Department of Computer Vision
Universität Siegen

Analysis of PointNet architecture

Studienarbeit

Mechatronics

Surya Vara Prasad Alla

First Examiner
Prof. Dr. Michael Möller — Chair of Computer Vision

Second Examiner
Hartmut Bauermeister — Chair of Computer Vision

This report of Studienarbeit is handed in according to the requirements of the Universität Siegen for the study program Master of Science (M. Sc.) Mechatronics.

First Examiner
Prof. Dr. Michael Möller

Second Examiner
Hartmut Bauermeister

Declaration of Originality

I hereby declare that this Studienarbeit is my own original work and only the indicated sources and resources were used. All references, ideas and direct quotes taken from other sources and used in my report have been fully and properly cited.

Furthermore, it does not contain any material previously published or submitted, neither in whole nor in part, for credit of any other degree at any institution.

.....	.....
Place, Date	Signature

Acknowledgements

This project would have not been possible without the guidance and support of many people. I would like to thank my mother, brother for the continued emotional support during the course of this project.

I would thank PointNet Paper researchers Charles R. Qi, Hao Su, Kaichun Mo and Leonidas J. Guibas for their very good work. I would thank people who are all involved in establishment of datasets MNIST, CIFAR10, ModelNet40.

I would like to thank Prof.Dr.Michael Möller for providing me the opportunity to work under the department of Computer vision at the University of Siegen. I wish to extend my special thanks to Hartmut Bauermeister for his wonderful mentoring throughout the project and I am thankful for the computation resources and project reporting references he provided that helped me understand the objectives of the project.

I would like to dedicate my work to my late father.

Abstract

Deep Learning on Point Sets for 3D Classification is revolutionized by PointNet architecture. PointNet architecture is evaluated as the state of the art method in 3d classification. In this Studienarbeit, PointNet architecture is applied to 2D image data and evaluated. PointNet is highly efficient and effective on 2D data. Empirically, it shows strong performance on par or even better than the usual three layered Convolutional classification network. In this report, it is explained theoritically why the network performs relatively very well.

Keywords: PointNet, Convolutional Neural Network, Point cloud

Contents

1 Introduction

PointNet and CNN architectures are introduced well in this introduction for further better understanding in Background and Method sections of this report.

Deep Neural network means Artificial Neural Networks ANN with multi layers. In the last decades, it is considered as one of the most powerful tools, and it has huge popularity with handling huge amount of data. The topic of deeper hidden layers had accumulated interest to surpass classical methods performance in different fields for example in classification, detection, pattern recognition. The Convolutional Neural Network(CNN) has became most popular in recent times. This name is taken from convolutional mathematical operation between matrixes. CNN consists of multiple layers like convolutional layer, non linearity layer like relu, pooling layer and fully connected layer. Some layers like pooling and non-linearity layers don't have parameters whereas convolutional and fully connected layers have parameters[1].The CNN has produced excellent performance in Deep Learning problems like applications with image data(largest image classification dataset Image Net). A common form of CNN architecture is illustrated in figure1.1.

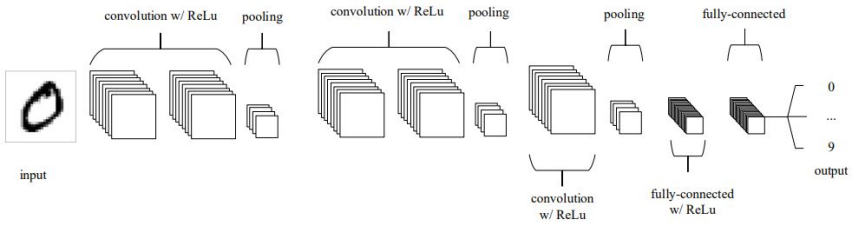

Figure 1.1: A common form of CNN architecture in which convolutional layers are structured with ReLus and pooling layer and later on passed to fully connected layers

[8]

Source of image: Keiron, Ryan's CNN works

Deep learning architectures capable of reasoning on 3D geometric data such as point clouds or meshes. Normal Convolutional Neural Networks require very regular formatted data like those of image grids, 3D voxels in order to apply kernel and weight computations on the data. meshes or point clouds are irregular formatted data structures and mostly these data structures are again converted to regular 3D voxel grids, collections of views and then feeded to the deep neural net network. But converting into 3D voxel grids increases the data volume and would be expensive to the computations on such data.

Due to this, we concentrate on employing point clouds as the sole input representation for 3D geometry, and we call the resulting deep nets PointNets. It is simpler to learn from point clouds since they are uniform, straightforward structures that do not have the combinatorial inconsistencies and complexity of meshes. However, the PointNet must still take into account the notion that a point cloud is simply a collection of points, making it invariant to permutations of its constituents and requiring particular symmetries in the net calculation. It is also necessary to take into account additional rigid motion invariances. Our PointNet is an integrated architecture that accepts point clouds as an input directly and generates class labels for the full input or per point segment for every input point.

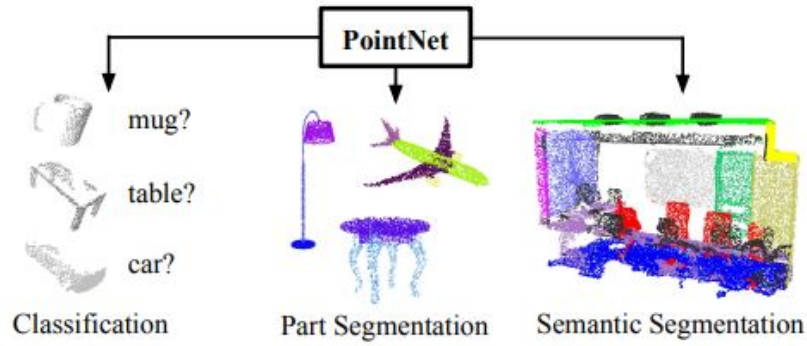

Figure 1.2: Applications of PointNet. It is an architecture to classify and segment 3D data

[11]

Source of image: Charles R. Qi, Hao Su, Kaichun Mo and Leonidas J. Guibas's PointNet Research Paper

PointNet's fundamental architecture is remarkably straightforward because each point is first treated similarly and independently. Each point is initially represented by merely its three coordinates (x, y, z). Calculating normals and other local or global properties may result in the addition of additional dimensions. The employment of a single symmetric function, max pooling, is essential to our strategy. In actuality, the network picks out interesting or instructive points from the point cloud and encodes the rationale behind its selection using a set of optimization functions/criteria.

The network's final fully connected layers either forecast per-point labels or combine these learned ideal values into a global descriptor for the entire shape as indicated before (shape classification) (shape segmentation). Because each point in our input format transforms separately, it is simple to apply rigid or affine transformations to it. In order to further enhance the results, we can add a data-dependent spatial transformer network that makes an effort to canonicalize the data before the PointNet analyzes them.

PointNet has multiple applications as illustrated in figure 1.2. More intriguingly, it turns out that PointNet network learns to condense a point cloud input into a sparse set of key points, which approximately resembles the visual skeleton of objects. The

theoretical study explains why PointNet is highly resilient to corruption caused by point insertion (outliers) or deletion as well as to tiny perturbations of the input points (missing data).

The PointNet is significantly faster under a unified design, and it also displays strong performance on par with or even better than state-of-the-art. PointNet network can fairly approximate any continuous set function. The fact that PointNet network learns to reduce a point cloud input into a sparse set of key points, which roughly reflects the visual skeleton of objects, is even more intriguing.

Theoretical research explains why PointNet is highly resistant to input point perturbations as well as corruption brought on by point insertion (outliers) or deletion (missing data).

2 Background

2.1 Three dimensional data

Real world is 3 dimensional rather than most Computer Vision represented in 2 dimensions. three dimensions consists of three coordinates in a reference coordinate axis considered.

A random point in 3 dimension can be mathematically represented as:

$$p = (a, b, c) \tag{2.1}$$

A 3d object is a random combination of many three dimensional points like p in equation 2.1. There are multiple data representations of three dimensional data. Point Cloud, 3D Mesh, Voxel Grid, Group of 2D images are mostly used. Point Cloud and 3D Mesh are explained mathematically in sections 2.1.1 and 2.1.2

2.1.1 Point Cloud

Point cloud is a group of points in 3D space. usually point cloud has three coordinate points and sometimes also has RGB values and intensity. Point cloud represents a shape in 3D or a 3D object. Point clouds are produced by many sources like Lidar, Depth Sensor, Time of Flight sensor and many more 3D scanners. There are multiple fields in which point clouds are used like to create CAD models, quality inspection, rendering and mass customization.

Point Cloud is unlike an image, an ordered set of 3 dimensional points in a reference frame of Cartesian coordinate system on the surface of objects.

Mathematically Point Cloud can be represented as:

$$P = \{(a_x, b_x, c_x) \mid x \in M\} \tag{2.2}$$

where a,b and c are the coordinates and it contains M number of points

Sometimes addition to point coordinates, we also have RGB color information for every point. In Other representation like mesh, we have vertices and edges.

$$P_{RGB} = \{(a_x, b_x, c_x, R_x, G_x, B_x) \mid x \in M\} \tag{2.3}$$

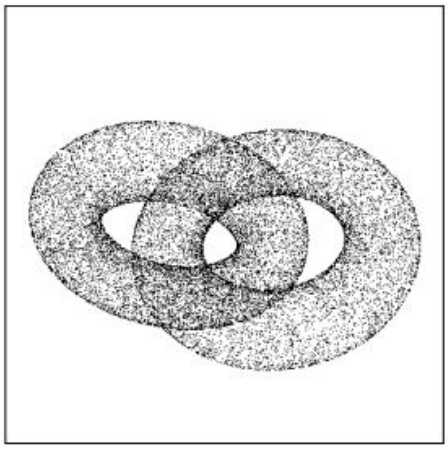

Figure 2.1: Sample Point Cloud
[5]

Source of image: Adam Hunag and Gregory M Nielsom's Surface approximation to point cloud data Research Paper

Point set registration is a process of aligning 3D models with other point clouds or two point clouds. This Alignment of two point clouds or a point cloud with 3D model(CAD, STL) has multiple applications like comparison of accuracies in manufacturing, 3D printing(CAD model compared with point cloud from sensor). A sample point cloud is as shown in figure 2.1. the sample point cloud shown is 3d visualized but looks like 2d as we projected on paper which is 2 dimensional.

Statistical aspects of points are frequently encoded through point features, which are also intended to be invariant to specific transformations, which are typically categorized as intrinsic or extrinsic. They can also be divided into local and international or global features.

2.1.2 3D Mesh

3D mesh has faces of multi sided polygon and vertices according to the number of vertices in the polygon. A mesh is a 3d geometric data structure that shows the surface by a group of polygons.

Meshes are mostly used in computer graphics to represent objects by discretizing the surfaces. Sample Triangle mesh shown in figure 2.2 is a visualized 3d bunny with many vertices, faces and edges. The meshes in image are not clearly visible as it is a zoomed out 2d image of a 3d image. An example of a mesh of 6 vertices, 9 edges, 4 faces is mathematically represented in table 2.1 or data is stored in shown format.

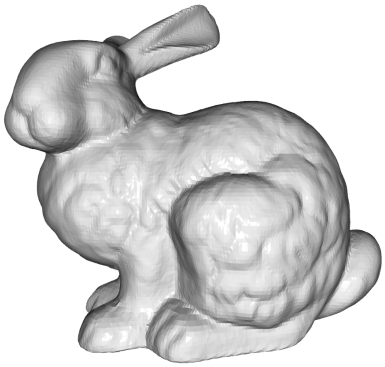

Figure 2.2: Sample Triangle mesh
[15]

Source of image: Taken from Open3D library

Object		
Vertices list	edge list	face list
V1: a1, b1, c1	e1: v1, v2	f1:e1, e8, e6
V2: a1, b1, c1	e2: v2, v6	f2:e2, e3, e7
V3: a1, b1, c1	e3: v6, v4	f3:e7, e9, e8
V4: a1, b1, c1	e4: v4, v5	f4:e9, e4, e5
V5: a1, b1, c1	e5: v5, v3	-
V6: a1, b1, c1	e6: v3, v1	-
-	e7: v2, v4	-
-	e8: v2, v3	-
-	e9: v4, v3	-

Table 2.1: Sample Triangle mesh

2.1.3 key properties of 3D points

3D points particularly point cloud has three key characteristics or problems:

Unordered: The point cloud is a collection of points with no particular order, unlike pixel arrays in pictures or voxel arrays in volumetric grids. In order to consume N 3D point sets, a network must be invariant to N! permutations of the input set in the order of data feeding.

Interaction between points: The data of points is in a space of distance metric. It implies that points are not isolated and that nearby points make up a significant subgroup. The classification neural network must therefore be able to represent local struc- tures from neighboring points as well as the combinatorial interactions between local structures.

Invariance when transformations: The learned representation of the point set from classification neural network ought to be invariant to some transformations since it is a geometric object. For instance, combining point rotation and translation

shouldn't change either the cat- egory of the overall point cloud or the way the points are divided

2.2 Deep Learning on 2D Data

CNN is the most well-known and often used algorithm in DL. The fundamental advantage of CNN over its forerunners is that it does it without human intervention, automatically identifying the pertinent features. CNNs have been widely used in a variety of industries, such as computer vision, speech recognition, and face recognition. CNNs have a structure akin to a traditional neural network and were modeled after the neurons found in human and animal brains. More specifically, the visual cortex in a cat's brain is made up of a convoluted pattern of cells; the CNN simulates this pattern. Three advantages of CNN were outlined by Goodfellow et al.: similar representations, sparse interactions, and parameter sharing.

In contrast to typical fully connected (FC) networks, the CNN uses shared weights and local connections to fully exploit 2D input-data structures, such as visual signals. This technique uses a remarkably minimal amount of parameters, which speeds up the network while also making training easier. The visual cortex cells also have this. Notably, rather than detecting the entire image, these cells only detect specific portions of it (i.e., these cells spatially extract the local correlation available in the input, like local filters over the input). A popular CNN variant, comparable to the multi-layer perceptron (MLP), has many convolution layers that come before sub-sampling (pooling) layers and FC layers at the end.An example of CNN Deep Learning architecture for image classification is well illustrated in Fig 2.3.

