Report

Lab-Neural Network for 3D Point Clouds: PointNet

MLP Neural Networks for Geometric Data - 3D Point Cloud Classification

Prepared by:

Ghzel Samar

Jaouadi Mohamed Amine

Master PAR-3D DEEP LEARNING

UNIVERSITÉ Clermont Auvergne

List of Figures:

Table of Contents

1) Introduction

PointNet is a deep learning architecture designed for processing point clouds, which are sets of 3D points in space. Point clouds are commonly used in computer vision tasks related to 3D perception, such as object recognition and scene understanding. Developed by researchers at Stanford University, PointNet aims to directly process unordered point sets without relying on predefined structures like grids or meshes.

The architecture involves a shared multi-layer perceptron (MLP) network applied independently to each point, followed by a max-pooling layer to obtain the global feature vector. This global feature vector, along with the individual point features, is then used for various tasks such as classification or segmentation.

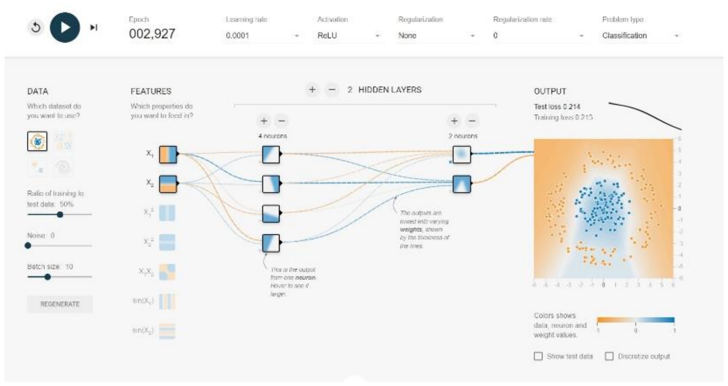
Through the utilization of PointNet, we are harnessing cutting-edge technology to enhance the efficiency and precision of our project.

2) Exercise 1: MLP for Geometric 2D Data Understanding

In exploring the dynamics of traditional neural networks applied to 2D geometric data, we will experiment with a neural network that can be trained using online resources. We can change for that the input data, the noise, the learning rate, the activation function and the number of hidden layers.

a) <u>Learning rate:</u>

In our initial experiment, we set the activation function to ReLU, opted for circular input data for clearer visualization, and configured the number of hidden layers to 2. This allowed us to systematically examine and observe the effects of varying the learning rate, and the subsequent outcomes are as follows.

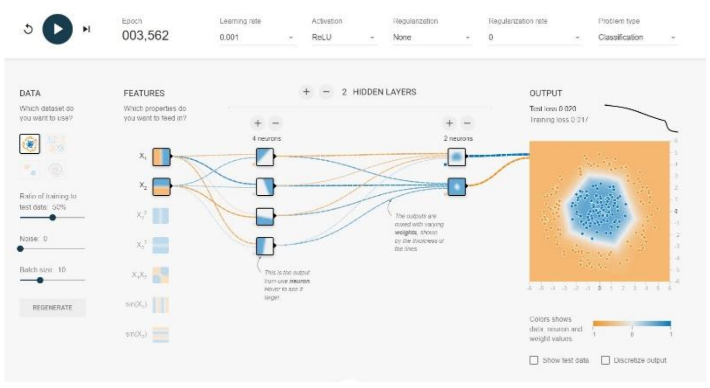


Epoch 002,927 · Learning rate 0.0001 · Activation ReLU · Regularization None · Regularization rate 0 · Problem type Classification

Ratio of training to test data: 50% · Noise: 0 · Batch size: 10 · 2 HIDDEN LAYERS: 4 neurons, 2 neurons

Test loss 0.214 · Training loss 0.215

Figure 3: Learning rate =0.0001

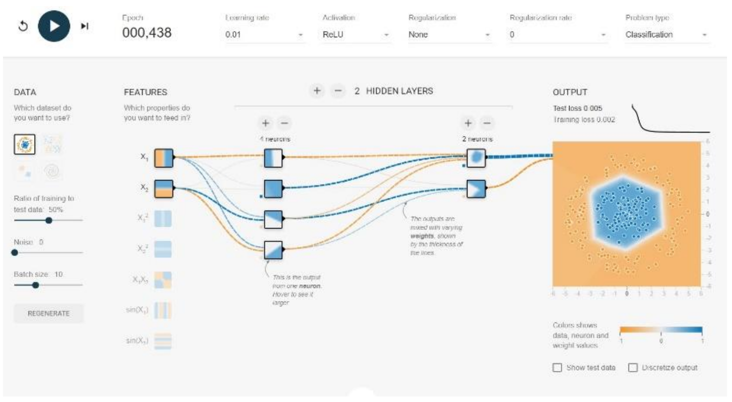


Epoch 003,562 · Learning rate 0.001 · Activation ReLU · Regularization None · Regularization rate 0 · Problem type Classification

Ratio of training to test data: 50% · Noise: 0 · Batch size: 10 · 2 HIDDEN LAYERS: 4 neurons, 2 neurons

Test loss 0.020 · Training loss 0.017

Figure 2:Learning rate =0.001

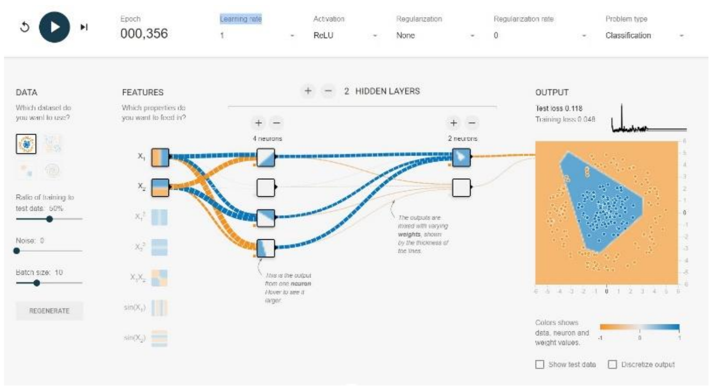


Epoch 000,438 · Learning rate 0.01 · Activation ReLU · Regularization None · Regularization rate 0 · Problem type Classification

Ratio of training to test data: 50% · Noise: 0 · Batch size: 10 · 2 HIDDEN LAYERS: 4 neurons, 2 neurons

Test loss 0.005 · Training loss 0.002

Figure 1:Learning rate =0.01



Epoch 000,356 · Learning rate 1 · Activation ReLU · Regularization None · Regularization rate 0 · Problem type Classification

Ratio of training to test data: 50% · Noise: 0 · Batch size: 10 · 2 HIDDEN LAYERS: 4 neurons, 2 neurons

Test loss 0.118 · Training loss 0.048

Figure 4 : Learning rate =1

The learning rate is a hyperparameter in machine learning algorithms, particularly in optimization algorithms used for training neural networks and other models. It represents the size of the steps taken during the optimization process to adjust the weights of the model.

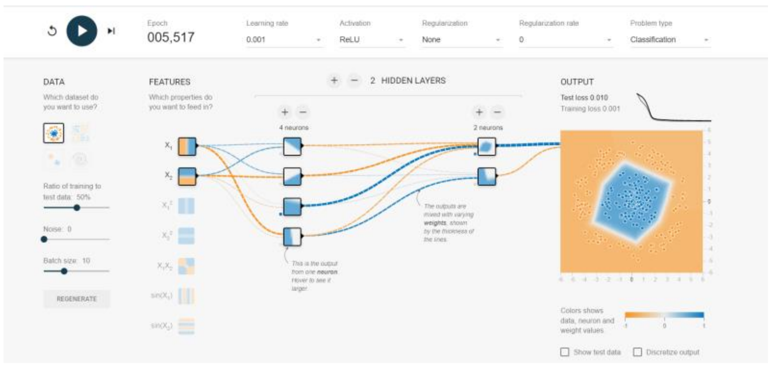
Learning rate	**Observation and conclusion**
0.0001	The convergence was very slow, and the model took a significant amount of time to learn. It also exhibited sensitivity to local minima. However, in certain cases, low learning rate may be necessary for noisy or challenging data.
0.001	Stable convergence is evident, with the model learning consistently without significant slowdown or the risk of divergence. This is typically a favorable starting point for the learning rate.
0.01	The convergence was quick initially, but there's a risk that the model may not fully converge. It could oscillate around the global minimum and even diverge. The weight updates are too large, leading to overshooting the global minimum.

As a conclusion, we use a very high learning rate (=1) it can cause several consequences such as Overshooting, Divergence and an unstable training process. To ensure a balance between speed and stability, we settled on a learning rate of 0.001.

b) Activation Function :

An activation function is a mathematical operation applied to the output of a neuron in a neural network. It introduces non-linearity to the model, enabling the network to learn complex patterns and relationships in the data.

