# Task 1

Information about the data set:

* Size of data: 2952 rows, 49 columns
* Size of train set: 2066
* Size of train set: 886

Final hyper parameters used:

1. Number of nodes of hidden layers = 200

2. Number of epochs = 150

3. Learning rates = 0.01 (Adaptive learning rate that divides by 1.5 every 10 epochs)

4. Mini-batch size = 20

Classification Error on Test set:

* Test accuracy: 1.77778 %
* Execution Time: 12.22337 s

# Task 2

## Changing number of hidden nodes:

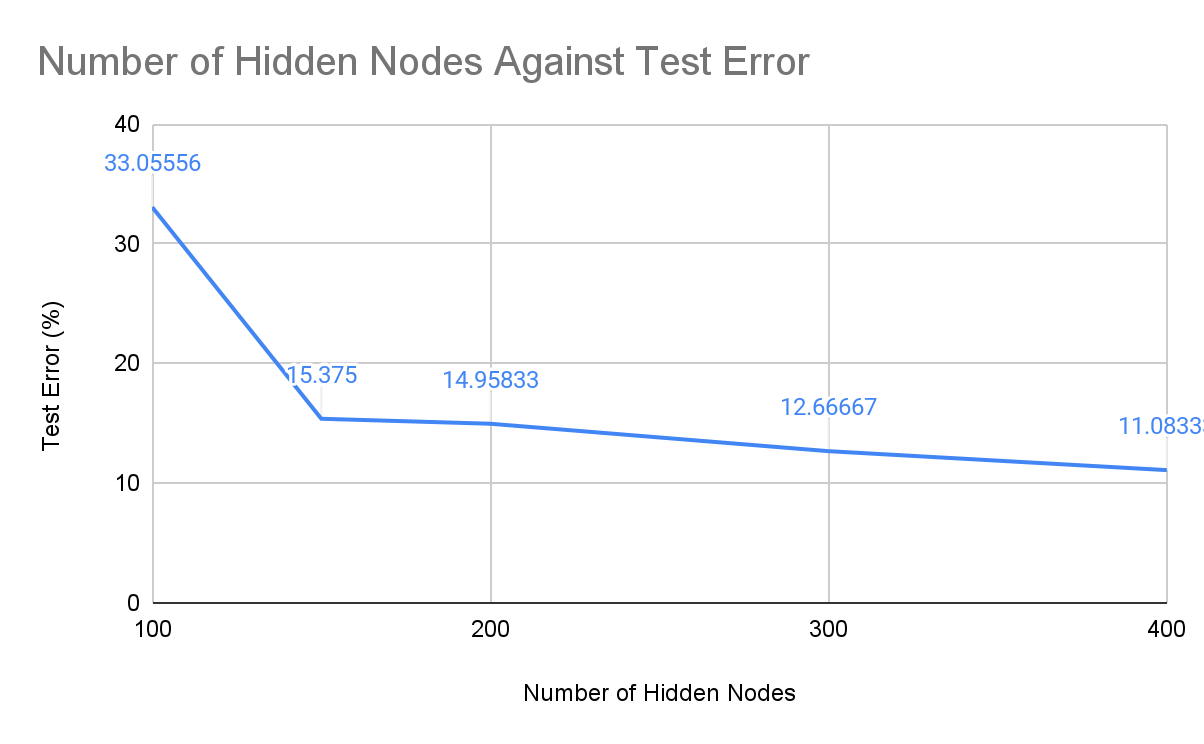
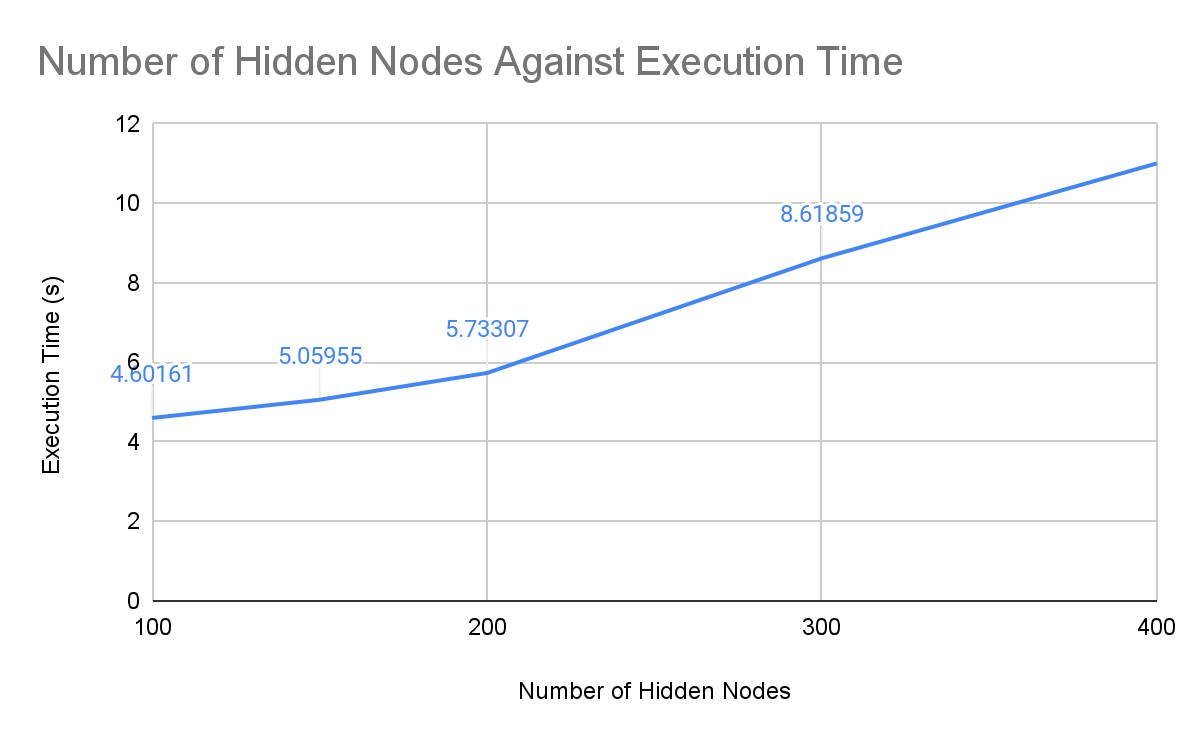
**Note: For the purpose of the experiments, I have used the same number of hidden nodes on every hidden layer.**

