International Institute of Information Technology Bangalore

Project Elective Report

AI-825

Semantic Segmentation of 3D point cloud

Adithya Nagaraja Somasalle

January 2024- April 2024

ज्ञानमुत्तमम्

Contents

1 Objective **2**

2 Dataset **2**

3 Segmentation Visualization Using Cloud Compare **2**

4 Feature Extraction **2**

5 Training Classifier Cloud Compare's Canupo plugin **2**

6 Training Classifier with machine learning **2**

6.1 $train - test\ split$ . . . 2

6.2 Adding Features of neighbourhood radius $r = 1$ . . . 3

6.3 Adding Features of neighbourhood radius $r = 1$ and $r = 0.125$ . . . 3

6.3.1 Set max threshold for number of points for each class . . . 3

6.4 Multi-Scale neighbourhood extraction . . . 3

6.5 Using XGBoost on miniTrain . . . 4

6.6 Using ensemble of random forest models . . . 4

1 Objective

To develop a machine learning model for semantic segmentation of 3D point cloud.

2 Dataset

The dataset used for training the model is Hessigheim 3D (H3D) dataset. H3D dataset contains train folder and test folder for Mar2018 and Mar2019. Each folder contains .laz files of point clouds over a village in Germany.

It contains 11 classes namely Low Vegetation, Impervious Surface, Vehicle, Urban Furniture, Roof, Facade, Shrub, Tree, Soil, Vertical Surface and Chimney.

3 Segmentation Visualization Using Cloud Compare

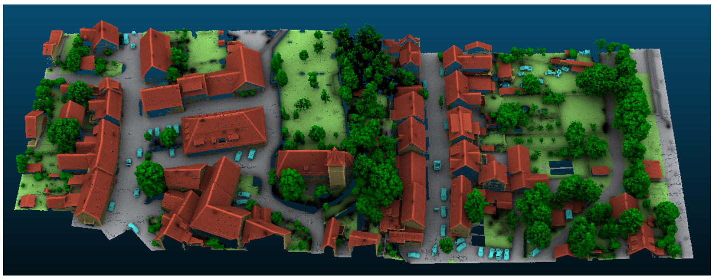

Figure 1: Segmentation on portion of train dataset
,

Here I have used the classification values to seperate each of the classes and manually gave custom colours to each of this classes and merge all the customly coloured classes.

4 Feature Extraction

Cloud Compare offers an option to calculate several features of a point cloud like PCA, Eigen value, linearity etc. Feature extraction is performed on train data and stored in txt format for later use.

5 Training Classifier Cloud Compare's Canupo plugin

Cloud Compare offers Canupo plugin which can train classifier but the drawback is that is only a binary classifier. It works great for predicting classes like trees. For training multiple classes we will have to run the classifier nearly $logn$ number of times where n is the number of classes. Clearly this approach is cumbersome.

6 Training Classifier with machine learning

6.1 $train - test\ split$

For training a random forest we first need to convert into .txt format so that we can read this data in python and pass it to random forest model. Mar2019 data contains 119,029,358 number of points, training random forest on these many points results in my computer and kaggle to crash, so I have considered only a part of this data for train and test. Let the name of our train data be $miniTrain$ as we are only choosing a part of the complete train data. All the lines containing NaN values were removed before passing into RandomForest.

6.2 Adding Features of neighbourhood radius $r = 1$

The following features were considered for training the random forest

```
feature_order = [
    'X', 'Y', 'Z', 'R', 'G', 'B', 'Sphericity_(0.1)', '1st_eigenvalue_(0.1)',
    'Verticality_(0.1)', 'Number_of_neighbors_(r=0.1)', 'Planarity_(0.1)',
    'Linearity_(0.1)', 'PCA1_(0.1)'
]
```

Metrics	**Results**
Accuracy	0.57471
F1-score	0.54322

6.3 Adding Features of neighbourhood radius $r = 1$ and $r = 0.125$

6.3.1 Set max threshold for number of points for each class

- Allot only 200000 points per class, if any class has number of points lesser than 200000 retain it.

```
[26]: accuracy = accuracy_score(labels, y_pred)
      print("Accuracy:", accuracy)

      Accuracy: 0.5150105174661561
```