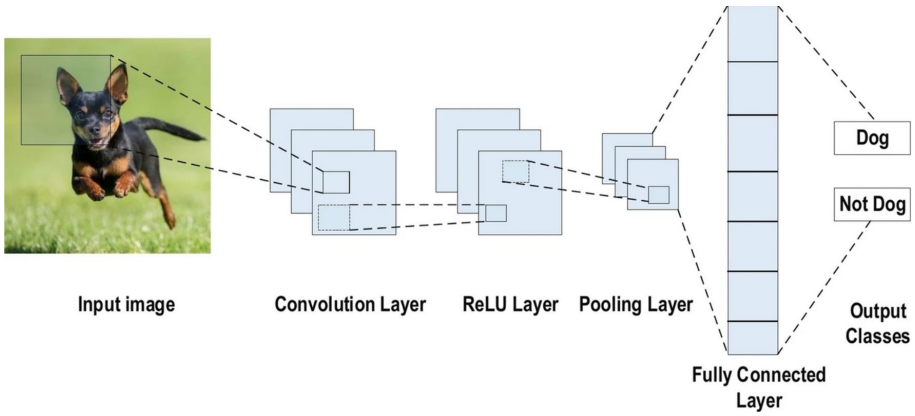

Figure 2.3: CNN Deep Learning on 2D Data [2]

Source of image: Laith Alzubaidi's Review of Deep learning research paper

2.2.1 Fully connected (FC) layer

A fully connected neural network consists of a series of fully connected layers. Each output dimension depends on each input dimension [4].In detail, the computations

of the k-th layer can be written as:

$$x^{(k)} = \sigma(W^{(k)}x^{(k-1)} + b^{(k)}) \tag{2.4}$$

Here $W^{(K)}$ represents the Weight Matrix of k-th layer $x^{(k)}\epsilon R^p$ represents the output of the k-th layer and x_0 is the vectorised sinogram input for the first layer.

2.2.2 Convolution Layer

A digital photo is represented in a numpy array format and it has one channel in a grayscale or black and white photo, three channels R. G and B in a color photo. A color image is represented in three matrices and each matrix has values in range of 0 and 255.

Convolution layers are the key components of CNNs. It involves partial connections compared with fully connected layers, where each node focuses on a local region of the input[4]. In detail denote $W_c^{(k)} = \{W_1^{(k)}, W_2^{(k)}, ..., W_c^{(k)}\}$ to represent a series of C^k filters and the computations of the k-th convolution layer can be written as:

$$x_c^{(k)} = \sigma(W_c^{(k)} \circledast x^{(k-1)} + b^{(k)}) \tag{2.5}$$

where $\circledast$ represents a convolution operator and σ is an activation function and C represents the number of channels.

2.3 Deep Learning on 3D Data

There are many widely used representations of 3D data, which has led to a variety of learning strategies. Volumetric CNNs are the first to use 3D convolutional neural networks on voxelized forms. However, due to data sparsity and 3D convolution's high computational cost, volumetric representation is limited in terms of resolution. Although FPNN and Vote3D suggested unique approaches to address the sparsity issue, they still operate on sparse volumes, making it difficult for them to handle very large point clouds. Multiview CNNs: [[13], [9]] have attempted to convert 3D point clouds or forms into 2D images, which were subsequently classified using 2D Conv nets. This line of approaches has obtained a dominant performance on shape classification and retrieval tasks using well-designed image CNNs [[12]].

The extension to scene comprehension or other 3D tasks like point classification and shape completion is not simple, though. CNN spectra: On meshes, spectral CNNs are used in certain recent publications. These techniques are now limited to manifold meshes, such as those found in biological items, and it is not clear how to adapt them to non-isometric shapes, such as those seen in furniture. Feature-based DNNs: first vectorize the 3D data by extracting conventional shape features, then classify the shape using a fully connected network. We believe they are limited by the characteristics' extracted representational capacity. figure 2.4 shows or represents a 3D CNN network.

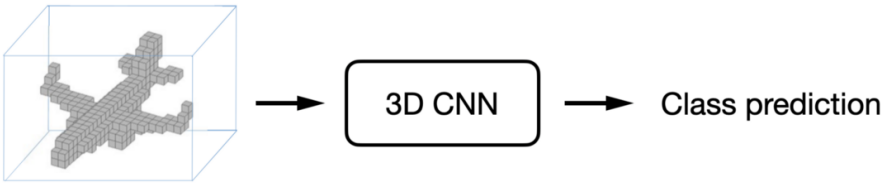

Figure 2.4: PointNet architecture
[10]

Source of image: Volumetric and multi-view cnns for object classification on 3d data Research Paper

2.3.1 The problems With Convolutional Neural Networks for 3d data

Convolutional neural networks are effective, but they are also not highly resistant to scale, rotation, and translation changes.

This means that the network won't be able to accurately identify a picture if it is rotated, translated, or scaled differently from the training data.

Images can frequently be translated, rotated, and scaled in ways that are not recognized during the training process, which might be an issue. A convolutional neural network, for instance, might not accurately identify a cat in a photo that has been rotated by 90 degrees.

Data augmentation, which can be done by adding rotated, translated photos or by producing brand-new data with the aid of a Generative Adversarial Network, is one technique to address these problems (GAN). In PointNet, a Spatial Transformer Network is used in the solution.

A neural network architecture called a spatial transformer network is made to place the input data in the best possible 3D position. As point clouds represent 3D structures, this enables the network to more faithfully represent complicated data structures like them.

2.4 Research Datasets

Collections of unprocessed information obtained during the course of a research project are known as datasets. Many organizations, including universities, government agencies, and research institutions, make the data they have gathered publicly accessible online for use by other researchers. The datasets MODELNET40, MNIST,and CIFAR10 that would be used in this Studienarbeit are expalined in detail in this section.

2.4.1 ModelNet40 dataset

The Princeton ModelNet project aims to offer a thorough, organized collection of 3D CAD models for objects to academics working in computer vision, computer graphics, robotics, and cognitive science. Using the facts gleaned from the SUN database, a list of the most prevalent object categories worldwide was created to form the dataset's core. They gathered 3D CAD models from each object category by searching for each phrase in each object category in online search engines. The models that did not fit into the top 10 popular item categories were manually removed to get a very clean dataset. Additionally, the orientation of the CAD models for this sample of 40 classes was manually aligned[14].

2.4.2 MNIST dataset

The training set has 60,000 samples, while the test set contains 10,000 examples in the MNIST database of handwritten digits, which is accessible from [3]. It is a portion of a larger set that is made public by NIST. The digits have been centered in a fixed-size image and size-normalized. It is a useful database for those who want to practice new skills and pattern recognition algorithms on actual data with the least amount of preprocessing and formatting work[3].

2.4.3 CIFAR10 dataset

The CIFAR-10 dataset consists of 6000 images per class in 10 classes totaling 60000 32x32 color images. 10000 test photos and 50,000 training images are available. Five training batches and one test batch, each with 10,000 photos, make up the dataset. Exact 1000 randomly chosen photos from each class make up the test batch. In random order remaining images are distributed, however certain training batches can have a disproportionate number of images from a particular class. each class combined, training batch has exactly 5000 photos inclusive of all classes. [7].

2.5 PointNet architecture

Fig.2.5 depicts the entire PointNet network architecture, showing how the segmentation and classification networks share many of the same structural elements.

The architecture is little straightforward and easy to understand. The classification network maps each of the n points from the three dimensions (x, y, and z) to 64 dimensions using a shared multi-layer perceptron (MLP). It's crucial to remember that each of the n points shares a single multi-layer perceptron (i.e., mapping is identical and independent on the n points). To map the n points from 64 dimensions to 1024 dimensions, this process is repeated. In R^{1024}, a global feature vector is made via max pooling with the points in a higher-dimensional embedding space. Finally, the global feature vector is mapped to k output classification scores using a three-layer fully connected network.

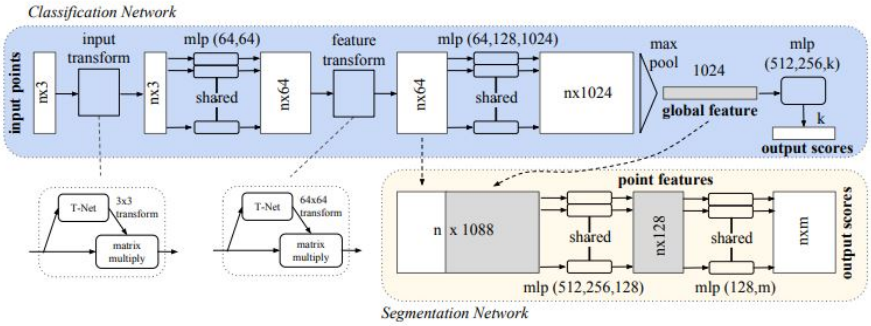

Figure 2.5: PointNet architecture
[11]

Source of image: Charles R. Qi, Hao Su, Kaichun Mo and Leonidas J. Guibas's PointNet Research Paper

A detailed explaination below will elaborate on the use of maxpooling and the transformation networks inturn solving the three characteristics or issues with 3d points as mentioned in section 2.1.3.

2.5.1 Permutation Invariance

Point clouds, which are portrayed or represented as numerical sets, are naturally unstructured data, as was previously established. More specifically, there are N! permutations for N data points.

The authors of PointNet used symmetric functions—those whose value given n parameters is the same regardless of the order of the inputs—to make Point Net invariant to input permutations. This is sometimes referred to as the commutative property for binary operators. Some examples of commutative property are:

$$\begin{aligned} sum(p,q) &= sum(q,p) \\ average(p,q) &= average(q,p) \\ max(p,q) &= max(q,p) \end{aligned} \tag{2.6}$$

The symmetric function is specifically used by the authors once the n input points are translated into higher-dimensional space, as depicted below. The outcome is a global feature vector intended to record the collective signature of the n input points. The global feature vector's expressiveness is logically correlated with its dimension (and thus the dimensionality of the points that are input to the symmetric function). The segmentation process uses both the local point features and the global feature vector in addition to direct classification.

As mentioned in 2.6, we could also try average, sum operations as symmetric functions in addition to max operation. Results in PointNet paper show that maxpool operation performed well. Observe the results in figure 2.7 below which are taken from Pointnet paper [11].

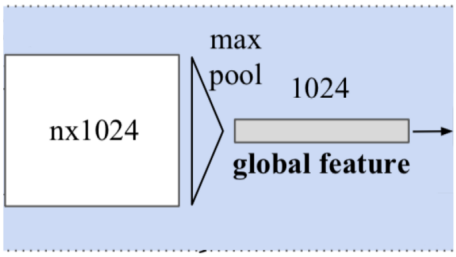

Figure 2.6: Use of MaxPool layer, a Symmteric function in PointNet [11]

Source of image: A screenshot of Point Net architecture

	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 2.7: Maxpool vs Average vs Sum [11]

Source of image: Results image taken from PointNet paper

2.5.2 Transformation Invariance

An object's classification (and segmentation) ought to be resistant to specific geometric modifications (e.g., rotation). The "input transform" and "feature transform" are modular sub-networks that aim to provide pose normalization for a given input. Authors of PointNet were inspired by Spatial Transformer Networks (STNs) [6].

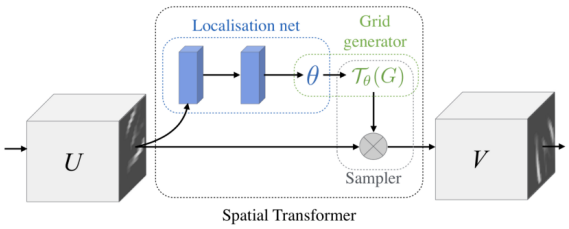

Figure 2.8: Components of Spatial Transformer [6]

Source of image: image taken from Spatial Transformer Networks paper

The Spatial Transformer makes the pose normalization to the input. making use of this pose normalization decreases the requirement for data augmentation in the image 2d and 3d data. 3 parts in STN are: Localization network, Grid generator and Sampler. Since objects can also assume an infinite number of poses, pose nor-

malization is advantageous when applied to point clouds. Figure 2.8 displays the components of Spatial Transformer.

Applying the appropriate rigid or affine transformation for the given input point cloud to accomplish pose normalization. Applying a geometric transformation merely entails multiplying each point by a transformation matrix because each of the n input points is represented as a vector and is individually mapped to the embedding spaces. There is no need for sampling, in contrast to the Spatial Transformers program that uses images. A snapshot of the input transform is shown in Fig. 2.9. The T-Net is a regression network that, like the localization net in STs, is charged with forecasting an input-dependent 3-by-3 transformation matrix that will subsequently be multiplied by the n-by-3 input matrix.

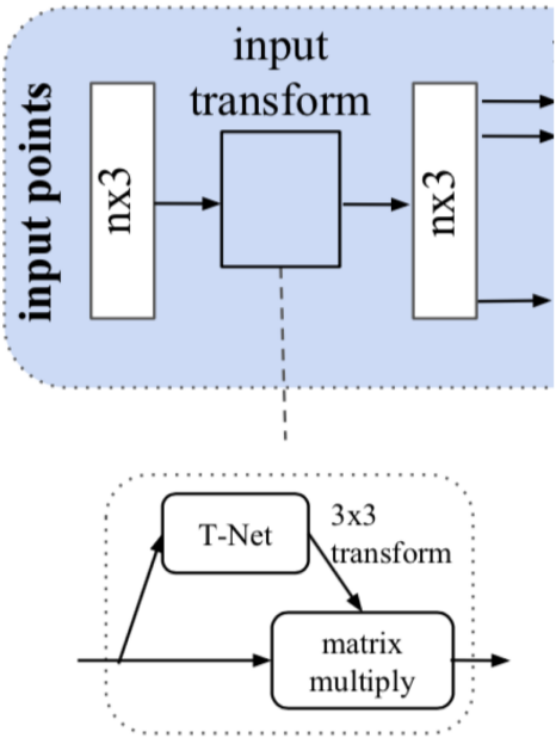

Figure 2.9: Input Transform
[11]

Source of image: image taken from PointNet paper

The higher-level architecture of PointNet serves as the driving force behind the T-Net operations. The input points are mapped independently and identically to a higher-dimensional space using MLPs (or fully-connected layers), and a global feature vector is encoded using max pooling before having its dimensionality reduced to R^{256} using FC layers. The final FC layer's input-dependent features are coupled with globally trainable weights and biases to create a 3-by-3 transformation matrix.

