INF 367- Mandatory 2 Report

The first setup

Data

The project uses the Iris data set. The data was first split into 60/20/20 Training/test/validation data. With a set random seed. Angle encoding was used for the input encoder.

Circuits

The project consists of 2 trainable circuits. One with 4 qbits using all the features and one with 8 qbits, copying all the features once.

The first 4-Qbit Circuit based on example 5.7.2 from the lecture notes:

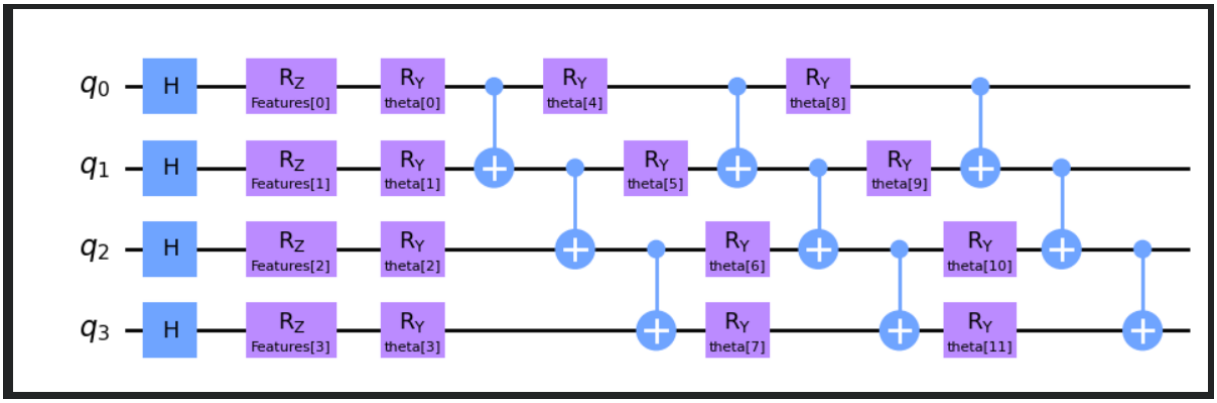

Figure 1 showing the first 4-qbit circuit we trained

The first 8-Qbit circuit also based on 5.7.2 with additional Ry gates at the end of each qbit

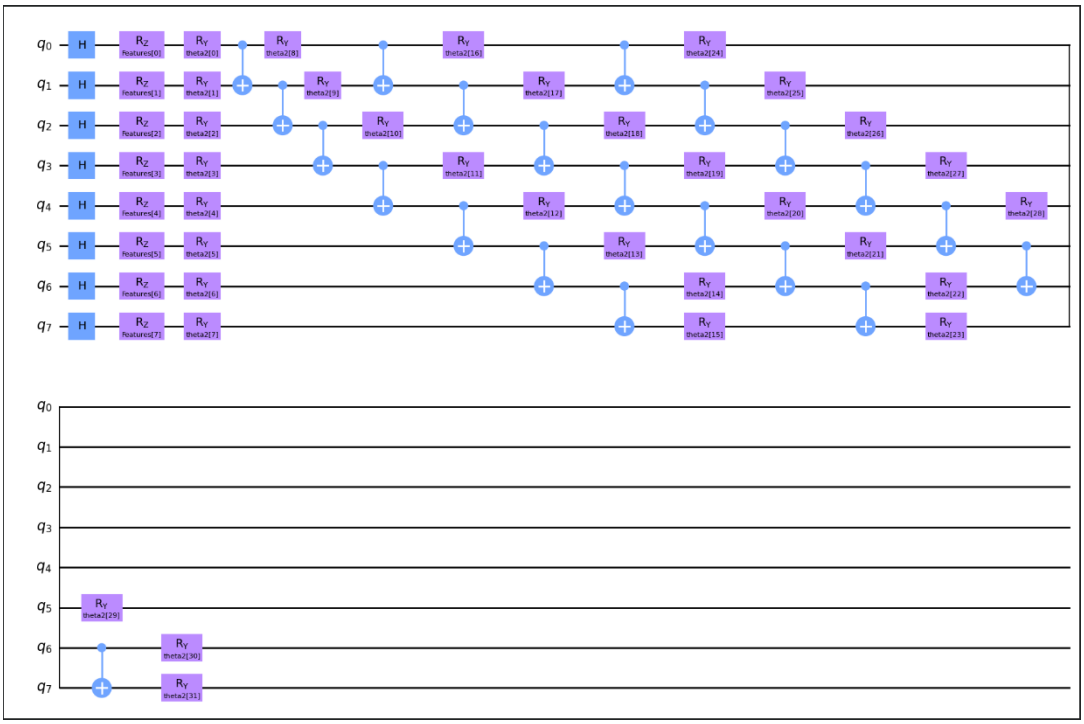

Figure 2: First 8 qbit circuit trained

Results

The circuits was trained using qiskits **samplerQNN** with **gradient descent** and **cross entropy** with a **learning rate of 0.01 and 50 epochs.** Below are the results from the first training without hyper parameter tuning.