Constant hyper parameters used:

1. Number of epochs = 100
2. Learning rates = 0.001 (Fixed)
3. Mini-batch size = 30

Results:

| **Number of Hidden Nodes** | **Execution Time** | **Test Error** |
| --- | --- | --- |
| 100 | 4.60161 | 33.05556 % |
| 150 | 5.05955 | 15.37500 % |
| 200 | 5.73307 | 14.95833 % |
| 300 | 8.61859 | 12.66667 % |
| 400 | 11.00542 | 11.08333 % |



From the results of the experiment and the graph above, we can see that the execution time increases as the number of hidden nodes increases. Hence we can observe that complexity of the neural network increases as we increase the number of hidden nodes.

We can also see that test error decreases as the number of hidden nodes increases, but slowly reaches a constant error rate. This suggests that generally, increasing the number of hidden nodes would improve the classification accuracy, but eventually reaches a stagnant point where increasing the number of hidden nodes would no longer improve the classification accuracy.

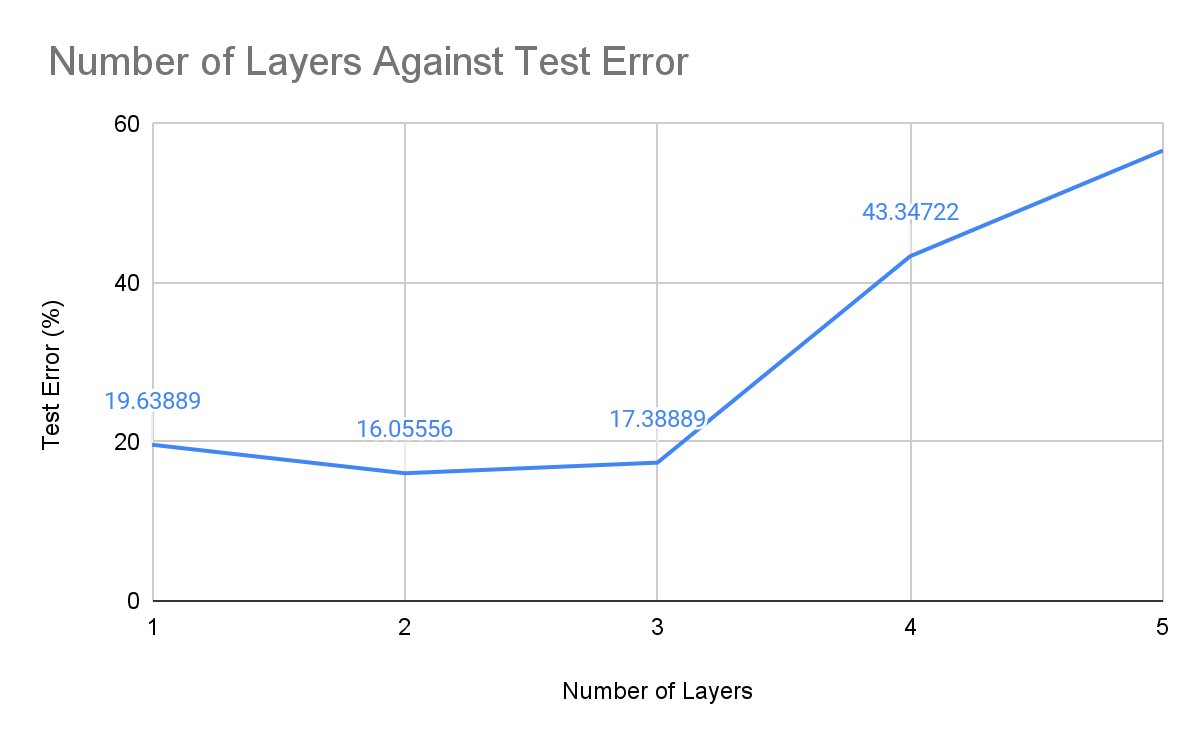
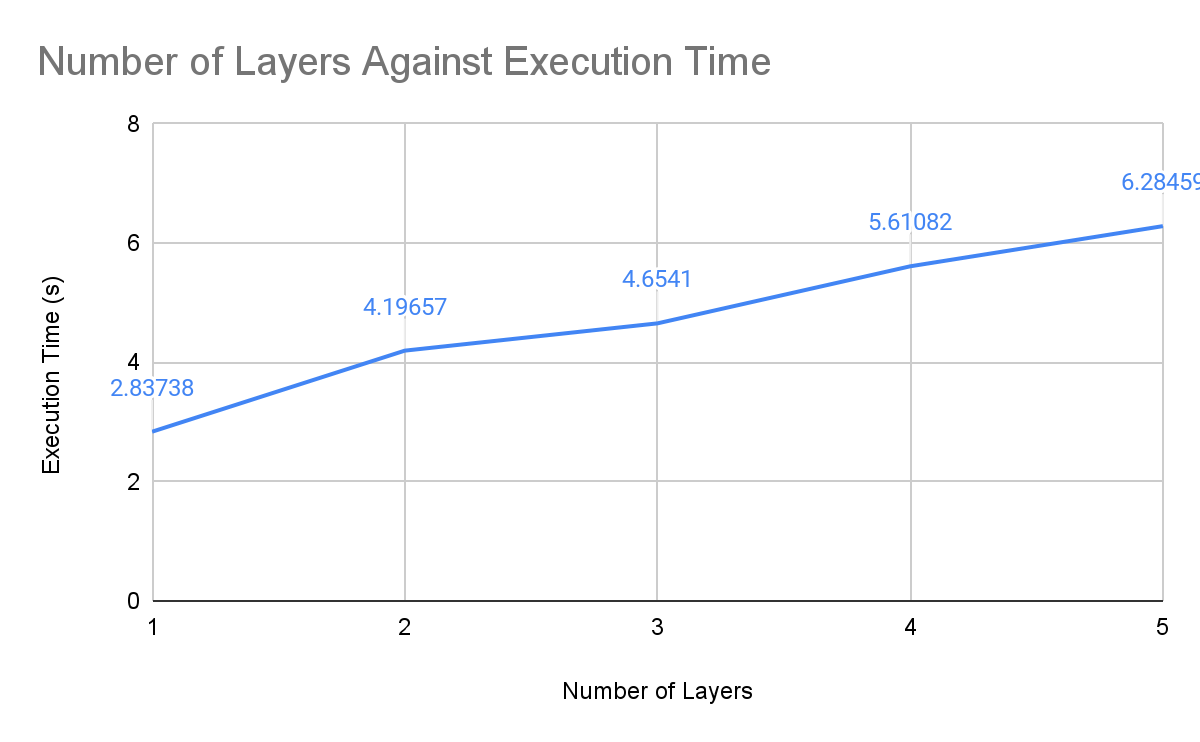
## Changing number of layers:

Constant hyper parameters used:

1. Number of epochs = 100
2. Learning rates = 0.001 (Fixed)
3. Mini-batch size = 30
4. Number of hidden nodes = 100

Results:

| **Number of Layers** | **Execution Time** | **Test Error** |
| --- | --- | --- |
| 1 | 2.83738 | 19.63889 % |
| 2 | 4.19657 | 16.05556 % |
| 3 | 4.65410 | 17.38889 % |
| 4 | 5.61082 | 43.34722 % |
| 5 | 6.28459 | 56.63889 % |

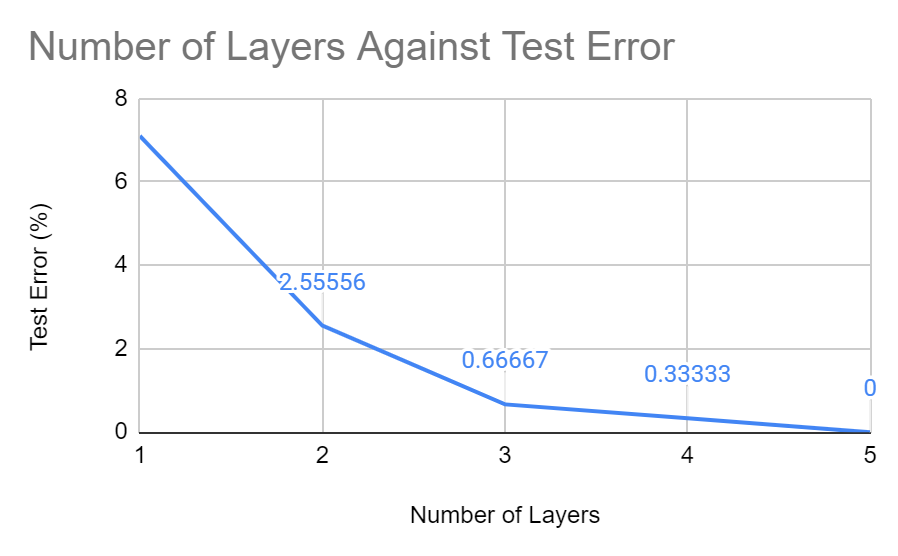
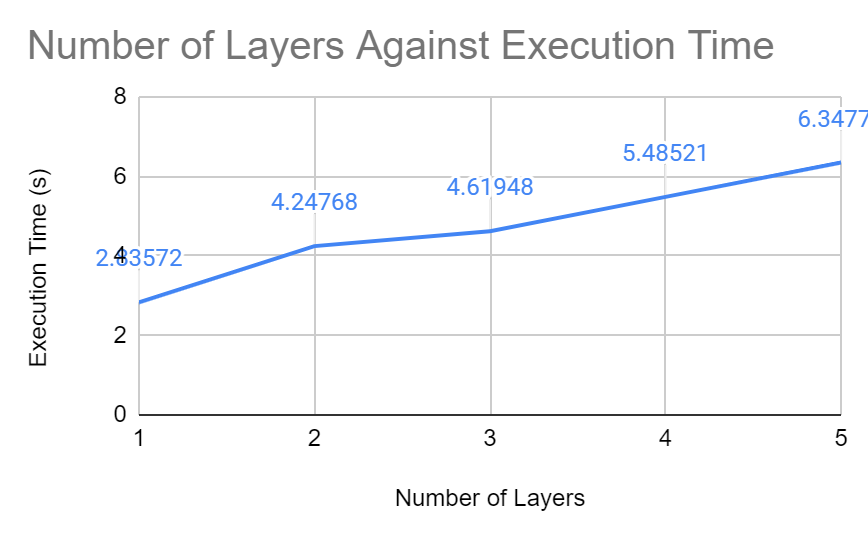


