

INF574 Project : Data-driven Point cloud Segmentation

Markus Chardonnet

September 25, 2024

Introduction

The goal of this project is to implement a method of point cloud segmentation. Having a 3D shape made of a point cloud that represents an object or a scene, segmentation is about assigning a label to each point of the cloud, according to the component it is part of. The method we use in this project is PointNet++, which is derived from PointNet, a Deep Learning architecture for analysing point clouds. At first, the general structure of PointNet++ will be presented. Then, the implementation will be explained as well as the training set used to train and test the method. The results will be presented and analysed in the last part.

Contents

1	Problem and Method Description	2
1.1	Point cloud Segmentation	2
1.2	PointNet	2
1.3	PointNet++	3
2	Implementation	3
2.1	Dataset : ShapeNetPart	3
2.2	Architecture	3
2.3	Parts description	4
3	Results	6
3.1	Farthest point sampling	6
3.2	Point grouping	7
3.3	Segmentation	8

1 Problem and Method Description

1.1 Point cloud Segmentation

As said previously, point cloud segmentation is about classifying each point of a point set into homogeneous regions. In this problem, we are interested in point clouds representing the shape of a 3D object, which can be divided into several parts. These parts usually have a semantic interpretation according to the object. For instance if we have a point cloud representing a table, components of it can be legs, feets, stretchers and the top. Thus, we want to construct a model who maps points of the cloud to component classes. In our case, the point cloud is presented as an unordered list of point coordinates. In order to gain information from points coordinates, we assume that they are embedded in the 3D euclidian space and that we can learn from distances induced by this metric space.

1.2 PoinNet

PointNet is a recent and very successful method to process point clouds. It is mainly used for classification and segmentation but can also be used to estimate normals and curvature. It has the advantage of directly processing points whereas traditional methods tended to transform point clouds to grids and process them using CNN as it is done for image analysis. Part of its success is the ability to process accurately point clouds with non-uniform densities. It has demonstrated high efficiency and accuracy for different tasks.

The basic idea behind PointNet is that given a point set $X = \{x_1, \dots, x_n\}$ in a metric space, we want to learn a set function $f(x_1, \dots, x_n)$, that produces semantic information on X . f has to respect some properties listed below :

- it has to be symmetric which means that it should extract the same information if points from X are permuted (since it will modelize the same shape)
- it has to be invariant under isometries. f should show similar mapping if the point set is rotated for instance
- it has to capture information about the interaction between points, according to the metric space in which the point set is embedded.

In practice, PointNet architecture learns $h : R^d \rightarrow R^K$ and $\gamma : R^K \rightarrow R^C$ such that $f(x_1, \dots, x_n)$ can be approximated by

$$\gamma \left(\max_{i=1..n} (h(x_i)) \right) \quad (1)$$

where h and γ are encoded as MLP's.

In the task of segmentation, f produces a global feature vector for the set, which has to be fed back to points and combined with per point feature to produce a labelling for each point. [1]

1.3 PointNet++

PointNet++ architecture was introduced because the original architecture was hardly able to capture local information (as we saw, it only combines global and individual features). The idea behind PointNet++ is organize PointNet in a hierarchical way in order to extract local feature vectors.

The hierarchical structure is organized into Set abstraction layers and Feature propagation layers.

A Set abstraction layer (SA) produces a feature vector encoding semantic information at a certain scale. It is composed of a Sampling layer, a Grouping layer and a Pointnet layer. The Sampling layer selects points from the input points which neighborhood will define the local regions from which information is extracted. It is coded with a Farthest point sampling algorithm. The Grouping layer groups input points into local regions around query points given by the Sampling layer. Groups are made of points within a ball of a certain radius around query points. The Pointnet layer encodes the function f , mapping features from points of a group to a single feature vector. The SA layer at level l takes as input from points and their features from level $l-1$ (features learned from a local scale) and outputs query points with more global features.

