Project Report : 3D Object Classification with Point Cloud Dataset

Hengye Jing
CS7643 Group126
Georgia Institute of Technology
hjing31@gatech.edu

Tianci Zhong
CS7643 Group126
Georgia Institute of Technology
tzhong9@gatech.edu

Yichen Wang
CS7643 Group126
Georgia Institute of Technology
ywang3716@gatech.edu

Abstract

Correctly classifying 3D objects from this data is important for real-world 3D scene understanding. Cloud point is an important type of geometric data structure and many real-world datasets are represented in this format, such as data collected by self-driving cars, robotics, etc. Current models for 3D cloud point data classification problems are complex with deep structure and expensive to train, such as PointNet. In this project, we explored the classification problem of 3D cloud point data. Inspired by PointNet, 3DShapeNet and derivative models, we proposed a simplified PointNet architecture that can achieve similar performance than the original model, and tested its performance on ModelNet10 dataset.

1. Introduction

Classification problems of 3D shapes are widely adopted in real-world detection by self-driving cars, robotics, medical treatment, etc. 3D data can be represented with different formats. The classical approaches usually involve converting 3D cloud point data to 3D voxel grids which is an expensive process that requires higher memory for storage. PointNet model [4] can take 3D cloud points coordinates representation, also known as point-based representation. This coordinate representation of 3D cloud data is a much-condensed data structure compared to voxel grids, so it saves more memory when storing the training and testing data. A recent publication states that point-based methods are the superior choice for point-cloud processing as they "do not introduce explicit information loss" [1].

Processing cloud point information is different from processing 3D voxel grids, a similar format as pixels in 2D images but represented as boolean in 3D space. Deep learning has achieved great results in the 2D images space and has many mature models but in the 3D cloud points space, there are still many opportunities for improvements. Many of the well-known models for 3D cloud points classification involve complex structures and millions of trainable parameters, such as PointNet.

We chose to build our model based on PointNet, which is a cloud point-based method. Therefore, our module will inherit those superior properties of point sets. The three main properties of the point sets are unordered, main interaction information among points, and invariance under basic transformations . The current version of the PointNet model is complex and has a deep structure which takes a long time to train the model. We reimplement the PointNet in the paper and train the model on the ModelNet10 dataset [5] with Colab default GPU. Our simpler Go3D model takes ten times less training time to achieve similar performance on 3D shape classification on ModelNet10 dataset.

2. Related Works

2.1. PointNet

PointNet [4] is a Point-Based methods for 3D Point-Cloud classification. It is one of the first proposed strategies and several iteration of improvement has been done on the original model. The model directly consume the cloud points data without converting them to voxel grids and uses convolutional neural networks to train the model. As suggested by Charles et al. [4], the key of the PointNet network is the use of a single symmetric function, max pooling. The other two key modules include a local and global information combination structure and two joint alignment networks that align both inputs and extracted features. The symmetric function here in the PointNet takes n vectors as input and outputs a new vector that is invariant to the in-

put order. More specifically, input is sorted into a canonical order and treated as a sequence to train and CNN, then aggregated by a single symmetric function from each point. The idea of using CNN here is to train the CNN to be invariant to input order. The CNN will take the point set as a sequential signal and be trained with randomly permuted sequences. The output from the symmetric function forms a vector that represents a global signature of the input set. As shown in Figure 1, the global cloud feature vector is then fed back to the per point features by concatenating the global feature with each of the point features. Then the new per point features can be extracted to have both local and global information. And therefore, PointNet is able to predict per point quantities with both local geometry and global semantics. T-net predicts an affine transformation matrix and directly applies this transformation to the coordinates of input points. The T-net is composed of basic modules of point independent feature extraction, max pooling and fully connected layers. This idea is further extended to align feature space. Another alignment network on point features is inserted to predict a feature transformation and align features from different input point clouds.This architecture ensures the model is invariant to transformation and has a high level of robustness.

