NANYANG TECHNOLOGICAL UNIVERSITY

CZ3005 – Artificial Intelligence

Semester 1 Academic Year 21/22

Assignment 1 Report

Name	Matric Number	Email Address
Neo Qi Xiang	U1921697H	qneo001@e.ntu.edu.sg

Task 1

You are asked to build a three-layer feed-forward neural network to solve the monitoring problem of injection molding machines. Your implementation must be in Pytorch and executable in Google Colab environments. The proportion of training and testing samples is 70:30 where your model must deliver the smallest testing error possible. In that case, you need to select the number of nodes of hidden layers, the number of epochs, the learning rates, the mini-batch size, etc. that lead to the smallest testing error. In this assignment, you have to use the SGD optimizer as exemplified in the lab materials under the mini-batch update fashion. The evaluation metric here is the classification error. No feature selection is allowed here.

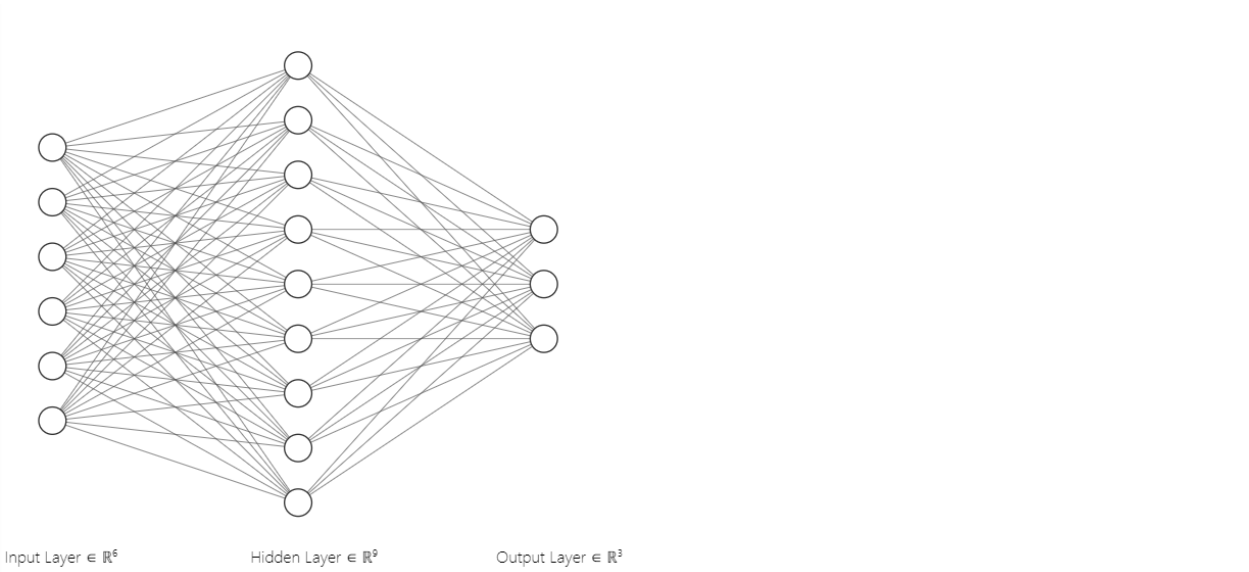
The task was to build a three layer feed forward neural network, I initialized a class of network using relu for the activation function. The sample code:

```
class Three_Layer(nn.Module):
    def __init__(self, input_size, hidden_size, hidden_size2, out_size):
        super().__init__()
        self.linear = nn.Linear(input_size, hidden_size )
        self.linear2 = nn.Linear(hidden_size, hidden_size2)
        self.linear3 = nn.Linear(hidden_size2, out_size)


    def forward(self, xb):
        x = self.linear(xb)
        x = F.relu(x)
        x = self.linear2(x)
        x = F.relu(x)
        x = self.linear3(x)
        return x

```

Example of a three layer feedforward Neural network:



We have 48 neurons in the input layer representing the 48 data features and 3 neurons in the output layer each representing the target of our model.

After many trials of changing batch size, learning rate and number of hidden nodes. The best model amongst my experiment has the following features:

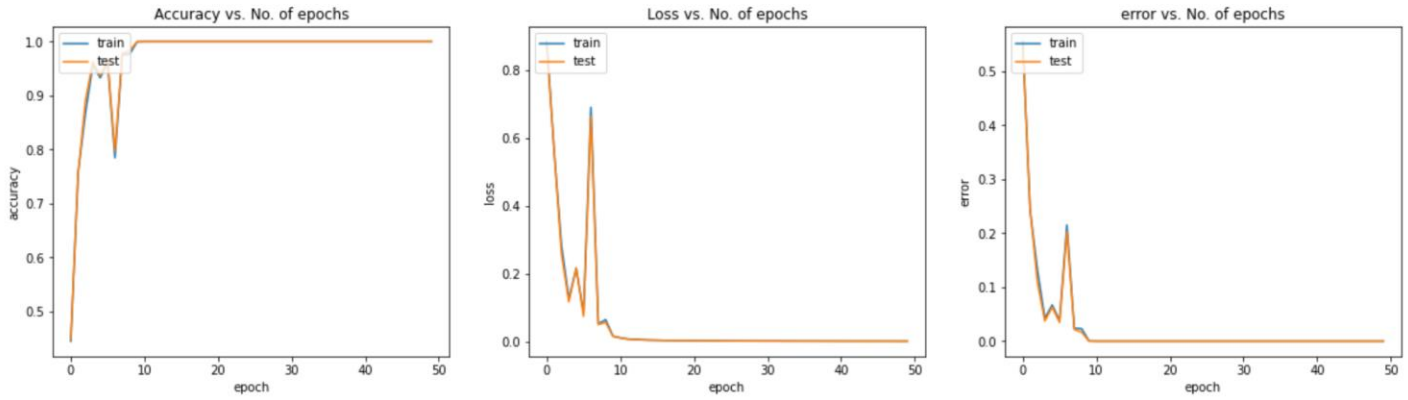
Learning rate using SGD optimizer: 0.05

Batch size: 8

Number of Neuron in the Hidden layer Neurons: 64

Statistics:

```
<matplotlib.legend.Legend at 0x1eb0181cbe0>
```



According to the graphs above, it takes about 10 epochs for the optimal model to be produced, with an error test rate of 0. This means that the model can successfully predict the three classes: normal, weaving and short forming based on the input feature.

