3D Classification and Semantic Segmentation: Refining of PointNet and Real-life Application

Xu Gelei 11911138, Dong Shuwen 11912306
Tang Ningzhi 11912521, Qin Haocheng 11911317

November 12, 2021

Abstract

After the thesis PointNet has been published, 3D point cloud data have been a widely used data form to deal with real-world 3D tasks. In this project, we are going to focus on 3D classification and segmentation with 3D point clouds. Firstly, we will try to reappear PointNet network as well as run the network on the official dataset. After that, we are going to repeat the work of Pointnet++ and try to apply the network on a new automatic driving dataset. Then we will focus on the improvement of our network based on math interpretability. In the end, we will concentrate on applying our upgrading network to the SUSTech automatic dataset, achieving a satisfactory result and contributing to the 3D segmentation of the automatic field.

1 Research Question or Problem

With the rapid development of 3D technologies, 3D sensors with 3D point cloud data have become widely used and affordable. Nowadays, the classification and segmentation of 3D point cloud data have numerous applications in different areas including autonomous driving, robotics and medical treatment, the research in this area will undoubtedly speed up the development of these fields.[1]

Nowadays, Deep learning on point clouds has been attracting more and more attention, especially in the last five years. However, deep learning on 3D point clouds still faces several significant challenges, such as the small scale of datasets, the high dimensionality, and the unstructured nature of 3D point clouds. At the early stage, researchers tried to convert it to voxels or collection of views to avoid unstructured nature, while they both have drawbacks as unnecessarily voluminous and obscure natural invariances.

Based on this, we introduce PointNet, handling the invariances with a symmetric function, thus can be applied directly to 3D point cloud. Comparing with the two

methods we mentioned, PointNet has a much more less space and time costs, and a better robustness. Thus we want to re-show this method and analyse why it works.

At the same time, we found some disadvantages of PointNet, such as cannot integrate local structure information, huge loss in maxpooling, and limited effect of discriminative features. We have to refine these by introducing PointNet++ methods with some tricky structure designs.

2 Research Goals and Objectives

Our group decided to research the classification and segmentation of 3D point cloud, which includes preliminary research, thesis reading, previous code reproduce and possible improvements. In the preliminary research part, we will search introduction and the frontier progress of this field to get a deeper understanding of 3D point cloud data and open-source datasets. In the thesis reading part, we will choose a classical thesis to discover its structure and get a deeper insight into 3D data classification and segmentations. After that we will reproduce the code of that thesis, getting the result on the public dataset and our dataset.

At last, we will try to analyze the network in detail, make some possible improvement on the network and compare it with the original work. We will also try to put our improved network on the automatic vehicles dataset to receive some practical results. After this project, we will achieve an accuracy of around 89% in classification on ModelNet dataset and the segmentation accuracy around 78% on S3DIS dataset due to the result of PointNet. We will also try to make improvements on the PointNet and achieve an accuracy promotion of 3% on classification and segmentation. In the end, we aim to create a visualization platform and a practical application. (3D semantic segmentation can be used in autonomous driving to detect pedestrians, etc)

3 Research Design and Methods

3.1 Methodology

The PointNet[2] model is based on deep learning and is mostly completed by Multi-Layer Perceptron(MLP).

First, the input is a $n \times 3$ vector, indicating the location of all the points in 3D point cloud with x, y, z. This input goes through a Spacial Transform Network (T-net) with 3 convolution networks with 1×1 kernels (actually, convolution networks with 1×1 kernels are just an MLP), converting this 3 channel input to a 64 channel vector then 128 and 1024. After convolution, this 256 channel feature will be reduced to a 3×3 vector by 3 fully connected layers. The 3×3 vector is used as an affine matrix to transform the input point cloud, so that the input object is expected to have the same angle related to the origin point. For example, if we classify a point cloud of a human, we always want his head up and feet down, which may make the classification process easier.

