3D Object Classification using Deep Learning

Zili Xie (zx979)
Jiaying Li (jl10919)

December 15, 2020

Abstract

In this paper, we conducted comparative experiments on three popular deep network structures: PointNet, VoxNet and Multi-View CNN that classify 3D shapes. We compared the accuracy and robustness of the 3D shape classification results with different deep network structures and further investigate the advantages and disadvantages of different classifiers by visualizing data and the experimental results. Our experiments show that thanks to the more mature two-dimensional image convolutional network, the two-dimensional multi-view method can still recognize and classify three-dimensional objects much better than those models directly trained with three-dimensional data.

1 Introduction

The classification of 3D objects has always been a hot research topic in 3D computer vision. Given hundreds to thousands of three-dimensional coordinate points, how to automatically find the category of such a geometric body among many categories is a complicated problem. This problem is mainly different from ordinary image classification problems in the following aspects.

- First, our input has changed from a four-dimensional image matrix of size $n * H * W * c$ to a three-dimensional vector set of length n, where n is the number of input points. Also, the relationship between 3d points are much subtler than the relationship between 2d pixels.
- For geometric objects, their corresponding categories should remain unchanged under certain geometric transformations, such as rotation, translation, scaling and folding in space. Therefore, we cannot naively judge the category of objects merely through some single perspective.
- Finally, the order of the points of the geometry does not affect the category of the geometry, which means that a 3D object composed of n points, its points can be arranged in at most $n!$ ways, and given any of these $n!$ permutations our classifier should give the same classification result.

Many classification methods for three-dimensional objects have been proposed by researchers in recent years. In this paper, we will conduct a horizontal comparison of the three existing deep network models that can be used for 3D object classification: PointNet, MultiView CNNs and Voxnet. Tests and evaluations the effects of three different deep learning models will also be made on the ModelNet10 data set.

- We will visualize some results of the models output to directly compare their 3D reconstruction ability. Models are supposed to summarize the shape of the object by a sparse set of key points.
- We will use confusion matrix to measure the performance of each model on 3D multi-object classification task. Also we would like to compare their abilities with the overall accuracy and the average accuracy among classes.

2 Literature survey

Because a geometric body can have many different manifestations, the classification method will be very different according to the different forms of "representations". An intuitive method is multi-view input learning. The geometric information contained in a three-dimensional object can be represented by images rendered from multiple perspectives. We can use a more mature two-dimensional image convolution network to extract the feature information from the images, and then aggregate the information from various angles to obtain the geometric features of the three-dimensional object, and finally get the classification result.[Gui17]

In other studies, 3D objects are discretized into uniform-sized voxel grids, and CNN is directly applied to the three-dimensional objects.[MS15c; Su+15] Two-dimensional CNN and three-dimensional CNN are used together to learn the properties of geometry, and the outputs of multiple CNNs are fused before generating prediction results.

Finally, it is the learning network pointNet that uses the point cloud as input geometry. Point clouds are important data structure that represent objects or space. Each point represents the X, Y, and Z geometric coordinates of a single point on an underlying sampled surface and sometimes plus extra feature channels such as color. In pointnet, we sample a fixed number of points on the surface of the mesh to obtain unified data input, and then apply one-dimensional convolution kernels on the standardized three-dimensional point cloud data to obtain the feature information of the point cloud[Gui17]

3 Dataset

ModelNet is a CAD model database collected and created by the Princeton ModelNet project. The data is manually filtered by human workers from the statistics in the SUN database to decide whether each CAD model belongs to the specified cateogries. There are ModelNet10 datasets and ModelNet40 datasets which are both subset of the project containing respectively 10 and 40 categories. ModelNet40 is also the primary dataset used for training pointNet and the multi-view CNN classifier.

