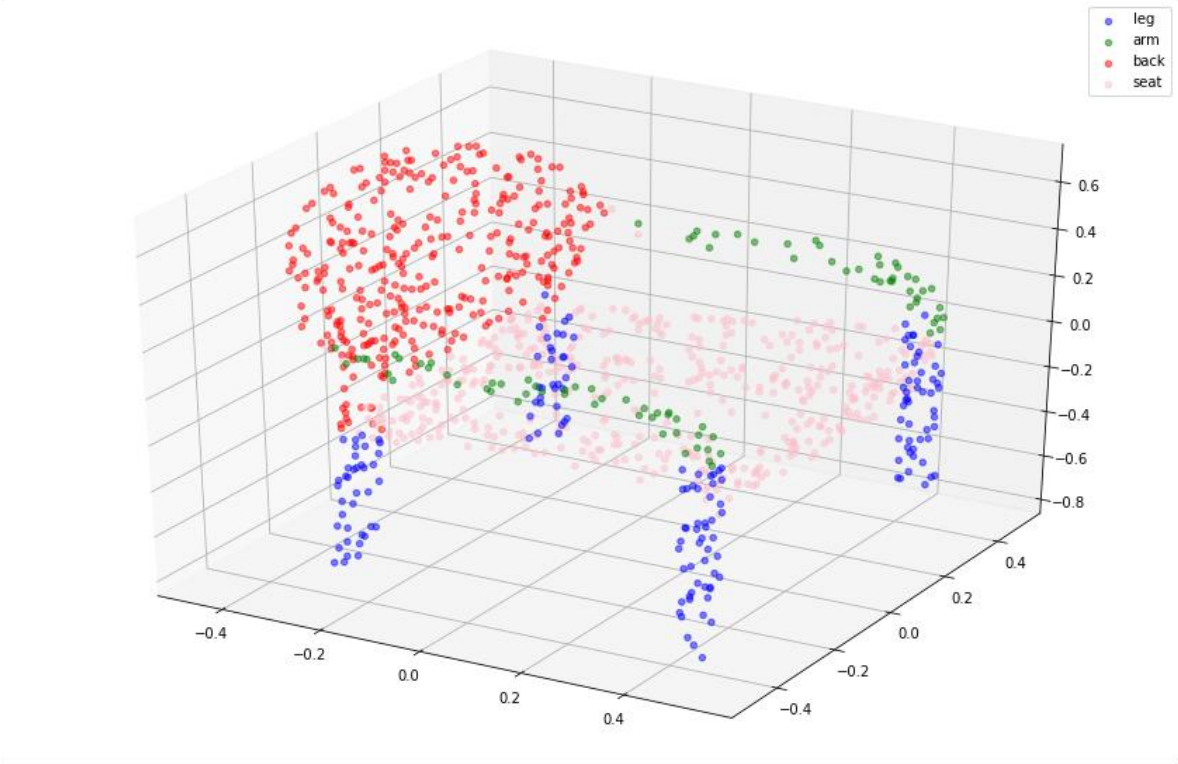


Figure 32 Here, the number of sample points is equal to 1024

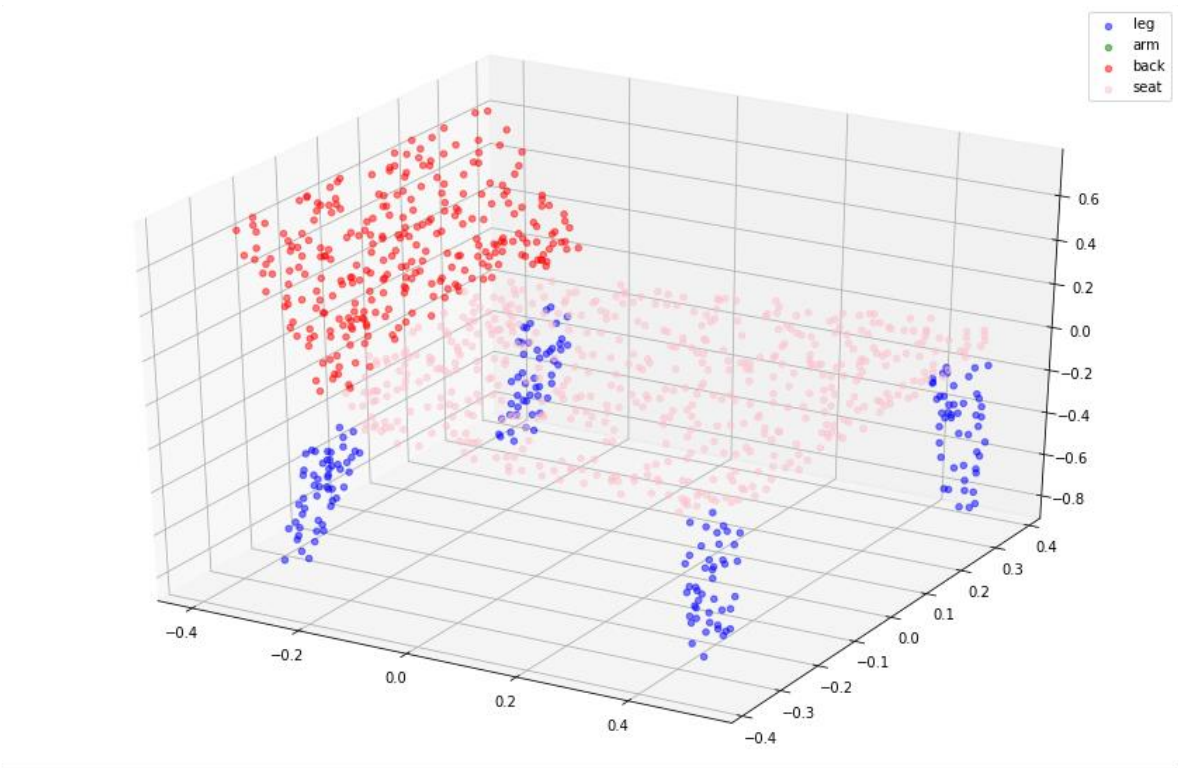


Figure 33 Here, the number of sample points is equal to 1024.

6.2.6 Creating TensorFlow datasets

For the training and validation data, we generate TensorFlow Dataset objects. We also augment the training point clouds by applying random jitter to them.

We can see that for chair model we have 6742 models for training and 739 models for testing.

6.2.7 PointNet model

We begin by implementing the fundamental blocks, such as the convolutional and multi-layer perceptron blocks.

To establish orthogonality in the feature space, we develop a regularizer. This is required to ensure that the modified features' magnitudes do not change too much. A transformation network is also implemented.
Finally, we piece the above blocks and implement the segmentation model

6.2.8 Training

The authors suggest employing a learning rate schedule that decays the starting learning rate by half every 20 epochs for the training. We'll use 15 epochs in this case. A learning rate schedule alters the learning rate throughout the learning process and is most commonly altered between epochs/iterations. This is primarily accomplished through the use of two parameters:

decay and momentum. The most popular learning rate schedules are time-based, step-based, and exponential.
Finally, we create a utility to help us execute the experiments and start model training as shown in Figure 34.

```
Epoch 50/60