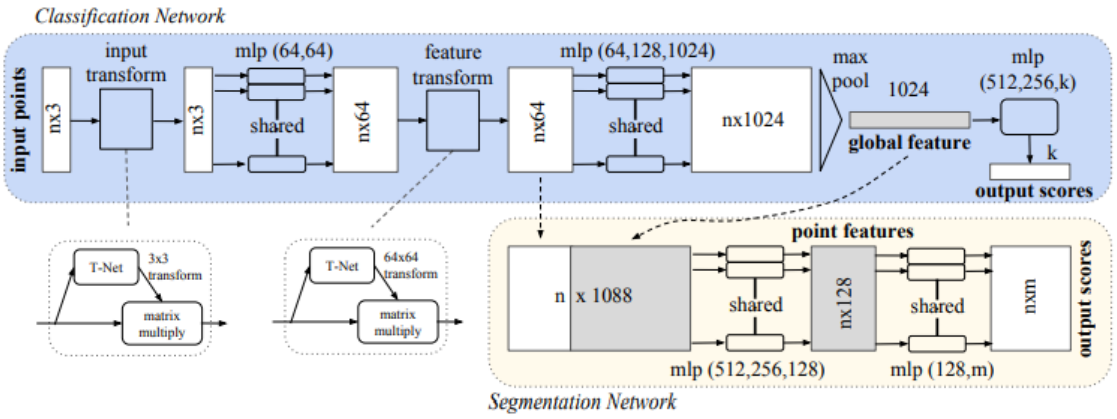

Figure 1: PointNet Architecture[2]

Second, the transformed vector goes through a MLP. The vector grows from 3 channels to 64 and becomes a $n \times 64$ vector.

Third, if we want to do spacial transform on feature again, another STN will be used. This is almost the same as the former T-net, except that the input channel is 64. The output is the transformed $n \times 64$ feature.

Fourth, the $n \times 64$ feature goes through a 2-layer MLP, which rise the dimension of the feature from 64 to 128, then 1024.

Fifth, the $n \times 1024$ feature goes through a max-pooling layer, which will choose the largest number among all n features in each dimension. After that, the $n \times 1024$ feature becomes a 1×1024 feature. According to the author, this max-pooling also solved the problem of the disorder of the point cloud. The max-pooling layer always chooses the largest one, so the order of these n features does not matter. This 1×1024 feature is called a global feature.

At last, if we want to do classification, the 1024 feature will go through another 3-layer MLP(actually 3 full connected layers), decrease the dimension of this 1024 feature to 512, then 256, then k, which indicate the score for k classes that we have in the dataset in a $1 \times k$ vector.

If we want to do segmentation, no matter part segmentation or semantic segmentation, the 1024 feature would be not enough because it is too small to represent the all point cloud (since the point cloud for segmentation is usually big). So we need to add some details about every point.

The method the author takes is to take the former $n \times 64$ feature out, adding the global feature after this feature, forming a new $n \times 1088(1024 + 64)$ feature which suppose to be big enough for segmentation. After that, we use a 4-layers MLP(actually a 4-layer full connected layer) to decrease the dimension of this $n \times 1088$ feature to 512, then 256, then 128, at last, m, which indicate the class that this point should belong to. In such a way, all the points can have a classification and be combined to be segmentation.

PointNet does a good job in point cloud classification and segmentation. However, PointNet has its disadvantages.

PointNet performs not well enough on segmentation because of information loss due to its rough pooling on the features of all points. PointNet++ does some promotion by using set abstraction doing downsampling step by step to reduce the information loss. Another reason for relatively bad performance on segmentation is that PointNet uses the information of points separately to do segmentation without combining them. PointNet++ use sampling and grouping to form local feature by neighbor points to improve its performance.

To get enough information for segmentation, PointNet simply combines global features and points features. But this may not be the best input for the latter layers. PointNet++ use another encoder-decoder frame to connect local and global feature, which makes it seems more rational.

To improve the PointNet, we plan to focus on its advantage on local features due to its separation of points. Our goal is to combine neighbor points to get the local features. We may employ or refer to the relative algorithm in PointNet++.[3]

3.2 Feasibility Analysis

For such a problem, it is just equivalent to fit a map f from points to labels, such f can be always assumed to be nice (which means smooth or continuous). We know that neural network may have a nice ability to fit any continuous f, but the max-pooling function in this model may destroy such ability, we now search for a nice approximation to recover it and estimate the size of the net.

From the text, we obtain the following theorems:

Theorem 1. *$f: 2^R \longrightarrow R \in C^0_{Hausdorff}$, let MAX to be the max pooling function, then for $\forall\ \epsilon > 0$, $\exists$ continuous γ and h, s.t. $|f(S) - \gamma \circ MAX \circ h(S)| < \epsilon$, $\forall\ S \in 2^R$.*

Firstly, we can extend this theorem to functions defined in 2^{R^d}, by letting the σ we took in the proof to be $\sigma(x)=\left\{\frac{[Kx^{(i)}]}{K}\right\}$, then $h_I(x) = \exp - \prod_{i \in I} d(x, [\frac{i-1}{K}, \frac{i}{K}])$, then we can follow the proof's steps. Another idea is to narrow Hausdorff continuity into Holder continuity, to make it easier to estimate the lower bound of the net, we can do this because Holder's continuous function maps neighborhood to neighborhood. Recall : $f \in C^0_{Holder}$ means $|f(x) - f(y)| < C|x - y|^\alpha$, for some constant C and α.