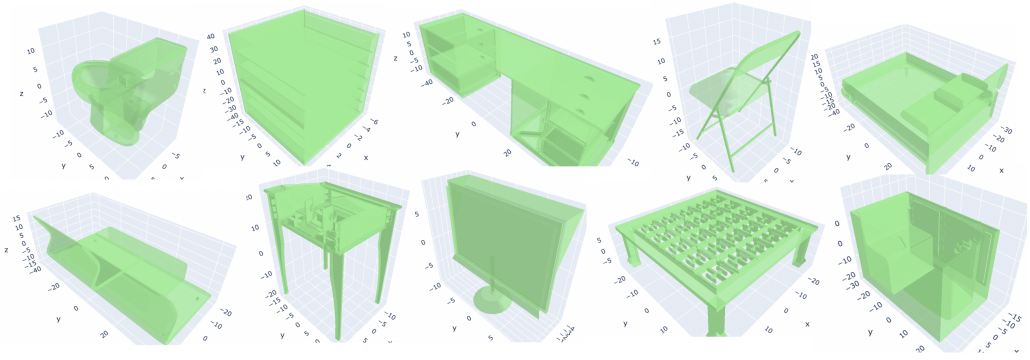

Figure 1: The first images of the 10 classes in ModelNet10

4 Models

In this section, we will focus on introducing the structural characteristics of each deep network and the corresponding data preprocessing methods.

4.1 Multi-View CNNs

Multi-View CNN uses two-dimensional renderings of the three-dimensional object from different "viewpoints" as the original training data. This is equivalent to performing "dimensionality reduction" processing on the training data, cleverly converting the 3D shape classification problem into the two-dimensional image classification problem.

4.1.1 Network Architecture

The network structure of MVCNN is consisted of two major parts. In the first part, each 2D image will be passed through a convolutional neural network for extracting features. This task can usually be done by feature extraction backbone such as ResNet and VGG11. The extracted features will then be aggregated through a view pooling layer, which is essentially a max pooling layer for gathering the most significant information from different views. In the following part, feature vectors will be sent through fully connected layers and finally a softmax classification layer for the final prediction.

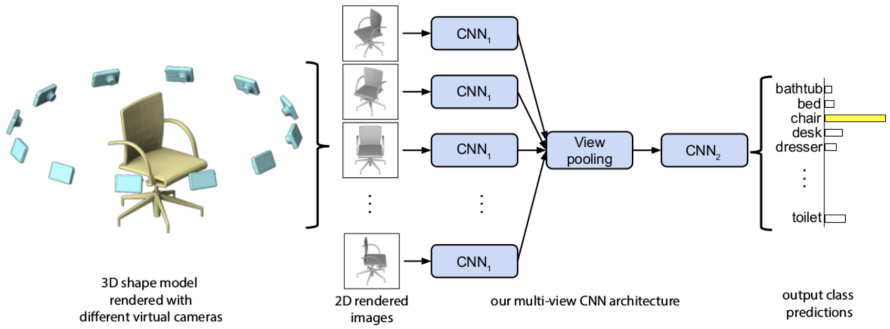

Figure 2: Network structure of Multi-view Convolutional neural network for 3D shape classification

4.1.2 Data preprocessing

Assume that the input 3D shape is placed upright according to a constant axis (Z-axis). In this case, the object is surrounded by 12 "virtual cameras", which is to say, in every 30 degrees, a 2D perspective rendering is generated. And when the camera is working, it has a horizontal angle of 30 degrees with the horizontal plane, and it points directly to the center of the 3D mesh object.

Figure 3: Prerendered images by 12 "virtual cameras"

4.2 VoxNet

VoxNet was first proposed in serveral early works [MS15a; MS15b], that uses convolution and pooling layers defined on 3D voxel grid. However, before further processing, the point cloud will be transformed into an input-voxel grid. It uses Occupancy Grid Map to represent the occupation probability of each voxel in space (probability of Occupied state), and then feed the modified data to the model as input for training. The output will be a sub-vector of the output class.

4.2.1 Network Architecture

VoxNet is similar to the 2D CNN network structure: it consists of 2 convolutional layers, 1 maximum pooling layer and 2 fully connected layers.