151/151 [==============================] - 64s 427ms/step - loss: 4.3712 - accuracy: 0.8998 - val_loss: 4.4951 - val_accuracy: 0.8639
Epoch 51/60
151/151 [==============================] - 64s 427ms/step - loss: 4.4057 - accuracy: 0.8902 - val_loss: 4.5446 - val_accuracy: 0.8527
Epoch 52/60
151/151 [==============================] - 64s 426ms/step - loss: 4.3849 - accuracy: 0.8964 - val_loss: 4.5193 - val_accuracy: 0.8561
Epoch 53/60
151/151 [==============================] - 64s 426ms/step - loss: 4.3662 - accuracy: 0.9023 - val_loss: 4.5120 - val_accuracy: 0.8546
Epoch 54/60
151/151 [==============================] - 64s 427ms/step - loss: 4.3608 - accuracy: 0.9033 - val_loss: 6.3231 - val_accuracy: 0.8331
Epoch 55/60
151/151 [==============================] - 64s 427ms/step - loss: 4.3530 - accuracy: 0.9058 - val_loss: 4.5078 - val_accuracy: 0.8580
Epoch 56/60
151/151 [==============================] - 65s 427ms/step - loss: 4.3490 - accuracy: 0.9069 - val_loss: 4.5169 - val_accuracy: 0.8578
Epoch 57/60
151/151 [==============================] - 65s 427ms/step - loss: 4.3640 - accuracy: 0.9026 - val_loss: 4.5538 - val_accuracy: 0.8558
Epoch 58/60
151/151 [==============================] - 65s 427ms/step - loss: 4.3478 - accuracy: 0.9079 - val_loss: 4.5249 - val_accuracy: 0.8646
Epoch 59/60
151/151 [==============================] - 65s 427ms/step - loss: 4.3414 - accuracy: 0.9098 - val_loss: 4.7202 - val_accuracy: 0.8349
Epoch 60/60
151/151 [==============================] - 64s 427ms/step - loss: 4.3604 - accuracy: 0.9040 - val_loss: 4.5380 - val_accuracy: 0.8622
```