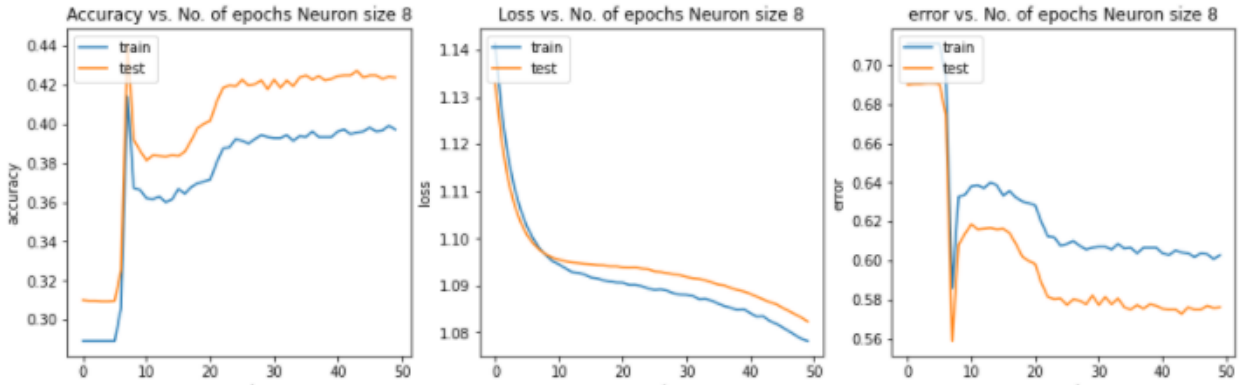
Task 2

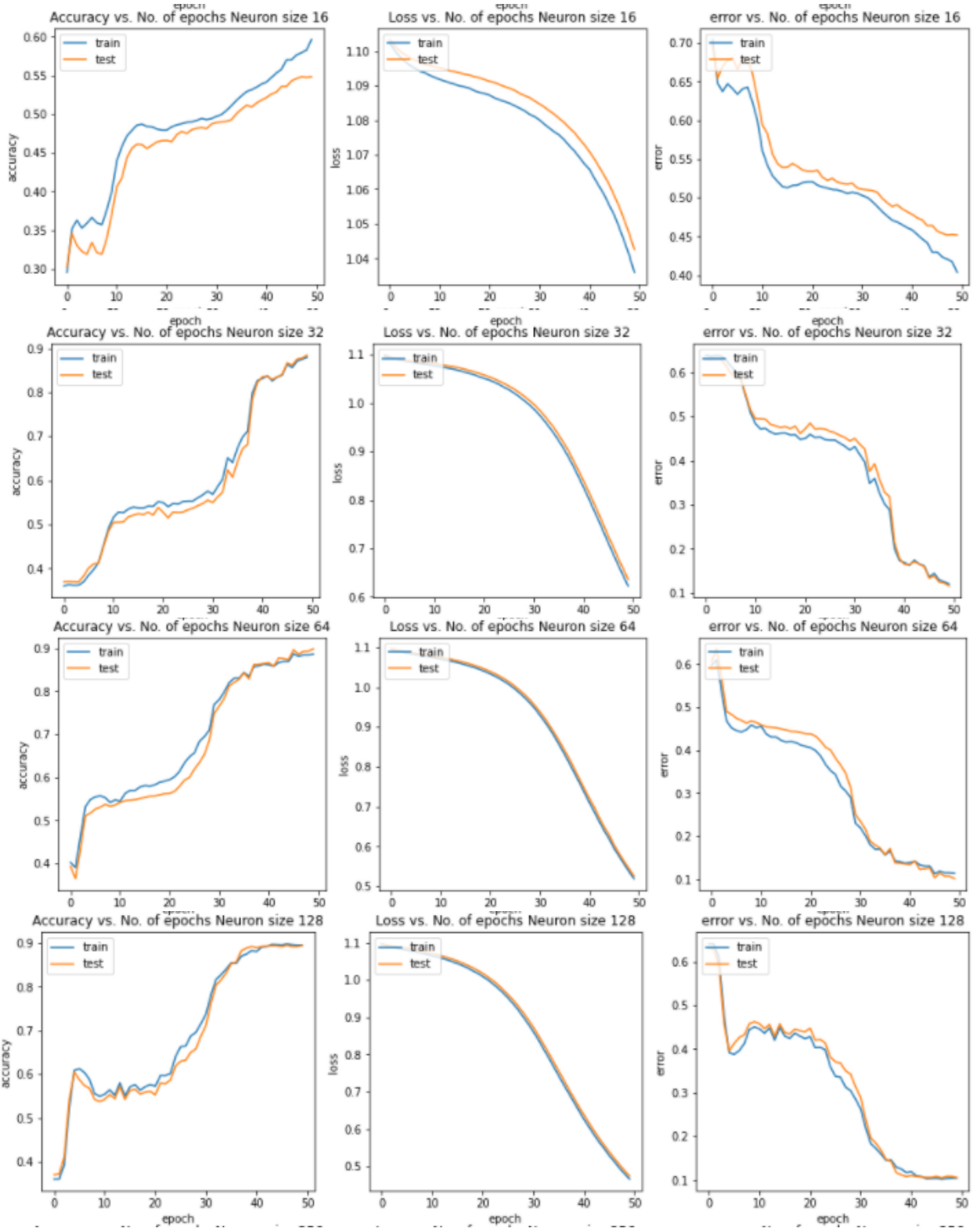
You are asked to study the effect of network structure: hidden nodes, hidden layers to the classification performance. That is, you try different network configurations and understand the patterns. Your experiments have to be well-documented in your Jupyter notebook file and your report. It has to cover different aspects of network configurations such as shallow network, wide network, deep network etc.