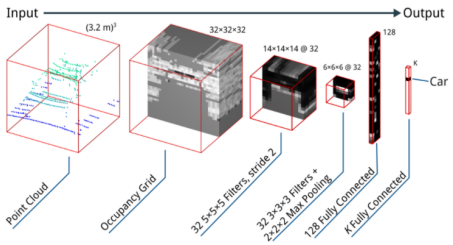

Figure 4: The VoxNet Architecture

4.2.2 Data preprocessing

Given the feature of VoxNet, the input 3D point cloud data should be translated into Occupancy Grid. Occupancy grids [ME85; Thr02] represent the state of the environment as a 3D lattice of random variables (each corresponding to a voxel) and maintain a probabilistic estimate of their occupancy as a function of incoming sensor data and prior knowledge.

4.3 PointNet

The structure of PointNet is very simple, it can directly input point cloud data, and then get the result of classification/segmentation. The model performs independent processing on each point of the disordered point cloud. The key structure in the network is a single symmetric function maximum pooling, which integrates the information of each point in the point cloud. The framework of pointNet consists of two types of micro structures: T-Net, MLP. T-Net is a micro network, which is used to generate an affine transformation matrix to normalize the rotation and translation of the point cloud. This transformation/alignment network is a miniature PointNet. The second structure is n perceptrons that process the point cloud and features. It is capable for increasing the dimension of the point cloud to 64 or 1024.

5 Method

In this section, we will present the implementation of the three models mentioned above for 3D shape multiclass classification. Codes can be found in our GitHub repository(https://github.com/zilixie/3D-Shape-Classifiers).

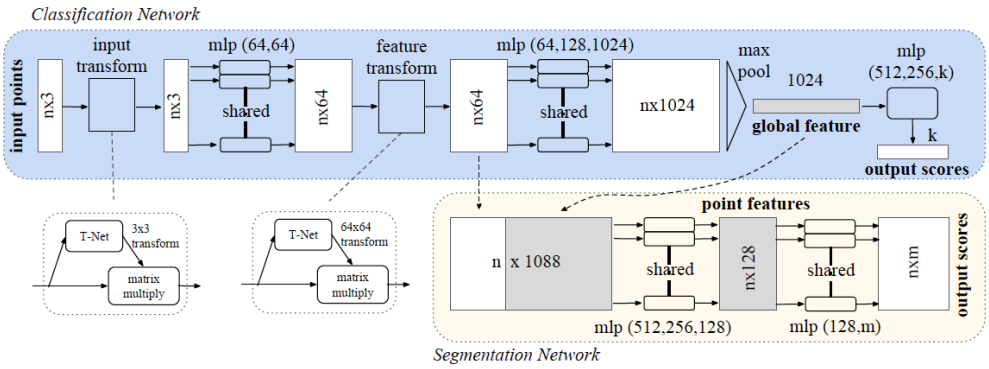

Figure 5: The PointNet Architecture

5.1 Loss function

For this project, we are solving multiclass classification task. Therefore, we use Cross Entropy Loss as our loss function.

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as:

$$-(ylog(p) + (1-y)log(1-p))$$

if $M > 2$, which means for the multiclass classification, we calculated a separate loss for each class label per observation and sum the result.

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$$

In the implementation, the loss can be described as:

$$loss(x, class) = -log(\frac{exp(x[class])}{\sum_j exp(x[j])})$$

$$= -x[class] + log(\sum_j exp(x[j]))$$

5.2 Data visualization

We have create functions to display animated rotation of meshes and point clouds, which will provide us good understanding of the point cloud data. We also create functions to visualize the data after data transformation so that we directly have the view of the difference of input data for these three models.

5.3 Data Transformation

5.3.1 PointNet

- Sampler. We created a sampler function to sample points on the surface uniformly for each point cloud data. We chose to sample 1024 points per cloud as in the paper.