Execution time increases linearly with the number of layers, which suggests that increasing the layers increases the complexity of the neural network. The test error decreases, then increases. This suggests that initially, increasing the number of layers led to better classification accuracy, but accuracy started decreasing as it is increased over a certain number of layers. This is likely because of the set of hyperparameters used, which does not allow the more complex models to train enough to converge to a ‘good’ solution.

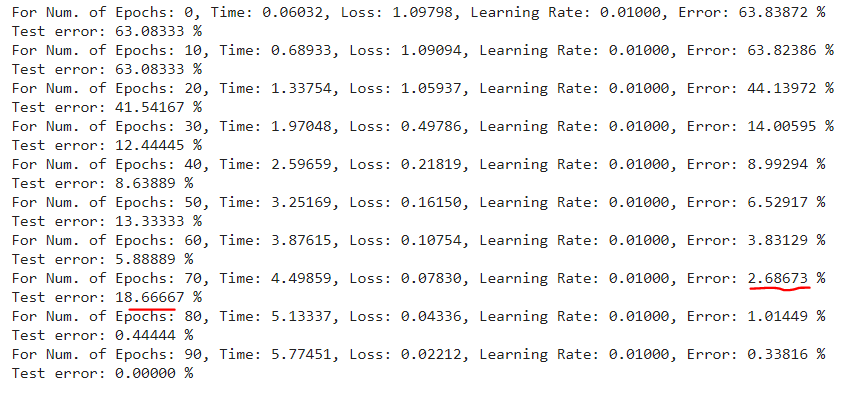
Hence, I repeated the experiment again, but with a higher **learning rate of 0.01**, so as to allow the model to train faster and converge to the optimal model quicker.

Results: *(Using learning rate = 0.01)*

| **Number of Layers** | **Execution Time** | **Test Error** |
| --- | --- | --- |
| 1 | 2.83572 | 7.09722 % |
| 2 | 4.24768 | 2.55556 % |
| 3 | 4.61948 | 0.66667 % |
| 4 | 5.48521 | 0.33333 % |
| 5 | 6.34779 | 0.00000 % |



Similar to the previous experiment, execution time increases when we add more layers, which suggests that the complexity of the model increases with more layers. However, in this experiment, we observe that the test error decreases exponentially and slowly converges to 0 with the increase in the number of layers. Hence we can conclude that the performance of the model does improve with more layers.



**NOTE:** During the training, we can also observe signs of overfitting in the 5 layer network. This is because as the model gets more complex, it tends to increase the chances of overfitting, hence becoming worse at generalization and producing lower accuracies on the test set.

## Shallow, Wide and Deep Networks:

Constant hyper parameters used:

1. Number of epochs = 100
2. Learning rates = 0.001 (Fixed)
3. Mini-batch size = 30

**Shallow network**: A shallow network generally consists of only 1 or 2 hidden layers. Given the trends observed above, the performance of a shallow network is usually not as good since less analysis and training is done on the data.

Shallow Neural Network:

| **Structure** | **No. of Epochs** | **No. of Hidden Nodes** | **Execution Time** | **Test Error** |
| --- | --- | --- | --- | --- |
| 1 hidden layer | 100 | 100 | 4.26615 | 15.51389 % |

**Wide network**: Wide neural networks usually have 1-2 hidden layers but with more neurons (hidden nodes) in each layer. Increasing the number of neurons between layers is able to detect the patterns in greater details, hence usually resulting in better accuracies. However, this also increases the complexity of the network. Another issue to take note of when using wide networks is that there is the risk of overfitting, resulting in the model being not as good for generalization. Wide neural networks usually work best if the data set is small and the problem is not too complex.