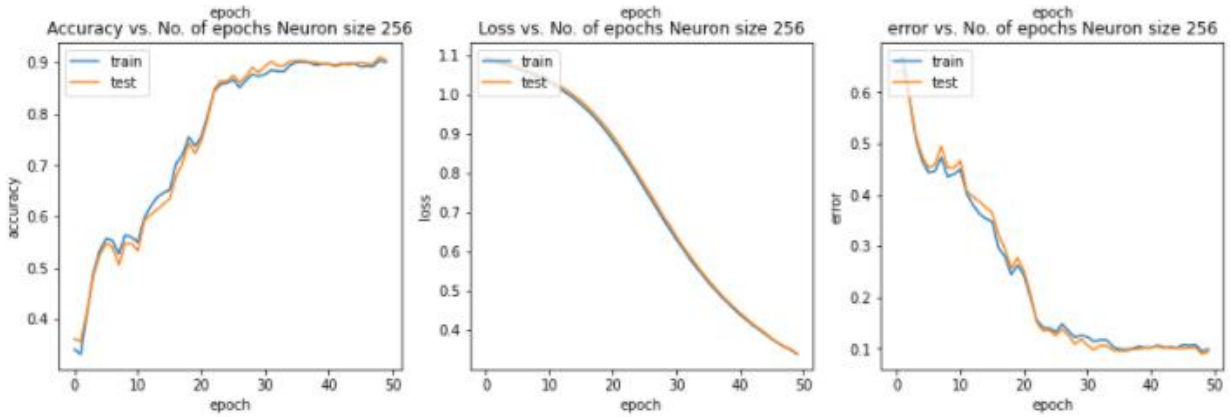
Wide vs Narrow network

To investigate the effect of a wide and narrow network, a three layer feed forward neural network is used with an input layer of 48 neurons and an output layer of 3 Neurons. The number of neurons in the hidden layer of the feed forward neural network is varied as such : [8,16,32,64, 128,256]. The loss, accuracy and error for each epoch is being recorded, in addition, the average time taken per epoch is recorded along with the number of epoch it takes to reach the smallest error rate.

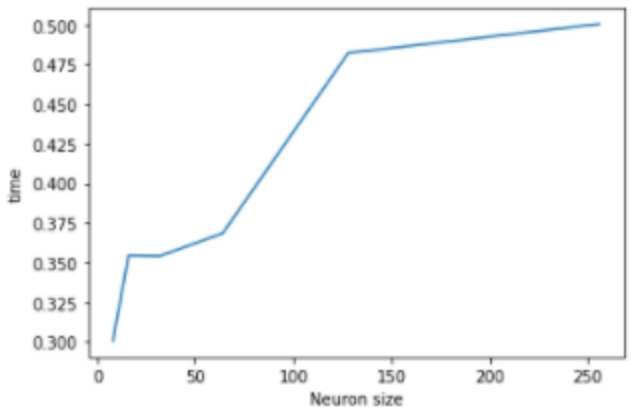
The result is shown below:




Accuracy vs. No. of epochs Neuron size 16
Loss vs. No. of epochs Neuron size 16
error vs. No. of epochs Neuron size 16
Accuracy vs. No. of epochs Neuron size 32
Loss vs. No. of epochs Neuron size 32
error vs. No. of epochs Neuron size 32
Accuracy vs. No. of epochs Neuron size 64
Loss vs. No. of epochs Neuron size 64
error vs. No. of epochs Neuron size 64
Accuracy vs. No. of epochs Neuron size 128
Loss vs. No. of epochs Neuron size 128
error vs. No. of epochs Neuron size 128
train
test
accuracy
loss
error
epoch



As seen in the graph above the test and train error and loss for the divergence between training and test experiment becomes lesser when the neuron size increases. A divergence between train and test could signify overfitting of data which is not desirable. It is also worthy to note that as the neuron size increases the number of epochs before the train and test data plateau also increases.



	Neuron Size	minimum error	epoch
0	8	0.558934	7
1	16	0.451952	47
2	32	0.115240	49
3	64	0.101351	49
4	128	0.105105	46
5	256	0.089715	48

