3D Object Classification from Partial Point Clouds

Arun Madhusudhanan*, Tejaswini Dilip Deore*

***Abstract*— Point clouds are widely used as geometric data in various deep learning tasks like object detection and segmentation. However, in real-world scenarios, partial point clouds are often encountered due to limitations in sensors, occlusions, and other factors. The classification of objects from partial point clouds is a difficult task because of the sparsity, noise, and lack of complete representation of objects. This project aims to create a 3D object classification system that can classify objects from partial point clouds. To overcome the challenges, the GRNet neural network architecture is used to predict the missing data and complete the partial point clouds, which are then processed by PointNet, a deep learning framework that directly handles raw point clouds for object classification. PointNet++ is used as a baseline for comparison, as it is an extension of PointNet that is specifically designed to handle varying-density point clouds and has demonstrated superior performance in object classification tasks. The proposed method in this project performs equally or better than PointNet++.**

I. Introduction

Point clouds are sets of points in three-dimensional space that represent the surfaces of objects or environments. They can be thought of as a 3D equivalent of a digital image, where each point in the point cloud represents a pixel in the image. Point clouds provide a detailed and accurate representation of the shape and structure of objects in three-dimensional space and hence can be used for 3D representation of data. Point clouds are simple structures and easy to learn from, unlike meshes which are irregular and complex. Point clouds can be easily generated from 3D scanning devices such as Lidar, which makes them a popular choice for applications such as autonomous vehicles, robotics, and augmented reality. Additionally, point clouds can be efficiently processed by deep learning algorithms to perform various tasks such as object detection, segmentation, and classification.

Point clouds are often used in 3D object classification, where the goal is to classify objects based on their 3D shape using point clouds as input. PointNet and PointNet++ are popular deep learning methods that are used for processing raw point clouds in 3D object classification tasks.

Point clouds can be affected by occlusions or missing data, which can make it difficult to reconstruct the complete geometry of an object or scene. Partial point clouds are a common occurrence in real-world scenarios due to occlusions, sensor limitations, and other factors. The main limitation of partial point clouds is the sparsity and noise in the data, which can lead to incomplete or inaccurate representations of objects. This makes it challenging to classify objects from partial point clouds, as the absence of complete information can result in misclassification or incorrect identification.

The partial point cloud problem can be solved through partial point cloud completion. Partial point cloud completion is the process of predicting the missing data in a partial point cloud to obtain a more complete and accurate representation of the object. This can be achieved using various techniques such as deep learning-based methods, which learn to predict the missing data based on the available data.

Several techniques have been developed to reconstruct a complete 3D point cloud from an incomplete one, such as PCN [1], GRNet [2], and TopNet [3]. However, methods like PCN and TopNet directly use Multi-layer Perceptrons (MLPs) to process point clouds, which may not capture all the structural details and context of the point cloud. In contrast, GRNet uses 3D grids as an intermediate representation to regularize the unordered point clouds. GRNet comprises two differentiable layers, Gridding and Gridding Reverse, which enable the conversion of point clouds to 3D grids and vice versa without losing structural information. GRNet also consists of a differentiable Cubic Feature Sampling layer, which extracts contextual features from neighboring points. Additionally, it has a new loss function called Gridding Loss, which measures the L1 distance between the predicted and ground truth 3D grids of the point clouds to recover fine details.

In this project, our aim is to classify objects from partial point clouds. We used GRNet for partial point cloud completion. These completed point clouds are then given to PointNet framework for classification. The partial point clouds required for the task are generated from ShapeNet [4] and ModelNet10 [4] datasets. Figure1 shows the proposed system architecture.

We compared the classification performance of our proposed method with PointNet++, which is considered as a strong baseline method. PointNet++ is built on PointNet by introducing hierarchical feature learning, where the input point set is recursively partitioned into a nested hierarchy of local regions. This allows PointNet++ to capture local structures in the point cloud with increasing contextual scales, which improves its ability to recognize fine patterns and generalize to complex scenes.

Additionally, we compared the performance of PointNet and PointNet++ on object classification from full point clouds.