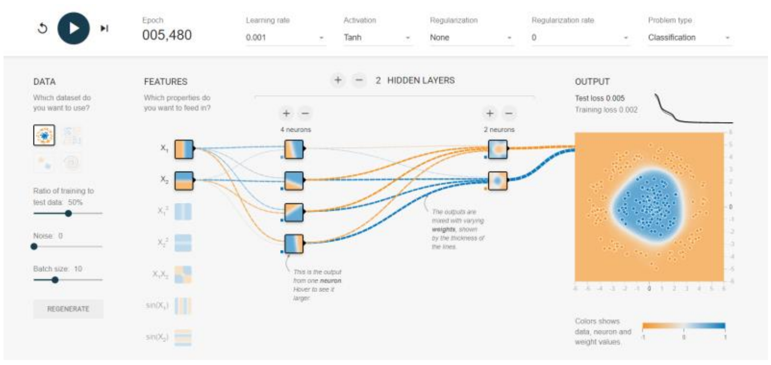
Relu / sigmoid / tanh


Epoch 005,517 | Learning rate 0.001 | Activation ReLU | Regularization None | Regularization rate 0 | Problem type Classification

Test loss 0.010
Training loss 0.001

DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
OUTPUT
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.
Show test data
Discretize output

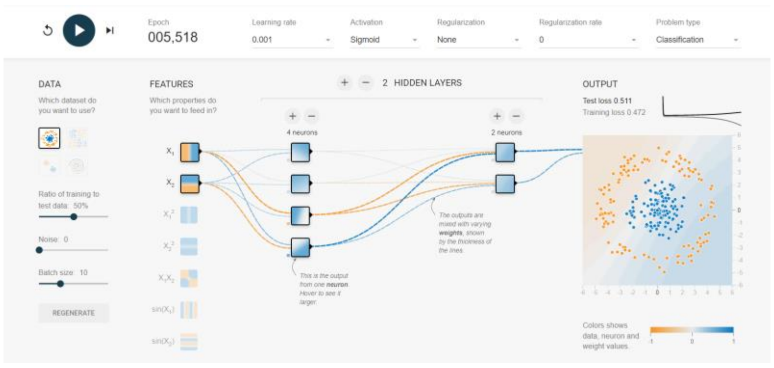
Figure 4: Activation Function = ReLu


Epoch 005,480 | Learning rate 0.001 | Activation Tanh | Regularization None | Regularization rate 0 | Problem type Classification

Test loss 0.005
Training loss 0.002

DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
2 HIDDEN LAYERS
4 neurons
2 neurons
OUTPUT
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.

Figure 5: Activation Function = Tanh



Epoch 005,518

Learning rate 0.001

Activation Sigmoid

Regularization None

Regularization rate 0

Problem type Classification

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

2 HIDDEN LAYERS

Test loss 0.511

Training loss 0.472

This is the output from one **neuron**. Hover to see it larger.

The outputs are mixed with varying **weights**, shown by the thickness of the lines.

Colors shows data, neuron and weight values.

Figure 6: Activation Function = Sigmoid

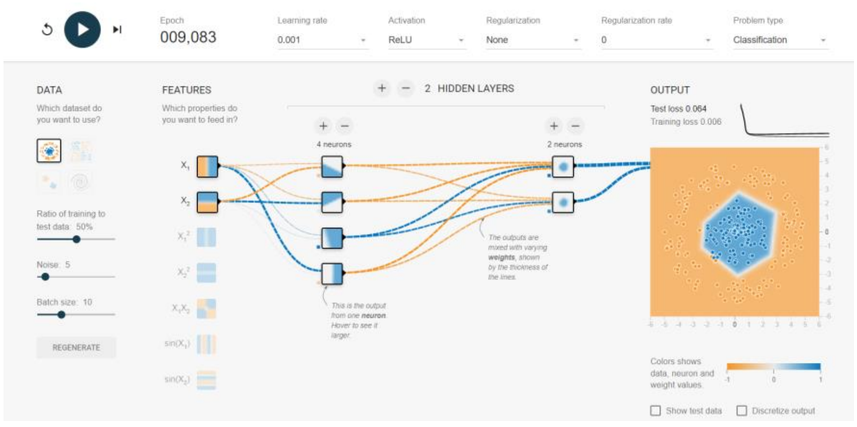
Adding non-linear activation functions allows neural networks to model intricate relationships and capture more complex patterns in the data. In our exercise, we experimented with different activation functions, including ReLU, sigmoid, tanh, and leaky ReLU. ReLU proved to be an effective activation function for the 2D spiral input data, aiding the network in capturing intricate non-linear relationships within the dataset.

We chose Relu because it often works well in hidden layers, promoting faster training.

c) Noise:

Noise can manifest as random variations or errors that are not part of the underlying pattern or information being analyzed. It can arise from various sources, including measurement errors, environmental factors, interference, or limitations in the data acquisition process.

Noise	Observation and conclusion
5	The overall structure of the geometric shapes remained relatively intact. the data points are perturbed slightly from their original positions.
10	Increasing the noise level to 10 introduced more noticeable distortion to the data points. Greater variability in the positions. the data might become more challenging for the neural network.
20	At a noise level of 20, the added variation is substantial, The data points will undergo significant random displacements, resulting in highly distorted shapes and patterns. It becomes much more difficult for the neural network to discern meaningful features or relationships within the data



Epoch 009,083 | Learning rate 0.001 | Activation ReLU | Regularization None | Regularization rate 0 | Problem type Classification

DATA — Which dataset do you want to use? Ratio of training to test data: 50% | Noise: 5 | Batch size: 10

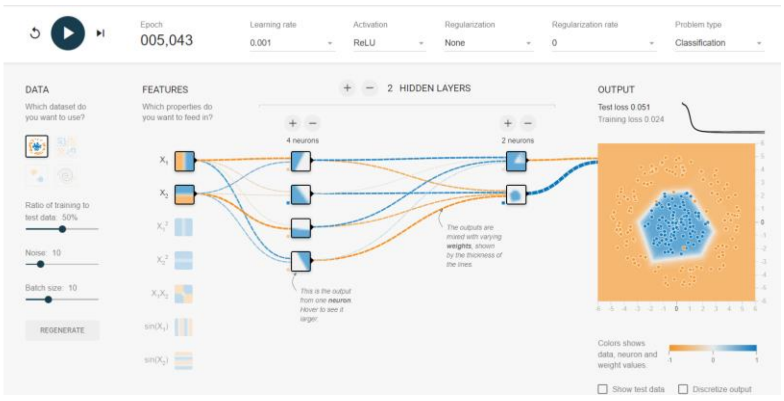
FEATURES — Which properties do you want to feed in?

2 HIDDEN LAYERS — 4 neurons, 2 neurons

OUTPUT — Test loss 0.064, Training loss 0.006

REGENERATE
X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.
-1 0 1
Show test data
Discretize output

Figure 7: Noise = 5



Epoch 005,043 | Learning rate 0.001 | Activation ReLU | Regularization None | Regularization rate 0 | Problem type Classification

DATA — Which dataset do you want to use? Ratio of training to test data: 50% | Noise: 10 | Batch size: 10

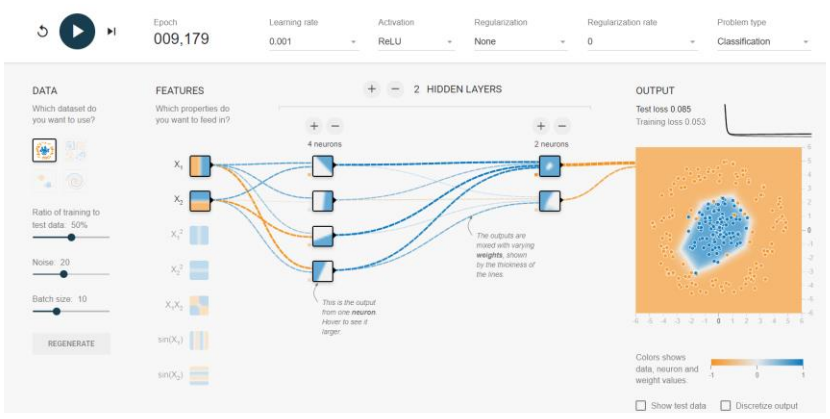
FEATURES — Which properties do you want to feed in?

2 HIDDEN LAYERS — 4 neurons, 2 neurons

OUTPUT — Test loss 0.051, Training loss 0.024

REGENERATE
X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.
-1 0 1
Show test data
Discretize output

Figure 8: Noise= 10



Epoch 009,179 | Learning rate 0.001 | Activation ReLU | Regularization None | Regularization rate 0 | Problem type Classification

DATA — Which dataset do you want to use? Ratio of training to test data: 50% | Noise: 20 | Batch size: 10

FEATURES — Which properties do you want to feed in?