Wide Neural Network:

| **Structure** | **No. of Epochs** | **No. of Hidden Nodes** | **Execution Time** | **Test Error** |
| --- | --- | --- | --- | --- |
| 1 hidden layer | 100 | 400 | 11.11809 | 8.98611 % |

**Deep network**: Deep neural networks usually have many hidden layers. An increase in the number of hidden layers allows the network to learn features at more levels of abstraction, hence usually improving the performance of the model. Multiple layers are also better at generalizing compared to wide networks. However, this will also increase the complexity of the model, which can make it computationally expensive to train. Deep networks are usually better for solving more complex problems with larger data sets.

Deep Neural Network:

| **Structure** | **No. of Epochs** | **No. of Hidden Nodes** | **Execution Time** | **Test Error** |
| --- | --- | --- | --- | --- |
| 3 hidden layer | 1000 | 200 | 60.06633 | 0.00000 % |

For this particular dataset, a deep neural network gives the best classification accuracy out of the 3. However, if given similar hyperparameters, deep networks would not perform as well. As its complexity is much higher than the other two, it would require more epochs, or a higher learning rate in order to perform to its full potential, which explains why it has a much longer execution time. However, since our given dataset is not very huge and the problem that we are solving is relatively simple, it would be more computationally efficient to choose the **wide neural network** for this case. This is because the wide neural network is able to produce a relatively reasonable classification accuracy without taking too long.

# Task 3

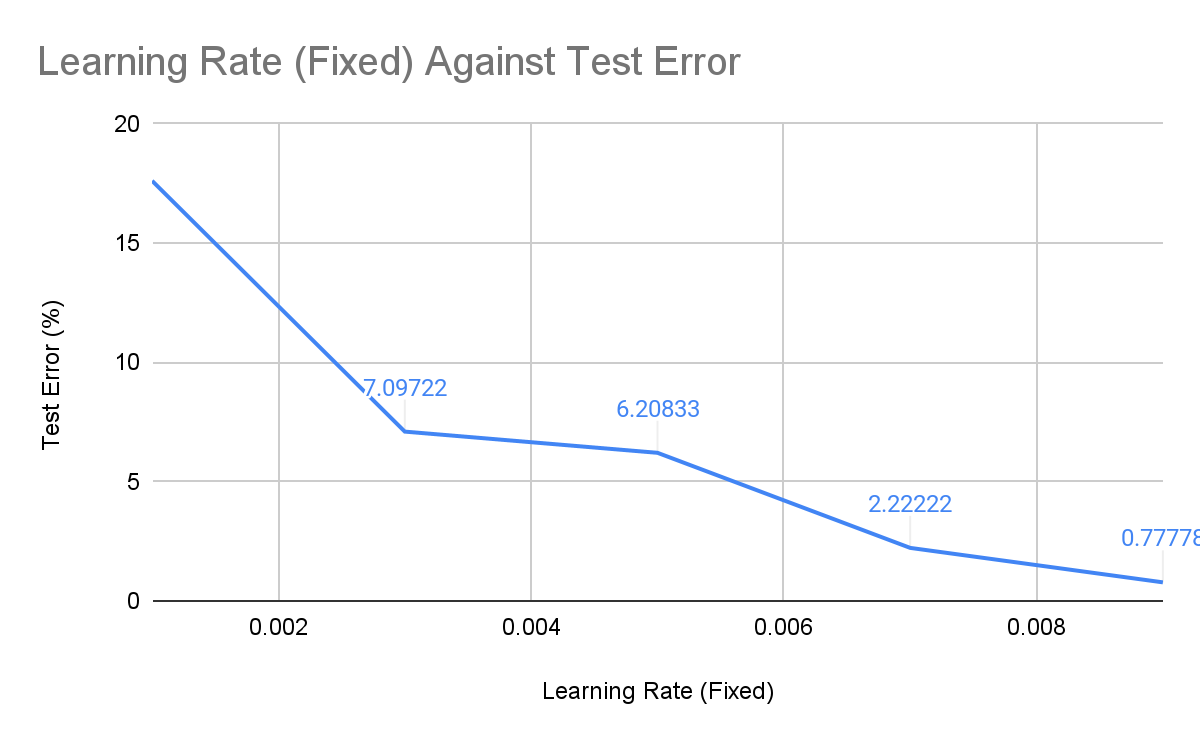
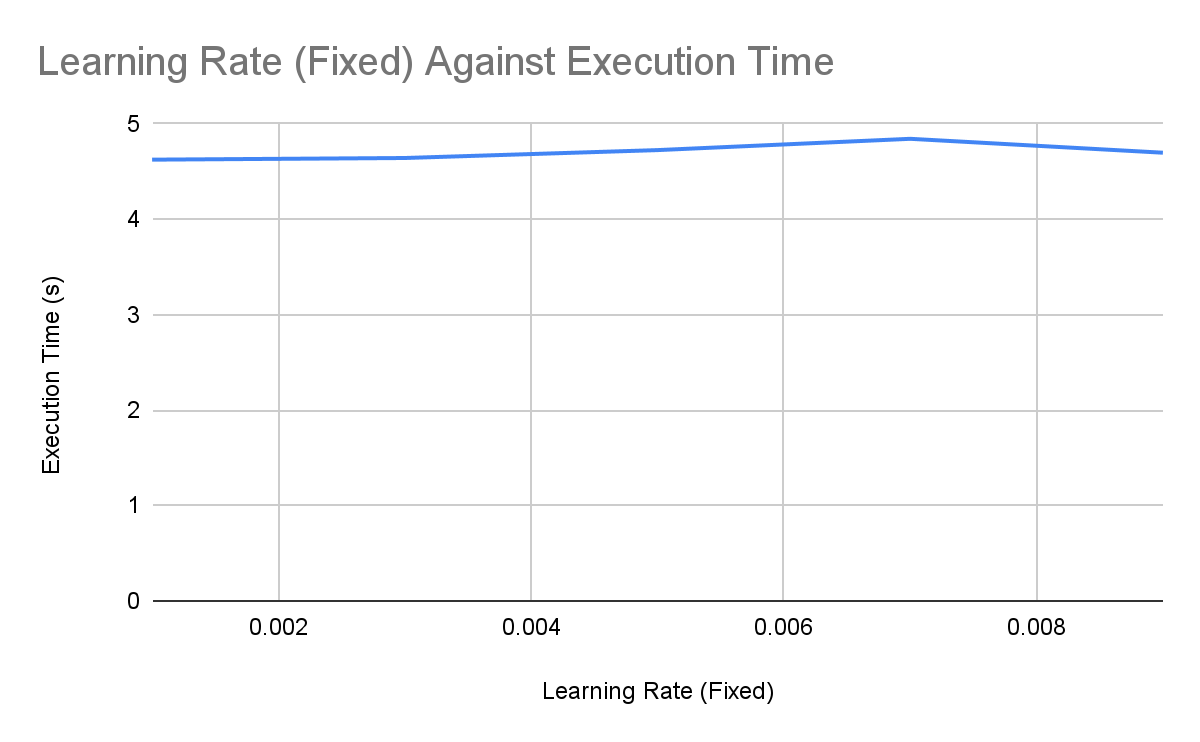
## Changing learning rates:

### Constant hyper parameters used:

1. Number of epochs = 100
2. Mini-batch size = 30
3. Number of hidden nodes = 100

### Results:

| **Learning Rate (Fixed)** | **Execution Time** | **Test Error** |
| --- | --- | --- |
| 0.001 | 4.62454 | 17.61111 % |
| 0.003 | 4.64185 | 7.09722 % |
| 0.005 | 4.72465 | 6.20833 % |
| 0.007 | 4.84306 | 2.22222 % |
| 0.009 | 4.69736 | 0.77778 % |



From the experiment, execution time is constant regardless of the learning rate. Test error decreases exponentially. This suggests that the learning rate does not affect the complexity of the model. Classification accuracy increases and the model performs better when the learning rate is increased.

## Adaptive learning rates:

### Constant hyper parameters used:

1. Number of epochs = 100
2. Mini-batch size = 30
3. Number of hidden nodes = 100

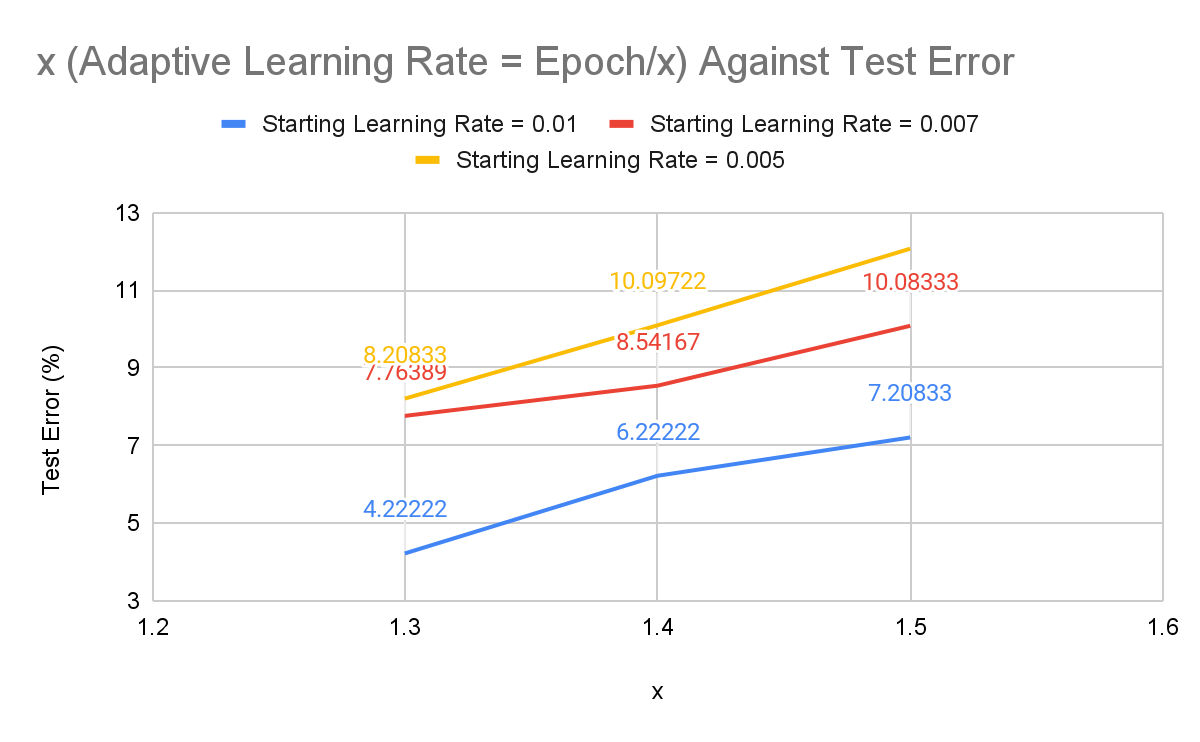
Divide the learning rate by x every 10 epochs

