3D Part Segmentation with Kernel Point Convolution

Genglin Liu Guanghao Wei

May 11, 2021

Abstract

3D cloud segmentation is a popular problem in stereo computer vision and graphics. This problem can be solved using 3D/geometric deep learning, and we will investigate this problem by employing the current state-of-art architecture kernel point convolution (KPConv). We replicate the experiment on Stanford S3DIS indoor scene dataset using rigid KPConv, and evaluate this network on a new dataset, Shapenet-Part, using our own data loading and evaluation pipeline. We also experimented with additional dropout layers in the network and different loss functions

1 Introduction

Accurately and efficiently processing 3D objects has been gaining popularity in 3D deep learning community and we believe that it has strong potential in real-world applications such as supporting the autonomous driving systems. The goal of our course project is to investigate the shape segmentation problem and implement noval architectures to improve the segmentation performance. We plan to apply one of the most advanced network architectures, kernel point convolution (KPConv) for the task of 3D shape segmentation on the Shapenet-Part dataset. We will perform segmentation on point cloud data. First we show a replication study on the official implementation of KPConv and how we achieved the same performance on a large indoor scene dataset using multi-GPU training. Then we discuss our interest in the 3D part labeling of individual objects and how we managed to perform part segmentation on the Shapenet-Part dataset. The rest of this manuscript will have a more detailed narration of our effort. Section 2 reviews the related work on 3d point cloud segmentation. Section 3 discusses our methods of collaboration, experimental designs and the pipeline that we adopt. Section 4 has in-depth evaluation and analysis of the results that we observe from the experiments.

2 Related Work

There have been numerous studies on 3D shape segmentation. In as early as 2010, Kalogerakis et al [1] have presented data-driven methods to learn 3D mesh segmentation and labeling. Recently, we also see approaches that project 3D images to 2D space and perform segmentation using the existing 2D image segmentation models [2] . These type of models project 3D objects onto the 2D planes and treat them with 2D models such as U-Net. We have more view-based approaches with high performances such as ShapePFCN by Kalogerakis et al [3]. We also inspect point-based methods such as PointCNN [4] and SplatNet [5] which specifically operates on point clouds, and then there is the current State-of-the-Art in ShapeNet-Part benchmark, KPConv with kernel point and deformable Convolutions [6]. We will spend most of our time studying KPConv and replicating the study in PyTorch.

3 Methodology

3.1 Experimental Designs

We have three primary goals in mind:

1. Replicate the scene segmentation experiment on S3DIS

2. Modify the KPConv architecture or loss function and explore the impact such modifications have on the same experiment with S3DIS.

3. Explore another dataset such as Shapenet-Part or PartNet and perform another cloud segmentation task on lighter, individual objects instead of a large indoor scene.

Due to the time constraint we decided to focus on the segmentation network KPFConv as opposed to the classification network.

3.2 Project Pipeline and Building Environment

At the same time where we were using the official KPConv implementation released by Hughes et al. we also attempted to create our own pipeline with a slightly different project structure and configuration setup.

We customized our project structure to contain source code for model, dataset preprocessing and loading, training and evaluation scripts, along with configuration files, virtual environment setup, figures, jupyter notebooks for temporary sketches or visualizations, documentation and written report all compactly together. We originally wanted to completely reconstruct our project and make it independent of the official repository that implements the KPConv paper.

We soon realized that the readability of our project gets compromised if we choose to adopt large segments of the official implementation, and that due to the many dependencies in the original script for training, we decided to replicate the experiments in a forked version of the official implementation with the hope that we can accurately replicate the results, while also developing the under-construction code in our own repository.

3.3 Shapenet-part Preprocessing and Data Loading

We found it particularly challenging to load a new dataset to the KPConv pipeline and run a cloud segmentation task on the new dataset. We decided to see if KPConv is capable of doing a part segmentation on Shapenet-part, but since the file structures of the two data sets are so vastly different, it took us a significant amount of time trying to recreating and debug the dataloading script to enable KPConv to take the new inputs. We had to reference the preprocessing and data loading procedure of the S3DIS dataset and modify the procedure such as creating the network input, matching tensor dimensions, sampling from the data loader, concatenating batches and dealing with concurrent training with a multi-threaded implementation.