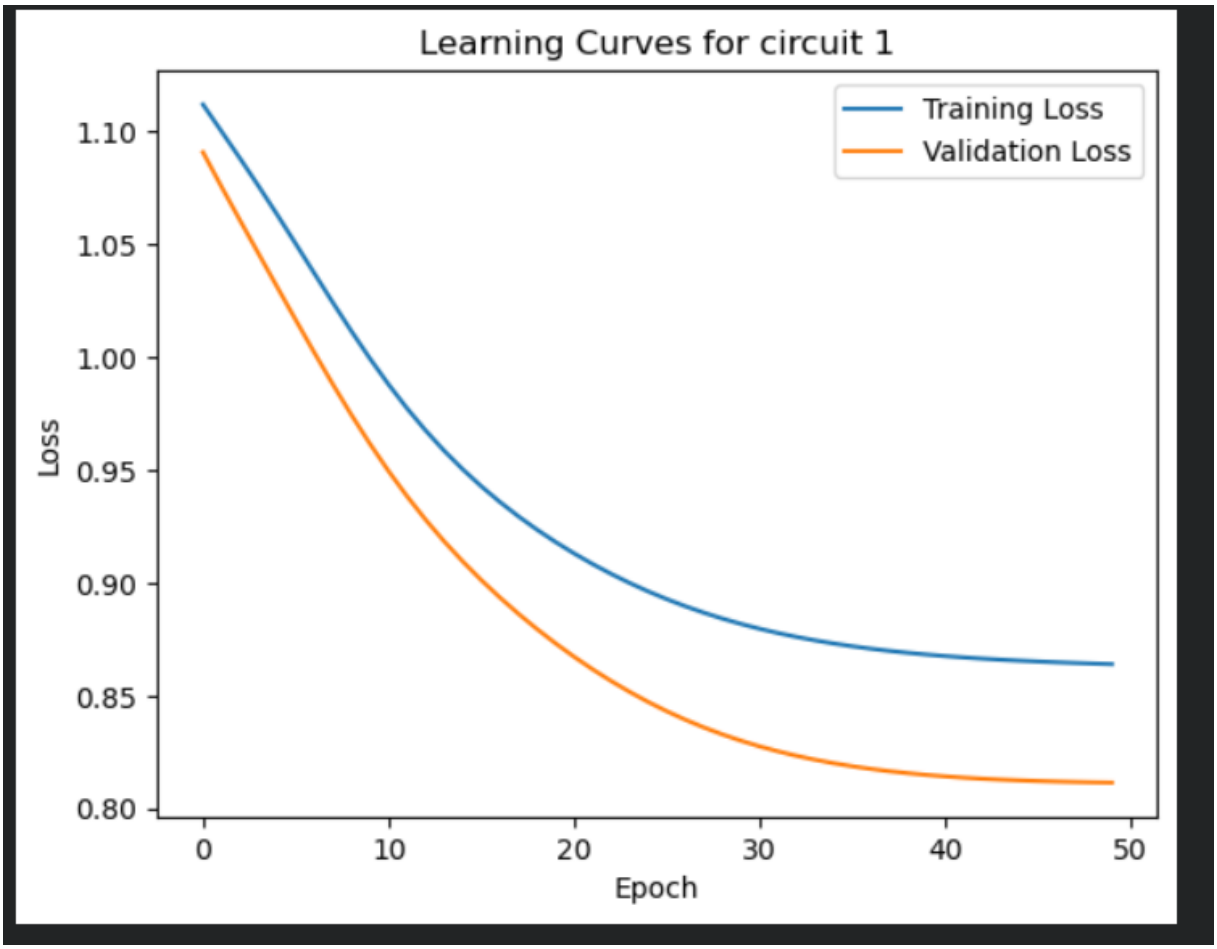

Figure 3: Learning curves for first 4-qbit circuit

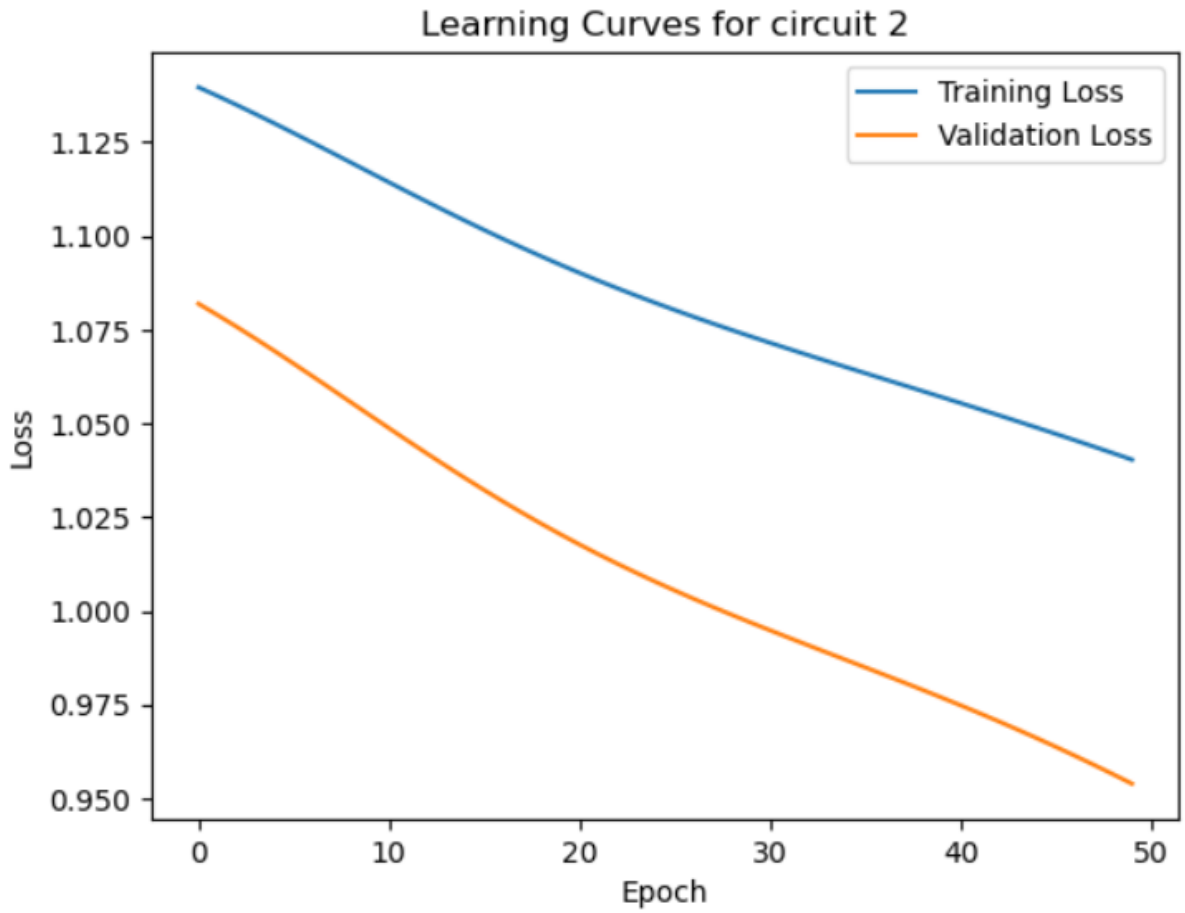

Figure 4: Learning curves for first 8-qbit circuit.

After plotting the validation and learning losses for each circuit, the quantum neural networks was tested on unseen data. Giving the following results and confusion matrices.

