1 Deliverable 1

Solution: Code submitted on gradescope as a zip file. □

2 Deliverable 2: Linear separable dataset

Solution:

Figure 1: Linear separable dataset (train, test set with 200 points each)

```
dim_in, dim_out = x_train.shape[1], 2
hidden_neuron_list = [1]
activation_list = ['ReLU', 'Sigmoid']
opt_init = None
opt_loss = L2Loss()
mlp = MLP(dim_in, dim_out, hidden_neuron_list, activation_list, opt_init)
opt_optim = SGD(mlp)
----------------------------------------

Model Summary
-------------
Layer 1: Linear - A Dim: 2, Output Dim: 1, Parameters: 3
Layer 2: ReLU
Layer 3: Linear - A Dim: 1, Output Dim: 2, Parameters: 4
Layer 4: Sigmoid
Total Parameters: 7
```

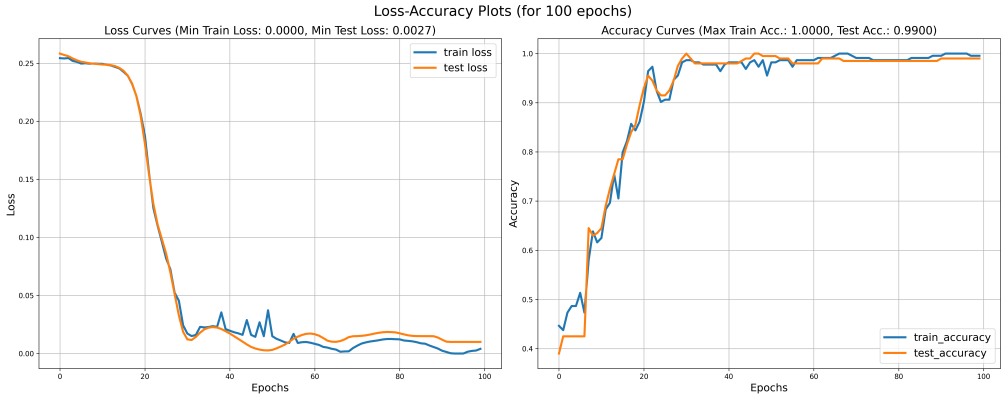

Figure 2: Loss and accuracy for linear separable dataset (train, test set with 200 points each)

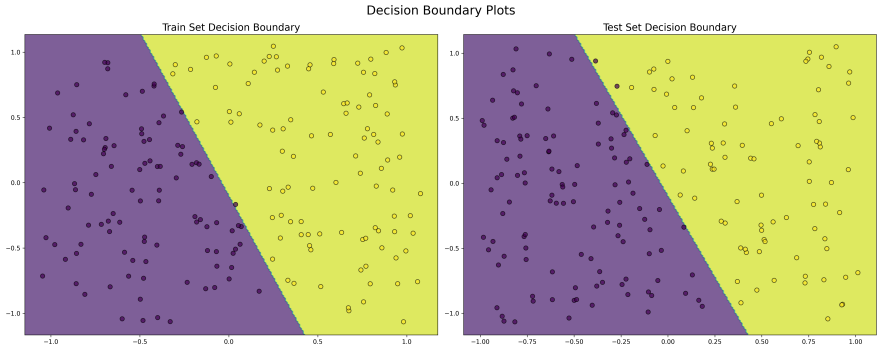

Figure 3: Decision boundary for linear separable dataset (train, test set with 200 points each)

□

3 Deliverable 3: XOR problem

Solution:

Figure 4: XOR dataset (train, test set with 200 points each)

```
# one hot encoding the output to get the final output (dim 1 for prediction)

dim_in, dim_out = x_train.shape[1], 2
hidden_neuron_list = [4,16]
activation_list = ['ReLU', 'ReLU','Sigmoid']
opt_init = 'xavier'
opt_loss = L2Loss()
mlp = MLP(dim_in, dim_out, hidden_neuron_list, activation_list, opt_init)
opt_optim = Adam(mlp)
print(mlp.summary())
----------------------------------------

Model Summary
-------------
Layer 1: Linear - A Dim: 2, Output Dim: 4, Parameters: 12
Layer 2: ReLU
Layer 3: Linear - A Dim: 4, Output Dim: 16, Parameters: 80
Layer 4: ReLU
Layer 5: Linear - A Dim: 16, Output Dim: 2, Parameters: 34
Layer 6: Sigmoid
Total Parameters: 126
```

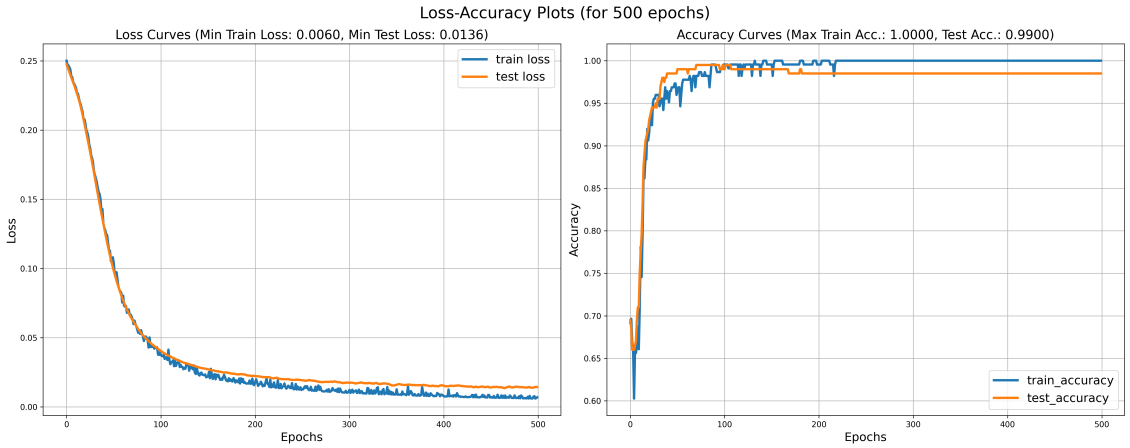

Figure 5: Loss and accuracy for XOR dataset (train, test set with 200 points each), Adam optimizer, 500 epochs, Cost function: L2Loss, Xaiver initialization