4 Experiments and Evaluation

4.1 Replication Study

The first task that we tried to perform was to replicate the experiments done on the S3DIS (Stanford Large-Scale 3D Indoor Spaces) dataset. We references the original KPConv paper, which reported a 65.4 mIoU performance on 3D scene segmentation. During our effort of replicating this experiment, we encountered the following obstacles.

We had access to Nvidia 2060Ti, TitanX and m40 GPUs and we attempted to replicate the experiment on all of those hardwares. The main performance metric we evaluated our tasks with is Intersection-Over-Union(IoU). IoU computes the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. For this multi-class problem, we will use the mean IoU (mIoU) over each class label to evalutate our model.

One of the important configurations is the input sphere radius. It turned out that we experience severe GPU memory shortage when we run the provided training script using the default radius of 1.5 on most of our graphic cards. To avoid the memory error we tried to lower the input sphere radius, and we ended up conducting the following experiments.

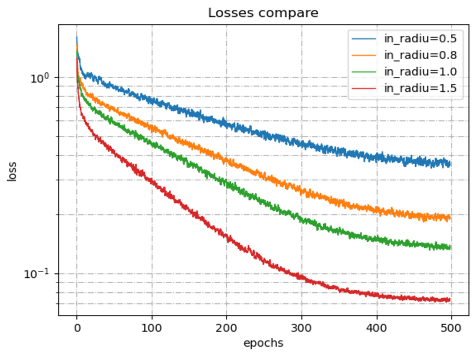


Figure 1: replication of the loss plots with different input sphere radius values

At almost 350th epoch one of our experiments with all input radius values were able to achieve a consistent validation mean IoU (mIoU), agreeing with what the paper reported which is that the train loss should be able to converge within 400 epochs. With radius of 1.5, the network eventually achieved a mIoU of 64.5% after running for almost 90 hours, indicating a lengthy but successful replication.

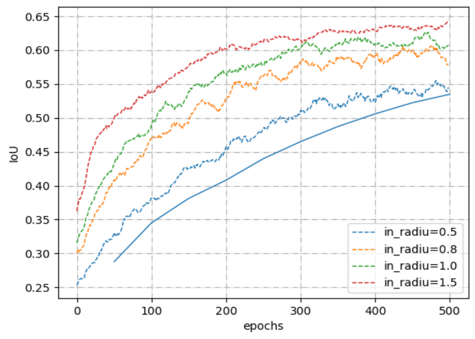


Figure 2: replication of the mIoU curves

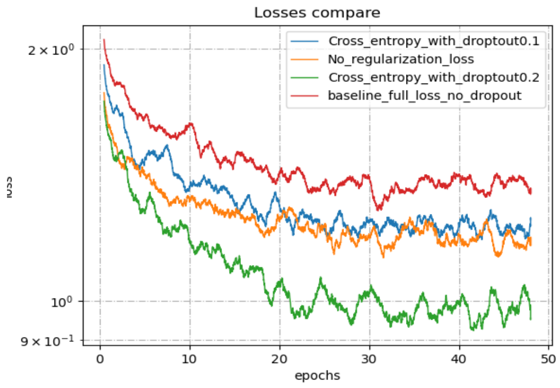
In terms of running time, we observe that the running time is linear in the number of epochs, and experiments with larger input sphere radius take longer time. One of the biggest challenges for us is the amount of time it requires to run each experiment. As we see here, each 100 epochs take well over 5 hours, and consequently we were not able to try out many different configurations to replicate the experiments presented in the original paper.

4.2 Modifying the Network Loss and Architecture

We also decided to perform a few exploratory tasks to see if the network can hold on to a similar level of performance with some minor modification to the layers or the loss functions. We start by reviewing the architecture. The encoder part of the network consists of 13 Resnet Bottleneck Block, each also consists of a KP convolutional layer and a residual block denoted as the 'unary block'. The unary block contains batch normalization, and our modification will add a dropout layer right after the batch normalization and the ReLU activation.

For loss functions, we conducted a small ablation study and a replacement experiment. The original KPConv network loss is a combination of output loss (defined by a cross entropy loss) and regularization loss which again consists of a 'fitting' and 'repulsive' loss that essentially makes the kernel points more evenly distanced from each other. In the original paper the authors argued for the benefit of this regularization loss, and we tried to see how much of a difference it actually makes in training.