The 64-dimensional embedding space is an extension of the pose normalization notion (referred to as the "feature transform" in Figure 2.5). With the exception of the

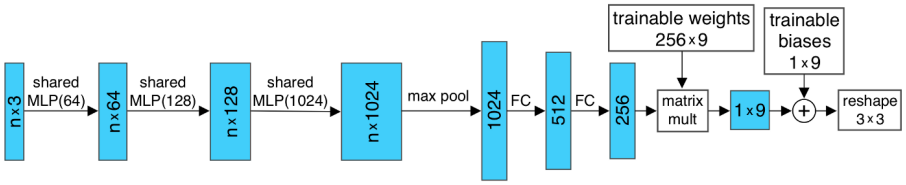

Figure 2.10: 3x3 Tnet architecture
[6]

Source of image: image taken from a Medium article inspired from Spatial Transformation paper

trainable weights and biases, which are 256-by-4096 and 4096, respectively, resulting in a 64-by-64 transformation matrix, the equivalent T-Net is virtually identical to that of Figure 2.10. A regularization term in equation 2.7 is introduced to the loss function to prevent overfitting and instability that could arise from the increased number of trainable parameters. The resulting 64 by 64 transformation matrix, denoted as B below, is encouraged to resemble an orthogonal transformation by the regularization term, which is illustrated below.

$$N_{reg} = ||I - BB^T||^2 \tag{2.7}$$

2.5.3 Local and Global points interaction for Segmentation

The shape of global features for classification can be simply used to train an SVM or multi-layer perceptron classifier. Point segmentation, however, needs a mix of local and global knowledge. We can accomplish this in a straightforward yet incredibly efficient way. We concatenate the global feature with each of the point features after computing the global point cloud feature vector in order to feed it back to the per-point features. Then, on the basis of the combined point features, we extract fresh per-point features; this time, the per-point feature is cognizant of both local and global information.PointNet network can now predict per-point quantities that depend on both local geometry and global semantics.

2.5.4 Expressiveness of PointNet

The dimensionality of the global vector of PointNet (in Pointnet paper mentioned as bottleneck dimension) is the expressiveness of PointNet.

Pointnet standard paper is with bottleneck size of 1024. The graph in figure 2.11 shows schematic comparison for number of points in input point and bottleneck size influencing the accuracy.

A carefully chosen symmetric function that was applied which gives the feature vector (for permutation invariance). Max pooling is specifically used by PointNet. The output of max pooling reduces the input point cloud's n points to a subset of

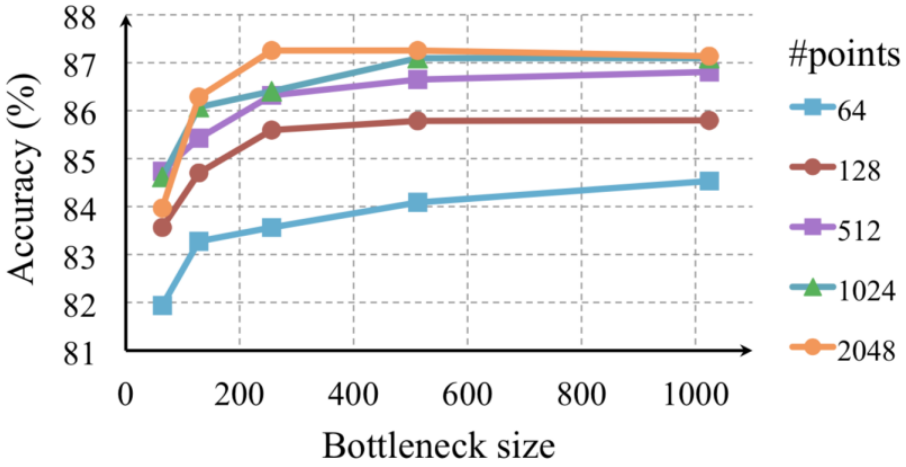

Figure 2.11: Bottle neck size
[11]

Source of image: image taken from PointNet paper

points, much like how the max operator reduces several real-valued inputs to a single value. In actuality, the global feature vector can be contributed to by a maximum of K points. The critical point set, which encrypts the input with a small number of key points, is made up of the points that do contribute to and form the global feature vector.

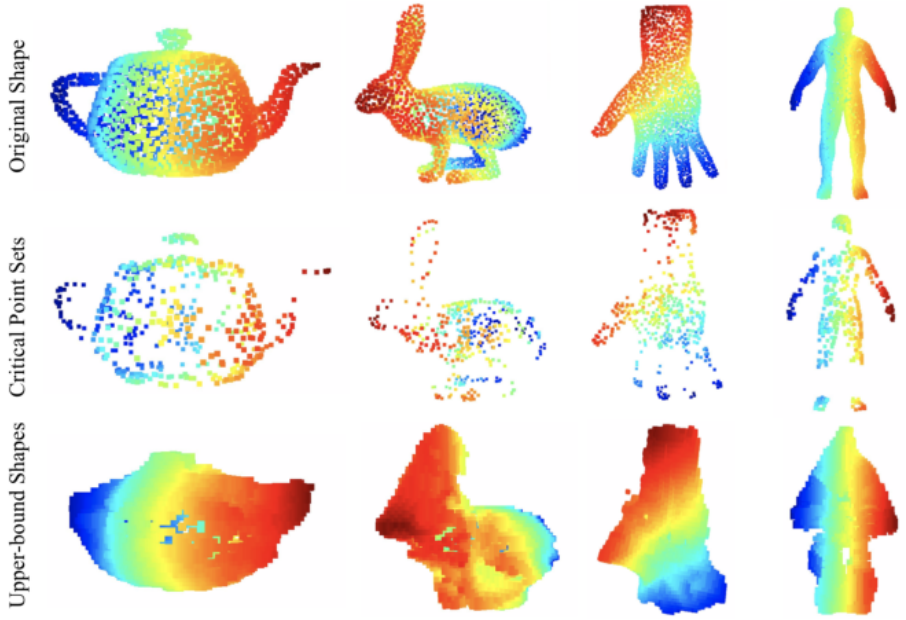

Figure 2.12: Visualization of critical point sets and upper-bound shapes
[11]

Source of image: image taken from PointNet paper

There is a bound on input points that won't affect the global feature vector, much to how inputs fewer than the true maximum have no effect on the output of the

max operator. The upper-bound form in figure 2.12 is used to depict this bound above. Keeping in mind that noise above the upper-bound form modifies the global feature vector, yet classification error may not always result. In conclusion, significant robustness is achieved because the global feature vector is unaltered for points between the critical point set and the upper-bound form.

2.6 PyTorch and Tensorflow

PyTorch is a modern deep-learning framework built on Torch. It was created by Facebook's AI research team and made accessible on GitHub in 2017 for natural language processing applications. PyTorch is known for its ease of use, versatility, efficient memory utilization, and dynamic computational graphs. Furthermore, it has a native feel, which speeds up processing and simplifies programming.

In 2015, Google created and released **TensorFlow**, a comprehensive open-source deep learning framework. It is well-known for its support for numerous abstraction layers, scalable production and deployment options, and interoperability with a variety of platforms, including Android. TensorFlow, a symbolic math framework for neural networks, is best used for dataflow programming across a wide range of activities.

3 Method

In this section of report, method approach understanding of experiments section is explained. We have multiple networks experimented on different datasets. the methods behind these experiments is explained below.

3.1 Three layered CNN network

Three layered CNN network has three Convolution neural layers with feature extraction. Inturn it also increases the number of channels to ease the extraction by increasing the feature size. for the classification task, the image will then be gone through dimensionality reduction. finally we have the scores for each class and more accurate class will have score near to 100 percent or 1. In the experiments section 4.1, we apply this method of approach to MNIST, CIFAR10, Grayscale CIFAR10 datasets.

3.2 3d PointNet

In 3d pointnet architecture, blue region of Figure 2.5 clearly shows the classification methodology. It can be explained as below:

1. Typical Input point cloud: Normally Input point cloud has three coordinates (p, q, r) with some m number of points giving it a shape of mx3. So, the typical input size of a pointnet is mx3.

2. Input Transformation: making all pointclouds to a standard transformation by multiplying each pointcloud with a 3x3 matrix, which is called T-Net. The size of output matrix from this transformation is mx3x3x3 = mx3.

3. The first multilayer perceptron which has 64 nodes in hidden layer and 64 nodes in output layer has its input from input transformation. The MLP is unique for all the pointclouds. The MLP output will have mx64 as it is fully connected and it has 64 nodes.

4. Feature Transformation now aligns every pointcloud into a unique alignment making it easy for feature extraction. it multiplies a 64x64 with mx64 giving output as mx64.

5. The second MLP is similar to First MLP in terms of its functionality except in number of nodes. it has 1024 nodes. the output of this MLP after feature transform is mx1024.

6. Maxpool layer in blue region of Figure 2.5 only gives the maximum value in each dimension of m dimensions and giving output 1x1024 from mx1024.

7. The third MLP(fully connected layers) has 3 layers - 512 nodes in first layer, 256 nodes in second layer and k nodes in output layer. These fully connected layers are for dimensionality reduction from 1024 to k classes of classification task.

3.2.1 for 2D datasets

The MNIST or CIFAR10 dataset is a 2d image data with pixel coordinates which can be seen as (x,y) coordinates. Input for 3d Pointnet is 3d pointcloud or other 3d data representations. It is needed to convert 2d data to 3d data to use 3d pointnet on MNIST or CIFAR10 dataset. for this, we initially need to read the 2d coordinates (x,y) from pixel coordinate axis. This can be read from 2d image by looking at a threshold level in pixel values of image array. After getting the pixel coordinates of pixels which has pixel value more than the threshold level, a histogram is drawn to interpret the amount of points in each image according to our threshold.

Then each image is sampled to this chosen number of points - if number of points less than this threshold points, the points are read repeatedly and if else, some points are sampled from point cloud until it reaches threshold. now, once we get the unique number of 2d points - we append a very small third dimension from gaussian distribution. this then effectively gives us, three points in point cloud with a label. this is inputted to above explained point net architecture for classification task.

3.3 2D PointNet for 2D datasets

The MNIST or CIFAR10 dataset is a 2d image data with pixel coordinates which can be seen as (x,y) coordinates. for 2d data, 3d Pointnet is adjusted by changing the input channels in the architecture and now it is called **2D Pointnet**. To use our 2d data on 2D pointnet, we need pixel coordinates. for this, we initially need to read the 2d coordinates (x,y) from pixel coordinate axis. This can be read from 2d image by looking at a threshold level in pixel values of image array. After getting the pixel coordinates of pixels which has pixel value more than the threshold level, a histogram is drawn to interpret the amount of points in each image according to our threshold.

Then each image is sampled to this chosen number of points - if number of points less than this threshold points, the points are read repeatedly and if else, some points are sampled from point cloud until it reaches threshold. now, we get the unique number of 2d points. this then effectively gives us, two point cloud with a label. this is inputted to above explained 2D point net architecture for classification.

4 Experiments

Experiments in this Studienarbeit include:

(1) 3 layered CNN Network on MNIST, CIFAR10, Grayscale CIFAR10 datasets

(2) 3D PointNet Network on ModelNet40, MNIST, CIFAR10, Grayscale CIFAR10 datasets

(3) 2D PointNet Network on MNIST, CIFAR10, Grayscale CIFAR10 datasets.

Validation of Networks on datasets, Visualization of results is reported in this chapter of Studienarbeit.

4.1 Three layered CNN Network

The three layered architecture in Figure 4.1 shows the detailed architecture using tensorboard graph.

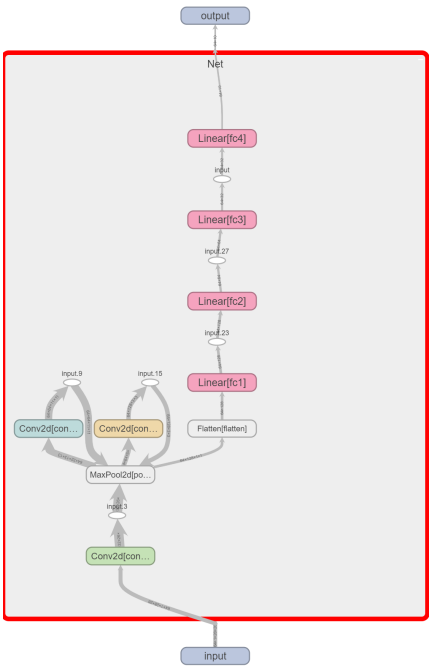

Figure 4.1: Tensorboard graph of 3 layer CNN architecture

4.1.1 MNIST

The feature maps of a MNIST image of digit 2 are shown in the figure 4.2. The three convolution layers of 3 layer CNN network have different weight and bias parameters. When matrix multiplied these parameters with images will give us feature maps.

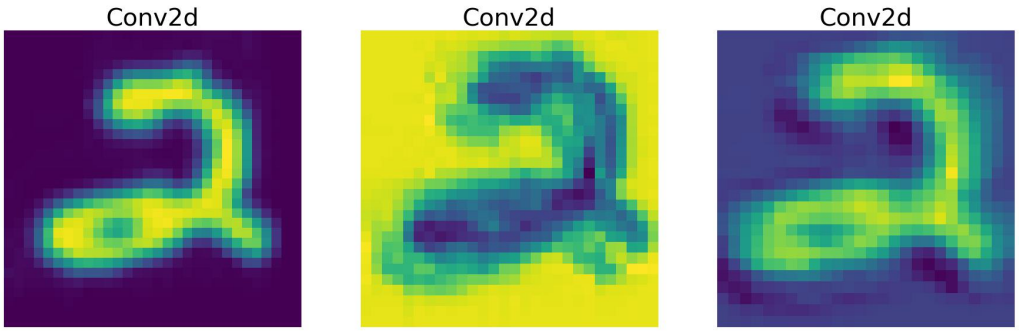

Figure 4.2: Feature maps of MNIST with 3 layer CNN architecture

The experiments of MNIST on 3 layered CNN network are executed using python. Test and Train accuracy, Test and Train loss for 15 epochs are displayed in Figures 4.3 and 4.4. The test and train accuracies of MNIST dataset has got good accuracy just after 2 and 3 epochs and performed consistently from fourth epoch till fifteen epochs.