2 HIDDEN LAYERS — 4 neurons, 2 neurons

OUTPUT — Test loss 0.085, Training loss 0.053

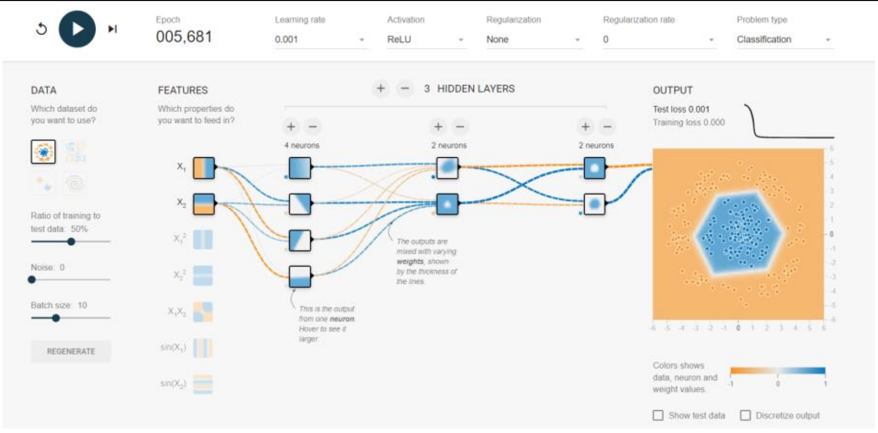
REGENERATE
X_1
X_2
X_1^2
X_2^2
X_1X_2
$\sin(X_1)$
$\sin(X_2)$
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.
-1 0 1
Show test data
Discretize output

Figure 9: Noise= 20

d) Number of hidden layers:

Hidden layers are intermediate layers of neurons in a neural network that exist between the input layer and the output layer. These layers play a crucial role in the network's ability to learn.

Number	Observation and conclusion
< 3	With fewer than three hidden layers, the neural network's performance fell short, prompting the necessity for improvement.
3	The result was good, and it captured some degree of hierarchical features in the data. Training was relatively faster compared to deeper architectures, and there is a balance between model complexity and computational efficiency.
> 3	A neural network with more than three layers is even deeper, increasing the risk of overfitting.



Epoch 005,681 — Learning rate 0.001 — Activation ReLU — Regularization None — Regularization rate 0 — Problem type Classification

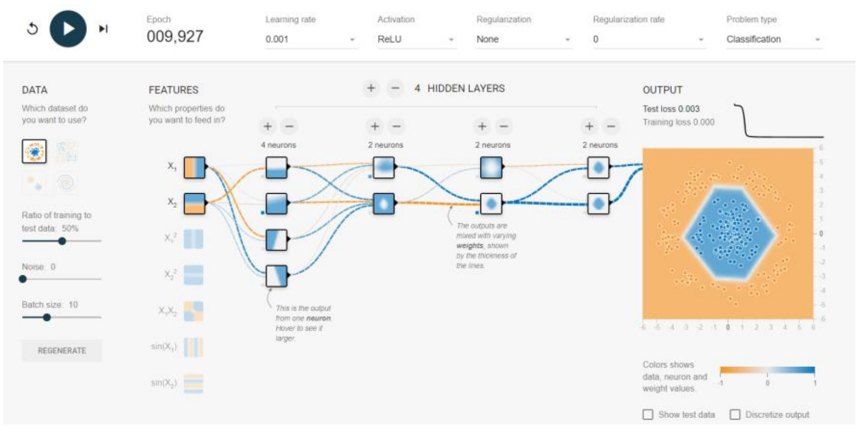
3 HIDDEN LAYERS: 4 neurons, 2 neurons, 2 neurons

Test loss 0.001
Training loss 0.000

Ratio of training to test data: 50% — Noise: 0 — Batch size: 10

DATA
Which dataset do you want to use?
FEATURES
Which properties do you want to feed in?
OUTPUT
REGENERATE
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.
Show test data
Discretize output

Figure 10: 3 hidden layers



Epoch 009,927 — Learning rate 0.001 — Activation ReLU — Regularization None — Regularization rate 0 — Problem type Classification

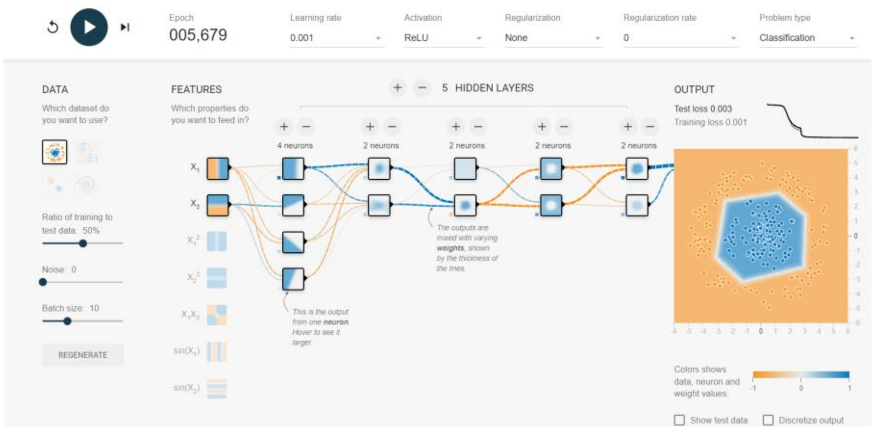
4 HIDDEN LAYERS: 4 neurons, 2 neurons, 2 neurons, 2 neurons

Test loss 0.003
Training loss 0.000

Ratio of training to test data: 50% — Noise: 0 — Batch size: 10

DATA
Which dataset do you want to use?
FEATURES
Which properties do you want to feed in?
OUTPUT
REGENERATE
This is the output from one neuron. Hover to see it larger.
The outputs are mixed with varying weights, shown by the thickness of the lines.
Colors shows data, neuron and weight values.
Show test data
Discretize output

Figure 11: 4 hidden layers



Epoch 005,679

Learning rate 0.001

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

5 HIDDEN LAYERS

Test loss 0.003

Training loss 0.001

This is the output from one **neuron**. Hover to see it larger.

The outputs are mixed with varying **weights**, shown by the thickness of the lines.

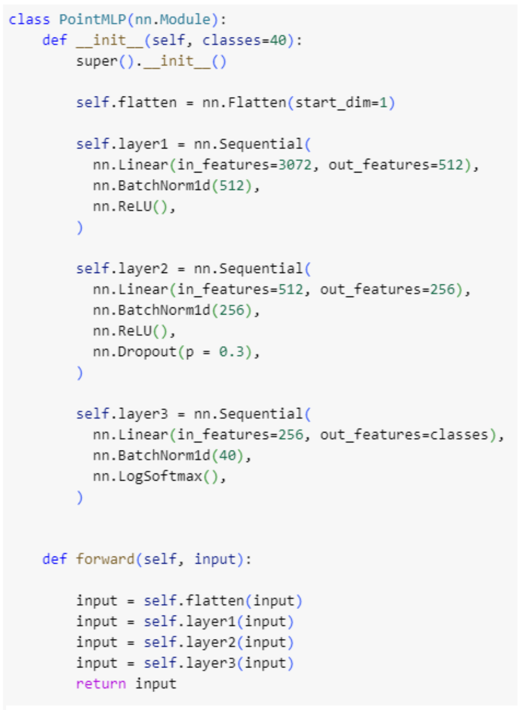
Colors shows data, neuron and weight values.

Figure 12: 5 hidden layers

3) Exercise 2: PointMLP in PyTorch

In the architecture of the classical 3 layers MLP neural network the input consists of a flattened point cloud with 1024 points, each represented by three coordinates (x, y, z), resulting in 3072 inputs. The first layer is a Multilayer Perceptron (MLP) with 3072 input nodes and 512 hidden nodes. The second layer is another MLP with 512 input nodes, 256 hidden nodes, and a weight dropout of p = 0.3 to prevent overfitting. The final layer is an MLP with 256 input nodes and an output size N, representing the number of classes. The architecture employs the logsoftmax () function to calculate scores for each class. Additionally, each layer incorporates batch normalization and ReLU activation functions, contributing to the model's capacity to learn and generalize effectively.

You will find this architecture implemented in our code.



```
class PointMLP(nn.Module):
    def __init__(self, classes=40):
        super().__init__()

        self.flatten = nn.Flatten(start_dim=1)

        self.layer1 = nn.Sequential(
          nn.Linear(in_features=3072, out_features=512),
          nn.BatchNorm1d(512),
          nn.ReLU(),
        )

        self.layer2 = nn.Sequential(
          nn.Linear(in_features=512, out_features=256),
          nn.BatchNorm1d(256),
          nn.ReLU(),
          nn.Dropout(p = 0.3),
        )

        self.layer3 = nn.Sequential(
          nn.Linear(in_features=256, out_features=classes),
          nn.BatchNorm1d(40),
          nn.LogSoftmax(),
        )


    def forward(self, input):

        input = self.flatten(input)
        input = self.layer1(input)
        input = self.layer2(input)
        input = self.layer3(input)
        return input
```

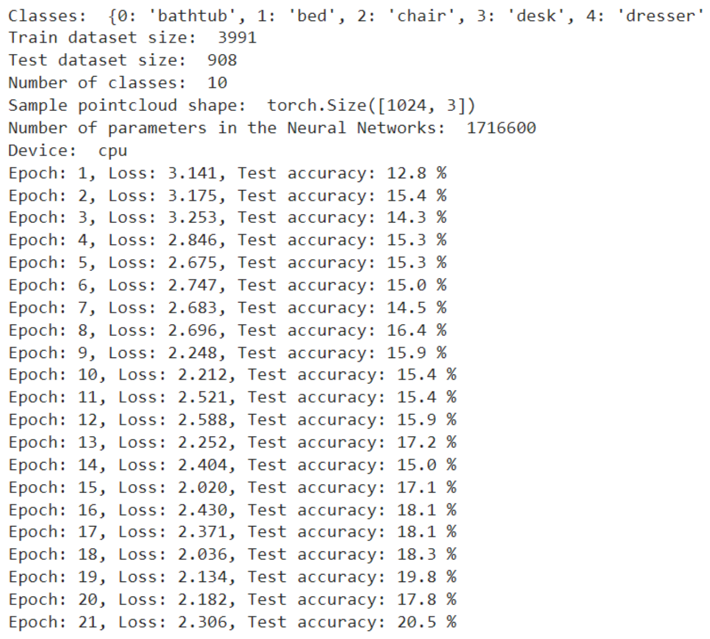
Figure 13: pointMLP code

a) Test on the ModelNet10 PLY

Initially, we set the number of epochs to 25 and observed an increase in accuracy from **14.1% to 21.5%,** accompanied by a decrease in loss from **3.411 to 1.709**. Despite the improvements, the obtained results were not satisfactory. As a result, we made the decision to increase the number of epochs to 50 and continue training the model.

