ModelNet10 Classification with PointNet

Scott Mau Nguyen

May 22, 2022

1 Introduction

In the recent years, there has been an increasing focus towards self-driving vehicles, both in the research domain and in industry. One of the most important features of self-driving cars, and autonomous robots in general, is an understanding of the surrounding 3D world. With that said, autonomous vehicles rely heavily on computer vision, utilizing lidar, radar, and camera systems, to probe the local environment to detect and classify objects. In the case of lidar sensing, points are sampled from the environment using a laser; a cloud of these points are generated to build up a 3D graphical model. The focus of this paper is on 3D object classification using point cloud data.

2 Problem

In order to classify 3D objects in a scene, the geometrical structure must be characterized given a series of data points. Three different methods are used for point cloud classification: multi-view methods, point-based methods, and volumetric methods [1]. Multi-view methods hinge on the idea of projecting a point cloud onto a series of 2D images. Volumetric methods take a different approach, the point cloud is voxelized into a series of voxels, developing 3D grids from the point cloud; a CNN can then be applied to this representation for object classification. Point-based methods model the points of the point cloud directly and use a global feature extracted from these points for classification. The focus of this paper is on the PointNet Architecture, proposed by Qi et al. in 2017 [2].

3 PointNet Architecture

The PointNet Architecture uses point-based methods to accomplish 3D point cloud classification and segmentation; this paper only focuses on classification. The goal of this architecture is to extract a global feature from the raw point cloud and then use this discriminative global feature to perform classification. A depiction of the PointNet Architecture is shown below in Figure 1, this was taken directly from the PointNet paper [2]. The highlighted blue section is the classification network and will be the focus of this paper.

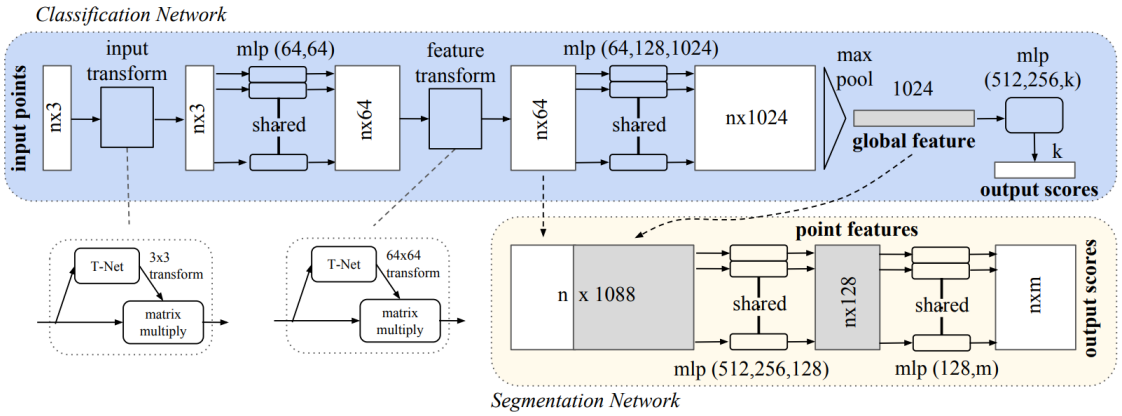

Figure 1: PointNet Architecture [1]

4 Properties of Point Cloud Data

A set of 3D points are the inputs to the network and these points are: unordered and transformation invariant [2]. Since the points are unordered, the network should be able to generate a result without a dependency on the permutation of the input points, therefore the network must be permutation invariant. Invariance to rigid transformations is another important characteristic for the architecture to have, since rotating and translating the object should have no effect on classification.