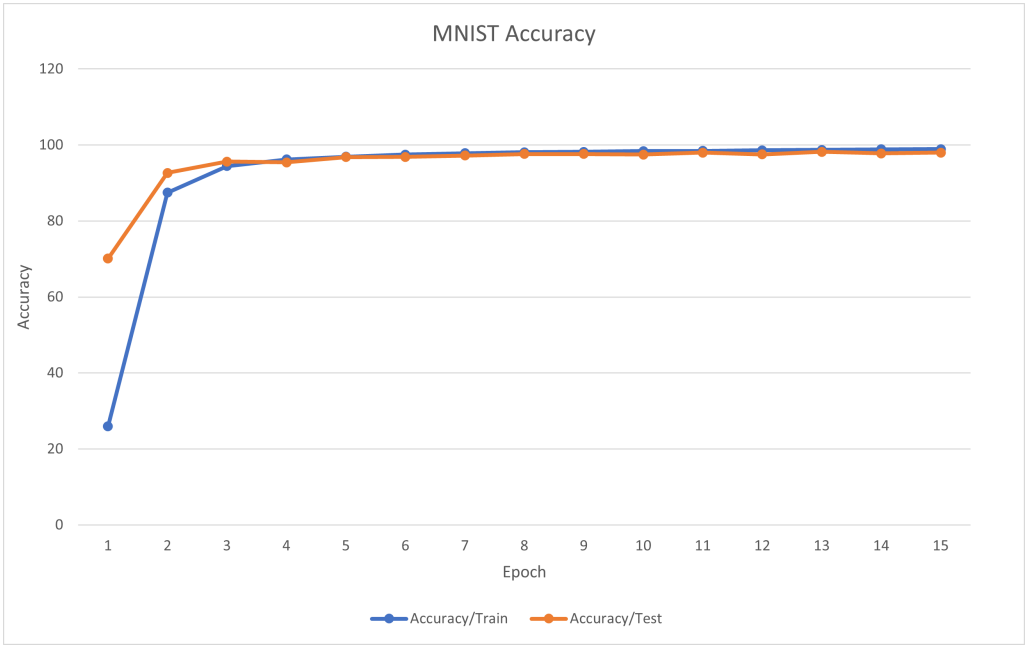

Figure 4.3: MNIST train and test accuracy

The test accuracy of MNIST on 3 layered CNN network is **97.98** percent. The 3 layered network performed very well with MNIST dataset.

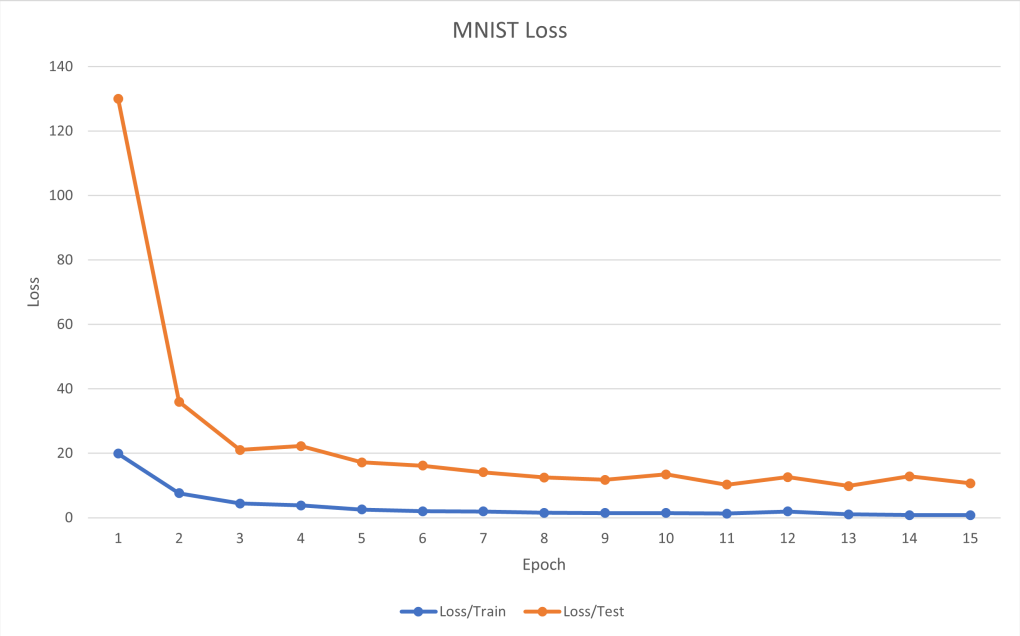

Figure 4.4: MNIST train and test loss

4.1.2 Grayscale CIFAR10

The feature maps of a Grayscale CIFAR10 image of dog are shown in the figure 4.5. The three convolution layers of 3 layer CNN network have different weight and bias parameters. When matrix multiplied these parameters with images will give us feature maps.

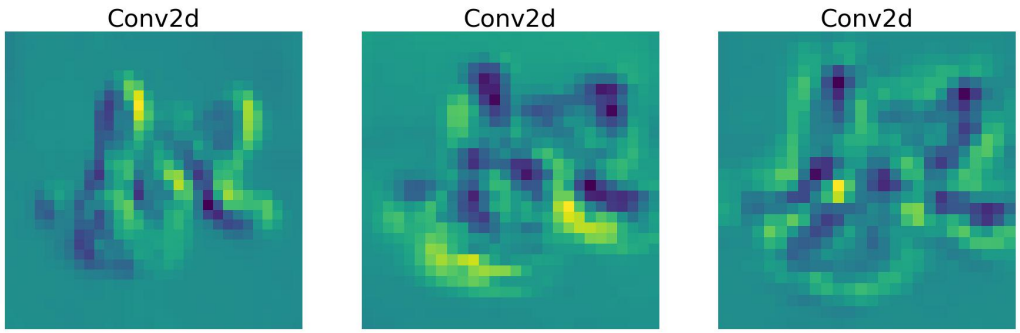

Figure 4.5: Feature maps of Grayscale CIFAR10 dog image with 3 layer CNN architecture

The experiments of Grayscale CIFAR10 dataset on 3 layered CNN network are executed using python. Test and Train accuracy, Test and Train loss for 15 epochs are displayed in Figures 4.6 and 4.7. The test and train accuracies of Grayscale CIFAR10 dataset has got good accuracy just after 3 and 4 epochs and performed consistently from fifth epoch till fifteen epochs.

The test accuracy of Grayscale CIFAR10 on 3 layered CNN network is **69.62** percent. The 3 layered network performed well with Grayscale CIFAR10 dataset.

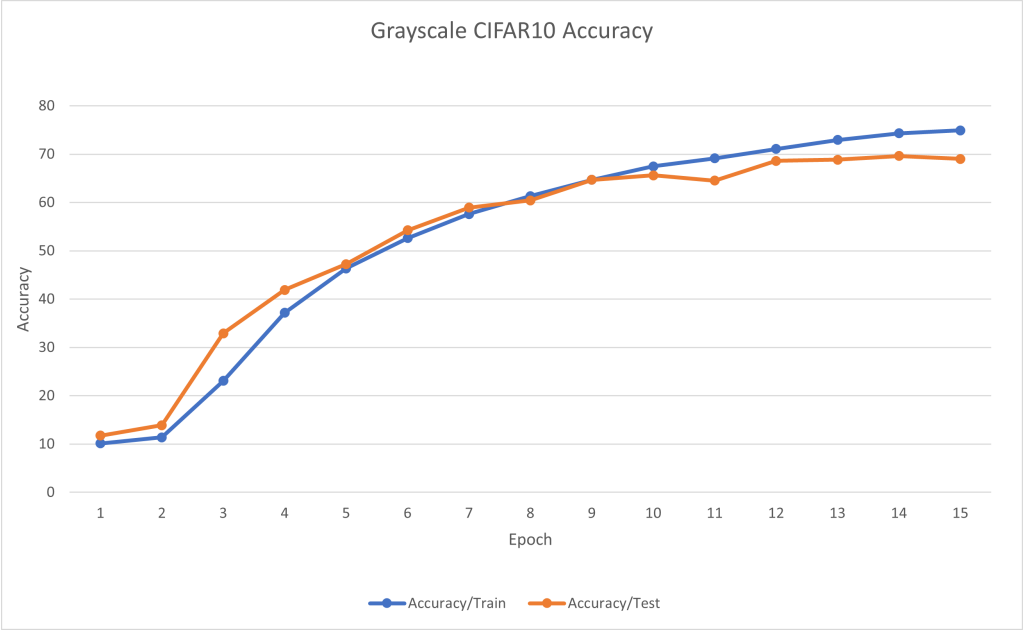

Figure 4.6: Grayscale CIFAR10 train and test accuracy

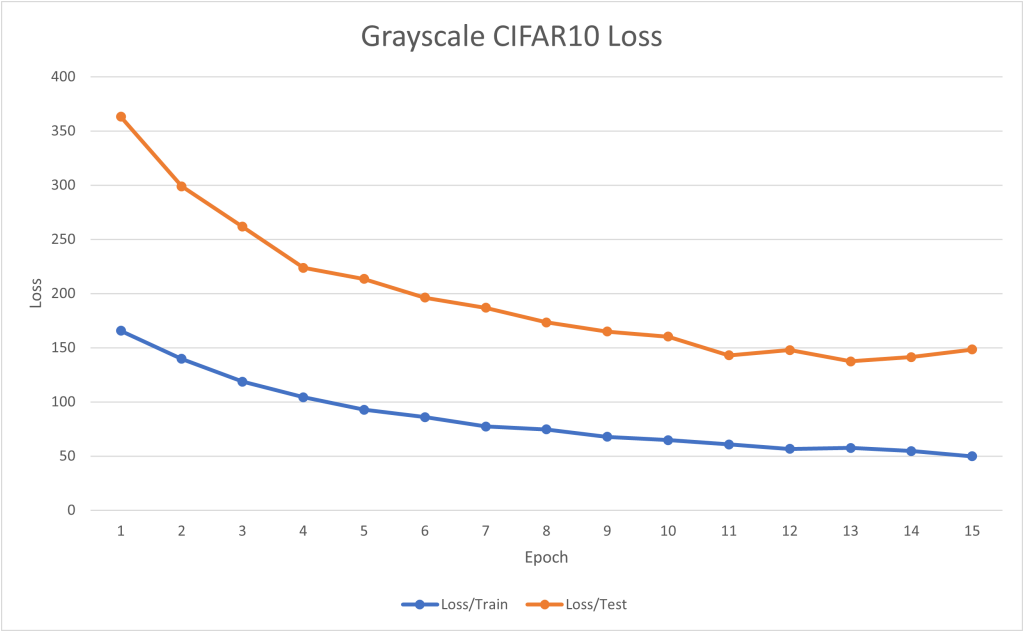

Figure 4.7: Grayscale CIFAR10 train and test loss

4.1.3 CIFAR10

The feature maps of a CIFAR10 image of dog are shown in the figure 4.8. The three convolution layers of 3 layer CNN network have different weight and bias parameters. When matrix multiplied these parameters with images will give us feature maps.

The experiments of CIFAR10 dataset on 3 layered CNN network are executed using python. Test and Train accuracy, Test and Train loss for 15 epochs are displayed in Figures 4.9 and 4.10. The test and train accuracies of CIFAR10 dataset has got

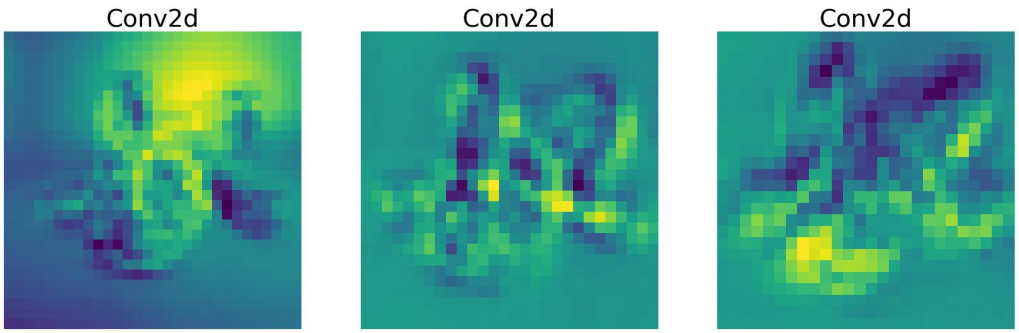

Figure 4.8: Feature maps of CIFAR10 dog image with 3 layer CNN architecture

good accuracy just after fifth epoch and performed consistently from sixth epoch till fifteen epochs.

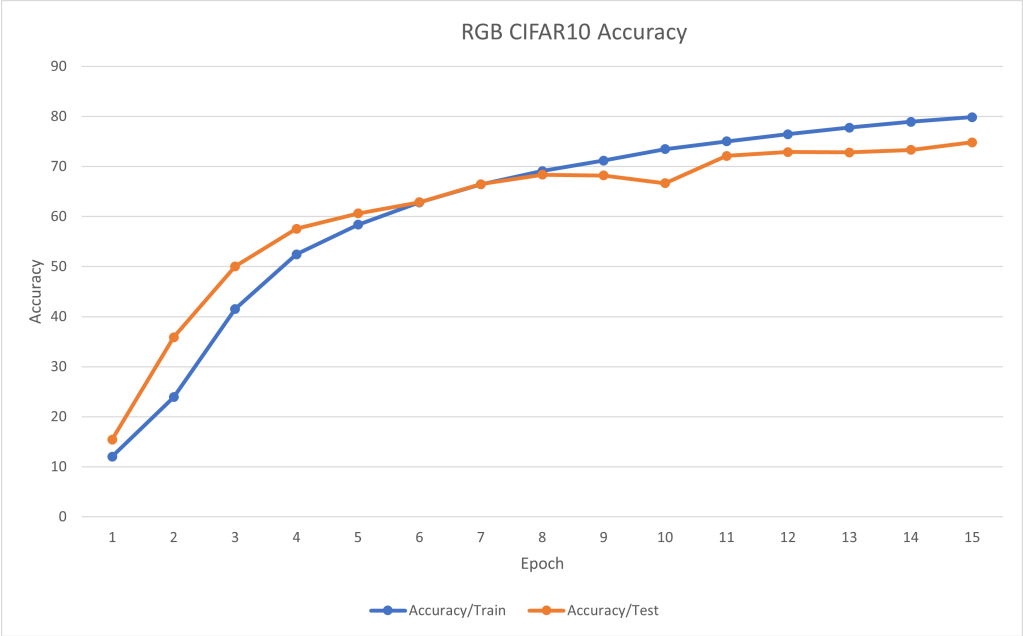

Figure 4.9: CIFAR10 train and test accuracy

The test accuracy of CIFAR10 on 3 layered CNN network is **74.83** percent. The 3 layered network performed well with CIFAR10 dataset.