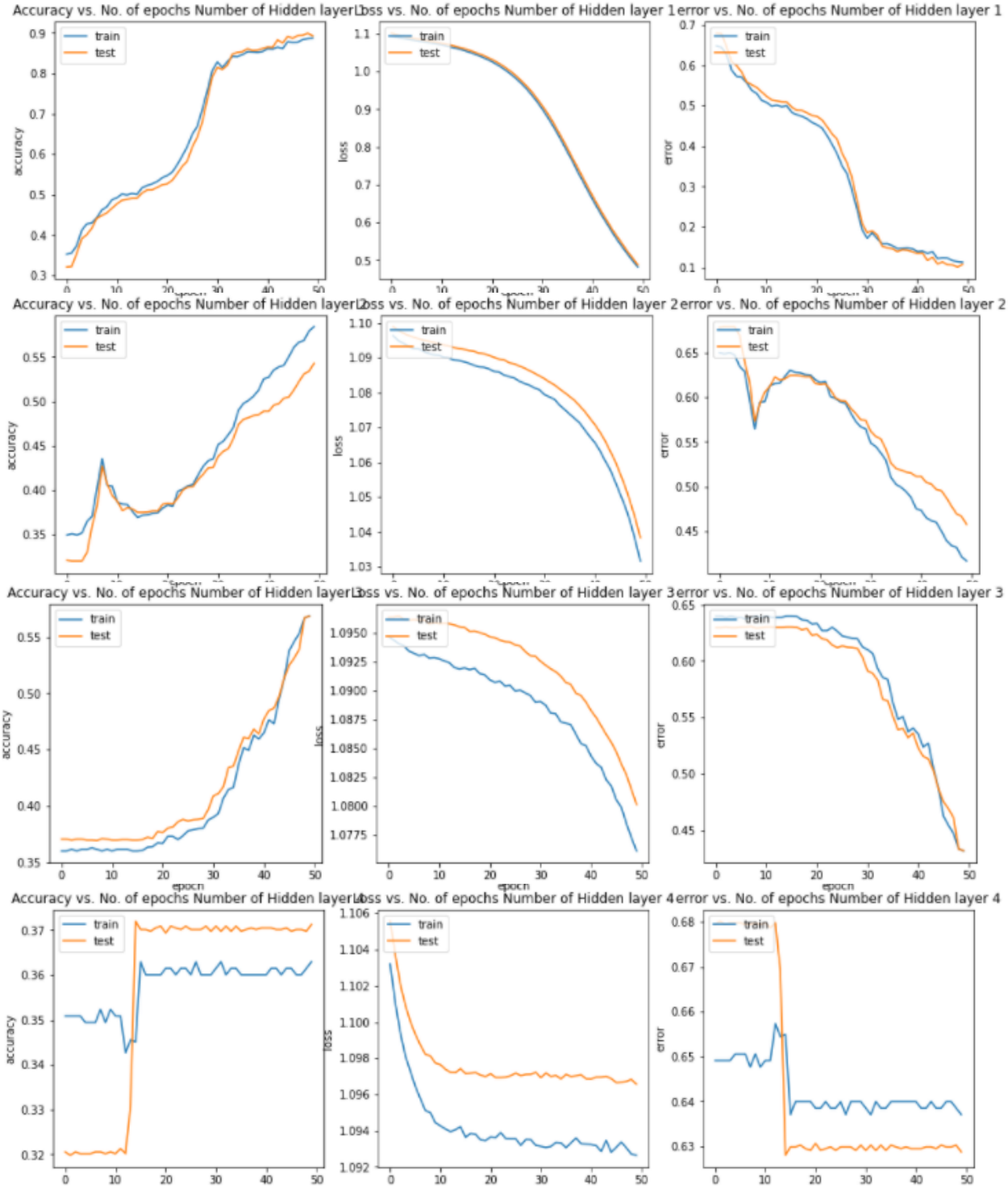
The graph on the left indicates the average time taken per epoch for each experiment, we can see that as the neuron size increases the training time increases. This is expected as the complexity of calculation increases as the number of neurons increases. Based on the result from the left table, the minimum error decreases as the number neuron size increases. The performance of the model actually became significantly better after 16 epochs. Two conclusions can be derived from the result, a wide network is a better model for this dataset. Second is when the hidden neuron size is lesser than the input layer, the network may not be effective as there could be a loss of information when the dimension is reduced from the first layer to the second layer.