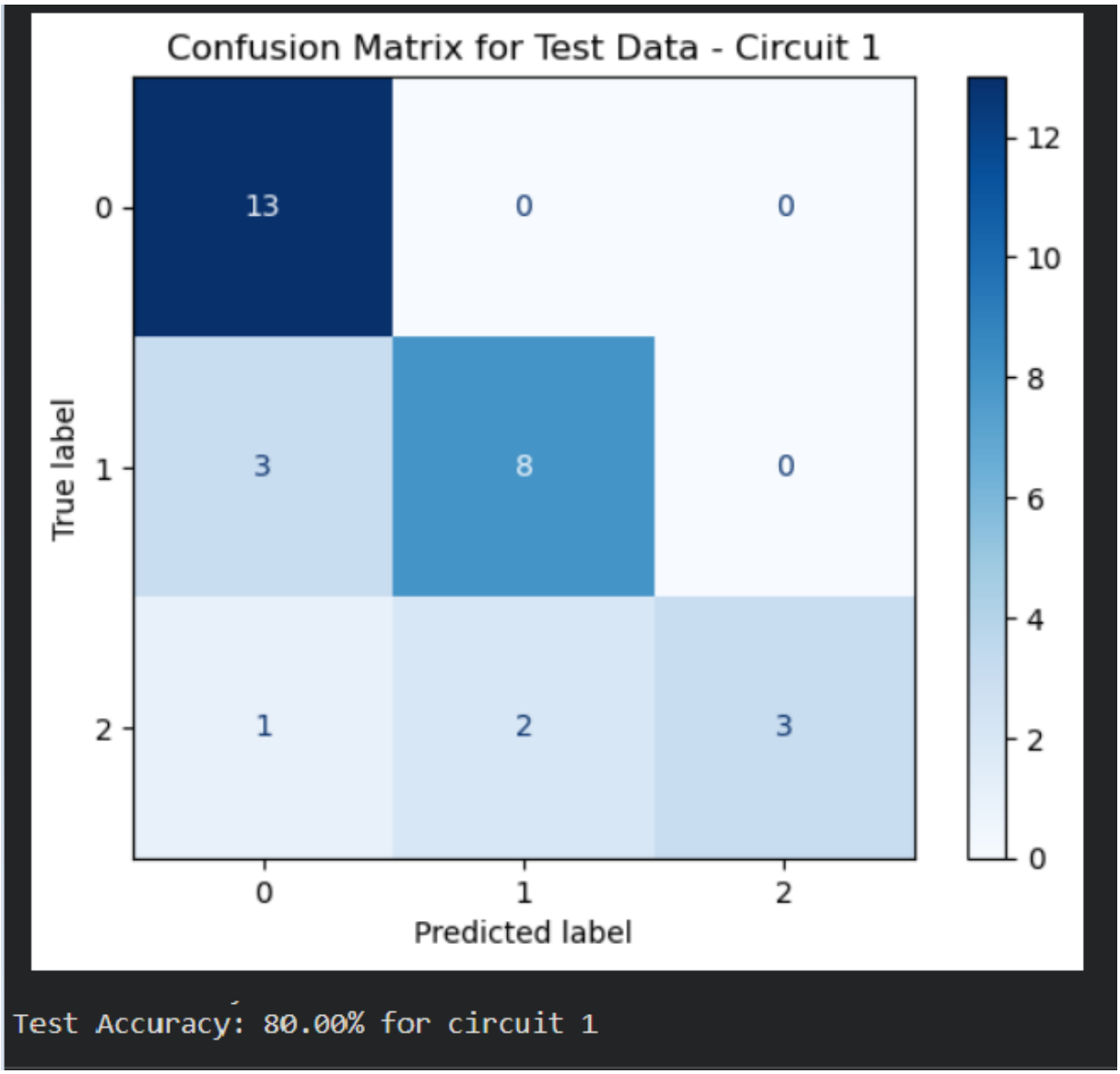

Figure 5: Results of Trained 4-qbit circuit on unseen data

```
Training time for circuit 1 with 8 qbits is: 351 seconds
```

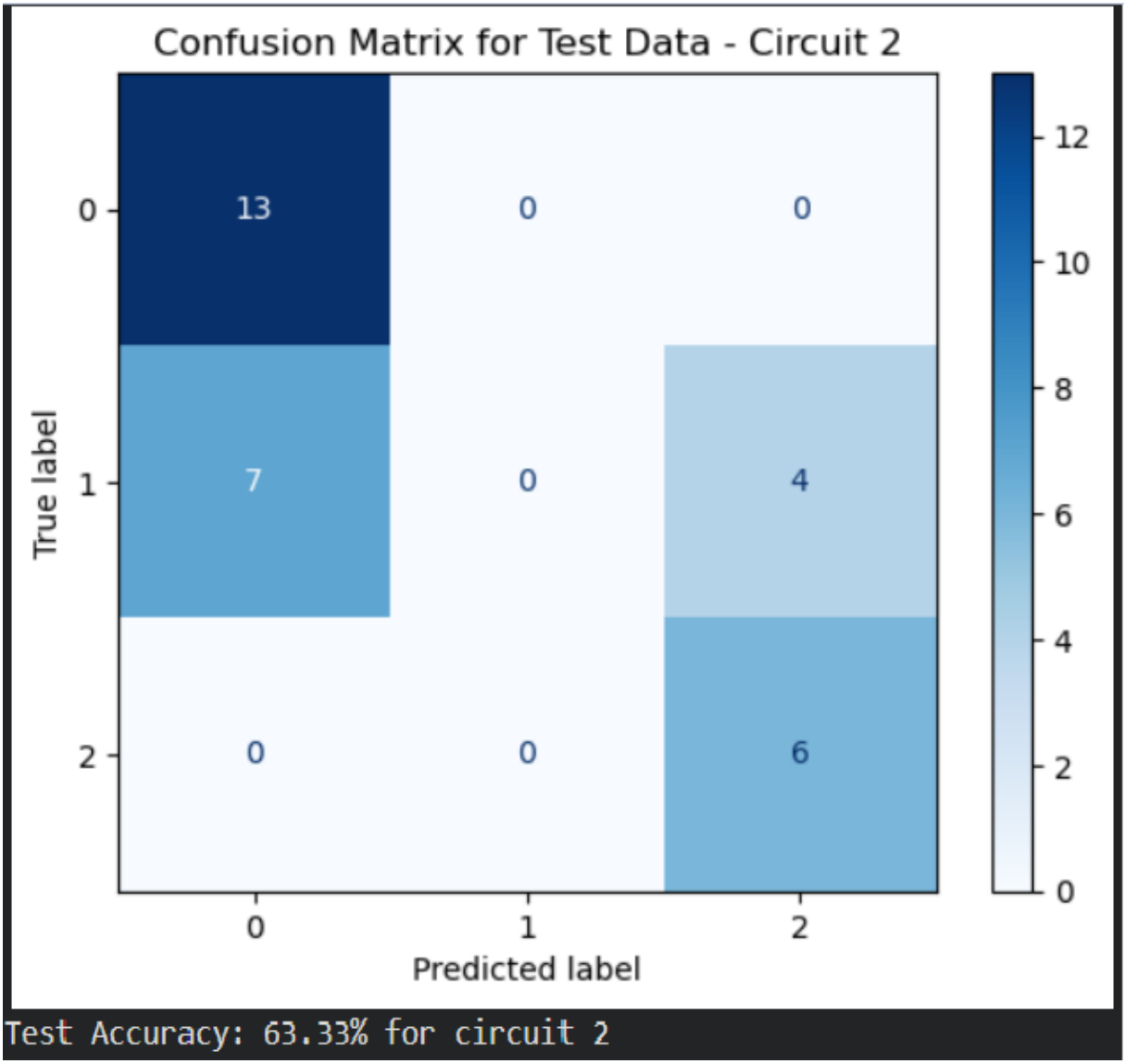

Figure 6: Results of first run Trained 8-qbit circuit on unseen data

```
Training time for circuit 2 with 8 qbits is: 2485 seconds
```

Discussion

After the first run this is results:

	Circuit 1	Circuit 2
Test Accuracy	80%	63.33%
Difference from last run	0%	0%

It seems that using a similar circuit on 8 qbits instead of 4 qbits gives a worse accuracy. It also takes about 10 times longer to learn the 8-qbit circuit. Given that the Iris dataset is a dataset that can easily be predicted with simple models the results are not sufficient. Some hyperparameter tuning is needed.

