

**CZ3005 Lab 2 Report**

**Cheong Jun Hao, U1921202F, FS9**

**Table of contents**

[**Question 1**](#_62mrvouj8zci) **3**

[**Question 2**](#_6el33ahdczo3) **4**

[Hidden Nodes](#_unvm5isv0esc) 4

[Findings](#_q0fbk7awzu3h) 5

[Hidden Layer](#_q90s0wku0ygm) 6

[Findings](#_bqhyed6wgjr) 7

[**Question 3**](#_u20yh8tgnq5l) **7**

[Findings](#_p8mv7mqh4b37) 9

[**Question 4**](#_vl4au46b8jdl) **9**

[Findings](#_sryp7upriit3) 10

[**Conclusion**](#_9zsudic88ow6) **10**

# Question 1

For question 1, I was tasked to create a three layer feed-forward neural network. This is to solve the monitoring control problem of the injection moulding machine. To start off, data was read from the mat file into google colab. After that the data was segmented into train and test sets in the proportion of 70:30 as stated in the question. The construction of the neural network was then done. For my neural network model, I have chosen to have 100 hidden nodes in each hidden layer, batch size of 30, 200 epoch and the learning rate was kept constant at 0.02. With this, I managed to achieve 0.0 percent training error and 0.0 percent test error as seen in fig 1.

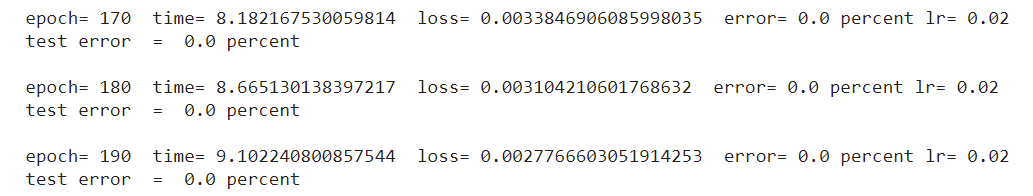


Fig1. Question 1 model training

This result is also reflected when the model is tested with random test data achieving accuracy of 98% as seen in fig 2.

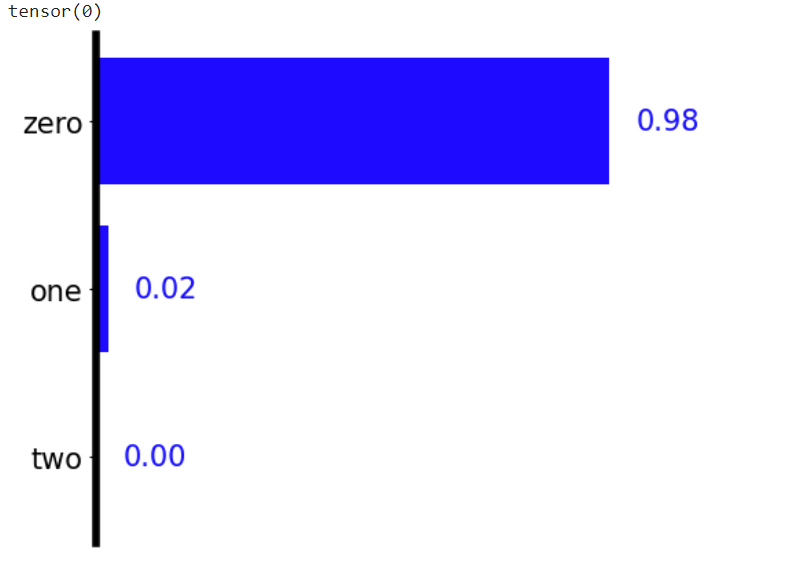


Fig2. Test with random test data

# Question 2

For question 2, the motive is to find out the effect of different network structure to the classification problem. I focused on 2 areas, Hidden nodes and Hidden layer. The epoch is maintained at 200. The batch size is kept at 100 for testing the number of hidden nodes and 40 for testing the number of hidden layers. FInally, the learning rate is kept constant at 0.02.

## **Hidden Nodes**

To find out the effect of hidden nodes, I tested with 50, 100, 150 and 200 hidden nodes in each hidden layer. For 50 nodes, it can be observed that at 190 epoch, the training error is at 0.333% while testing error is at 0.0%. It also took 3.01 seconds to train this model with 50 nodes in each hidden layer.

