**Assignment 1 Report for Basic Part**

**HUANG Chuanshan**

**1155076957**

1. **Introduction**

In the basic part of this assignment, we are required to implement a supervised neural network to retrieve similar pointclouds from a database given query pointclouts. The whole implementation is based on PointNet network proposed by Qi et al [1].

1. **Overview of the solution**

The diagram below gives an illustration of the structure of our solution to this problem:

TRAIN THE MODEL FOR SOME EPOCHS UNTIL LOSS OF CLASSIFICATION STOP DECLINING

IMPLEMENT THE NEURAL NETWORK ACCORDING TO THE ASSIGNMENT SPECIFICATION

COMPUTE THE GLOBAL FEATURES OF EACH POINTCLOUD IN THE QUERY FILE AND THE DATABASE

MODIFY THE MODEL SO THAT IT OUTPUTS THE GLOBAL FEATURES OF AN POINTCLOUD

FIND THE MINIMUM DISTANCE BETWEEN QUERY OBJECTS AND DATABASE POINTCLOUDS THEN RETRIEVE IT

1. **Details of implementation**

There are mainly four Python files we used to achieve the solution. Below we will give concrete explanation of what these files are doing:

* 1. *asg1\_pointcloud\_model.py:*

According to the model design in the specification, we insert three CNNs to extract 128 features from each pointcloud before we call the Tensorflow max pooling function. The above process yields the global features of a pointcloud. After that, we use three consecutive fully connected layer in order to flatten the features into a tensor of 40 elements, each of them representing a score for an object class.

In the source code of the model, based on the original *PointNet* structure, we detect the the location where the global features of the input pointcloud is generated, then we set it as the output of the model such that we can obtain the global feature when we run it in a session. Also, we tune the channels of CNNs and fully connected layers so that the model satisfies the requirements in the specification.

* 1. *asg1\_train.py*:

This file is, as its name indicates, for training the model. It contains the codes for setting the hyperparameters. **In our design, we set all the hyperparameters to be exactly the same as those of the assignment specification.** Basically it’s the same as *train.py* in the *PointNet* package, despite some codes we added for printing out variables, such as loss, for debugging.

* 1. *asg1\_evaluate.py*:

Same as asg1\_train.py, it’s generally a replica of evaluate.py in the *PointNet* package except that we asked our version to print out certain tensors.

* 1. *asg1\_retrieve.py*:

This file is responsible for the whole object retrieval procedure. Before actually doing anything, we setup the whole environment by specifying command-line inputs and hyperparameters. After that we feed input to a running session in order to obtain the global features of query and database pointclouds. Noted that we compute the global features of all pointclouds, which means that we are holding a 40x128 tensor in hand after we feed query file into the session, and a 800x128 one after feeding the database file.

After we obtain the global features, we compute the Euclidean distance between each query pointcloud feature and database pointcloud feature. This operation will give a 40x800 list, with each element being the distance from the query pointcloud feature to database pointcloud feature.

The last thing this file is doing is to sort each row of the above-mentioned 40x800 list, which is all the distances from a single query pointcloud to all pointclouds in the database. After this we obtain the desired pointclouds of highest similarity from the database.

1. **Results**
   1. Accuracies:

The overall accuracy for object classification is around 0.81, as shown below:

eval mean loss: 0.616125

eval accuracy: 0.808750

eval avg class acc: 0.808750

The accuracy of each class is given below:

airplane: 1.000 bathtub: 0.750 bed: 1.000 bench: 0.700

bookshelf: 0.850 bottle: 0.950 bowl: 0.900 car: 1.00

chair: 0.950 cone: 0.950 cup: 0.600 curtain: 0.750

desk: 0.800 door: 0.900 dresser: 0.500

flower\_pot: 0.050 glass\_box: 0.800 guitar: 1.000

keyboard: 1.000 lamp: 0.850 laptop: 1.000 mantel: 1.000

monitor: 0.900 night\_stand: 0.900 person: 0.850

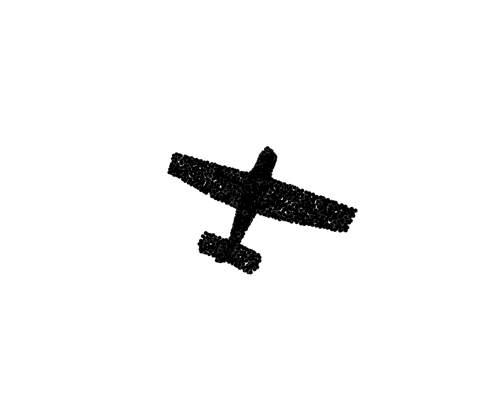
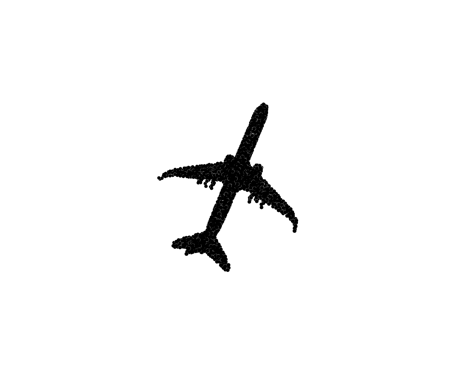
piano: 0.950 plant: 0.650 radio: 0.550