This decision was based on the understanding that increasing the number of epochs might allow the model to learn more complex patterns in the data and potentially improve its performance further. By extending the training duration, we aim to refine the model's ability to generalize and make accurate predictions on unseen data.

After setting the number of epochs to 50, we observed a significant advancement in the model's performance. The accuracy improved notably, reaching 24.0%, while the loss decreased to 1.426. This clear progress suggests that extending the training duration allowed the model to gain a deeper understanding of the underlying patterns in the data. By training for a longer period, the model had more opportunities to adjust its parameters and optimize its predictions, leading to enhanced accuracy and reduced loss.



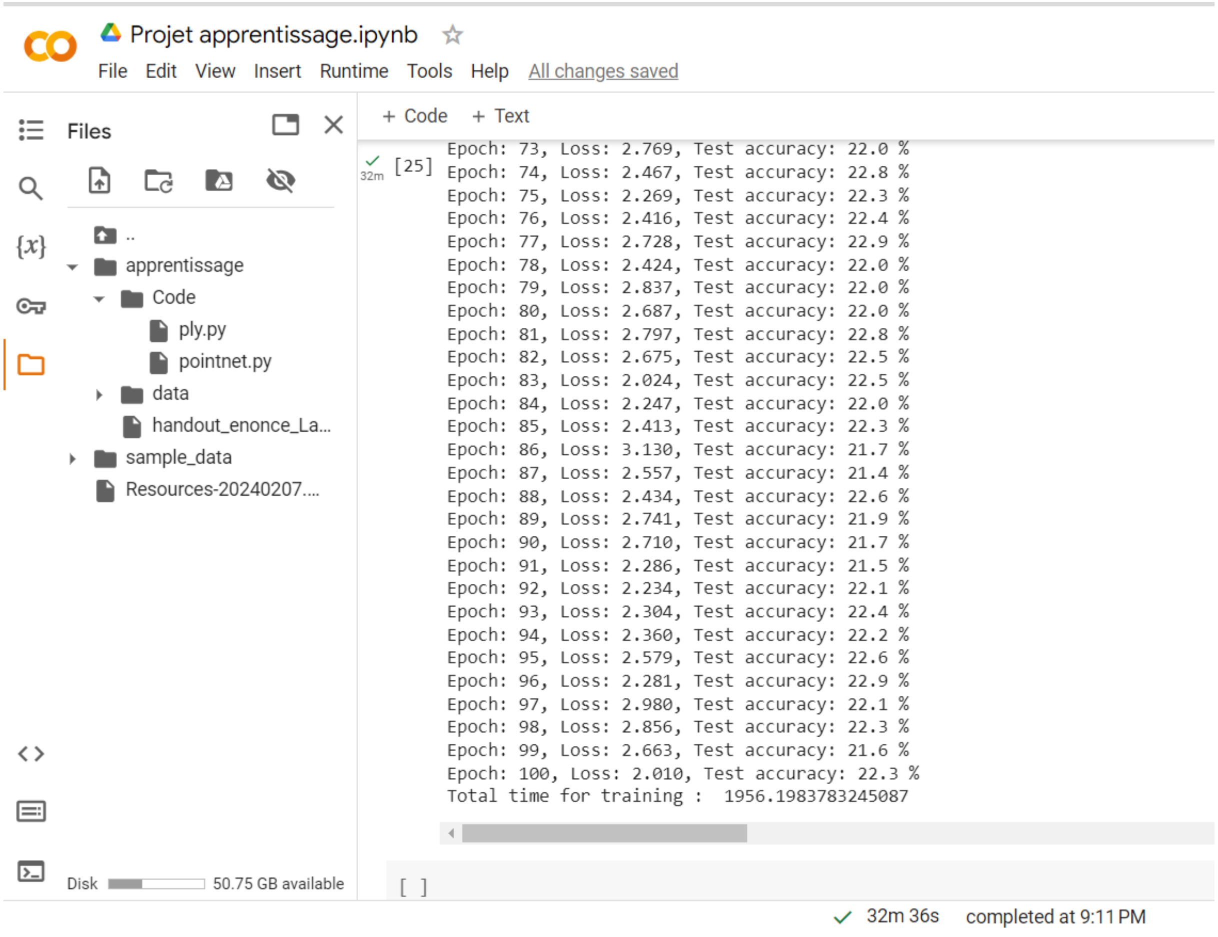
```
Classes:  {0: 'bathtub', 1: 'bed', 2: 'chair', 3: 'desk', 4: 'dresser'
Train dataset size:  3991
Test dataset size:  908
Number of classes:  10
Sample pointcloud shape:  torch.Size([1024, 3])
Number of parameters in the Neural Networks:  1716600
Device:  cpu
Epoch: 1, Loss: 3.141, Test accuracy: 12.8 %
Epoch: 2, Loss: 3.175, Test accuracy: 15.4 %
Epoch: 3, Loss: 3.253, Test accuracy: 14.3 %
Epoch: 4, Loss: 2.846, Test accuracy: 15.3 %
Epoch: 5, Loss: 2.675, Test accuracy: 15.3 %
Epoch: 6, Loss: 2.747, Test accuracy: 15.0 %
Epoch: 7, Loss: 2.683, Test accuracy: 14.5 %
Epoch: 8, Loss: 2.696, Test accuracy: 16.4 %
Epoch: 9, Loss: 2.248, Test accuracy: 15.9 %
Epoch: 10, Loss: 2.212, Test accuracy: 15.4 %
Epoch: 11, Loss: 2.521, Test accuracy: 15.4 %
Epoch: 12, Loss: 2.588, Test accuracy: 15.9 %
Epoch: 13, Loss: 2.252, Test accuracy: 17.2 %
Epoch: 14, Loss: 2.404, Test accuracy: 15.0 %
Epoch: 15, Loss: 2.020, Test accuracy: 17.1 %
Epoch: 16, Loss: 2.430, Test accuracy: 18.1 %
Epoch: 17, Loss: 2.371, Test accuracy: 18.1 %
Epoch: 18, Loss: 2.036, Test accuracy: 18.3 %
Epoch: 19, Loss: 2.134, Test accuracy: 19.8 %
Epoch: 20, Loss: 2.182, Test accuracy: 17.8 %
Epoch: 21, Loss: 2.306, Test accuracy: 20.5 %
```