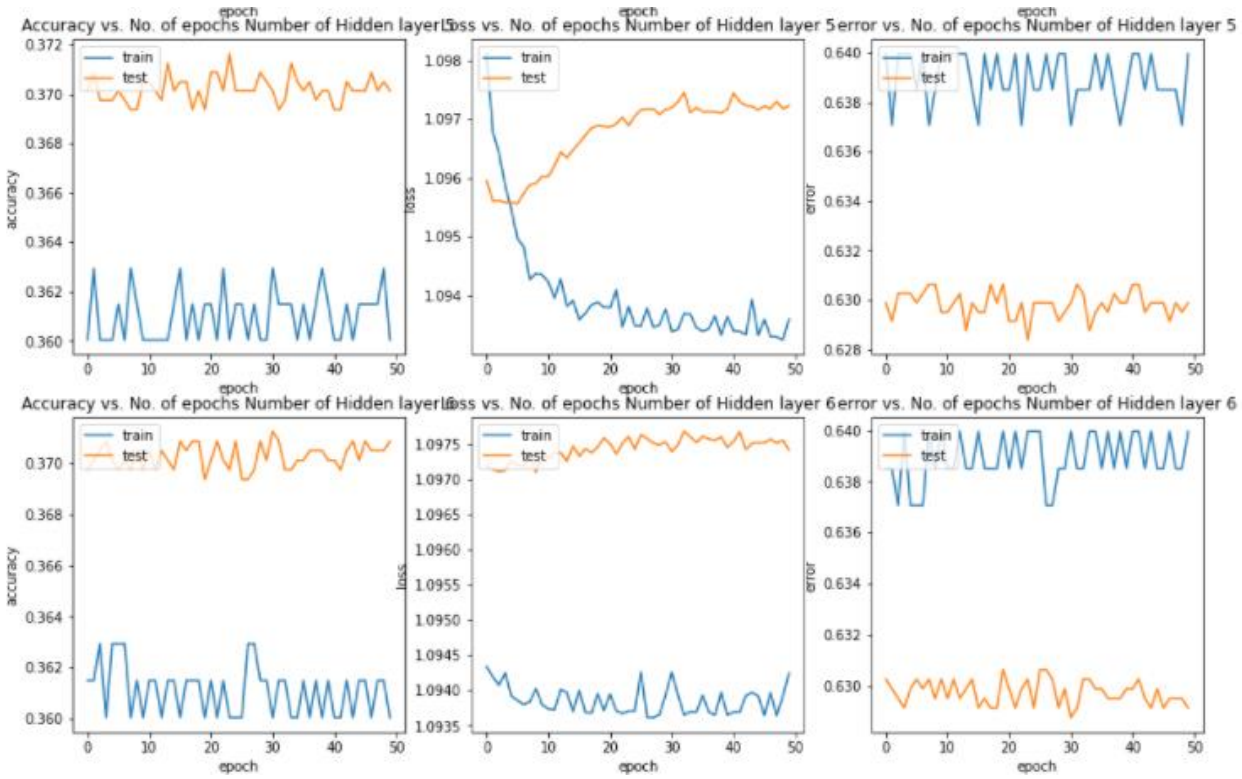
Shallow vs Deep Network

The aim of this experiment is to investigate how the number of layers in a neural network affect the performance of the model. The neuron size chosen for the hidden layers is 64 as the previous experiment shows that the error rate is amongst the lowest, in addition the training time per epoch is much lower than bigger neuron size. The number of layers varies from 1 hidden layer to 6 layers.

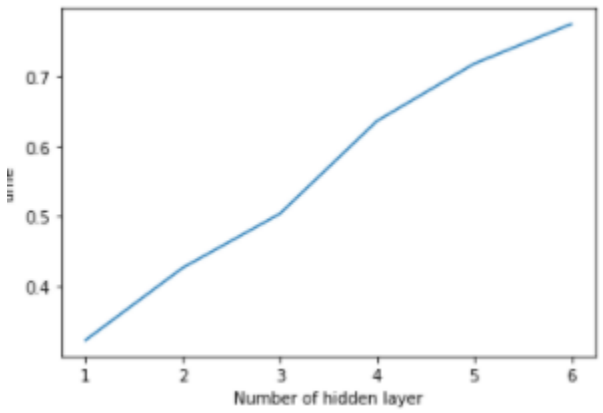
\

The results:


Accuracy vs. No. of epochs Number of Hidden layer 1
Loss vs. No. of epochs Number of Hidden layer 1
error vs. No. of epochs Number of Hidden layer 1
Accuracy vs. No. of epochs Number of Hidden layer 2
Loss vs. No. of epochs Number of Hidden layer 2
error vs. No. of epochs Number of Hidden layer 2
Accuracy vs. No. of epochs Number of Hidden layer 3
Loss vs. No. of epochs Number of Hidden layer 3
error vs. No. of epochs Number of Hidden layer 3
Accuracy vs. No. of epochs Number of Hidden layer 4
Loss vs. No. of epochs Number of Hidden layer 4
error vs. No. of epochs Number of Hidden layer 4
train
test
accuracy
loss
error
epoch



As seen in the graphs above, there is a significant divergence between the train and test data set when the number of layers increases. For layer 5 and 6 there is no sign of the model learning from the data set as the error rate seems to be consistent.



	Number of Layer	minimum error	epoch
0	1	0.100601	48
1	2	0.457207	49
2	3	0.431682	49
3	4	0.628003	14
4	5	0.628378	23
5	6	0.628754	30

Figure on the left shows that the time taken for one epoch increases as the number of layers increases. This is expected as the number of computations increases as the number of hidden layers increases. The graph on the represents the lowest number of epochs for the model to produce minimum error. The trend suggests that as the number of layers increases the minimum error increases. It can be concluded that a shallow model is better for this model.

Conclusion for task 2

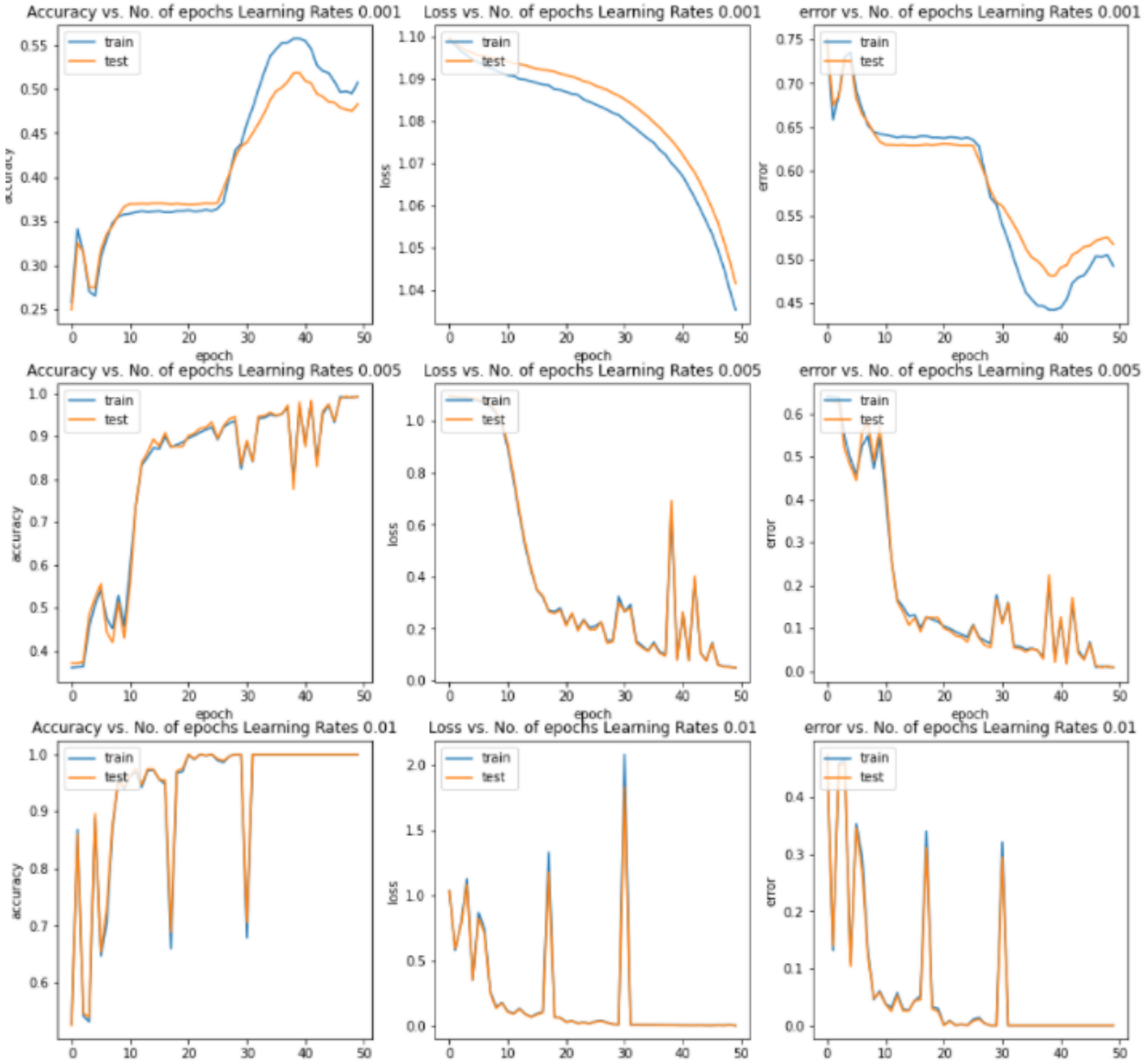
A wide and shallow model is better to accurately predict the three target classes.