This theorem indicates that we can use a normal neural network before and after max-pooling to get the estimation ability. We now show that we can do it.

Theorem 2. *For an f satisfies theorem 1, it also has that, $\forall$ M=f(S), for some S, $\exists C_M, D_M$, for $\forall C_M \subseteq T \subseteq D_M, f(T) = M$.*

By this theorem, we got to know that we can just take away o(n) points and don't change f too much, indicating robustness.

Theorem 3. *For $f \in C^0_{Holder}$, we want to estimate it with error ϵ as theorem 1 did, then the error of γ and h should be $\epsilon_\gamma = O(\epsilon)$, $\epsilon_h = O(Poly(\epsilon)^{\frac{1}{2}})$. Poly($\epsilon$) means some polynomial of ϵ determined by the Holder coefficient. Also, to achieve this, $K = O(\epsilon^{-2})$.*

Then the only thing we need to do is to show such error of γ and h can be reached. Noticing that h and γ can be estimated by polynomials with the same size error by Weierstrass theorem:

Theorem 4. *Polynomials are dense in $C[\mathbb{R}]$.(Weierstrass)*

Then we can easily see this by the following lemmas.[4]

Lemma 1. *For polynomial $f(x) = \sum_i^p a_i x^i$, $\sum_i^p |a_i| \leq 1$, there exists an MLP with $O(p + log\frac{p}{\epsilon})layers$, $O(log\frac{p}{\epsilon})$ binary step units and $O(plog\frac{p}{\epsilon})$ ReLU steps to estimate f with $\tilde{f}$, s.t. $|\tilde{f} - f| < \epsilon$.*

Lemma 2. *For polynomials $f_i(x)$ satisfies lemma 1, then we can estimate the combination $\sum f_i(x)$ with error ϵ by an MLP with $O(log\frac{1}{\epsilon})layers$, $O(log\frac{1}{\epsilon})$ binary step units and $O(log^2\frac{1}{\epsilon})$ ReLU steps.*

Finally, we got our estimation theorem for PointNet:

Corollary 1. *For $f \in C^0_{Holder}$ with coefficient α, then there exists a PointNet with $O(log\frac{1}{\epsilon})layers$, $O(log\frac{1}{\epsilon})$ binary step units and $O(log^2\frac{1}{\epsilon})$ ReLU steps before max-pool and $O(log\epsilon^{-\frac{\alpha}{2}})layers$, $O(log\epsilon^{-\frac{\alpha}{2}})$ binary step units and $O(log^2\epsilon^{-\frac{\alpha}{2}})$ ReLU steps after max-pool and max-pooling with size $K = O(\epsilon^{-2})$, to estimate f with $\tilde{f}$, s.t. $|\tilde{f} - f| < \epsilon$.*

Which shows us the feasibility of the model and indicates its structure. We can also see from such theories that, in this model T-Net doesn't make enough donations, we can just replace them with some proper normal layers.

Also, when doing segmentation tasks, we apply the same global feature with 1024 dimensions to each point with 64 features, then in such R^{1088} space, these points are all concentrated in a small subspace, so it's equivalent to do the fit on such space. But we know that the measure of subspace is equal to 0 in the whole space, which means it lacks the ability to reveal the performance for the whole space.

On the other hand, this fitting may be sensetive to the sparsity of the data. If we have a fitting polynomial works for a dense data, for example, a cubic hermite interpolation result, then it may doesn't satisfy the need of error for a sparse dataset, for such $error \propto Max|X_i - X_{i+1}|$, which will be bigger in a sparse dataset.

Thus, we wanna introduced PointNet++ to refine these things.

4 Initial Results

We have evaluated the performance of PointNet on three different tasks on 3D point cloud: Classification, Part Segmentation, and Semantic Segmentation. The evaluation results have a small gap but are generally comparable to those reported in the paper. We think the gap results from much fewer running epochs than reported. The result is shown in the table.

The code repository we used is PointNet_PointNet++_Pytorch[5]. We ran the training and testing code on CUDA(10.0).

Table 1: Initial Results

Task	Metric	Experimental	Reported	Difference
Object Classification	Accuracy	84.1	89.2	-5.71%
Part Segmentation	mIoU	78.5	83.7	-6.21%
Semantic Segmentation	mIoU	78.91	78.62	+0.37%

4.1 Object Classification

For the classification task, we have trained the model for about 5 hours with 20 minutes. The epochs that successfully trained is 18. Then we tested it on the testing dataset and gain an average accuracy over all classes is 84.1, with has a 5.1 gap from reported (89.2). We think the reason is from lacking training epochs (200 reported).

