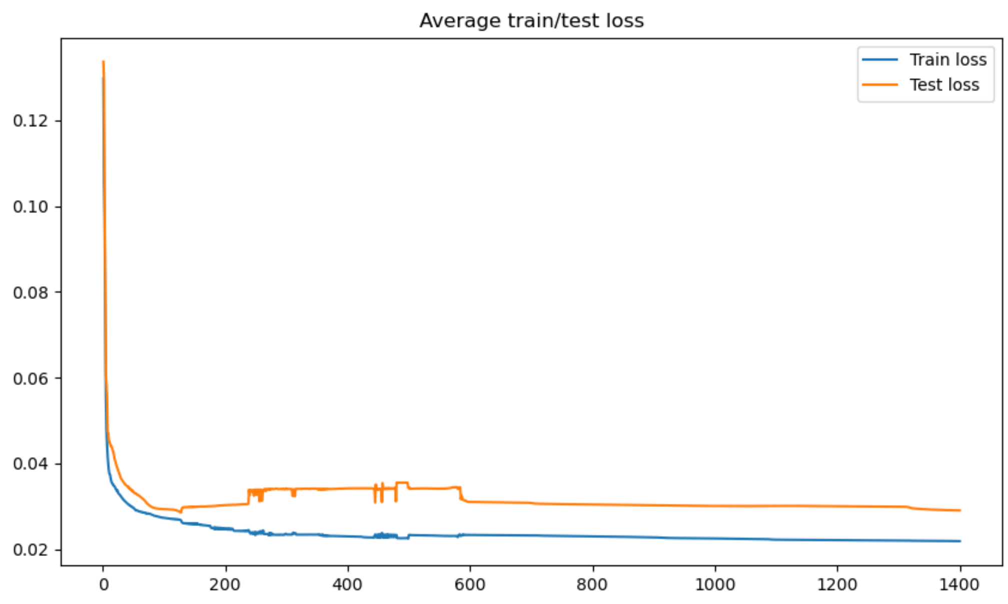
# ASSIGNMENT 3

**Christos Eleftheriou**

**1009537**

## Part B – Classification and Experiments

### B.1 Training Example

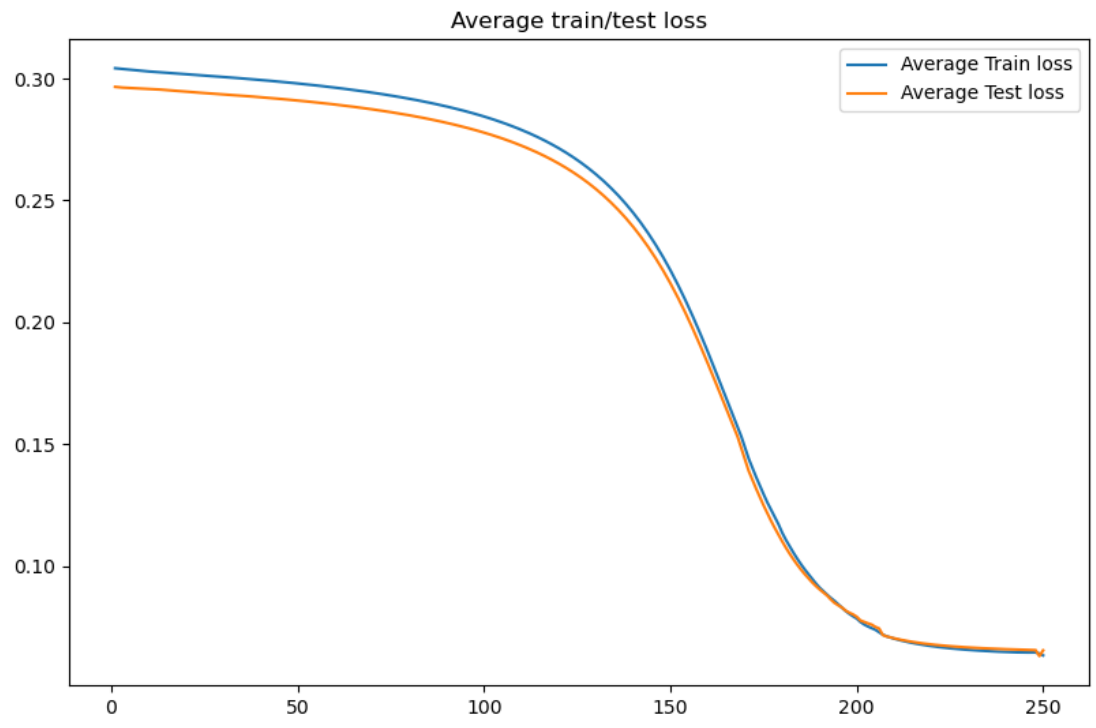


At the start both the train and the test losses decrease rapidly. The train loss continues to decrease at a slower rate, whereas the test loss gets higher and lower (fluctuates) around a certain value and then it also stays the same. The problem is that at some point the decrease in the losses is not significant, which shows us that further training the model after that point does not benefit us, so 1400 epochs are not necessary. A solution to this problem could be early stoppage, where we stop training the model after we stop seeing significant results. Additionally, the test loss has a small increase at some point, while the train loss is lower, and this may be caused by overfitting. However, the difference of the losses does not clearly indicate that there is overfitting in this case.

