NANYANG TECHNOLOGICAL UNIVERSITY

CZ4042: Neural Networks

Project 1

Building and Evaluating Neural Networks for Classification and Approximation Problems

Submitted by:

Khare Simran (U1423025D)
Marathe Ajinkya Avinash (U1522716K)

Table of Contents

Part A: Classification Problem

Introduction

The objective of the first part of the project is to build a neural network to classify the Landsat satellite dataset. Provided in this dataset are multi-spectral values of pixels in 3x3 neighborhoods in a satellite image and the classification associated with the central pixel in each neighborhood. Given these multi-spectral values, our neural network must be able to predict this classification.

The data provided has already been split into *training* and *testing*. There are 36 input features or attributes and 6 class labels. The class labels 1 to 7 correspond to red soil, cotton crop, grey soil, damp grey soil, soil with vegetation stubble, mixture class and very damp grey soil respectively. When reading the data, we take into account that there are no class 6 examples.

Task 1: Designing a 3-layer feedforward neural network

A neural network as in the figure below was designed. It has 36 input nodes corresponding to the 36 attributes, 10 hidden layer neurons and 6 output neurons corresponding to the 6 class labels. The hidden layer has a logistic activation function and the output layer is a softmax layer. The values of the learning rate (0.01), decay parameter (10^{-6}) and batch size (32) were assumed as provided in the question.

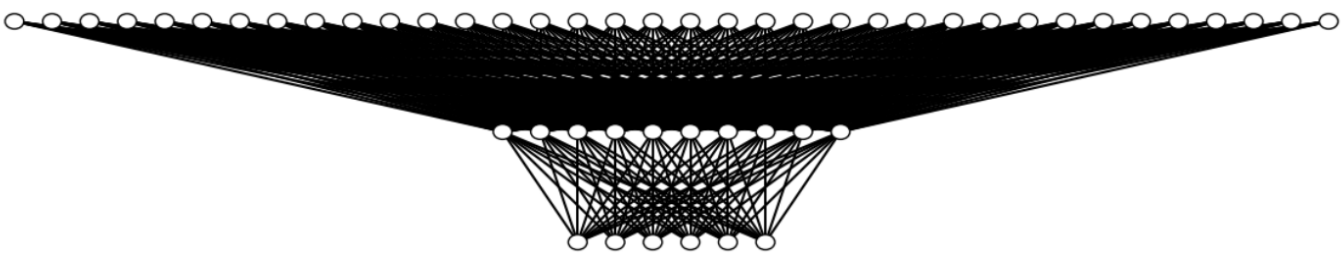


Fig 1.1. Neural Network with 36 input nodes, 10 hidden layer neurons and 6 output neurons

Task 2: Finding the optimal batch size for mini-batch gradient descent

We trained the neural network for the batch sizes 4, 8, 16, 32 and 64. For each batch size, we plotted the following graphs:

(i) Training error versus Number of iterations
(ii) Test accuracy versus Number of iterations
(iii) Time taken to update the parameters versus Batch Size

The information provided by these graphs was useful in determining the optimal batch size. Below are the graphs as well as an analysis and justification for the chosen optimal batch size.

(i) Training error versus Number of iterations

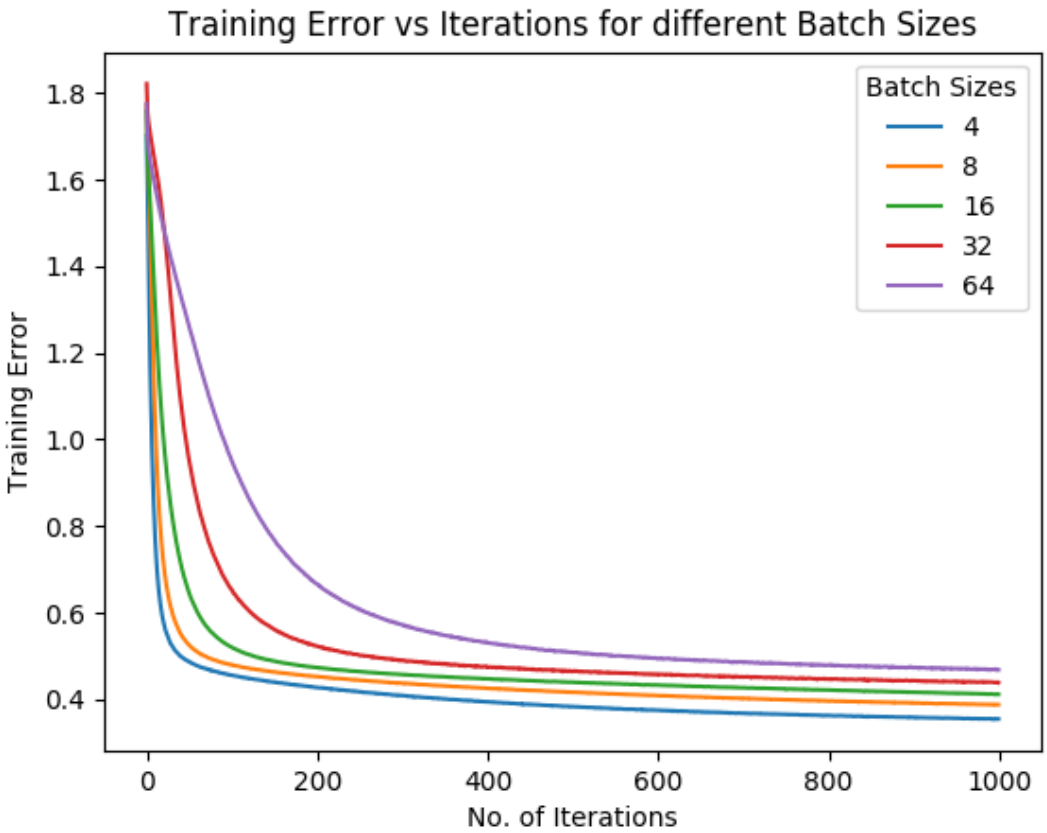


Fig 2.1

In the plot above (Fig 2.1) we observe that the Training Error or Cross Entropy is the least for batch size 4 at the 1000th iteration. This value is higher for higher batch sizes. Hence we would prefer a lower batch size over a larger one.