*Equal contribution to the work

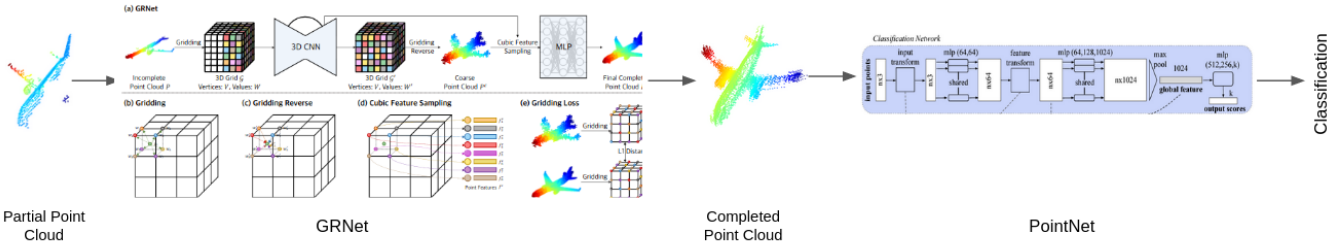

Fig. 1: **Proposed System Architecture:** The proposed system architecture consists of two key components - the GRNet network and PointNet. The system takes partial point clouds as input. The GRNet architecture predicts missing data and completes partial point clouds, followed by processing the completed point clouds for object classification. Overall, the proposed system architecture combines the strengths of both GRNet and PointNet to provide an end-to-end solution for completing partial point clouds and classifying objects within them.

II. Background

A. GRNet

GRNet [2] is a deep learning architecture which is capable to generate high-quality, dense point clouds from incomplete point clouds. GRNet, as shown in figure 2, is composed of three differentiable layers: Gridding, Gridding Reverse, and Cubic Feature Sampling in addition to 3D CNN and MLP. The gridding layer is designed for point cloud completion. It first converts partial point cloud into a regular grid. The grid is regular in the sense that the points in the grid are evenly spaced. In Gridding layer, all vertices of the 3D grid cell corresponding to a point to are weighted with an interpolation function to obtain the geometric relations of the point cloud. After this, a 3D CNN with skip connections (residual network) is applied to learn context and spatially-aware features for completing missing cells of the incomplete point cloud. Gridding Reverse transforms this output 3D grid into a coarse point cloud, and Cubic Feature Sampling does features extraction for each point in the coarse point cloud by concatenating the features of the respective eight vertices of the 3D grid cell. Finally, an MLP is used to obtain the completed point cloud.

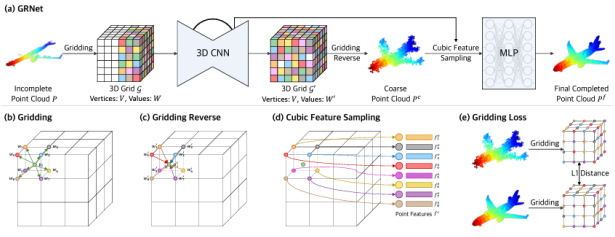

Fig. 2: GRNet Architecture

B. PointNet

Figure 3 shows the architecture of the PointNet [5] network for classification and segmentation. PointNet is highly effective in classifying point clouds that are in arbitrary orientations and positions. PointNet comprises two main components: an input transformation layer and a feature transformation layer. The input transformation layer utilizes a shared multi-layer perceptron to convert the input point cloud into a canonical representation that is invariant to permutations and geometric transformations. Meanwhile, the feature transformation layer learns features from the canonical representation that are also invariant to permutations and geometric transformations. The point features are then aggregated by max-pooling and fed to a multi-layer perceptron (MLP) to output classification scores for k classes. PointNet uses a symmetry function, which is max pooling in this case, to handle unordered input point clouds. This ensures that PointNet can classify point clouds irrespective of the order of the points in the input. The numbers in brackets represent the layer sizes in the MLP, and batch normalization is used for all layers with ReLU activation.