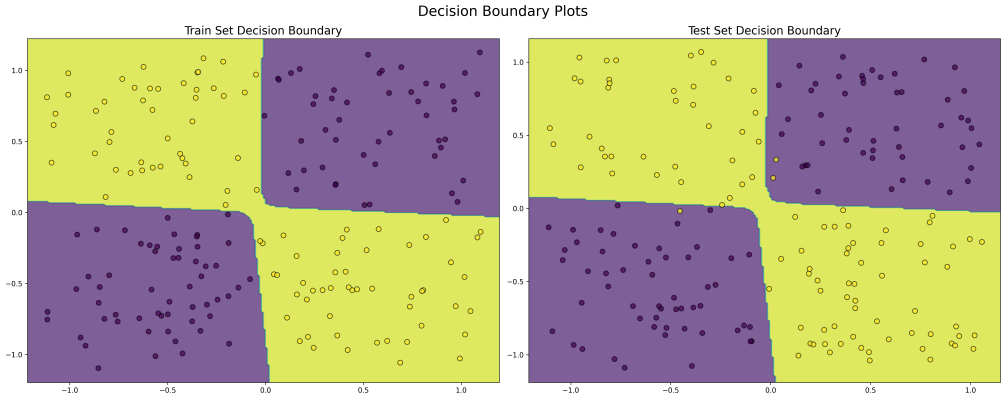

Figure 6: Decision boundary for XOR separable dataset (train, test set with 200 points each)

□

4 Deliverable 4: Differences in cost function (Circle dataset)

Solution:

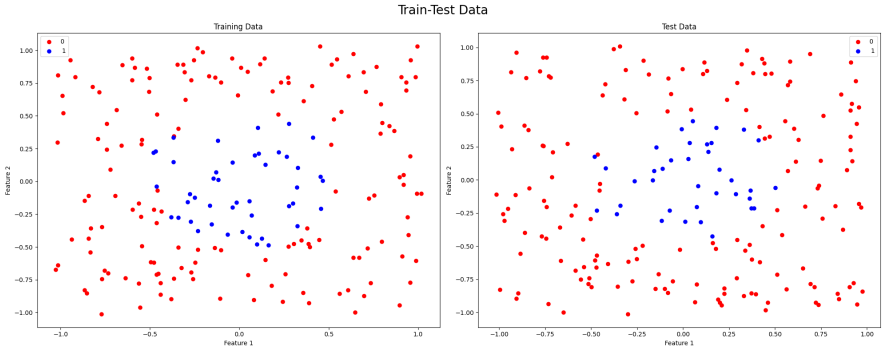

Figure 7: Circle dataset (train, test set with 200 points each)

Key Differences Noted

- The decision boundary for the L2Loss is less smooth compared to the CrossEntropyLoss. This is because the L2Loss is a regression loss and the CrossEntropyLoss is a classification loss.
- Related to the above point, the L2Loss decision boundary seems more oval than the one from CrossEntropyLoss, which is more circular (hence closer to the actual decision boundary).
- Final Accuracy and loss are the same, however, we see the L2Loss has a more jittery loss graph compared to the CrossEntropyLoss, evident in the spikes in the loss graph.

4.1 L2Loss

```
# one hot encoding the output to get the final output (dim 1 for prediction)
dim_in, dim_out = x_train.shape[1], 2
hidden_neuron_list = [4,16]
# ReLU was giving similar accuracy but more boxy decision boundary
activation_list = ['Tanh', 'Tanh','Sigmoid']
opt_init = 'xavier'
opt_loss = L2Loss()
mlp = MLP(dim_in, dim_out, hidden_neuron_list, activation_list, opt_init)
opt_optim = Adam(mlp)
print(mlp.summary())
----------------------------------------------------------------
Model Summary
-------------
Layer 1: Linear - A Dim: 2, Output Dim: 4, Parameters: 12
Layer 2: Tanh
Layer 3: Linear - A Dim: 4, Output Dim: 16, Parameters: 80
Layer 4: Tanh
Layer 5: Linear - A Dim: 16, Output Dim: 2, Parameters: 34
Layer 6: Sigmoid
Total Parameters: 126
```

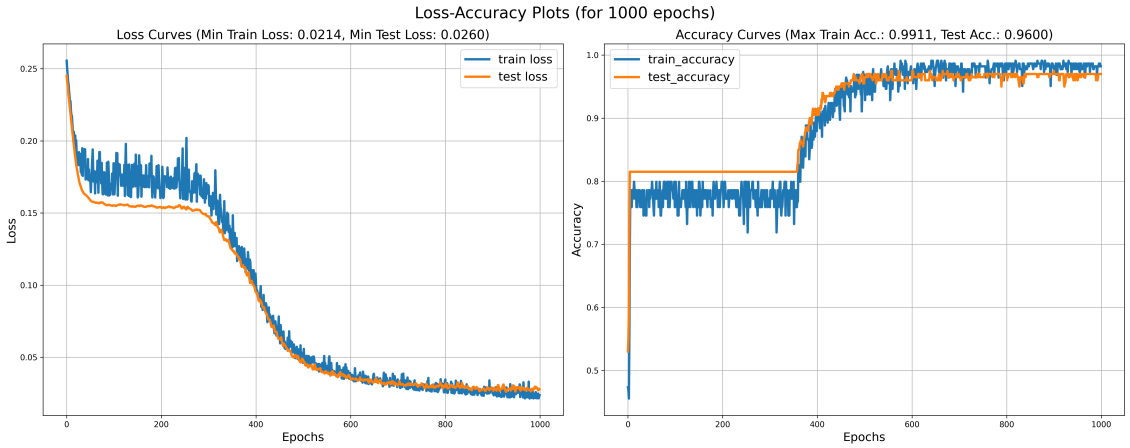

Figure 8: Loss and accuracy for Circle dataset (train, test set with 200 points each), Adam optimizer, 500 epochs, Cost function: L2Loss, Xaiver initialization

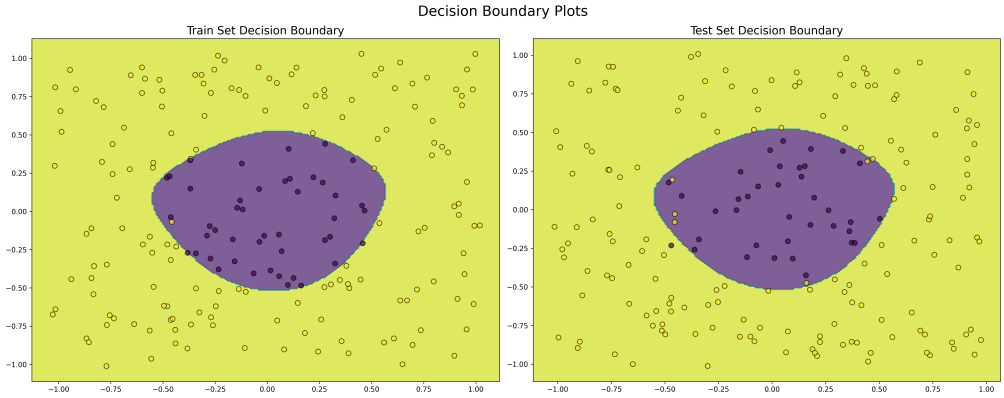

Figure 9: (L2Loss) Decision boundary for Circle separable dataset (train, test set with 200 points each)