(ii) Test accuracy versus Number of iterations

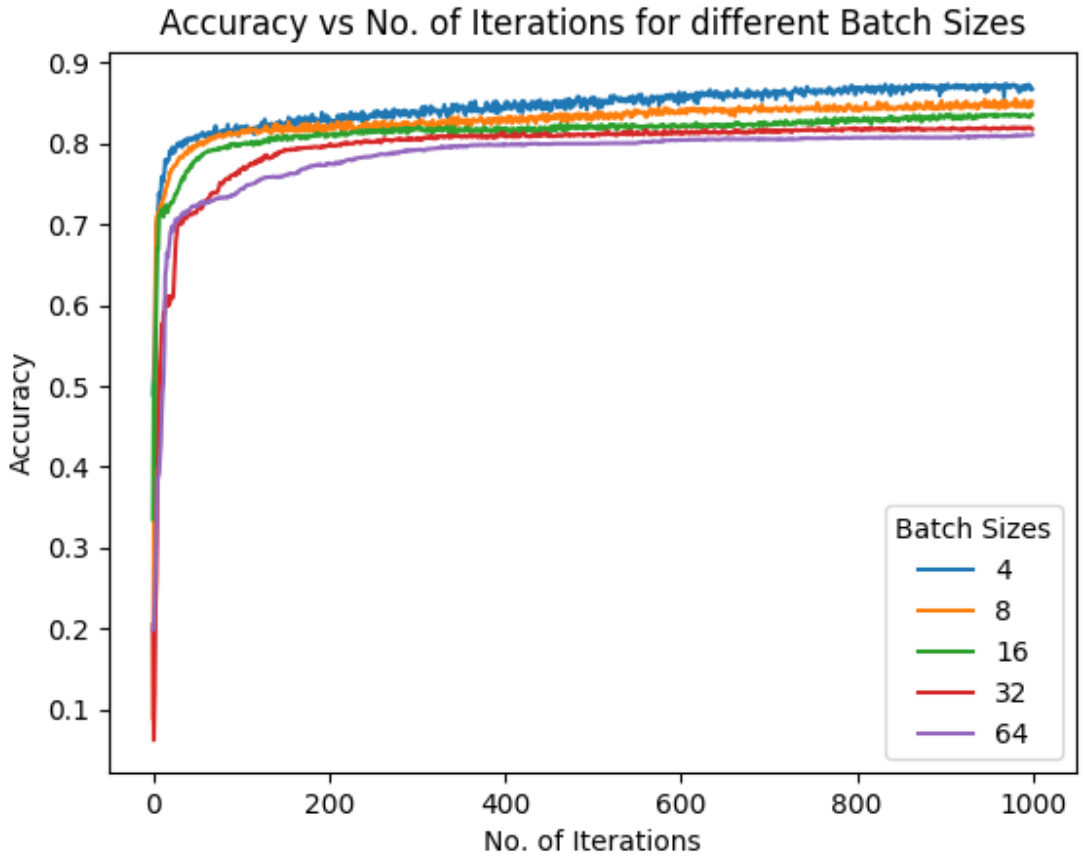


Fig 2.2

The plot above also suggests that a smaller batch size would be preferable, with batch size 4 achieving an accuracy of 0.869 and batch size 8 achieving 0.854. However we do note that the test accuracy plot for batch size 8 is smoother than that for 4.

(iii) Time taken to update the parameters versus Batch Size

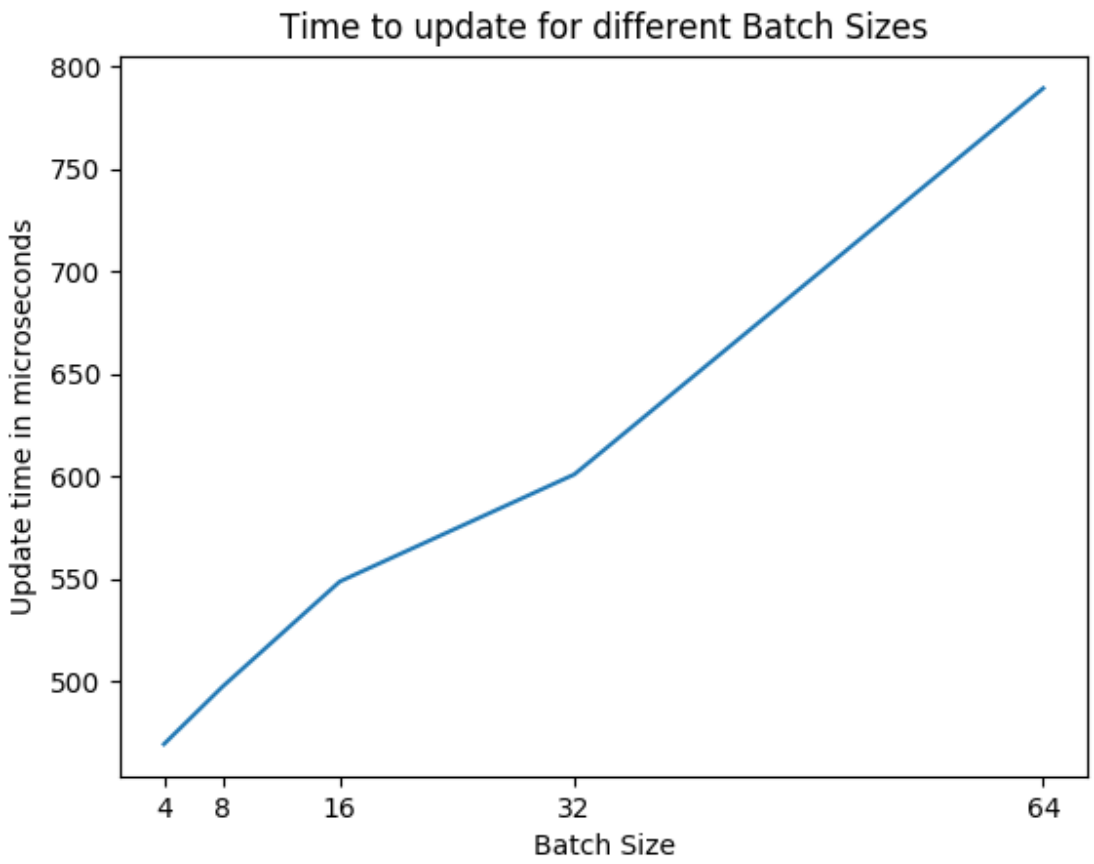


Fig 2.3

As we may expect, the time taken to update the parameters, i.e. the weights and biases is higher for larger batch sizes. We know that these parameters are updated once per mini-batch. For a small batch size, the updates of weights and biases will be ***more frequent***. This is because a smaller batch size would mean smaller matrices and hence less computationally intensive calculations. **However, we note that if the batch size is small, the number of mini-batches per epoch will be large. Hence, the total time for convergence will be higher for smaller batch sizes. We would like to pick a batch size that is big enough and the time for weight updates is reasonably small. Hence batch size 8 seems to be a good choice.**

Conclusion:

Based on the observations above, we take the optimal batch size to be 8.

Task 3: Finding the optimal number of hidden neurons for the 3-layer network.

In order to find the optimal number of hidden layer neurons, we examine the training error, test accuracy and the time to update the parameters i.e. weights and biases for different number of hidden layer neurons. Following are our observations.

(i) Training error versus Number of iterations for different number of hidden neurons