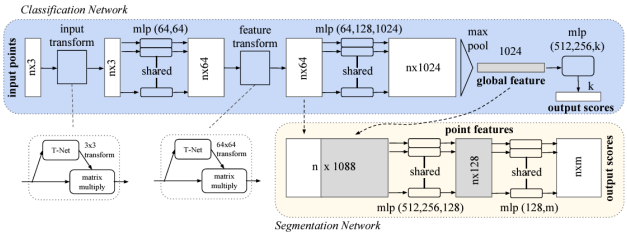

Fig. 3: PointNet Architecture

C. PointNet++

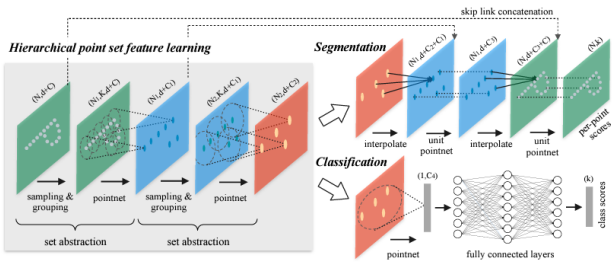

Fig. 4: PointNet++ Architecture

PointNet [5] and PointNet++ [6] differ in their methods of aggregating point sets. While PointNet uses a single max pooling operation for the entire set, PointNet++ creates a hierarchical structure of point groups and gradually abstracts larger local regions as shown in figure 4. The structure is composed of several set abstraction levels, each consisting of three layers: Sampling, Grouping, and PointNet. The Sampling layer selects a group of points to define the centroids of local regions, while the Grouping layer constructs local region sets by identifying neighboring points around the centroids. The PointNet layer employs a mini-PointNet to encode local region patterns into feature vectors. Non-uniform point set density presents a significant challenge for feature learning, but PointNet++ addresses this by extracting multiple scales of local patterns at each abstraction level and intelligently combining them based on the local point densities. After the hierarchical grouping and abstraction process is completed, the resulting features are fed into fully connected layers for object classification.

III. Experiments

In this section, we provide an explanation of the various datasets that are utilized to train and test the different networks that are employed in our project. Following that, we present a comparison of the outcomes achieved by PointNet, our own proposed system, and PointNet ++ for the purpose of 3D object detection from point clouds.

A. Object Detection from ModelNet10 (complete point clouds)

PointNet and PointNet++ were tested on the ModelNet10 [4] dataset for shape classification using full point clouds. The dataset consist of 4,899 mesh files belonging to 10 object categories. The categories include bathtub, bed, chair, desk, dresser, monitor, nightstand, sofa, table, and toilet. The dataset were divided into two sets - 3,991 for training set and 908 for testing set. Farthest point algorithm (FPS) is used to uniformly sample 1024 points and the points are normalized. To improve training performance, the point clouds were augmented by randomly rotating the object and jittering the position of each point by adding Gaussian noise with a zero mean and a standard deviation of 0.02. We used the parameters shown in table II for training.

B. Object Detection from Derived Dataset (complete and partial point clouds)

In order to conduct our experiments, we also created a custom dataset by combining the ShapeNet dataset and the ModelNet10 dataset. We selected 8 categories from the ShapeNet dataset, including airplane, bench, cabinet, car, chair, lamp, sofa, and table, as well as 2 categories from the ModelNet10 dataset, which were bathtub and bed. The ShapeNet dataset contains both complete and partial point clouds, and we used 600 complete point clouds from each category of ShapeNet for training. For the ModelNet10 dataset, we used 106 point clouds for bathtub and 515 for bed for training.

The input format for PointNet and PointNet++ is .txt, but ShapeNet dataset is in .pcd format. To resolve this, we converted the entire dataset to .txt before feeding it into PointNet and PointNet++. Prior to conversion, the pcd files were downsampled using the farthest point algorithm, mean centered, and normalized. The distribution of dataset for training and testing is shown in table I. We used the parameters shown in table II for training.

