**Project – Machine Learning on Graphs 097922**

**Task 1: Point Cloud Classification on ModelNet10**

Roni Fridman – 205517097

Simon Bellilty – 345233563

**1. Introduction**

In this project, we aimed to classify 3D objects in the ModelNet10 dataset, which includes categories such as chairs, tables, and cars, using a point cloud-based classification model. Unlike traditional grid-based image data, point clouds are unordered, irregular, and highly dimensional, making them challenging to process.

To tackle these challenges, we selected a modified PointNet architecture, which directly processes point clouds by applying Multi-Layer Perceptrons (MLPs) to each point independently, as introduced in class. Our model was trained to recognize spatial features in point clouds and classify them into one of the 10 classes provided in the dataset.

**2. Dataset and Preprocessing**

The ModelNet10 dataset contains 3D CAD models represented as point clouds across 10 different classes. We loaded this dataset using PyTorch Geometric, which provided built-in transformations. Our preprocessing steps, who’s goal are to assure uniform input dimensions and a standardized scale, which are essential for enhancing the model's learning efficiency and performance, included sampling and normalization:

**-**We reduced each object to 1024 points using the SamplePoints transformation, providing consistency in point cloud size across all samples.

**-** Each point cloud was scaled to fit within a unit cube, minimizing the influence of size variations and allowing the model to focus primarily on shape rather than scale.

**3. Model Architecture: PointNet**

Our model is based on the **PointNet** architecture, designed specifically for point cloud data, which is unordered and irregular in structure; In this architecture, each point in a point cloud is processed individually, and key features from each point are learned before being combined into a global representation of the entire object.

The model start with passing the input point cloud through a **T-Net** network, which is a module that learns a spatial transformation matrix. This transformation aligns the point cloud, *to reduce variability from different orientations and help the model minimize rotational invariance*. The T-Net’s final layer initializes the transformation matrix close to an identity matrix, ensuring minimal distortion of the input before learning the optimal alignment.

Following the alignment, each point is processed through a series of **1D convolutional layers**. These layers, acting like multi-layer perceptrons (MLPs), *independently learn features for each poi*nt. To maximize training stability and faster convergence, batch normalization is applied after each convolutional layer.

Once point-level features have been extracted, the model aggregates them into a global feature vector using **max pooling**. The global feature vector is then passed through a set of **fully connected layers**, which refine this representation *for classification*. To prevent overfitting and improve generalization, particularly given the limited size of the dataset, dropout is applied in the fully connected layers following global feature extraction.

The model outputs a log-probability distribution over the classes, allowing us to compute and monitoring the loss during training.

With these modifications, including the T-Net for spatial alignment, batch normalization for stability, and dropout for regularization, our model builds on the standard PointNet, enhancing both stability during training and overall classification accuracy.

**4. Training Procedure**

We trained our PointNet model with the following parameters:

**Optimizer**: Adam with a learning rate of 0.001.

**Loss Function**: Negative Log-Likelihood Loss.

**Epochs**: 19 epochs were run to balance model convergence with the risk of overfitting.

**Seed**: A seed of 42 was set to ensure reproducibility.

**Batch size**: 32 for the training set and 64 for test set.

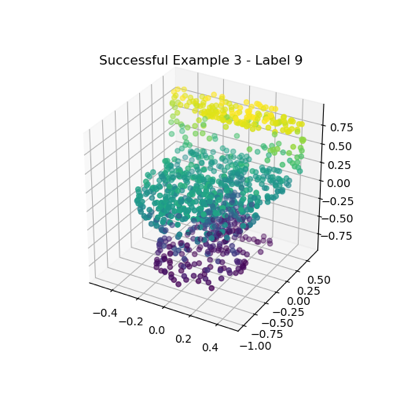
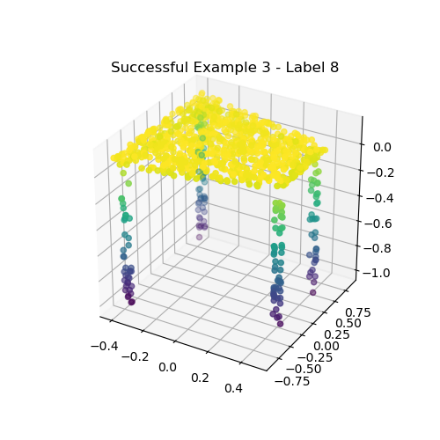
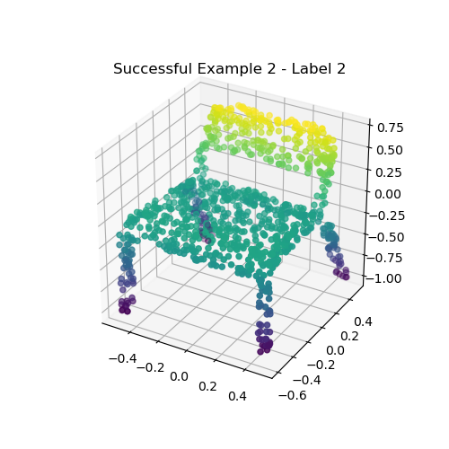
During training, we monitored the loss and accuracy on the test set at each epoch. to ensure steady improvement