- Allot only 200000 points per class, if any class has number of points lesser than 200000 than duplicates data belonging to that class randomly until the number of points belonging to that class is 200000

```
[40]: accuracy = accuracy_score(labels, y_pred)
      print("Accuracy:", accuracy)

      Accuracy: 0.3622475
```

- Another attempt was to just set the threshold as the class having maximum number of points.

```
[13]: accuracy = accuracy_score(labels, y_pred)
      print("Accuracy:", accuracy)

      Accuracy: 0.34356716560814476
```

6.4 Multi-Scale neighbourhood extraction

The following features were considered

```
feature_order = [
    'X', 'Y', 'Z', 'R', 'G', 'B',  'Planarity_(0.1)', 'Linearity_(0.1)', 'PCA1_(0.1)', 'Sphericity_(0.1)', 'Verticality_
    'Planarity_(0.125)', 'Linearity_(0.125)', 'PCA1_(0.125)', 'Sphericity_(0.125)', 'Verticality_(0.125)', '1st_eigenval
    'Planarity_(0.25)', 'Linearity_(0.25)', 'PCA1_(0.25)', 'Sphericity_(0.25)', 'Verticality_(0.25)', '1st_eigenvalue_(0
    'Planarity_(0.5)', 'Linearity_(0.5)', 'PCA1_(0.5)', 'Sphericity_(0.5)', 'Verticality_(0.5)', '1st_eigenvalue_(0.5)',
    'Planarity_(1)', 'Linearity_(1)', 'PCA1_(1)', 'Sphericity_(1)', 'Verticality_(1)', '1st_eigenvalue_(1)'
]
```

Results:

Metrics	**Results**
Accuracy	0.705809
F1-score	0.680613

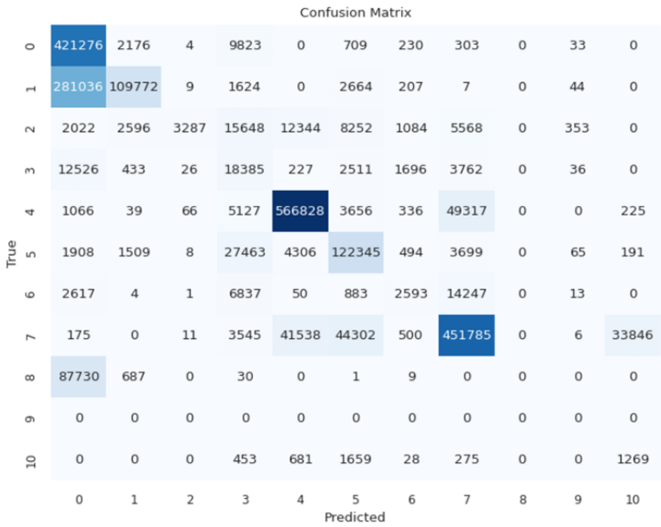

Figure 2: Confusion Matrix

Extending the features for $r = 2$ and $r = 3$

Results:

Metrics	**Results**
Accuracy	0.7223
F1-score	0.7056

6.5 Using XGBoost on miniTrain

Instead of training *miniTrain* with RandomForest, I tried training it with XGBoost with *early stop rounds* **Results:**

Metrics	**Results**
Accuracy	0.72719
F1-score	0.69322

6.6 Using ensemble of random forest models

Since we can't pass the entire data of original train (kaggle crashes), I split the entire train data including features (27 GB) into 5 files and train random forest on each of these data seperately. At the end for predicting classes on *test data* I take the majority vote among them.

5 models are generated namely model1,model2, model3, model4 and model5.

Results:

Since the model1 had bad accuracy it wasn't involved in majority vote classifier.