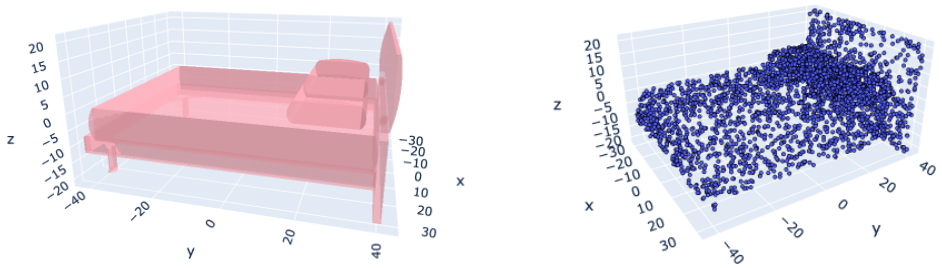

Figure 6: Visualizations of the point cloud data. This is a point cloud data in "bed" class. The left figure is the point cloud data in mesh format. The right figure is the original point cloud data in point cloud format.

- Data augmentation. We perform random rotation of the whole point cloud and add random noise to the points, so as to improve models robustness.
- Normalization. We perform normalization on the points that on the same axis(x, y, z).

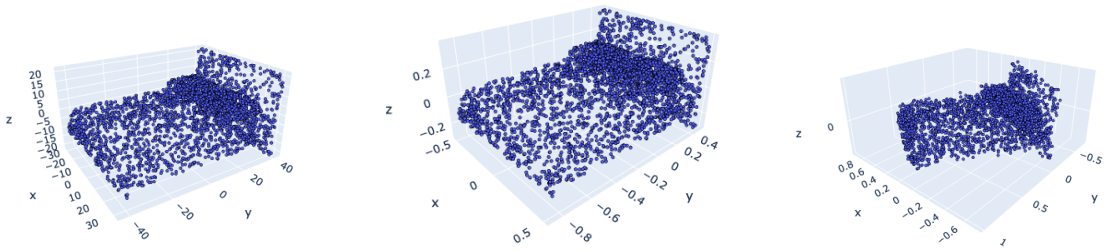

Figure 7: Visualization of the point cloud data after normalization and data augmentation. The figure on the left is the original point cloud data. The figure in the middle is the point cloud data after normalization. As we can see, the range of three axes(x, y, z) have changed. The figure on the right is the point cloud data after data augmentation. As we can see, more noise appears and the point cloud has been rotated

5.3.2 VoxNet

We preform voxelization to all the point cloud data in ModelNet10. The tool we use to voxlize the point cloud is Binvox(https://www.patrickmin.com/binvox/). Binvox is a straight-forward program that reads a 3D model file, rasterizes it into a binary 3D voxel grid, and writes the resulting voxel file. In the voxelization, all the 3d point cloud objects have been scaled to fit a 30 x 30 x 30 grid and all the object would be focus in the center of the grid.

5.3.3 Multi-View CNNs

In order to simplify the experimental process, we used the rendered images of the ModelNet10 point cloud data as the input of the network. This approach does not require the rendering process to be integrated into the training, so it can effectively speed up the training of the network.

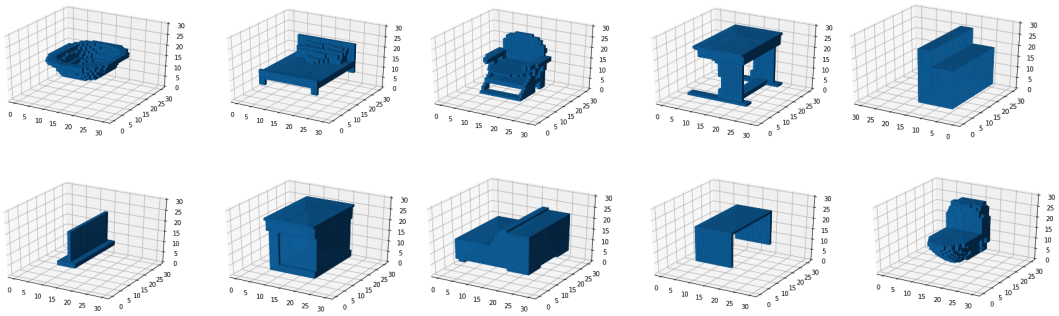

Figure 8: Samples of point cloud data after voxelization. The first row: bathtub, bed, chair, desk, dresser. The second row: monitor, night stand, sofa, table, toilet