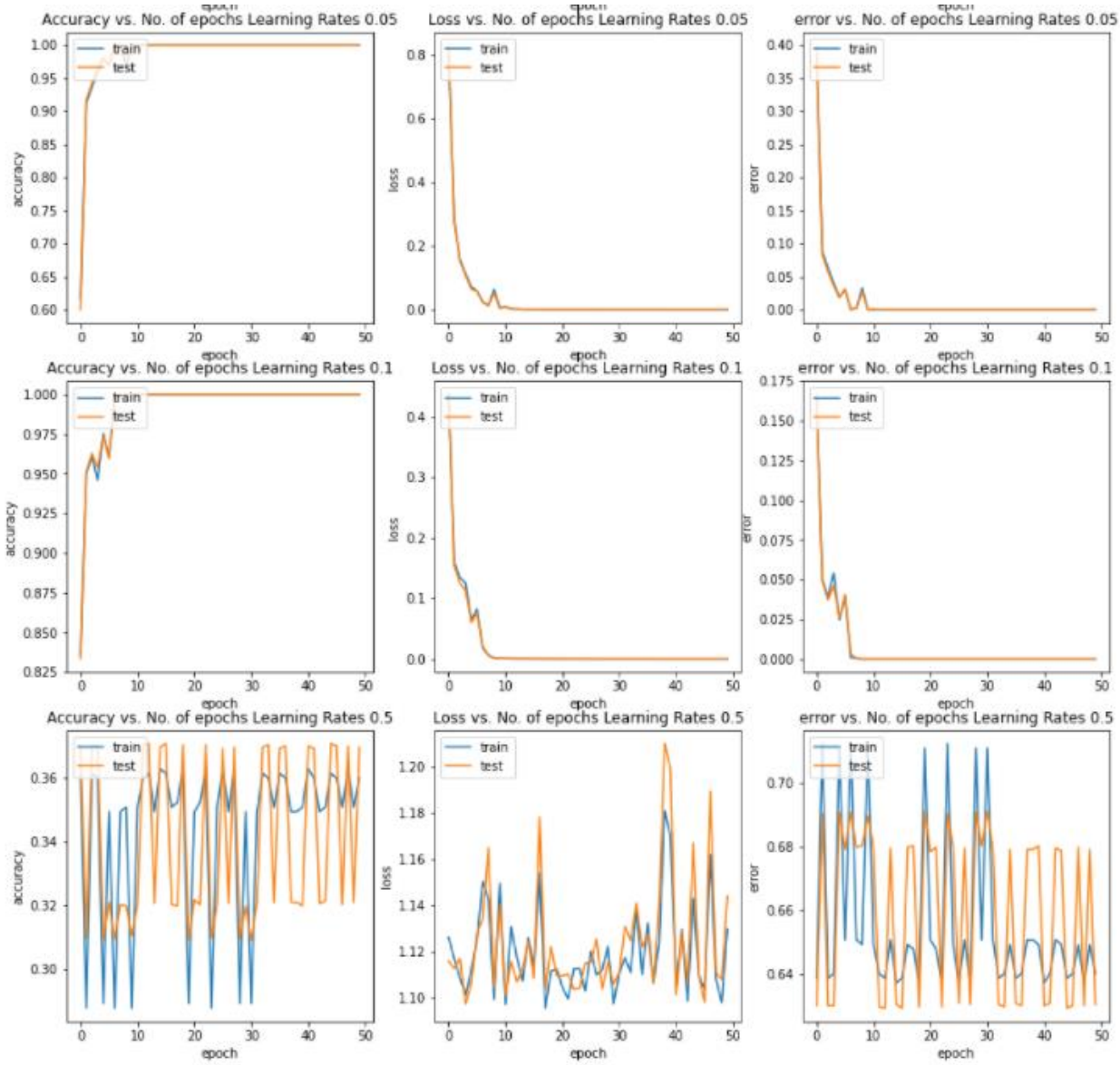
Task 3

You are asked to study the effect of learning rates. As with Task 2, your experiments have to be well documented. You need to give the correct conclusion and give suggestions on how learning rates should be set. This includes possible adaptive learning rates where the value increases or decreases as the increase of epochs.