Table 1: Model Evaluation

Model	Accuracy	F1 Score
model1	0.1724	0.0916
model2	0.6676	0.6683
model3	0.4681	0.4719
model4	0.7345	0.7080
model5	0.6471	0.6245
ensemble	0.7395	0.7292

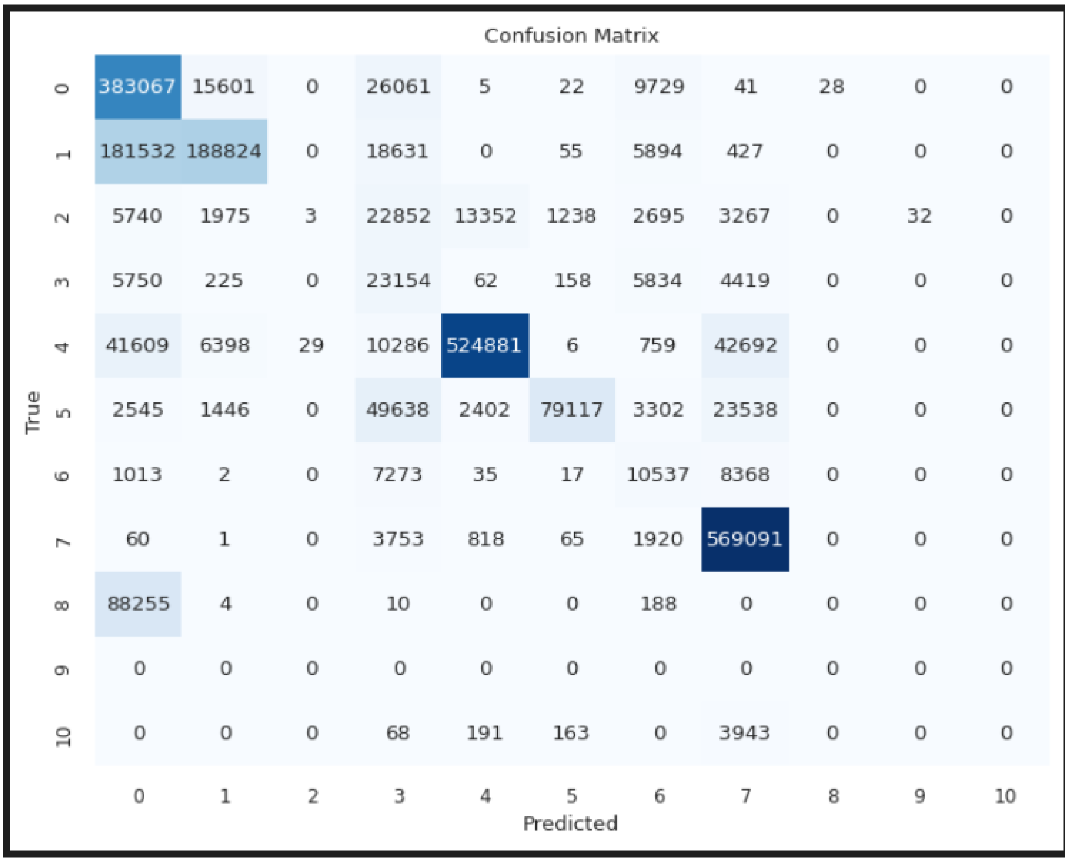

Figure 3: Confusion matrix of ensemble model