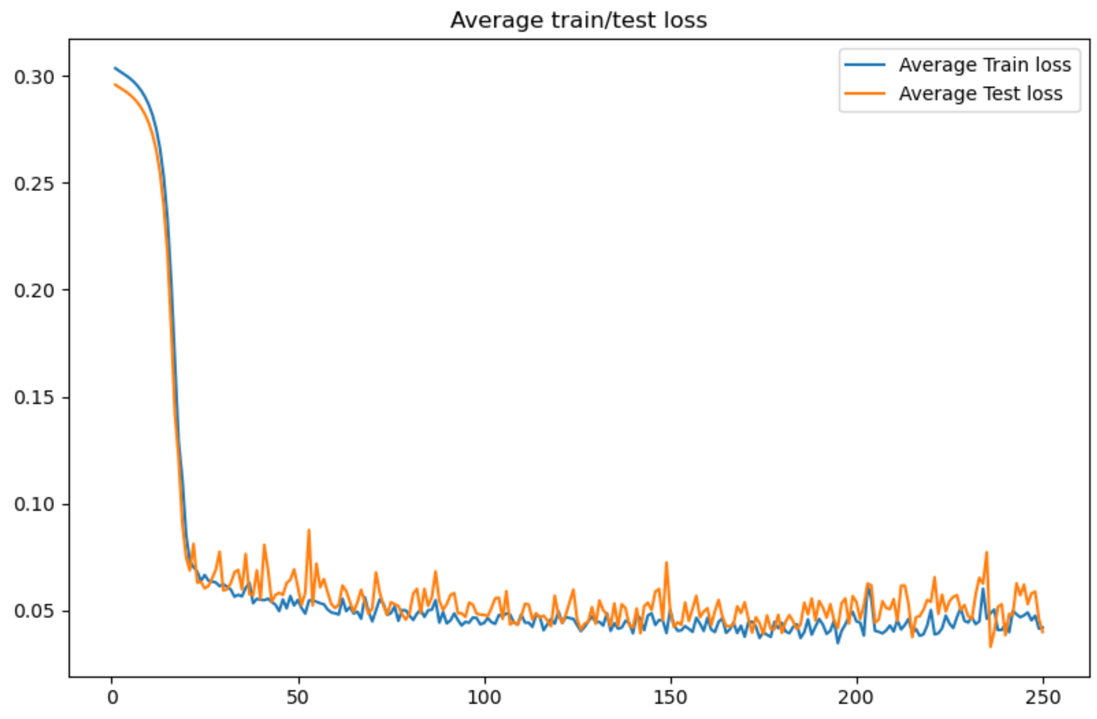
### B.2.1 Learning Rate Experiments

**1. Plot a line chart of the average train/test loss during training (250 iterations)**

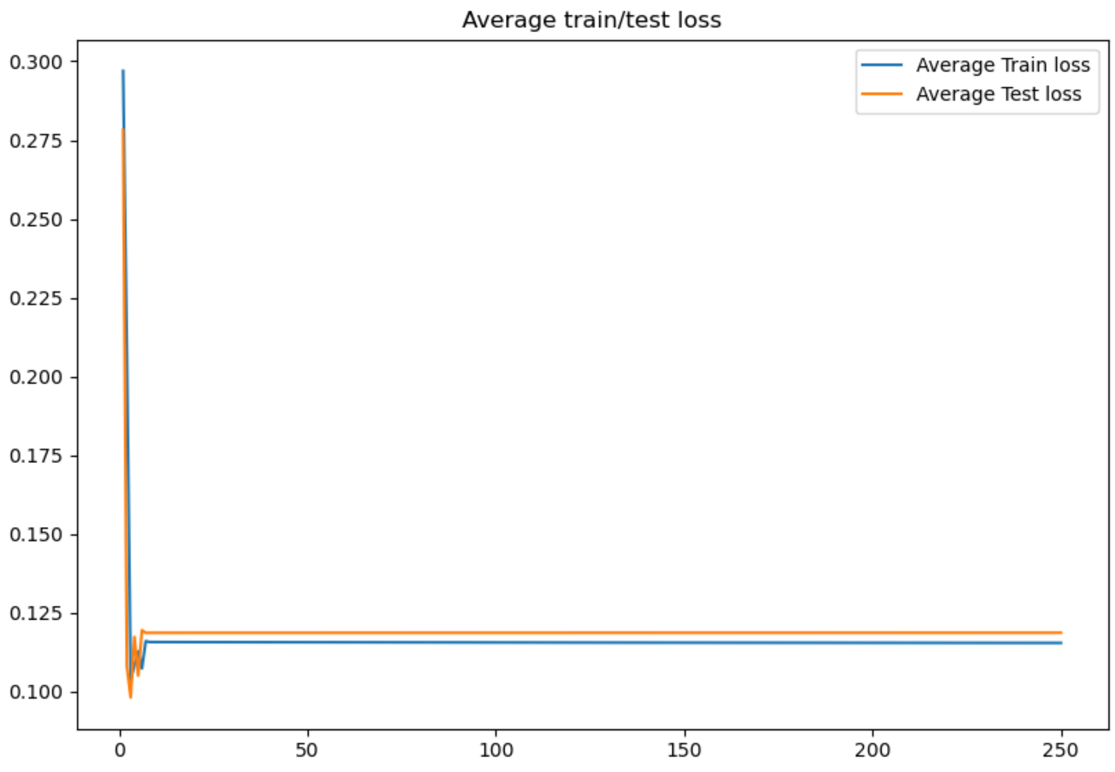
Lr=0.0001



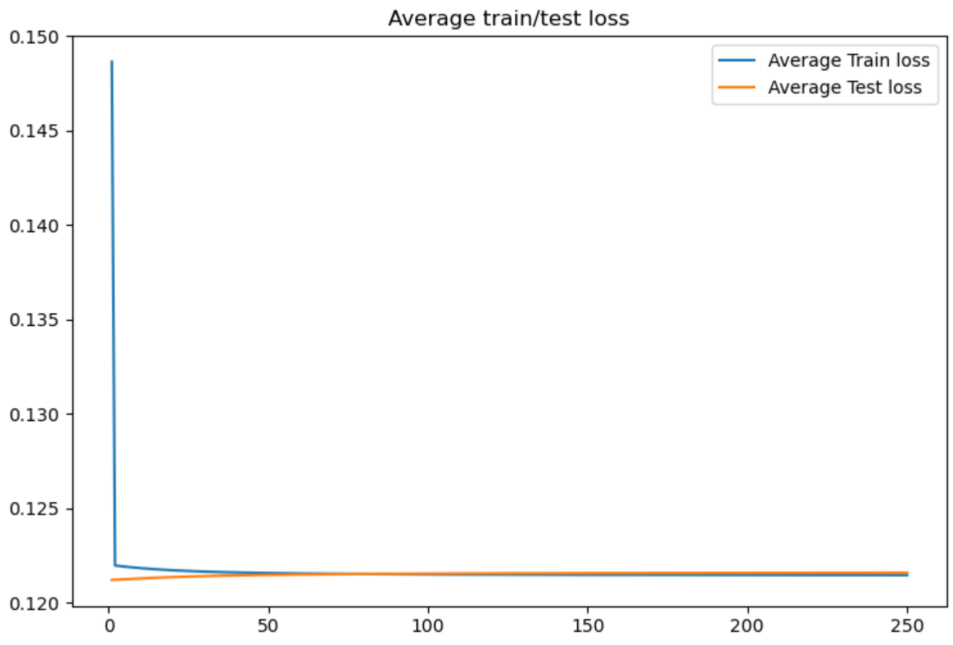
Lr=0.001



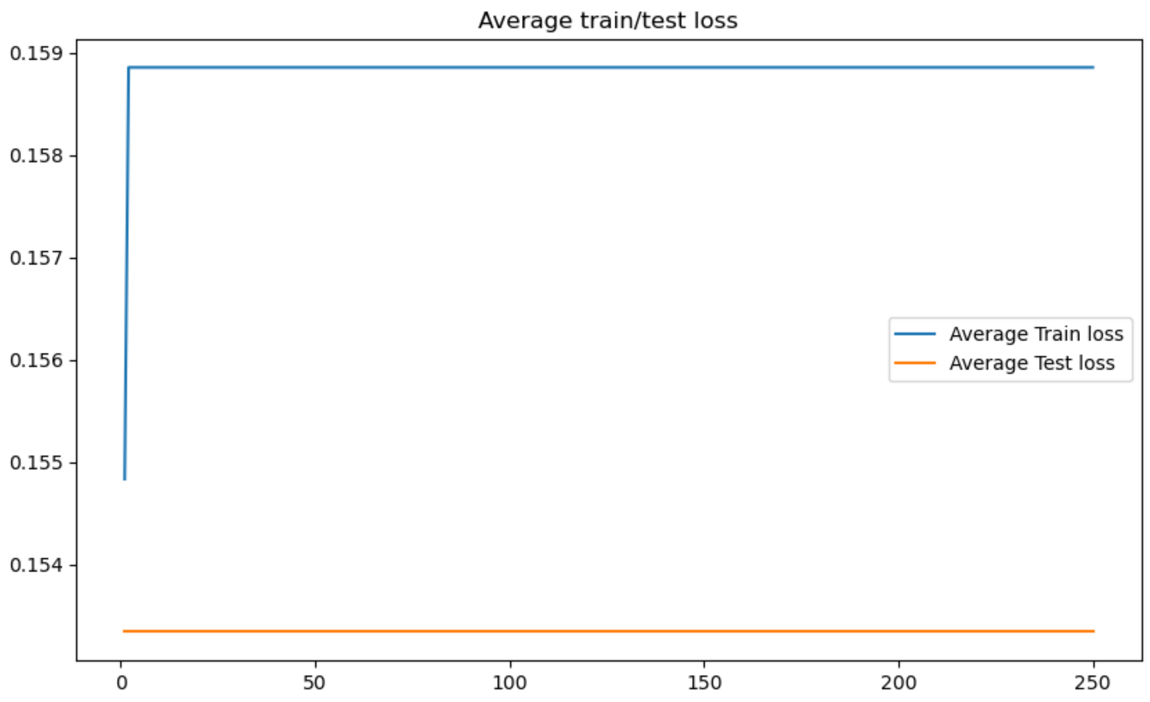
Lr=0.01


Average train/test loss
Average Train loss
Average Test loss
0.300
0.275
0.250
0.225
0.200
0.175
0.150
0.125
0.100
0
50
100
150
200
250

Lr=0.1

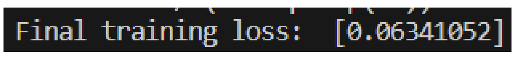

Average train/test loss
Average Train loss
Average Test loss
0.150
0.145
0.140
0.135
0.130
0.125
0.120
0
50
100
150
200
250

Lr=1



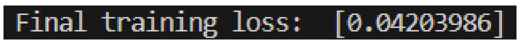
**2. Print the final training loss of the neural network.**

Lr=0.0001



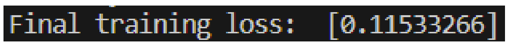
```
Final training loss:  [0.06341052]
```

Lr=0.001



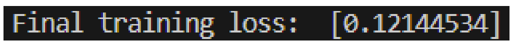
```
Final training loss:  [0.04203986]
```

Lr=0.01



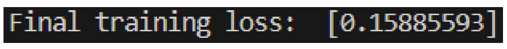
```
Final training loss:  [0.11533266]
```

Lr=0.1



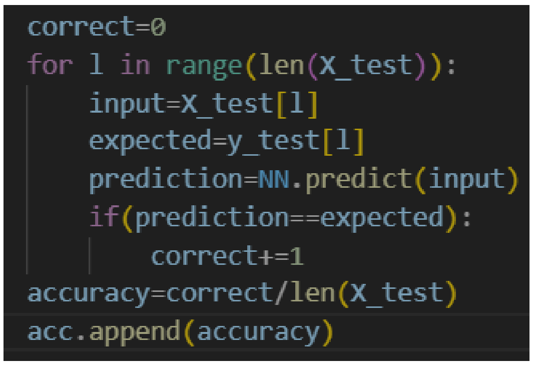
```
Final training loss:  [0.12144534]
```

Lr=1



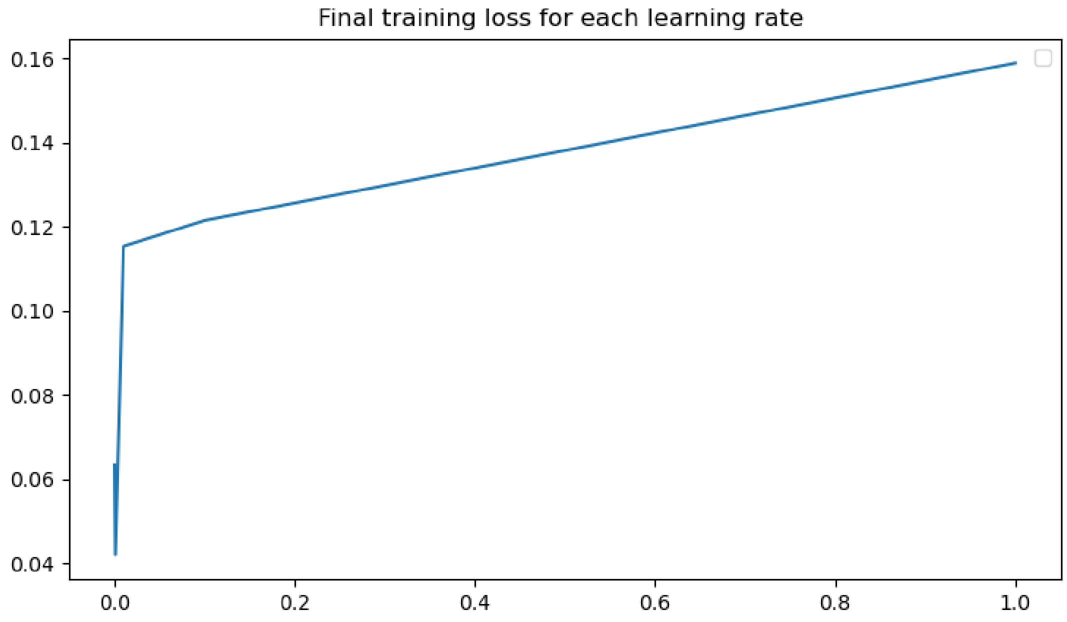
```
Final training loss:  [0.15885593]
```