A Feature propagation (FP) layer back propagates features from level $l+1$ to level l . Basically, for each point at level l , it combines and process (with a MLP) features from individual points (comes from SA at level l) and features from near sample points (sampled in SA at level $l+1$) produced by FP at level $l+1$. [2]

2 Implementation

2.1 Dataset : ShapeNetPart

This dataset contains point sets under HDF5 format. These were extracted from original shapes using farthest point sampling algorithm to produce uniform point clouds of 2048 points. The points coordinates are centered and scaled and pre-aligned. There are more than 16000 shapes of 16 categories labelled with 50 segmentation parts (each category having two to five different parts). This dataset is equipped with two python files : dataset and visualize. The first one is for reading the hdf5 files, and extract points and segmentation parts as tensors, and category/part names. The second file, given a shape, produces an XML file, which can be transformed into an EXR image using Mitsuba.

2.2 Architecture

My implementation uses Python as a programming language and the Pytorch library for building the architecture.

Architecture : I did implement two configurations of PointNet++ detailed in the research paper. The first one is designed to learn object part segmentation as well as semantic scene segmentation. Since semantic scene segmentation is usually computationally more expensive than object part segmentation, this configuration is very heavy and takes much time to forward and back propagate. I did implement a second one, only designed for object part segmentation, to be able to learn a model reasonably.

The configurations are described bellow. $SA(N, r, G, [l_1, l_2, l_3])$ refers to Set abstraction layer where : N is the number of points to sample, r is the grouping radius, G is the maximal size of a group and l_i is the layer size at level i of the PointNet layer. $SA([l_1, l_2, l_3])$ refers a global Set abstraction layer (as in original PointNet) where all the point from the lower level are grouped together. $FP([l_1, ..., l_m])$ refers to Feature propagation layers where l_i is the layer size from the MLP at level i.

First configuration :

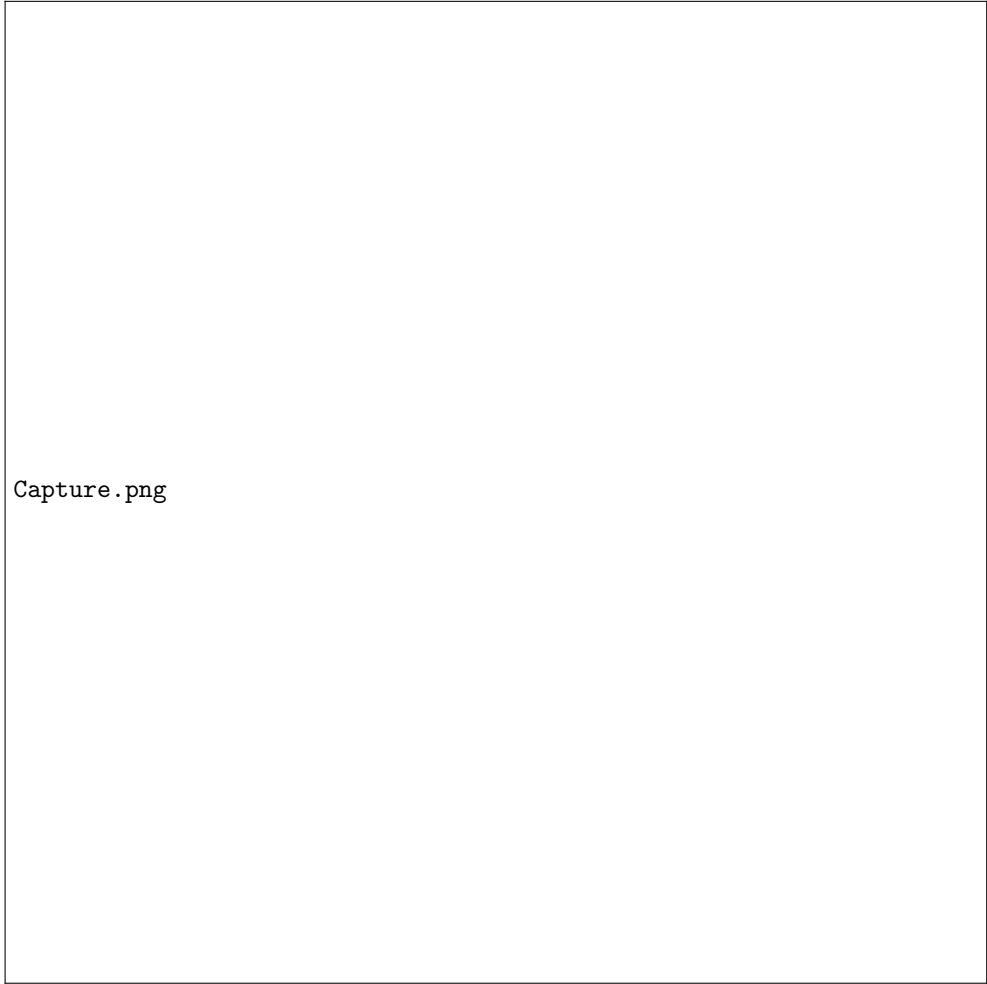
$$\begin{aligned} &SA(1024, 0.1, [32, 32, 64], 32) \rightarrow SA(256, 0.2, 32, [64, 64, 128]) \rightarrow \\ &\rightarrow SA(64, 0.4, 32, [128, 128, 256]) \rightarrow SA(16, 0.8, 32, [256, 256, 512]) \rightarrow \\ &\rightarrow FP([256, 256]) \rightarrow FP([256, 256]) \rightarrow FP([256, 128]) \rightarrow \\ &\rightarrow FP([128, 128, 128, 128, K]) \end{aligned}$$

