Graph PointNet and Graph PointNet++
Machine learning for Computer Vision Project Work

Ludovico Granata
Master's Degree in Artificial Intelligence, University of Bologna
ludovico.granata@studio.unibo.it

Abstract

This study aims to investigate the efficacy of integrating Graph Neural Networks into PointNet and PointNet++ for the task of 3D part segmentation. We perform experiments using Graph Convolutional Networks and Graph Attention Networks, and evaluate two methods for constructing the graph: Nearest Neighbor and Ball Query. Our findings suggest that utilizing Graph Attention Networks and the Ball Query method can yield slightly improved results compared to using PointNet and PointNet++ alone.

1 Introduction

Point Cloud Part segmentation is the task of assigning part category label to each point of a Point Cloud for a specific object model. This is an active area of research in the field of 3D Computer Vision and it has numerous application in the context of Robotics or Augmented Reality. The task is particularly challenging as Point Clouds have no explicit structure and the sampling density is not even.

The PointNet(Qi et al., 2016) paper, published in 2017, was one of the first deep learning-based approaches for point cloud processing, and showed that deep neural networks can effectively process unstructured point clouds. Also the subsequent paper PointNet++(Qi et al., 2017) had a wide impact on the field, inspiring many other works.
We propose to start from this important works and to incorporate Graph Neural Networks motivated by the idea that this type of networks are very effective when working with unstructured data. PointNet and PointNet++ will serve to build strong point features, that will be utilized in the graph network.

We trained and tested the models on the ShapeNet Dataset. We experiment using Graph Convolutional Network, Graph Attention Network and also we tried to use two different method to build the graph structure: Nearest Neighbor and Ball Query.

2 Background

2.1 Graph Convolutional Network

Graph Convolutional Networks (GCNs) (Kipf and Welling, 2016) are deep learning models designed to work on graph-structured data. Given the structure of the graph, described by an the adjacency matrix, we can define the output H^{l+1} of the convolution as:

$$H^{l+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$$

where $\tilde{A}$ is the adjacency matrix summed with the identity matrix, $\tilde{D}$ is defined as $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and is used to normalize the adjacency matrix. The matrix $W^{(l)}$ contains the trainable weights and the $\sigma(\cdot)$ is the activation function like ReLU.

2.2 Graph Attention Network

Graph Attention Neworks (GATs)(Veličković et al., 2017) are very similar to GCNs but the key difference is that GATs allow to assign different importances to nodes of the same neighborhood using the attention mechanism. The attention coefficient are computed by

$$e_{ij} = a(\mathbf{W}h_i, \mathbf{W}h_j)$$

h_i and h_j are the features of two nodes, that will be multiply by the weight matrix $\mathbf{W}$. The resulting features $\tilde{h}_i$ and $\tilde{h}_j$ are concatenated and will the input to $a(\cdot)$, a single-layer feedforward neural network. All the coefficient e_{ij} of the same neighborhood will be passed to a softmax to make them sum up to 1. The normalized coefficient will be used to do a weighted sum over the neighbors nodes.
The use of multi-head attention can enhance the performance of attention mechanism in graph neural networks. Instead of having a single weight

matrix **W** and a single coefficient vector a, multiple such sets, each independent from the others, are employed. The final results can be obtained by either averaging or concatenating the outputs of these multiple heads.

3 Data

The dataset we used for this entire project is ShapeNet(Chang et al., 2015), in particular we used a subset of it called ShapeNetPart. ShapeNetPart includes 16,881 3D models from 16 categories annotated with 50 segmentation parts in total. For each model it provides the x,y,z coordinates of each point and also the coordinate of the normal with respect to each point.

4 Architecture

The proposed architecture is shown in Fig. 1 and consists of two stages: in the first stage, a feature extractor is employed to extract features for each point in the point cloud, in the second stage, a Graph Neural Network is utilized to provide a more comprehensive understanding of the object, thereby enabling better classification of each part.