**3. Calculate the accuracy of the neural network on the test set.**

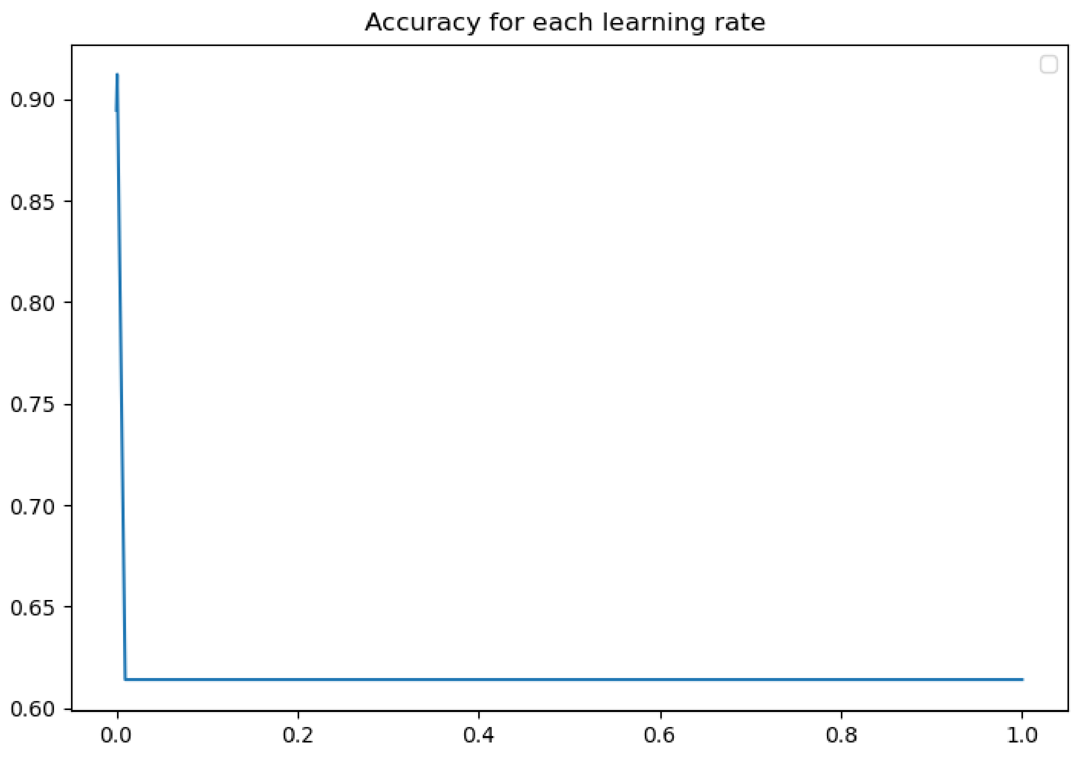


```
correct=0
for l in range(len(X_test)):
    input=X_test[l]
    expected=y_test[l]
    prediction=NN.predict(input)
    if(prediction==expected):
        correct+=1
accuracy=correct/len(X_test)
acc.append(accuracy)
```

**4. Plot a line chart of the final training loss for each learning rate (x axis).**



**5. Plot a line chart of the accuracy of the neural network for each learning rate (x axis).**



**Comments:**

Average train/test losses

On the first graph we can see that both the train and test losses decrease at a slow and steady rate and reach a point where the decrease is not significant. On the second graph, the train and test losses have a very big decrease at the start, and then they fluctuate around the same values for the rest of the epochs. The big fluctuation could be caused by the subjection of the model to variance in the dataset. Similarly, on the third graph there is also a big decrease of the train and test loss at the start, but this time their values stay the same for the rest of the epochs. On the fourth graph we can see that again the train loss has a very big decrease at the start and both losses stay the same for the rest of the epochs. For the learning rates 0.01 and 0.1 we can see that the model converges faster than the previous learning rates. In the final graph, for learning rate 1, the train and test losses stay the same throughout all the epochs, which is not normal behavior. Generally, a learning rate of 1 is not usual in machine learning, so this might be the cause of that. We can also see that the lower learning rates converge slower, but reach lower test and train losses than with higher learning rates.

Final training loss for each learning rate

We can see on the line chart that the final training loss gets higher as we increase the learning rates.

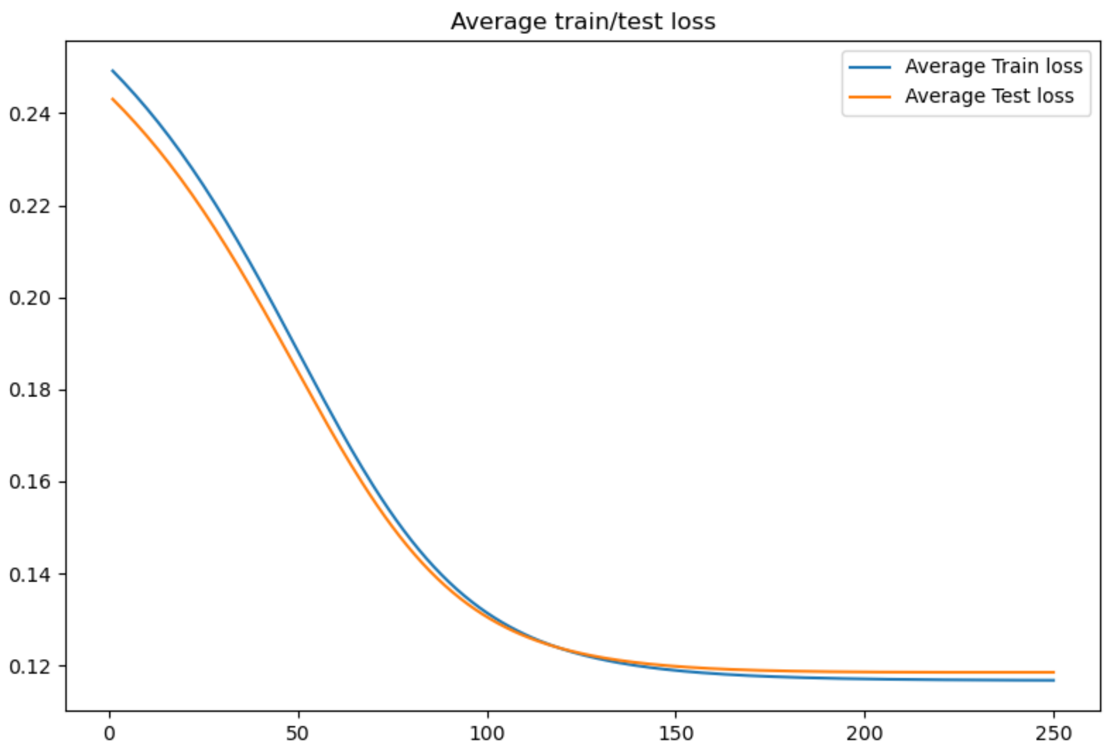
Accuracy for each learning rate

As we can see on the graph, the accuracy gets smaller for bigger learning rates.

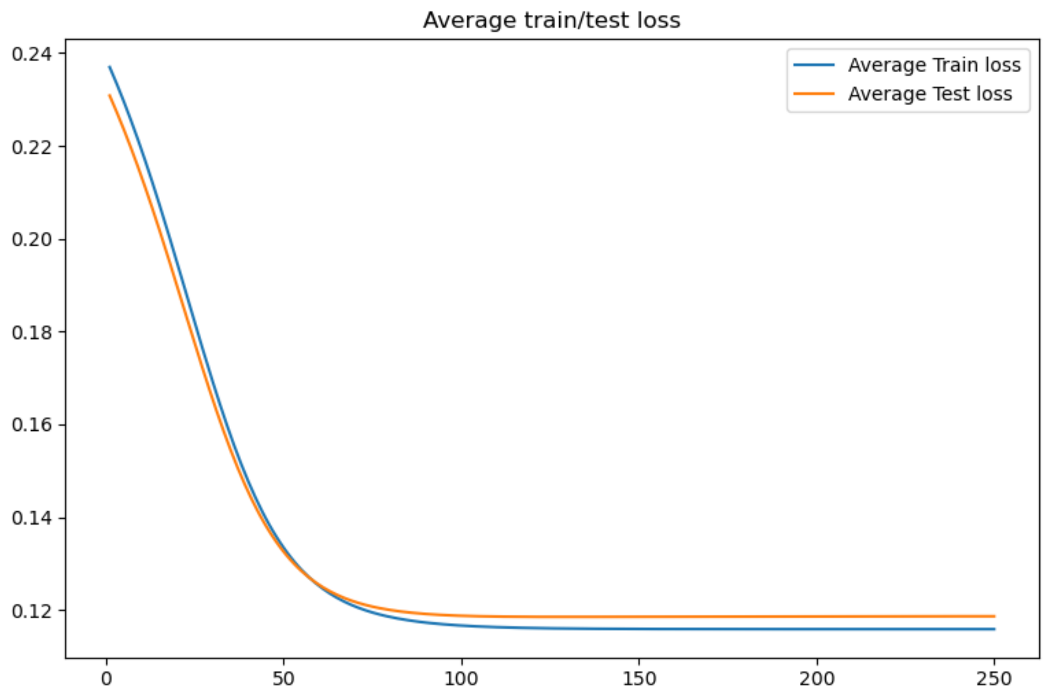
**B.2.2 Hidden Layer Nodes Experiments**

