Spring 2018 CSE Independent Study Report

Learning Classifiers for Point Clouds

Han Liu
hanliu@wustl.edu

May 6, 2018

Contents

1 Preface

This independent study deals with the topic of point clouds. Processing point clouds is an important task since many 3D sensors produce point clouds as outputs. During this semester, I learned and explored effective techniques to analyze point clouds and acquire information from them. Starting from studying current literature, I implemented several neural networks, understood the mechanism behind them, and improved the performance by changing architectures. In the following sections, I will first summarize the main papers I read, and then present and analyze the experiments I ran in this semester.

2 Literature Reviews

There are several effective techniques that analyze point clouds and acquire information from them. A common idea among these techniques is the application of neural networks. In this semester, I mainly studied the mechanism of PointNet [1] and its upgraded version PointNet++ [2] that take 3D point clouds as inputs for object classification and segmentation. I also investigated the Multi-view CNN method [3], which process multiple 2D colorful images for similar tasks.

2.1 PointNet

PointNet is a model that takes 3D point clouds as inputs and acquires information from the coordinates of each point. The basic idea of PointNet is to use multi-layer perceptrons to learn a representation of 1024 features of each point. The most important properties of point clouds are that the input is unordered. PointNet uses MaxPooling to process the unordered information learned from the huge point set. MaxPooling is a symmetric function, which is able to produce the same output no matter what order of the input points it takes. After the global features are learned, they are concatenated to each point so that another multi-layer perceptrons could deal with the segmentation task. Another featuer of PointNet is the application of the transform net (T-Net). A robust model must also be invariant to any geometric transformation of the input. Rotating and translating points should not change the global information we learn from the point cloud. The T-Net predicts an affine transformation matrix that can be multiplied with the coordinates. The presence of T-Nets makes the model invariant to possible geometric transformations of point clouds.

2.2 PointNet++

PointNet++ improves PointNet by capturing local features at different scales. It is challenging for PointNet to classify and segment point clouds with different density of points or generalize well for large complex scenes. A hierarchical model is introduced to learn features of neighborhoods of points at different scales. The model first samples a set of points as centroids using the farthest point sampling technique, and then groups the points within a ball of each centroids together. After sampling and grouping, a PointNet is applied to each group to capture the local feature of the neighborhood. The features of each group are then concatenated together to represent the local feature learned from that level. After applying this prodecure hierarchically for several levels, the model uses a multi-layer perceptron for the top level to predict results for classification. During segmentation tasks, the model propagates the label for each point by interploting feature values from the nearest groups at each level.

2.3 Multi-view CNN

Multi-view CNN renders a 3D shape model into several 2D images from multiple perspectives, applies convolutional neural networks to each image and combines the features learned by an element-wise MaxPooling across all of the images at each pixel. In the simple setting, 12 images are rendered from views taken from virtual cameras placed horizontally every 30 degree. In the complex setting, a virtual camera is placed at the center of each triangle of an icosahedron (20-faces polyhedron) surrounding the 3D model and takes 4 images by rotating itself every 90 degree. The most obvious reason that it performs better than models processing point clouds is that it can learn the color information contained in 2D images.

3 Implementation

I implemented the neural network from scratch according to the model described in the PointNet paper with Python and TensorFlow (1.7.0). Some abstract idea were drawn from the implementation published by the original authors on GitHub (link: `https://github.com/charlesq34/pointnet`). A big difference between my implementation and theirs is that my model can take any number of points for either training or evaluation. This is feasible because the max pooling operation does not necessarily need a fixed-size window if implemented with the `reduce_max` function and wise reshaping of the tensor.

3.1 Problems encountered

During the progress of implementing the models, I encountered several problems and bugs. One thing worth to mention is the manipulation of batch normalization. After adding the batchnorm layers to the model according to the paper, I found out that the test performance of the model remained poor while the training accuracy increased. However, this phenomenon did not appear when the `is_training` flag were set to true during evaluation phases. This is caused by different estimations of the data distribution calculated from the batches used in the two different phases. We can also solve this by lowering the momentum of batch normalization. Different momentum of batch normalization have different performance. Generally, lower momentums yield faster updating of moving mean and variance across the batch, and produce better results. To avoid the possible disturbance of this problem, I set the `is_training` flag to true in all of the following experiments.