Fig 34 Training of Data Set

6.2.9 Visualize the training landscape

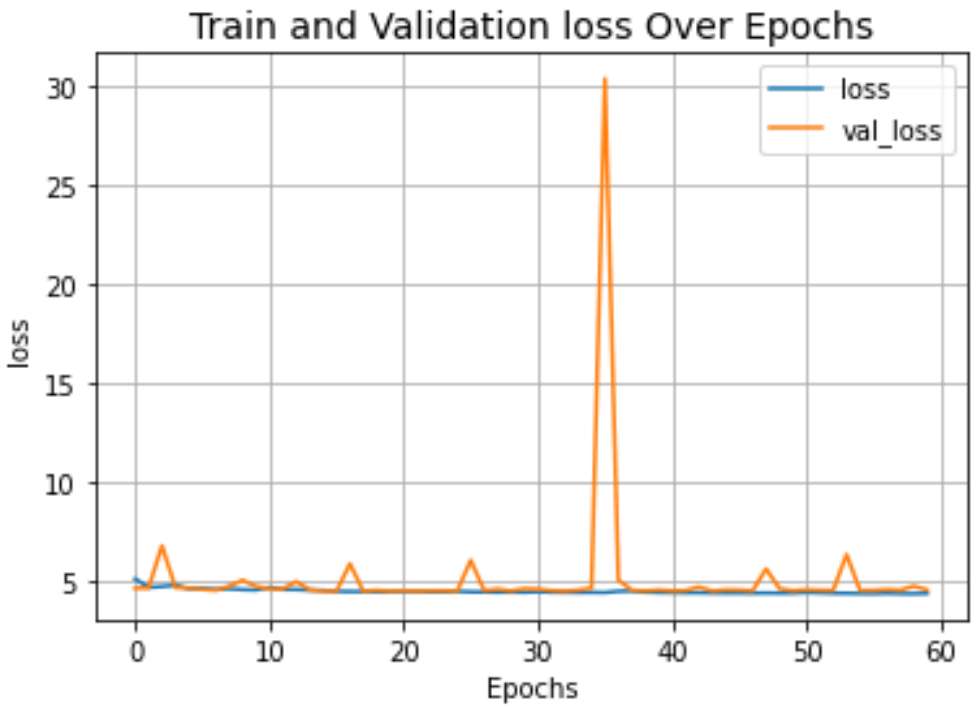


Figure 35 Train and Validation loss Over Epochs

The above Figure 35 shows the variation of loss with respect to the epochs. After 60 epochs the validation loss is almost equal to the loss.

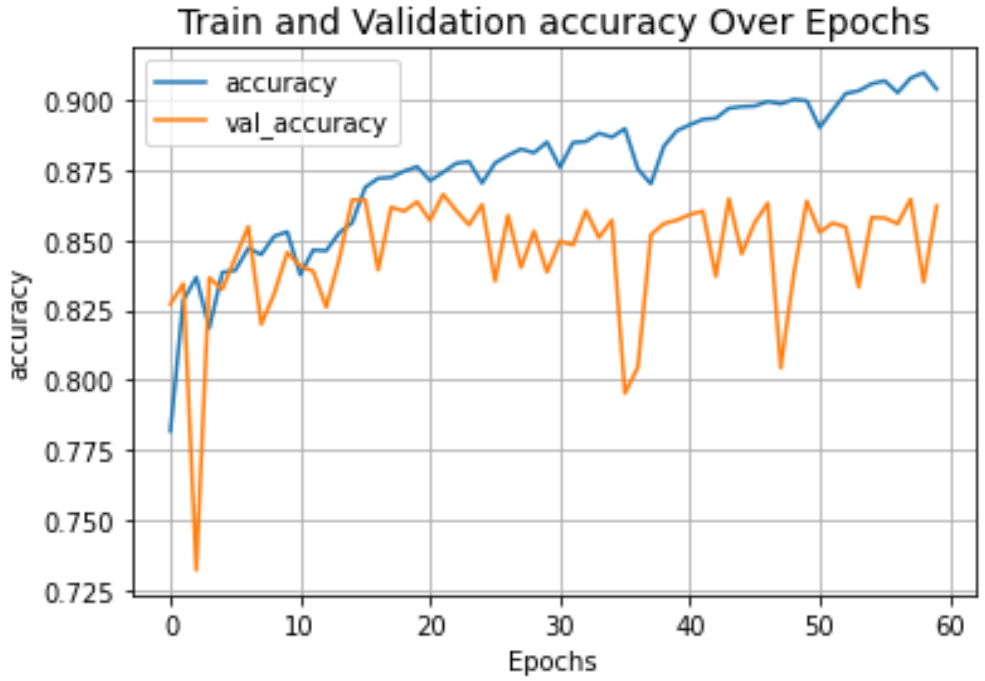


Figure 36 Train and Validation accuracy Over Epochs

The above Figure 36 shows the variation of accuracy with respect to the epochs.