**6. Plot a line chart of the average train/test loss during training (250 iterations).**

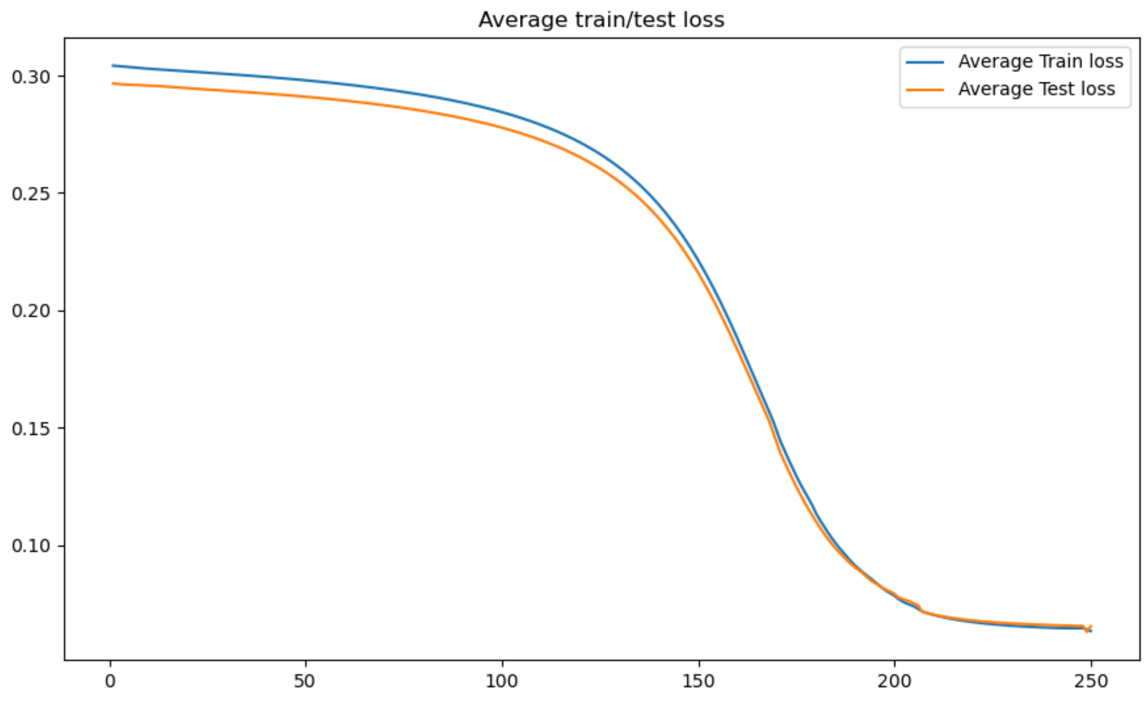
Hl=8



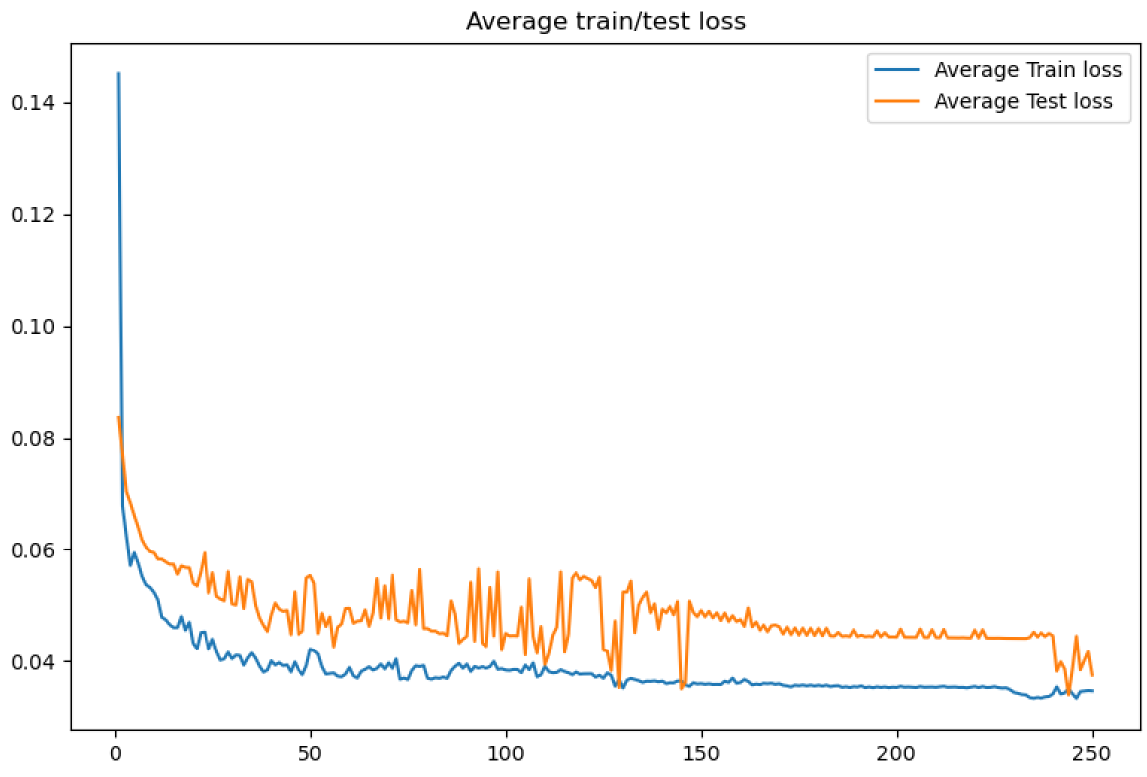
Hl=16


Average train/test loss
Average Train loss
Average Test loss
0.24
0.22
0.20
0.18
0.16
0.14
0.12
0
50
100
150
200
250

Hl=32

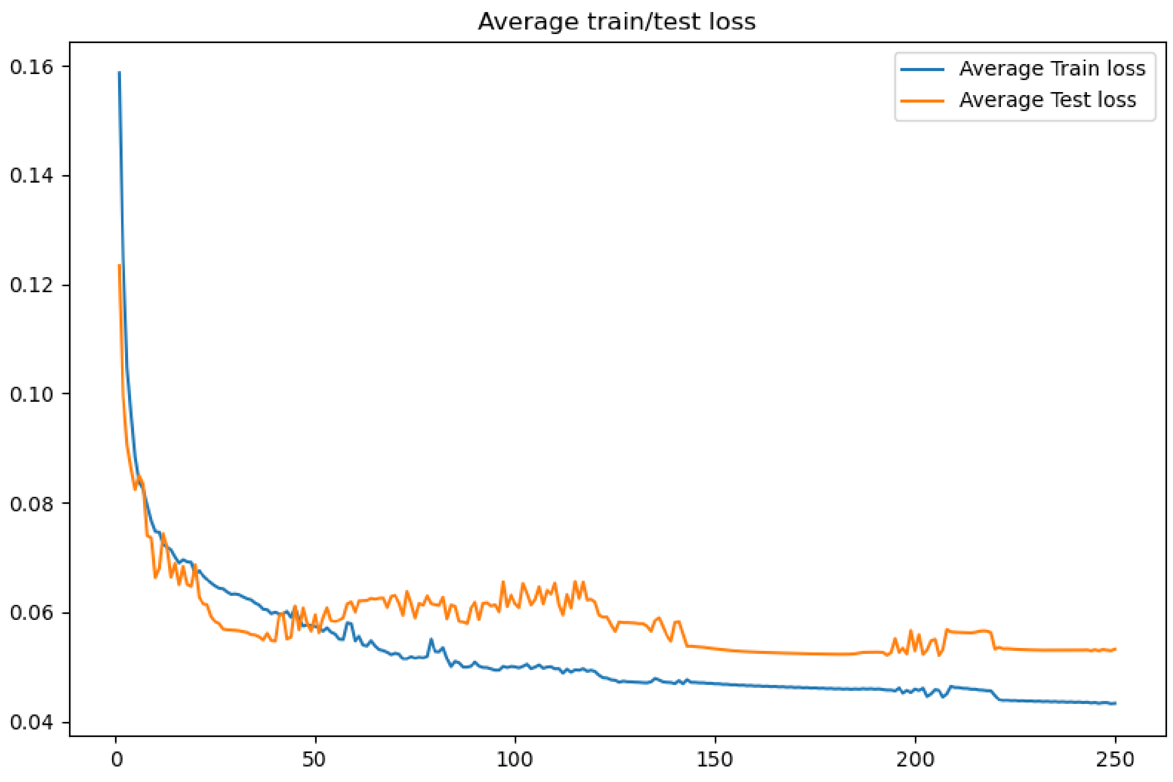

Average train/test loss
Average Train loss
Average Test loss
0.30
0.25
0.20
0.15
0.10
0
50
100
150
200
250

Hl=64



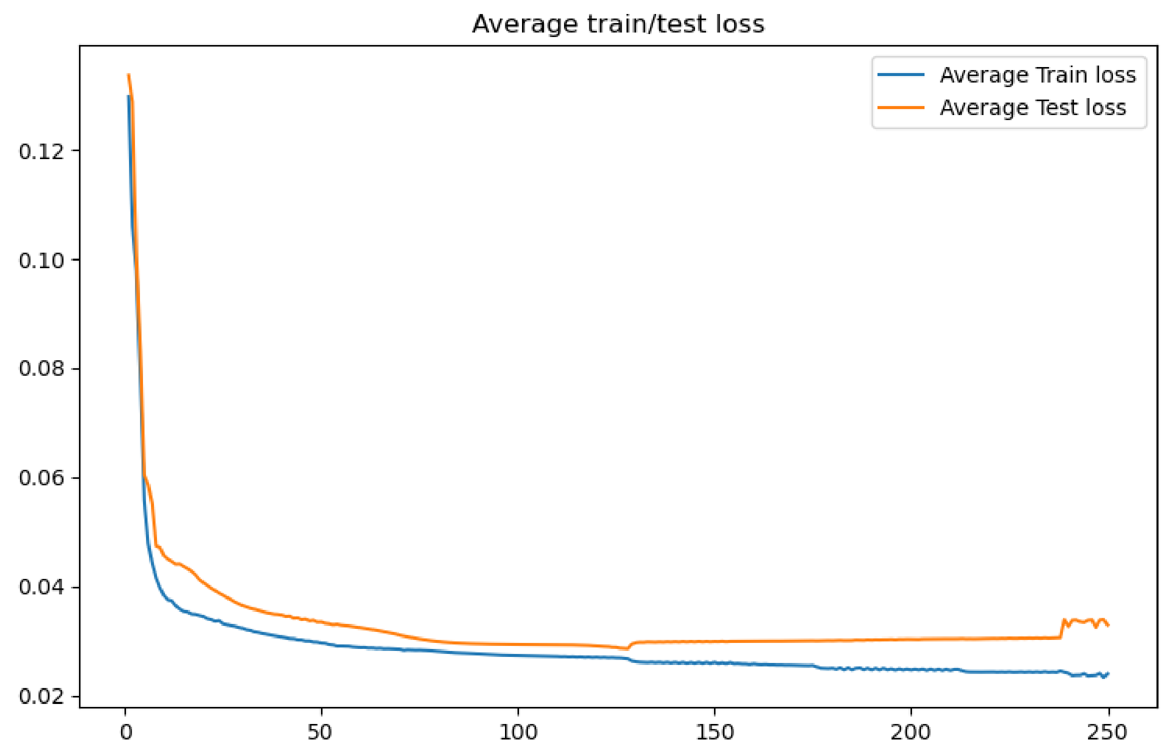
Average train/test loss
Average Train loss
Average Test loss
0.14
0.12
0.10
0.08
0.06
0.04
0
50
100
150
200
250