Figure 14:result with the modelnet10_ply

b) Test on the ModelNet40 PLY

Epochs=25

Classes: {0: 'airplane', 1: 'bathtub', 2: 'bed', 3: 'bench', 4: 'bookshelf', 5: 'bottle', 6: 'bowl', 7: 'car', 8: 'chair', 9: 'cone', 10: 'cup', 11: 'curtain', 12: 'desk', 13: 'door', 14: 'dresser', 15: 'flower_pot', 16: 'glass_box', 17: 'guitar', 18: 'keyboard', 19: 'lamp', 20: 'laptop', 21: 'mantel', 22: 'monitor', 23: 'night_stand', 24: 'person', 25: 'piano', 26: 'plant', 27: 'radio', 28: 'range_hood', 29: 'sink', 30: 'sofa', 31: 'stairs', 32: 'stool', 33: 'table', 34: 'tent', 35: 'toilet', 36: 'tv_stand', 37: 'vase', 38: 'wardrobe', 39: 'xbox'} Train dataset size: 9843 Test dataset size: 2468 Number of classes: 40 Sample pointcloud shape: torch.Size([1024, 3]) Number of parameters in the Neural Networks: 1716600 Device: cuda:0 Epoch: 1, Loss: 3.821, Test accuracy: 5.3 % Epoch: 2, Loss: 3.624, Test accuracy: 7.7 % Epoch: 3, Loss: 3.323, Test accuracy: 7.3 % Epoch: 4, Loss: 3.247, Test accuracy: 8.4 % Epoch: 5, Loss: 3.073, Test accuracy: 9.6 % Epoch: 6, Loss: 3.120, Test accuracy: 10.3 % Epoch: 7, Loss: 2.707, Test accuracy: 11.5 % Epoch: 8, Loss: 2.915, Test accuracy: 13.7 % Epoch: 9, Loss: 2.847, Test accuracy: 14.1 % Epoch: 10, Loss: 2.624, Test accuracy: 14.5 % Epoch: 11, Loss: 3.023, Test accuracy: 14.7 % Epoch: 12, Loss: 3.188, Test accuracy: 14.3 % Epoch: 13, Loss: 2.747, Test accuracy: 15.4 % Epoch: 14, Loss: 2.929, Test accuracy: 16.4 % Epoch: 15, Loss: 2.486, Test accuracy: 17.3 % Epoch: 16, Loss: 3.315, Test accuracy: 16.5 % Epoch: 17, Loss: 2.563, Test accuracy: 17.7 % Epoch: 18, Loss: 2.476, Test accuracy: 17.3 % Epoch: 19, Loss: 2.713, Test accuracy: 17.0 % Epoch: 20, Loss: 3.111, Test accuracy: 17.9 % Epoch: 21, Loss: 2.892, Test accuracy: 19.0 % Epoch: 22, Loss: 2.435, Test accuracy: 18.4 % Epoch: 23, Loss: 2.881, Test accuracy: 18.7 % Epoch: 24, Loss: 2.603, Test accuracy: 18.5 % Epoch: 25, Loss: 2.241, Test accuracy: 18.3 % Total time for training : 476.3494756221771



```
Epoch: 73, Loss: 2.769, Test accuracy: 22.0 %
Epoch: 74, Loss: 2.467, Test accuracy: 22.8 %
Epoch: 75, Loss: 2.269, Test accuracy: 22.3 %
Epoch: 76, Loss: 2.416, Test accuracy: 22.4 %
Epoch: 77, Loss: 2.728, Test accuracy: 22.9 %
Epoch: 78, Loss: 2.424, Test accuracy: 22.0 %
Epoch: 79, Loss: 2.837, Test accuracy: 22.0 %
Epoch: 80, Loss: 2.687, Test accuracy: 22.0 %
Epoch: 81, Loss: 2.797, Test accuracy: 22.8 %
Epoch: 82, Loss: 2.675, Test accuracy: 22.5 %
Epoch: 83, Loss: 2.024, Test accuracy: 22.5 %
Epoch: 84, Loss: 2.247, Test accuracy: 22.0 %
Epoch: 85, Loss: 2.413, Test accuracy: 22.3 %
Epoch: 86, Loss: 3.130, Test accuracy: 21.7 %
Epoch: 87, Loss: 2.557, Test accuracy: 21.4 %
Epoch: 88, Loss: 2.434, Test accuracy: 22.6 %
Epoch: 89, Loss: 2.741, Test accuracy: 21.9 %
Epoch: 90, Loss: 2.710, Test accuracy: 21.7 %
Epoch: 91, Loss: 2.286, Test accuracy: 21.5 %
Epoch: 92, Loss: 2.234, Test accuracy: 22.1 %
Epoch: 93, Loss: 2.304, Test accuracy: 22.4 %
Epoch: 94, Loss: 2.360, Test accuracy: 22.2 %
Epoch: 95, Loss: 2.579, Test accuracy: 22.6 %
Epoch: 96, Loss: 2.281, Test accuracy: 22.9 %
Epoch: 97, Loss: 2.980, Test accuracy: 22.1 %
Epoch: 98, Loss: 2.856, Test accuracy: 22.3 %
Epoch: 99, Loss: 2.663, Test accuracy: 21.6 %
Epoch: 100, Loss: 2.010, Test accuracy: 22.3 %
Total time for training :  1956.1983783245087
```

Test accuracy on ModelNet40_PLY with PointMLP neural network went from 5.6% to 22% with 100 epochs. By increasing the number of training epochs from 50 to 100, you observed a significant improvement in test accuracy. The accuracy jumped to 22%, indicating that the model's performance substantially improved with additional training epochs.

This suggests that the model needed more training time to learn meaningful representations from the data and improve its ability to generalize to unseen examples. Alongside the accuracy improvement, you also observed a decrease in loss from 3.45 to 2.010. A decrease in loss indicates that the model's predictions are getting closer to the ground truth labels during training. This suggests that the model is learning to minimize its prediction errors as training progresses.

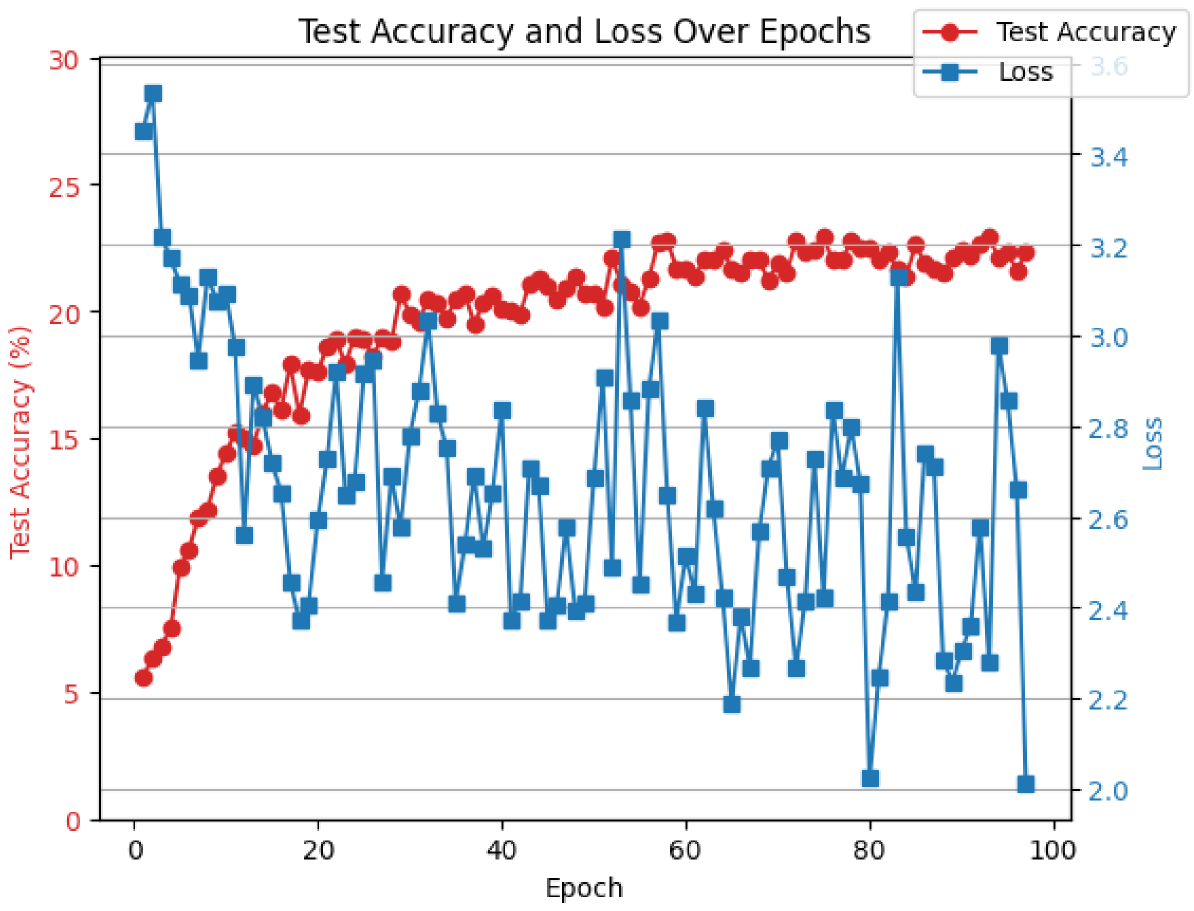


Figure 15: PointMLP Test on the ModelNet40 PLY

c) Conclusion

The difference between ModelNet10 and ModelNet40 lies in the number of object categories they contain and the diversity of objects within those categories. ModelNet10: Contains 10 object categories however ModelNet40: Contains 40 object categories. It includes a wider variety of object types compared to ModelNet10, covering more specific subclasses within each category. ModelNet40: Offers a more diverse set of objects, with a greater number of categories representing a wider range of object types. This diversity allows for more comprehensive training and evaluation of algorithms.

The better performance of ModelNet40 compared to ModelNet10 from the first 25 epochs underscores **the importance of dataset diversity and complexity** in training deep learning models for object recognition tasks. The broader range of object categories and variations in ModelNet40 provides a richer training experience, leading to improved model performance.

4) Exercise 3 PointNet in PyTorch:

The PointNet architecture, introduced in the original paper "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," is a pioneering deep learning model for processing point cloud data directly.

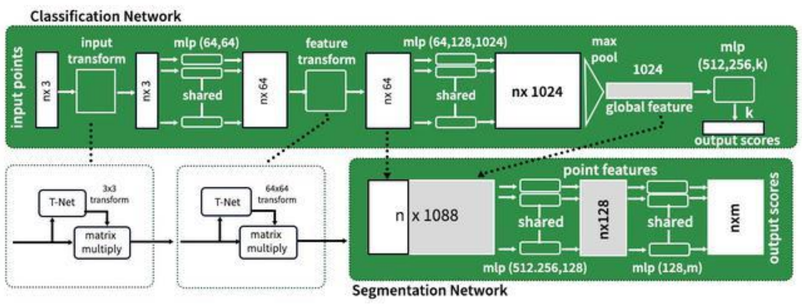


Figure 16:PointNet architecture

Here's a concise summary of the architecture outlined in the paper.