Dataset Type		Complete Point Clouds		Partial Point Clouds
		Training	Testing	Testing
ShapeNet	All Categories	600	150	150
ModelNet10	Bathtub	106	100	None
	Bed	515	100	None

TABLE I: Distribution for training and testing data from derived dataset

Epoch	15
Batch Size	64
Learning Rate	0.001
Optimizer	Adam
Loss Function	NLLLOSS

TABLE II: Parameters used for training

After training, we tested our models on both complete and partial point clouds. We tested on all 8 categories from the ShapeNet dataset, using 150 complete point clouds for each category. For the ModelNet10 dataset, we used 100 point clouds for bathtub and 100 for bed for testing on full point clouds.

Only the 8 categories from ShapeNet had partial point clouds, so we used only those for testing on partial point clouds. Each category had 150 partial point clouds.

We then used our proposed system comprising of GRNet and PointNet for testing on partial point clouds. GRNet had already been pretrained on this dataset hence we used the pre-trained weights itself for GRNet architecture. While GRNet had already been pretrained on this dataset, we needed to train PointNet and PointNet++ from scratch as they had not been pre-trained on this dataset. However, GRNet outputs are stored in 'h5' file format, requiring us to develop a separate jupyter notebook. This notebook runs GRNet first, followed by the implementation of PointNet from scratch. This allows us to input the output from GRNet in 'h5' directly to PointNet, resulting in a single pipeline for our proposed method.

IV. Result Analysis

We used two parameters, instance accuracy and class accuracy, to compare the performance. Instance accuracy refers to the proportion of correctly classified instances across the whole dataset. It is calculated by dividing the number of correctly predicted class labels with the total number of labels. Class accuracy is calculated by computing the accuracy of the model for each class separately, and then taking the mean of those accuracies.

A. Object Detection from ModelNet10 (complete point clouds)

The training accuracy of PointNet was found to be 92.24%, whereas PointNet++ achieved a training accuracy of 92.8%, as shown in figure 5 and figure 6 respectively. The models were then tested on the test point clouds of ModelNet10, and the corresponding confusion matrices were obtained, as shown in figure 7 and figure 8 respectively. The instance accuracy and class accuracy obtained for PointNet and PointNet++ are shown in table III.

Object Detection from ModelNet10	Instance Accuracy	Class Accuracy
PointNet	92.07%	91.55%
PointNet++	92.29%	91.46%

TABLE III: Accuracy for Object Detection from ModelNet10

The authors had reported instance accuracies of 90.6% for PointNet and 92.2% for PointNet++ based on their testing on the larger ModelNet40 dataset. However, since ModelNet10 is a subset of ModelNet40 [4], the higher testing accuracy obtained in our experiment could be attributed to the fact that there were fewer test objects in the ModelNet10 dataset.

Overall, as expected, the results indicate that PointNet++ outperforms PointNet in terms of accuracy.

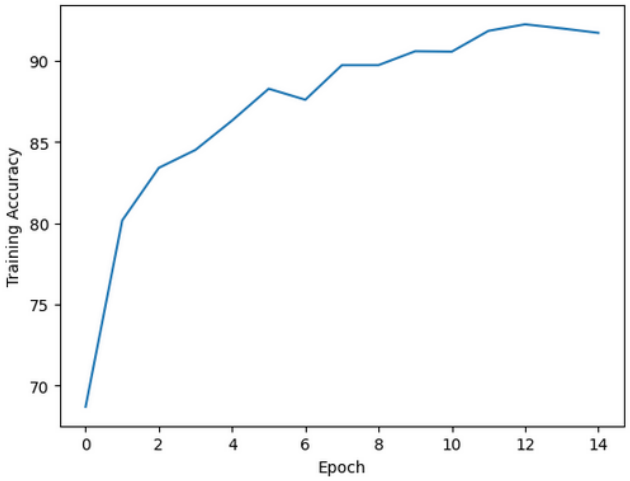

Fig. 5: Training Accuracy plot for PointNet on ModelNet10

B. Object Detection from Derived Dataset (complete and partial point clouds)

The training accuracy of PointNet was found to be 91.63%, whereas PointNet++ achieved a training accuracy of 93.22%, as shown in figure 9 and figure 10 respectively.