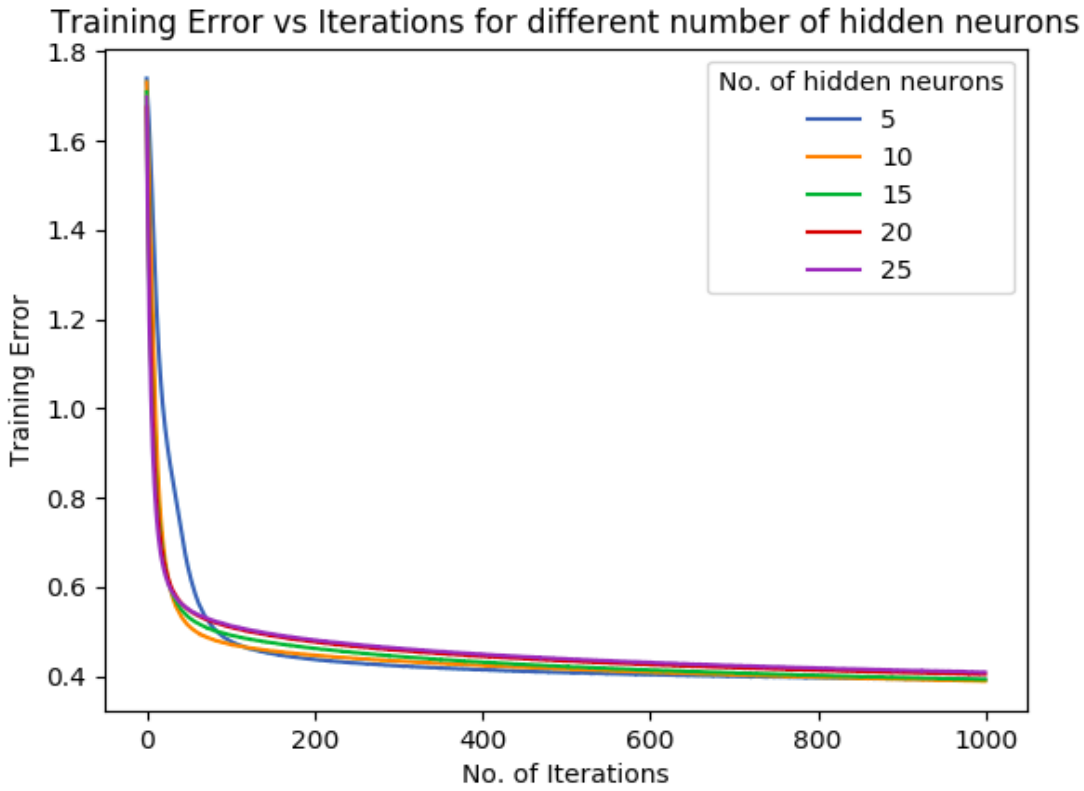


Fig 3.1.1

In Fig 3.1 we observe that the Training error is the lowest when the number of hidden layer neurons is 15 (0.3918).

(ii) Test accuracy versus Number of iterations for different number of hidden neurons

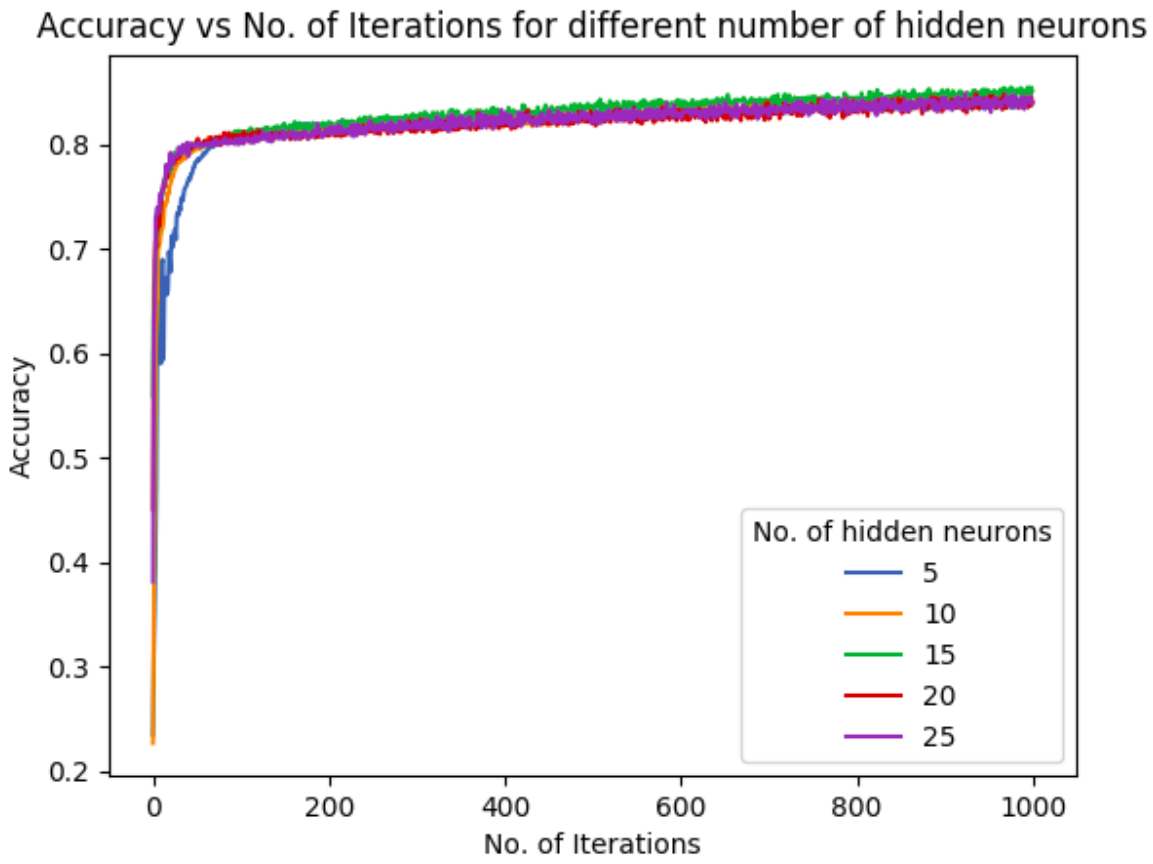


Fig 3.2

It can be observed that the test accuracy is the highest (0.844) when there are 15 neurons in the hidden layer.

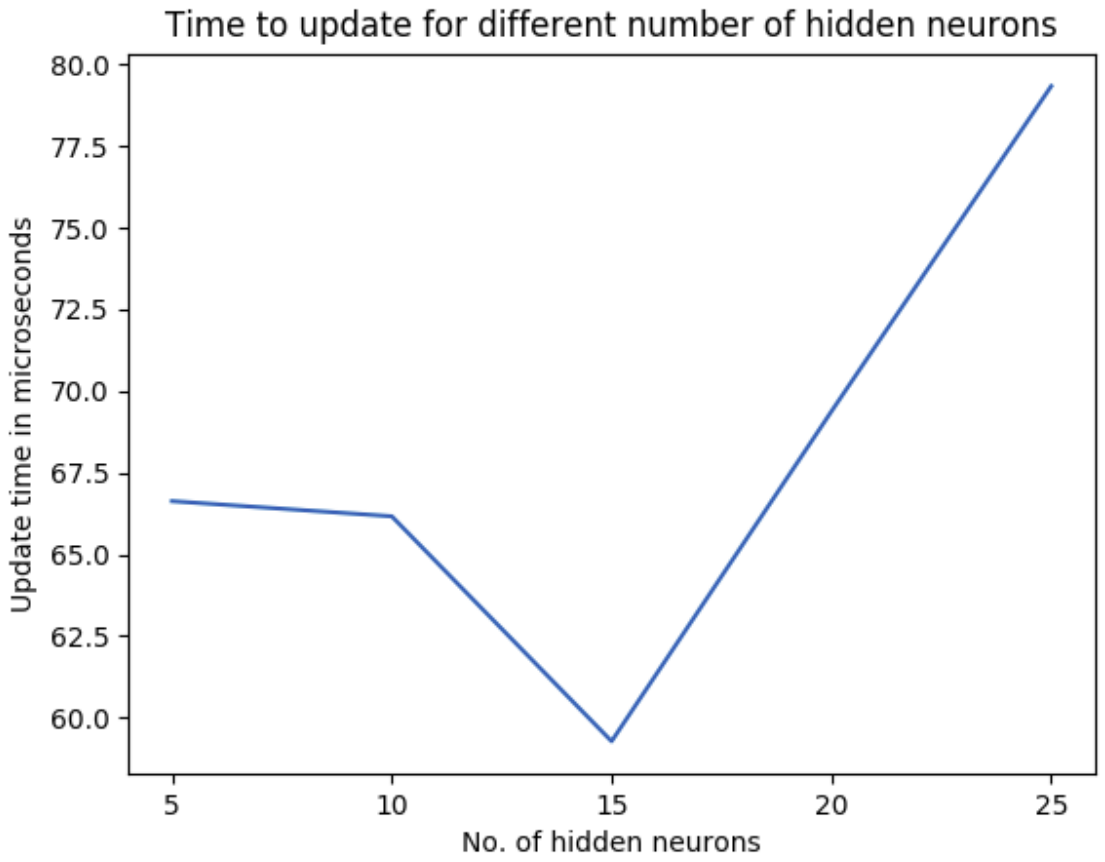


Fig 3.3

It can be observed that when the number of hidden neurons is 15, considerably lesser time is required for updates.