The second setup

Changes

- Changed the data split
- Change learning rate of 8-qbit circuit to 0.1 to make it converge faster.
- Increase gradient descent epochs from 50 to 75.
- Changed the 8-qbuit circuit.

Data

Instead of splitting into 60/20/20 Training/test/validation data. The data gets split into 70/15/15 Training/test/validation data. This results in more training data for the Quantum neural networks to train on.

Circuits

Removed The Ry gates at the end of the 8-qbit circuits so that both circuits are the same exept for amount of qbits. It now looks like this:

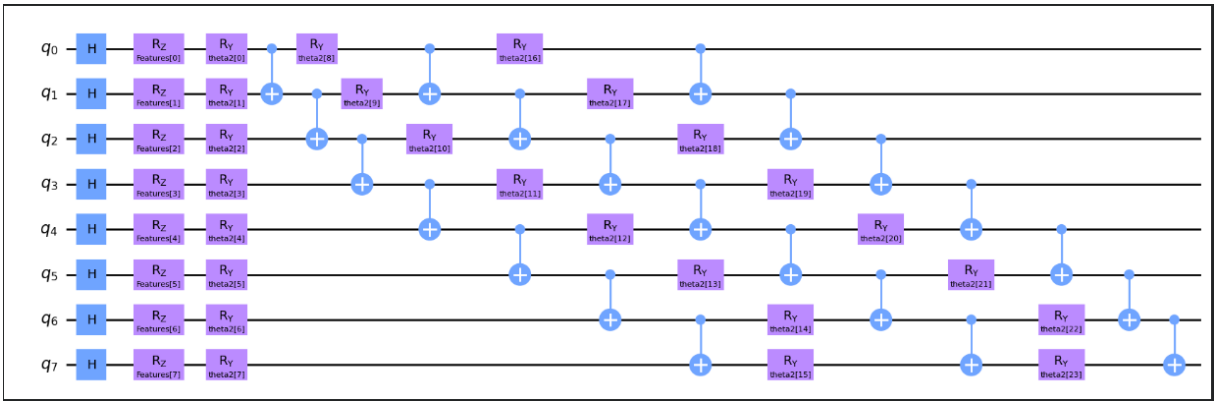

Figure 7: changed 8-qbit circuit

Results

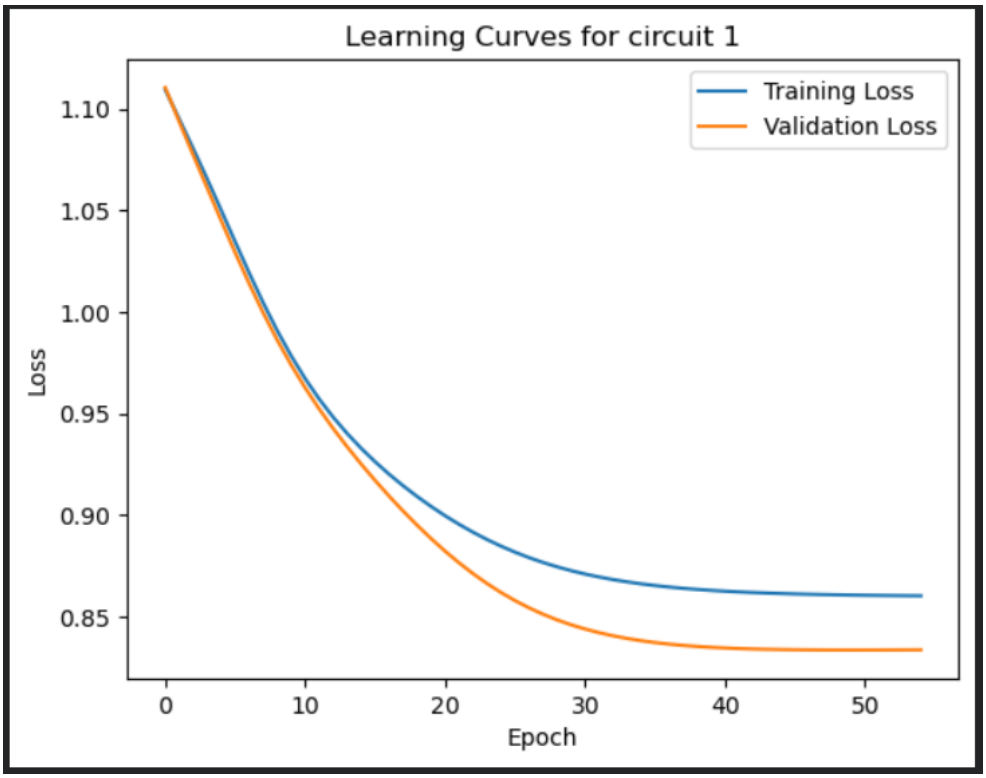

Figure 8: Learning curves for second run 4-qbit circuit

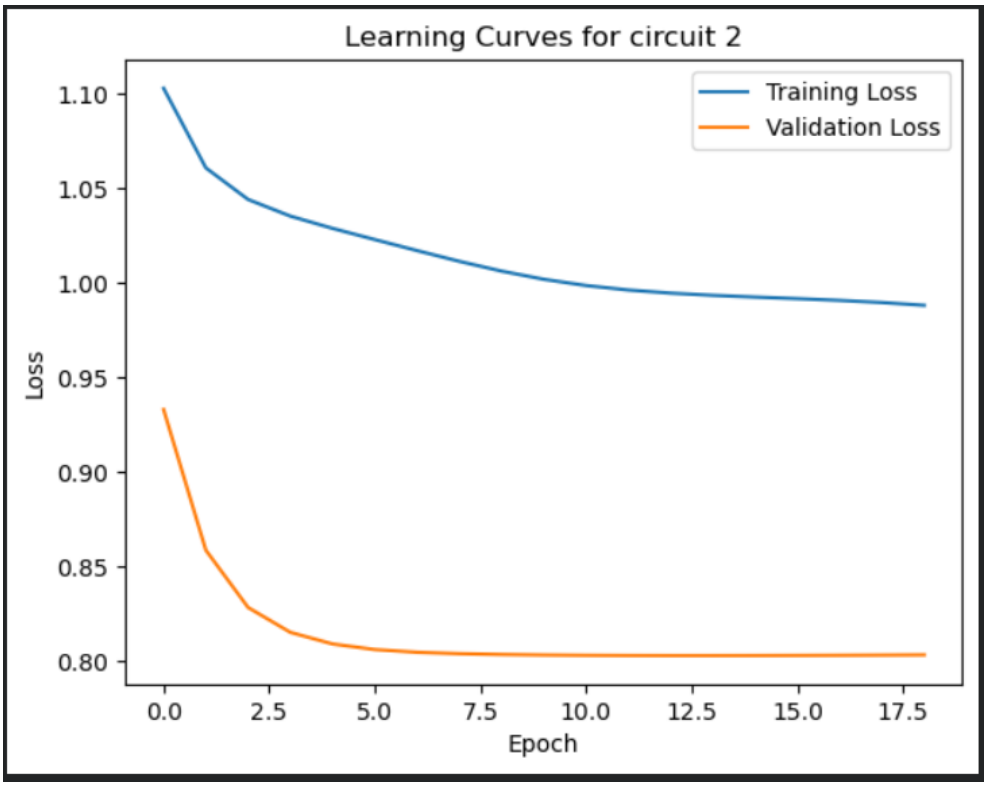

Figure 9: Learning curves for second run 8-qbit circuit

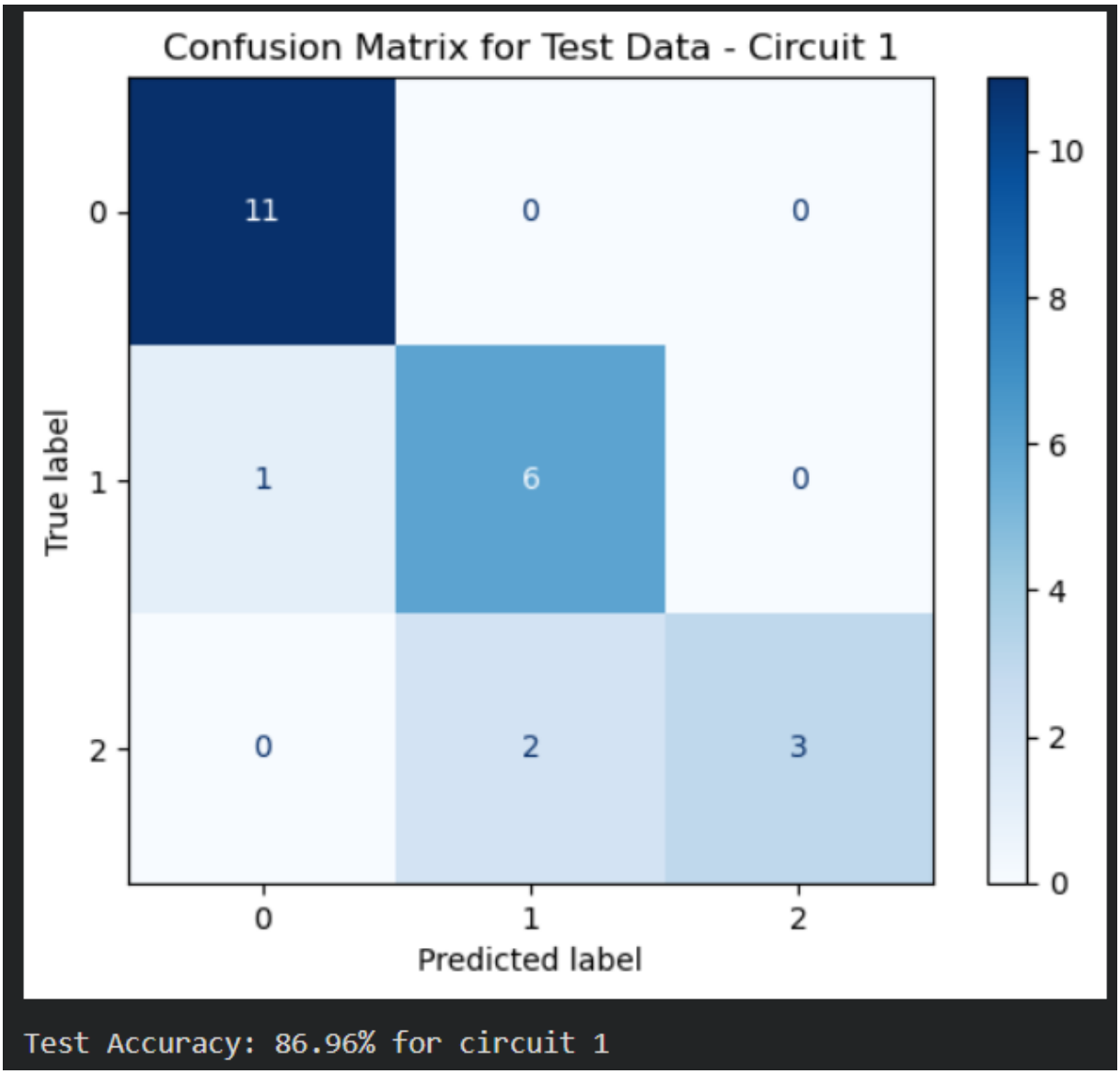

Figure 10: Results of second run Trained 4-qbit circuit on unseen data

```
Training time for circuit 1 with 8 qbits is: 434 seconds
```