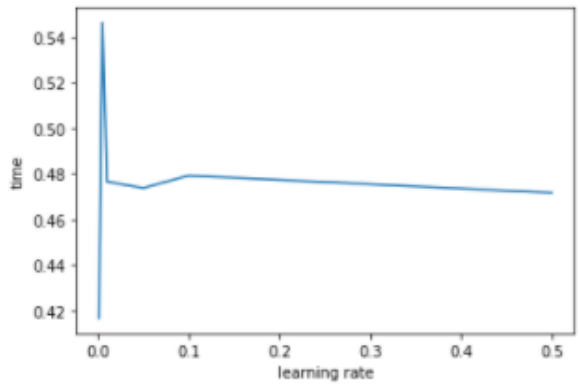
Learning rate adjusts how much the model changes according to the estimated error for each epoch, when the weights and biases are adjusted. This experiment is designed to study the effect of learning rate on the model. The model used is a three layer feed forward neural network with the input layer size 48, hidden layer size of 64 and an output layer of 3 neurons. The learning rate is varied as follows: [0.001, 0.005, 0.01 ,0.05, 0.1, 0.5].

The results:





Apart from a learning rate of 0.5, the other model seems to train the model decently well, with a learning rate of 0.010 - 0.1 achieving 0 error on the test dataset.



	Learning rate	minimum error	epoch
0	0.001	0.481231	39
1	0.005	0.007883	47
2	0.010	0.000000	20
3	0.050	0.000000	6
4	0.100	0.000000	7
5	0.500	0.629129	12

In terms of training time, the average time taken for an epoch is roughly the same with learning of 0.001 as an exception. In terms of performance, the model seems to train at an optimal level with a learning rate of 0.005 - 0.100 with 0.05 learning rate producing 0 error rate within 6 epochs.

Learning rate of 0.5 seems to perform poorly for this model as the learning rate could be too large which results in learning a non-optimize weight too quickly and hence it produces high error rate. For the lowest learning rate of 0.001 from the loss graph above we can see that the model is learning on the train data set, perhaps more epoch is required to achieve the minimum error,

Conclusion

A small learning rate will result in a higher number of epochs to potentially reach the lowest error rate. On the other hand, high learning may not provide the optimal training on the dataset as the weights and biases are adjusted too drastically.