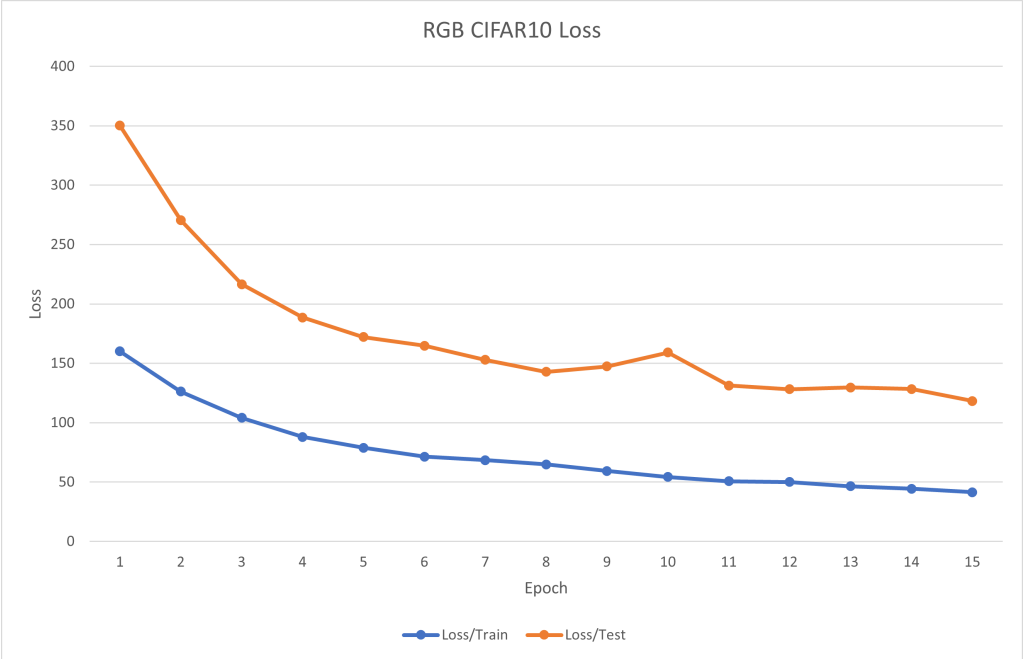

Figure 4.10: CIFAR10 train and test loss

3 layered CNN network				
Dataset	Train loss	Train accuracy	Test loss	Test accuracy
MNIST	0.83	98.93	10.69	97.98
GrayscaleCIFAR	49.86	74.94	148.35	69.62
CIFAR	41.51	79.87	118.18	74.84

Table 4.1: Performance table of 3 layered CNN network for 15 epochs on whole train and test datasets

4.2 3D PointNet Network

3d point cloud data from ModelNet40 dataset considered has unordered 3d coordinate points. Now these points have to be in a chronological order to regenerate the object. How do we generate this order? We need to make sure that it is rotation or scale invariant and we make sure it my using a spatial transformation network in the algorithm.

4.2.1 MODELNET40

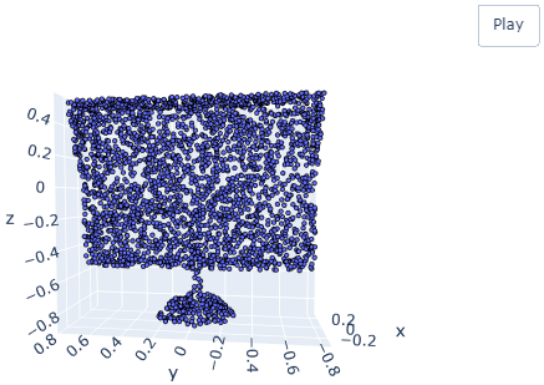

Figure 4.11: Plot of a point cloud in ModelNet40

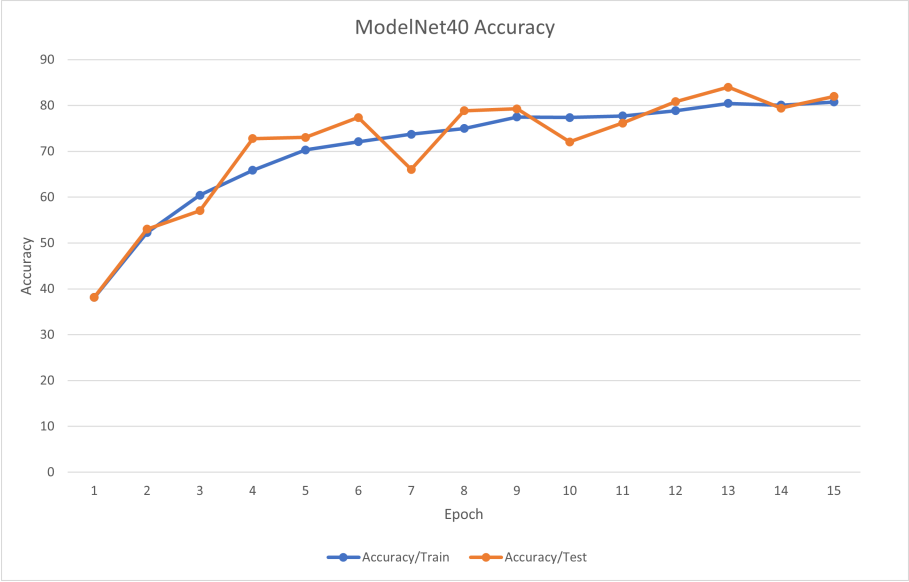

Figure 4.12: Modelnet40 with PointNet accuracy plot

4.2.2 MNIST

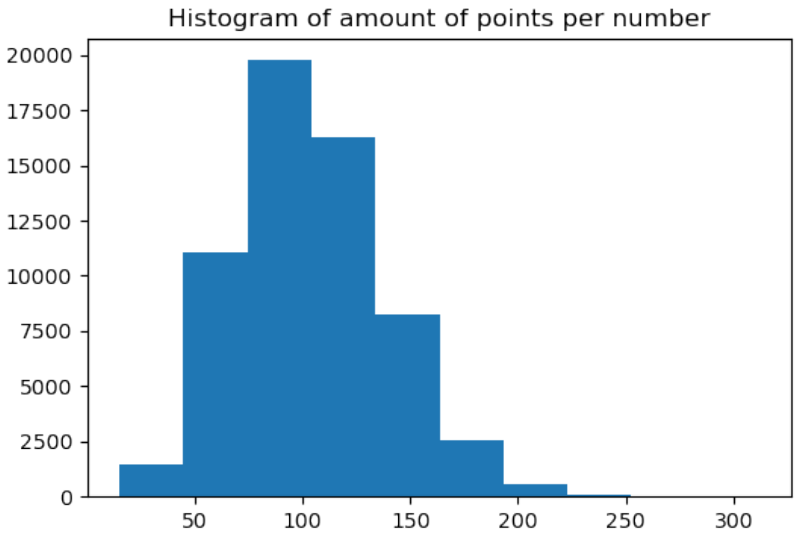

Figure 4.13: Histogram of points in MNIST

for MNIST to be inputted to 3d pointnet, we use the method in 3.2.1. From histogram in figure 4.13, we get the maximum of length of points of all the MNIST dataset converted to 2d pixel coordinates. from this maximum value, we learn to make every image with 2d pixel coordinates to reach this maximum number of pixel coordinates. Now, we need to make a third coordinate to convert 2d to 3d data. for this, we need to append a small gaussian noise as a pixel coordinate to the 2d image with pixel coordinates. Now a 3d image is achieved as shown in figure 4.14.

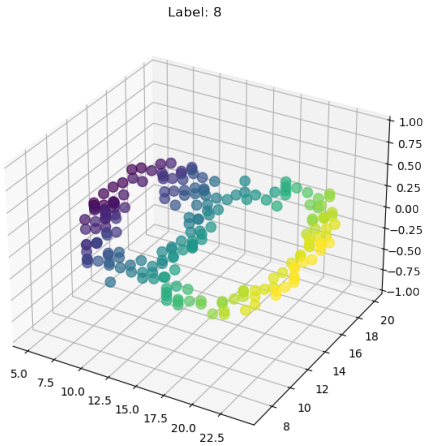

Figure 4.14: 3d data achieved from MNIST 2d image

then network and training goes as similar to network training in Pointnet for 3d

data in section 4.2.1. The results from training and validation are shown below in figure 4.17 The testing of MNIST converted to 3d data on pointnet architecture gave an accuracy of **0.951** or **95.1** percent.

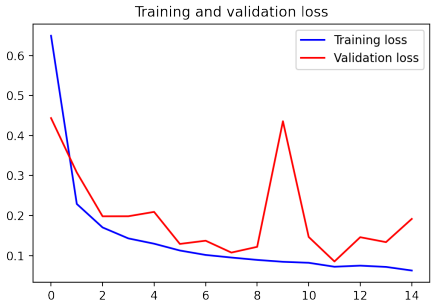

Figure 4.15: Training and Validation loss

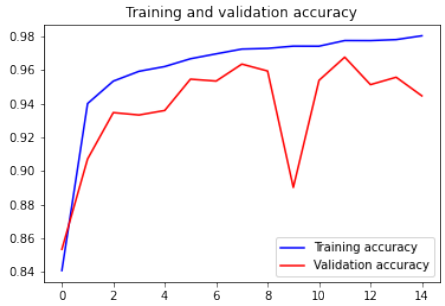

Figure 4.16: Training and Validation accuracy

Figure 4.17: loss and accuracy of MNIST on PointNet

Plot of input and output of T-Net:

The input and output of Tnet in the architecture for MNIST dataset is shown in figure 4.18.

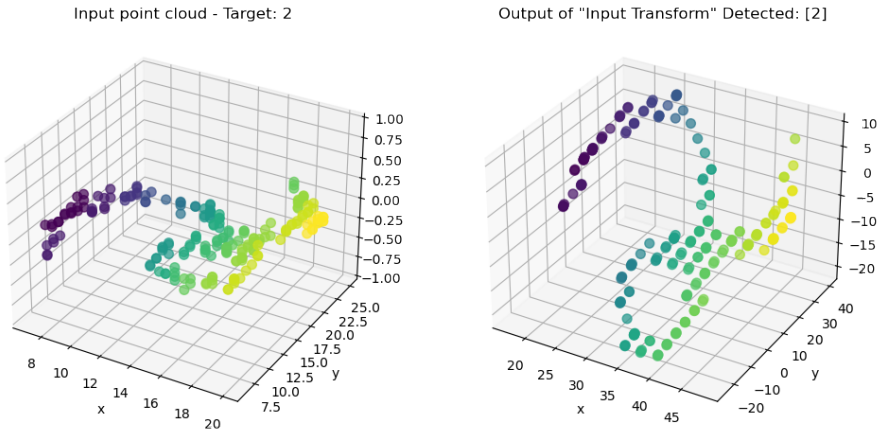

Figure 4.18: input and output of T-Net

Testing Invariance of Permutations:

We can see that the shuffled distribution of points and non-shuffled gave the similar prediction label for this sample of 4 in figure 4.19. but when tested for all the test samples, the result is not the same. it went wrong for some samples.

There seems to be around 10 samples where this invariance is failing to classify the correct label. This one of the samples is shown in figure 4.20

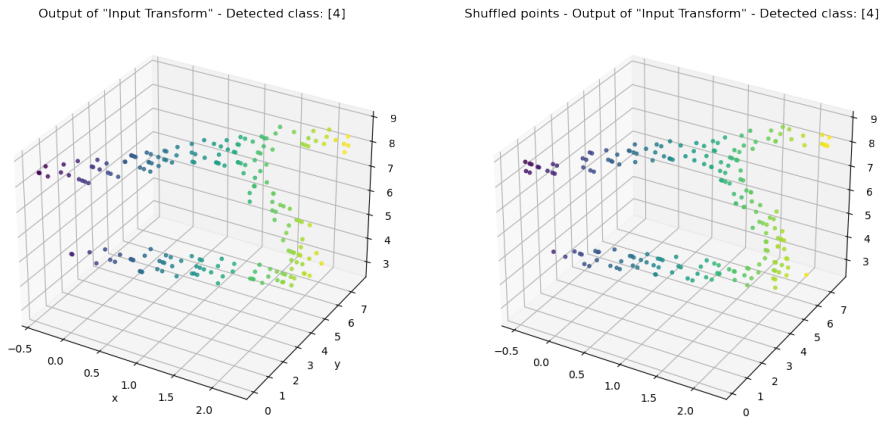

Figure 4.19: Testing Invariance of Permutations

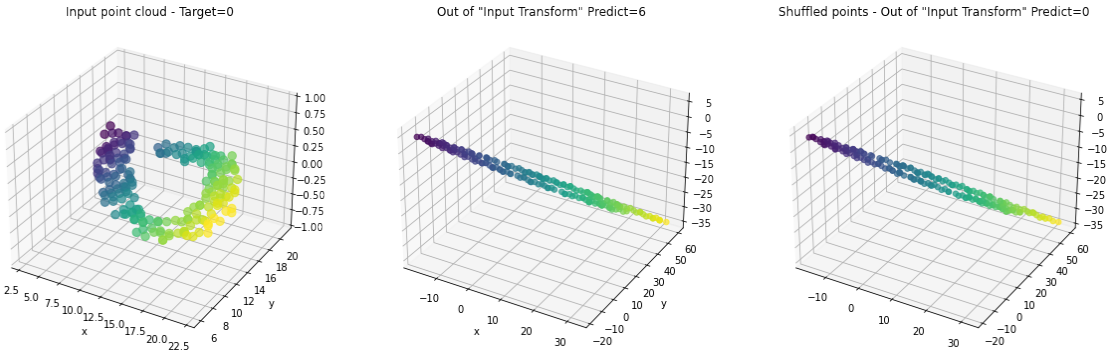

Figure 4.20: failed invariance of Permutations

Visualizing PointNet Critical points:

PointNet learns to condense a point cloud input into a small number of essential points, which roughly correlates to the visual skeleton of an item or object. The point sets that made up the majority of the pooled feature are the crucial ones. The figure in 4.21 shows input point cloud, critical points visualization with predicted label when input is shuffled and unshuffled. The figure shows predicted label of 4 when unshuffled in second plot and shuffled with predicted label 9 in third plot. This figure shows clear influence of disparity in three visualizations with critical points.