6.2.10 Validation prediction

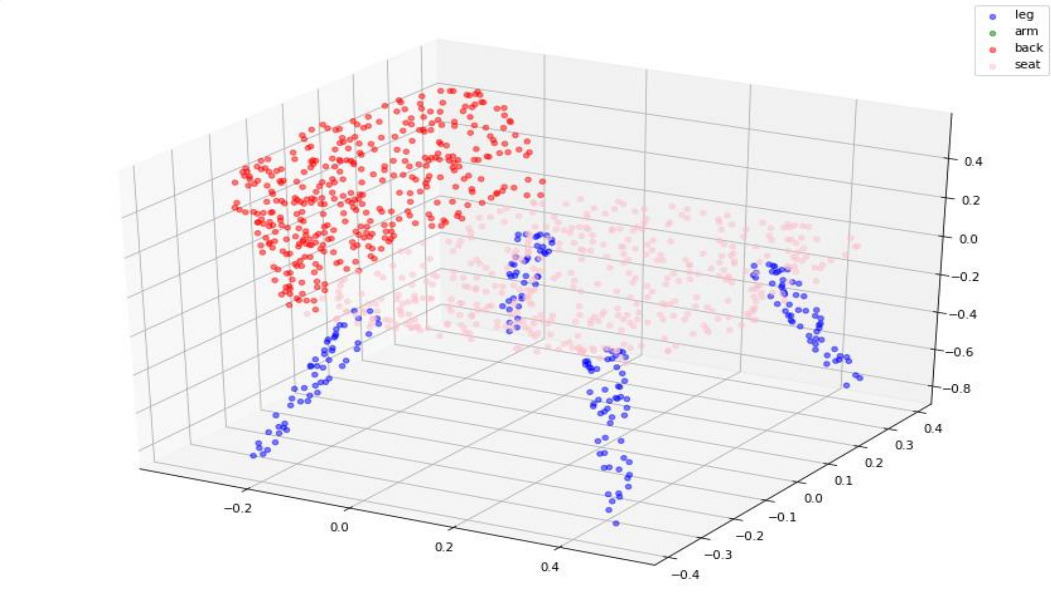


Figure 37 Ground Truth

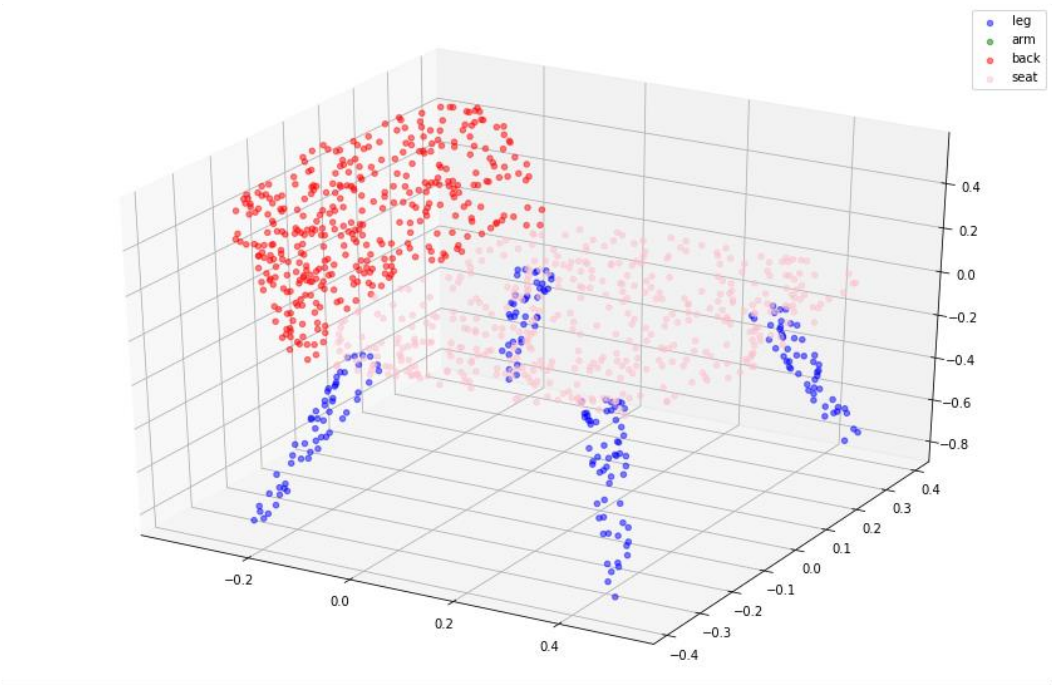


Figure 38 Predicted Labels

From Figure 37 and Figure 38 we can observe that with 86% accuracy the predicted model gives almost the correct prediction of labels for most of the points.

