# Object Classification with PointNet

Jun Young Yun[1]

Electrical Engineering and Computer Science (EECS)
Daegu Gyeongbuk Institute of Science and Technology (DGIST)
Daegu, Republic of Korea
jyyun1209@dgist.ac.kr

**Abstract.** In this project, we implemented PointNet with PyTorch, and examined the role and effect of each layer that composes PointNet.

**Keywords:** PyTorch · PointNet.

## 1 Introduction

As deep learning are widely used in various fields, We can easily find many deep learning networks online. Thus, we can use them in our work without fully understanding the structure or principles of networks. However, in order to handle the network more specifically to one's research, it is necessary to understand the structure, principles, and their influences. In this project, we implement the PointNet, which is an pioneer paper on object classification using point cloud data with thoroughly explore the inside of PointNet by conducting various experiments.

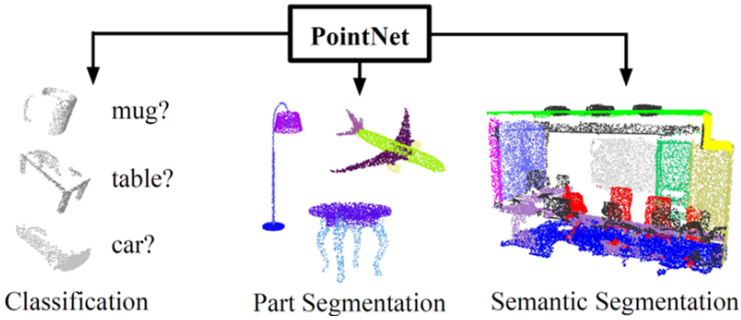


**Fig. 1.** Applications of PointNet

## 2 Related Works

### 2.1 Object Classification

Object classification is a task to classify which class the data belongs. The data can be a camera image or LiDAR point cloud [2].

### 2.2 PointNet

PointNet is a network that performs object classification, part-segmentation, and semantic-segmentation through point cloud data (Fig. 1). The overall structure of PointNet can be seen in Fig. 2. Classification network consists of two consecutive pair of transformation and shared MLP. Transforms (input transform and feature transform) have a same formation which multiplies input data with transformation matrix generated in T-net. Please check the original PointNet paer for more information [3].

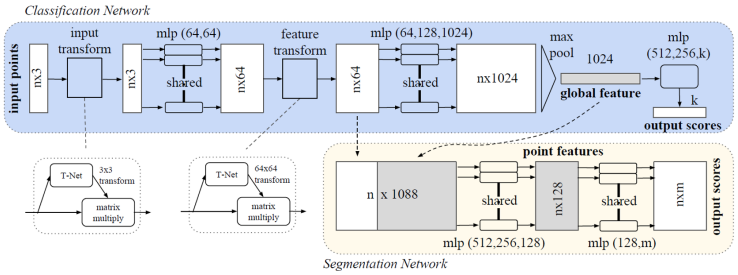


**Fig. 2.** Architecture of PointNet

### 2.3 ModelNet-10 Dataset

The ModelNet-10 dataset is a dataset in which 10 types of indoor furniture are modeled in CAD (Fig. 3). We can create a point cloud from the CAD model and use it as an input for a deep learning network.

## 3 Key Contribution

We implemented PointNet with PyTorch which was implemented with Tensorflow by authors. By sharing this project on GitHub [4], we helped developers who are more familiar with PyTorch than Tensorflow. In addition, the effect of each layer used in PointNet was experimented and proved the necessity of each layer. Furthermore, we provide an intuitive understanding to the readers by visualizing the features created in the layers.

## 4 Methods

The PointNet authors explained PointNet structure well in their paper. Thus, implementing PyTorch version of original PointNet was proceeded without any additional materials. After implementing the original PointNet, five versions were