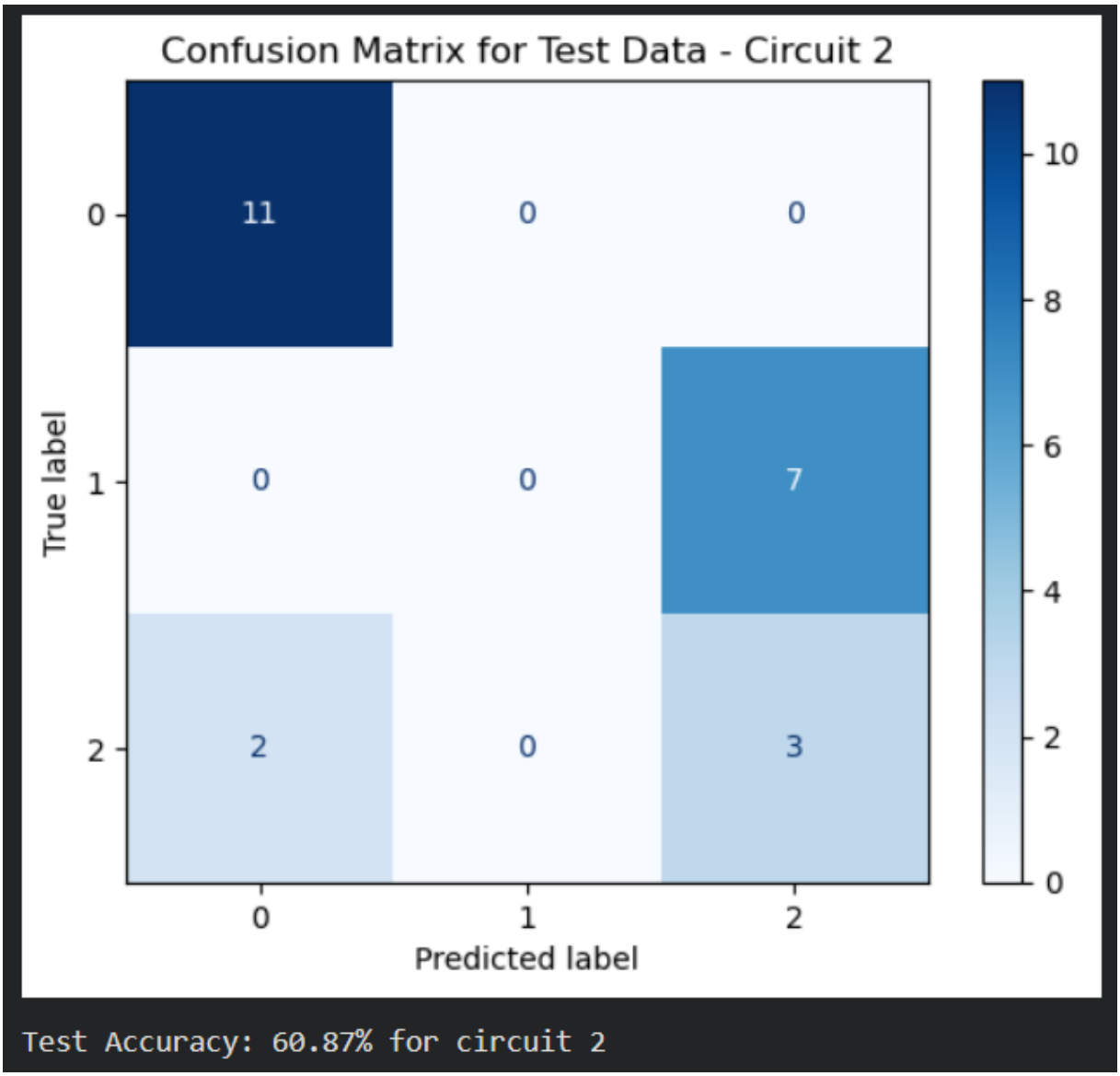

Figure 11: Results of second run Trained 4-qbit circuit on unseen data

```
Training time for circuit 2 with 8 qbits is: 731 seconds
```

Discussion

After the second run this is results:

	Circuit 1	Circuit 2
Test Accuracy	86.96%	60.87%
Difference from last run	+ 6.96%	- 2.46%

The accuracy of the 4-qbit QNN increased with a good 6.96% from last run. While the 8-qbit QNN decreased with 2.46%. The training time for the 8-qbit QNN was significantly faster going from 42 minutes to 12 minutes. While the training time for the 4-qbit QNN increased slightly but in turn gave better accuracy. It seems like some circuit tuning is needed.

The final setup

For the sake of the report this is the last setup. Tuning the circuit took by far the longest time. The result in this section is the best results after tuning.

Changes

For the final setup the changes mainly consist of changes to the circuits. Also decreased epochs from 75 to 50 for the 8-Qbit circuit. Since this one takes quite a long time to train.

Data

The data stays the same as last setup. With the same split.

Circuits

After several changes to the rotations and entanglements the eventual best circuit for the 4-qbit QNN came to be a simple change of the last Ry gates over to Rx gates.

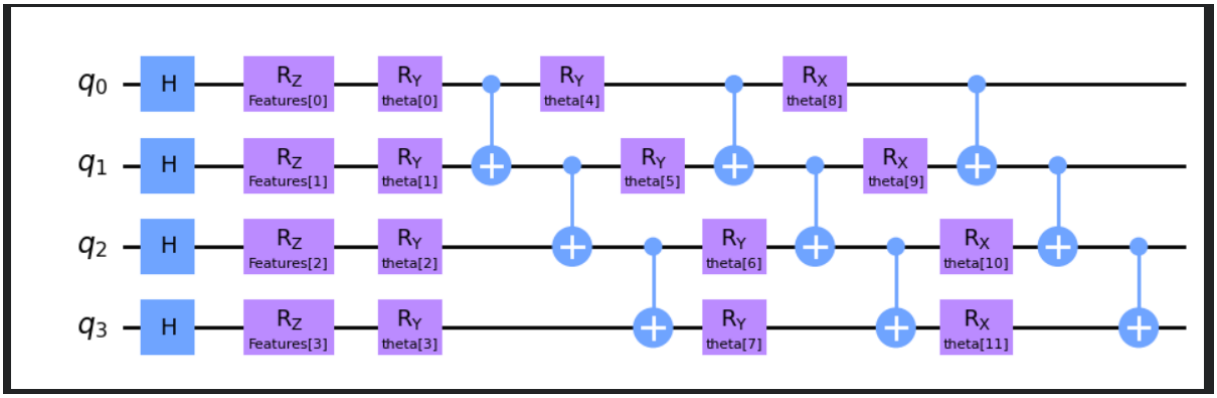

Figure 12:Changed 4-qbit circuit

As for the 8-qbit QNN. The best circuit consisted of a bit more changes. With alternating Ry and Rx rotations as well as Cx and Cz entanglements.