Second configuration :

$$\begin{aligned} &SA(512, 0.2, 32, [64, 64, 128]) \rightarrow SA(128, 0.4, 64, [128, 128, 256]) \rightarrow \\ &\rightarrow SA([256, 512, 1024]) \rightarrow \\ &\rightarrow FP([256, 256]) \rightarrow FP([256, 128]) \rightarrow FP([128, 128, 128, 128, K]) \end{aligned}$$

2.3 Parts description

Here, I explain certain aspects of the implementation. Forwarding a point cloud with PointNet++ is like climbing a hill and then descending it. This is illustrated on the figure bellow taking the second configuration described before.



Capture.png

Farthest Point Sampling Algorithm : Farthest Point Sampling consists in, given a point set Q of size N and an integer $n < N$, choosing a subset P of n points such that the minimal distance between two points of the subset is maximal. Producing such an algorithm can be quite complex and computationally expensive.

Here, we use a simple greedy algorithm :

- 1 choose a point randomly
- 2 take the farthest point p from it
- 3 initialize $P = [p]$
- 4 take point p' which distance to P (minimal distance among all the points of P) is maximal

- 5 add p' to P
- 6 repeat from 4 until P has n points

Grouping : In the grouping task, we start with an original point set, a sample of it, a radius r and a integer G . For each sample point, we make a group of original points. This group is composed of a maximum of G points that are within a query ball of radius r from the sample point. This maximum is here to have balanced groups between sample points on borders and sample points in the middle. In the case where there is not G such points in the query ball, we put several time the same point to reach G . This actually rarely happens since G is chosen smartly.

Pointnet : The goal of PointNet layers is to compute features for sampled points according to points within its group. The feature vectors of points come from the set abstraction at a lower level or the 3d coordinates for level 0. We add (by concatenation) to the feature vector the relative coordinates to sampled points. Each features vector from point of the group is forwarded through a MLP. Then we keep the per-coordinates maximal value among the new feature vectors computed. This gives the feature vectors for sampled point at a higher hierarchical level.

