**Point Cloud Object Detection**

(Project Report)

Foundations of Machine Learning (CS 725)

Computer Science and Engineering

By

Santhosh Shivansh Helen Sreekant Siva Nadesh

23d0369 22m2120 22m2115 23M0794 23M0747 21q050003

Guided By

Prof. Sunita Sarawagi



Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

October 31, 2023

# Table of Contents

[**Table of Contents 2**](#_9j8mtcs3m21f)

[**1. Introduction 3**](#_lx6xtdliep7c)

[**2. Related Work 4**](#_ukap38yd6t10)

[2.1. PointNet 4](#_o6caujtjlhl8)

[2.2. Dynamic Graph CNN (DGCNN) 4](#_trgm12sdgz4b)

[2.3. SuperPoint Transformer (SPT) 5](#_wydr91oyuudq)

[2.4. Triplet Loss 5](#_mdqldtxfcad8)

[**3. Datasets 6**](#_mjmp3vabctrg)

[**4. Proposed Solution 8**](#_w050yburppnw)

[**5. Implementation Details 9**](#_l3x8wc5kwacp)

[5.1 Activities completed till date 9](#_olv7hejlczpu)

[5.2 Activities post Midterm report are 9](#_dp3vksjqugry)

[5.3 Challenges Faced 9](#_702cqqdppx22)

[5.4 Triplet Loss Implementation: 9](#_jvm9kxu465m0)

[**6. Preliminary results 11**](#_rnu98g20fb10)

[6.1 DGCN Results: 11](#_b16ixcr4fd6n)

[6.1.1 Training Results: 11](#_9lj07mne9d4n)

[6.1.2 Validation Results: 11](#_jux7ud3d7oa1)

[6.2 PointNet results 11](#_8u6w0ku41vs8)

[6.2.1 Training Results 11](#_huzo1v7izjid)

[6.2.2 Validation Results 11](#_pkw2pzr0xb30)

[**7. References 12**](#_72a9ajm16gkn)

[**Appendix: 13**](#_tr3hxid1mrzn)

# 

# 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 Section 2. We would compare the performance of this model using Triplet Loss and see if the model performance is better (given limited labeled datasets). Section 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.

# 

# 2. Related Work

In this section 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](https://arxiv.org/pdf/1612.00593.pdf)

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, applies 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 \* ℝ¹⁰⁸. 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](https://arxiv.org/pdf/1801.07829.pdf) (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.

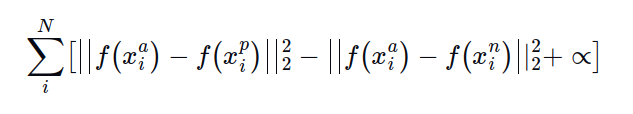
## 2.3. [SuperPoint Transformer](https://arxiv.org/pdf/2306.08045.pdf) (SPT)

The superpoint transformer architecture used in [this](https://arxiv.org/pdf/2306.08045.pdf) 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

* 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.

# 

# 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

There are multiple widely-used point cloud datasets available. Some are them are:

* Semantic KITTI - <http://www.semantic-kitti.org/dataset.html>
* Paris-Lille-3D - <https://npm3d.fr/paris-lille-3d>
* Toronto-3D - <https://github.com/WeikaiTan/Toronto-3D>

Here is a rough number of classes in each dataset related to each broad category.

|  | **Semantic KITTI** | **Paris-Lille-3D** | **Toronto-3D** |
| --- | --- | --- | --- |
| Road/Ground | 4 | 3 | 1 |
| Buildings/Structures | 2 | 2 | 1 |
| Vehicles | 5 | 15 | 1 |
| Vegetation | 2 | 3 | 1 |
| People/Bicyclists | 3 | 1 | 0 |
| Road/Traffic Elements | 5 | 9 | 3 |
| Furniture/Objects | 1 | 15 | 1 |

SemanticKITTI extends KITTI Vision by adding semantic annotations to existing dataset sequences, providing numerous scans for a full 360-degree view with LiDAR. Labeled at 10 Hz, it uses temporal data for semantic scene understanding, featuring classes for moving and stationary traffic participants (e.g., cars, trucks, pedestrians).

Paris-Lille-3D is generated using a Mobile Laser System (MLS) in Paris and Lille, France. It's hand-labeled with 50 diverse classes, used for developing automated point cloud segmentation and classification algorithms.

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.

# 

# 4. Proposed Solution

Given the challenges in obtaining labeled data for 3D point cloud datasets, we propose to use 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 as explained in section 2.4

# 

# 5. Implementation Details

### 5.1 Activities completed till date

1. Setup of PointNet algorithm and execution on standard (Kitti) dataset.
2. Setup of DGCN (Dynamic Graph Convolution Network) algorithm and execution on standard (Kitti) dataset.
3. Setup of SuperPoint Transformer model.
4. Toronto 3D dataset preprocessing and transformation to use with the SuperPoint Transformer model.
5. Completed preprocessing of Toronto 3D LO04.ply file. Waiting for the other three file transformation to complete before fine tuning the parameters.
6. Completed the modification to include Triplet loss instead of regular cross-entropy loss in the algorithm.