4.2 CrossEntropyLoss

```
    dim_in, dim_out = x_train.shape[1], 2
hidden_neuron_list = [4,16]
# LinearActivation used in the last layer because sigmoiding in the loss function forwa

activation_list = ['Tanh', 'Tanh','LinearActivation']
opt_init = 'xavier'
opt_loss = CrossEntropyLoss()
mlp = MLP(dim_in, dim_out, hidden_neuron_list, activation_list, opt_init)
opt_optim = Adam(mlp)
print(mlp.summary())
----------------------------------------------------------------
Model Summary
-------------
Layer 1: Linear - A Dim: 2, Output Dim: 4, Parameters: 12
```

```
Layer 2: Tanh
Layer 3: Linear - A Dim: 4, Output Dim: 16, Parameters: 80
Layer 4: Tanh
Layer 5: Linear - A Dim: 16, Output Dim: 2, Parameters: 34
Layer 6: LinearActivation
Total Parameters: 126
```

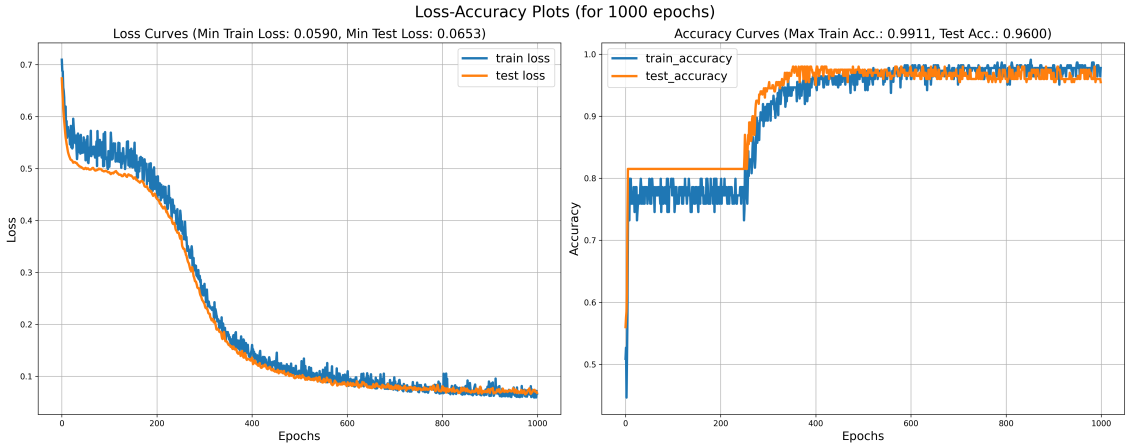

Figure 10: Loss and accuracy for Circle dataset (train, test set with 200 points each), Adam optimizer, 500 epochs, Cost function: CrossEntropyLoss, Xaiver initialization

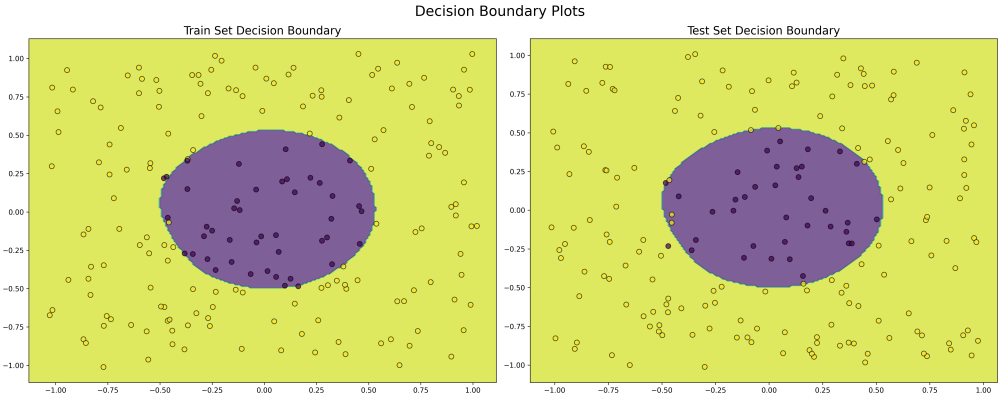

Figure 11: (CrossEntropyLoss) Decision boundary for Circle separable dataset (train, test set with 200 points each)

□

5 Deliverable 5: Differences in Optimizers (Sinusoidal dataset)

Solution:

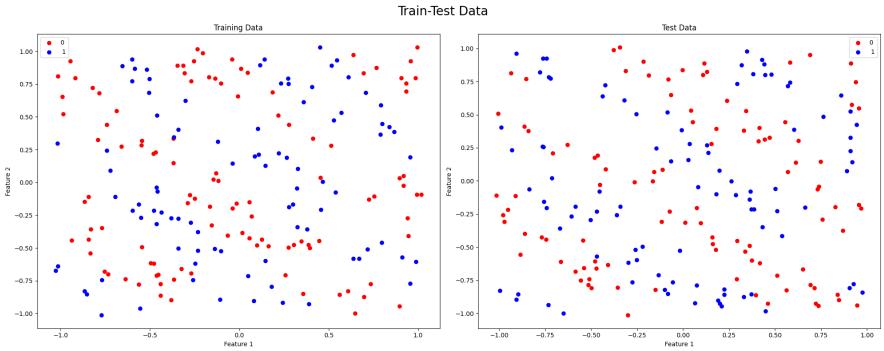

Figure 12: Sinusoidal dataset (train, test set with 200 points each)

Model Description

```
dim_in, dim_out = x_train.shape[1], 2
hidden_neuron_list = [8, 16, 32, 8]
activation_list = ['ReLU','ReLU', 'ReLU','ReLU','LinearActivation']
# linear at the last layer because sigmoiding in the loss function forward pass for Cro
opt_init = 'xavier'
opt_loss = CrossEntropyLoss()
mlp = MLP(dim_in, dim_out, hidden_neuron_list, activation_list, opt_init)
print(mlp.summary())
----------------------------------------------------------------
Model Summary
-------------
Layer 1: Linear - A Dim: 2, Output Dim: 8, Parameters: 24
Layer 2: ReLU
Layer 3: Linear - A Dim: 8, Output Dim: 16, Parameters: 144
Layer 4: ReLU
Layer 5: Linear - A Dim: 16, Output Dim: 32, Parameters: 544
Layer 6: ReLU
Layer 7: Linear - A Dim: 32, Output Dim: 8, Parameters: 264
Layer 8: ReLU
Layer 9: Linear - A Dim: 8, Output Dim: 2, Parameters: 18
Layer 10: LinearActivation
Total Parameters: 994
```

5.1 Optimizer Performance

We see that the Adam optimizer with learning rate 0.01 performs the best in terms of loss and accuracy. The decision boundary is also the most accurate. The performacne for the SGD optimizer with learning rate 0.01 and momentum is the second best (with lesser train accuracy but simlar test performance). We do see that tho the decision boundary is over fitting to train in this case.