Hl=128

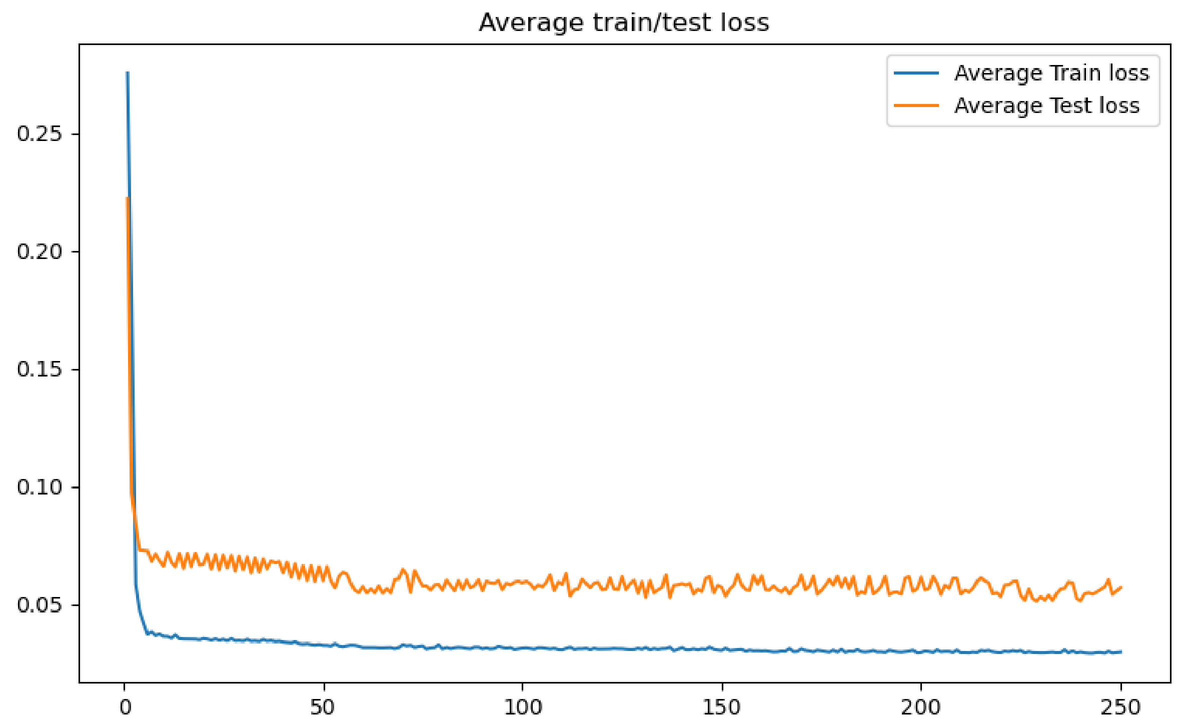


Average train/test loss
Average Train loss
Average Test loss
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0
50
100
150
200
250

Hl=256

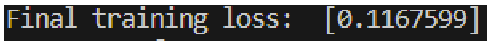

Average train/test loss
Average Train loss
Average Test loss
0.12
0.10
0.08
0.06
0.04
0.02
0
50
100
150
200
250

Hl=512


Average train/test loss
Average Train loss
Average Test loss
0.25
0.20
0.15
0.10
0.05
0
50
100
150
200
250

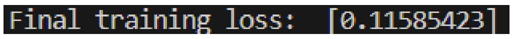
**7. Print the final training loss of the neural network.**

Hidden nodes=8



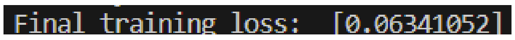
```
Final training loss:  [0.1167599]
```

Hidden nodes=16



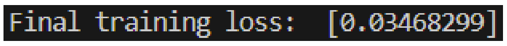
```
Final training loss:  [0.11585423]
```

Hidden nodes=32



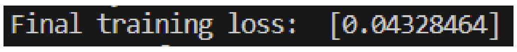
```
Final training loss:  [0.06341052]
```

Hidden nodes=64



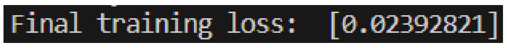
```
Final training loss:  [0.03468299]
```

Hidden nodes=128



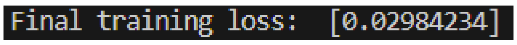
```
Final training loss:  [0.04328464]
```

Hidden nodes=256



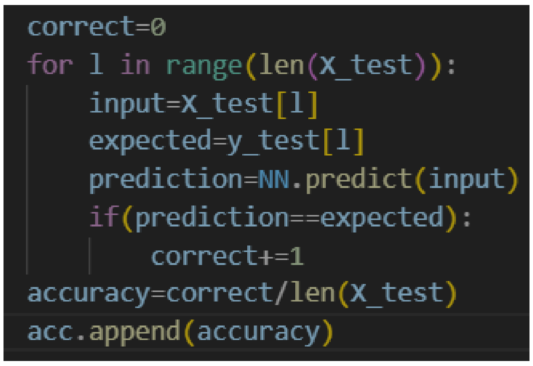
```
Final training loss:  [0.02392821]
```

Hidden nodes=512



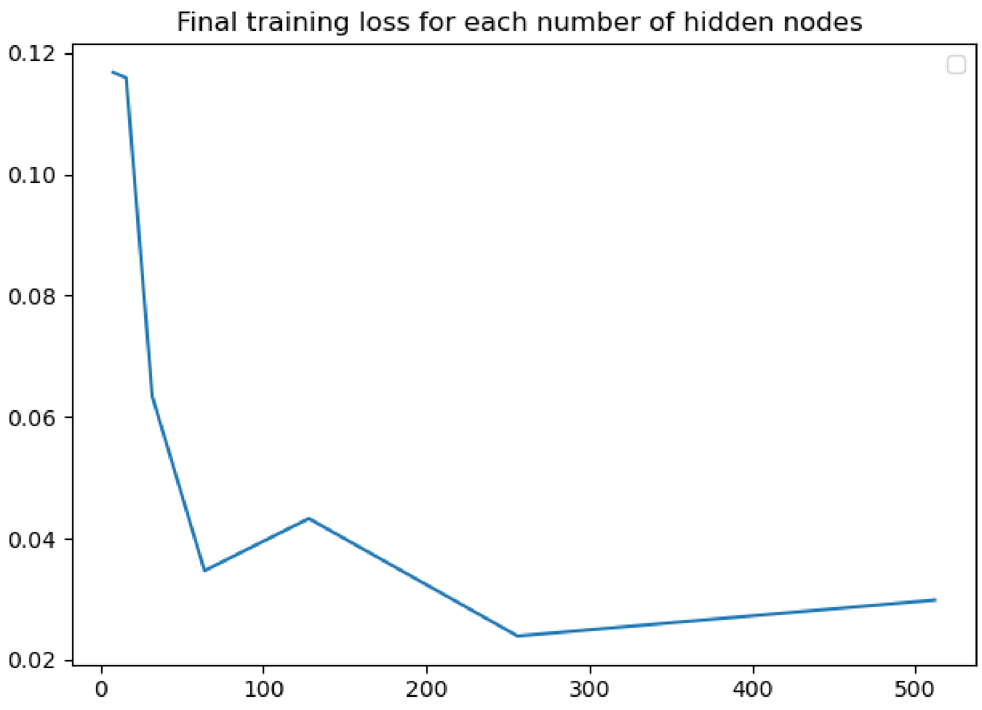
```
Final training loss:  [0.02984234]
```

**8. Calculate the accuracy of the neural network on the test set.**

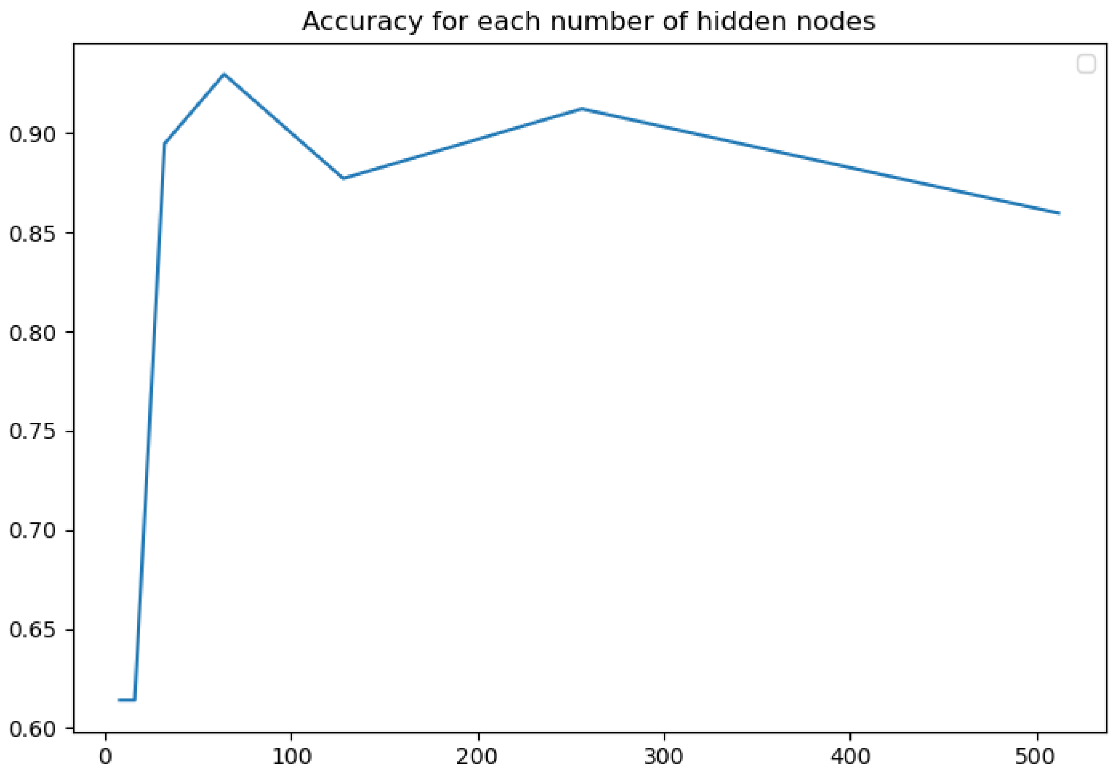


```
correct=0
for l in range(len(X_test)):
    input=X_test[l]
    expected=y_test[l]
    prediction=NN.predict(input)
    if(prediction==expected):
        correct+=1
accuracy=correct/len(X_test)
acc.append(accuracy)
```