4.1 Permutation Invariance

In order to map point cloud data to classified objects, the following function is approximated:

$$f(\{x_1, ..., x_n\}) \approx g(h(x_1), ..., h(x_n)) \tag{1}$$

where $x_1, ..., x_n$ represents n point cloud data points. The function, g, is a symmetric function and by using a symmetric function on a collection of functions, $h(x_1), ..., h(x_n)$, which depend on the input points, the classification network becomes permutation invariant. Thus, symmetry is used to ignore how the inputs are fed into the network. The function, h, is approximated using a shared multi-layer perceptron network and the symmetric function, g, is expressed by a max pooling function; this was shown to work well as discovered in the PointNet paper [2]. The max pooling function generates a global feature for a given input and a multi-layer perceptron is then trained to classify objects based on this global feature, as shown on the far right of Figure 1 [2].

4.2 Geometric Transformation Invariance

Classification of 3D objects should be independent of rotation and translation. The PointNet archicture deals with this problem by using joint alignment networks to approximate affine transformation matrices. The input points are fed through a mini transformation network, as shown in Figure 1. This first mini transformation network approximates a spatial transformation matrix that transforms the input points into a canonical space so that features can be extracted regardless

of the translation and rotation of the point clouds. The same mini transformation network is used after feature extraction, a transformation matrix is approximated to align features from different point clouds [2]. There is an issue with this approach since the dimension of the spatial transformation matrix is a lot smaller than the dimension of the feature transformation matrix; this makes optimization much more difficult as asserted by Qi et al. [2]. To improve the performance of optimization a regularization term is added to the softmax loss function and can be expressed as:

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

where I is the identity matrix and A is the feature transformation matrix. This regularizer effectively keeps the feature transformation matrix close to orthogonal so that there will be no information loss [2].

5 Converting ModelNet10 Dataset to Point Cloud Data

The ModelNet10 dataset is composed of 10 classes: toilet, table, bed, desk, dresser, chair, monitor, night stand, bathtub, and sofa. Each class is partitioned into training and testing data as shown in Table 1.

Class Name	Train	Test
toilet	344	100
table	392	100
bed	515	100
desk	200	86
dresser	200	86
chair	889	100
monitor	465	100
night stand	200	86
bathtub	106	50
sofa	680	100

Table 1: ModelNet10 Dataset: Training and Testing Data

The objects in the ModelNet10 dataset were converted into point clouds using the open3d library. From each 3D object mesh, 1024 points were uniformly sampled to form a representative point cloud; the point clouds were all normalized to ensure robustness. Once the point clouds were extracted from the 3D meshes and partitioned into training and testing data, both the training and testing data were randomly shuffled so that the classification network doesn't model the pattern in which objects are fed into the system. Also, to improve robustness and generalization of the model, the individual points of each point cloud were uniformly jittered by +/- 0.006.

6 Mini Transformation Network

As depicted in Figure 1, the input data is first fed into a mini transformation network to transform the points into a canonical space, once again this is meant to approximate a spatial transformation