2.2. 3D ShapeNets and derivatives

Xu and Todorovic (2016) reported significant improvement of classification accuracy on ModelNet10 data by their final 3D convolutional neural network (CNN) model compared to classical 3DShapeNets model while their model had only 0.6% (80K/12M) parameters of 3DShapeNets [5].The whole searching process started from a simple CNN, and iterated through searching between 2 actions: add new convolutional filters or new convolutional layer to a parent CNN. The parameters of the parent CNN were transferred to its children to ensure the efficiency. During the search, the states were tracked through a heuristic function. The best-found model has 16 filters of size 6 and stide2 as 1st layer, 64 filters of size 5 and stride 2 as 2nd layer, a fully-connected layer with hidden units as last layer [6]. The model maintained the same structure as the classical 3D CNN while reducing the number of parameters involved in the model to be trained.

3. Motivation

There are many benefits of a simpler model. An unnecessary complex and deep model is not efficient and takes a longer time to train the model. The higher number of trainable parameters also requires a larger set of training data during training to achieve good performance. From the computational power perspective, GPU is expensive and also high consumption of electricity is not environmentally friendly. From the performance perspective, a simpler model is less prone to overfitting than a more complex model. From the practical point of view, a simpler model takes less time to train and allows us to try out more combination of higher parameters' value.

4. Data

We use the ModelNet10 data from Princeton University, a comprehensive clean collection of 3D CAD models for objects. It contains the ten most common object categories in the world (bathtub, bed, chair, desk, dresser, monitor, night stand, sofa, table, toilet). There are 156 instances of bathtub, 615 instances of bed, 989 instances of chair, 286 instances of desk, 286 instances of dresser, 565 instances of monitor, 286 instances of night stand, 780 instances of sofa, 492 instances of table, 444 instances of toilet. It is a sample of instances from a larger set that is collected by Amazon Mechanical Turk to manually decide whether each CAD model belongs to the specified categories. A label associated with each instance is assigned by humans. There should be no relationships between individual instances and the individual instance is independent of each other. The data is already split for training and testing in the original dataset. The dataset is self-contained and can be directly downloaded as a zip file from https://modelnet.cs.princeton.edu/. The dataset is not related to people and does not contain offensive or confidential or sensitive information.

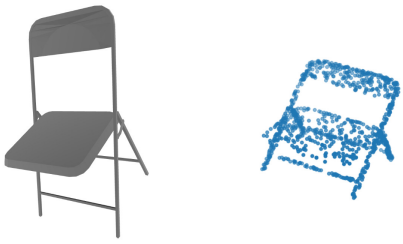

Figure 1. Example rendering of a chair's CAD and point cloud image

5. Approach

Our model is based on the PointNet structure, which has been reported to effectively perform 3D classification on point clouds input [4]. Partial reason for picking this model is that this model represented a novel deep net architecture that consumes raw point cloud without voxelization or rendering. This will allow us to focus on researching the model architecture without spending too much time on data rendering. PointNet was originally reported to test on ModelNet40, while we aimed to implement it on the ModelNet10 dataset to benchmark the 3D CNN model found by Xu and Todorovic (2016) [6].

However, PointNet consists of 3.5M parameters after being implemented by tensorflow, which implies it would consume large amounts of training data and computational resources to get robust predictions. Meanwhile, a limitation of 3D shape classification is the amount of available labelled data. Currently available 3D shape datasets are smaller by at least an order of magnitude than 2D datasets. For example, our benchmark 3D shape datasets ModelNet dataset [5] has 150K shapes, whereas the 2D shape dataset ImageNet [3] has 1.5M images. The number of parameters is especially a big concern when we consider the limited amount of computational power we have for the project. Therefore, our aim of this project is to constructed a PointNet-like network and reduce the number of parameters, while keeping its performance at a comparable level.