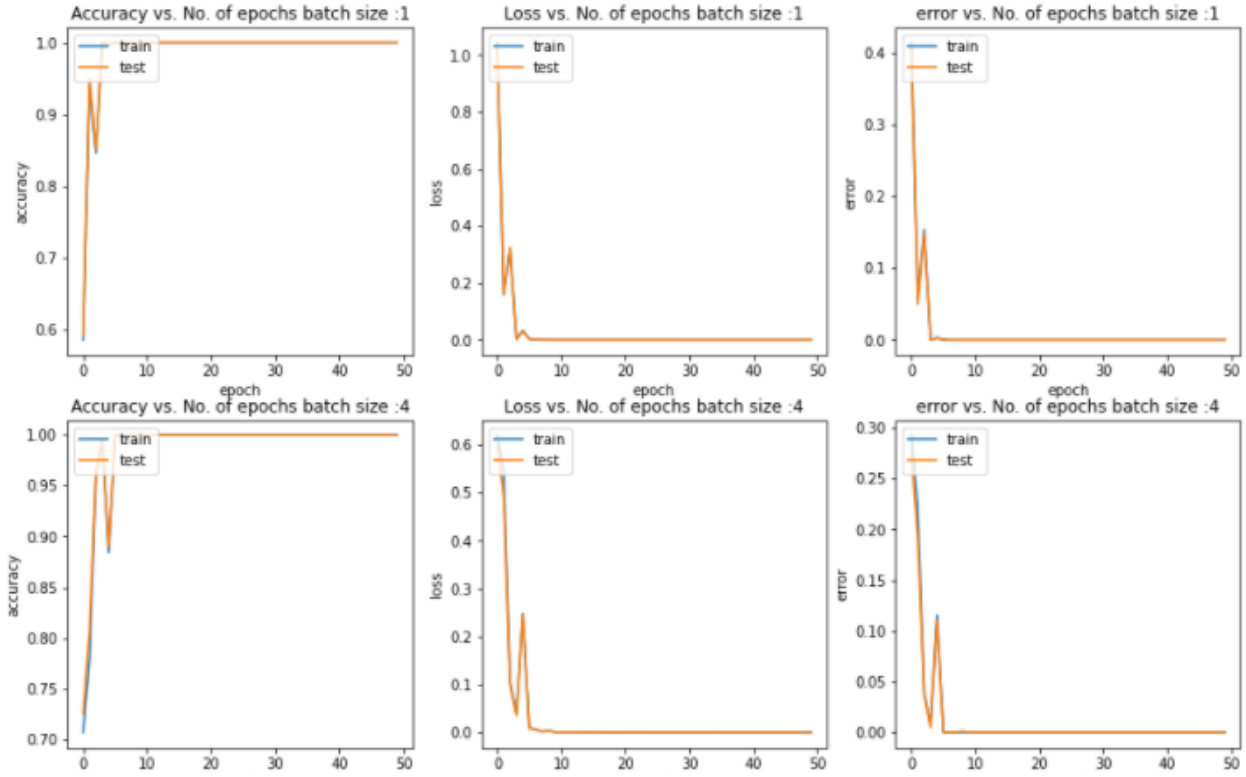
Task 4

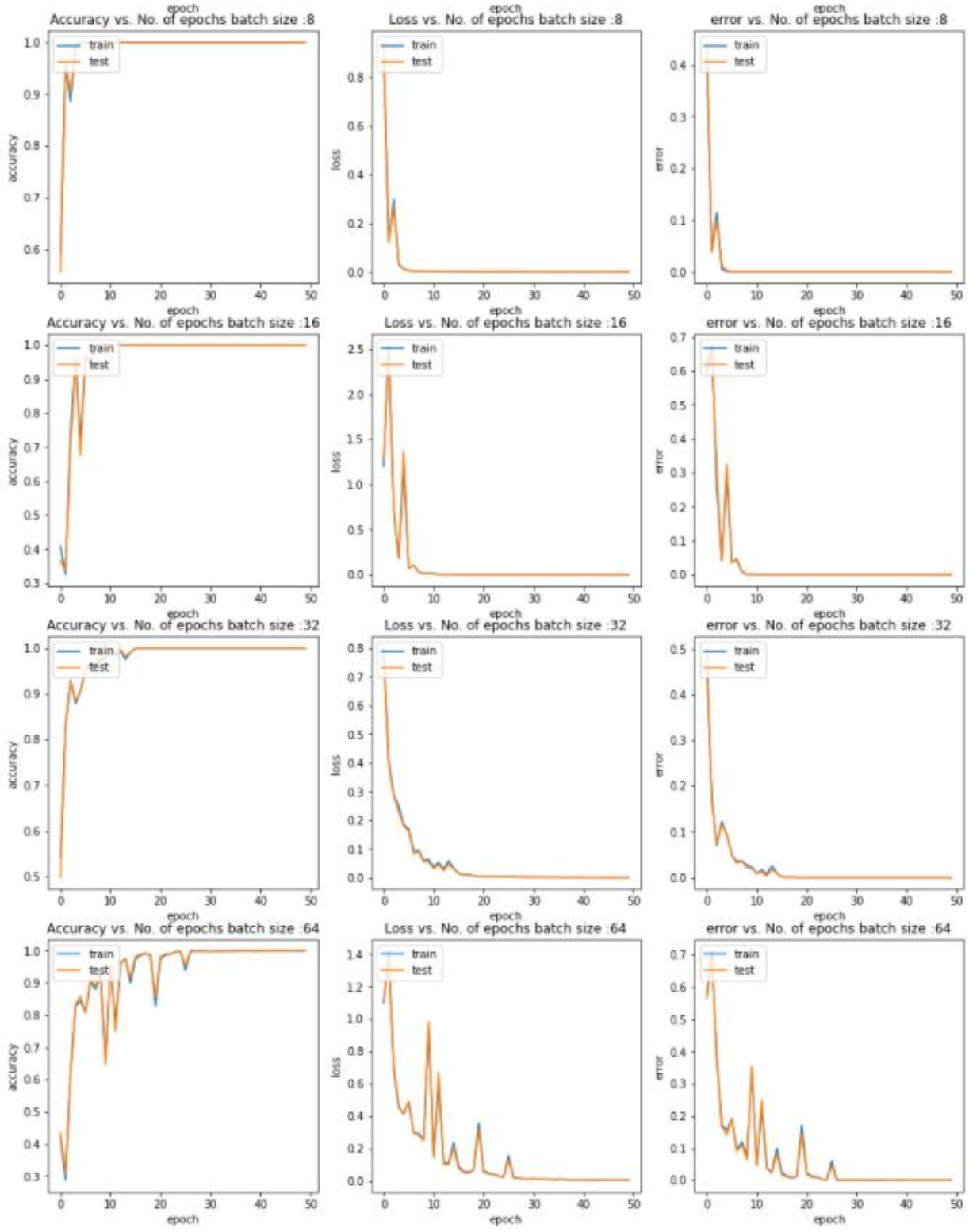
You are asked to study the effect of mini-batch size. You can set the mini-batch size to be 1 (stochastic gradient descent), N (batch gradient descent) or any other size. The most important aspect is to be conclusive with your findings. The mini-batch size really depends on the problem size.

For this experiment we are investigating the effects of batch sizes during training. The batch size is varied as follows: [1,4,8,16,32,64].

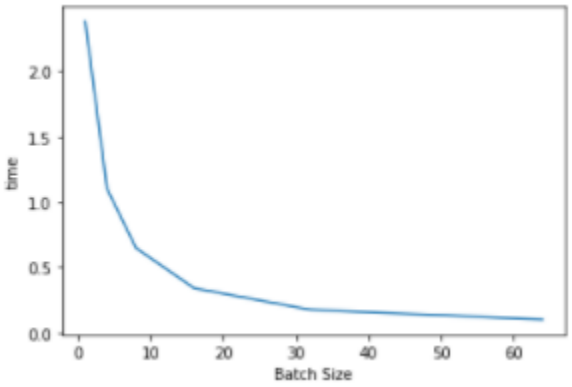
The model used for this experiment is the same as the above experiment, with a learning rate of 0.05 and varying batch size.

The results:





The model performed decently well with different batch sizes. There is not much divergence between the test and training dataset. The only thing left is to compare the time taken for training an epoch and also the number of epoch required to reach the lowest error rate.



	Batch Size	minimum error	epoch
0	1	0.0	3
1	4	0.0	5
2	8	0.0	5
3	16	0.0	8
4	32	0.0	16
5	64	0.0	36

The figure on the left shows that when batch size increases the time taken per epoch decreases, this is true as the weights and biases are updated based on batches, the higher the batches the lesser number of changes is made to the weight and biases. According to the graph on the right it shows that the number of epochs required to hit the minimum error increases as the batch size increases. With batch size 1 requiring only 1 epoch to reach the lowest error. On the other hand, a batch size of 64 will take 36 epoch to hit the lowest error rate.

Conclusion:

Personally, I feel that batch size 16 will be the most optimal as it takes a few epochs to reach minimum error plus the time taken per epoch is around the shortest amongst the 6 experiments.

References:

https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

https://www.analyticsvidhya.com/blog/2021/07/perform-logistic-regression-with-pytorch-seamlessly/