Similarly, we've tried for other models of Chair as shown in Figure 39, Figure 40, Figure 41 and Figure 42.

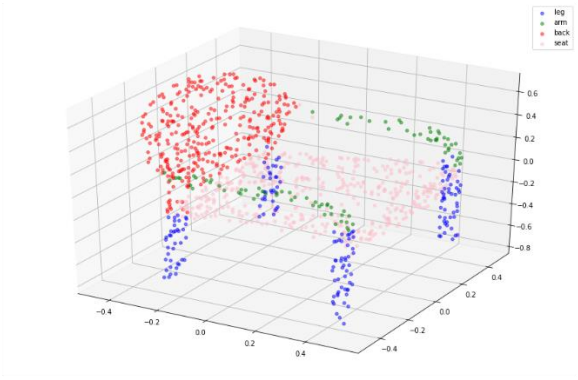


Figure 39 Ground Truth

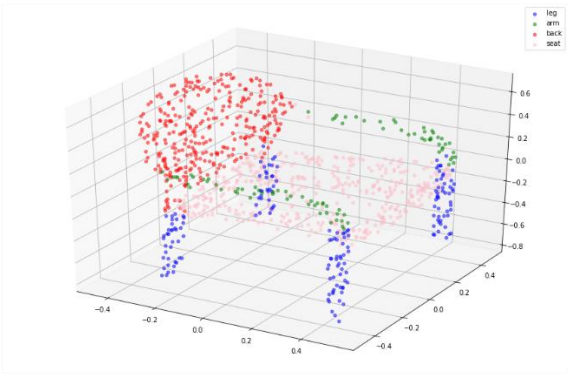


Figure 40 Predicted Labels

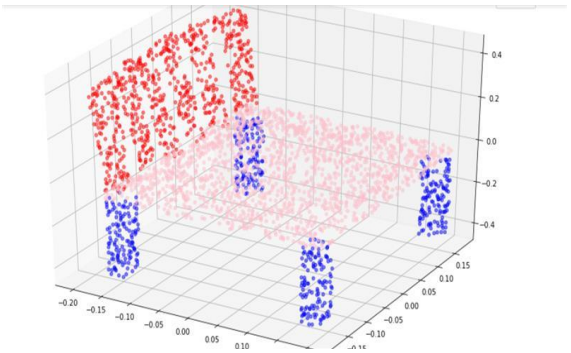


Figure 41 Ground Truth

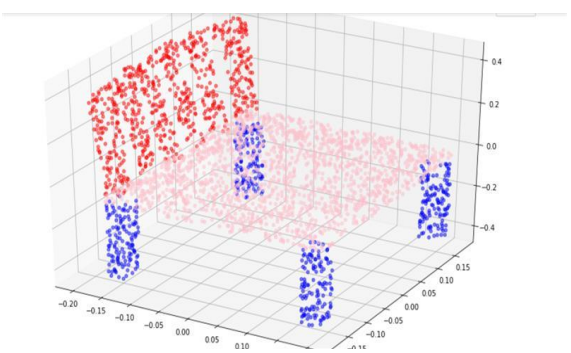


Figure 42 Predicted Labels

Similarly for Airplane model the ground truth and validation prediction are shown in Figure 43, Figure 44.

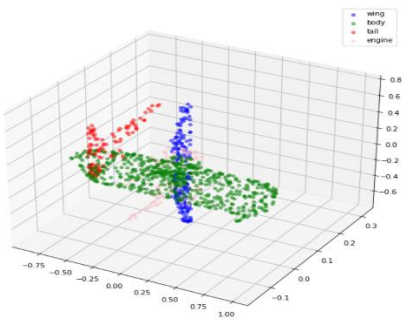


Figure 43 Ground Truth

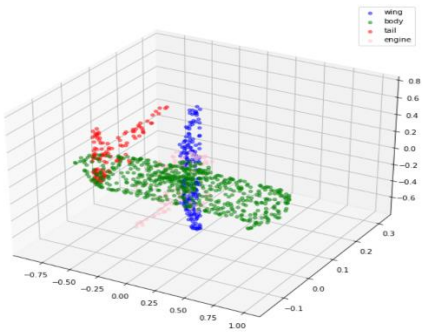


Figure 44 Predicted Labels

For another model of Airplane as shown below in Figure 45 and Figure 46.