Our modification of the architecture is inspired by the paper "Beam Search for Learning a Deep Convolutional Neural Network of 3D Shapes" [6], where they perform beam search to find an optimal architecture with less parameters adopted from the framework of another deep 3D convolutional network classifier, 3D ShapeNets [5]. Using this as a proof of concept, we reduce the size of PointNet in a similar manner while keeping its key features, then perform fine-tuning for better performance. Although, to implement the whole reinforcement-learning-like beam search won't fit the timeframe of a short summer quarter, we decided to construct a model based on the reported framework by reducing the number of parameters in the model. Meantime, we would like to keep the main structure as well as the main components that made reported model work.

In order to maintain the basic properties of the PointNet, we kept its main structure, but reduced the dimension of 1D convolutional layer within the T-net. This was inspired by Xu and Todorovic (2016) [6], whose best-found model from beam search was a parameter-reduced version of 3D ShapeNets by Wu et al. (2015) [5]. As there are 3 convolutional layer in the T-net and there are 2 T-net structure in the PointNet for different feature extraction, thus, by reducing the dimension of convolutional layers by half, our Go3D network has only 20% of the number of parameters to be trained comparing to original PointNet model. Our hypothesis is that by maintaining the major structure of Pointnet, with fewer parameters, we can still extract necessary features for 3D classification while fewer parameters will ensure the model's robustness with the same amount of data we have. We also implemented data augmentation and L2 regularization for better robustness and generalization. Below are what we expected from the our approach:

1. Equal or better performance will be achieved.

2. Significant reduced run time.

3. More robust model.

6. Experiments and Results

6.1. Experiments

To implement the model are using the Keras module in the Tensorflow. More details can be found in our repository as https://github.com/tiancizhong/GO3D. Keras was picked for quick and simple implementation, inspired by a simple implementation of ModelNet-CNN in Keras at https://github.com/guoguo12/modelnet-cnn. We also refer to PointNet implementation from the original author's repository https://github.com/charlesq34/pointnet and the famous Princeton SimpleView repository https://github.com/princeton-vl/SimpleView. Trainable parameters in our model are for convolutional layers, batch normalization and dense layers. The remaining part of the network, including global max pooling layers, dropout layers and activation layers do not have trainable parameters. The trainable parameters were updated using Adam optimizer [2].

The model takes point clouds as input. Cloud point data are selected from the ModelNet10 data that represent 3D coordinates (x, y, z). In order to get the point cloud input, we sampled points on the mesh surface using uniform random sampling. The model outputs the label for the most likely class, and uses sparse categorical cross-entropy loss function. The number of points sampled per image is a hyperparameter determined by fine tuning. Other hyperparameters include dropout rate, batch size, number of epochs and learning rate. The hyperparameters are tuned in a grid-search manner.

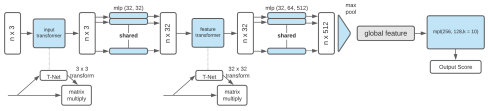

Figure 2. Architecture GO3D network

We implemented Tensorflow in a Linux environment (Ubuntu 20.04.2.0). Tensorflow used CUDA cores from NVIDIA GeForce RTX 2070 SUPER. Teamviewer was used for remotely working as a team as we only have 1 GPU. However, due to the limited graphic memory of RTX 2070 SUPER, when we tuned the batch size, we found that we can only choose the batch size of 32. By doubling the batch size to 64, we will have memory overflow. Thus, batch size was set fixed at 32.