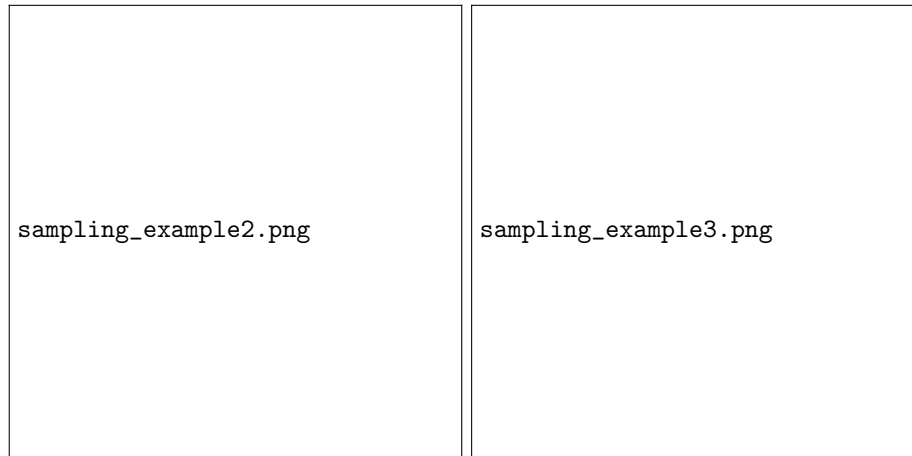
Feature propagation : In this step we propagate features from a higher level in the hierarchy. Basically, for each point p of level l , we compute the mean feature vector of the closest points at level $l+1$ (I used the 3 closest points in my code). This vector is concatenated with the feature vector of p . The concatenated vector is forwarded through a MLP which gives the new feature vector of point p . Notice that arrived at level 0, points do not have features except coordinates (which are not concatenated). Instead, we directly use feature vectors from level 1.

3 Results

In this section, I present some results of the methods used in PointNet++. I also show partial result of the a semantic prediction of my model. Visual results are shown as images produced using mitsuba. These images are made by taking 512 random points from the original 2048 points of a shape. Each result is shown on the shape of a plain.

3.1 Farthest point sampling

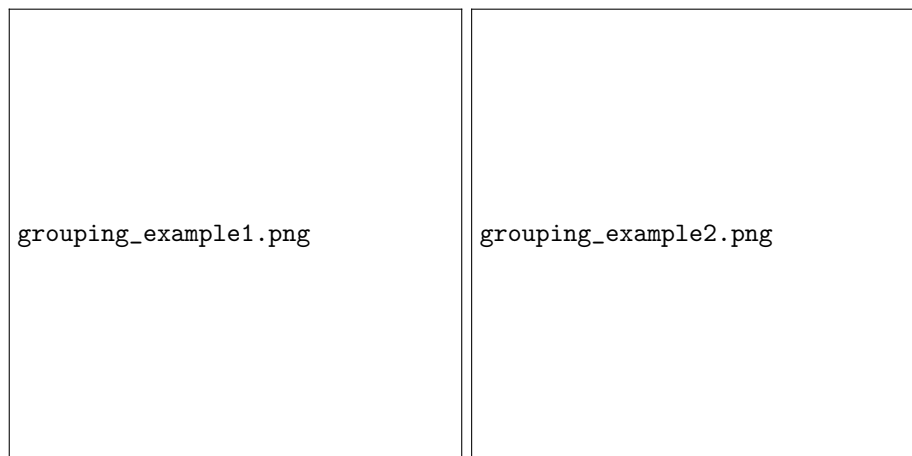
The images bellow show results of my the greedy FPS algorithm I implemented. Sampled points are colored with green/red, the other with blue.



For the left image, I used FPS to compute a sample of size 256 from the original 2048 points. For the right image, I computed a sample of size 64 from the 256 points computed before.

3.2 Point grouping

The images bellow show result of the grouping process. I just displayed some groups within the one I computed since group overlap. Every group member is shown in any colour but in blue. Every other group has it's own colour.



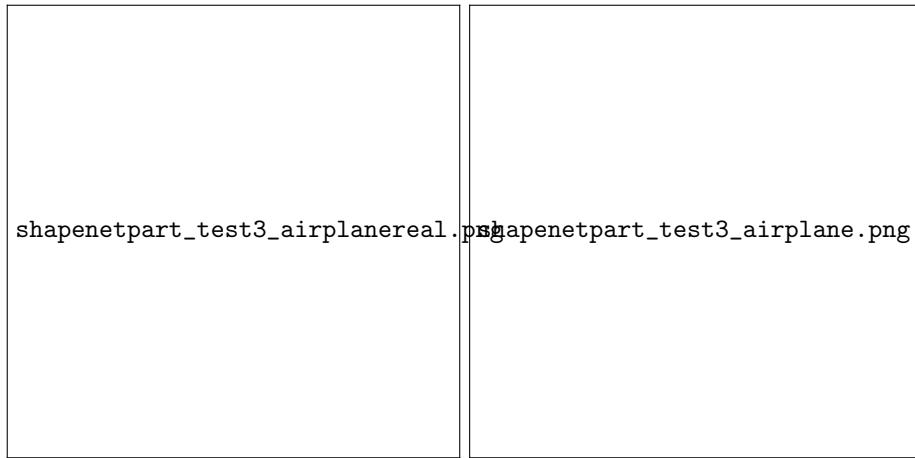
For the left image, groups are made of points within a radius of 0.2 from sampled points. For the right image, the radius is 0.4 from sampled points. In this case points forming groups are points from a higher hierarchical level. This explains

why lot of blue points are mixed with colored points.