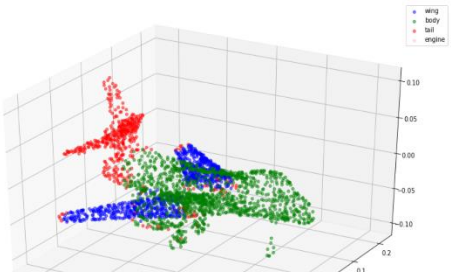


Figure 45 Ground Truth

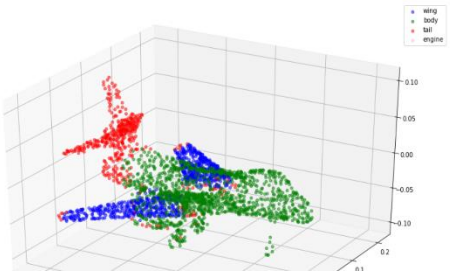


Figure 46 Predicted Labels

Ground Truth and Validation prediction for Table model as shown in Figure 47 and Figure 48.

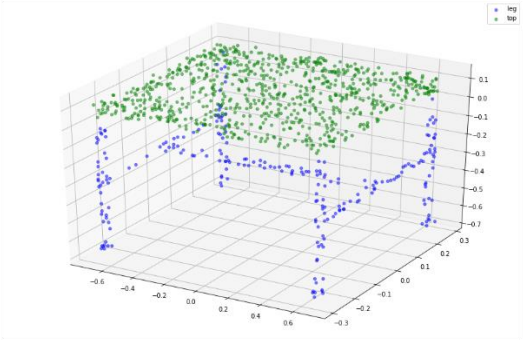


Figure 47 Ground Truth

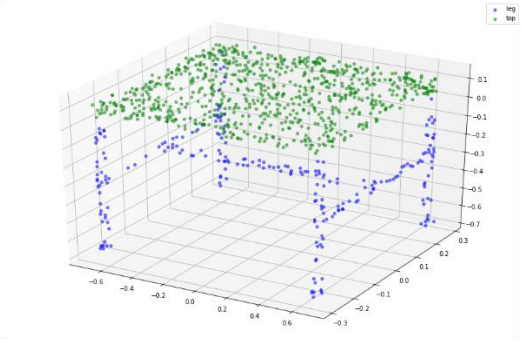


Figure 48 Predicted Labels

For another model of Table as shown below in Figure 49 and Figure 50.

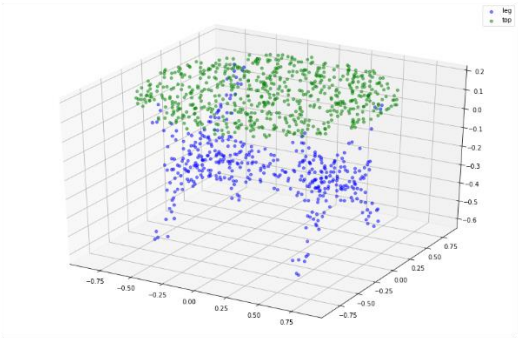


Figure 49 Ground Truth

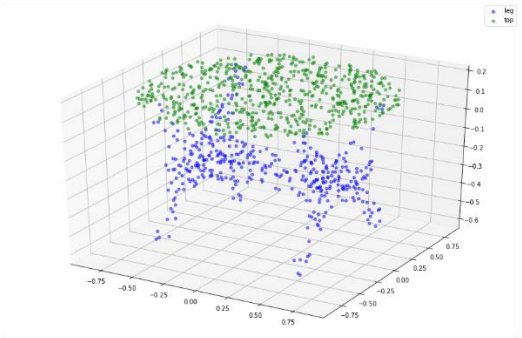


Figure 50 Predicted Labels

CHAPTER 7
Conclusion

7.1 Conclusion

In this project, we explain about a deep neural network called PointNet that directly consumes point clouds and provides a combined approach to various 3D recognition tasks. Our design for PointNet architecture is faced with two challenges, the first challenge is how to design a neural network for unordered inputs. To overcome this, we employ augmentation and symmetric function. Next, we design an architecture to make our network more robust to input geometric transformations. The basic idea is to line the input point cloud canonical space, this alignment is achieved by applying a transformation to input point coordinates. The transformation is predicted by a mini point net which we call T-net, which is n to n trained with the rest of the network and to finally obtain global features that helps in classification we use MaxPooling. Further, the local features and the global features are concatenated to give labels for each of the points in the point cloud thus helping in segmentation of the 3D object.