4 Experiments

For experiments, I used a 3D MNIST dataset published on Kaggle, which does not include rotation of point clouds. The link to the dataset is `https://www.kaggle.com/daavoo/3d-mnist/data`. By default, I use batchsize 50 and run totally 22 epochs. I set the learning rate to 0.001 in the first 20 epochs and drop it by a factor of 10 in the last 2 epochs. Other settings are same as the original paper. In the following subsections, I will explain the experiments I ran with manipulations of the parameters and the conclusions that can be drawn from them.

4.1 Number of points

The inputs of the model may have different numbers of points of each point cloud for either training or evaluation or both. I built a matrix for 25 results using a permutation of [128, 256, 512, 1024, 2048] points for training and evaluation.

# of points for	Evaluation				
Training	128	256	512	1024	2048
128	87.7	89.6	90.5	89.6	89.3
256	84.6	89.9	90.5	90.6	91.2
512	84.	88.1	90.9	92.	92.4
1024	81.7	88.4	90.6	92.4	92.
2048	80.2	87.4	90.9	91.5	92.7

Table 1: Matrix of the evaluation accuracy (%)

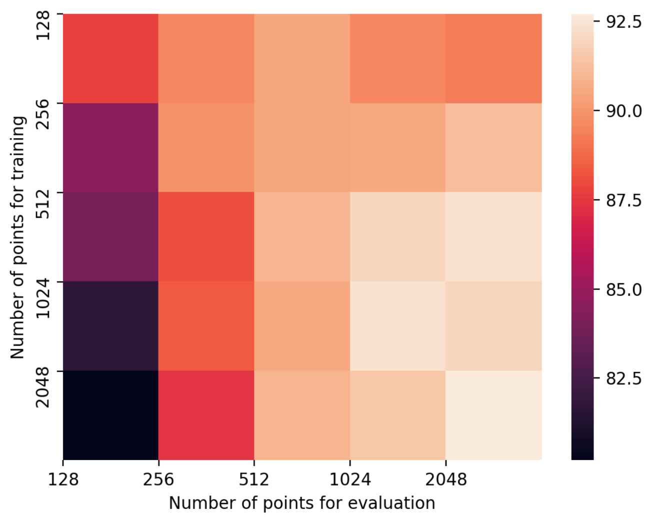

Figure 1: Heatmap of the evaluation accuracy (%)

To help us visualize the trend, I also plotted the accuracy in 3D scatter and surface plot.

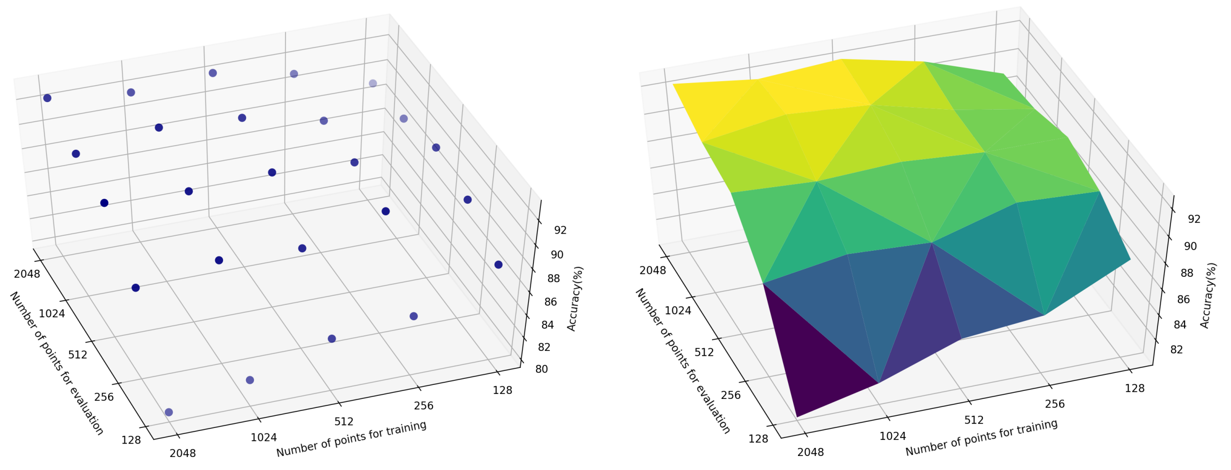

Figure 2: Scatter and surface plot of the evaluation accuracy (%)

It is not hard to see that when we have the less points for evaluation the accuracy goes down with increasing points for training, and when we have the more points for evaluation the accuracy goes up with increasing points for training. We can also see that the accuracy when we have less points for training but more points for evaluation is better than the accuracy when we have more points for training but less points for evaluation.