Conclusion:

The optimal number of neurons in the hidden layer would be those that give us a reasonably low training cost, high test accuracy and a shorter time to update the weights and biases. From our observations above, we notice that for 15 hidden neurons we achieve a significantly better performance in terms of time, and we also get reasonably high accuracy and low training cost. Hence we can conclude that the optimal number of neurons in the hidden layer would be 15.

Task 4: Finding the optimal decay parameter

We try to find the optimal value for the decay parameter, such the Training Error is the lowest and the Test Accuracy is the highest. Following are the graphs along with the observations made.

(i) Training Error versus Iterations for different values of the Decay Parameter

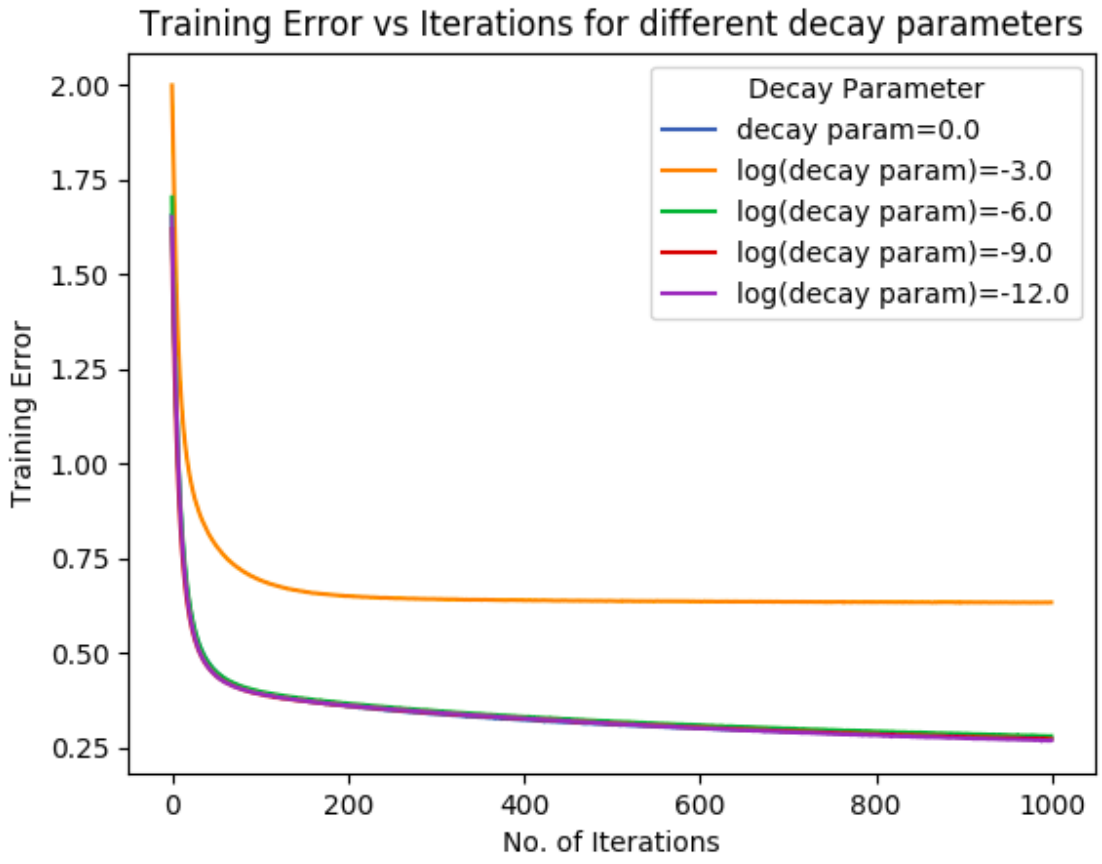


Fig 4.1

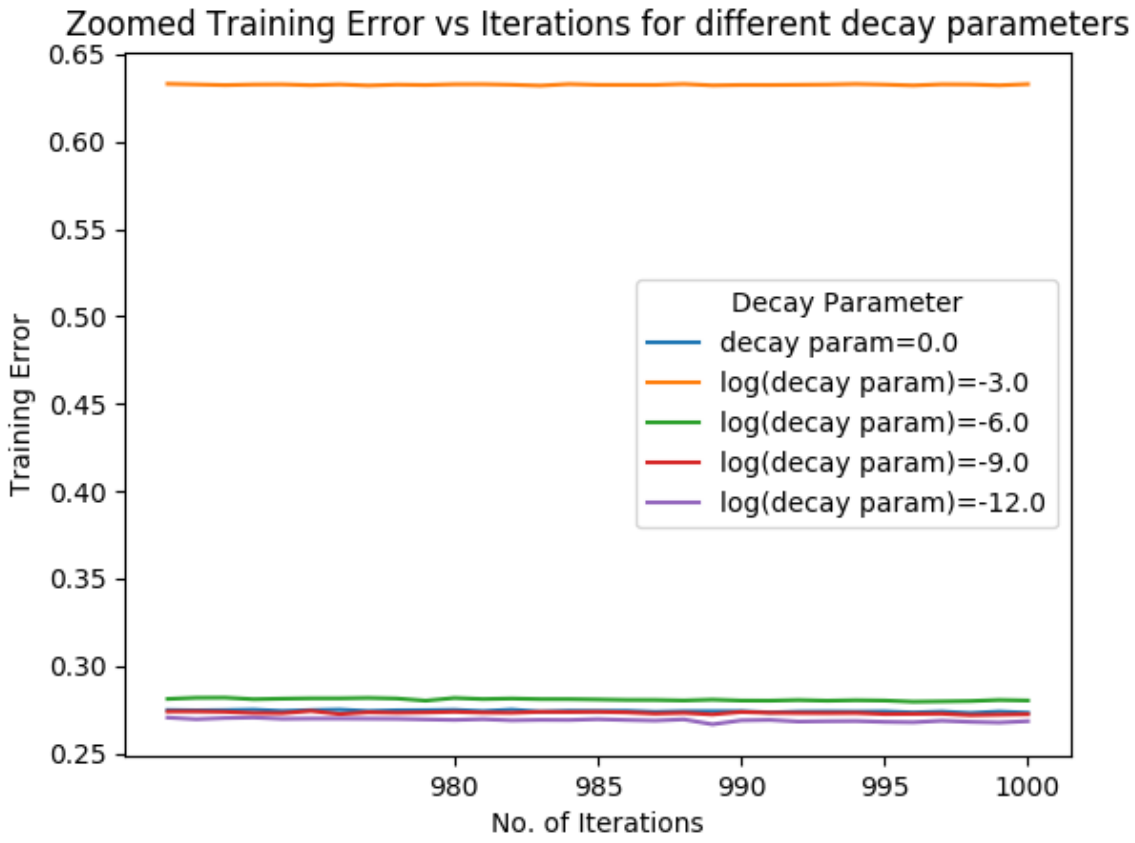


Fig 4.1.1 Zoomed version of Fig 4.2.1

From the plot above and its zoomed version, we can observe that the Training Cost is the lowest when the decay parameter = 10^{-12}