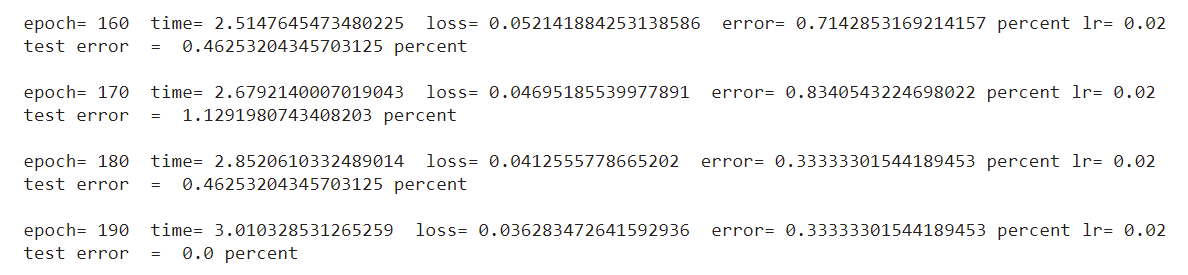


Fig3. Results of training 50 nodes in each hidden layer

Following this, there are 100 nodes in each layer. At 190 epoch, the training error is at 0.0% and the testing error is 0.0%. It took 3.51 seconds to train this model with 100 nodes in each hidden layer.

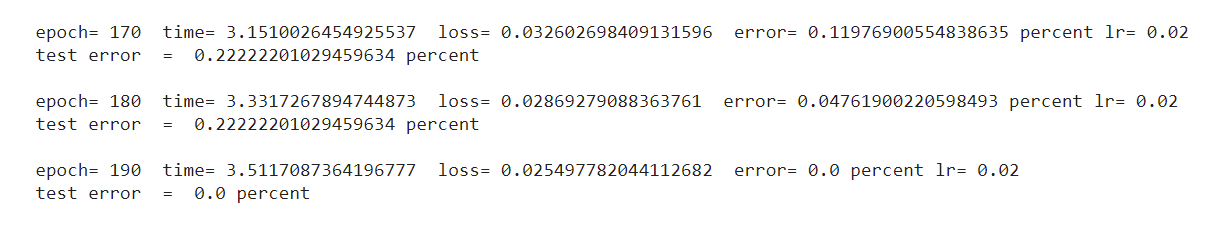
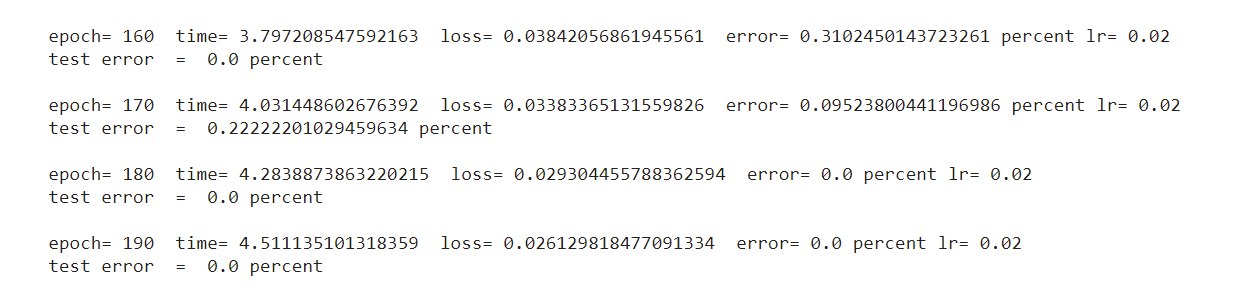


Fig4. Results of training 100 nodes in each hidden layer

Now, I tested with 150 nodes in each layer. At 190 epoch, the training error is at 0.0% and testing error is 0.0%. The time taken is 4.51 seconds to train this model with 150 nodes in each hidden layer. Fig5. Results of training 150 nodes in each hidden layer

Lastly, there are 200 nodes in each layer. At 190 epoch, the training error is at 0.0% and testing error is 0.0%. The time taken is 5.39 seconds to train the model with 200 nodes in each hidden layer.