The models were then tested on the test complete point clouds of the derived dataset, and the corresponding confusion matrices were obtained, as shown in figure 11 and figure 12 respectively. The instance accuracy and class accuracy obtained for PointNet and PointNet++ are shown in table IV.

Although the accuracies were not compared to those claimed by the author due to different testing datasets,

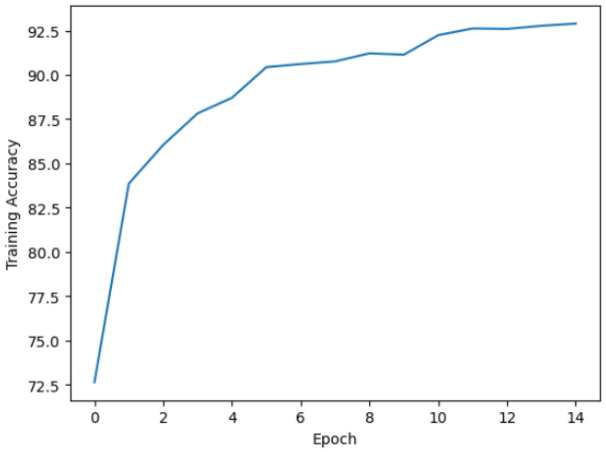

Fig. 6: Training Accuracy plot for PointNet++ on ModelNet10

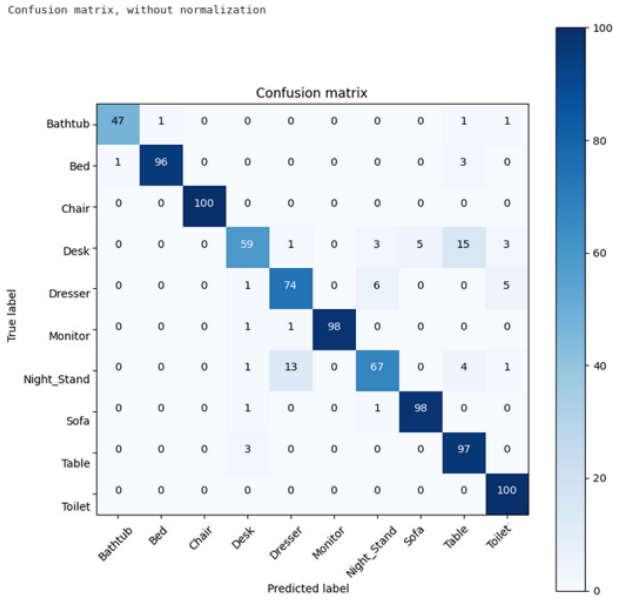

Fig. 7: Confusion Matrix for PointNet on ModelNet10

Object Detection from Derived Dataset (Complete Point Cloud)	Instance Accuracy	Class Accuracy
PointNet	93.18%	93.63%
PointNet++	94.81%	94.83%

TABLE IV: Accuracy for object detection from derived dataset (Complete Point Cloud)

PointNet++ performed better than PointNet on full point clouds as expected.

Object Detection from Derived Dataset (Partial Point Cloud)	Instance Accuracy	Class Accuracy
PointNet	76.91%	76.91%
PointNet++	68.75%	68.75%
Proposed Method (GRNet + PointNet)	93.83%	93.83%

TABLE V: Accuracy for object detection from derived dataset (Partial Point Cloud)