(ii) Test Accuracy versus Iterations for different values of the Decay Parameter

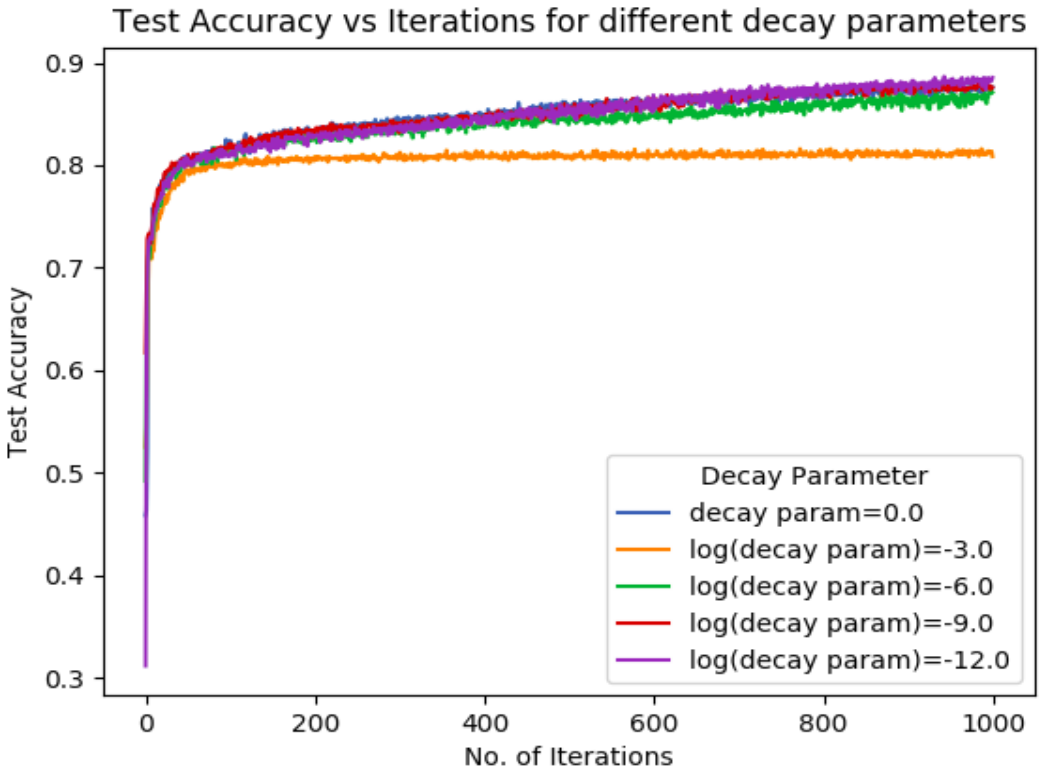


Fig 4.2.1

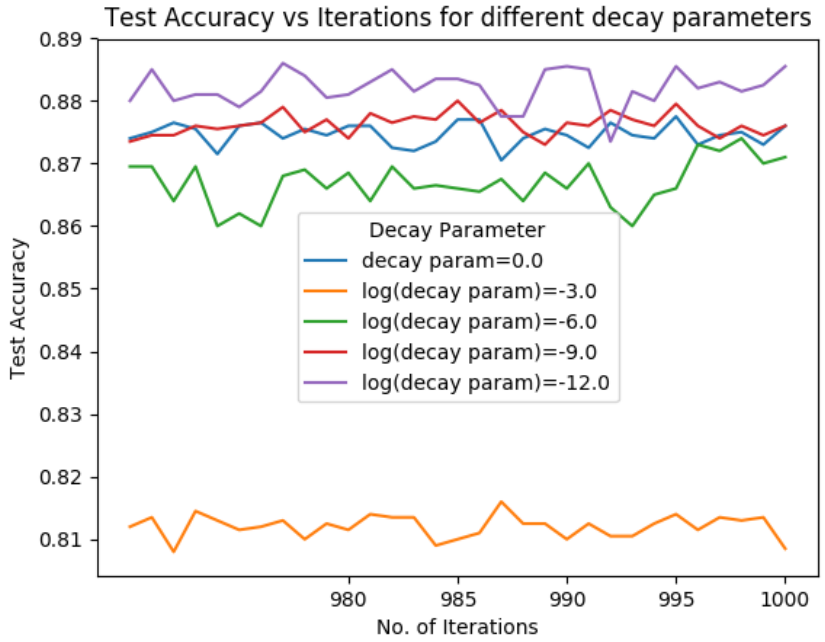


Fig 4.2.2 Zoomed version of Fig 4.2.1

Upon observing the plots above, we can conclude that for any number of iterations the Test Accuracy is highest for decay parameter = 10^{-12}

Conclusion:

The optimal value of the decay parameter would be the one that gives us the lowest Training Error and the highest Accuracy, hence it is 10^{-12}

Task 5: Designing and evaluating the performance of 4-layer neural network

A neural network was designed as follows with an input layer comprising 36 neurons, 2 hidden layers with 10 hidden neurons each and an output layer comprising 6 neurons. A batch size of 32 was used and the decay parameter was set to 10^{-6}. Following are the plots to compare the performances of 3 and 4 layer networks in terms of the training error and test accuracy.

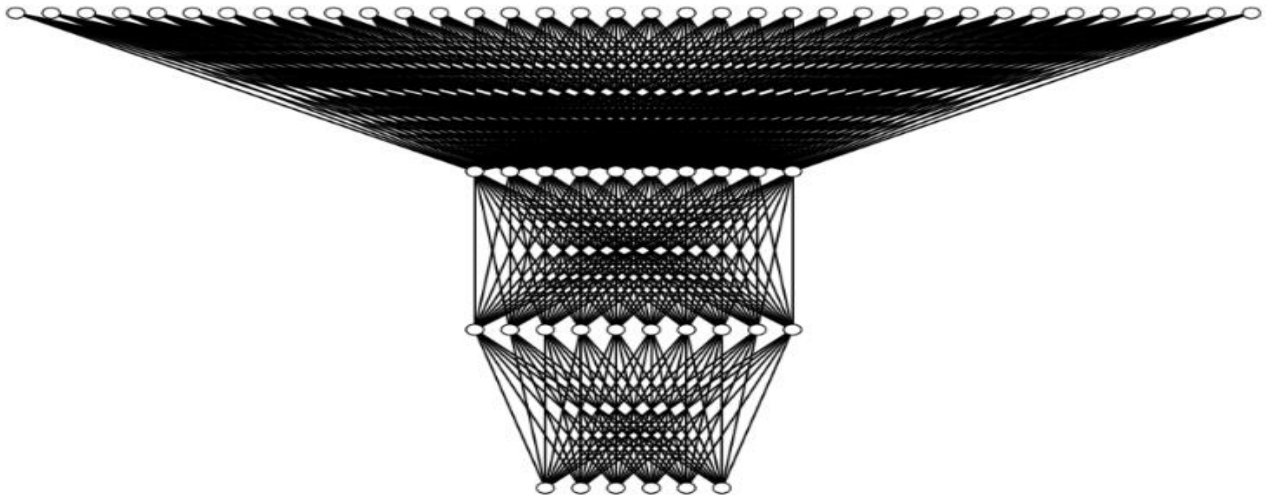


Fig 5.1. Neural Network with 36 input nodes, 2 hidden layer with 10 neurons each and 6 output neurons