Adam optimizer with learning rate 0.001 is lesser accurate, followed by vanilla SGD which gives the worst performance in terms of loss and accuracy. The decision boundary is also the most inaccurate.

5.2 MLP with SGD Optimizer of learning rate 0.01 and default parameters (no momentum, no weight decay)

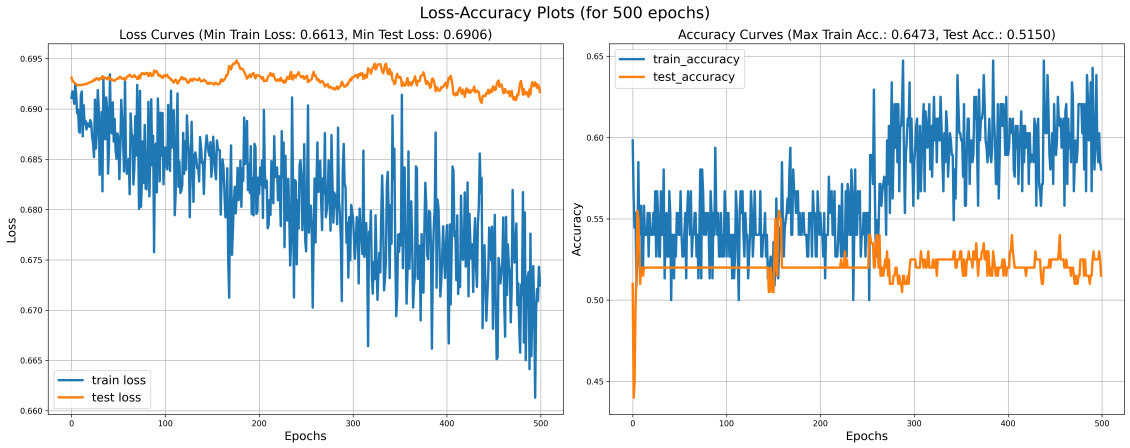

Figure 13: Loss and accuracy for Circle dataset (train, test set with 200 points each) SGD optimizer ($\text{lr} = 1e-2$), 400 epochs, Cost function: CrossEntropyLoss, Xaiver initialization

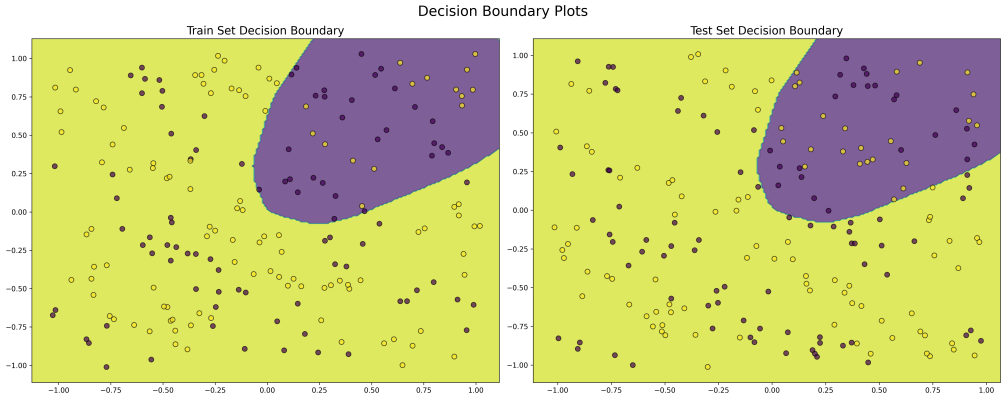

Figure 14: (L2Loss) Decision boundary for Circle separable dataset (train, test set with 200 points each) SGD ($\text{lr} = 1e-2$)

5.3 MLP with SGD Optimizer of learning rate 0.01 momentum 0.9, no weight decay

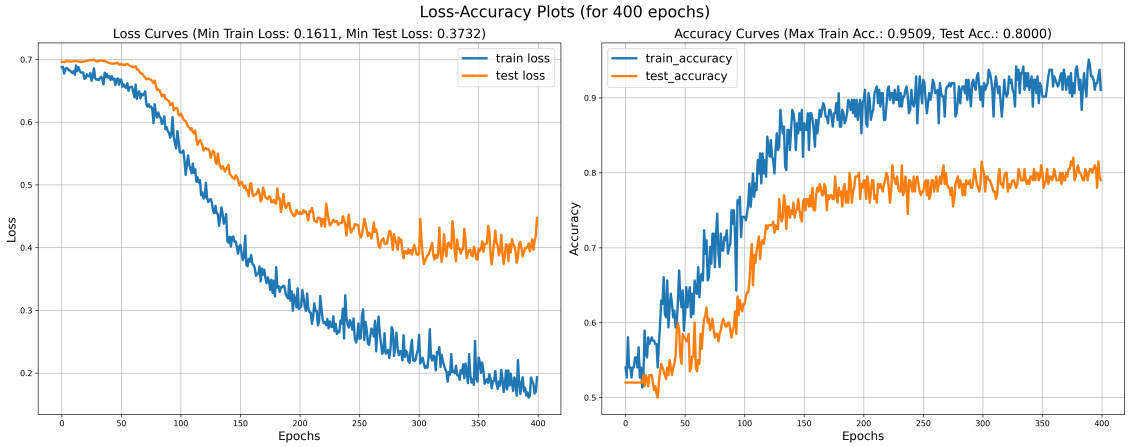

Figure 15: Loss and accuracy for Circle dataset (train, test set with 200 points each) Adam optimizer (lr $= 1e-3$), 400 epochs, Cost function: CrossEntropyLoss, Xaiver initialization

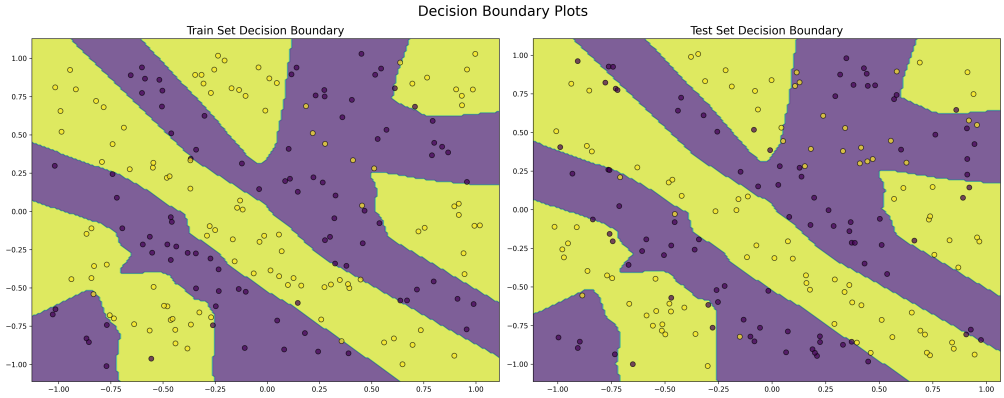

Figure 16: (L2Loss) Decision boundary for Circle separable dataset (train, test set with 200 points each) Adam optimizer (lr $= 1e-3$)