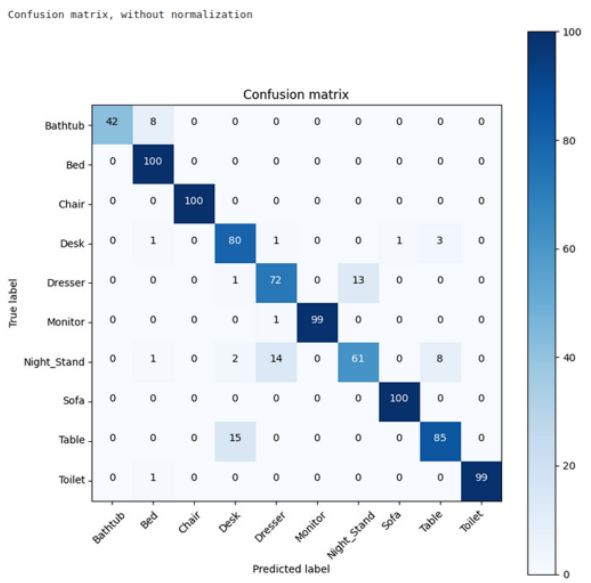

Fig. 8: Confusion Matrix for PointNet++ on ModelNet10

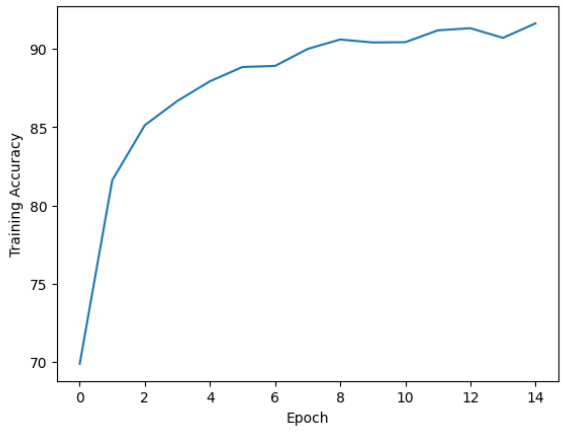

Fig. 9: Training Accuracy plot for PointNet on derived dataset (complete point clouds)

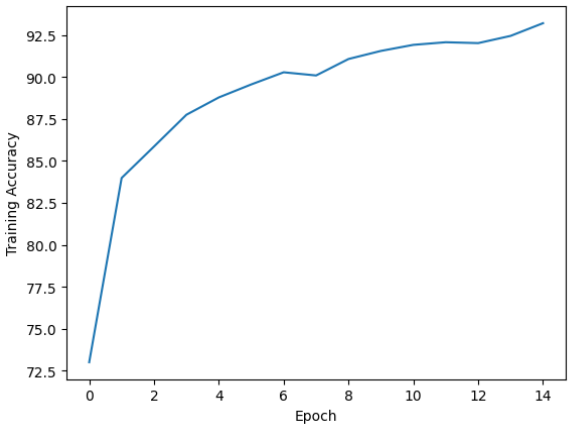

Fig. 10: Training Accuracy plot for PointNet++ on derived dataset (complete point clouds)

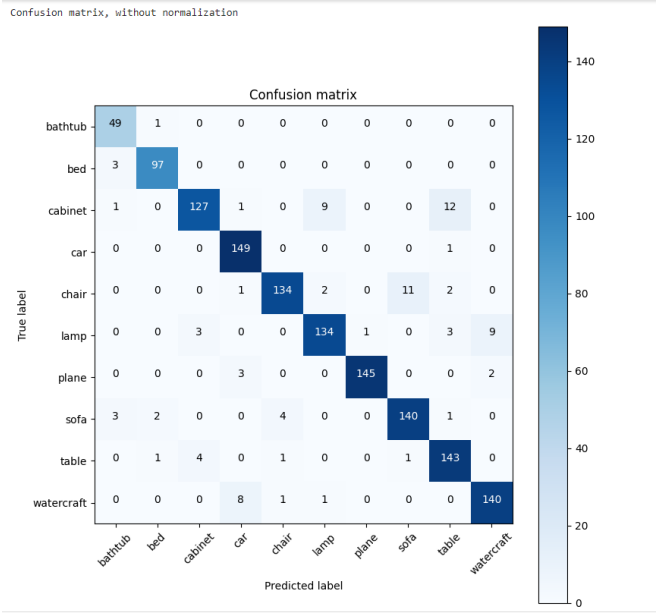

Fig. 11: Confusion Matrix for PointNet on derived dataset (complete point clouds)