(i) Training Error versus Iterations for 3 and 4 layer networks

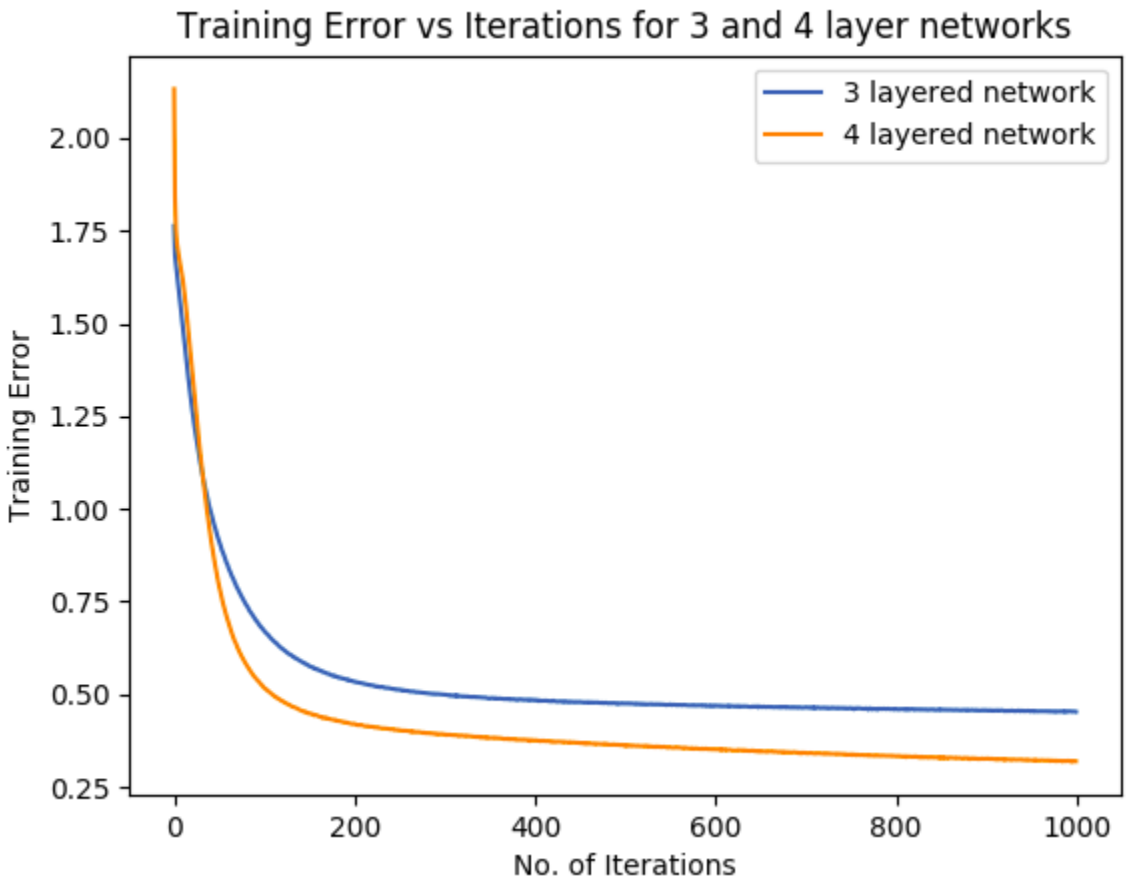


Fig 5.2

We observe that the training error in case of the 4 layered network is considerably lower than that for the 3 layered network, the values being 0.45 and 0.32 for the 3 and 4 layer networks respectively.

(i) Test accuracy versus Iterations for 3 and 4 layer networks

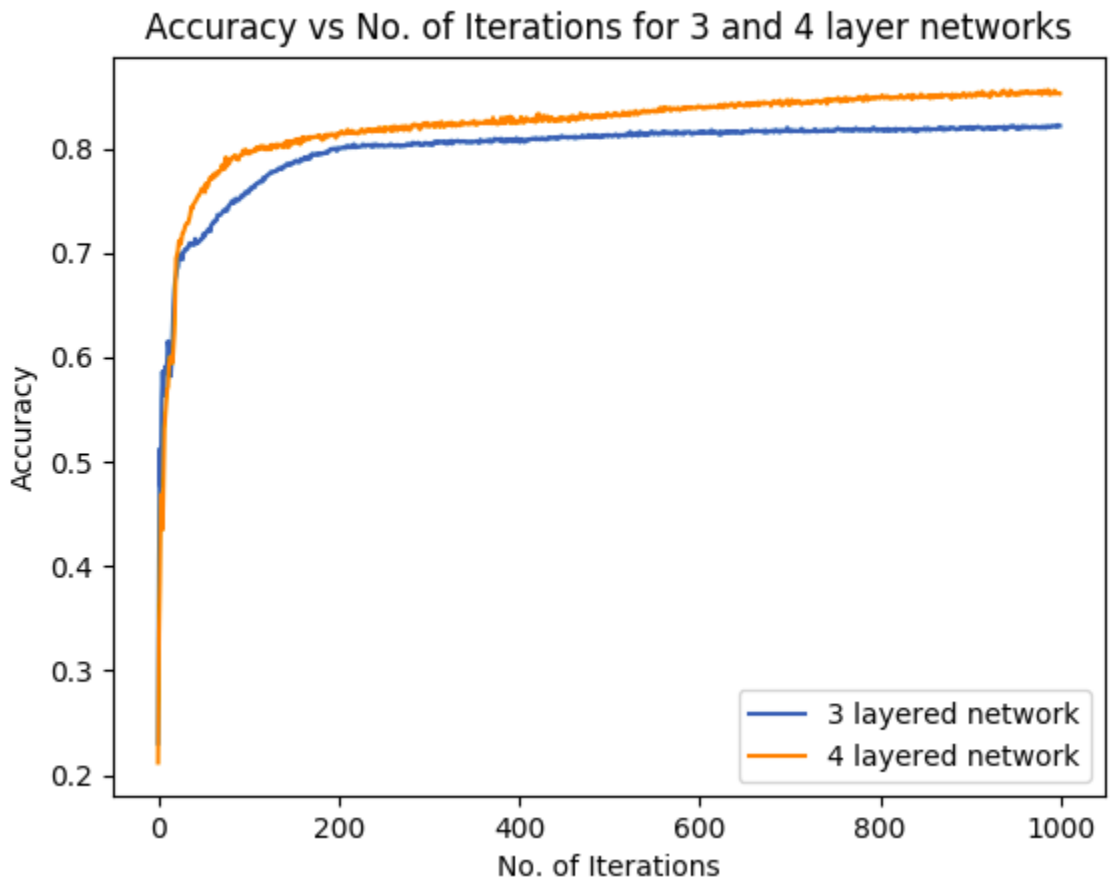


Fig 4.3

It can be seen that a higher test accuracy is achieved by the 4 layer neural network (0.85) as compared to the 3 layered network (0.82).