**9. Plot a line chart of the final training loss for each number of nodes (x axis).**



**10. Plot a line chart of the accuracy of the neural network for each number of nodes (x axis).**



**Comments:**

<u>Average train/test losses</u>

The first two graphs are similar, and we can see that the losses have a big decrease at the start and stay at around the same values for the rest of the epochs, however the losses on the second graph are a bit lower. On the third graph the average train and test losses decrease but at a later point than the previous graphs, and the losses are lower than both. Graphs four, five, six and seven are quite similar because the losses have a big decrease at the start and then fluctuate for the rest of the epochs. Graph 4 has the biggest fluctuations, followed by graph five and graph seven, and graph six has the smaller fluctuations of them all. Graph six (hidden nodes=256) has the smallest losses of them, followed by four, five and seven. Another observation is that on graph seven, the test loss is always higher than the train loss. Generally, we can see that we have the smallest losses for 256 hidden nodes.

<u>Final training loss for each learning rate</u>

We can see from the graph that the final training loss gets lower as we increase the number of the nodes in the hidden layer from 8 to 16 to 64, then increases for 128, then decreases for 256 and increases for 512. We observe the lowest error when we use 256 hidden layer nodes.

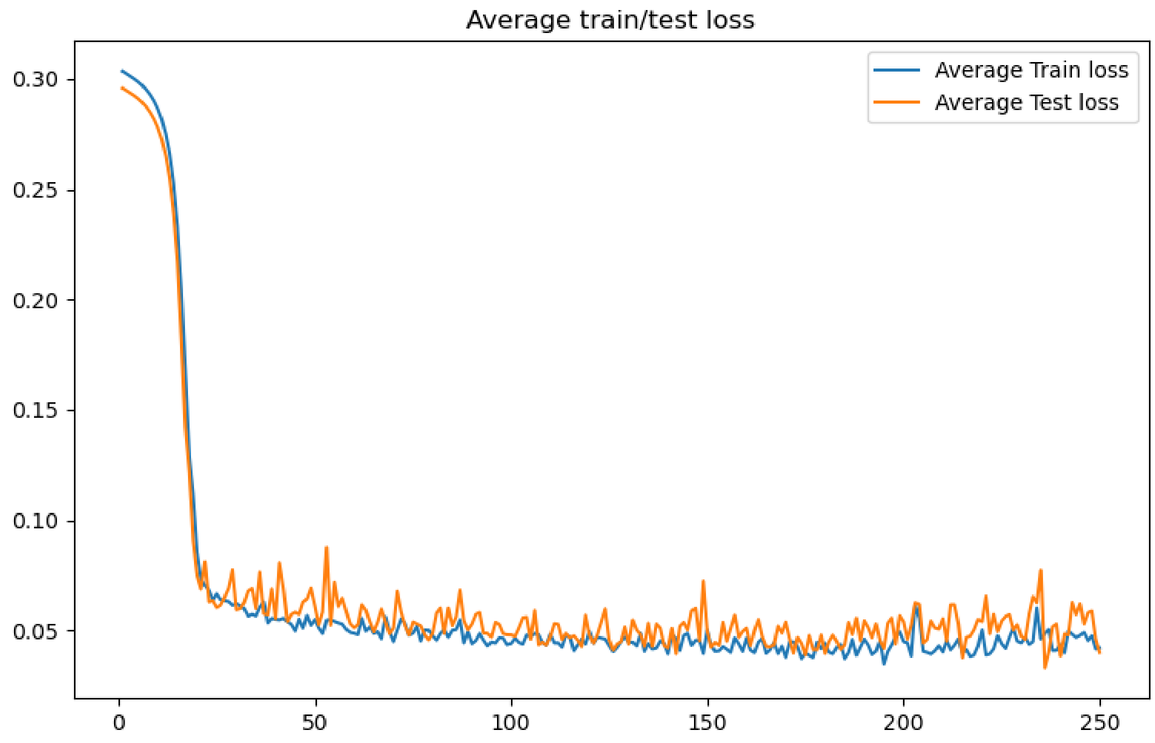
<u>Accuracy for each learning rate</u>

As we can see on the graph, the accuracy increases as we increase the hidden layer nodes from 8 to 16 to 32 and 64, then decreases for 128, then increases again for 256, and then decreases for 512. We observe the highest accuracies for 64 and 256 nodes.

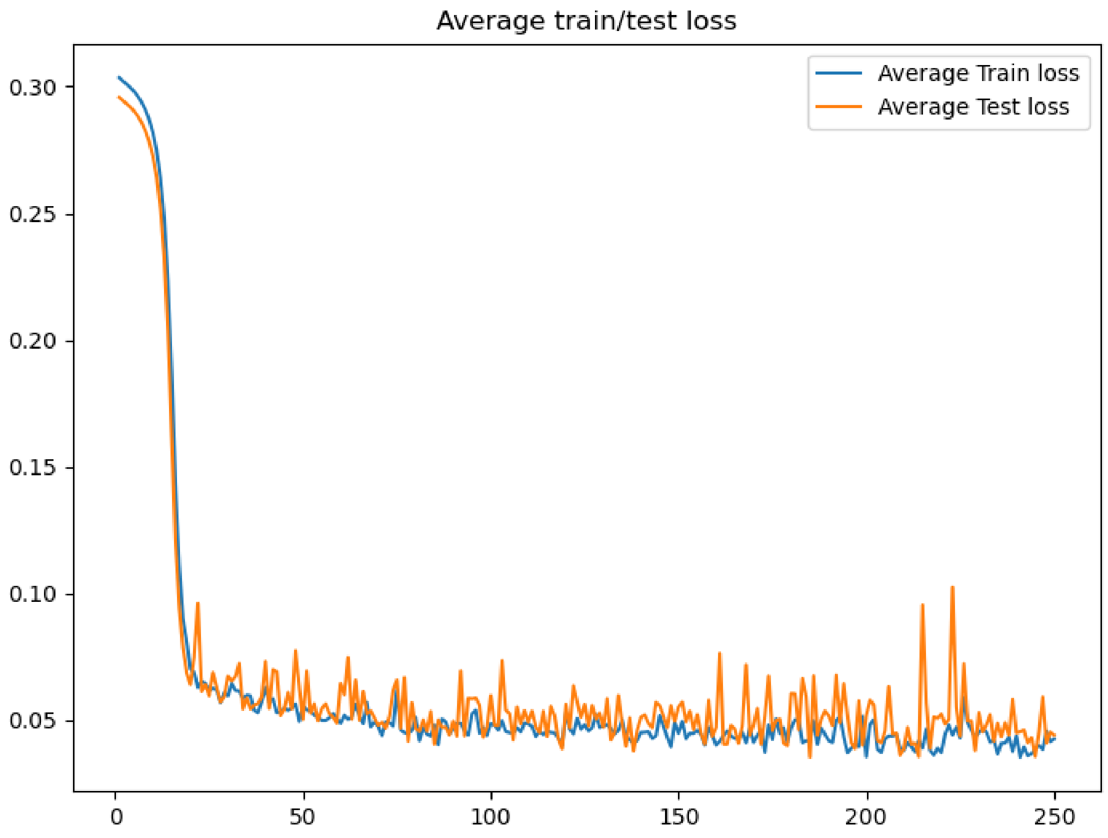
**B.2.3 Momentum Experiments**

**11. Plot a line chart of the average train/test loss during training (250 iterations).**

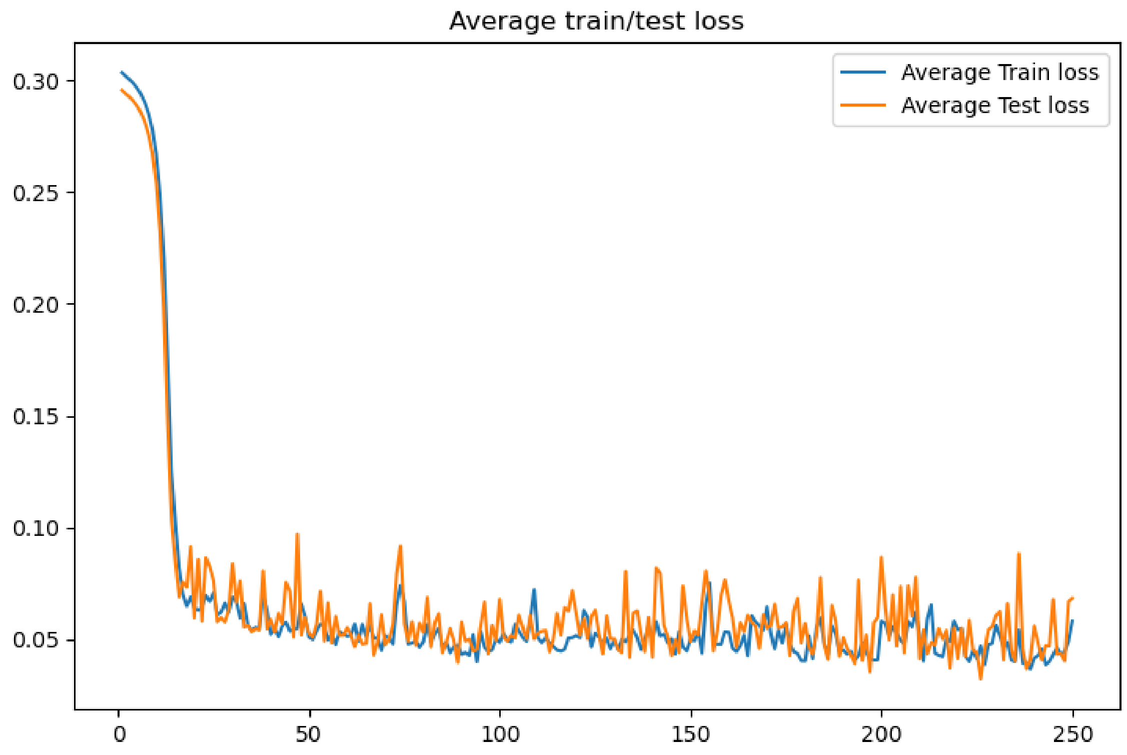
M=0


Average train/test loss
Average Train loss
Average Test loss
0.30
0.25
0.20
0.15
0.10
0.05
0
50
100
150
200
250