5.4 Training

We use gradient descent algorithm in Mini-batches Learning way to train our models. We use Adam optimizer for all three models. For PointNet, learning rate is set as 0.001. For VoxNet, learning rate is set as 0.001. For MVCNN, learning rate is set as 5e-5. For PointNet, the batch size is 32 for training process and 64 for vaildation. For VoxNet, the batch size is 32. For MVCNN, the batch size is 8. PointNet is trained for 15 epochs. VoxNet is trained for 15 epochs. MVCNN is trained for 30 epochs.

6 Results

6.1 Accuracy

In Table 1, we compared the the prediction accuracy of PointNet, VoxNet and MVCNN. MVCNN achieves the highest average accuracy on both per instance and per class. Its performance is 3.5% higher than VoxNet and 6.3% higher than PointNet on average accuracy per instance, and 4.7% higher than VoxNet and 7.0% higher than PoinNet on average accuracy per class. Then followed by VoxNet. PointNet has the lowest accuracy on both per instance and per class. VoxNet and PointNet have similar performance. VoxNet is 2.8% higher than PointNet on average accuracy per instance and 2.3% higher on average accuracy per class.
To some extent, we can conclude that the multi-view imaged-based model has far higher accuracy than the point-based model and voxel-based model. Point-based model and voxel-based model have similar performance, while voxel-based model is slightly better than Point-based model.

Models	average accuracy per instance	average accuracy per class
PointNet	84.8%	84.3%
VoxNet	87.6%	86.6%
MVCNN	91.1%	91.3%

Table 1: Accuracy of PointNet, VoxNet and MVCNN

6.2 Performance

In Table 2, we compared the the number of parameters, processing time and memory of PointNet, VoxNet and MVCNN. As we can see, MVCNN has the most parameters, taking the most memory and it is the slowest one. However, it is also the best model that achieves the highest accuracy. This is a tradeoff, sacrificing the memory and run time to get higher accuracy, even though the number of

parameters MVCNN has is 3582% more than the parameters of PointNet and 9207% more than the parameters of VoxNet. As for PointNet and VoxNet, the parameters, run time and memory are more than 2 times than VoxNet. However, the accuracy of PointNet is lower than VoxNet. From this aspect, VoxNet might be a better choice than PointNet.
To some extent, we can conclude that the multi-view imaged-based model is much more heavier than the point-based model and voxel-based model. Voxel-based model is highly cost-effective. Point-based model is also lightweight but performance is not as good as voxel-based model.

Models	Parameters	Forward-pass time	Memory
PointNet	3.5M	3.1ms	4.4GB
VoxNet	1.4M	1.3ms	2.0GB
MVCNN	128.9M	25.8ms	10.0GB

Table 2: Parameters, Forward-pass time and Memory of PointNet, VoxNet and MVCNN

6.3 Confusion Matrix

In Figure 9, we compared the the confusion matrix of PointNet, VoxNet and MVCNN. As we can see, all the three models have frequently regard "table" as "desk". And all of them are excellent preformance on correctly recognizing "bed", "chair", "monitor", "sofa" and "toilet". The reason might be that the definition of "desk" and "table" is blurred. Objects could be called desk can also be called as table in our real life. And in the ModelNet10, the some objects tagged as "desk" and "table" look similar. For those classes that models performed well on, have their distinguished features so that they would be easily and correctly classified by the models.

Apart from desk, the three models also didn't perform robustly on the class "night stand" and "dresser". They usually mistakenly regarded "dresser" as "night stand" and mistakenly regarded "night stand" as "dresser" and "monitor". And the reason is that the objects of these three classes are somehow similar, but each of this class has some different features, like monitor only has one leg, night stand looks more like a solid rectangle and dresser has hollow space under the desktop. Among this three models. Among these three models, MVCNN performs better and then the VoxNet. PointNet has the least ability to distinguish these three classes. Thus, we can conclude that MVCNN is better on spotting the global and local features than PointNet and VoxNet. PointNet is slightly inferior to VoxNet on this ability.