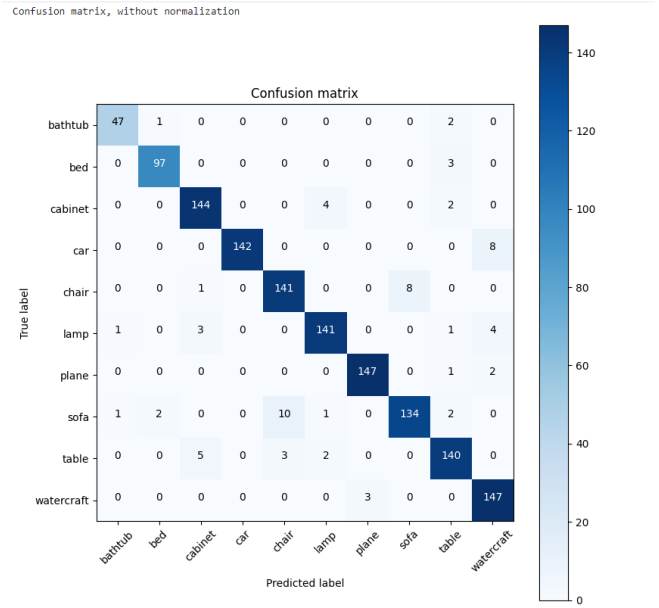

Fig. 12: Confusion Matrix for PointNet++ on derived dataset (complete point clouds)

Next, the trained models were tested on partial point clouds of the same derived dataset, and the corresponding confusion matrices were obtained, revealing that the instance accuracy of PointNet dropped to 76.91%, whereas PointNet++ dropped to 68.75%. The class accuracies of both models were also dropped. The confusion matrices obtained are shown in figure 13 and 14. The accuracy values are shown in table V.

This observation shows that PointNet++ is more sensitive to the missing of points than PointNet. This might be due to the reason that PointNet++ is a hierarchical architecture

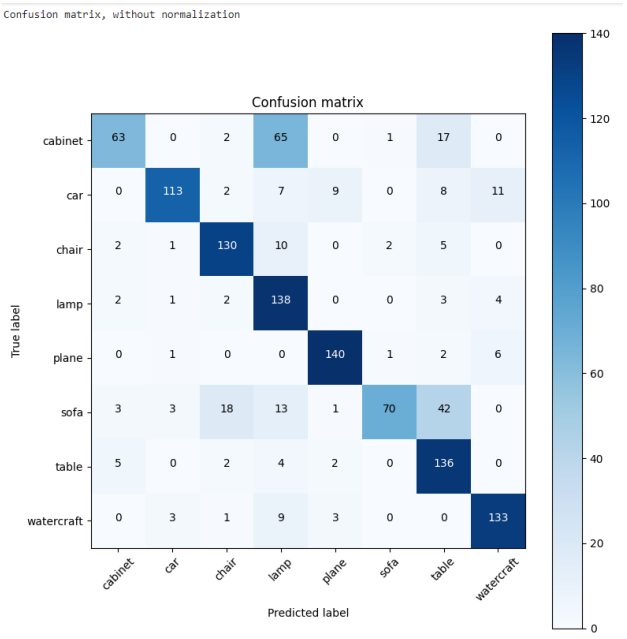

Fig. 13: Confusion Matrix for PointNet on derived dataset(Partial Point Clouds)

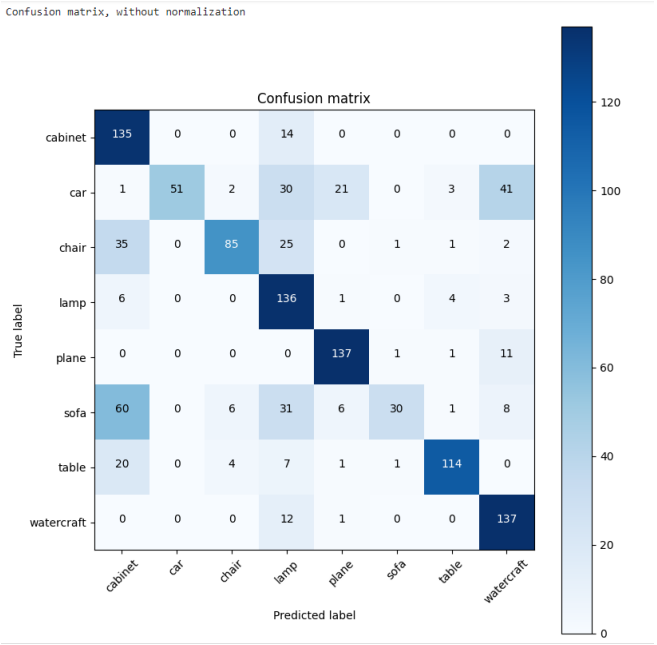

Fig. 14: Confusion Matrix for PointNet++ on derived dataset (Partial Point Clouds)