5.4 MLP with Adam Optimizer of learning rate 0.01 and default parameters

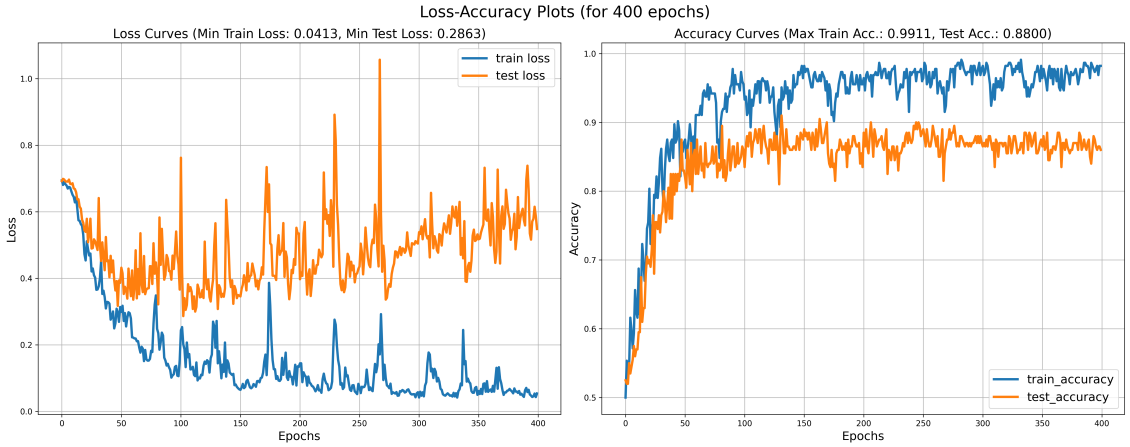

Figure 17: Loss and accuracy for Circle dataset (train, test set with 200 points each) Adam optimizer (lr $= 1e-2$), 400 epochs, Cost function: CrossEntropyLoss, Xaiver initialization

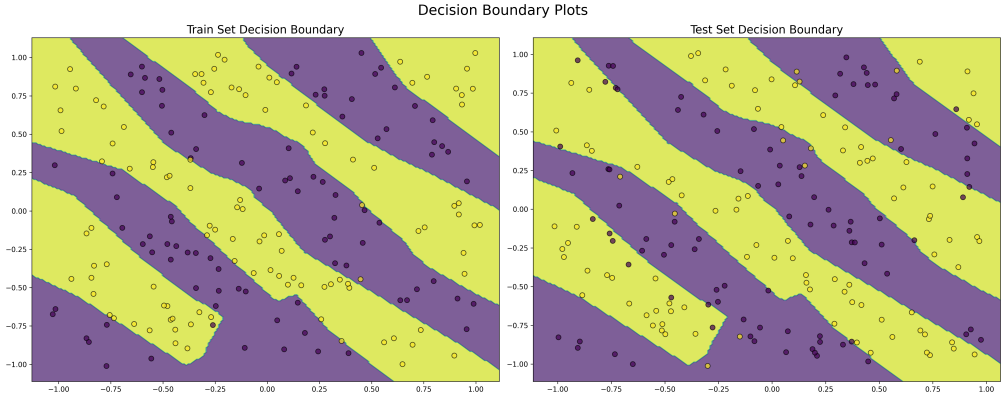

Figure 18: (L2Loss) Decision boundary for Circle separable dataset (train, test set with 200 points each) Adam optimizer (lr $= 1e-2$)

5.5 MLP with Adam Optimizer of learning rate 0.001 and default parameters

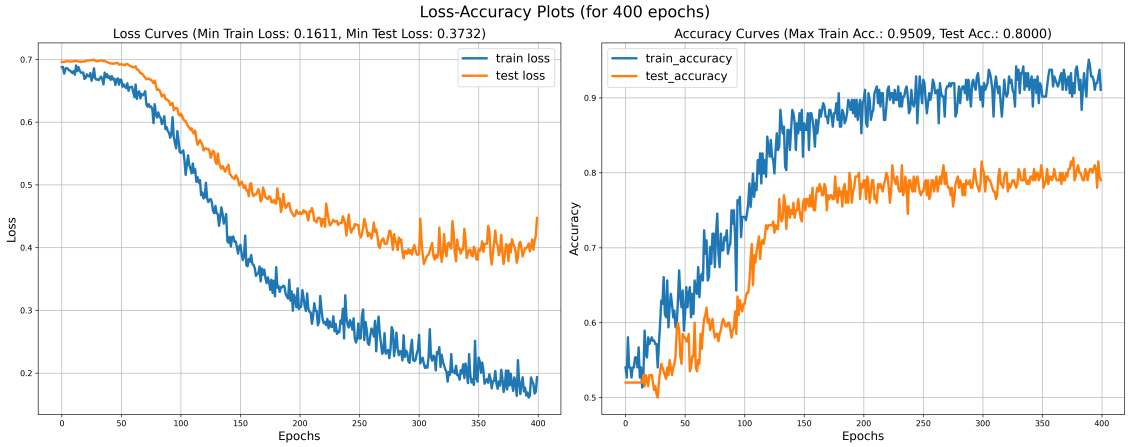

Figure 19: Loss and accuracy for Circle dataset (train, test set with 200 points each) Adam optimizer ($\text{lr} = 1e-3$), 400 epochs, Cost function: CrossEntropyLoss, Xaiver initialization

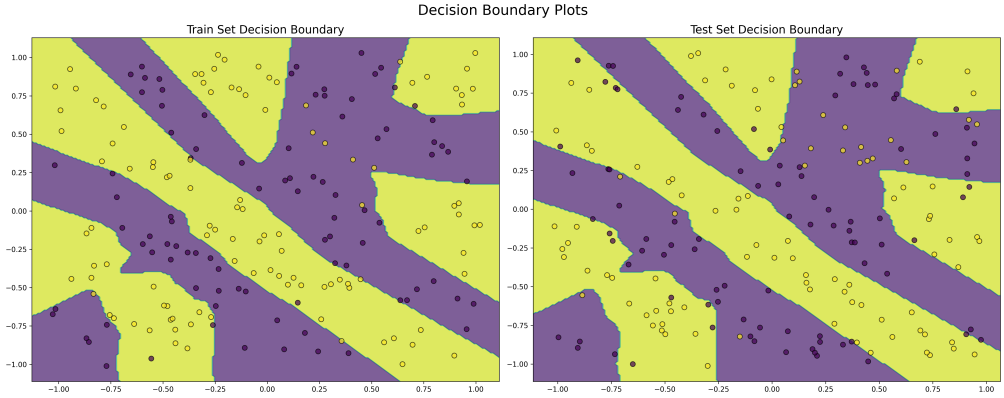

Figure 20: (L2Loss) Decision boundary for Circle separable dataset (train, test set with 200 points each) Adam optimizer ($\text{lr} = 1e-3$)