Go3D network was implemented to compare against the original PointNet model. Hyperparameters were tuned to ensure performance comparison between optimal models. Here are the list of hyperparameters have been tuned:

- Number of points to input: 1024, 2048

- Number of epochs: range(1, 31)
- Data augment (point jittering): True, False
- Learning rate: 0.0005, 0.001, 0.005
- Dropout rate: 0.3, 0.5

For each combination of the hyperparameters, we saved the model and predication and later evaluated the results by using Tensorboard as UI from Keras CallBack APIs. The main performance metrics we looked at is the accuracy from validation dataset. The accuracy is so-called SparseCateoricalAccuracy in Keras, which labels the prediction by applying argmax of the probability from classes and then calculates the classification accuracy. We also looked at an Overfitting Index, which is defined by ourselves as:

$$Overfitting\ index = \frac{Accuracy_{Train} - Accuracy_{Validate}}{Accuracy_{Train}}$$

For each model fit, a learning curve was examined by looking at accuracy on train and validate datasets with the increasing number of epochs trained.

6.2. Results

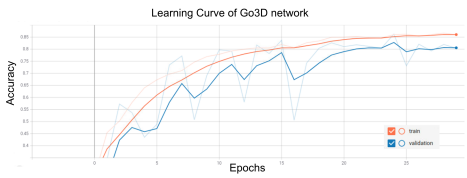

Figure 3. Learning curve of GO3D network

Our Go3D network (Figure 2) has a total of 748,979 parameters, which contains 742,899 trainable parameters, while the original PointNet model has 3,472,339 trainable parameters from a total of 3,484,499 parameters (Go3D structure link). The runtime by fitting Go3D 30 40% of the runtime on fitting original PointNet with the same hyperparameters. Although the parameters of Go3D to be trained have been reduced by 80% comparing to the PointNet, we still maintain the key components of the PointNet model: i) max pooling as a symmetry function to aggregate information from unordered input and ii) two joint alignment networks (T-net) that align input to a canonical space before feature extraction as well as alignment of feature space extracted. And therefore, Go3D network was expected to maintain the same properties as the PointNet: i) invariant to unorder input 2) global and local information aggregation. However, with fewer parameters to be trained, we expected the model to converge faster and provide at least equivalent performance. This was inspired by Xu and Todorovic (2015)'s research, whose proposed model kept a similar structure as the classical ShapeNet model while reducing significant parameters. If the reduction of parameters doesn't comprise the key structure of the network that contributes to feature extraction or information aggregation, the performance can be expected to improve or remain the same. This is due to the fact that, with the same volume of training data, the fewer parameters to be trained leads to higher degree of freedom. The model will also be more robust for the same reason, especially when the final fully connected layers have fewer parameters with L2 regularization added. We did not tune the L2 regularization parameters as we expect minimal impact on the performance.

As shown in the Table 1, both networks reached optimal at the same hyperparameters: 1024 points data clouds rendered from CAD, a batch size of 32, number of epochs at 24, data augment with points jittering, learning rate of 0.0005 and dropout rate of 0.3 for the 2 dropout layers. The learning rate has the most impact on the model performance. With a larger learning rate, the model showed hard to converge to the optimal. The accuracy can drop with more epochs trained. This indicated the model fitting was wandering around the optimal, however, the step might be larger so that it passed the optimal point each time. Both number of points and dropout rate has little impact on the model performance. The data augment with points jittering helped improve the model robustness. With the same hyperparameters, the model with the data augmentation will have better performance on validate as well as a smaller overfitting index. This can be explained by the fact that the data augmentation makes point cloud data more continuous and the jittering process help make the data more invariant to transformation.

The Go3D model showed accuracy of 86.23% on validate dataset, as shown in Table 1. Original PointNet model showed almost the same performance, 86.56% on validate dataset. However, considering the significant reduction of parameters, we still consider Go3D as a success. The parameters have been reduced by 80% compared to the PointNet while providing the same performance. The training time of a single model has also been reduced significantly given the same hyperparameters. Figure 3 showed the learning curve of the optimal Go3D. It clearly illustrated the optimum had been reached at the 24 epochs.