- ✓ First a T-Net network output a 3 3 matrix to align 3D point clouds
- ✓ 2 layers shared MLP over points to compute features of dim 64:
  - i. MLP (3, 64)
  - ii. MLP (64, 64)
- ✓ A new T-Net network output 64 64 matrix to align points in feature space of dim 64

- ✓ 3 layers shared MLP over points to compute features of dim 1024:
  - i. MLP (64, 64)
  - ii. MLP (64, 128)
  - iii. MLP (128, 1024)
- ✓ A symmetric function (max pooling) to compute a global feature of dim 1024
- ✓ Finally, 3 layers MLP from the global feature to output scores for each class:
  - i. MLP (1024, 512)
  - ii. MLP (512, 256) with a weight dropout of p = 0.3
  - iii. MLP (128, 1024) with N the number of classes

a) Basic version of POINTNET:

We will first implement a basic version of pointnet without the T-Nets. All layers, except the last one include ReLu and batch normalization. So, as we did in the first exercise, we will try it on ModelNet10_PLY and ModelNet40_PLY to see the test accuracy and loss of each one and do a comparison.

i. *Test on the ModelNet10 PLY*

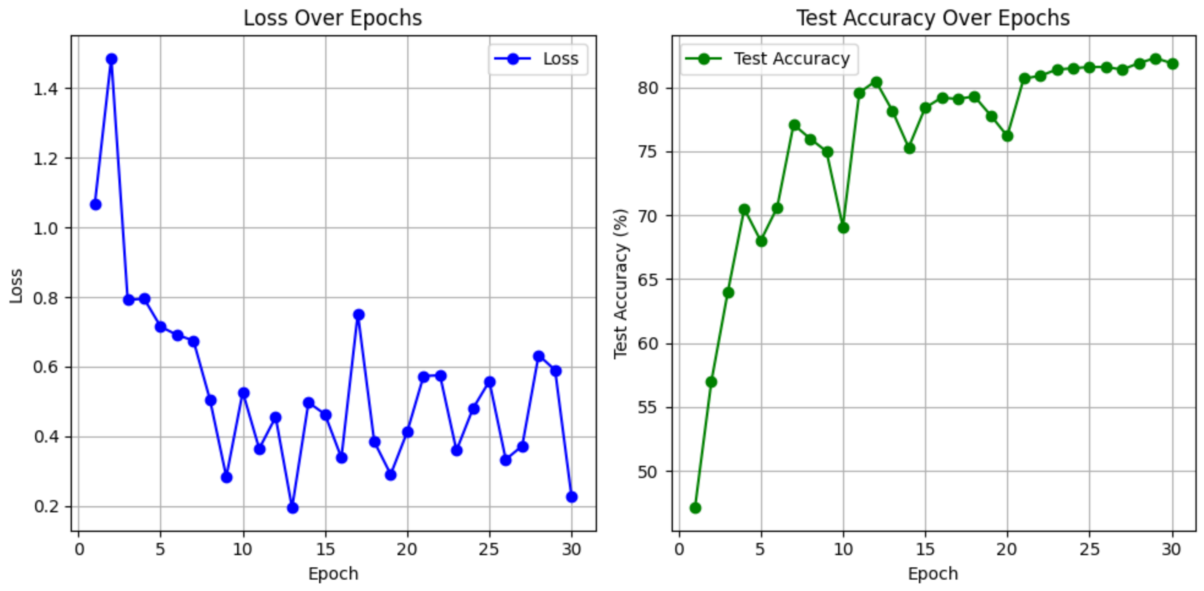


Figure 17: Basic PointNet Test on the ModelNet10 PLY

```
Classes:  {0: 'bathtub', 1: 'bed', 2: 'chair', 3: 'desk', 4: 'dresser',
5: 'monitor', 6: 'night_stand', 7: 'sofa', 8: 'table', 9: 'toilet'}

Train dataset size:  3991
Test dataset size:  908
Number of classes:  10
Sample pointcloud shape:  torch.Size([1024, 3])
Number of parameters in the Neural Networks:  810378
Device:  cuda:0
Epoch: 1, Loss: 1.068, Test accuracy: 47.1 %
Epoch: 2, Loss: 1.487, Test accuracy: 57.0 %
Epoch: 3, Loss: 0.792, Test accuracy: 64.0 %
Epoch: 4, Loss: 0.795, Test accuracy: 70.5 %
Epoch: 5, Loss: 0.715, Test accuracy: 68.0 %
Epoch: 6, Loss: 0.691, Test accuracy: 70.6 %
Epoch: 7, Loss: 0.675, Test accuracy: 77.1 %
Epoch: 8, Loss: 0.506, Test accuracy: 76.0 %
Epoch: 9, Loss: 0.283, Test accuracy: 75.0 %
Epoch: 10, Loss: 0.527, Test accuracy: 69.1 %
Epoch: 11, Loss: 0.364, Test accuracy: 79.6 %
Epoch: 12, Loss: 0.455, Test accuracy: 80.5 %
Epoch: 13, Loss: 0.194, Test accuracy: 78.2 %
Epoch: 14, Loss: 0.497, Test accuracy: 75.3 %
Epoch: 15, Loss: 0.463, Test accuracy: 78.4 %
Epoch: 16, Loss: 0.338, Test accuracy: 79.2 %
Epoch: 17, Loss: 0.752, Test accuracy: 79.1 %
Epoch: 18, Loss: 0.384, Test accuracy: 79.3 %
Epoch: 19, Loss: 0.290, Test accuracy: 77.8 %
Epoch: 20, Loss: 0.412, Test accuracy: 76.2 %
Epoch: 21, Loss: 0.573, Test accuracy: 80.7 %
Epoch: 22, Loss: 0.575, Test accuracy: 80.9 %
Epoch: 23, Loss: 0.359, Test accuracy: 81.4 %
Epoch: 24, Loss: 0.479, Test accuracy: 81.5 %
```

```
Epoch: 25, Loss: 0.558, Test accuracy: 81.6 %
Epoch: 26, Loss: 0.333, Test accuracy: 81.6 %
Epoch: 27, Loss: 0.372, Test accuracy: 81.4 %
Epoch: 28, Loss: 0.633, Test accuracy: 81.9 %
Epoch: 29, Loss: 0.590, Test accuracy: 82.3 %
Epoch: 30, Loss: 0.227, Test accuracy: 81.9 %
Total time for training:  253.83259868621826
```

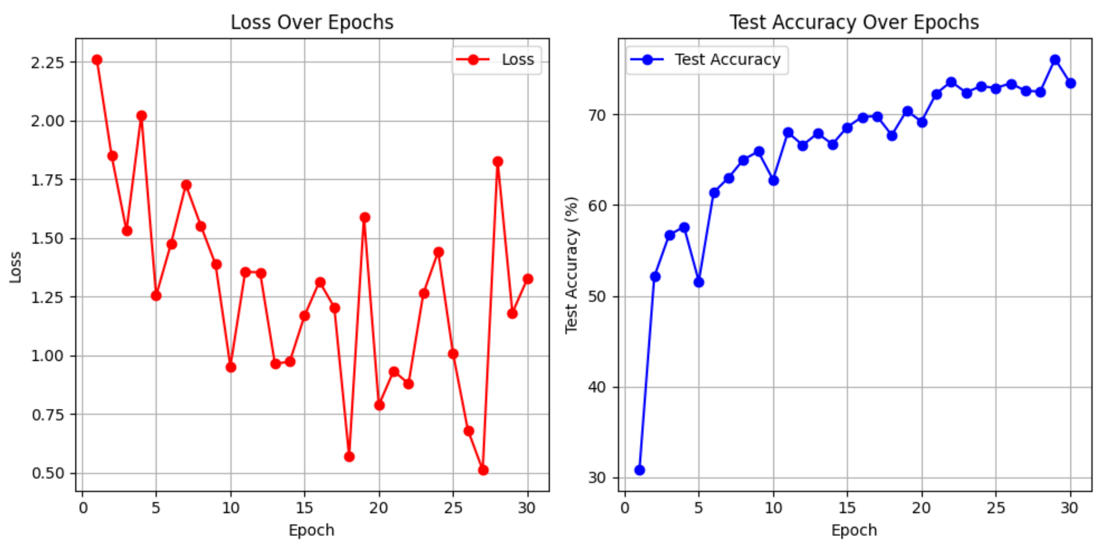
The initial loss at the first epoch is 1.068, and it decreases progressively over time.

The final loss at the 30th epoch is 0.227. The test accuracy improves gradually throughout the training process. The initial test accuracy at the first epoch is 47.1%, and it increases steadily.

The final test accuracy at the 30th epoch is 81.9%.