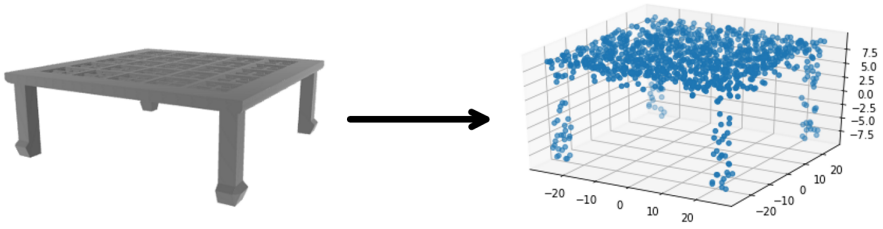

Figure 2: Converting 3D mesh to point cloud

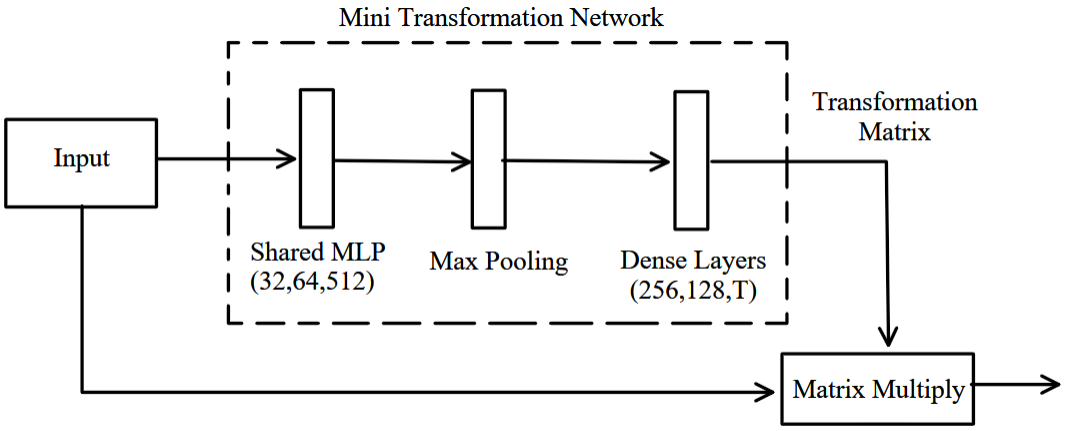

Figure 3: Mini Transformation Network

of the point cloud. The network is composed of a shared multi-layer perceptron network, a max pooling function, and three dense layers; the mini transformation network is depicted in Figure 3. The shared MLP network is composed of a 1-dimensional convolution (with a kernel size of 1 and valid padding), batch normalization (with no momentum), and ReLU activation. Each fully connected dense layer also has batch normalization and ReLU activation, except the last dense layer of size 'T', which is the size of the transformation matrix that is being approximated. A regularization loss, given by Equation 2, was added to the last dense layer with a weight of 0.001 to constrain the transformation to be orthogonal.

7 Main Network

Once the input data is fed into the input transformation network, it is passed to a shared MLP network for feature extraction. The features are then passed to a feature transformation network for feature alignment. After the data features are aligned, the data is passed to another shared MLP and then a symmetric max pooling function is used to extract the global feature of the point cloud. The global feature is used to train an MLP for classification. Batch normalization and ReLU activation are used for each layer in the network, except the output layer. Dropout regularization

is also used on the last MLP network with a keep ratio of 0.7. The main network architecture used in this project is depicted in Figure 4. Note that the number of hidden units in each layer were reduced by a factor of 2 in order to make training faster; this is acceptable since the ModelNet10 dataset is much smaller than the dataset used in the PointNet paper [2].