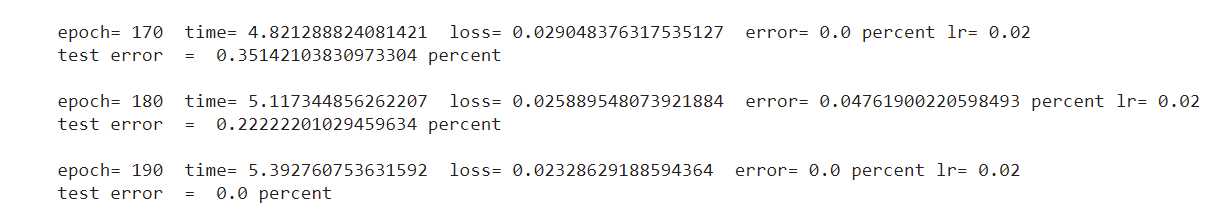


Fig6. Results of training 200 nodes in each hidden layer

## **Findings**

Based on the results, patterns can be observed. Firstly, it can be seen that the complexity is worse as the number of hidden nodes increases. This is observed by the increase in time taken to reach 190 epochs from 3.01 seconds for 50 nodes to 5.39 seconds for 200 nodes. It can also be observed that the accuracy of the model increases. Training error decreases from 0.333% at 50 nodes in each hidden layer to 0.0% at 200 nodes in each hidden layer and testing error remaining at 0.0% for 50,100,150 and 200 nodes in each hidden layer. It is also observed that loss has decreased from 0.026 at 50 nodes in each hidden layer to 0.023 at 200 nodes in each hidden layer.

## **Hidden Layer**

After testing with hidden nodes, I observed the effects of changing the number of hidden layers. I tested with 0,1,2,3 hidden layers. I first tested with 0 hidden layers. At 190 epoch, the training error is at 4.62%, while the test error is at 4.99%. The time taken is 4.09 seconds for training the model with 0 hidden layer.

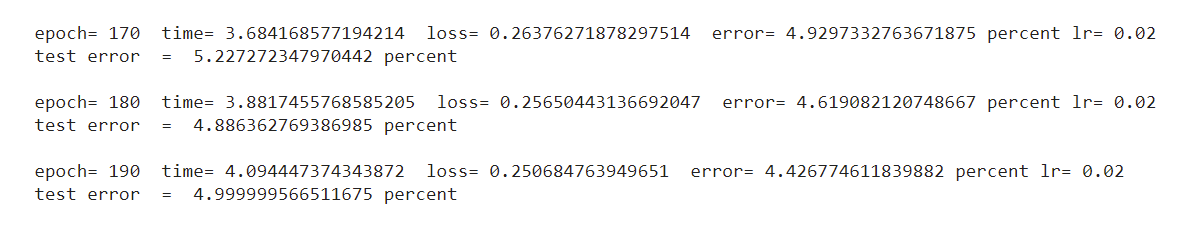


Fig7. Results of training with 0 hidden layer

Next, I tested with 1 hidden layer. At 190 epoch, the training error is at 0.048%, while the test error is at 0.0%. The time taken is 5.29 seconds for training the model with 1 hidden layer.

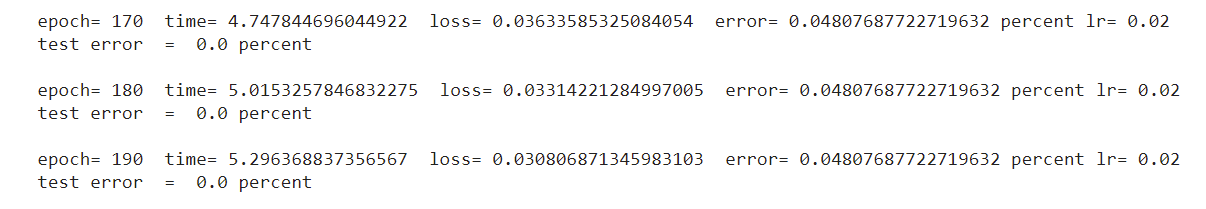


Fig8. Results of training with 1 hidden layer

Following that, I tested with 2 hidden layers. At 190 epochs, the training error is at 0.0%, while the test error is at 0.0%. The time taken is 6.69 seconds for training the model with 2 hidden layers.

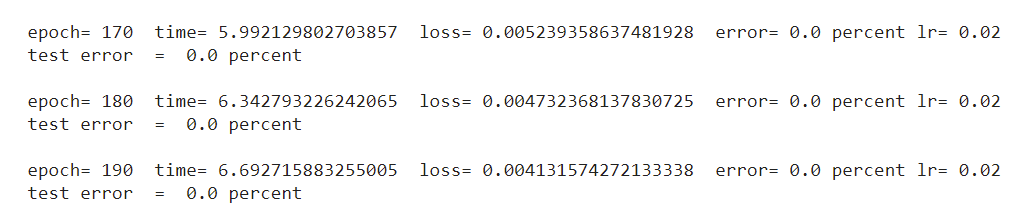


Fig9. Results of training with 2 hidden layer

Finally, I tested it with 3 hidden layers. At 190 epochs, the training error is at 0.0%, while the test error is at 0.0%. The time taken is 8.04 seconds for training the model with 3 hidden layers.