The final test accuracy of 81.9% indicates that the basic PointNet model without T-Nets performs reasonably well on the ModelNet10_PLY dataset, achieving good classification results across various object categories.

ii. *Test on the ModelNet40 PLY*



visualization of the accuracy and loss over epochs using basic version of PointNet with ModelNet40_PLY

Figure 18:Basic PointNet Test on the ModelNet40 PLY

```
Classes:  {0: 'airplane', 1: 'bathtub', 2: 'bed', 3: 'bench', 4:
'bookshelf', 5: 'bottle', 6: 'bowl', 7: 'car', 8: 'chair', 9: 'cone',
10: 'cup', 11: 'curtain', 12: 'desk', 13: 'door', 14: 'dresser', 15:
'flower_pot', 16: 'glass_box', 17: 'guitar', 18: 'keyboard', 19:
'lamp', 20: 'laptop', 21: 'mantel', 22: 'monitor', 23: 'night_stand',
24: 'person', 25: 'piano', 26: 'plant', 27: 'radio', 28: 'range_hood',
29: 'sink', 30: 'sofa', 31: 'stairs', 32: 'stool', 33: 'table', 34:
'tent', 35: 'toilet', 36: 'tv_stand', 37: 'vase', 38: 'wardrobe', 39:
'xbox'}
Train dataset size:  9843
Test dataset size:  2468
Number of classes:  40
```

```
Sample pointcloud shape:  torch.Size([1024, 3])
Number of parameters in the Neural Networks:  818088
Device:  cuda:0
Epoch: 1, Loss: 2.261, Test accuracy: 30.8 %
Epoch: 2, Loss: 1.849, Test accuracy: 52.1 %
Epoch: 3, Loss: 1.530, Test accuracy: 56.7 %
Epoch: 4, Loss: 2.023, Test accuracy: 57.6 %
Epoch: 5, Loss: 1.254, Test accuracy: 51.6 %
Epoch: 6, Loss: 1.475, Test accuracy: 61.4 %
Epoch: 7, Loss: 1.725, Test accuracy: 63.0 %
Epoch: 8, Loss: 1.553, Test accuracy: 65.0 %
Epoch: 9, Loss: 1.387, Test accuracy: 65.9 %
Epoch: 10, Loss: 0.953, Test accuracy: 62.8 %
Epoch: 11, Loss: 1.355, Test accuracy: 68.0 %
Epoch: 12, Loss: 1.353, Test accuracy: 66.6 %
Epoch: 13, Loss: 0.964, Test accuracy: 67.9 %
Epoch: 14, Loss: 0.973, Test accuracy: 66.7 %
Epoch: 15, Loss: 1.169, Test accuracy: 68.6 %
Epoch: 16, Loss: 1.314, Test accuracy: 69.7 %
Epoch: 17, Loss: 1.204, Test accuracy: 69.8 %
Epoch: 18, Loss: 0.571, Test accuracy: 67.7 %
Epoch: 19, Loss: 1.589, Test accuracy: 70.4 %
Epoch: 20, Loss: 0.788, Test accuracy: 69.2 %
Epoch: 21, Loss: 0.934, Test accuracy: 72.3 %
Epoch: 22, Loss: 0.879, Test accuracy: 73.6 %
Epoch: 23, Loss: 1.263, Test accuracy: 72.4 %
Epoch: 24, Loss: 1.441, Test accuracy: 73.1 %
Epoch: 25, Loss: 1.008, Test accuracy: 72.9 %
Epoch: 26, Loss: 0.682, Test accuracy: 73.4 %
Epoch: 27, Loss: 0.512, Test accuracy: 72.6 %
Epoch: 28, Loss: 1.829, Test accuracy: 72.5 %
Epoch: 29, Loss: 1.181, Test accuracy: 76.1 %
Epoch: 30, Loss: 1.325, Test accuracy: 73.5 %
Total time for training:  662.2685887813568
```

The loss fluctuates across epochs but generally decreases over time, indicating that the model is learning from the training data. The initial loss at the first epoch is 2.261, and it fluctuates throughout training. The final loss at the 30th epoch is 1.325. The test accuracy shows fluctuations during training but exhibits an overall increasing trend.

The initial test accuracy at the first epoch is 30.8%, and it varies throughout training. The final test accuracy at the 30th epoch is 73.5%.

iii. Conclusion

The test accuracy achieved with the basic version of PointNet on the ModelNet10_PLY dataset is considerably higher compared to the PointMLP neural network.

- The training loss decreases over epochs, indicating that the model is learning from the training data. The test accuracy also shows an increasing trend over epochs.

- The gap between the training and test accuracy is relatively small, suggesting that the model is not overfitting significantly to the training data.

- The basic version of PointNet seems to be more effective in capturing the underlying patterns in the point cloud data compared to PointMLP. This could be due to the ability of PointNet to learn hierarchical features directly from the raw point cloud data.

b) PointNet with T-Net network:

The T-Net, or Transformation Network, is a vital part of PointNet. It's designed to align input point clouds into a standard orientation, making it easier for PointNet to understand and extract important features. This alignment boosts the model's performance by ensuring consistent processing of point clouds, regardless of their initial orientation or position. The T-Net typically consists of a series of fully connected layers (MLPs) that predict transformation matrices. These matrices are applied to the input point clouds to perform the desired transformations.

Firstly, the shared MLP phase extracts a feature vector from the original point cloud, which serves as the basis for predicting the 3 x 3 matrix. Following that, the max pooling operation aggregates information from all points within the point cloud, while the subsequent global MLP phase generates the final matrix responsible for aligning the data. By integrating the T-Net into PointNet, our objective is to decrease the variability in input data and enhance the overall performance of the network.

i. Test on the ModelNet40 PLY

The loss decreases gradually across epochs, indicating effective learning from the training data. The initial loss at the first epoch is 2.340, and it decreases progressively over time. The final loss at the 30th epoch is 0.889.

The test accuracy improves steadily throughout the training process. The initial test accuracy at the first epoch is 45.4%, and it increases steadily. The final test accuracy at the 30th epoch is 83.5%.

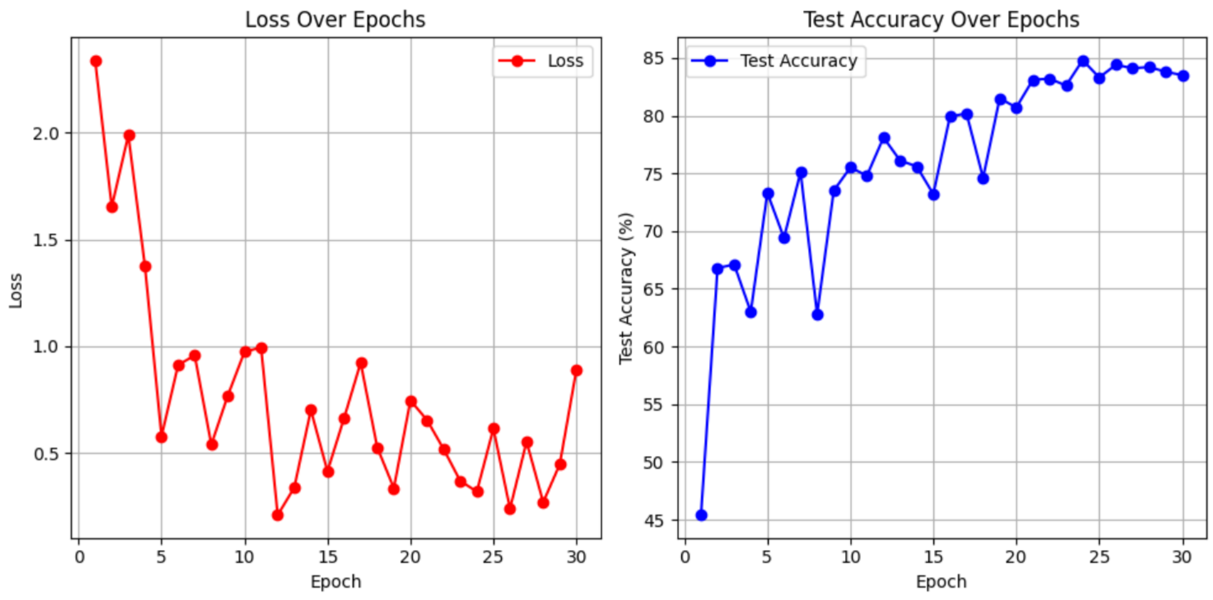


Figure 19 : PointNetFull Test on the ModelNet40 PLY

The final test accuracy of 83.5% indicates that the PointNetFull model, with the integration of the T-Net, performs exceptionally well on the ModelNet40_PLY dataset, achieving high classification accuracy across various object categories.