We performed four experiments that ran for 50 epochs and 500 training steps each, on the S3DIS dataset. The results are relative to each other so we can still make some useful observations and inferences.



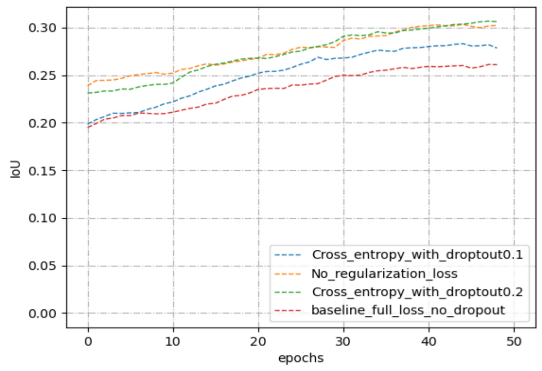


Figure 3, 4: Loss and mIoU of different dropout probabilities and partial loss

As we can see from the plots, the baseline uses the combined loss and no dropout layers. The network tends to have relatively lower loss values with dropout of 0.2 embedded inside each of the unary block. Our speculation is that the dropout layers might bring a lighter network to train since some of the neurons during the forward pass are being deactivated; furthermore it prevents the deep network to overfit on our dataset. We noticed that higher dropout probability (0.2) makes the loss decrease faster but then it appears to be a less stable.

Having the regularization loss removed from the loss calculation, we observed the the loss curves look almost identical, as if the curve just got shifted downward by a constant. In terms of validation performance, the model without regularization loss and dropout seemed to perform even better than the original implementation while taking less time, at least from our 50 epochs of observation. We also attempted to train the network with different loss functions such as KL Divergence or Multi-label Soft Margin Loss, Considering that the authors recommended 400 epochs of training and we have seen stable increase of mIoU, we are optimistic that with sufficient time and training, dropout might be able to improve the performance of KPConv.

4.3 Cloud Part Segmentation with Shapenet-Part

4.3.1 Challenges with PartNet

Besides the indoor scene S3DIS, we also explored Shapenet-part dataset [7] and PartNet [8] on our own and we investigated how KPConv would perform on these two datasets.

The challenge we almost immediately met when we started looking into PartNet was that the dataset is too large. We obtained more than 100GB of compressed data and more than 300GB of data which was simply too large for us to process. PartNet also consists of a structure that is relatively hard to organize and preprocess. With our time and compute resources, it was almost infeasible for us to perform tasks on such a large scale dataset without any pretrained models and it was difficult for us to even pick out a few classes to train on.

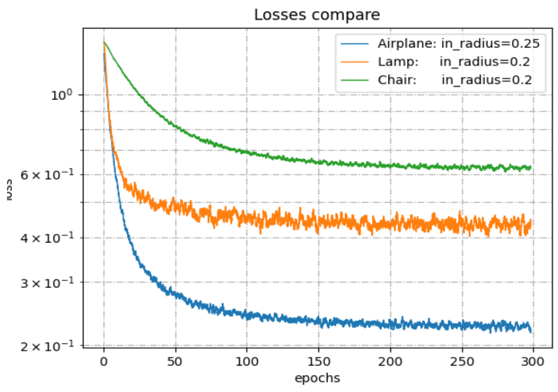


Figure 5: Loss plot of three categories

We then agreed that upon the time constraint, we first move on to the Shapenet-Part. ShapeNet-Part is another dataset that consists of point clouds of 3D objects and supports part segmentation task.

4.3.2 Experiments on Shapenet-Part

Shapenet-part contains per-point labeling of 31,963 models in 16 shape categories, therefore being a very good candidate for point cloud part segmentation. With the time constraint, we fitted KPConv networks for three categories that had more numbers of plane, car and lamp.