range\_hood: 0.800 sink: 0.700 sofa: 0.950 stairs: 0.800

stool: 0.700 table: 0.850 tent: 0.950 toilet: 1.000

tv\_stand: 0.750 vase: 0.800 wardrobe: 0.350 xbox: 0.600

* 1. Object Retrieval:

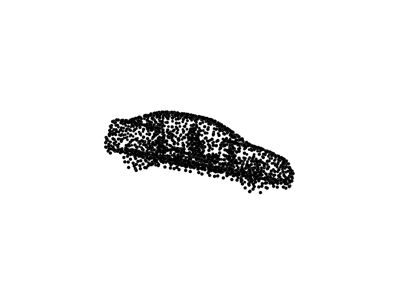
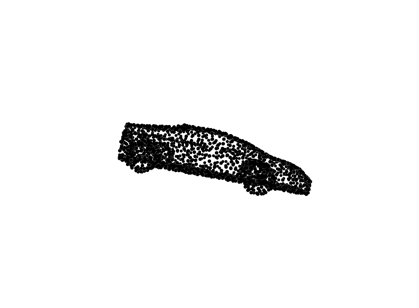
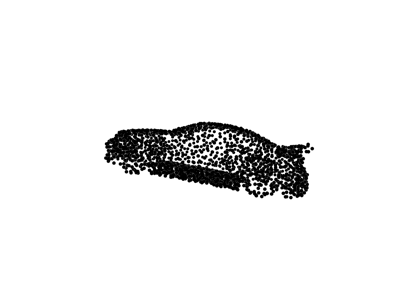
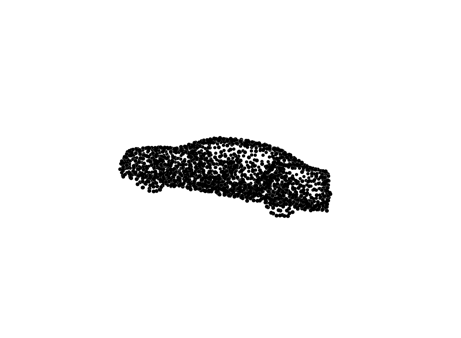
Below shows some results of object retrieval.

**Label 0: Airplane**



**Label 7: Car**

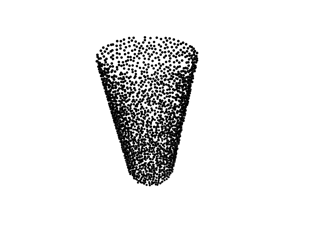
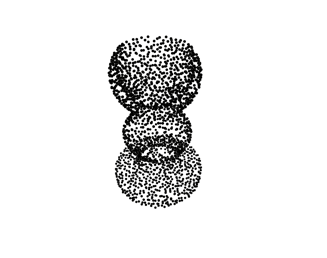
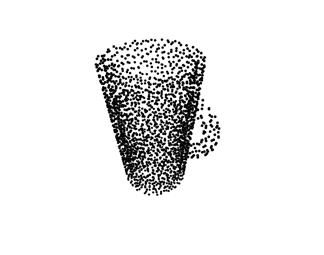




1. **Discussion on the results**

The above simple neural network achieves an acceptable accuracy of around 0.81. However, when we look into individual class accuracy, we find out that class “flower\_pot” (highlighted above as red) is only accurately classified with a probability of 0.05. The retrieval result of class “flower\_pot” is shown below:

**Label 15: flower\_pot**



**Label 15: flower\_pot Label 10: cup Label 37: vase Label 37: vase Label 10: cup**

We can see that the network made mistakes in cases where the query pointcloud and the database pointcloud look similar. A possible reason for this is that we only obtain the global features from the pointcloud – but not the local features. Thus, some small details of the object might be ignored, for example, the handle of a cup. To tackle this, we simply need to extract local features from pointclouds and then apply them into object retrieval.

**References:**

[1] Qi C R, Su H, Mo K, et al. PointNet: Deep learning on point sets for 3D classification and segmentation. CVPR, 2017.