The detailed implementation changes can be found at

<https://github.com/shivansh1010/FML-project/tree/main>

### 5.2 Activities post Midterm report are

1. Integrate Toronto-3D dataset with DGCN
2. Integrate Toronto-3D dataset with PointNet
3. Fine tune model parameters to improve accuracies.
4. Present the results

### 5.3 Challenges Faced

1. 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.
2. 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
3. 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.

### 5.4 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)*

# 

# 6. Preliminary results

## 6.1 DGCN Results:

### 6.1.1 Training Results:

Train 249, loss: 1.286651, train acc: 0.998779, train avg acc: 0.997489

Test 249, loss: 1.442699, test acc: 0.918963, test avg acc: 0.888419

### 6.1.2 Validation Results:

Namespace(batch\_size=32, dataset='modelnet40', dropout=0.3, emb\_dims=1024, epochs=250, eval=True, exp\_name='dgcnn\_1024\_eval', k=18, lr=0.01, model='dgcnn', model\_path='checkpoints/dgcnn\_1024/models/model.t7', momentum=0.1, no\_cuda=False, num\_points=1024, seed=1, test\_batch\_size=16, use\_sgd=True)

Using GPU : 0 from 1 devices

**Test :: test acc: 0.008104, test avg acc: 0.025000**

## 6.2 PointNet results

### 6.2.1 Training Results

Train 249, loss: 1.582438, train acc: 0.890472, train avg acc: 0.831194

Test 249, loss: 1.527525, test acc: 0.894652, test avg acc: 0.842297

### 6.2.2 Validation Results

Namespace(batch\_size=32, dataset='modelnet40', dropout=0.5, emb\_dims=1024, epochs=250, eval=True, exp\_name='pointnet\_1024\_eval', k=18, lr=0.001, model='pointnet', model\_path='checkpoints/pointnet\_1024/models/model.t7', momentum=0.9, no\_cuda=False, num\_points=1024, seed=1, test\_batch\_size=16, use\_sgd=True)

Using GPU : 0 from 1 devices

**Test :: test acc: 0.008104, test avg acc: 0.025000**

# 

# 7. References

1. Implementation Repository Link - <https://github.com/shivansh1010/FML-project>
2. Pointnet Slides - <http://stanford.edu/~rqi/pointnet/>
3. Pointnet Paper - <https://arxiv.org/pdf/1612.00593.pdf>
4. Pointnet Github Code - <https://github.com/charlesq34/pointnet>
5. DGCNN Paper - <https://arxiv.org/pdf/1801.07829.pdf>
6. DGCNN Github Code - <https://github.com/WangYueFt/dgcnn>
7. Superpoint Transformer Paper - <https://arxiv.org/abs/2306.08045>
8. Superpoint Transformer Github Code -

<https://github.com/drprojects/superpoint_transformer/tree/master>

1. Datasets-
   1. Semantic KITTI - <http://www.semantic-kitti.org/dataset.html>
   2. Paris-Lille-3D - <https://npm3d.fr/paris-lille-3d>
   3. Toronto-3D - <https://github.com/WeikaiTan/Toronto-3D>

# 

# 

# Appendix:

**1. Feedback Comments**

Your report must include details on the following:

1. An enhanced and well-defined problem statement, taking into account the received feedback. - Section 1, 5
2. Description of the proposed solution approach - Section 4
3. A code survey where you include links to the relevant codebases that you refer to while implementing the solution
   1. Section 2 and links are provided in the reference section
4. Datasets - Section 3
5. Implementation details - Section 5
6. Preliminary results - Section 6
7. A roadmap for the remainder of the semester - Section 5.2

**2. Abstract Feedback - Response**

The proposed task looks good. feasible to finish within time. There are several things you need to take care of:

1. The performance of the proposed models is already reported (as per the dataset details). The project should not just be reproducing those results. You need to introduce originality and should try new approaches to improve upon the mentioned models. Please be more precise about the method (how exactly are you going to tackle the mentioned task).

* The project is not proposing reproducing the results. We have identified these classes Fence, Road Marking and Poles that have low accuracy and propose to apply the concepts learned from class to improve these accuracies.

2. It can't be straightaway judged but please be mindful of the compute power required.

* Yes, we have access to server with Nvidia RTX A4000 GPU with 16GB memory and Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz with 60GB RAM

3. The project should incorporate topics from class also.

* Topics from the class are Kernel Methods, Siamese networks, Triplet Loss, CNN Model and Transformers