4.2 Size of the last MLP layer

I also investigated the influence of the size of the last MLP layer on the performance of the model. Originally, we use 1024 neurons to represent the information learned from the xyz coordinates of a point. We also tried [64, 128, 256, 512] neurons.

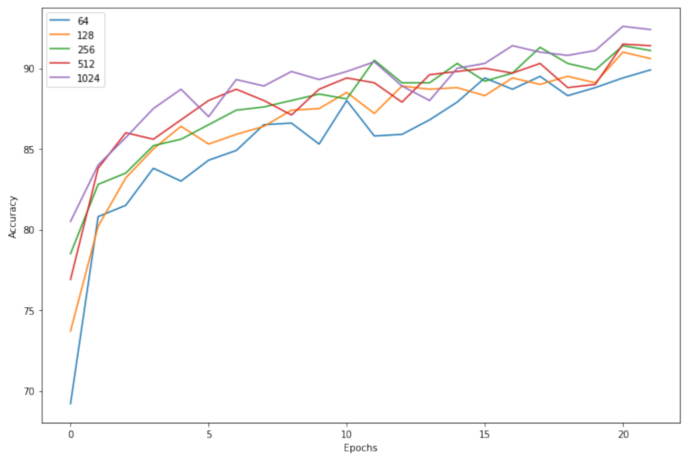

Figure 3: Evaluation accuracies (%) of different number of neurons

It is surprising to find that even with only 64 neurons the information acquired is still enough for the model to perform well (89.9% with baseline 92.4%).

4.3 Learning rate

The value of learning rates also affects the training speed and performance of the model.

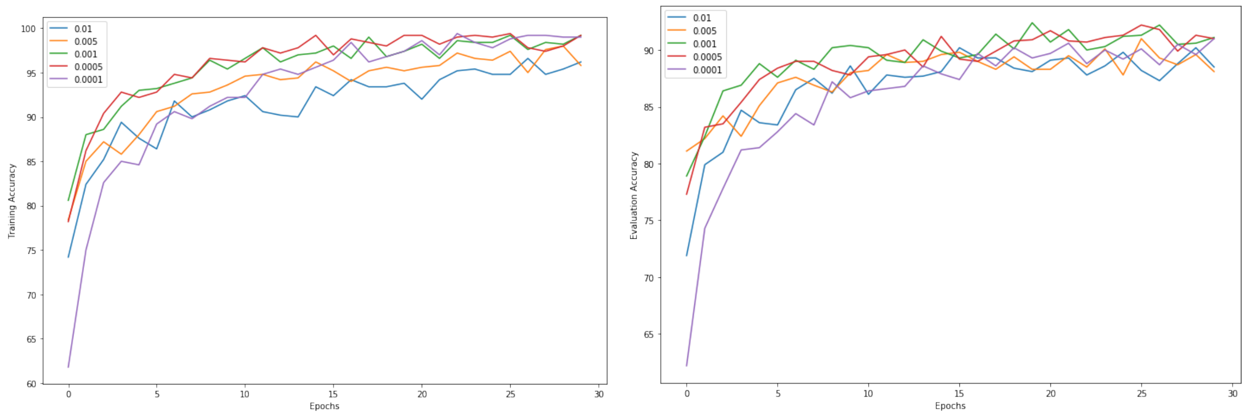

Figure 4: Training (left) and evaluation accuracies (%) of different learning rates

We can see that although the evaluation performance does not change much, the training accuracy of a large learning rate fluctuates dramatically after a few epochs (around 10 in our case). Therefore, it is good to drop the learning rate by some factor as the model trains for more epochs.

5 Summary

In this study, I have explored several leading techniques in 3D classification and specificly investigated the PointNet model, which is a notable model for processing point clouds. I also inquired into the ability of the model by running experiments on a popular dataset with different parameters. A future work may be running the model on a modified version of the dataset where 3-dimensional random rotation of point clouds are applied. We may then learn a good understanding of the importance of the transform net in the model.

This independent study is conducted under the supervision of Professor Ayan Chakrabarti. Special thanks to his insightful guidance and valuable advices.

References

[1] 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.

[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017.

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