□

6 Deliverable 6: Swiss roll

Solution:

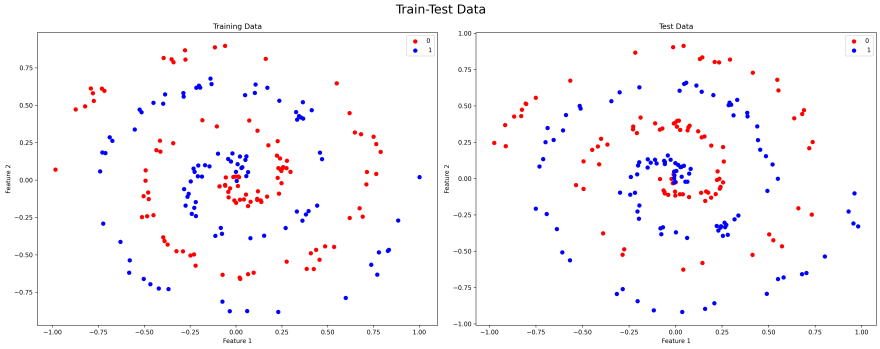

Figure 21: Swiss-Roll dataset (train, test set with 200 points each)

```
dim_in, dim_out = x_train.shape[1], 2
hidden_neuron_list = [8, 16, 64, 12]
activation_list = ['ReLU','ReLU', 'ReLU','ReLU','LinearActivation']
opt_init = 'xavier'
opt_loss = CrossEntropyLoss()
mlp = MLP(dim_in, dim_out, hidden_neuron_list, activation_list, opt_init)
opt_optim = Adam(mlp, learning_rate=0.01)
print(mlp.summary())
```

```
----------------------------------------------------------------
Model Summary
-------------
Layer 1: Linear - A Dim: 2, Output Dim: 8, Parameters: 24
Layer 2: ReLU
Layer 3: Linear - A Dim: 8, Output Dim: 16, Parameters: 144
Layer 4: ReLU
Layer 5: Linear - A Dim: 16, Output Dim: 64, Parameters: 1088
Layer 6: ReLU
Layer 7: Linear - A Dim: 64, Output Dim: 12, Parameters: 780
Layer 8: ReLU
Layer 9: Linear - A Dim: 12, Output Dim: 2, Parameters: 26
Layer 10: LinearActivation
Total Parameters: 2062
```

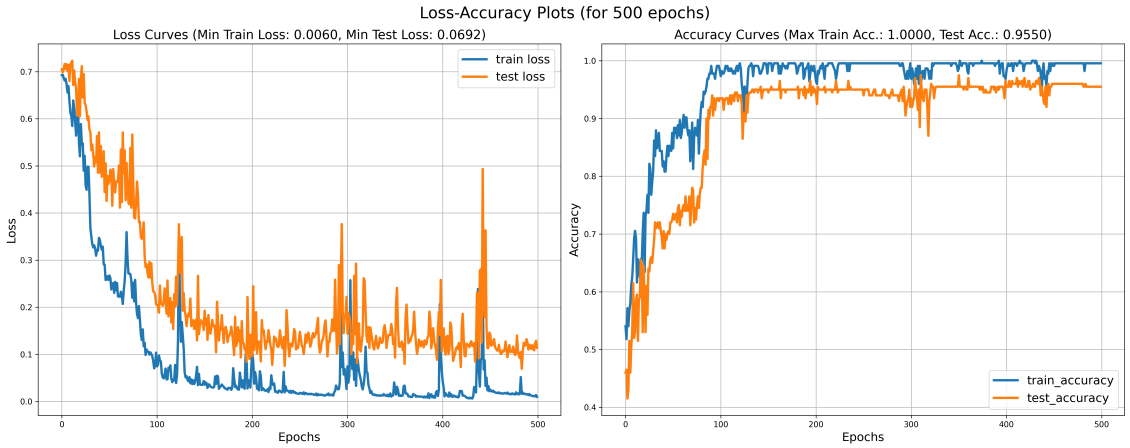

Figure 22: Loss and accuracy for Swiss-roll dataset (train, test set with 200 points each), Adam optimizer (lr = .01), 500 epochs, Cost function: CrossEntropyLoss, Xaiver initialization

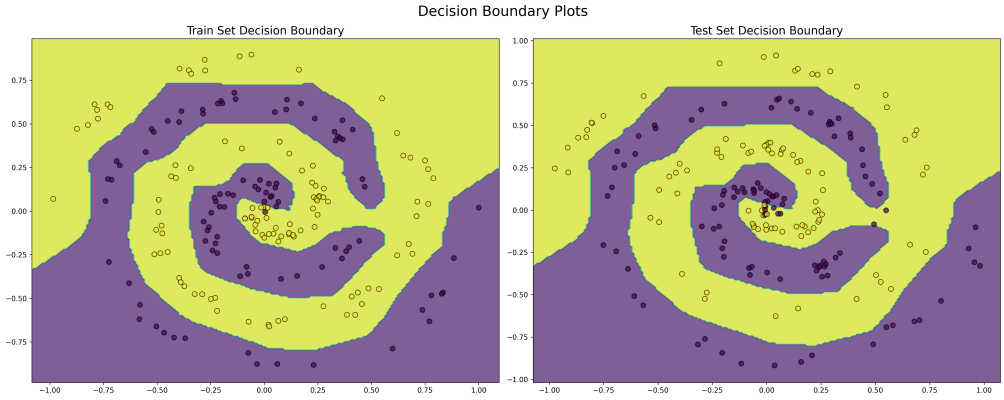

Figure 23: (L2Loss) Decision boundary for Swiss-Roll separable dataset (train, test set with 200 points each)

□

7 Deliverable 6: Non-linear embeddings

7.1 Circular Data, Non-Linear Embedding: $x^2 + y^2$

Solution:

MLP used for linearly separable data after adding non linear feature

```
dim_in, dim_out = 3, 2
hidden_neuron_list = [1]
activation_list = ['ReLU', 'Sigmoid']
opt_init = 'xavier'
opt_loss = L2Loss()
mlp = MLP(dim_in, dim_out, hidden_neuron_list, activation_list, opt_init)
opt_optim = Adam(mlp)
print(mlp.summary())
```

```
----------------------------------------------------------------
Model Summary
-------------
Layer 1: Linear - A Dim: 3, Output Dim: 1, Parameters: 4
Layer 2: ReLU
Layer 3: Linear - A Dim: 1, Output Dim: 2, Parameters: 4
Layer 4: Sigmoid
Total Parameters: 8
```