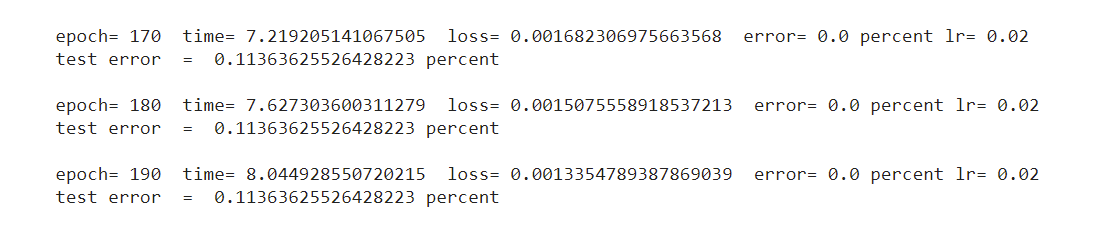


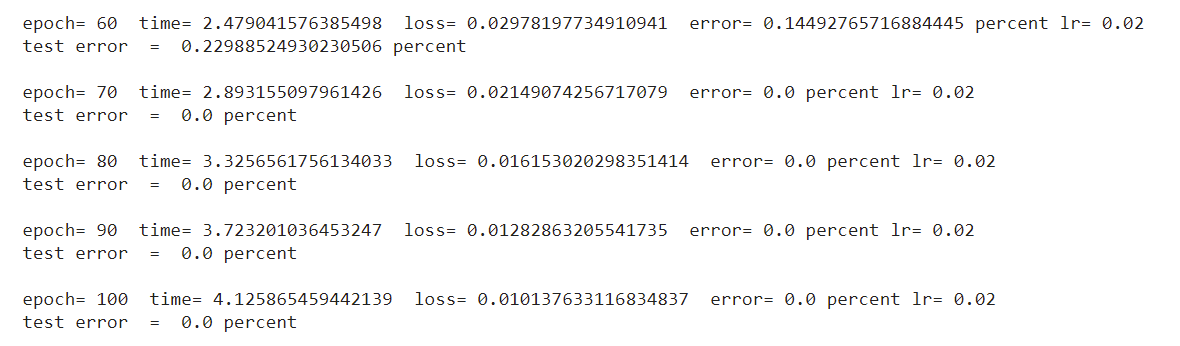
Fig10. Results of training with 3 hidden layer

## **Findings**

Based on the result, patterns can be observed. The complexity gets worse as the number of hidden layers increases. When the number of hidden layers is 0, the time taken is 4.09 and it increases more as the number of layers increases up till 8.04 seconds at the 3rd layer. Next, the accuracy of the training model increases when the number of layers increase, this is seen when the training error is 4.62% and testing error was 4.99% when the number of hidden layers is 0 and it will decrease as the number of hidden layers increases up until 0% for training and testing errors for 3 hidden layers. The loss can also be observed as decreasing from 0.25 at 0 hidden layers to 0.0013 at 3 hidden layers. It can also be noted that the effect of having more hidden layers is more significant than having more hidden nodes on the classification problem.

# Question 3

For question 3, I am studying the effect of the learning rate. The variable to be tested is the learning rate, while the epoch size remains at 200 and batch size is 30. The three layer feed forward neural network is used with 60 hidden nodes in each hidden layer. For the learning rate of 0.02, the training error is 0% from epoch 60 and the testing error is also 0% from epoch 60. It took 7.79 seconds to train the model up to 190 epoch with a learning rate 0.02.