Results:

\*\* when x >1, learning rates decreases with epochs

\*\* when x<1, learning rates increases with epochs

| **Starting Learning Rate** | **x** | **Execution Time** | **Test Error** |
| --- | --- | --- | --- |
| 0.01 | 1.3 | 4.61761 | 4.22222 % |
| 1.4 | 4.66165 | 6.22222 % |
| 1.5 | 4.65058 | 7.20833 % |
| 0.007 | 1.3 | 4.52604 | 7.76389 % |
| 1.4 | 4.63525 | 8.54167 % |
| 1.5 | 4.79620 | 10.08333 % |
| 0.005 | 0.8 | 4.64237 | 1.22222 % |
| 0.9 | 4.65208 | 1.77778 % |
| 1.3 | 4.56385 | 8.20833 % |
| 1.4 | 4.64512 | 10.09722 % |
| 1.5 | 4.61243 | 12.06945 % |
| 0.003 | 0.8 | 4.60890 | 3.44445 % |
| 0.9 | 4.66232 | 5.00000 % |



\*\* This graph does not include x < 1

Regardless of the starting learning rate and the value of x, execution time is relatively constant. Hence, we can conclude that complexity is not affected by the learning rates. Higher learning rates generally produce better classification accuracies and perform better on the test set.

For **fixed learning rates**, I would suggest using higher learning rates, especially if execution time is a priority. This is because higher learning rates are able to achieve higher accuracies quickly and with a smaller number of epochs. However, we may run into the risk of overfitting, especially at the larger epochs, resulting in a sub-optimal final set of weights.

Hence, the better way would be to use **adaptive learning rates.** Adaptive learning rates can reduce training time, and also help to alleviate some of the pressure of choosing a learning rate and learning rate schedule. For adaptive learning rates, if we start with a larger learning rate, we can make the learning rate decrease as epochs increase, to achieve better performance. (As seen from the experimental results when x > 1). Vice versa, if we start with a smaller learning rate, we can make the learning rate increase as epochs increase. (As seen from the experimental results when x < 1).