that first partitions the point cloud into smaller sub-clouds, and then it applies PointNet to each sub-cloud. The output of PointNet for each sub-cloud is then aggregated to produce a global feature vector for the entire point cloud. If a point is missing from a sub-cloud, then PointNet++ will not be able to learn any features for that sub-cloud. This can lead to a significant drop in performance.PointNet, on the other hand, does not partition the point cloud into smaller sub-clouds. This means that PointNet is less sensitive to the missing of points. Even if a point is missing, PointNet will still be able to learn features for the other points in the point cloud. This can lead to better performance on partial point clouds.

Then our proposed method, comprising of the architecture of GRNet and PointNet, is tested on the same partial point clouds and the corresponding confusion matrixes were obtained. As shown in table V, the instance accuracy and class accuracy increased to 93.8%. This is much better than the baseline performance of PointNet++ on partial point clouds.

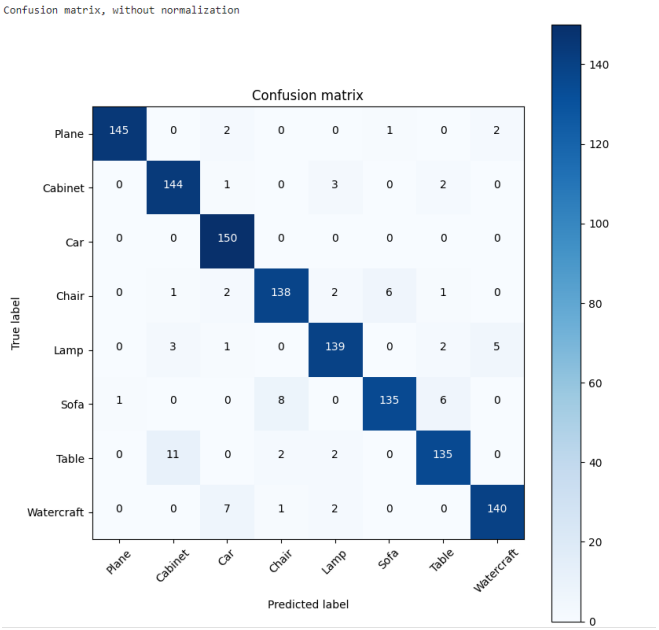

Fig. 15: Confusion Matrix for the Proposed System on derived dataset ((Partial Point Clouds)

V. Conclusion

In this project, we proposed a system for the classification of 3D objects that is capable of handling partial point clouds as inputs. The performance of the proposed system is evaluated and compared against PointNet++ as our baseline model. The experimental results indicate that our proposed model achieves equal or better performance than PointNet++ when partial point clouds are given as inputs.

Acknowledgement

The successful completion of this project is attributed to the exceptional guidance and support of Professor Robert Platt, without whom this work would not have been feasible. We also want to extend our gratitude to the Teaching Assistants whose invaluable assistance and support facilitated our progress throughout the course and project.

References

[1] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "Pcn: Point completion network," in *2018 international conference on 3D vision (3DV)*. IEEE, 2018, pp. 728–737.

[2] H. Xie, H. Yao, S. Zhou, J. Mao, S. Zhang, and W. Sun, "Grnet: Gridding residual network for dense point cloud completion," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX*. Springer, 2020, pp. 365–381.

[3] L. P. Tchapmi, V. Kosaraju, H. Rezatofighi, I. Reid, and S. Savarese, "Topnet: Structural point cloud decoder," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 383–392.

[4] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[5] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[6] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.