6.4 Loss and Accuracy curves

In Figure 10, we compared the the loss and accuracy curves of PointNet, VoxNet and MVCNN. For PointNet, the loss is steadily decreasing and the accuracy is steadily increasing, which shows the good stability of the model learning process. For VoxNet, the accuracy on training data rises rapidly and the loss drop rapidly at the very beginning and reach convergence stage soon. However, the accuracy and loss on validation dataset keep fluctuating in a small range. It shows that VoxNet might slightly overfitting on validation data. For MVCNN, both the accuracy curve and loss curve are very flat. One reason of this is that MVCNN can use pretrained VGG for the feature extraction from the rendered images and the weights do not need to much adjustment. Another reason is thatAnother reason is that in traditional image recognition tasks, only need to be given a picture to meet the requirements of image recognition, and in mvcnn we have multiple pictures as input

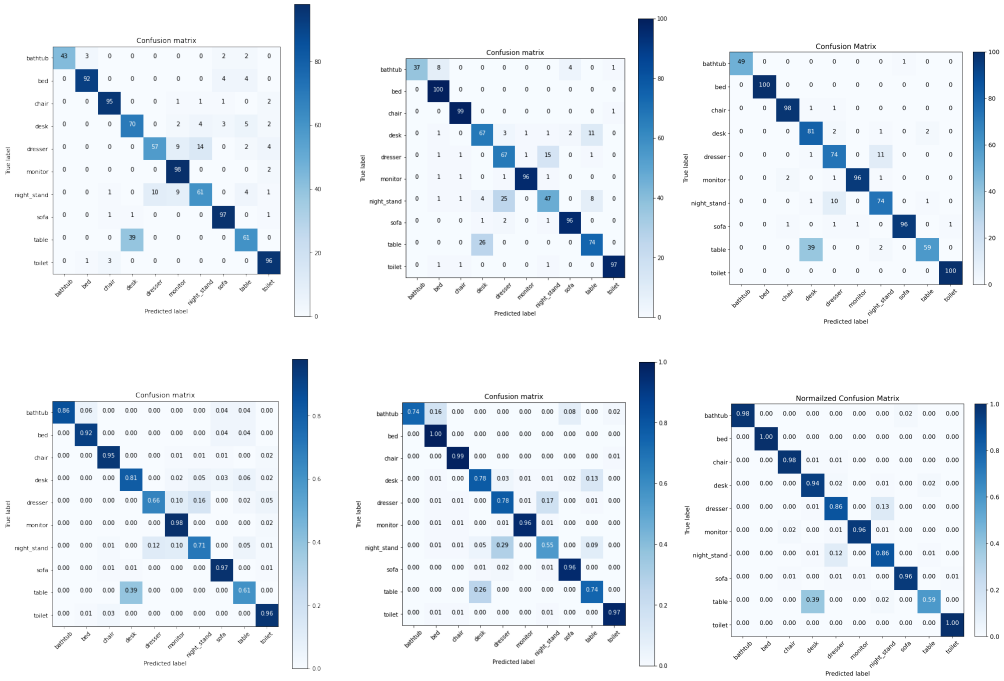

Figure 9: Confusion Matrix of PointNet, VoxNet and MVCNN. The first row is the confusion matrixs without normalization. The second row is the confusion matrixs with normalization. The first, second and third column are PointNet, VoxNet and MVCNN