We were able to segment models like Chair, Aeroplane and Table with variable accuracies of around 80% to 86%, the accuracies can further be increased by changing the number of epochs and learning rate. Therefore, we have successfully executed Classification and Segmentation using PointNet deep neural network architecture that handles 3D point clouds directly.

REFERENCES

[1]R. Q. Charles, H. Su, M. Kaichun and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77-85, doi: 10.1109/CVPR.2017.16.0
(Accessed: 09 September 2021)

[2] Adam Conner-Simons, Deep learning with point clouds (October 21,2019), MIT Computer Science & Artificial Intelligence Lab. Available at: https://news.mit.edu/2019/deep-learning-point-clouds-1021(Accessed: 13 January 2022).

[3] Loic Landrieu, Semantic Segmentation of 3D point Cloud (2019), Université Paris-Est — Machine Learning and Optimization working Group. Available at: https://www.labex-bezout.fr/wp-content/uploads/2019/04/Landrieu_GT_appr_opt.pdf (Accessed: 22 January 2022)

[4] Charles R. Qi et al., Volumetric and Multi-View CNNs for Object Classification on 3D Data (2016), Available at: arxiv.org (Accessed: 17 December 2021)

[5] C. Qi et al, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Available at: https://arxiv.org/abs/1612.00593 (Accessed: 10 December 2021)

[6]Beard, R. (1967). Symmetric Function and Allied Tables. By F. N. David, M. G. Kendall and D. E. Barton. [pp. x 278, Cambridge: Published for the Biometrika Trustees at the University Press, 1966. 70s.]. *Journal of the Institute of Actuaries, 93*(1), 155-157. doi:10.1017/S0020268100039615 (Accessed: 15 December 2021)

[7]H. M. Kasem, K. -W. Hung and J. Jiang, "Revised Spatial Transformer Network towards Improved Image Super-resolutions," *2018 24th International*

Conference on Pattern Recognition (ICPR), 2018, pp. 2688-2692, DOI: 10.1109/ICPR.2018.8546080. (Accessed: 16 January 2022)

[8]Computer Vision Toolbox,Design and test computer vision, 3D vision, and video processing systems. Available at: https://in.mathworks.com/help/vision/ug/point-cloud-classification-using-pointnet-deep-learning.htm (Accessed: 5 December 2021)

[9] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A Survey on 3D Object Detection Methods for Autonomous Driving Applications," IEEE Trans. Intell. Transp. Syst., vol. 20, no. 10, pp. 3782–3795, 2019.(Accessed:17 January 2022)

[10] S. Su and X. Tang, "Systematic Constructions of Rotation Symmetric Bent Functions, 2-Rotation Symmetric Bent Functions, and Bent Idempotent Functions," in *IEEE Transactions on Information Theory*, vol. 63, no. 7, pp. 4658-4667, July 2017, doi: 10.1109/TIT.2016.2621751. (Accessed: 15 January 2022)

[11] M. Jaderberg et al, "Spatial Transformer Networks", 2015 Available at: https://arxiv.org/abs/1506.02025?context=cs (Accessed: 15 January 2022)

[12] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 1(2):4, 2017 (Accessed: 13 January 2022)

[13] Bao-Liang Lu, Yan Bai, Hajime Kita and Y. Nishikawa, "An efficient multilayer quadratic perceptron for pattern classification and function approximation," *Proceedings of 1993 International Conference on Neural Networks (IJCNN-93-Nagoya, Japan)*, 1993, pp. 1385-1388 vol.2, doi: 10.1109/IJCNN.1993.716802. (Accessed: 23 January 2022)

[14]Gujjar, Harish S (2018). *A Comparative Study of VoxelNet and PointNet for 3D Object Detection in Car by Using KITTI Benchmark. International Journal of Information Communication Technologies and Human Development, 10(3), 28–38*. doi:10.4018/ijicthd.2018070103 (Accessed: 23 January 2022)

[15] Wentao Yuan, David Held, Christoph Mertz, Martial Hebert. (18 October,2019) Iterative Transformer Network for 3D Point Cloud. Available at: https://arxiv.org/pdf/1811.11209.pdf (Accessed: 14 January 2022)

[16] Gujjar, Harish S (2018). *A Comparative Study of VoxelNet and PointNet for 3D Object Detection in Car by Using KITTI Benchmark. International Journal of Information Communication Technologies and Human Development, 10(3), 28–38.* doi:10.4018/ijicthd.2018070103 (Accessed: 23 January 2022)

[17] baeldung (2021). Epoch in Neural Network. Available at: https://www.baeldung.com/cs/epoch-neural-networks (Accessed: 15 January 2022)