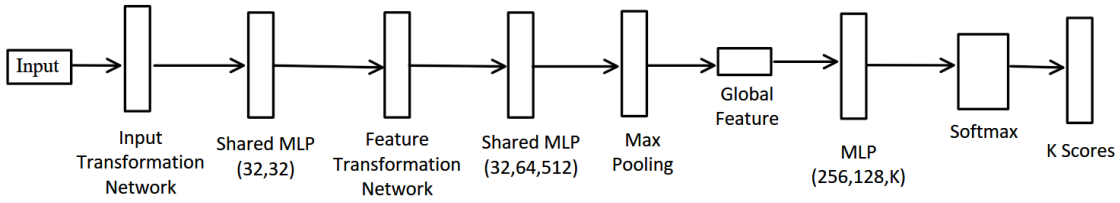

Figure 4: Main Network

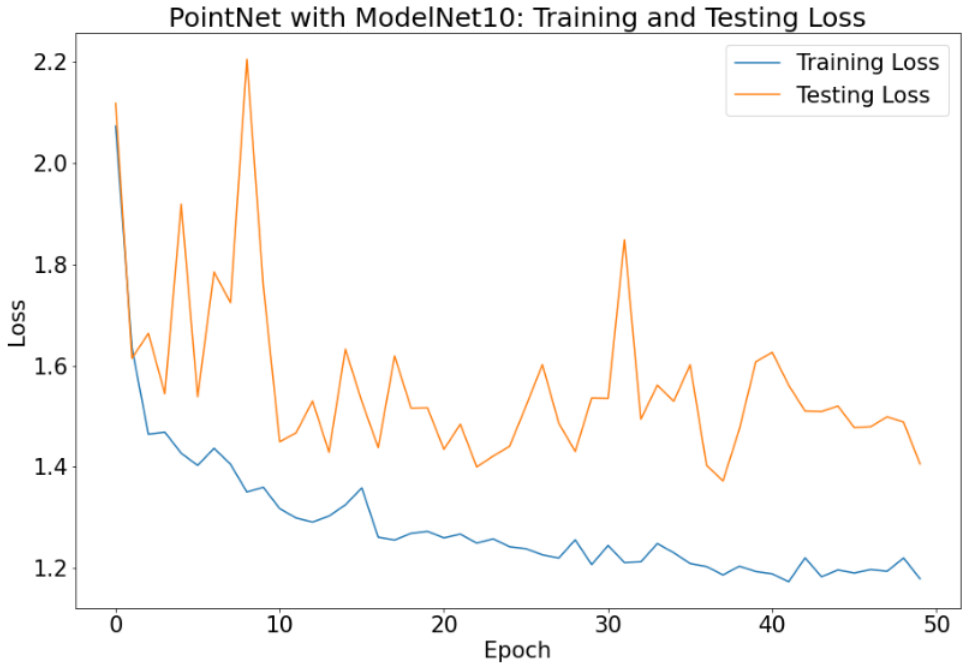

Figure 5: Sparse categorial crossentropy loss for training (orange) and validation set (blue)

8 Results

After constructing the PointNet architecture using TensorFlow, the model was trained on the ModelNet10 training data using Adam optimizer and a learning rate of 0.001. A batch size of 32 was used and the model was trained for 50 epochs. The sparse categorial crossentropy loss was

recorded for both the training and validation data and is depicted in Figure 6. The loss continued to decay up to 50 epochs for both the training and validation set, however, the loss didn't change much after 50 epochs.

8.1 Visualization & F1 Score

Using the trained model on the ModeNet10 test data, the model performed with an 89.5% accuracy and the F1 score was about 0.893 (Figure 2). A series of the point cloud predictions are shown in Figure 6.

