Point Cloud Object Detection

Project Report

Foundations of Machine Learning (CS 725)
Computer Science and Engineering

Submitted by:

Santhosh 23d0369

Shivansh 22m2120

Sreekant 23M0794

Siva 23M0747

Nadesh 21q050003

Helen Sathvika Bonam 22M2115

Under the Guidance of
Prof. Sunita Sarawagi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

2023-2024

Contents

Chapter 1

Introduction

World Health Organization has estimated an average of 1.3 million road deaths every year. Road marking extraction is emerging as an important remote sensing application to meet United Nations goal to reduce road injuries by half by 2030. One of the ways to reduce Highway injuries is to ensure Road marking, Lighting Poles and GuardRails are in place and in proper condition. The process of analysis of these highway objects is a manual, tedious and time-consuming process. With advancements in data capture technologies, cost of 3D point cloud data capture have drastically reduced over the last decade. Today point cloud and photogrammetry data capture methods are widely used and processed to identify road assets and its conditions to monitor and maintain road safety.

Machine learning algorithms for point cloud have evolved since 2017. PointNet was the first algorithm to use 3D based convolution in 2019. Since then, different methods of 3D based neural network algorithms have evolved. Different methods of convolution for 3D point cloud include voxelization, graph network methods and point cloud transformers each with its own advantages and disadvantages.

In this project, we would look at exploring three different neural network models (Convolution based, Graph Based and Transformer based) for classification of Fence, Road Marking and Poles that are described in chapter 2. We would compare the performance of this model using Triplet Loss and see if the model performance is better (given limited labeled datasets). Chapter 3 has a brief description of the dataset we are using and in Section 4, we describe the proposed solution. Section

5 has the list of activities done till now and roadmap for the remainder of the semester. Finally, Section 6 has preliminary results.

Chapter 2

Related Work

In this chapter we are presenting a brief overview of the three deep learning neural network models PointNet, DGCNN and SPT we are using in our project.

2.1 PointNet

The PointNet architecture is designed to directly consume point cloud data, which are sets of points in 3D space. It can perform various 3D recognition tasks, such as object classification, part segmentation, and scene semantic parsing. The architecture has the following properties:

- It is simple and efficient, using only fully connected layers and max pooling to process point sets.
- It is invariant to the order of input points, using a symmetric function to aggregate information from all points.
- It is robust to input perturbation and corruption, using spatial and feature transformations to align the data.
- It can learn both global and local features of point sets, using a combination of global shape signature and per-point features.

The PointNet architecture consists of two networks: a classification network and a segmentation network. The classification network takes n points as input, ap-

plies input and feature transformations, and then aggregates point features by max pooling. The output is classification score for m classes. The segmentation network is an extension to the classification net. Each of the n inputs needs to assign one of the m segmentation classes. Because segmentation relies on local and global features, the points in the 64-dimensional space are concatenated with the global feature space, resulting in possible feature space of $n * R^{108}$. The PointNet architecture has these key modules: the max-pooling layer, a local and global combination structure, and two joint alignment networks that align both local and global networks

2.2 Dynamic Graph CNN (DGCNN)

Dynamic Graph Convolutional Neural Network, a deep learning architecture that reads graphs directly and learns a classification function. The main building block of DGCNN, which operates on graphs dynamically computed in each layer of the network1. It generates edge features that describe the relationships between a point and its neighbors, and aggregates them using a symmetric function2. EdgeConv has several appealing properties, such as permutation invariance, partial translation invariance, and non-local diffusion of information throughout the point cloud3. DGCNN is used for point-cloud-related high-level tasks including category classification, semantic segmentation, and part segmentation. It achieves state-of-the-art performance on several benchmark datasets.

2.3 SuperPoint Transformer (SPT)

The superpoint transformer architecture used in this paper is a novel method for semantic segmentation of large point clouds1. It has the following main components:

- Hierarchical superpoint partition: It segments the input point cloud into a multi-scale structure of geometrically homogeneous superpoints, which reduces the input size and adapts to the local complexity of the data2.

- Superpoint transformer: It uses a U-Net-like network that operates on the superpoints at different scales, using self-attention to capture the relationships between superpoints within and across levels3. It also uses handcrafted features and adjacency encoding to characterize the superpoints and their interactions.
- Hierarchical supervision and augmentation: It leverages the hierarchical partition structure to supervise the model with both label frequency and distribution, and to augment the data by randomly dropping superpoints at different levels.

2.4 Triplet Loss

Triplet loss is a way to teach machine-learning algorithms to recognize not only the similarity, but also differences between items. The objective of triplet loss is to reduce the loss between similar items and increase the gap between different items. The mathematical depiction is shown below: Where

$$\sum_{i}^{N}[\left|\left|f(x_i^a) - f(x_i^p)\right|\right|_2^2 - \left|\left|f(x_i^a) - f(x_i^n)\right|\right|_2^2 + \propto]$$

- f(x) accepts an input of x and generates a 128-dimensional vector w
- I represents the i'th input
- The subscript a denotes an anchor image, p is a positive image, and n is a negative image refers to the bias.

The goal is to minimize the above equation by minimizing the first term and maximizing the second term, and bias acts as a threshold.

Chapter 3

Datasets

A Point in space is generally characterized by three parameters, the X, Y, and Z coordinates. A large collection of these points is termed Point Cloud. These are particularly useful for object classification and detection in 3D, which is then utilized for autonomous driving systems. This data is usually captured using 3D scanners or LiDAR technology (Light Detection And Ranging). In practice, each data point consists of various parameters as well apart from coordinates.