For the classification task, the dataset used is ModelNet[6]. We have trained the model for about 5 hours with 20 minutes. The epochs that successfully trained is 18. Then we tested it on the testing dataset and gained an average accuracy over all classes is 84.1, with has a 5.1 gap from reported (89.2). We think the reason is from lacking training epochs (200 reported).

4.2 Part Segmentation

For the part segmentation task, the dataset used is ShapeNet[7]. We have trained our model for 12 epochs in about 1 hour and 23 minutes. Here the metric we evaluated is mean Intersection over Union ($mIoU$). The formula is defined as follows:

$$mIoU = \frac{TP}{TP + FP + FN}$$

where TP is the number of points whose prediction and label are both positive, FP is the number of points whose predictions are positive but labels are negative, and FN is the number of points whose predictions are negative but labels are positive.

The testing $mIoU$ we got is 78.5, which has a 5.2 gap from the reported (83.7). The situation may be from lacking training epochs (203 reported).

4.3 Semantic Segmentation

For the semantic segmentation task, the dataset used is S3DIS[8]. Since there is a pre-trained model in the repository, and the training time is too long. We just evaluated it on the testing set. The testing $mIoU$ we got is 78.91, which has a little higher than reported (78.62).

We also have running the visualization script in the code repository and got one image showing the semantic segmentation result.

We plan to train and test PointNet on other general datasets and adjust the layer structure and hyper-parameters to gain higher evaluation results. We also plan to

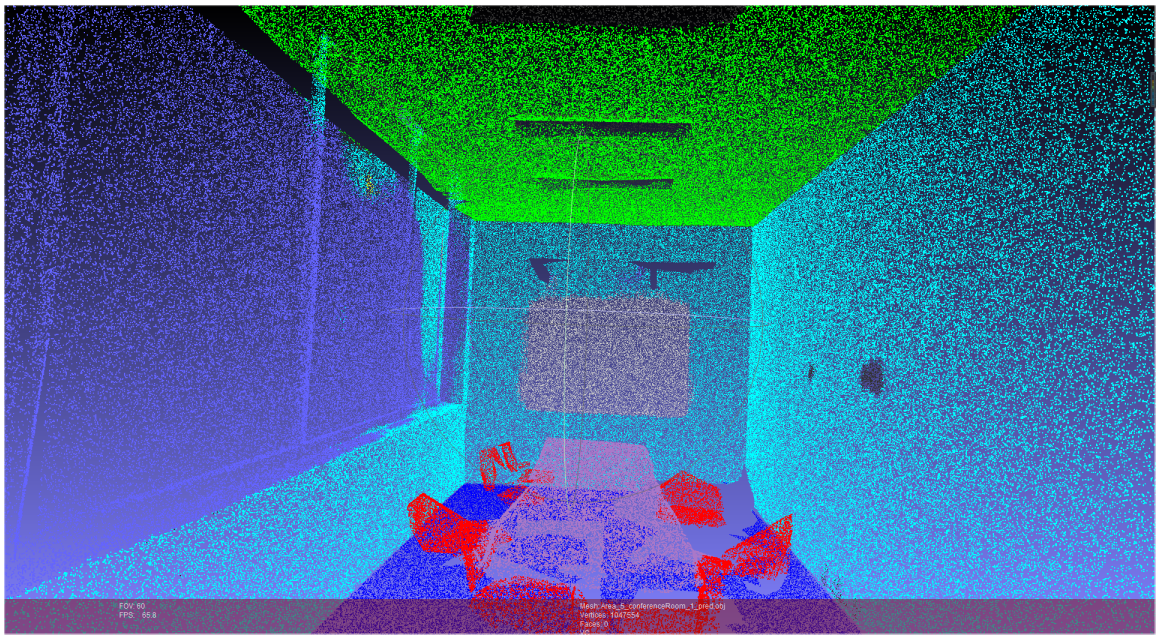

Figure 2: Visualization of Semantic Segmentation Result

explore PointNet++ and other models designed for 3D classification and semantic segmentation tasks to make a comparison and analyze the reason for the difference.

5 Staffing Plan

Qin Haocheng majors in Statistic and is adept at mathematical derivation. He is responsible for theoretical background research and theoretical derivation.

Dong Shuwen majors in Computer Science and is adept at data collecting and coding. He is responsible for extra data collecting and model analysis.