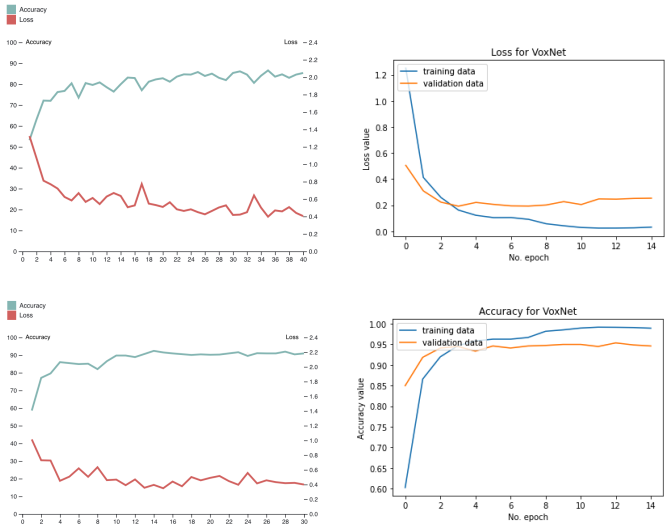

Figure 10: Loss and Accuracy curves of PointNet, VoxNet and MVCNN. The graph at top left is the curves of PointNet. The graph at bottom left is the curves of MVCNN. The graph at top right is the loss curves of VoxNet. The graph at the bottom right is the Accuracy curves of VoxNet.

7 Conclusion

We investigated on three models with different 3D input representations for 3D object multiclass classification task. We chose PointNet as the representation of point-based model, VoxNet as the representation of voxel-based model, and MVCNN as the representation of multi-view imaged-based model. Before the models implementation, we perform data preprocessing, data transformation and

data augmentation on the ModelNet10 dataset so as to meet the different input format requirement of these three models. Besides, we create visulization functinos to visulize the input data so as to more directly interpret the 3D input data of each model. We successfully implemented these models and achieved the calssificaiton task, and also get their performance information. With the comparison on their accuracy, performance, confusion matrix and loss/accuracy curves, we reached some conclusions:

- MVCNN has far higher accuracy than the PointNet and VoxNet. PointNet and VoxNet have similar performance, while VoxNet is slightly better than PointNet.
- Multi-view imaged-based model is much more heavier than the point-based model and voxel-based model. Voxel-based model is highly cost-effective. Point-based model is also lightweight but performance is not as good as voxel-based model.
- MVCNN is better on spotting the global and local features than PointNet and VoxNet. PointNet is slightly inferior to VoxNet on this ability.
- MVCNN has great learning ability. It converges after few epochs. VoxNet can also converge in comparatively fast speed but not as fast as MVCNN. PointNet takes more time to get convergence.

From the network architecture of these three models, we can find the reasons of conclusions. For Multi-view imaged-based model, it needs to have Deep CNN architecture to supports its feature extraction and layers to aggregated features from different views, it has more parameters. As a result that it observes the objects from multiple views, it would have better understanding on features from different angles. And since the input data would be several rendered images for each object, it would be burdensome for the model to analyze 3D object. MVCNN use pretained VGG in its architecture, it takes less time to converge. For voxel-based model, VoxNet here has light architecture with less layers, thus it runs faster. For point-based model, PointNet has more complicated architecture than VoxNet. Also, there is an advantage of point-based networks, which is that they are more robust to point perturbations, while multiview and voxel-based networks can be fooled by imperceptible perturbations.

References

[Gui17] Charles R. Qi* Hao Su* Kaichun Mo Leonidas J. Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation.* In IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017.

[ME85] H. Moravec and A. Elfes. "High resolution maps from wide angle sonar". In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation.* Volume 2. 1985, pages 116–121. `https://doi.org/10.1109/ROBOT.1985.1087316`.

[MS15a] D. Maturana and S. Scherer. "3D Convolutional Neural Networks for Landing Zone Detection from LiDAR". In: *ICRA*. 2015.

[MS15b] D. Maturana and S. Scherer. "VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition". In: *IROS*. 2015.

[MS15c] D. Maturana and S. Scherer. *Voxnet: A 3d convolutional neural network for real-time object recognition.* In IEEE/RSJ International Conference on Intelligent Robots and Systems. 2015.

[Su+15] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. *Multi-view Convolutional Neural Networks for 3D Shape Recognition.* 2015. arXiv: `1505.00880 [cs.CV]`.

[Thr02] Sebastian Thrun. "Learning Occupancy Grids With Forward Sensor Models". In: (Dec. 2002).