4.1 Feature extractor

The feature extractor is implemented by PointNet(Qi et al., 2016) by removing the last fully connected layer that outputs the final scores. With the intent of testing the validity of our method on larger spectrum, we also use PointNet++(Qi et al., 2017) for our experiment, using it in the same fashion as PointNet[1]. The input to the feature extractor are the 3D coordinate of each point and also their normal, so that we have 6 values for each point. The output is an array of size 128 for each point.

4.2 Graph Neural Network

For each Point Cloud, a graph is constructed in a online fashion by representing each point as a node and by establishing relationships between nodes using two different methods, Nearest Neighbor and Ball Query for comparative analysis during experimentation. The Nearest Neighbor approach considers relationship of a point with the closest n points, while the Ball Query approach considers each point's relationship with all points within a specific radius. In our experiments, we set n=15 and the radius to 0.1 to ensure a comparable number of connections.
Once the graph is constructed we can apply graph convolution, here we employ 3 layers of graph convolution with residual connections. We experiment with two type of graph neural network, one is Graph Convolutional Network(Kipf and Welling, 2016) and the second is Graph Attention Network(Veličković et al., 2017). The output is a array of length 128 for each point, that will be the input to a shared fully connected layer to obtain the final scores.

5 Experimental setup and results

5.1 Setup

We performed all of our experiments using a single NVIDIA RTX 2080 Ti graphics card.
For all of our experiments, the following parameters were kept the same: we utilized the Adam optimizer with a learning rate of 1e-3, and a step scheduler was employed with a step size of 20 epochs and a decay of 0.5. The batch size was set to 8, and early stopping was implemented with a patience of 10. In all of our experiments, 1024 points were randomly selected from each Point Cloud.

5.2 Metrics

To evaluate and compare our results, we use the mean Intersection-over-Union (mIoU) metric. For a single object, the mIoU is calculated by determining the number of points that are correctly classified for each part and dividing it by the number of points that are classified as that part, either in the estimations or in the ground truth; the results are then averaged for each part. We calculate the mIoU for each category, and we also compute the class average mIoU, which is obtained by averaging all the categories together. Additionally, we also compute the instance average mIoU by averaging the mIoU of each individual object, regardless of its category.

5.3 Results

In 1 and in 3 we show the results on the test set when using PointNet as feature extractor, while in 2 and in 4 we use PointNet++. From the results in 3 and 4 it is evident that using Graph Convolutional Network does not enhance performance according to the used metrics. The use of the attention mechanism for graph, however, leads to noticeable improvement both in term of Class average mIoU and

[1]Implementation come from https://github.com/yanx27/Pointnet_Pointnet2_pytorch

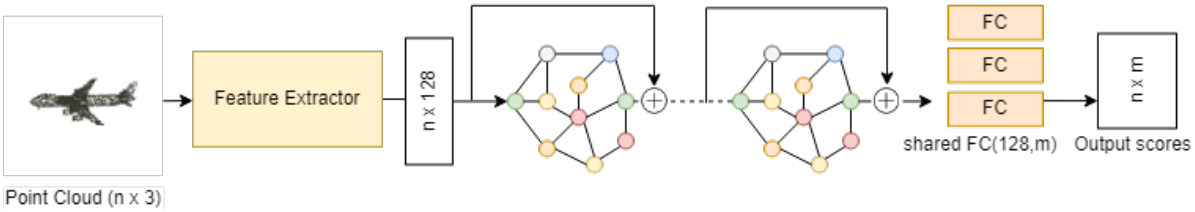

Figure 1: Architecture overview

Instance average mIoU. Additionally the method used to construct the graph has a significant impact, as the use of the Ball Query method over the Nearest Neighbor bring substantial improvement.