[18] Liu, Weiping; Sun, Jia; Li, Wanyi; Hu, Ting; Wang, Peng (2019). *Deep Learning on Point Clouds and Its Application: A Survey. Sensors, 19(19), 4188–* . doi:10.3390/s19194188 (Accessed: (14 January 2022)

[19] Neural Photo Editing with Introspective Adversarial Networks Andrew Brock, Theodore Lim, J.M. Ritchie, Nick Weston (2016) Available at: https://arxiv.org/abs/1609.07093 (Accessed: 15 December 2021)

[20]Madhushree Basavarajaiah, Available at:https://medium.com/@bdhuma/6-basic-things-to-know-about-convolution-daef5e1bc411, (Accessed: 11 December 2021)

[21]S. Jeyanthi and M. Subadra, "Implementation of single neuron using various activation functions with FPGA," *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, 2014, pp. 1126-1131, doi: 10.1109/ICACCCT.2014.7019273. (Accessed: 11 December 2021)

[22] Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov*; 15(56):1929−1958, 2014.Submitted 11/13; Published 6/14 Department of Computer Science University of Toronto 10 King's College Road, Rm 3302 Toronto, Ontario, M5S 3G4, Canada(Accesses: 15 January 2021)

[23]S. Zhai *et al*., "S3Pool: Pooling with Stochastic Spatial Sampling," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4003-4011, doi: 10.1109/CVPR.2017.426. (Accessed: 15 December 2021)

[24]keras(2018) Dense layer, Available at:https://keras.io/api/layers/core_layers/dense/(Accessed: 11 December 2021)

[25]A. H. Khan, X. Cao, S. Li, V. N. Katsikis and L. Liao, "BAS-ADAM: an ADAM based approach to improve the performance of beetle antennae search optimizer," in *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 461-471, March 2020, doi: 10.1109/JAS.2020.1003048. (Accessed: 11 December 2021)

[26]J. Zheng, H. Sun, X. Wang, J. Liu and C. Zhu, "A Batch-Normalized Deep Neural Networks and its Application in Bearing Fault Diagnosis," *2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 2019, pp. 121-124, doi: 10.1109/IHMSC.2019.00036.(Accessed: 11 December 2021)

[27] Hong Chang and Dit-Yan Yeung, "Semisupervised metric learning by kernel matrix adaptation," *2005 International Conference on Machine Learning and Cybernetics*, 2005, pp. 3210-3215 Vol. 5, doi: 10.1109/ICMLC.2005.1527496 (Accessed: 12 December 2021)

[28] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao
3D ShapeNets: A Deep Representation for Volumetric Shapes
Proceedings of 28th IEEE Conference on Computer Vision and Pattern Recognition (**CVPR2015**)
(Accessed: 24 January 2022)

[29] Sun, J.; Ovsjanikov, M.; Guibas, L. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. In Computer Graphics Forum; Blackwell: Oxford, UK, 2009.(Accessed: 10 january 2022)

[30] J. -H. Chen, G. -H. Lin, C. M. Yelamandala and Y. -C. Fan, "High-Accuracy Mapping Design Based on Multi-view Images and 3D LiDAR Point Clouds," 2020 IEEE International Conference on Consumer Electronics (ICCE), 2020, pp. 1-2, doi: 10.1109/ICCE46568.2020.9043066. (Accessed: 15 January 2022)

[31]VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition

Daniel Maturana and Sebastian Scherer
Conference Paper, Proceedings of (IROS) IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 922 - 928, *September, 2015.(Accessed:25 January 2022)*

[32] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In IEEE/RSJ International Conference on Intelligent Robots and Systems, September 2015.(Accessed: 20 January 2022)

[33] Goodfellow, Bengio & Courville 2016, p. 200, "Furthermore, back-propagation is often misunderstood as being specific to multi-layer neural networks, but in principle it can compute derivatives of any function".
Availableat:https://www.omgwiki.org/ai/doku.php?id=ai:private:taxonomy:10_append:08_glossary:b:backpropagation. (Accessed:20 January 2022)

[34]Max-Pooling Dropout for Regularisation of Convolutional Neural Networks Haibing Wu and Xiaodong Gu Department of Electronic Engineering, Fudan University, Shanghai 200433, China(Accessed: 28 January 2022)

[35]Z. Wu, R. Shou, Y. Wang, and X. Liu. Interactive shape cosegmentation via label propagation. Computers & Graphics, 38:248–254, 2014.

[36] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3d shape collections. SIGGRAPH Asia, 2016.

[37] Nguyen, A., & Le, B. (2013). 3D point cloud segmentation: A survey. 2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM). doi:10.1109/ram.2013.6758588
10.1109/RAM.2013.6758588