# Task 4

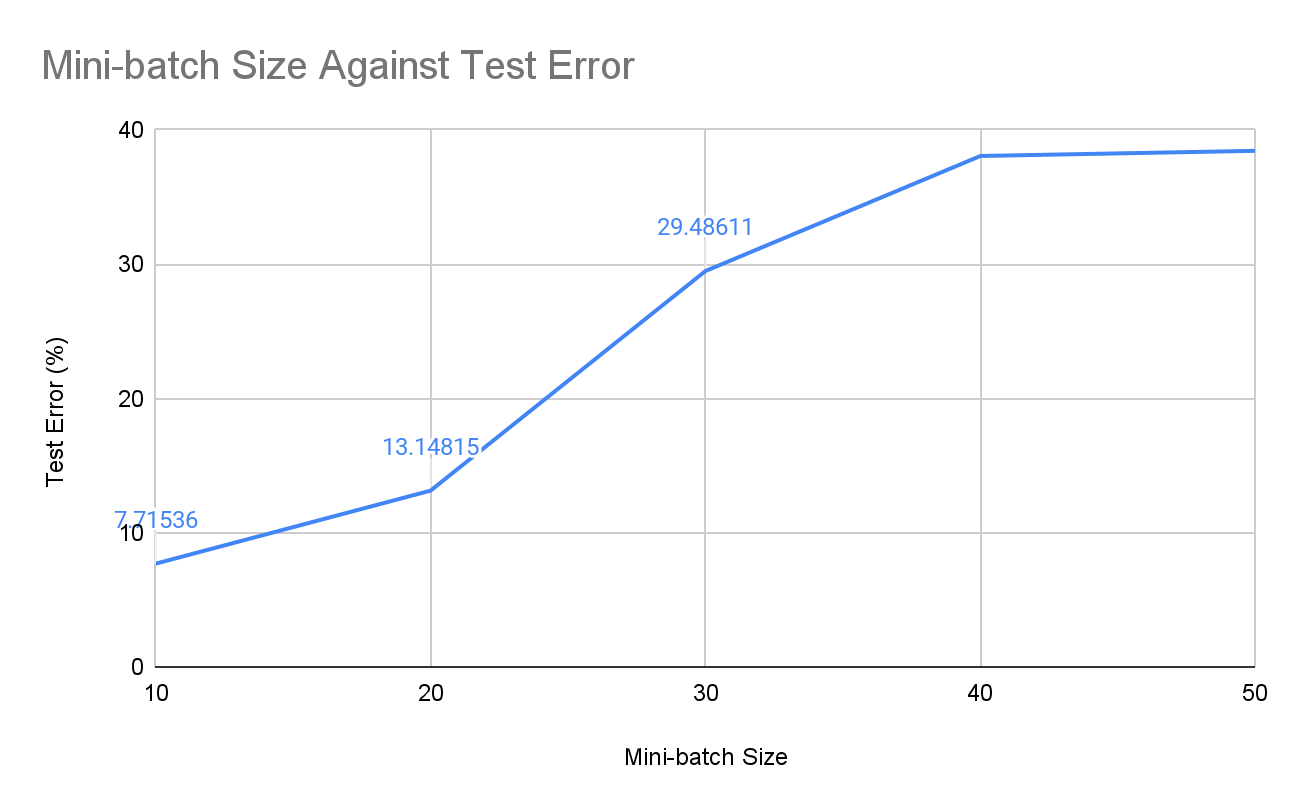
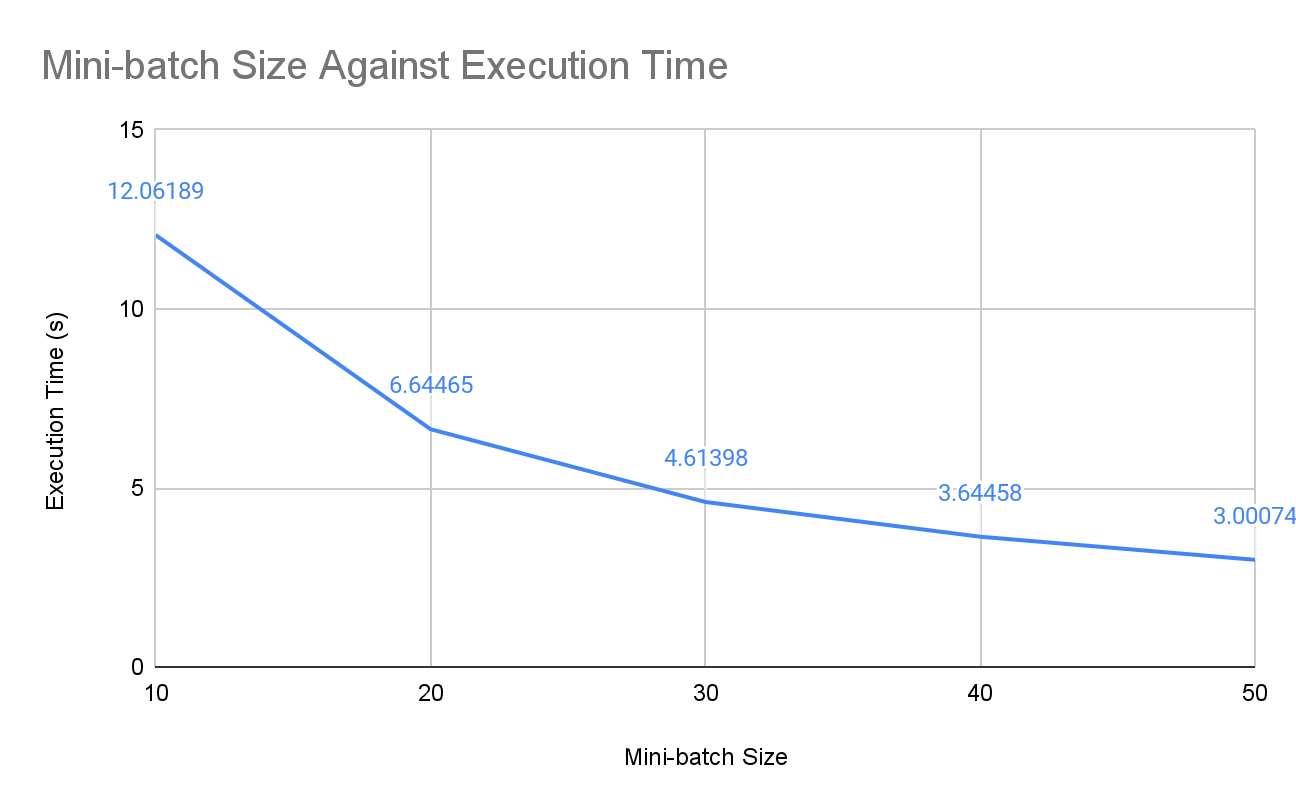
## Changing mini-batch size:

### Constant hyper parameters used:

1. Number of Epochs = 100
2. Learning rates = 0.001 (Fixed)
3. Number of hidden nodes = 100

### Results:

| **Mini-batch Size** | **Execution Time** | **Test Error** |
| --- | --- | --- |
| 10 | 12.06189 | 7.71536 % |
| 20 | 6.64465 | 13.14815 % |
| 30 | 4.61398 | 29.48611 % |
| 40 | 3.64458 | 38.04348 % |
| 50 | 3.00074 | 38.42593 % |



As seen from the experiment, execution time decreases as the mini-batch size increases. This suggests that the network gets less complex when the mini-batch size is larger. However, the test error increases as mini-batch size increases. This means that the performance of the model degrades as mini-batch size increases, resulting in lower classification accuracy. Hence, although larger mini-batch sizes increase computational speed, it results in poor generalization. On the other hand, although smaller batch sizes require a longer execution time, it has faster training dynamics and also better generalization, resulting in higher classification accuracies.