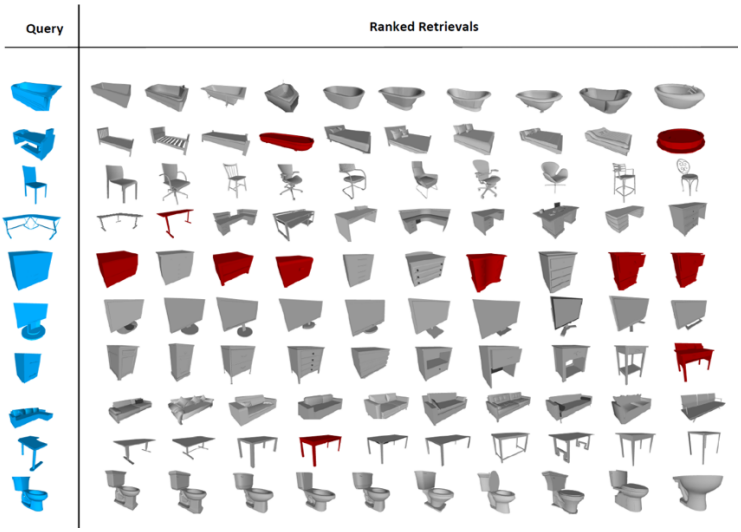


**Fig. 3.** ModelNet-10 dataset

additionally implemented: 1. a version without all transformations (input transform, feature transform), 2. a version with only using a input transform, 3. a version with only using a feature transform and cross entropy loss, 4. a version with only using a feature transform and loss function newly defined by the authors, 5. a version using cross entropy loss. In Table. 1, they are named PointNet w.o. any, PointNet input, PointNet feat. CE, PointNet feat. and PointNet CE repectively. These are trained with same environment and epochs (25 epochs), and compared quantitatively by accuracy.

Furthermore, we plotted some meaningful point clouds for visual understandings: Original point cloud, point cloud right after the input transform, and hot-points. Original point cloud is an input point cloud which comes into the PointNet, and point cloud right after the input transform is a point cloud after multiplying transform matrix generated from input transform. Lastly, hot-point means the point set where each of the 1024 global features (in the classification network in Fig. 2) extracted by the network was obtained from. We could trace back to the points because order of points in all features extracted by shared MLPs are maintained before the maxpool calculation. Intuitively, hot-points are the points that paying the most attention when the network proceeds classification. These three point clouds are in the Fig. 4.

## 5 Results

As you can see in the Table 1, a model with using two transformations and newly defined loss function shows the highest testing accuracy. This result is inevitable,

**Table 1.** Accuracy comparison for various models.

| **Model** | **Training Acc. (%)** | **Validation Acc. (%)** | **Testing Acc. (%)** |
|---|---|---|---|
| PointNet w.o. trans. | 94.73 | 94.56 | 89.32 |
| PointNet input | **96.79** | **96.30** | 85.46 |
| PointNet feat. CE | 96.21 | 95.79 | 91.74 |
| PointNet feat. | 95.42 | 95.24 | 91.63 |
| PointNet CE | 95.98 | 95.79 | 92.07 |
| PointNet | 95.37 | 94.68 | **92.95** |

but the weird thing is that the PointNet without any transform shows better performance than the PointNet with using only a input transform. This is why we did additional experiments with the PointNet with using only a feature transform. Surprisingly, using both input transform and feature transform together shows the highest accuracy while models with a single transform versions show relatively low accuracy. In Fig. 4, you can find some visualizations for above models. The point cloud after input transformation shows that there are some distortions compared to the original point cloud and hot-point seems to reflect the edge of the object which is red colored points in figures on the right side.

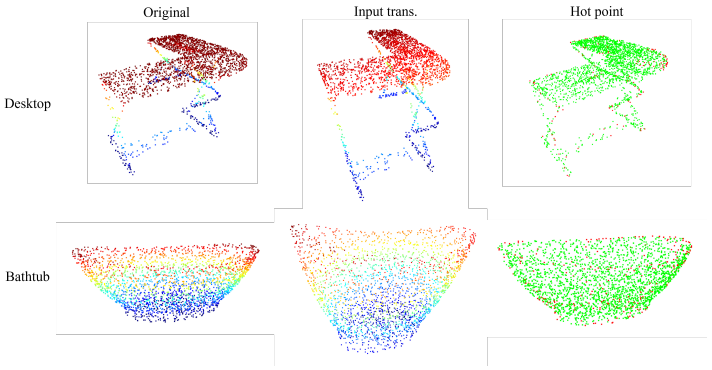


**Fig. 4.** Visualization results (Original: Original point cloud, Input trans.: Point cloud after input transformation, Hot point: Point cloud with the hot-point)

## References

1. Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., ... & Yu, F. (2015). Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012.
2. Shen, X. (2019). A survey of object classification and detection based on 2d/3d data. arXiv preprint arXiv:1905.12683.
3. Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 652-660).
4. Yun, J. (2022). JYYUN1209/pointnet: Pytorch implementation of PointNet. GitHub. Retrieved December 7, 2022, from https://github.com/jyyun1209/PointNet.