6 Conclusion

The purpose of this study was to investigate the potential benefits of using Graph Neural Networks with features extracted using PointNet and PointNet++ for 3D part segmentation task on Point Cloud. The results indicate that utilizing Graph Convolution Network is not effective, however, incorporating attention mechanism into the graph does result in a significant improvement over the performance of PointNet and PointNet++ alone. This is maybe due to the ability of GATs to more effectively capture structural information by assigning appropriate importance to neighboring nodes with respect to GCNs.

Additionally, we observed that the method used to construct the graph has an impact on the results. Utilizing the Ball Query method lead to better performance compared to the Nearest Neighbor approach.

Future work of this study could involve exploring more recent works as feature extractor, instead of using PointNet and PointNet++. Moreover, experimenting with different Graph Neural Network architectures could also lead to improved results.

7 Links to external resources

GitHub public implementation of the work

References

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907.

Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2016. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2017. Graph attention networks.

A Appendix: Tables

Models	Graph Type	Graph Method	Airplane	Bag	Cap	Car	Chair	Earphone	Guitar	Knife	Lamp	Laptop	Motorbike	Mug	Pistol	Rocket	Skateboard	Table
PointNet	x	x	**82.5**	80.0	**86.2**	75.2	89.8	73.3	**90.8**	87.5	81.1	95.1	63.6	93.1	80.2	52.7	75.3	82.4
Graph PointNet	GCN	NN	80.6	77.0	81.8	73.3	89.7	72.5	90.1	87.4	82.0	95.2	61.4	91.8	80.0	61.1	69.8	82.7
Graph PointNet	GAT	NN	82.3	79.4	84.8	**76.3**	**90.1**	72.9	90.7	86.3	**82.4**	**95.4**	63.0	**95.2**	**81.5**	59.8	**75.6**	82.5
GraphPointNet	GAT	BQ	82.2	**80.4**	84.5	74.5	90.0	**76.3**	90.1	**89.0**	82.3	95.3	**65.4**	93.5	80.4	**61.7**	72.0	**83.2**

Table 1: mIoU for each category with 1024 points with PointNet feature extractor.

Models	Graph Type	Graph Method	Airplane	Bag	Cap	Car	Chair	Earphone	Guitar	Knife	Lamp	Laptop	Motorbike	Mug	Pistol	Rocket	Skateboard	Table
PointNet++	x	x	83.4	78.0	**86.2**	78.5	**90.9**	74.6	91.2	**87.7**	83.9	95.2	**71.2**	94.3	79.5	59.0	76.8	82.8
Graph PointNet++	GCN	NN	82.2	78.3	80.1	78.2	90.5	**74.8**	90.6	86.7	**84.5**	95.1	60.7	93.1	78.2	**59.7**	75.0	83.1
Graph PointNet++	GAT	NN	**83.6**	82.9	85.4	78.6	90.2	**74.8**	90.6	86.7	84.2	95.1	60.8	94.3	**79.6**	**59.7**	75.6	83.1
Graph PointNet++	GAT	BQ	83.3	**84.9**	85.0	**79.1**	90.6	74.1	**91.3**	87.3	84.4	**95.5**	69.9	**94.9**	79.3	59.3	**77.8**	**83.8**

Table 2: mIoU for each category with 1024 points with PointNet++ feature extractor.

Model	Graph Type	Graph Method	Class avg mIOU	Instance avg mIOU
PointNet	x	x	80.6	84.2
Graph PointNet	GCN	NN	79.8	83.9
Graph PointNet	GAT	NN	81.1	84.5
Graph PointNet	GAT	BQ	**81.3**	**84.6**

Table 3: Class avg mIoU and Instance avg mIoU using 1024 points with PointNet as feature extractor.

Model	Graph Type	Graph Method	Class avg mIOU	Instance avg mIOU
PointNet++	x	x	82.1	85.4
Graph PointNet++	GCN	NN	80.7	84.9
Graph PointNet++	GAT	NN	82.2	85.7
Graph PointNet++	GAT	BQ	**82.5**	**86.0**

Table 4: Class avg mIoU and Instance avg mIoU using 1024 points with PointNet++ as feature extractor.