Project-3 of "Neural Network and Deep Learning"

June 8, 2020

Abstract

(1) This is the third project of our course. The deadline is 5:00pm, June 25, 2020. Please upload the report via elearning.

(2) The goal of your write-up is to document the experiments you've done and your main findings. So be sure to explain the results. The report can be written by Word or Latex. Generate a single pdf file of your mini-projects and turned in along with your code. package your code and a copy of the write-up pdf document into a zip or tar.gz file and named as Project3-*your-student-id*_your_name.[zip|tar.gz]. Only include functions and scripts that you modified. Also put the names and Student ID in your paper.

(3) About the deadline and penalty. In general, you should submit the paper according to the deadline of each mini-project. The late submission is also acceptable; however, you will be penalized 10% of scores for the delay. We need to submit the final score by June 27th; so donot significantly delay this project.

(4) We will allocate GPUs.

1 3D Object Classification

The remarkable advances in the Deep Learning (DL) architectures on 2D data has been coupled with notable successes in the computer vision field by achieving impressive results in many tasks such as: classification, segmentation, detection and localization, recognition and scene understanding. The key strength of deep learning architectures are in their ability to progressively learn discriminative hierarchical features of the input data. Most of the DL architectures are already established on 2D data. DL architectures on 2D data showed the requirement for large amount of training data. Due to this fact, applying DL on the 3D domain was not as effective as 2D. Fortunately, with the latest advances in 3D sensing technologies and the increased availability of affordable 3D data acquisition devices such as structured-light 3D scanners and time-of-flight cameras, the amount of the available 3D data has tremendously increased. 3D data provides rich information about the full geometry of 3D objects. Driven by the breakthroughs achieved by DL and the availability of 3D data, the 3D

computer vision community has been actively investigating the extension of DL architectures to 3D data.

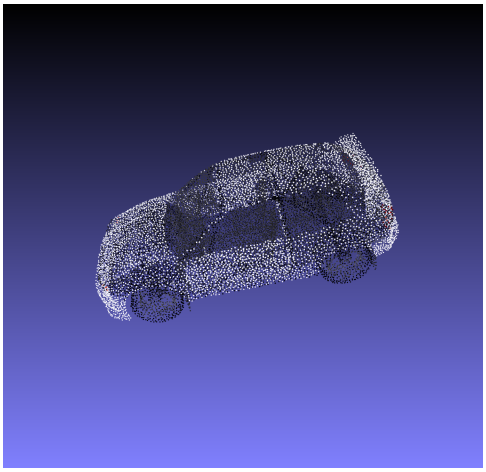

Figure 1: 3D point cloud

In this project, you will try to use neural network to classify 3D point clouds, and initially understand the use of 3D data and the application of deep learning in the 3D field.

1.1 Dataset

A point cloud is a set of data points in space. Point clouds are generally produced by 3D scanners, which measure many points on the external surfaces of objects around them. A point cloud can be seen as a set of unstructured 3D points that approximate the geometry of 3D objects. Such realization makes it a non-Euclidean geometric data representation. However, point clouds can also be realized as a set of small Euclidean subsets that have a global parametrization and a common system of coordinates and invariant to transformations such as translation and rotation.

Our experimental data comes from ModelNet40. It contains CAD models from the 40 categories used to train the deep network. We make available to HDF5 files provided by Charles R. Qi et al with all models of different categories pre-organized, so you may need to install h5py (pip install h5py). Each point cloud contains 2048 points uniformly sampled from a shape surface. Each cloud is zero-mean and normalized into an unit sphere.

Click here to download ModelNet40 (416MB).

Note: ply_data_train0~5.h5 for training and ply_data_test0~1.h5 for testing

1.2 Problem Statement

You need to design a deep learning framework that directly consumes unordered point sets as inputs. A point cloud is represented as a set of 3D points $\{P_i \mid i = 1, ..., n\}$, where each point p_i is a vector of its (x, y, z) coordinate. For the object classification task, the input point cloud is either directly sampled from a shape or pre-segmented from a scene point cloud. The deep network outputs k scores for all the k candidate classes.

2 Experiments

To investigate deep learning in 3D data, we will use following experimental setup: 3D object classification on ModelNet40 dataset with a basic convolutional neural network. And all sample codes have been tested with *Python 3.6, Pytorch 1.1, CUDA 10 on Ubuntu 16.04*.

We break this task down into two parts. In the first part, we will extract a model in the dataset and save it as an obj file. The second part is implementing a point cloud classifier, which takes the vertices of models in a batch as input and produces as output the class (e.g. airplane, rocket, bag, etc).

To make your life easier, we have provided a Pytorch ModelNetDataset loader for the above dataset. You may need to reconfigure the data path according to where you store the data.

2.1 Save 3D Point Cloud

To understand a deep learning task, you first need to be familiar with the form of the data and its annotations. The data used in this experiment is stored in the form of HDF5 files, in order to have a deeper understanding of the 3D point cloud data, in this part, we try to take a 3D model from the dataset and save it. If you are not familiar with the *obj* format file, this is important 3D Graphics knowledge, so please see the Wikipedia article on .obj file.

In this task, we require that you can store a point cloud correctly. Do it as following:

1. Create a dataset loader using the *pytorch dataset* provided in the sample code;
2. Read the data with the dataset loader and take out one of the models (2048 $\times$3);
3. Save the model as the format of *obj* file using numpy (*Note: You need to first understand the storage structure of the obj file from the above materials*);
4. (Not required) This experiment only requires you to understand the format of the *obj* file, if you want to view what point cloud looks like, then you need to install some 3D tools such as MeshLab to import obj file.

Please write a file ***save_pointcloud.py*** or a function ***save_pointcloud***() to complete the process described above (1, 2, 3). The final submitted files need to contain the point cloud data (*obj*) you obtained from the dataset.

2.2 Training and Testing

In this part you need to write three files: *model.py, train.py* and *test.py*, details as follows.

We suggest to start from the example model code in *model.py* and design your own 3D classification network. We want a network with both accuracy and efficiency. For this project, feel free to collaborate on solving the problem but **write your code individually**. In particular, **do not copy** code from other students or from online resources.

Now we believe that you have been able to write a file for training independently. So please write *train.py* to run your model.

There are some basic requirements:

1. Clearly define model, optimizer and loss function (you will want to use the **categorical_crossentropy** loss);

2. Set hyperparameters such as *total train epoch, learning rate, batch size*, etc;

3. Display the loss value of each step in order to monitor the training situation, of course, you can use some web-based visualization tools, you just need to add comments in the codes.

After completing the training, you need to write *test.py* to test the performance of the model on the test dataset. We use accuracy as our metric, so you may need a function *get_accuracy()* in *test.py*

Please report the hyperparameters you selected. If you design the network structure yourself, please report the successful network structure you have referred to or try to explain the reasons for your design.

2.3 Questions

1. In what form is the *obj* file storing vertex color data? If in addition to the vertex coordinates (x, y, z), you also want to store the color data of each vertex in the *obj* file, what should you do? (If you can achieve it, please submit the final *obj* file or 3D model screenshot)

2. What are the differences between neural networks for point clouds and for ordinary images?

3. (Bonus) Suppose you have an image of a car and its corresponding 3D model in the world coordinate system, how do you transform the coordinates in the three-dimensional space to correspond to the pixels of the image, i.e. pixel coordinate system (*refer to 01-camera-models.pdf*)?

4. (Open question) If you want to design a framework to learn to color a 3D model utilizing the information of single image, what do you think?