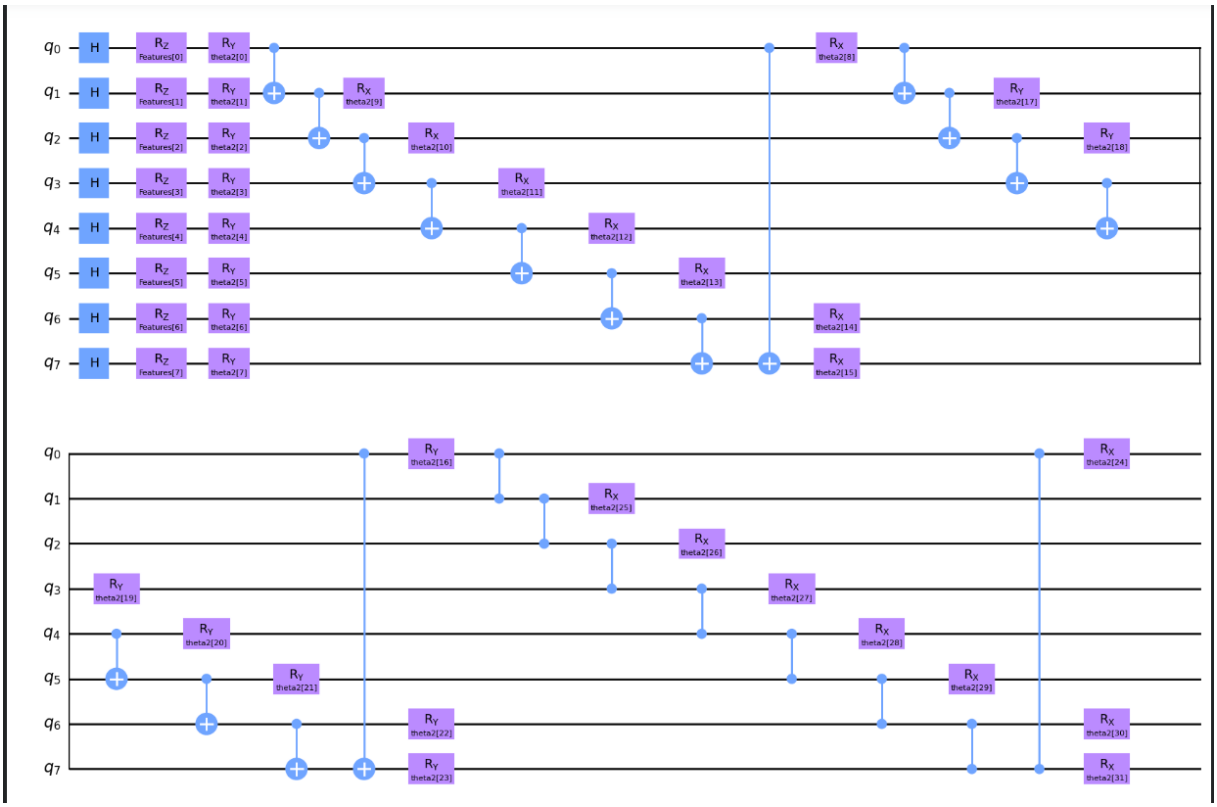

Figure 13: Changed 8-qbit circuit

Results

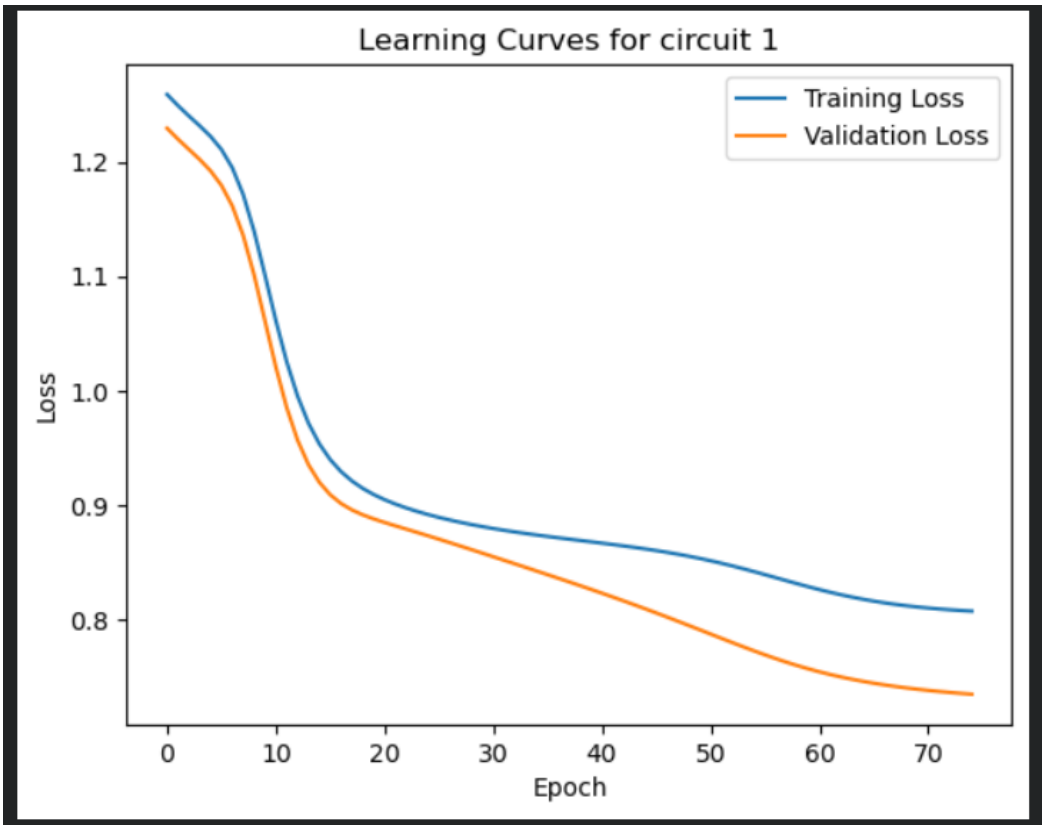

Figure 14. Learning curves for third run 4-qbit circuit

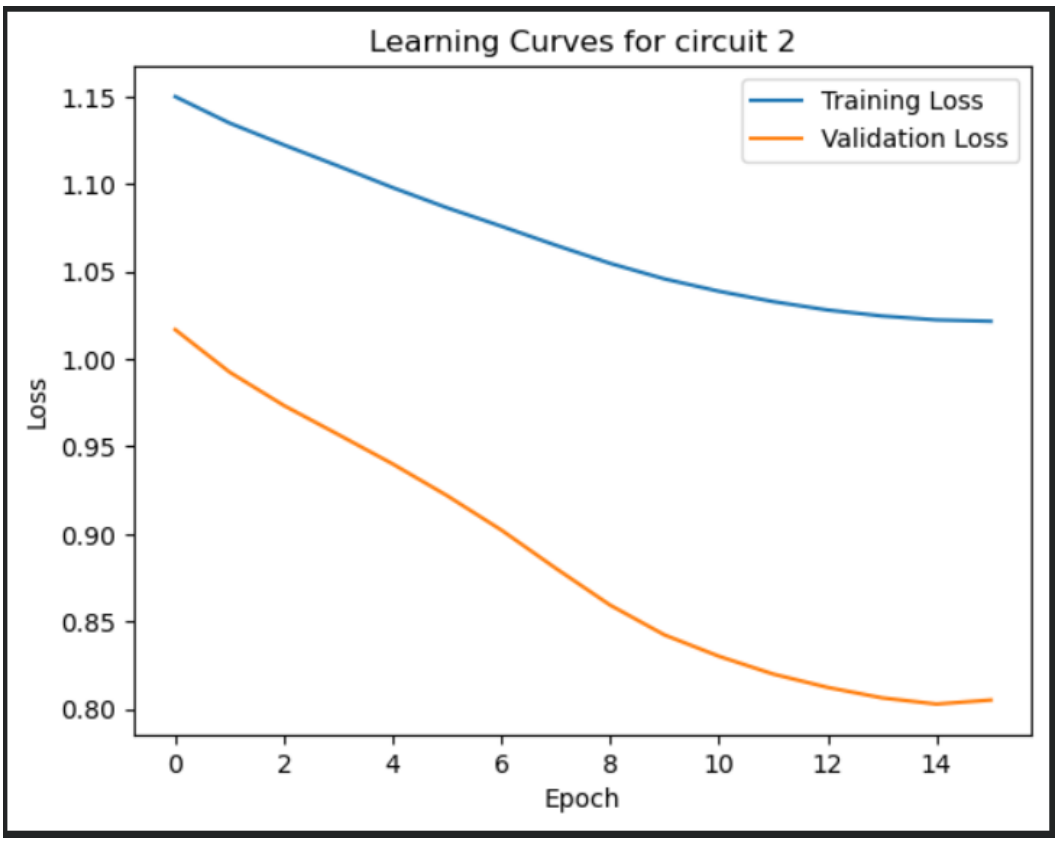

Figure 15: . Learning curves for third run 8-qbit circuit

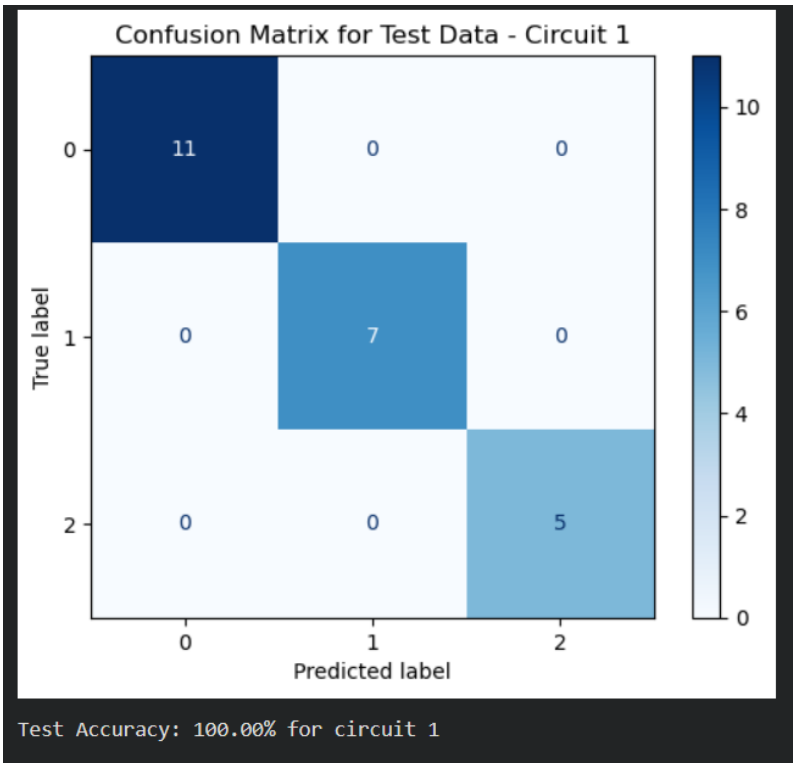

Figure 16: Results of third run Trained 4-qbit circuit on unseen data

```
Training time for circuit 1 with 8 qbits is: 603 seconds
```

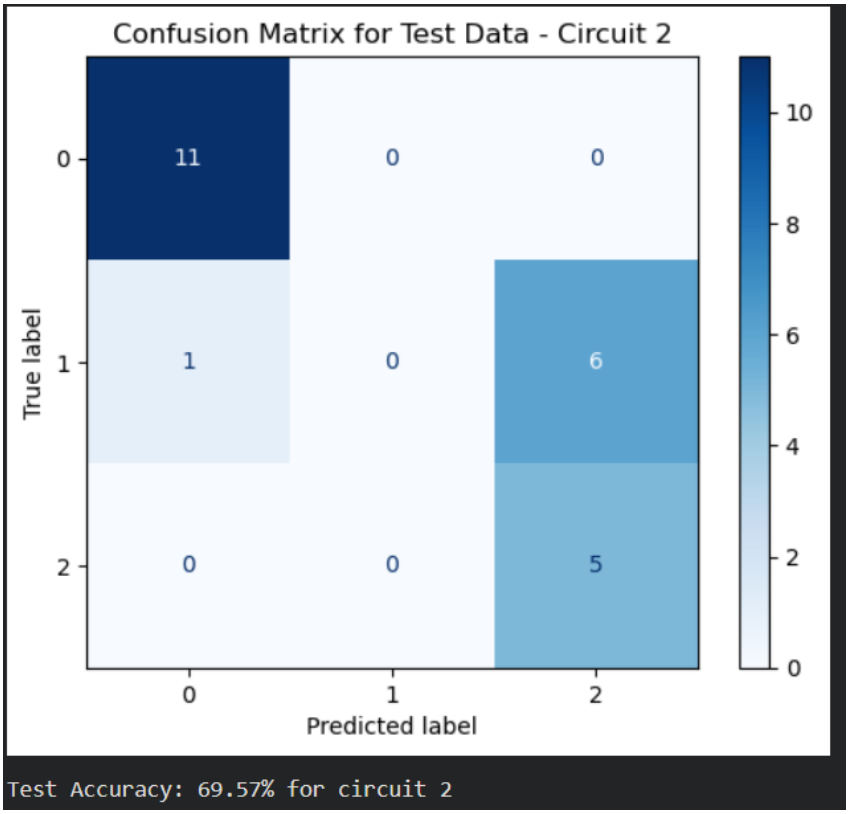

Figure 17: Results of third run Trained 8-qbit circuit on unseen data

```
Training time for circuit 2 with 8 qbits is: 926 seconds
```

Discussion

After the third and final run this is the results:

	Circuit 1	Circuit 2
Test Accuracy	100%	69.57%
Difference from last run	+ 13.04%	+ 8.7%

After the final tuning of the circuit, the results start to look satisfactory. For the 4-qbit qnn the result is 100% on the test accuracy, this is a 13.04 increase in accuracy. The confusion matrix also shows this. Training a good 8-qbit Qnn proves to be more difficult. After many tests, it seems that with the current setup that a test accuracy of more than 70% Is hard to achieve. The 8-qbit qnn seems to struggle with classifying the “versicolor” flower (label 1). However, because of time constraints this will have to suffice.

Final thoughts

Overall training a Qnn and trying to find the best hyperparameters have been insightful. It has also taught me a bit about neural networks both quantum and conventional ones. Finding the best circuit often takes a long time when working with more qubits, this was a bit frustrating. Because of this it was easier to find a good circuit on 4-qbits compared to finding a good circuit on 8-qbits. Overall, the project has been instructive.

Two unexpected things from the project:

1. Our training loss is higher than validation loss.
2. 4-qbit Qnn perform better than 8-qbit Qnn

Both unexpected outcomes might be the result of mistakes in the code. However, after looking through the code, I cannot seem to find any bugs causing this. Another explanation might be the size of the dataset, the choice of random seed or a undetected data leak.

Also getting 100% test accuracy from just switching the last set of rotation gates from Ry gates to Rx gates was a bit unexpected.