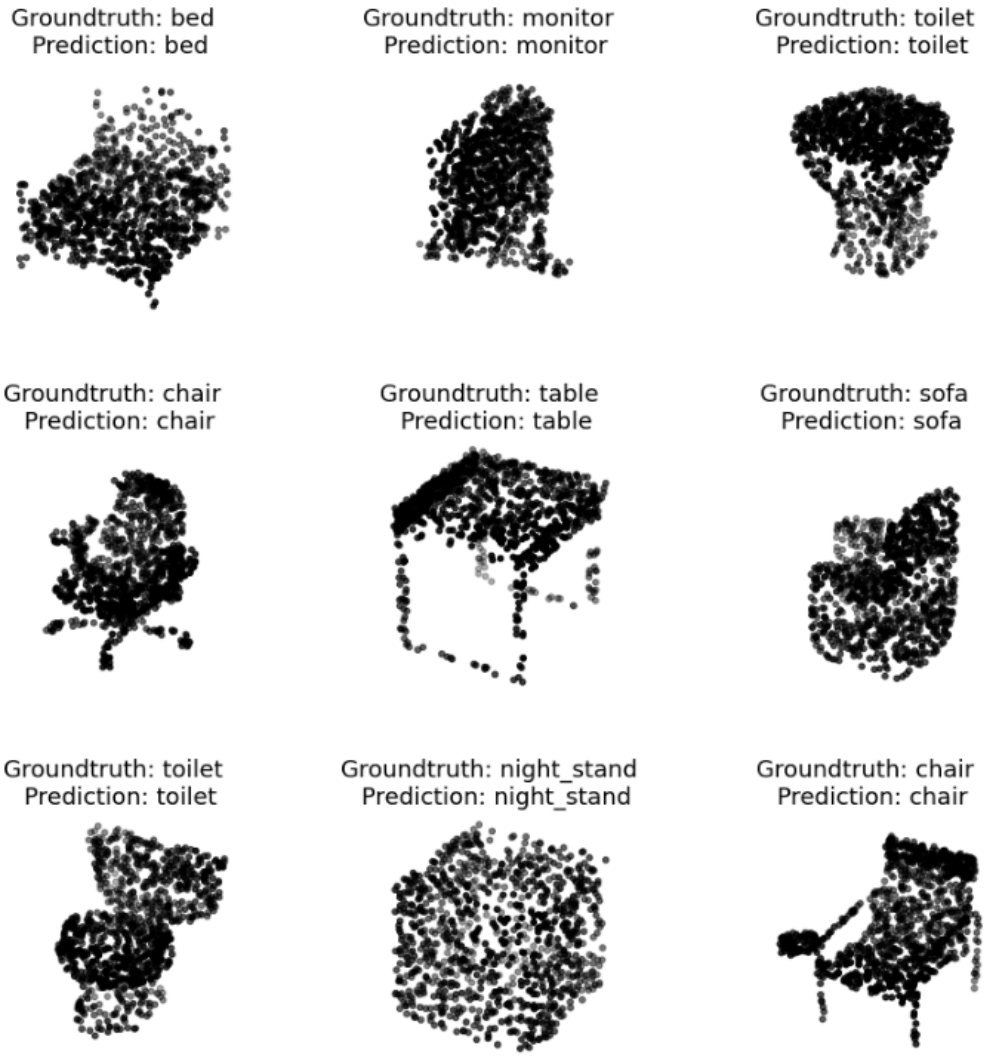

Figure 6: Point cloud predictions on test data

Performance Metrics	Test
Accuracy	0.895
F1 Score	0.893

Table 2: Performance metrics for ModelNet10 test data

8.2 Generalization: Extension to Online CAD Models

The model worked well with the test data from the ModelNet10 dataset, however it did not work well when trying to extend this to classification of CAD models found on the internet. For each class, 3D CAD Models were found on the website, Thingiverse, and were used to build up a dataset to test the generalization abilities of the trained model. The test set generated consisted of 86 objects (Table 3), The model did really well at classifying tables, however, the overall accuracy and F1 score were very low as shown in Table 4.

Class Name	Test
toilet	6
table	11
bed	8
desk	8
dresser	7
chair	11
monitor	9
night stand	8
bathtub	8
sofa	10

Table 3: STL file dataset retrieved from Thingiverse.com

Performance Metrics	Test
Accuracy	0.232
F1 Score	0.166

Table 4: Performance metrics for STL file dataset

9 Conclusion

After training the PoinNet classification network with the ModelNet10 dataset, it can be concluded that training on this dataset does not generalize well to other objects within the same class that can be found online. To reduce this generalization error different regularizers can be introduced and experimented with to see if improvements can be made. Also, the feature space could likely be tuned to better to identify point cloud features. This analysis only involved the PointNet architecture introduced in 2017, there are many more that exist and research is currently being done to improve the performance of object classification [1].

References

[1] Yulan Guo et al. "Deep Learning for 3D Point Clouds: A Survey". In: *CoRR* abs/1912.12033 (2019). arXiv: 1912.12033. URL: http://arxiv.org/abs/1912.12033.

[2] Charles Ruizhongtai Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *CoRR* abs/1612.00593 (2016). arXiv: 1612.00593. URL: http://arxiv.org/abs/1612.00593.

Link to Github: https://github.com/scottmaunguyen/CSE676-Final-Project