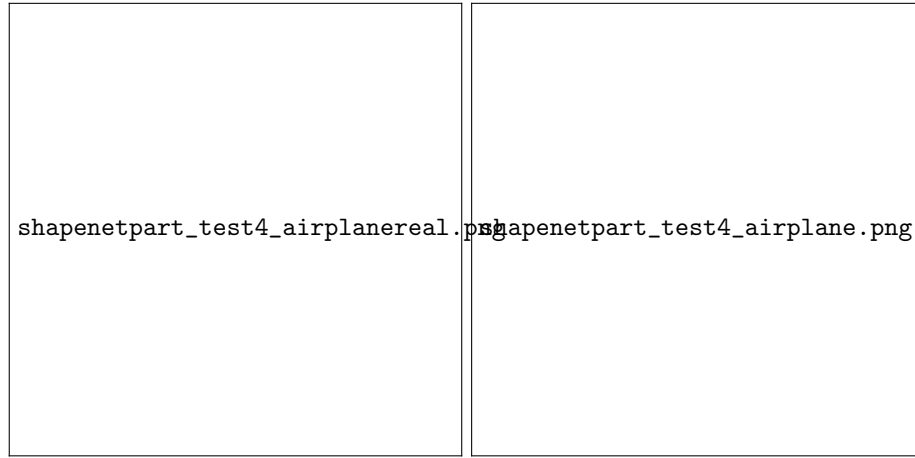
3.3 Segmentation

Here I show the best results I got until now for the final task of semantic labelling. Since I lacked in computation power, I decided to learn only on shapes of plains.

For two different shapes, I display an image showing the real labels on the left and an image showing the predicted labels. Bellow, I showed the prediction table.



Real \ Predicted	Blue	Green	Red	Purple
Blue	635	55	48	482
Green	12	210	121	100
Red	0	85	92	29
Purple	0	11	57	111



Real \ Predicted	Blue	Green	Red	Purple
Blue	668	191	103	465
Green	39	227	76	115
Red	0	31	92	41
Purple	0	0	0	0

This was the most accurate version of the model I achieved to learn. One can see that it gives the right label for 1048/2048 points in the first case and for 987/2048 points in the second case. By making this calculation on 10 shapes, I had an average of 48% accuracy.

Optimizer and Learning : As an optimizer, I mainly used Sochastic Gradient Descend which gave the result I am displaying. I did learn on 128 different shapes by doing 5 epochs to get this result. I also tried Adam, but I haven't made any progress with it.

Computation time : As I mentionned before, I implemented two configurations of PointNet++. The reason I decided to put the first one, it is because it was taking far to much time. Forwarding one shape would take about 3 minutes. Changing to the second configuration allowed me to go down to one minute.

Conclusion

In conclusion, PointNet++ is a method that can just using point coordinates, learn how segment a shape in different parts, by exploiting local and global information. On my results, we can clearly see the great potential it has. However, it is highly limited by the computation power it requires, in terms of

speed and space. To overcome this problem, one would need a powerful GPU, designed to do such calculations. There are many applications of this method, in the industry for instance. I was thinking at another nice application. Point segmentation could be very useful while studying the shapes of ancient ruins. It could be used to recognize or find parts and singularities on ruins that could not be discovered without the help of computers.

References

- [1] Charles R. Qi*, Hao Su*, Kaichun Mo, Leonidas J. Guibas, *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*, Stanford University, 2017
- [2] Charles R. Qi*, Hao Su*, Kaichun Mo, Leonidas J. Guibas, *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*, Stanford University, 2017