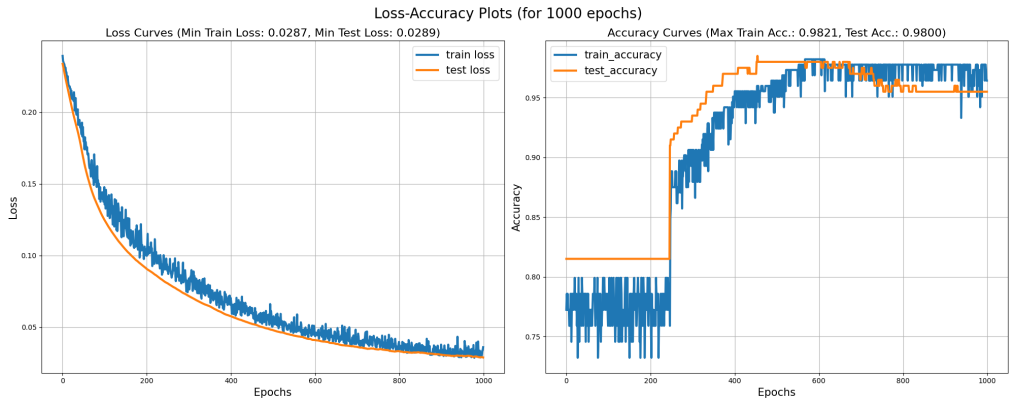

Figure 25: Loss Accuracy plot for circular data after adding $x^2 + y^2$ feature, with MLP of 1 hidden neuron

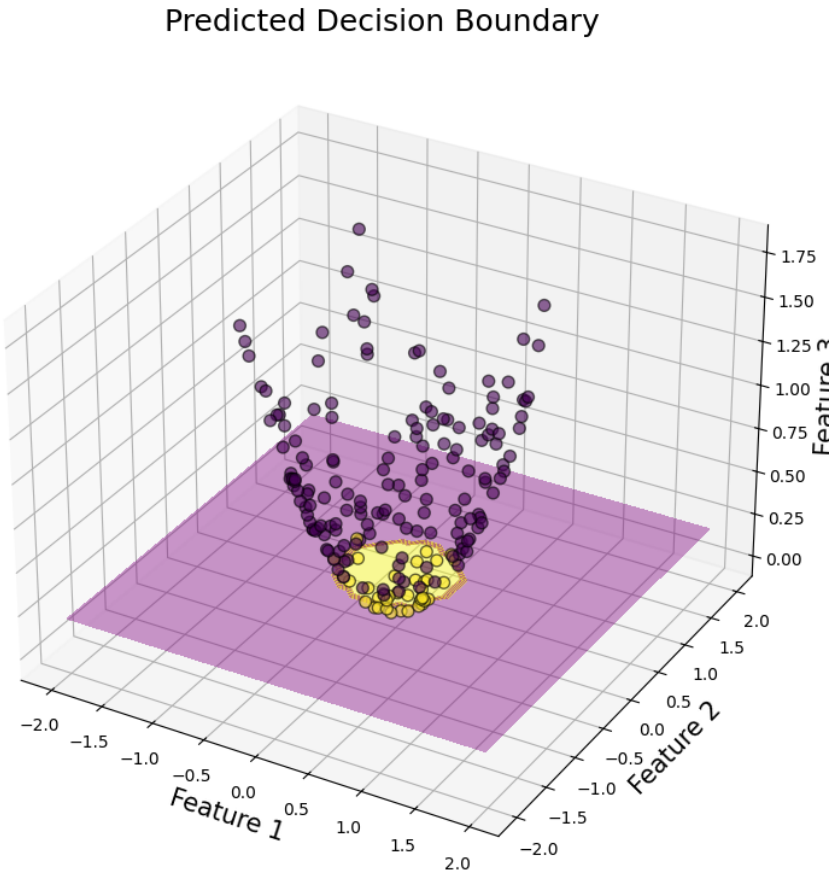

Figure 26: Predicted Decision boundary for circular data after adding $x^2 + y^2$ feature (x: feature 1, y: feature 2, $x^2 + y^2$: feature 3). We see that the decision boundary after adding the non-linear feature is a hyperplane across the purple and yellow points.

□

7.2 XOR Data, Non-Linear Embedding: $x \times y$

Solution: Same model as above (that was used for circular data) is used for XOR data after adding non-linear feature $x \times y$.

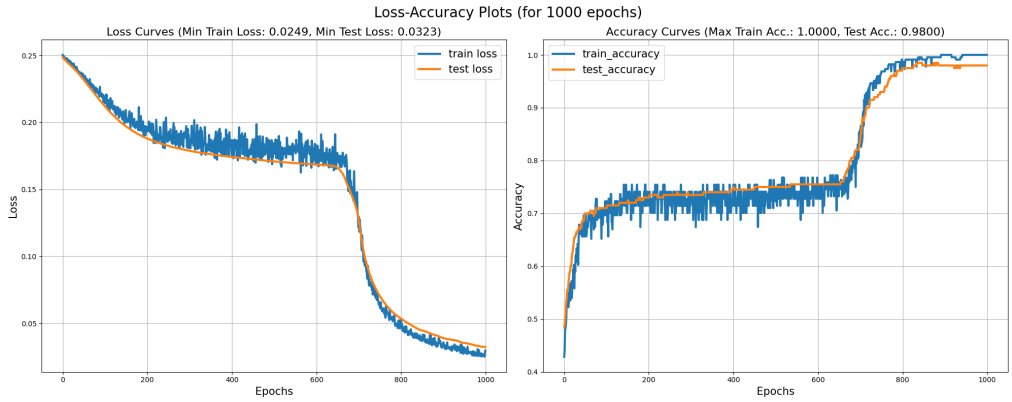

Figure 28: Loss Accuracy plot for circular data after adding $x^2 + y^2$ feature, with MLP of 1 hidden neuron