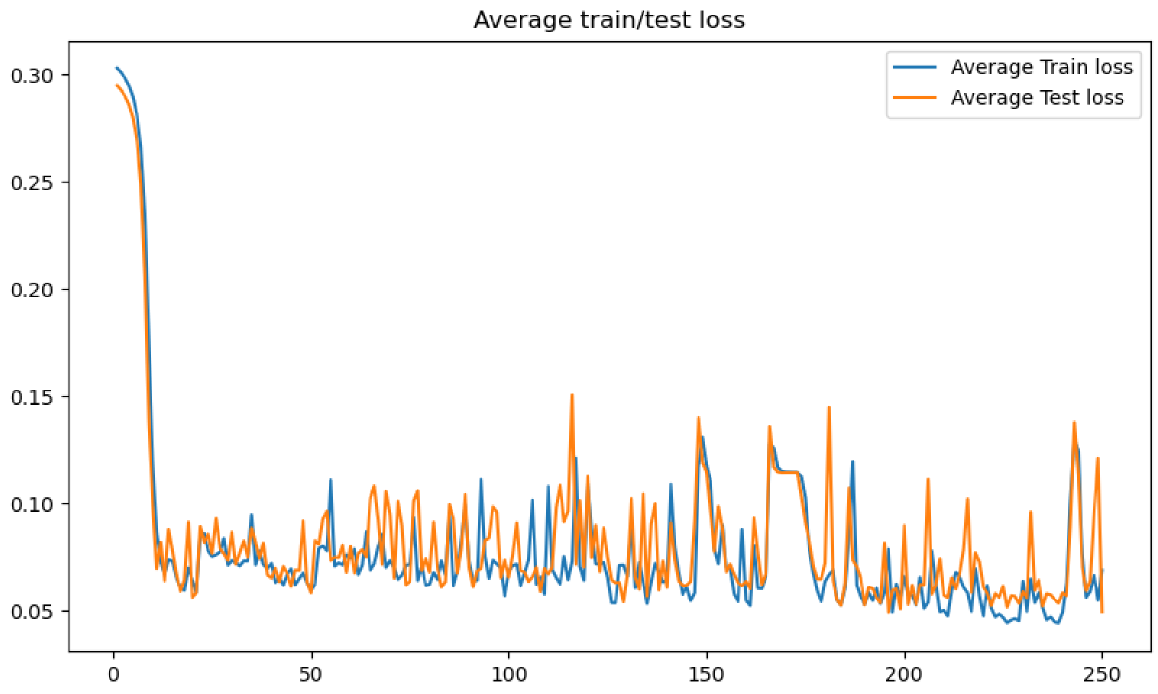
M=0.1


Average train/test loss
Average Train loss
Average Test loss
0.30
0.25
0.20
0.15
0.10
0.05
0
50
100
150
200
250

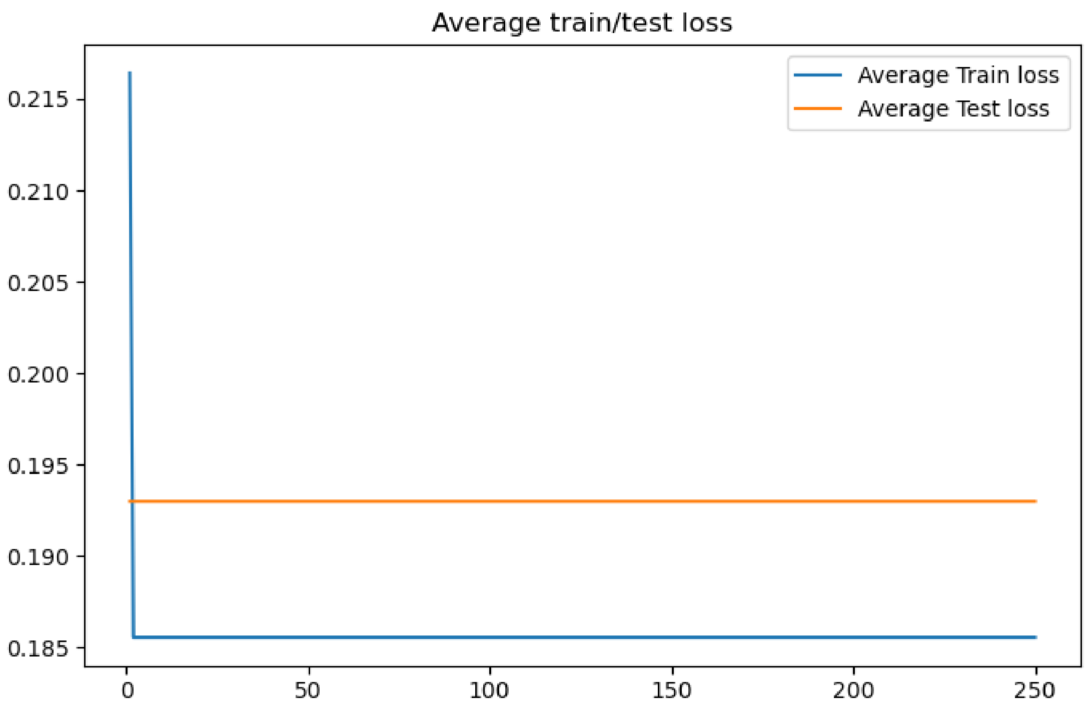
M=0.25


Average train/test loss
Average Train loss
Average Test loss
0.30
0.25
0.20
0.15
0.10
0.05
0
50
100
150
200
250

M=0.5

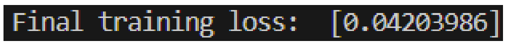

Average train/test loss
Average Train loss
Average Test loss
0.30
0.25
0.20
0.15
0.10
0.05
0
50
100
150
200
250

M=1



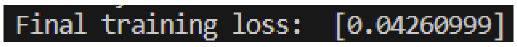
**12. Print the final training loss of the neural network.**

Momentum=0



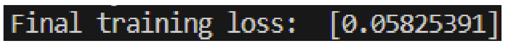
```
Final training loss:  [0.04203986]
```

Momentum=0.1



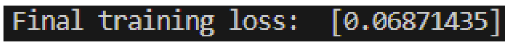
```
Final training loss:  [0.04260999]
```

Momentum=0.25



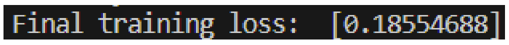
```
Final training loss:  [0.05825391]
```

Momentum=0.5



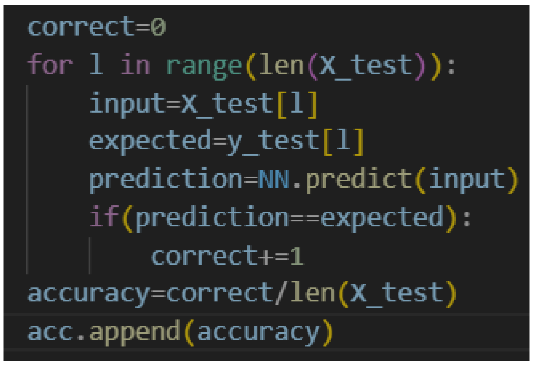
```
Final training loss:  [0.06871435]
```

Momentum=1



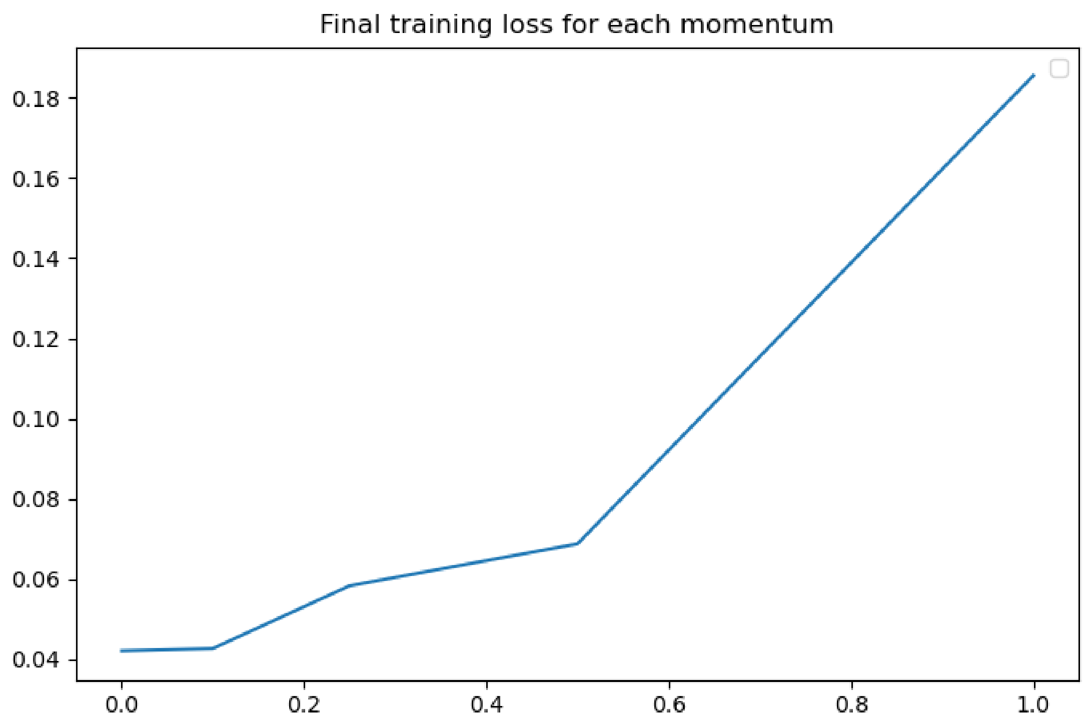
```
Final training loss:  [0.18554688]
```