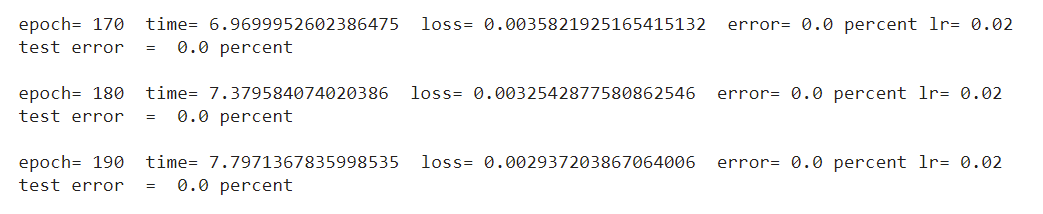
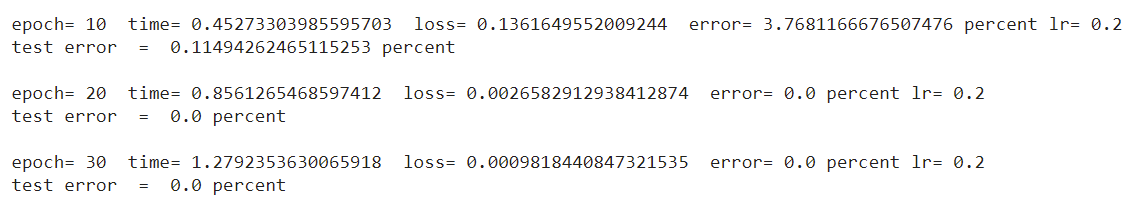


Fig11. Results of training model with learning rate 0.02

Next, I test with a learning rate of 0.20. The training error and testing error is 0% from epoch 20. The time taken to train the model with the learning rate of 0.2 is 7.73 seconds running up to 190 epoch.



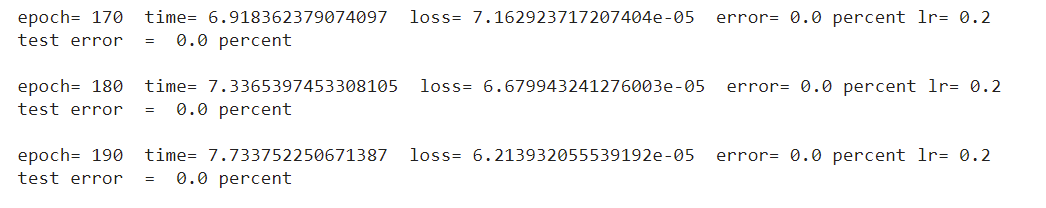


Fig12. Results of training model with learning rate 0.20

I experimented with increasing adaptive learning rate too. At 190 epoch, the training error and testing error is 0%. The time taken to train the model is 7.79 seconds for the increasing learning rate starting from 0.01.

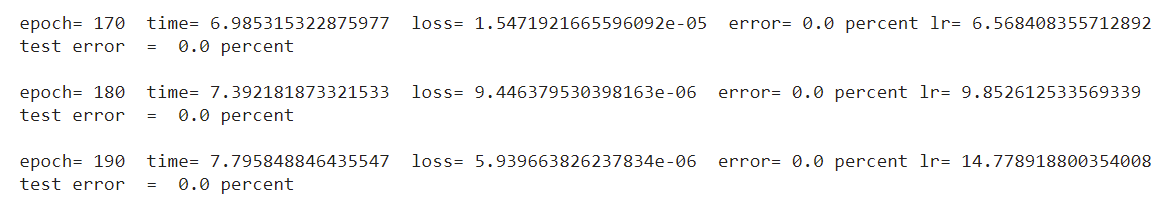


Fig13. Results of training model with increasing adaptive learning rate

I also tried with decreasing adaptive learning rate. At 190 epoch, the training error is 6.38% while the testing error is 5.97%. The time taken to train the model is 7.87 seconds for the decreasing learning rate starting from 0.01.

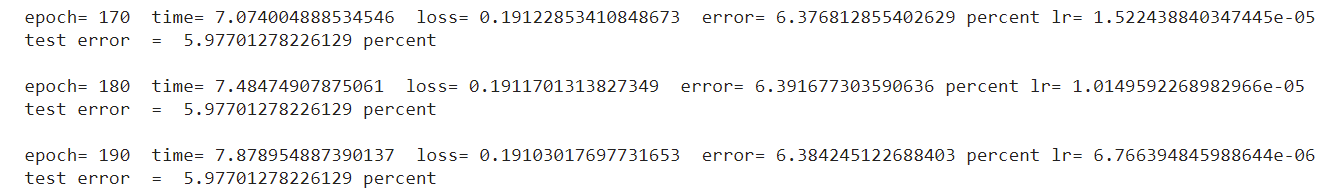


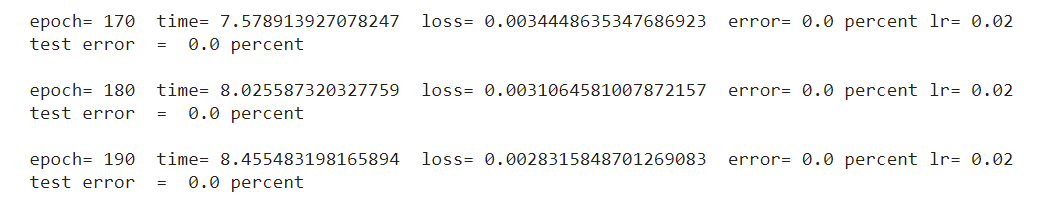
Fig13. Results of training model with decreasing adaptive learning rate