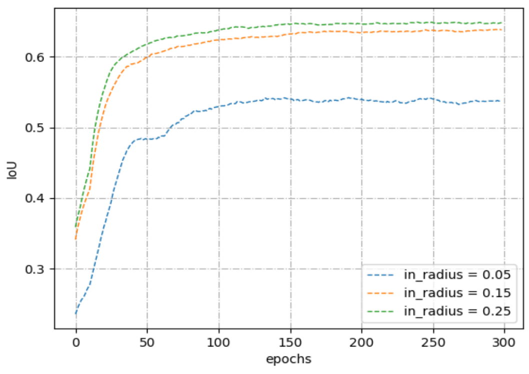


Figure 6: mIoU plot of three categories

We performed part segmentation on these networks and the biggest challenge was to find appropriate hyperparameters. We noticed that these objects from Shapenet are under a very different scale compared to the large indoor scenes in S3DIS, along with the scale is the very different densities in the point cloud. As KPConv considers a convolutional operation spatially, we speculate that it requires a very different set of hyperparameter configurations for the network to reach a good performance in terms of segmentation mIoU.

We also obtained experimental results from segmentation of three separate categories. As we observe from the loss plots, airplane tends to have the fastest converging one and respectively it has the highest validation IoU. Lamps and Chairs tend to be harder to segment and we speculate that the reason lies in the label distribution. Lamp and Chair has relatively fewer and more imbalanced labels so the training might not be as easy.

4.3.3 Varying Feature Dimensions

KPConv processes points and encodes them into a feature vector before the segmentation task. Besides the input points of shape (N, 3), we have a feature tensor of shape (N, D) concatenated to it. We explored the effects of having different numbers of feature dimension before the training. In a 5-dim feature tensor, the first one is the encoded 1's, the next three features are the RGB color of each point, and the final dimension denotes the z-axis coordinate of the point for spatial information. In a 1 dimensional feature, we only have initialized all 1's.

Here are the results after 300 epochs of training on airplane shapes:

We observed that with 1 dimensional feature the network virtually does not learn anything, the loss barely decreases and IoU stays at a very low level. We tried to have a 4-dimensional feature input (without the z coordinate) and it turned out to perform a little less than feature dim of 5. Because of time constraint we did not get to plot that one fully, but we speculate that a larger feature dimension introduces a larger feature space and increases the expressive and descriptive power of our model. It therefore performs stronger.

4.4 Discussion and Future Work

In this project, we studied kernel point convolution, a recent and advanced deep learning model that processes 3D point cloud data. To get to know how the pipeline works we first conducted a replication study and replicated the S3DIS indoor scene segmentation task. As we reached a similar performance reported in the paper, we also explored another dataset, Shapenet-part, and conducted object part segmentation using a different data loading pipeline. The part segmentation ended up having lower performance than we anticipated but the results showed that we were moving in the right direction. In addition to that, we also made several minor modifications to the network and its loss function and performed experiments on a smaller scale. We challenged the idea of the regularization loss advertised in the paper, and added dropout layers to the network and observed a potential increase in performance. For future work, we would like to be able to load new dataset into the pipeline more easily and have the network be more robust to point clouds under different scales.

References

[1] Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. “Learning 3D Mesh Segmentation and Labeling”. In: 29.4 (July 2010). ISSN: 0730-

0301. DOI: `10.1145/1778765.1778839`. URL: `https://doi.org/10.1145/1778765.1778839`.

[2] Yecheng Lyu, Xinming Huang, and Ziming Zhang. *Learning to Segment 3D Point Clouds in 2D Image Space*. 2020. arXiv: `2003.05593 [cs.CV]`.

[3] Evangelos Kalogerakis et al. *3D Shape Segmentation with Projective Convolutional Networks*. 2017. arXiv: `1612.02808 [cs.CV]`.

[4] Yangyan Li et al. *PointCNN: Convolution On $\mathcal{X}$-Transformed Points*. 2018. arXiv: `1801.07791 [cs.CV]`.

[5] Hang Su et al. *SPLATNet: Sparse Lattice Networks for Point Cloud Processing*. 2018. arXiv: `1802.08275 [cs.CV]`.

[6] Hugues Thomas et al. "KPConv: Flexible and Deformable Convolution for Point Clouds". In: (2019). arXiv: `1904.08889 [cs.CV]`.

[7] Li Yi et al. "A Scalable Active Framework for Region Annotation in 3D Shape Collections". In: *SIGGRAPH Asia* (2016).

[8] Kaichun Mo et al. "PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.