**13. Calculate the accuracy of the neural network on the test set.**

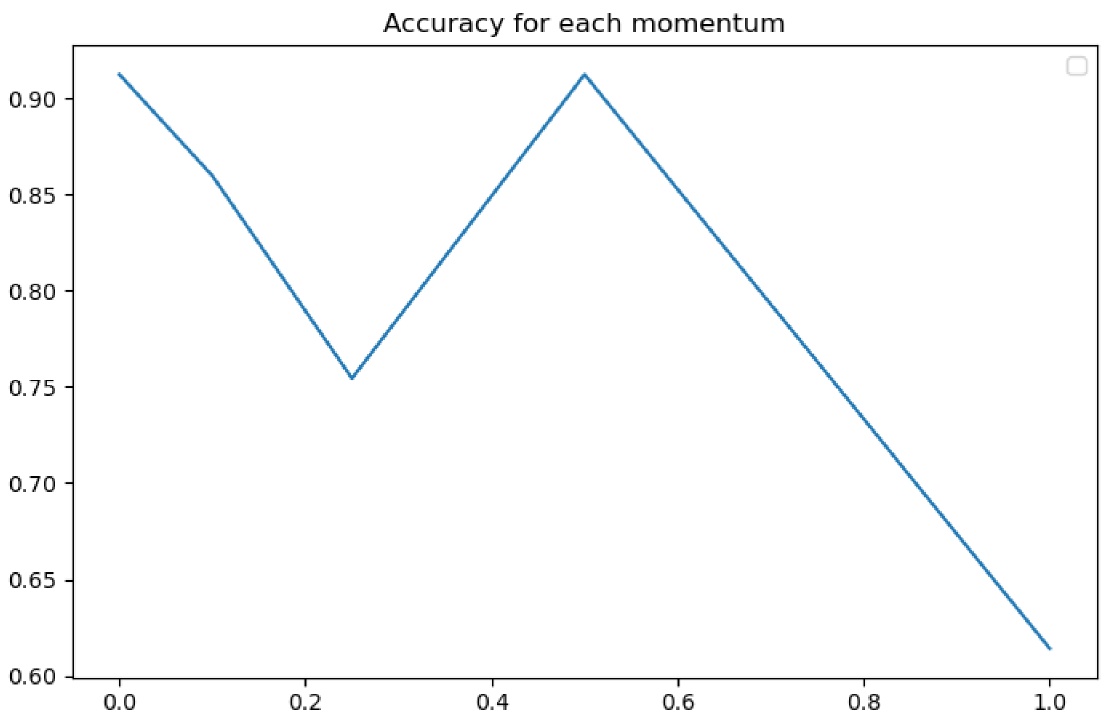


```
correct=0
for l in range(len(X_test)):
    input=X_test[l]
    expected=y_test[l]
    prediction=NN.predict(input)
    if(prediction==expected):
        correct+=1
accuracy=correct/len(X_test)
acc.append(accuracy)
```

**14. Plot a line chart of the final training loss for each momentum (x axis).**



**15. Plot a line chart of the accuracy of the neural network for each momentum (x axis).**



**Comments:**

Average train/test losses

The first four graphs are very similar. As we can see both train and test losses have a big decrease at the start, and then fluctuate around the same values for the rest of the epochs, with the fourth graph having the biggest fluctuations. On the fifth graph, the train and test loss stay at the same value throughout all the epochs, which is not normal behavior.

Final training loss for each learning rate

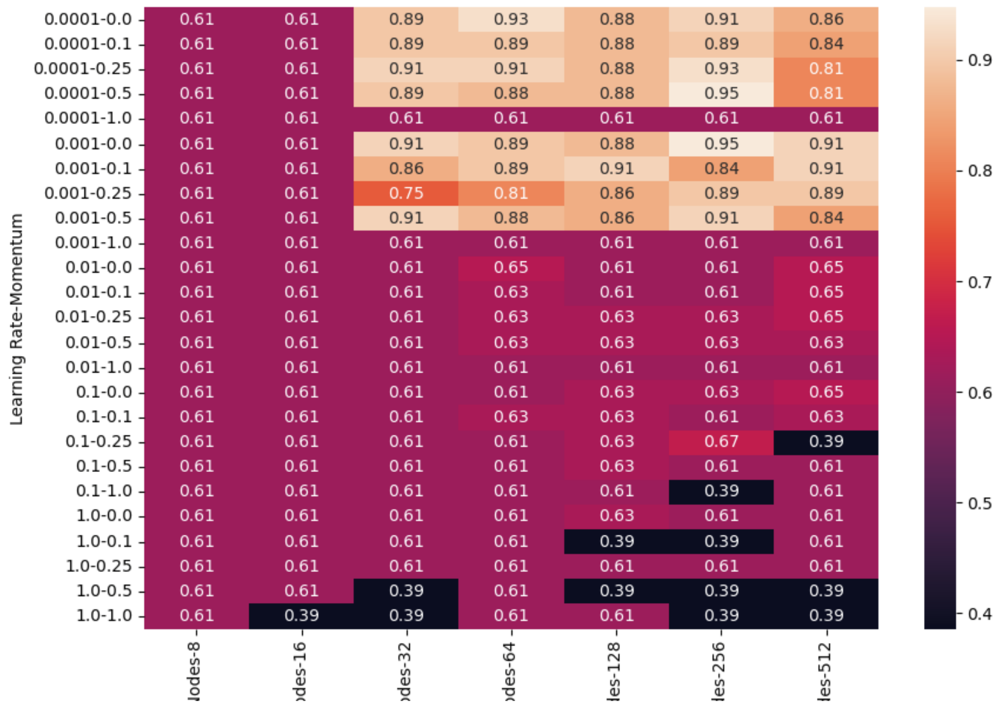
We can see from the graph that as we increase the momentum, the final training loss gets bigger.

Accuracy for each learning rate

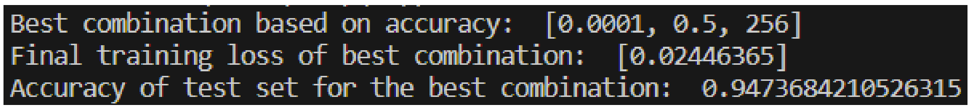
As we can see on the graph, the accuracy decreases for momentum equal to 0.1 and 0.25, then increases again for 0.5, and then decreases as we increase the momentum. The biggest accuracy is observed for momentum equal to 0.5 and 0.

**B.2.4 Final Neural Network – Hyperparameter finetuning**

**Visualize the accuracy of each hyperparameter combination on a table or chart, and select the best combination based on accuracy**



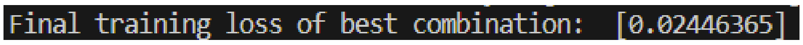
**For the best hyperparameter combination**



```
Best combination based on accuracy:  [0.0001, 0.5, 256]
Final training loss of best combination:  [0.02446365]
Accuracy of test set for the best combination:  0.9473684210526315
```

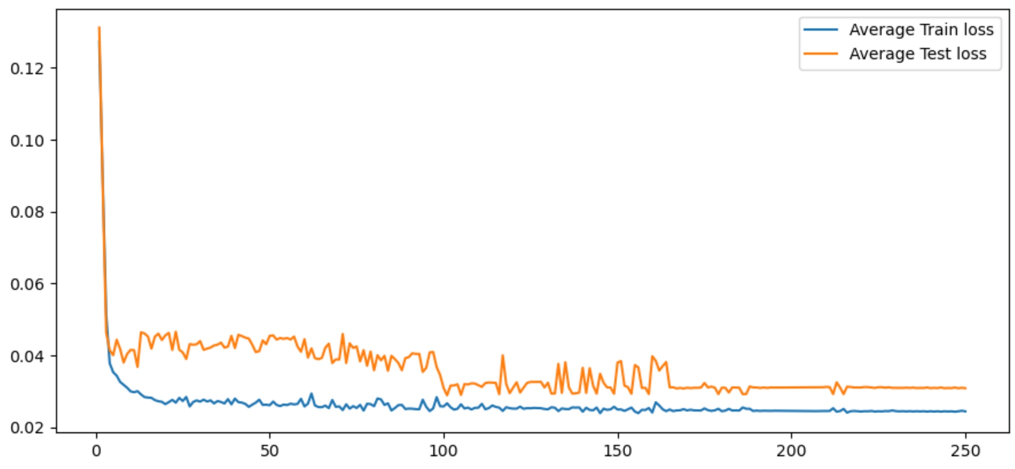
**Plot a line chart of the average train/test loss during training (250 iterations).**

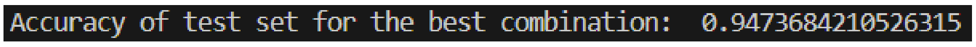
**Print the final training loss of the neural network.**



```
Final training loss of best combination:  [0.02446365]
```

**Print the accuracy of the neural network on the test set.**





```
Accuracy of test set for the best combination:  0.9473684210526315
```