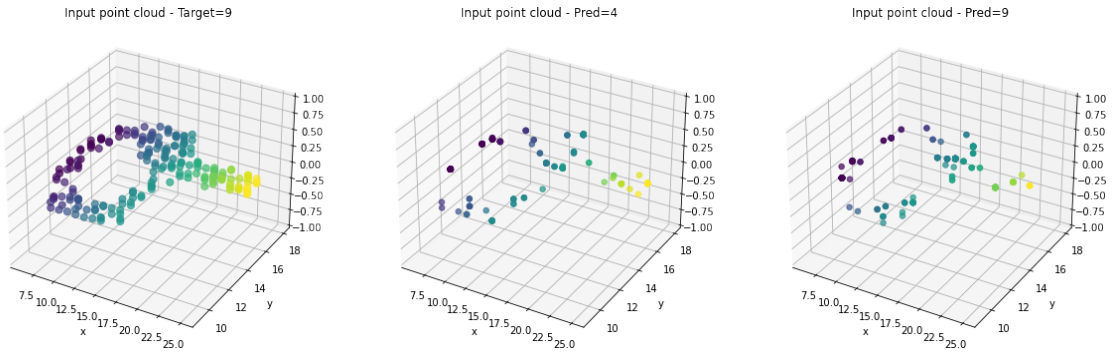

Figure 4.21: Visualizing PointNet Critical points 3d MNIST

4.2.3 Grayscale CIFAR10

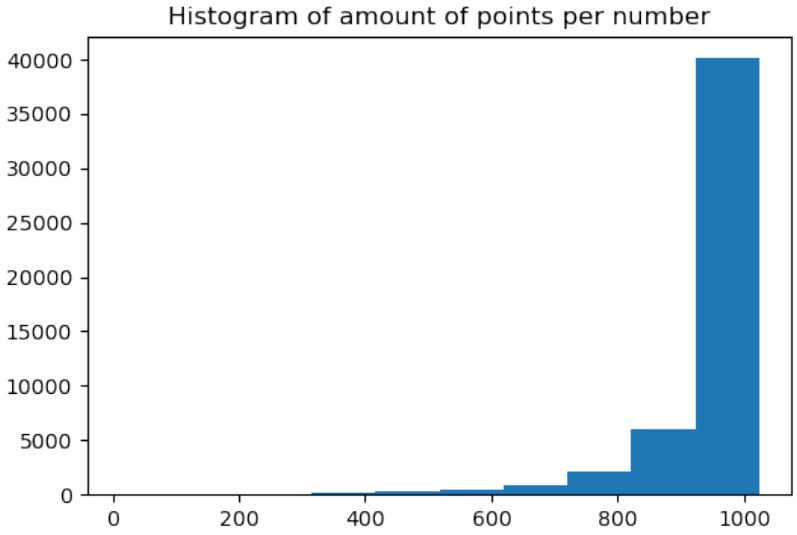

Figure 4.22: Histogram of points in CIFAR10

for Grayscale CIFAR10 or CIFAR10 to be inputted to 3d pointnet, we use the method in 3.2.1. CIFAR10 or Grayscale CIFAR10 give the same accuracy and losses as we get the same pixel indices from both. From histogram in figure 4.22, we get the maximum of length of points of all the CIFAR10 dataset converted to 2d pixel coordinates. from this maximum value, we learn to make every image with 2d pixel coordinates to reach this maximum number of pixel coordinates. Now, we need to make a third coordinate to convert 2d to 3d data. for this, we need to append a small gaussian noise as a pixel coordinate to the 2d image with pixel coordinates. Now a 3d image is achieved as shown in figure 4.23.

then network and training goes as similar to network training in Pointnet for 3d data in section 4.2.1. The results from training and validation are shown below in

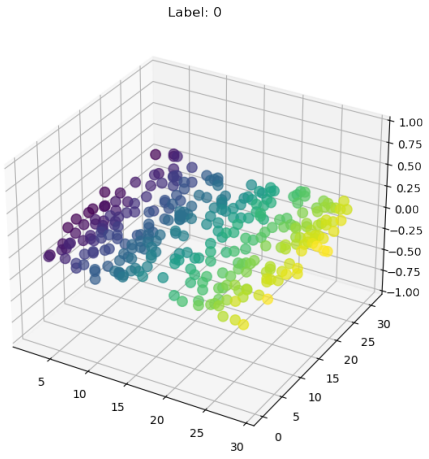

Figure 4.23: 3d data achieved from CIFAR10 2d image

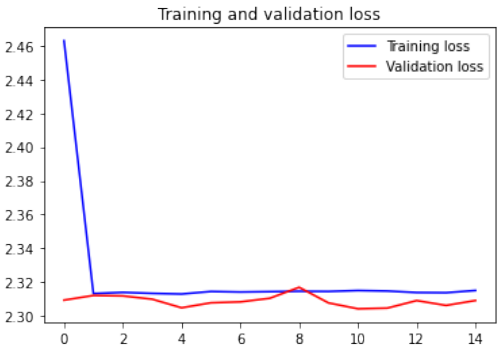

Figure 4.24: Training and Validation loss

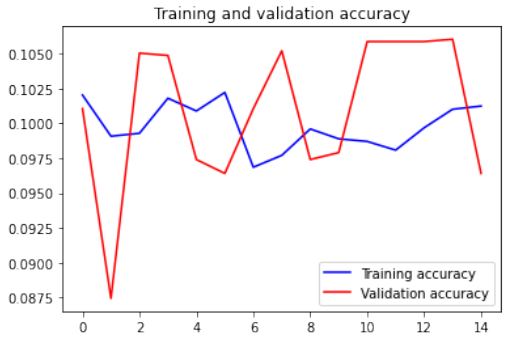

Figure 4.25: Training and Validation accuracy

Figure 4.26: loss and accuracy of CIFAR10 on PointNet

figure 4.26 The testing of CIFAR10 converted to 3d data on pointnet architecture gave an accuracy of **0.10** or **10.1** percent.

Plot of input and output of T-Net:

The input and output of Tnet in the architecture for CIFAR10 dataset is shown in figure 4.27.

Testing Invariance of Permutations:

We can see that the shuffled distribution of points and non-shuffled gave the similar prediction label for this sample of frog class in figure 4.28. but when tested for all the test samples, the result is not the same. it went wrong for some samples.

There seems to be very less accuracy where this invariance is failing to classify the correct label. This one of the samples is shown in figure 4.29.

Visualizing PointNet Critical points:

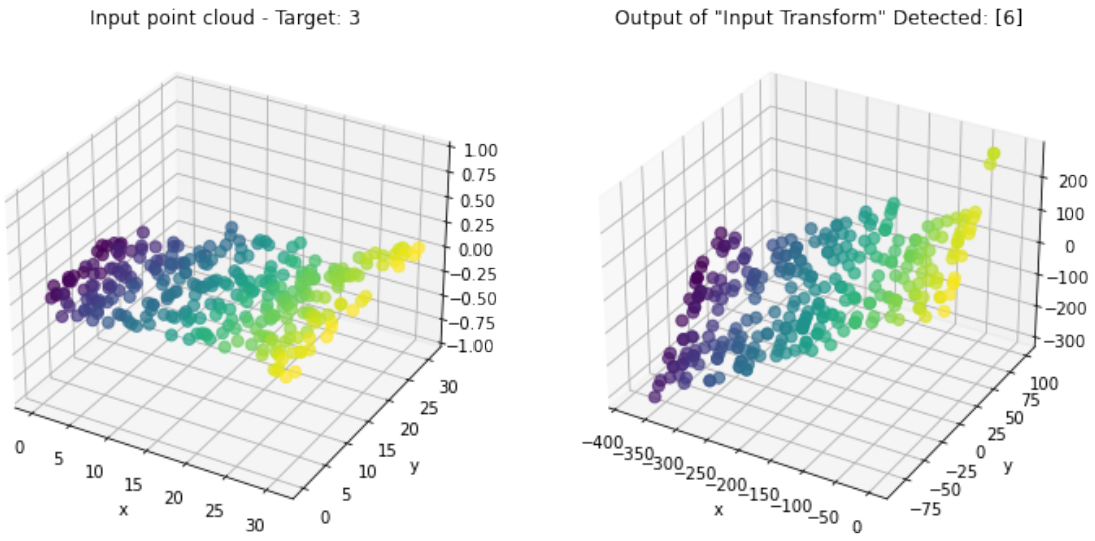

Figure 4.27: input and output of T-Net with input of cat point cloud but got wrong label

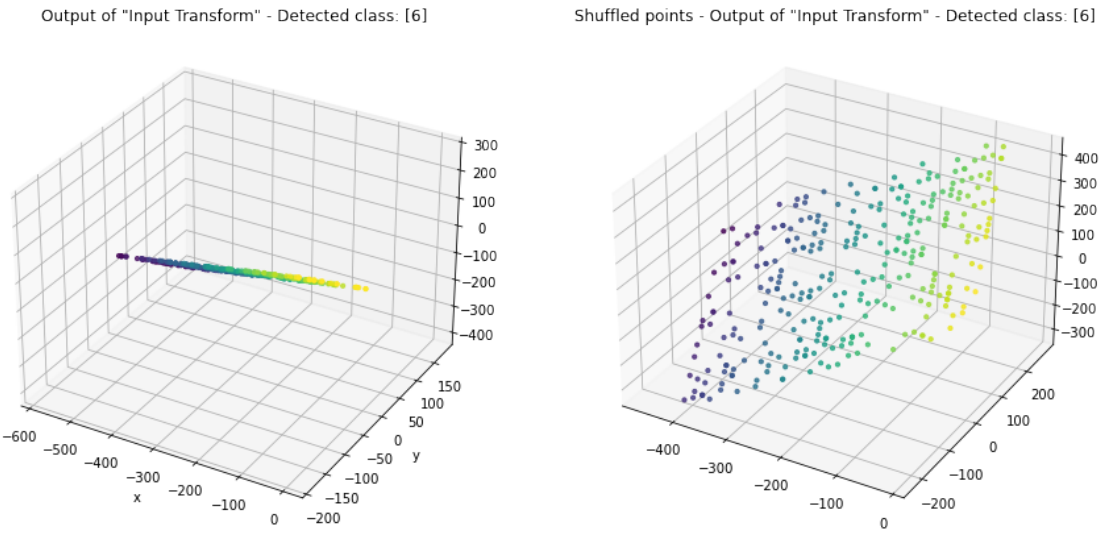

Figure 4.28: Testing Invariance of Permutations of Frog point cloud

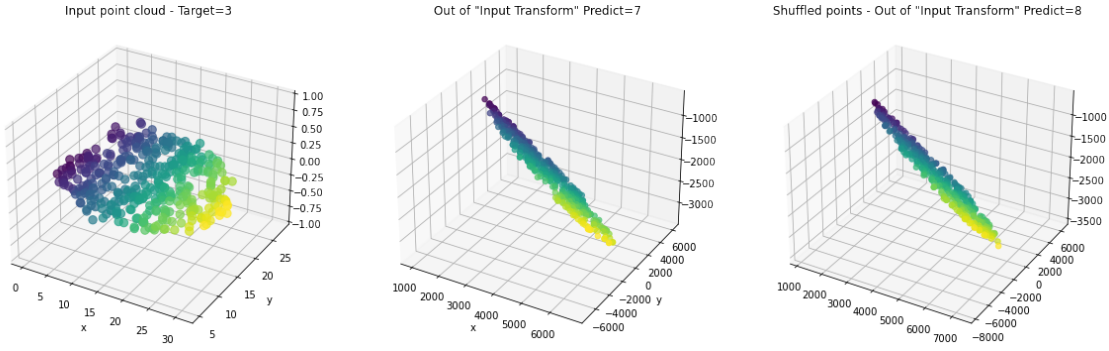

Figure 4.29: failed invariance of Permutations

PointNet learns to condense a point cloud input into a small number of essential points, which roughly correlates to the visual skeleton of an item or object. The point sets that made up the majority of the pooled feature are the crucial ones. The figure in 4.30 shows input point cloud, critical points visualization with predicted label when input is shuffled and unshuffled. The figure shows predicted label of 4 when unshuffled in second plot and shuffled with predicted label 9 in third plot. This figure shows clear influence of disparity in three visualizations with critical points.

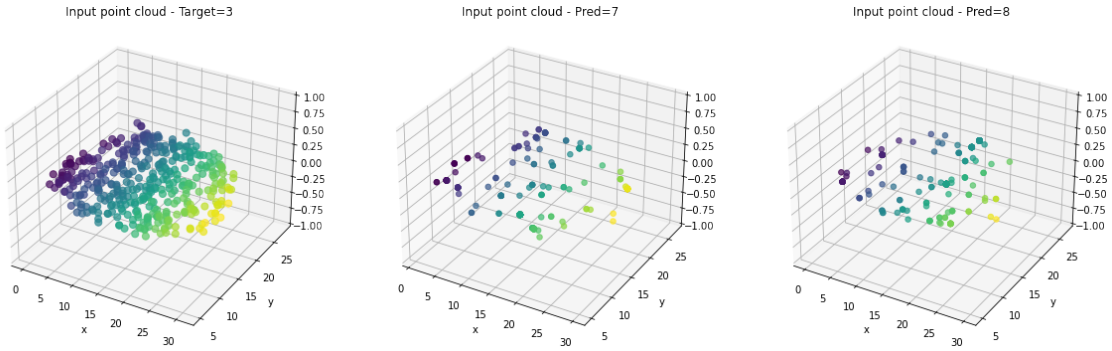

Figure 4.30: Visualizing PointNet Critical points 3d CIFAR10

3d PointNet network				
Dataset	Train loss	Train accuracy	Test loss	Test accuracy
MODELNET40	41.51	80.7724	118.18	81.9822
MNIST	0.95	98.1	1.25	94.98
GrayscaleCIFAR	2.3137	10.1	2.3055	10.11

Table 4.2: Performance table of 3d PointNet network for 15 epochs on whole train and test datasets

4.3 2D PointNet Network

We have 2d dataset of MNIST here and need to reconstruct a third dimension inorder to use the 3d point cloud architecture. For this we need to preprocess the 2d dataset. It is preprocessed by appending a null or zero z coordinate to the 2d data and now resulting in three coordinate data. We need to apply some normalisation methods like random noise, random horizontal and vertical flip, normalise the three coordinates in a range.