- XYZ Coordinates
- RGB color values
- Intensity
- GPS time
- Scan angle details

ModelNet40 dataset contains synthetic object point clouds. The ModelNet40 consists of 12,311 CAD-generated meshes in 40 categories (such as airplane, car, plant, lamp). The corresponding point cloud data points are uniformly sampled from the mesh surfaces, and then further preprocessed by moving to the origin and scaling into a unit sphere.

Toronto-3D is a large-scale urban outdoor point cloud dataset acquired in Toronto,Canada. It covers approximately 1km of road and consists of about 78.3 million points. This

dataset includes 9 attributes in which we are mainly focusing on the Fence, Road marking, Pole attributes. In this project, we will be focusing on this dataset.

Chapter 4

Implementation Details

4.1 Toronto 3D Integration

Toronto 3D dataset is in ply format and it can't be used directly for training the already existing models. These models takes datasets in numpy format. So, Toronoto 3D dataset format is converted into numpy format by using numpy library. The Toronto 3D dataset is read in ply format using read function in plyData library and converted into numpy array.

4.2 SuperPoint Transformer (SPT) Model

The existing SuperPoint Transformer is using cross entropy loss function. The test accuracy and train accuracy for training test is 87.778% and 76.976% . The validation test accuracy is 77.136%. We used Siamese Networks to get better classification. We use two identical networks with the same parameters and weights to compute the similarity and dissimilarity between inputs. Since the Siamese neural network uses pairwise learning, we cannot use cross entropy loss. SNNs involve pairwise learning, we cannot use cross entropy loss, hence have used triplet loss function. Included intensity of data feature for SPT

4.2.1 Triplet Loss Implementation

```
import torch.nn.functional as F
def batch_all_triplet_loss(anchor, positive, negative, margin=0.2):
"""
Compute triplet loss using the batch all strategy.
"""
distance_matrix = compute_distance_matrix(anchor, positive, negative)
loss = torch.max(torch.tensor(0.0), distance_matrix[:, 0] - distance_matrix[:, 1] +
margin)
loss += torch.max(torch.tensor(0.0), distance_matrix[:, 0] - distance_matrix[:, 2]
+ margin)
return torch.mean(loss)
```

The batch_all_triplet_loss function computes the triplet loss using the batch-all strategy for a given set of anchor, positive, and negative samples and calculates the triplet loss based on the computed distance matrix, enforcing a margin of 0.2 between the positive and negative distances. The function returns the mean of the computed loss across all samples in the batch.

4.3 Pointnet Model

The existing Pointnet model is trained for object classification on ModelNet40 dataset. The test accuracy and train accuracy of training test is 0.831 and 0.842. The test accuracy and train accuracy of validation test is 0.008 and 0.025. We implemented pointnet model on Toronto 3D dataset. Toronto 3D dataset can't be used directly for training the Pointnet model. So, we integrated Toronto 3D into model by changing the format. In Toronto 3D dataset, each point need to be classified. So, we reimplemented pointnet model for semantic segmentation. Semantic segmentation recognize a collection pixels into distinct categories such as traffic signs, poles, pavements etc.. Series of MLP are done on input features

and output scores of each pixel is obtained.

4.4 DGCNN Model

The existing DGCNN model is trained for object classification on ModelNet40 dataset. The test accuracy and train accuracy of training test is 0.998 and 0.888. The test accuracy and train accuracy of validation test is 0.008 and 0.025. We implemented DGCNN model on Toronto 3D dataset. Toronto 3D dataset can't be used directly for training the DGCNN model. So, we integrated Toronto 3D into model by changing the format. In Toronto 3D dataset, each point need to be classified. So, we reimplemented DGCNN model for semantic segmentation. Semantic segmentation recognize a collection pixels into distinct categories such as traffic signs, poles, pavements etc.. Series of MLP are done on input features and output scores of each pixel is obtained.

4.5 Results

Model Name	Train Accuracy	Test Accuracy	Validation Accuracy
SPT	49.492%	59.77%	59.475%
Pointnet	50%	0%	0%
DGCNN	50%	0%	0%

4.6 Challenges Faced

- Three of the four files in Toronto-3D files are greater than 1GB. Hence gives CUDA out-of-memory error (on a 16GB GPU machine). We have disabled use of GPU and running on CPU mode.
- SuperPoint Transformers uses FRNN module which uses a very old version of CUDA library. Downgrading the CUDA library is a challenge as the server is a shared server. We are looking at CPU only FRNN library
- Faced a lot of issues in early days related to library version mismatch. We explored various versions and now have finalized on a "conda" environment

that is working.

Chapter 5

Conclusion

We have done

- Integrate Toronto-3D dataset with DGCNN
- Integrate Toronto-3D dataset with PointNet
- Completed the modification to include Triplet loss instead of regular cross-entropy loss in the algorithm.
- Included intensity of data feature for SPT
- Toronto 3D dataset preprocessing and transformation to use with the Super-Point Transformer model.
- Implemted semantic segmentation for pointnet and DGCNN
- Fine tune model parameters to improve accuracies.

References

[1] "Implementation Repository Link"

[2] Poinet Slides

[3] Poinet Paper

[4] Pointnet Github Code

[5] DGCNN Paper

[6] DGCNN Github Code

[7] SPT Paper

[8] SPT Github Code

[9] Toronto 3D