Tang Ningzhi majors in Computer Science and is adept at data collecting and coding. Thus he is responsible for code running on the server and point cloud visualization.

Xu Gelei majors in Computer Science and is adept at data collecting and coding. Thus he is responsible for information organizing and network improvement.

6 Timeline

The figure above describes the timeline of our project, which is divided into three parts: Early-stage, mid-term, and later stage.

After the project theme was decided, we first did abundant research on 3D point cloud, which contains a brief introduction of 3D point cloud dataset and a summary of previous work. Afterward, we choose a classical thesis Pointnet to repeat. Up to now we have finished the above tasks and got our initial results.

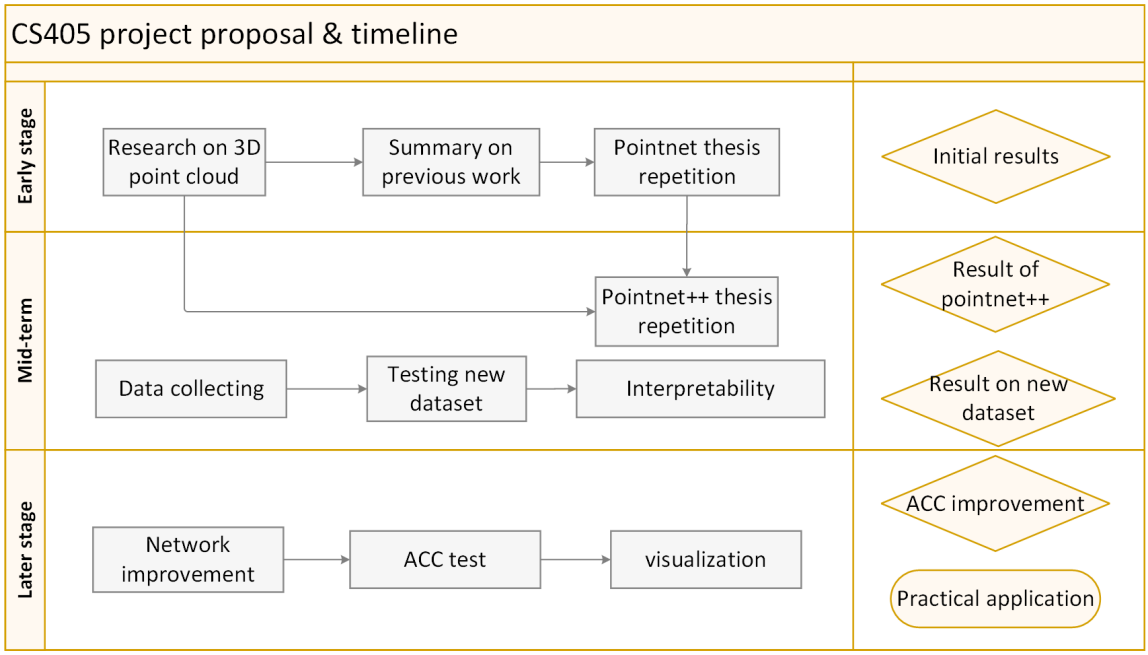

Figure 3: Timeline of Our Project

In the mid-term stage, we will focus on two parts: the repetition of PointNet++ and the test on newly collected datasets on automatic driving. PointNet++ is an advanced version of PointNet. By comparing the result of PointNet and PointNet++, we will gain a deeper understanding of the structure of PointNet series. At the same time, we will run the Pointnet series structure on other point cloud datasets to acquire robustness. If the outcome accuracy drops, we will try to interpret the result without analysis of the net. After this period, we will gain the result of Pointnet++ as well as the result on the new dataset.

When it comes to the later stage, we will pay attention to the improvement of the network to achieve higher accuracy, which includes the adjustment of our network, the accuracy-test, and the interpretability of our adjustment. Since point cloud can be visualized straightforwardly, we will do a visualization of point cloud to present our work. Moreover, we are looking forward to putting our achievements into practical application. For instance, 3D semantic segmentation can be used in autonomous driving to detect pedestrians. We aim to end up the project with a functional network that can be 'used' in practice.

References

[1] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "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, 2017.

[3] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[4] S. R. Liang, Shiyu, "Why deep neural networks for function approximation?," *eprint arXiv:1610.04161*.

[5] X. Yan, "Pointnet/pointnet++ pytorch," 2019. https://github.com/yanx27/Pointnet_Pointnet2_pytorch.

[6] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas, "A scalable active framework for region annotation in 3d shape collections," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 6, pp. 1–12, 2016.

[7] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

[8] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese, "3d semantic parsing of large-scale indoor spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1534–1543, 2016.