ii. Test on the ModelNet10 PLY

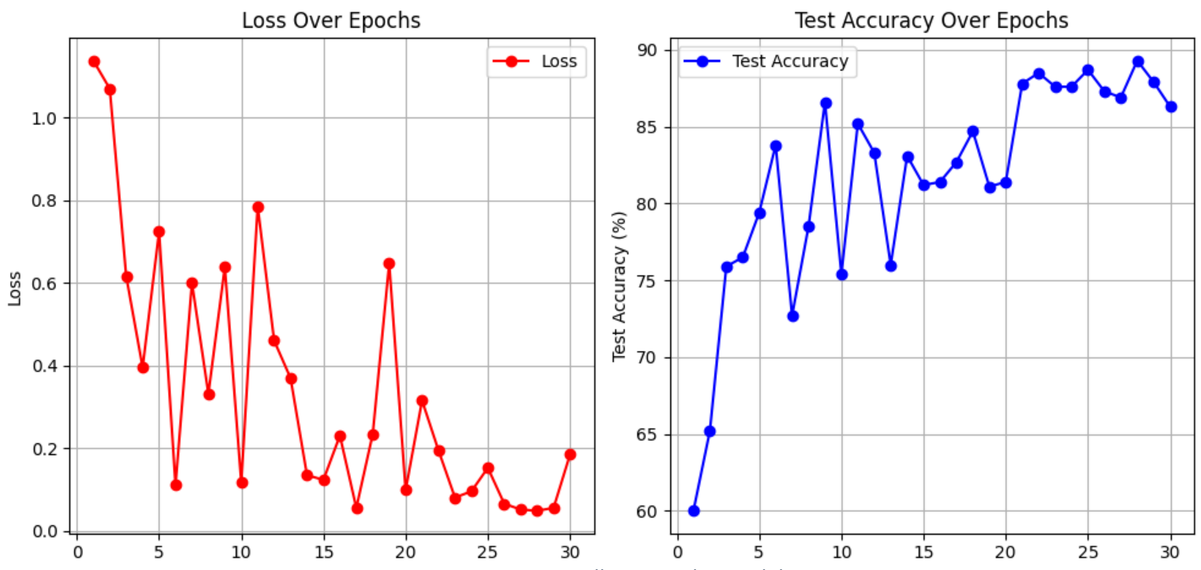


Figure 20: PointNetFull Test on the ModelNet10 PLY

iii. Conclusion

The test accuracy achieved with PointNet adding the 3x3 T-Net on ModelNet40_PLY is notably higher compared to the basic version of PointNet.

- The addition of the 3x3 T-Net module seems to have improved the model's ability to capture and learn complex features from the point cloud data.
- The training loss decreases over epochs, indicating that the model is effectively learning from the training data. The test accuracy also shows an increasing trend over epochs.
- The gap between the training and test accuracy remains relatively small, suggesting that the model is not overfitting significantly to the training data.

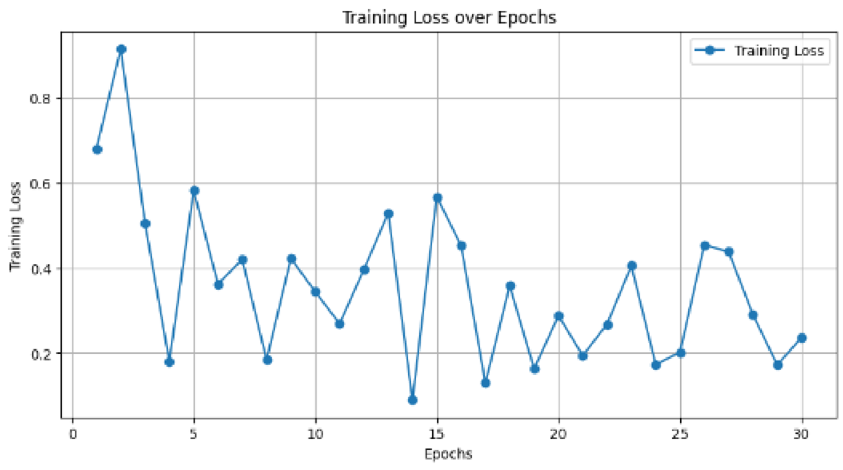
Overall, the performance enhancement achieved by incorporating the 3x3 T-Net module demonstrates the importance of leveraging more sophisticated architectures to improve the performance of PointNet-based models on 3D point cloud classification tasks.

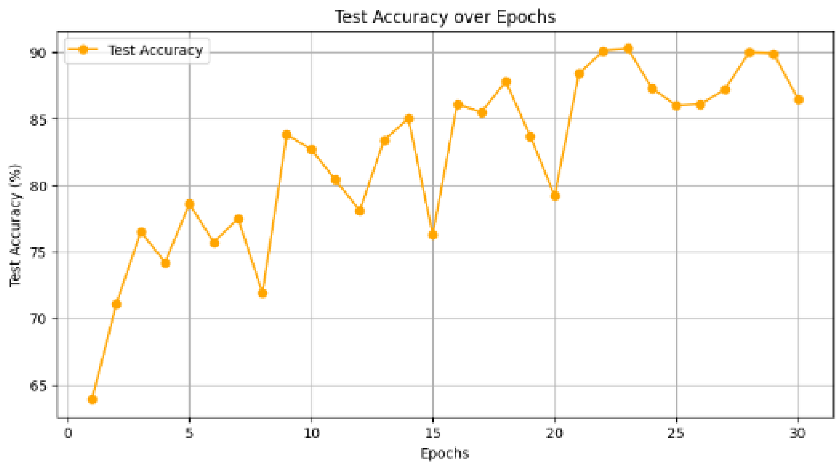
c) Data augmentation for 3D data:

In our code we have 3 methods of data augmentation, RandomRotation_z and Random Noise and Shuffle Points. The purpose of this exercise is to find a new data augmentation on 3D point clouds.

As a new data augmentation on 3D point clouds, we can use **Random Scaling** involves randomly scaling the entire point cloud along each axis independently. This augmentation technique introduces variations in the scale of the object represented by the point cloud, mimicking changes in distance or size perception.
To implement this method, we have 2 steps:

- Select Scaling Factors: Randomly choose scaling factors for each axis (x, y, z) independently. These factors can be within a predefined range, such as [0.8, 1.2], to ensure reasonable variations in scale.
- Apply Scaling: Multiply each coordinate of every point in the point cloud by the corresponding scaling factor along each axis. This effectively scales the entire point cloud in 3D space.





```
Classes: {0: 'bathtub', 1: 'bed', 2: 'chair', 3: 'desk', 4: 'dresser', 5:
'monitor', 6: 'night_stand', 7: 'sofa', 8: 'table', 9: 'toilet'}
Train dataset size:  3991
Test dataset size:  908
Number of classes:  10
Number of parameters in the Neural Networks:  1606419
Device:  cuda:0
Epoch: 1, Loss: 0.680, Test accuracy: 63.9 %
Epoch: 2, Loss: 0.914, Test accuracy: 71.1 %
Epoch: 3, Loss: 0.505, Test accuracy: 76.5 %
Epoch: 4, Loss: 0.180, Test accuracy: 74.2 %
Epoch: 5, Loss: 0.582, Test accuracy: 78.6 %
Epoch: 6, Loss: 0.362, Test accuracy: 75.7 %
Epoch: 7, Loss: 0.420, Test accuracy: 77.5 %
Epoch: 8, Loss: 0.184, Test accuracy: 71.9 %
Epoch: 9, Loss: 0.422, Test accuracy: 83.8 %
Epoch: 10, Loss: 0.344, Test accuracy: 82.7 %
Epoch: 11, Loss: 0.269, Test accuracy: 80.4 %
Epoch: 12, Loss: 0.396, Test accuracy: 78.1 %
Epoch: 13, Loss: 0.529, Test accuracy: 83.4 %
Epoch: 14, Loss: 0.090, Test accuracy: 85.0 %
Epoch: 15, Loss: 0.567, Test accuracy: 76.3 %
Epoch: 16, Loss: 0.452, Test accuracy: 86.1 %
Epoch: 17, Loss: 0.130, Test accuracy: 85.5 %
Epoch: 18, Loss: 0.358, Test accuracy: 87.8 %
```

```
Epoch: 19, Loss: 0.163, Test accuracy: 83.7 %
...
Epoch: 28, Loss: 0.290, Test accuracy: 90.0 %
Epoch: 29, Loss: 0.172, Test accuracy: 89.9 %
Epoch: 30, Loss: 0.237, Test accuracy: 86.5 %
Total time for training:  289.374400138855
```

Observation:

Without transformation: Final test accuracy: **86.3%**

With transformation: Final test accuracy: **86.5%**

➔ It seems that in this specific case, the data augmentation did not significantly improve the test accuracy of the PointNet model on the ModelNet10_PLY dataset. Both with and without transformations, the model achieved similar test accuracies. However, it's essential to note that data augmentation can have a more significant impact on performance in other datasets or models.

Additionally, when considering the mean test accuracy:

Mean test accuracy **without transformation**: approximately 82.57%

Mean test accuracy **with transformation**: approximately 83.03%

➔ Therefore, when looking at the average performance across multiple runs, we observe a slight improvement of about 0.46% when using data augmentation. However, this improvement is relatively small and may not be statistically significant.

5) Conclusion

In conclusion, this project has provided valuable insights into the design, implementation, and evaluation of PointMLP and PointNet architectures for processing 3D point cloud data. By exploring various architectural variations and techniques such as T-Net integration and data augmentation, we have gained a deeper understanding of their impact on model performance and learned effective strategies for improving the robustness and generalization capabilities of deep learning models for 3D object recognition. By testing on datasets like ModelNet10 and ModelNet40, we've evaluated the scalability and applicability of PointMLP and PointNet architectures across varying task complexities. This process has reinforced our understanding of their strengths and limitations in real-world applications.

UNIVERSITÉ
Clermont Auvergne