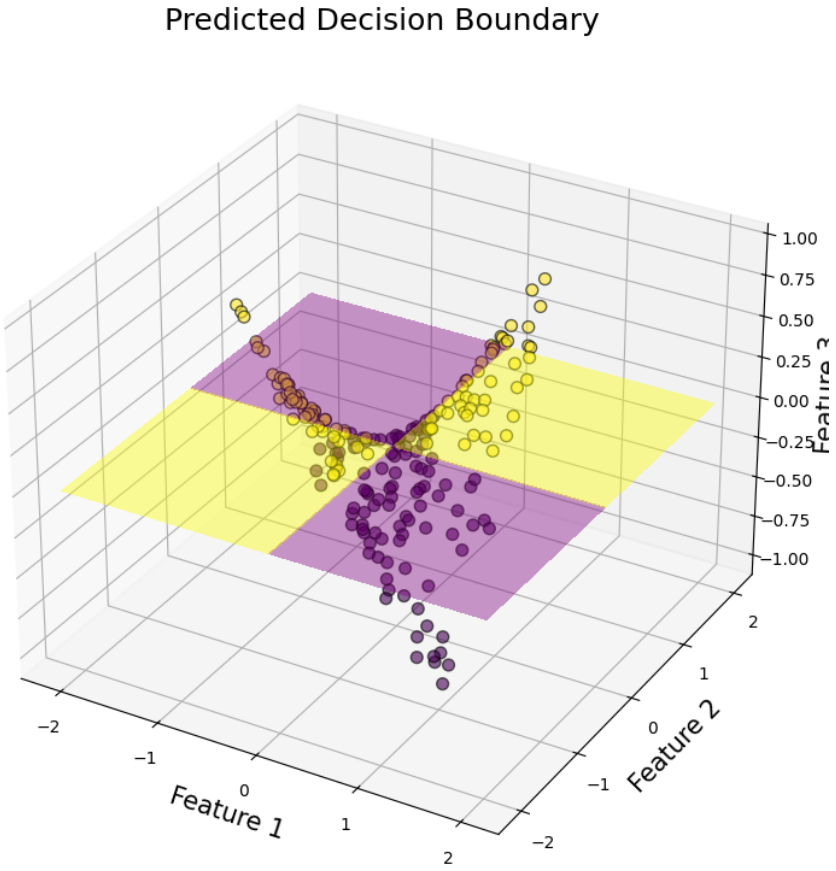

Figure 29: Predicted Decision boundary for xor data after adding $x \times y$ feature (x: feature 1, y: feature 2, $x \times y$: feature 3)

□

7.3 Swiss-Roll Data, Non-Linear Embedding: $x^2, x \times y, y^2$

Solution:

MLP used for swiss-roll data after adding non linear features

```
dim_in, dim_out = 5, 2
hidden_neuron_list = [8,4]
activation_list = ['ReLU','ReLU','Tanh']
opt_init = 'xavier'
opt_loss = L2Loss()
mlp = MLP(dim_in, dim_out, hidden_neuron_list, activation_list, opt_init)
opt_optim = Adam(mlp, learning_rate=.001)
print(mlp.summary())
```

```
----------------------------------------------------------------
Model Summary
-------------
Layer 1: Linear - A Dim: 5, Output Dim: 8, Parameters: 48
Layer 2: ReLU
Layer 3: Linear - A Dim: 8, Output Dim: 4, Parameters: 36
Layer 4: ReLU
Layer 5: Linear - A Dim: 4, Output Dim: 2, Parameters: 10
Layer 6: Tanh
Total Parameters: 94
```

We see that we are able to get accuracy of around 94% on the swiss-roll test-data after adding non-linear features $x^2, x \times y, y^2$ for a MLP with only 2 hidden layers of 8 and 4 neurons each. This is a much bigger improvement over running the similar architecture on just the 2D data, without adding the non-linear features.

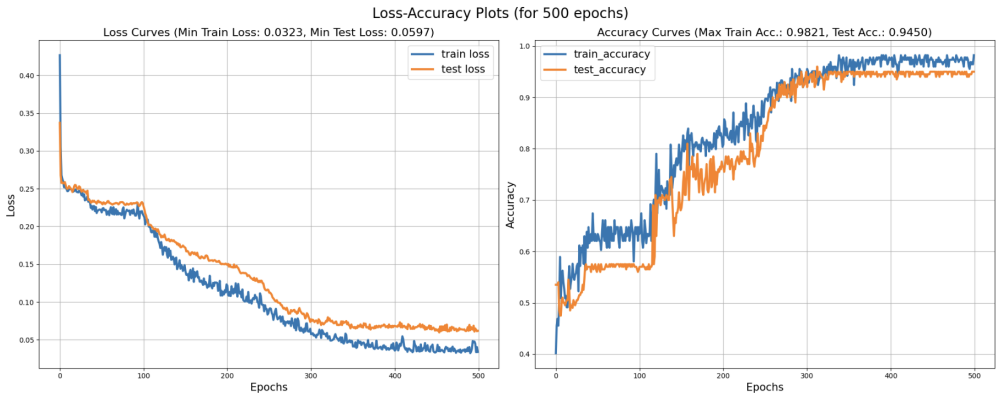

Figure 30: Loss Accuracy plot for swiss-roll data after adding $x^2, x \times y, y^2$ feature

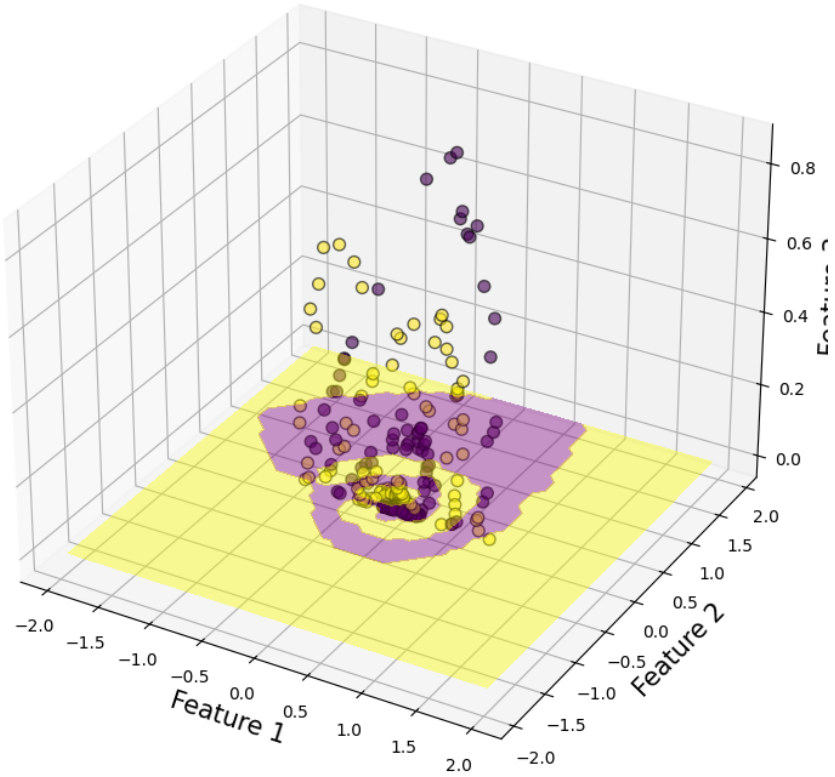

Figure 31: Predicted Decision boundary for swiss-roll data after adding $x^2 + y^2$ feature (x: feature 1, y: feature 2, $x \times y$: feature 3)

□