**5. Results and Analysis**

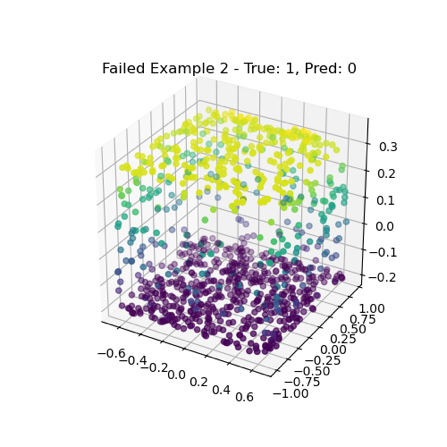
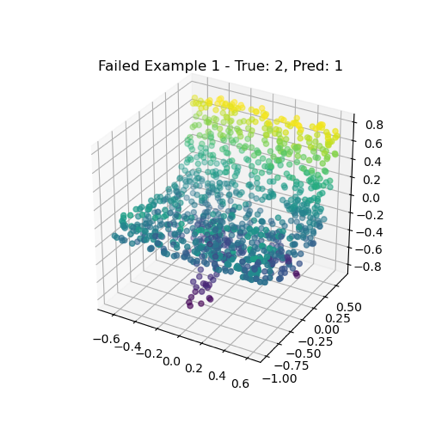
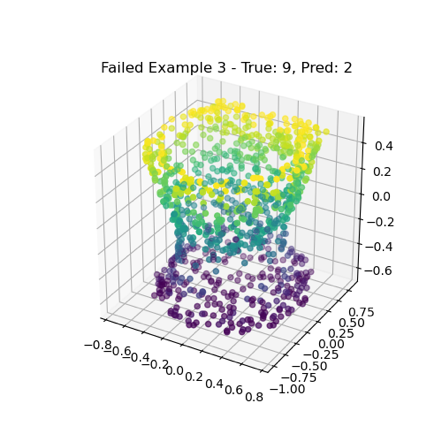
Our final model achieved a test accuracy of **90.09%** , it was able to correctly classify most of the objects in the test set ,which shows strong generalization to new data.

We analyzed both successful and unsuccessful classifications to try to gain insights into the model's strengths and weaknesses.

**Correct classified images examples:**



**Incorrect classified images examples :**

****

In the successful examples, clear and distinguishable regions, such as the separate legs of a chair or a table, provided the model with distinct features, enabling PointNet to recognize these objects accurately. In contrast, the misclassified examples probably have more ambiguous shapes, making it challenging for PointNet to differentiate between classes. In particular when objects shared similar structures. These misclassifications could be from insufficient separation in point regions, orientation issues that limit the T-Net’s effectiveness, or structural similarities that the model struggles to distinguish between classes.

**6. Conclusion**

In this project, we successfully implemented a PointNet model for 3D object classification using the ModelNet10 dataset. Our approach,allowed the model to learn robust spatial features from point clouds. This architecture efficiently handled the unordered and irregular nature of point cloud data, achieving a test accuracy of 90.09%. Through tuning of hyperparameters and preprocessing steps, our model demonstrated strong generalization on new data. This results highlights the model’s capacity to classify 3D objects across multiple categories. Future work could explore enhancements such as hierarchical processing to capture more detailed spatial information (cf PointNet++).