4.3.1 MNIST

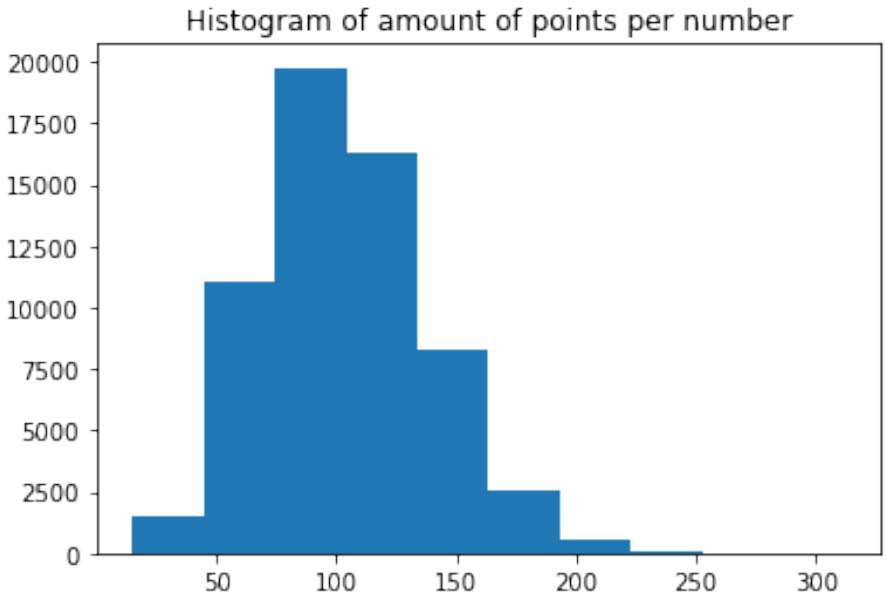

Figure 4.31: Histogram of points in MNIST

From histogram in figure 4.31, we get the maximum of length of points of all the MNIST dataset converted to 2d pixel coordinates. from this maximum value, we learn to make every image with 2d pixel coordinates to reach this maximum number of pixel coordinates. Now a 2d image of pixel coordinates is achieved as shown in figure 4.32.

then network and training goes as similar to network training in Pointnet for 3d data in section 4.2.1. The results from training and validation are shown below in figure 4.35 The testing of MNIST converted to 2d pixel data on pointnet architecture gave an accuracy of **0.932** or **93.2** percent.

Plot of input and output of T-Net:

The input and output of Tnet in the architecture for MNIST dataset is shown in figure 4.36.

Testing Invariance of Permutations:

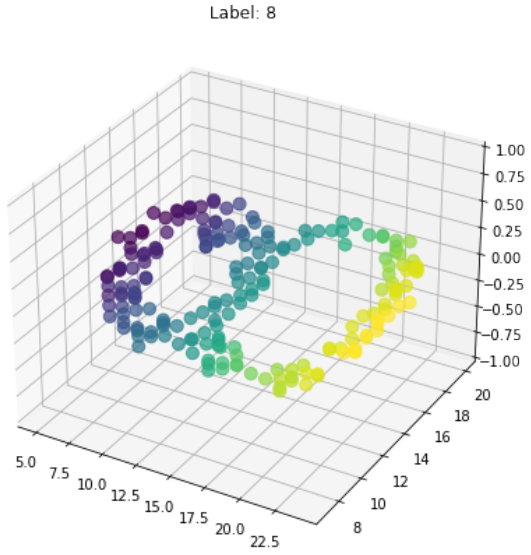

Figure 4.32: 2d pixel data achieved from MNIST 2d image

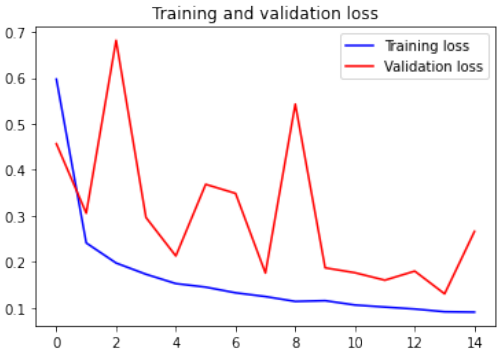

Figure 4.33: Training and Validation loss

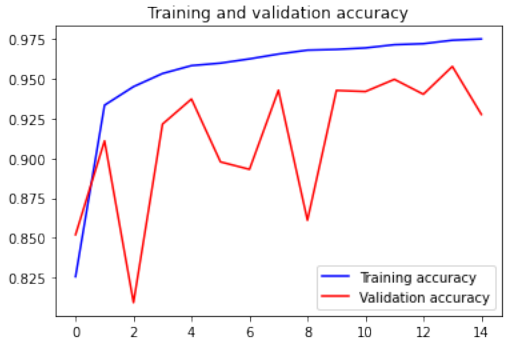

Figure 4.34: Training and Validation accuracy

Figure 4.35: loss and accuracy of MNIST on PointNet

We can see that the shuffled distribution of points and non-shuffled gave the similar prediction label for this sample of 4 in figure 4.37. but when tested for all the test samples, the result is not the same. it went wrong for some samples.

There seems to be around 10 samples where this invariance is failing to classify the correct label. This one of the samples is shown in figure 4.38

Visualizing PointNet Critical points:

PointNet learns to condense a point cloud input into a small number of essential points, which roughly correlates to the visual skeleton of an item or object. The point sets that made up the majority of the pooled feature are the crucial ones. The figure in 4.39 shows input point cloud, critical points visualization with predicted label when input is shuffled and unshuffled. The figure shows predicted label of 4 when unshuffled in second plot and shuffled with predicted label 9 in third plot. This figure shows clear influence of disparity in three visualizations with critical points.

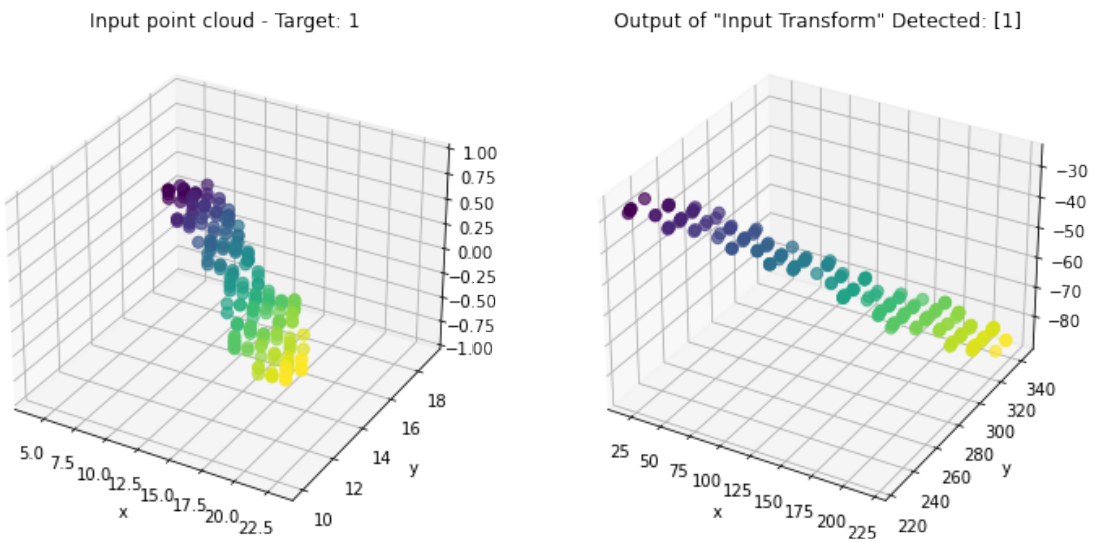

Figure 4.36: input and output of T-Net

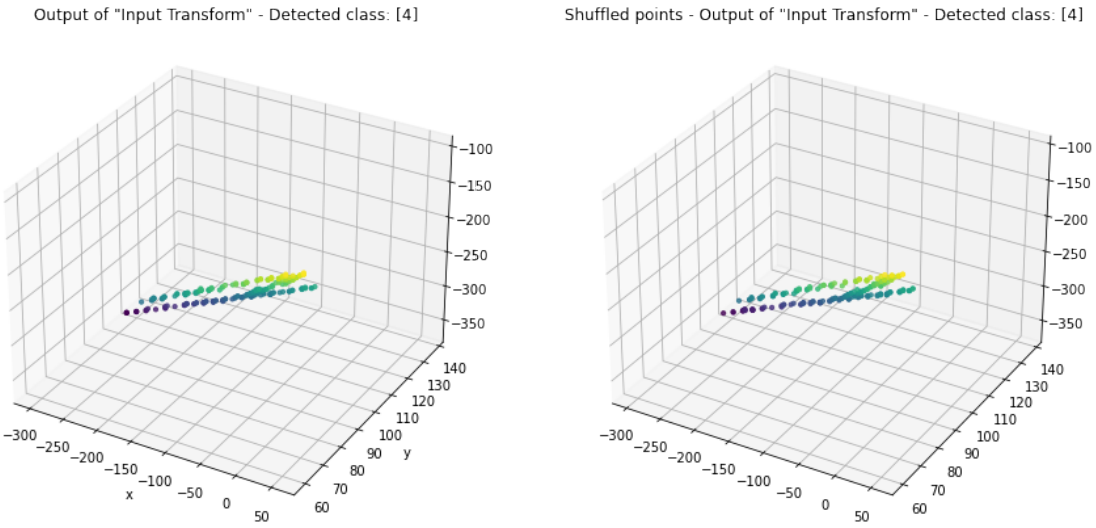

Figure 4.37: Testing Invariance of Permutations

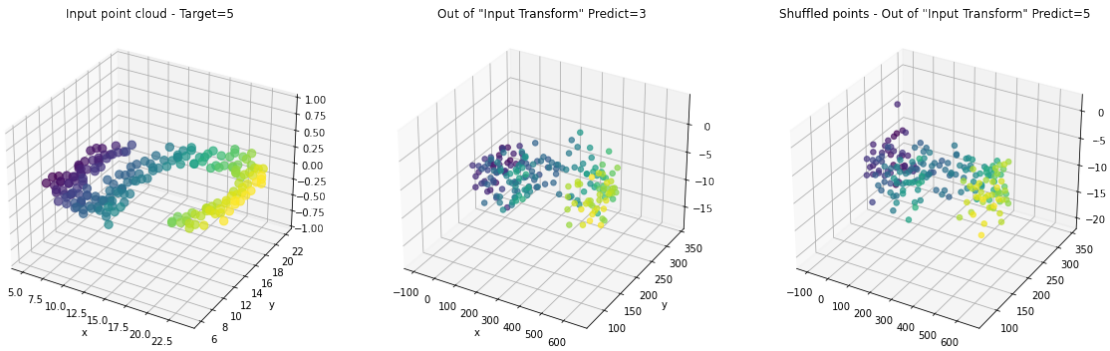

Figure 4.38: failed invariance of Permutations

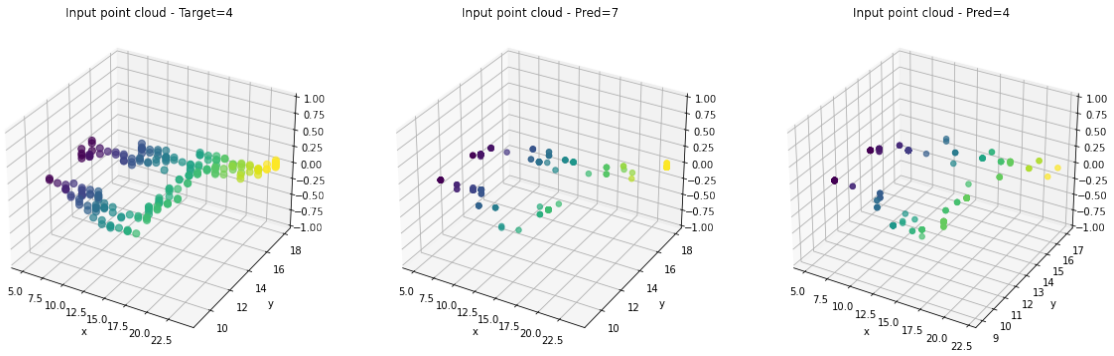

Figure 4.39: Visualizing PointNet Critical points 2d pixel MNIST

4.3.2 Grayscale CIFAR10

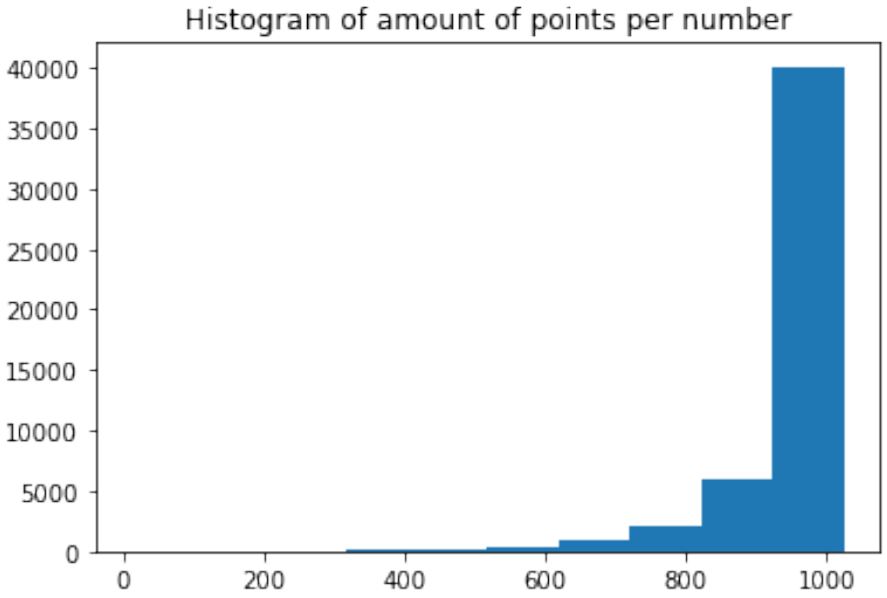

Figure 4.40: Histogram of points in Grayscale CIFAR10

From histogram in figure 4.40, we get the maximum of length of points of all the MNIST dataset converted to 2d pixel coordinates. from this maximum value, we learn to make every image with 2d pixel coordinates to reach this maximum number of pixel coordinates. Now a 2d image of pixel coordinates is achieved as shown in figure 4.41.

then network and training goes as similar to network training in Pointnet for 3d data in section 4.2.1. The results from training and validation are shown below in figure 4.44 The testing of Grayscale CIFAR10 converted to 2d pixel data on pointnet architecture gave an accuracy of **0.117** or **11.7** percent.