Table 2 and Figure 4 demonstrated the prediction of Go3D on the validate datasets. The confusion matrix (Table 2) showed the challenge of classification of chair object, as most misclassification is either mis-labelling the chair as other objects or mis-labeling other objects as chair. Figure 4 gave some examples of predictions from the validate dataset, one of them showed mis-classification of dresser as chair. Those misclassifications could be due to the fact that Go3D is hard to capture the invariance of the chair, espe-

Model	Accuracy		Optimal Hyperparameters					
	Train	Validate	Num Points	Batch Size	Num Epochs	Augmentation	Learning Rate	Dropout Rate
PointNet	86.32%	86.56%	1024	32	24	Ture	0.0005	0.3
Go3D	86.02%	86.23%	1024	32	24	Ture	0.0005	0.3

Table 1. Performance of implemented PointNet and Go3D networks on Train and Validate dataset with their optimal hyperparameters.

	Bed	Monitor	Desk	Chair	Dresser	Toilet	Sofa	Table	Night Stand	Bathtub
Bed	87	0	2	1	2	2	1	1	2	2
Monitor	0	42	1	0	0	1	2	2	2	0
Desk	0	0	96	0	0	1	2	0	0	1
Chair	1	0	0	52	13	10	1	4	1	4
Dresser	1	0	2	0	65	0	2	0	0	16
Toilet	0	0	2	10	0	84	0	0	0	4
Sofa	0	0	2	0	1	3	94	0	0	0
Table	0	0	1	0	3	1	0	95	0	0
Night Stand	0	5	5	0	1	3	0	1	85	0
Bathtub	0	0	0	0	6	0	1	0	0	79

Table 2. Confusion Matrix of Go3D prediction on Validate datasets

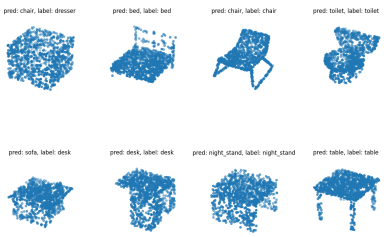

Figure 4. Example prediction result from Go3D network

cially when the chair can be very different from type to type. This means that Go3D was not generalized enough to capture all kinds of chairs. Another possible reason could be that although Go3D extracted more local information than global information. One possible solution is the segmentation network, which feeds calculated global point cloud feature vector back to per point features by concatenating the global feature with each per point features. Then the new per point feature can be extracted to have local geometry and global semantics. This could be our future work to be researched.

7. Conclusion

In this project, we explored 3D classification networks with similar feature to PointNets but require less learned parameters. 3D object classification is very challenging due to its unordered nature and invariance under transformation. PointNet is a proven network that is able to capture invariance from unsorted sequential data by its symmetrical function as well as its joint alignment network. However, it contains 3.5M parameters. Go3D is able to provide the same performance as PointNet on ModelNet10 while reducing the trainable parameters by 80%. This significantly saves the runtime and increases the degree of freedom for model fitting.

8. Work Division

Detailed contributions among team members can be found in Table 3. As mentioned in the previous sections, the implementation was done by the team using Teamviewer remote desktop function for remotely working on one machine with GPU together. Although each team member led one aspect of the work, majority of work was achieved by the whole team.

Student Name	Contributed Aspects	Details
Hengye Jing	Implementation and Analysis	Implemented network construction module in Tensorflow. Replicated PointNet model. Constructed Go3D.
Tianci Zhong	Structure Design and Data Rendering	Designed code base structure. Implemented data rendering from ModelNet10 datasets. Implemented data augmentation.
Yichen Wang	Tools Creation and Performance Evaluation	Implemented the tensorboard for performance review. Created necessary tools for results output.

Table 3. Contributions of team members.

References

[1] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *CoRR*, abs/1912.12033, 2019. 1

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 3

[4] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016. 1, 2

[5] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 1, 2, 3

[6] Xu Xu and Sinisa Todorovic. Beam search for learning a deep convolutional neural network of 3d shapes. *CoRR*, abs/1612.04774, 2016. 2, 3