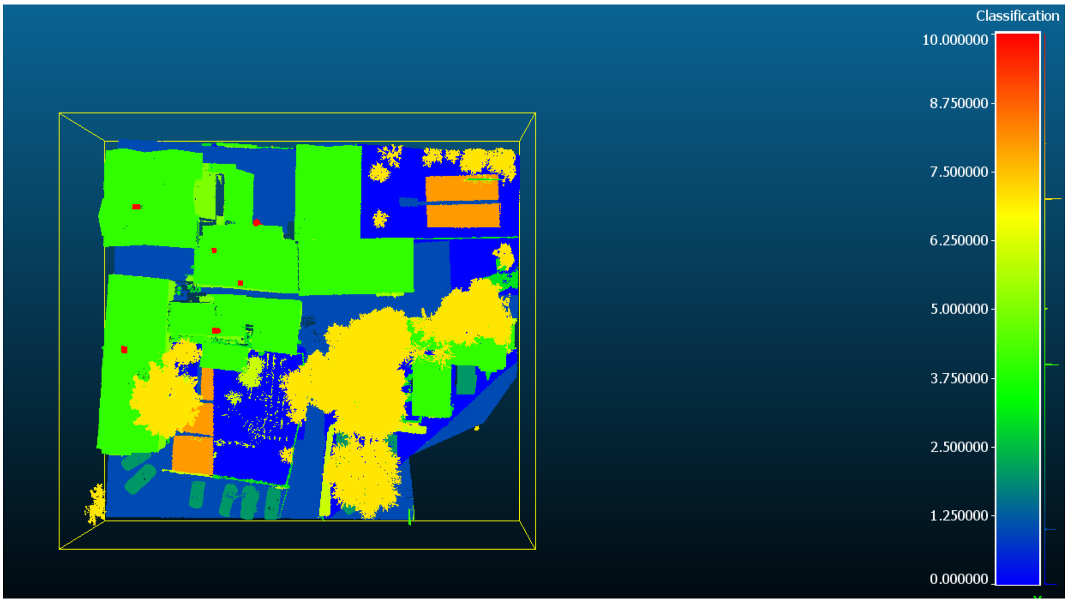

Figure 4: Original Test Data

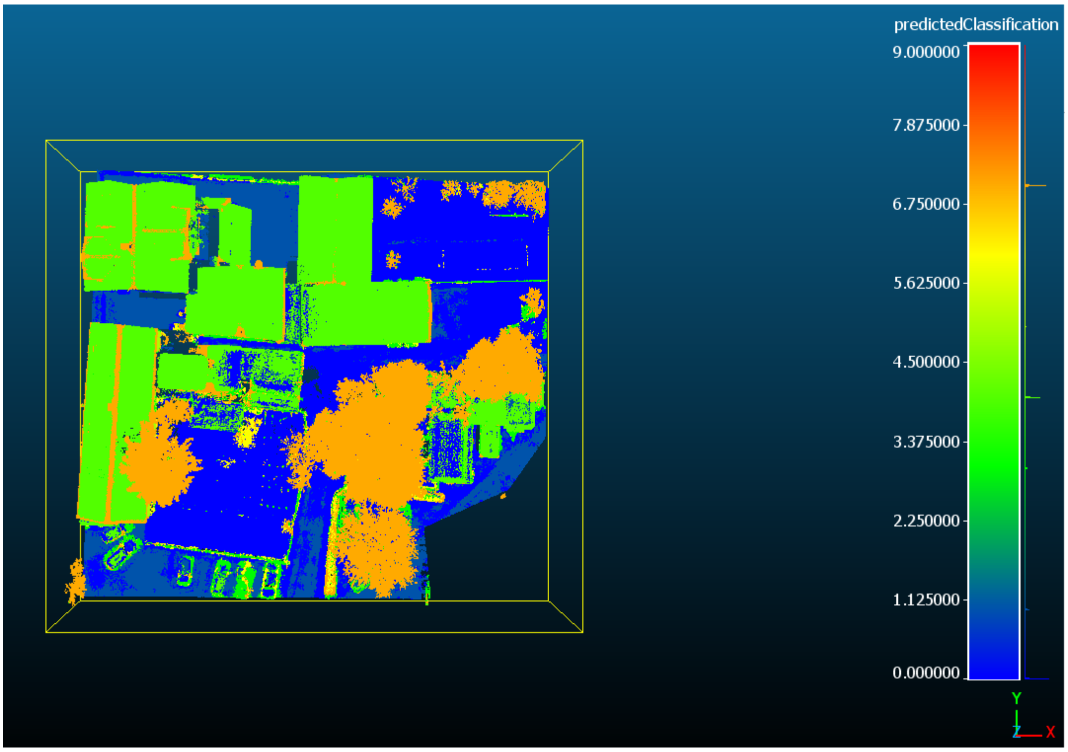

Figure 5: Ensemble model predictions

Overall the ensemble method gave the best accuracy and F1 score among all. The major drawback of this model is that it failes to identify classes 9,10 and 11.