## **Findings**

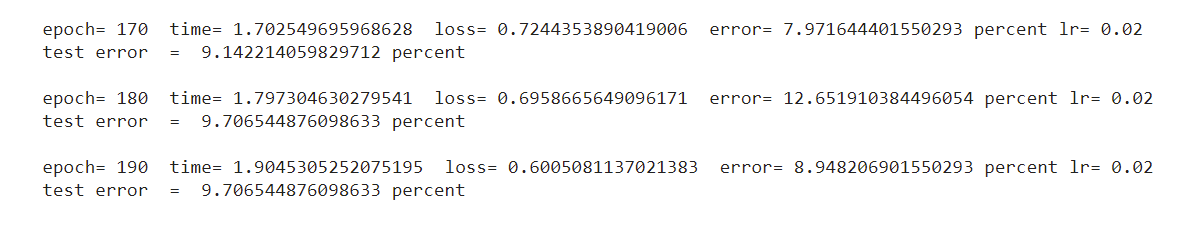
From question 3, it can be observed that the complexity for different learning rates is around the same. The time taken to train the model for learning rate of 0.02 is 7.79 seconds which is not much difference compared to the 7.73 seconds taken to train the model with learning rate of 0.2. The accuracy of the model is similar as they both have training and testing error at 0% at epoch 190. However, it can be observed that the model with higher learning rate reached 0% faster than the model with lower learning rate. As such, it may be said that the high learning rate may be better for training the model given the batch size and hidden layer that is used. Now, the increasing adaptive learning rate is also seen to be more accurate than the decreasing adaptive learning rate as the training error for the increasing adaptive learning rate is 0% compared to the 6.38% for decreasing adaptive learning rate. The test error for increasing adaptive learning rate is 0% which is lower than the testing error of 5.97% for decreasing adaptive learning rate. The complexity for the increasing and decreasing adaptive learning rate is around the same. The increasing adaptive learning rate took 7.79 seconds to reach 190 epochs which is close to 7.87 seconds taken by the decreasing adaptive learning rate to reach 190 epochs. Given the batch size, hidden layers and hidden nodes that are used, increasing adaptive learning rate would be preferred over decreasing adaptive rate.

# Question 4

For question 4, I am studying the effect of batch size. The variable to be tested is the batch size, while the epoch stays at 200 and the learning rate is kept at 0.02. A three-layer feed forward neural network is used with 100 hidden nodes in each hidden layer. For a batch size of 30, the training error and testing error is 0.0% at 190 epochs. The time taken to train the model is 8.45 seconds for batch size 30.



Next, a batch size of 1024 is used. At 190 epoch, the training error is 8.98% and the testing error is 9.706%. The time taken to train the model is 1.90 seconds for a batch size of 1024.



## **Findings**

Based on the results, it can be observed that the complexity gets better as the batch size increases. This is seen when the time taken to reach 190 epochs reduces from 8.45 seconds for batch size of 30 to 1.9 seconds for batch size of 1024. The accuracy is reduced as the batch size increases. The training error increases from 0.0% to 8.98% while the testing error increases from 0.0% to 9.706%. The loss can be seen as increasing as batch size increases. As seen in the results, the loss is at 0.0028 for batch size of 30 and it increased to 0.6 for batch size of 1024.

# Conclusion

Through this experiment, patterns can be observed when changing the batch size, learning rates, hidden nodes and hidden layers. There are more considerations which were not taken into account in this experiment. For example, for a larger batch size, increasing the learning rate would benefit the training of the model. The problem of overfitting also exists when the neural network has too many layers. Another consideration is if the learning rate is too high, the model will converge too quickly to a suboptimal solution. Hence, we can observe patterns and may try to take measures to reduce the error as much as possible but at the same time we have to factor in other issues overfitting and the relationship between learning rate and batch size.