Plot of input and output of T-Net:

The input and output of Tnet in the architecture for MNIST dataset is shown in figure 4.45.

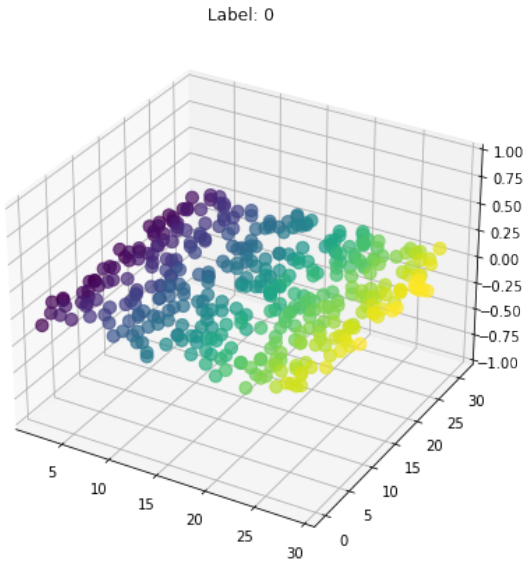

Figure 4.41: 2d pixel data achieved from GrayscaleCIFAR10 2d image

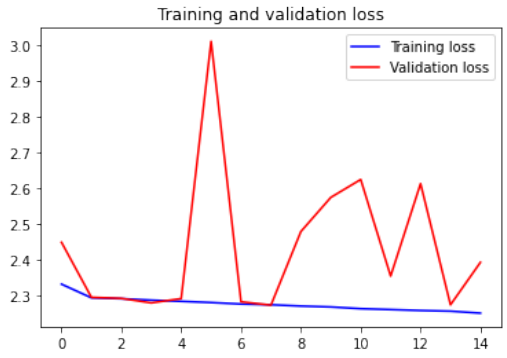

Figure 4.42: Training and Validation loss

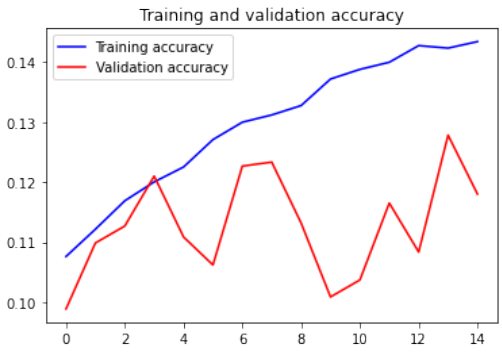

Figure 4.43: Training and Validation accuracy

Figure 4.44: loss and accuracy of GrayscaleCIFAR 10 on 2d PointNet

Testing Invariance of Permutations:

We can see that the shuffled distribution of points and non-shuffled gave the similar prediction label for this sample of 4 in figure 4.46. but when tested for all the test samples, the result is not the same. it went wrong for some samples.

There seems to be around 10 samples where this invariance is failing to classify the correct label. This one of the samples is shown in figure 4.47

Visualizing PointNet Critical points:

PointNet learns to condense a point cloud input into a small number of essential points, which roughly correlates to the visual skeleton of an item or object. The point sets that made up the majority of the pooled feature are the crucial ones. The figure in 4.48 shows input point cloud, critical points visualization with predicted label when input is shuffled and unshuffled. The figure shows predicted label of 4

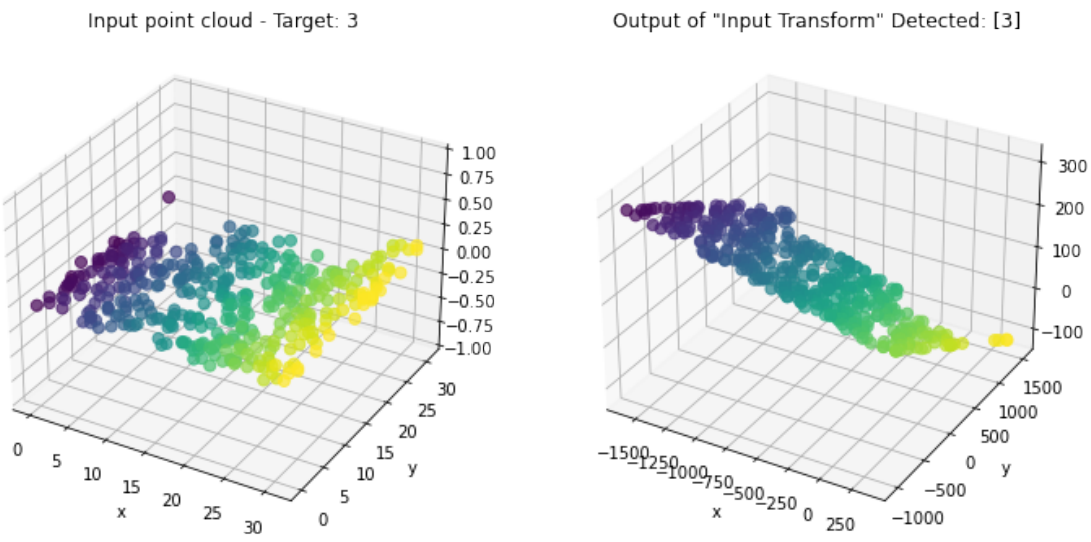

Figure 4.45: input and output of T-Net

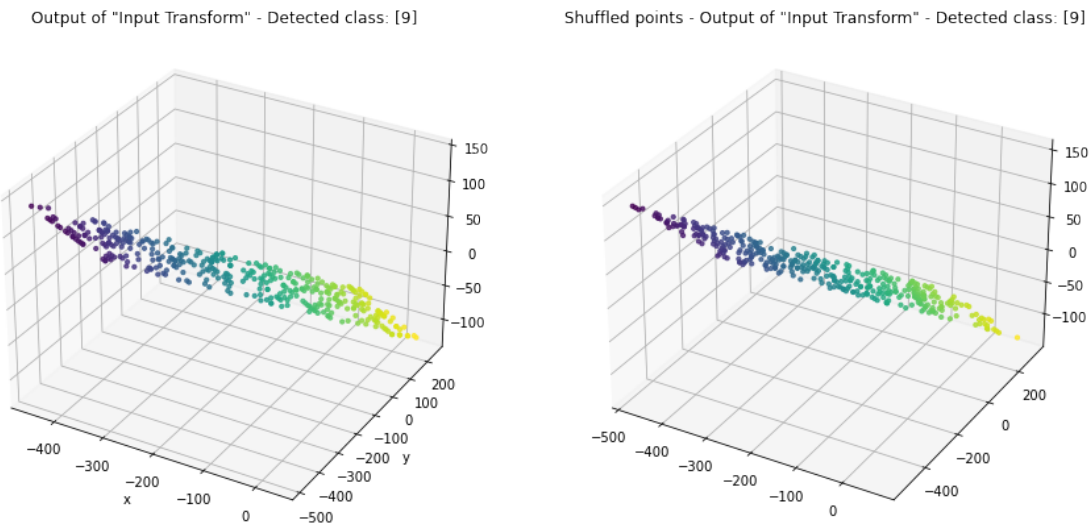

Figure 4.46: Testing Invariance of Permutations

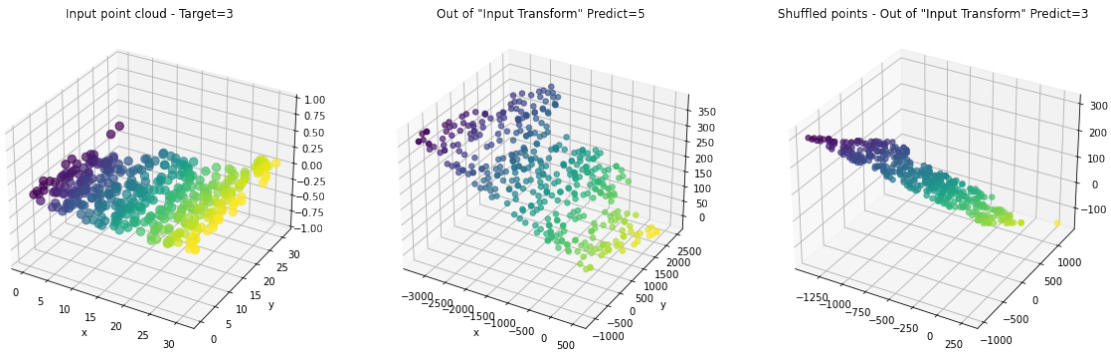

Figure 4.47: failed invariance of Permutations

when unshuffled in second plot and shuffled with predicted label 9 in third plot. This figure shows clear influence of disparity in three visualizations with critical points.

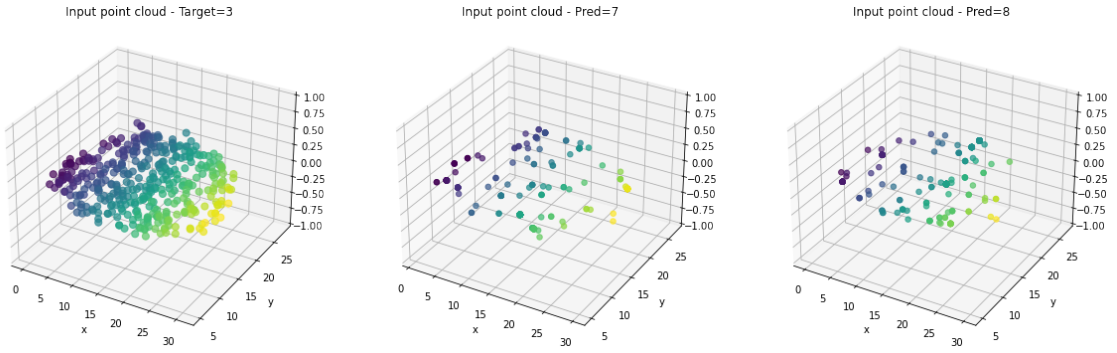

Figure 4.48: Visualizing PointNet Critical points 2d pixel Grayscale CIFAR10

2d PointNet network				
Dataset	Train loss	Train accuracy	Test loss	Test accuracy
MNIST	0.83	97.9	10.69	97.54
GrayscaleCIFAR	49.86	9.8	148.35	9.81

Table 4.3: Performance table of 2d PointNet network for 15 epochs on whole train and test datasets

5 Conclusions

5.1 Summary

In this studienarbeit report, described the cutting-edge deep neural network PointNet, which directly consumes point cloud. The PointNet network offers a unified method for a variety of 3D recognition tasks, such as object classification, component segmentation, and semantic segmentation, while outperforming state-of-the-art systems on common benchmarks. Additionally, I offered conceptual interpretations and visualizations to aid in the comprehension of the PointNet network.

I used the PointNet network design to analyze 2D data from the MNIST, grayscale CIFAR10, and CIFAR10 datasets, and the results were comparable to or superior to state-of-the-art on common benchmarks. Grayscale CIFAR10 working poor with 3D and 2D pointnet. MNIST works very well almost equally efficient with CNN.

5.2 Future Work

Future work would be to work with R,G,B color data and original color image dataset. We could try removing T-Net transformation from the PointNet and analyse the performance with MNIST, grayscale CIFAR10, and CIFAR10 datasets.

List of Figures

List of Tables

Bibliography

[1] Saad Albawi; Tareq Abed Mohammed; Saad Al-Zawi. „Understanding of a convolutional neural network“. In: *2017 International Conference on Engineering and Technology (ICET)*. 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[2] Laith Alzubaidi et al. „Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions“. In: *Journal of big Data* 8 (2021), 1, pp. 1–74.

[3] Li Deng „The mnist database of handwritten digit images for machine learning research“. In: *IEEE Signal Processing Magazine* 29 (2012), 6, pp. 141–142.

[4] Kun Hu et al. „Deep learning techniques for in-crop weed identification: A Review“. In: *arXiv preprint arXiv:2103.14872* (2021).

[5] Adam Huang; Gregory M Nielson. „Surface approximation to point cloud data using volume modeling“. In: *Data Visualization*. Springer. 2003, pp. 333–343.

[6] Max Jaderberg et al. *Spatial Transformer Networks*. 2015. DOI: 10.48550/ARXIV.1506.02025. URL: https://arxiv.org/abs/1506.02025.

[7] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[8] Keiron O'Shea; Ryan Nash „An introduction to convolutional neural networks“. In: *arXiv preprint arXiv:1511.08458* (2015).

[9] Charles R. Qi et al. *Volumetric and Multi-View CNNs for Object Classification on 3D Data*. 2016. DOI: 10.48550/ARXIV.1604.03265. URL: https://arxiv.org/abs/1604.03265.

[10] Charles R Qi et al. „Volumetric and multi-view cnns for object classification on 3d data“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 5648–5656.

[11] Charles R Qi et al. „Pointnet: Deep learning on point sets for 3d classification and segmentation“. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[12] Manolis Savva et al. „Shrec16 track: largescale 3d shape retrieval from shapenet core55“. In: *Proceedings of the eurographics workshop on 3D object retrieval*. Vol. 10. 2016.

[13] Hang Su et al. *Multi-view Convolutional Neural Networks for 3D Shape Recognition*. 2015. DOI: 10.48550/ARXIV.1505.00880. URL: https://arxiv.org/abs/1505.00880.

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

[15] Qian-Yi Zhou; Jaesik Park; Vladlen Koltun „Open3D: A modern library for 3D data processing“. In: *arXiv preprint arXiv:1801.09847* (2018).