Conclusion:

From the plots above and our analysis we can safely conclude that deeper networks, i.e. networks with more layers give us better results in terms of accuracy and training cost. A lower training cost and a higher accuracy is achieved by the 4 layer network.

Part B- California Housing Dataset

Introduction:

The California Housing Prices Dataset consists of 8 different attributes of housing complexes in California like location, number of bedrooms, number of bathrooms etc. The data set also contains prices of the houses. The objective of this project is to predict housing prices by training the neural networks with the given attributes. The following experiments give an insight into choosing parameters like the right learning rate, number of hidden neurons, number of hidden layers for obtaining the best model to predict the housing prices.

Train data and Test data:

The California housing price dataset has been divided into subsets for training and testing in the ratio of 70:30 respectively

Five-Fold Cross Validation:

The training data has been further divided into a training subset and the validation subset in the ratio of 4:1. This is called as the five-fold cross validation method where 1/5 of the data is reserved for validation. The 5-fold cross validation method will train the neural network over the whole training dataset and also avoid overfitting.

Epochs:

A 1000 epochs are used for the experiments.

Batch-Size:

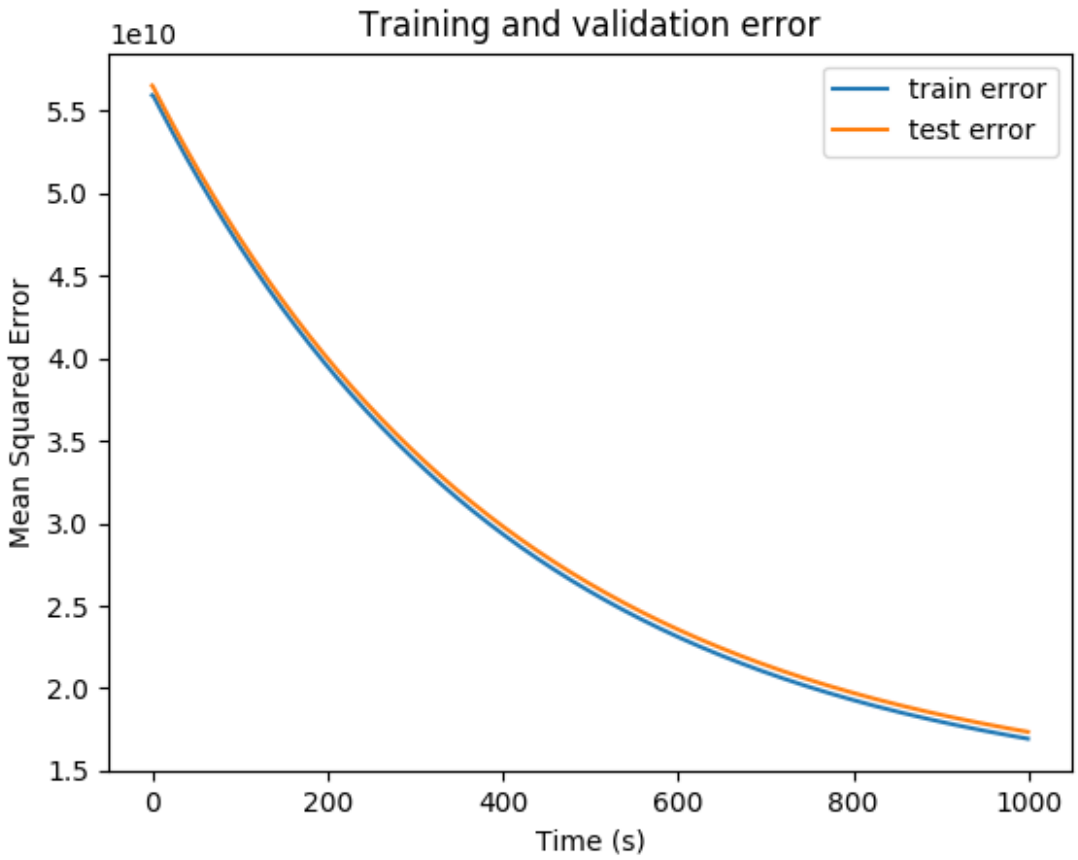
Stochastic gradient descent was initially used but it took a lot of time to converge, so we used gradient descent algorithm for the training of the neural network

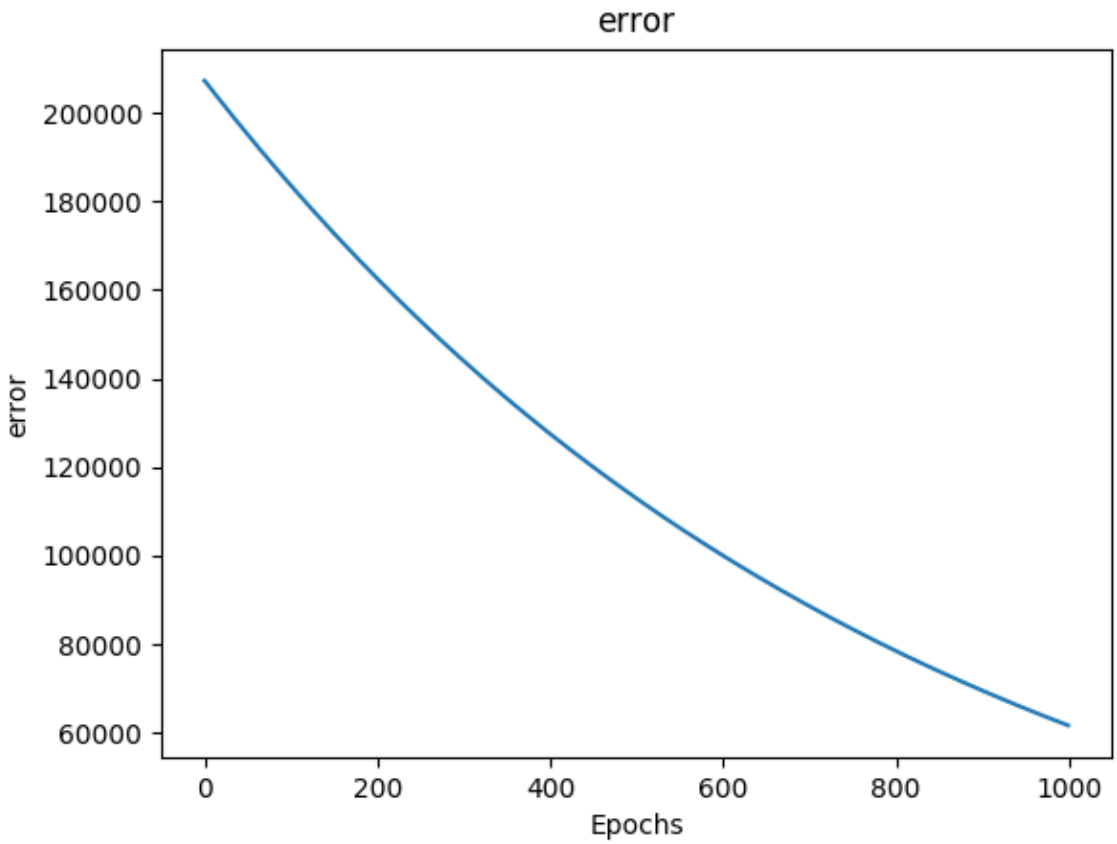
Activation Function:

Sigmoidal activation function was used for the hidden layers and a linear layer was used as the output layer

Task 1: Designing and evaluating the performance of 3-layer neural network

We build a basic 3-layer network and experimented with different learning rates and number of hidden neurons in the hidden layer. The result was as documented below

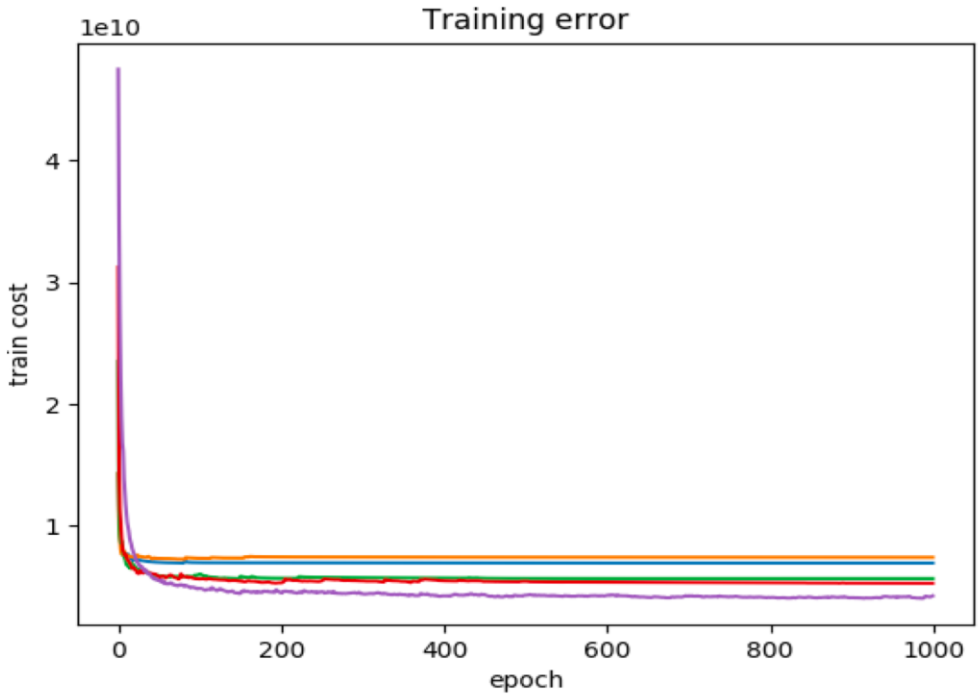


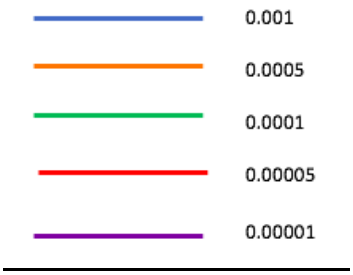

error
200000
180000
160000
140000
120000
100000
80000
60000
error
0
200
400
600
800
1000
Epochs

Task 2: Evaluating the 3-layer network with different learning rates

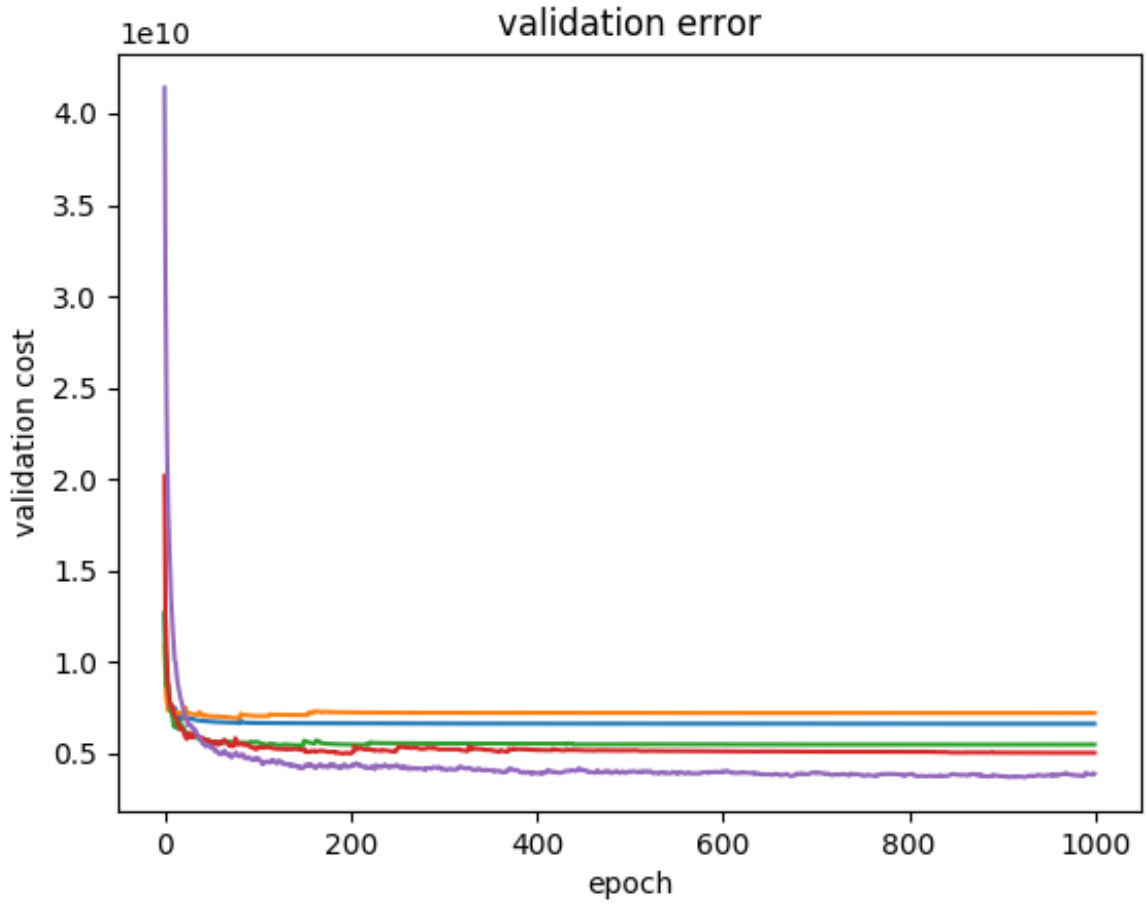
The 3-layer network designed in the stage above was trained with different learning rates using 5-fold cross validation in order to find the optimal learning rates. The different learning rates that were tried were [0.001,0.5*0.001,0.0001,0.5*0.0001,0.00001].
The curves of the training, validation and testing errors against the number of iterations have been plotted below. As we can see the performance of the neural network increased when the alpha value decreased.
As we can see that alpha= 0.00001 converges faster than the other rates and hence that should be chosen as the optimal learning rate = 0.00001 (1e-5) or 10 raised to -5

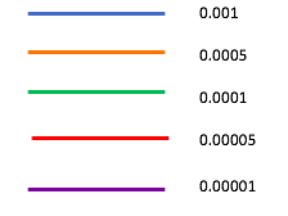
<u>Training error of the different learning rates against the number of epochs</u>





Validation error of the different learning rates against the number of epochs

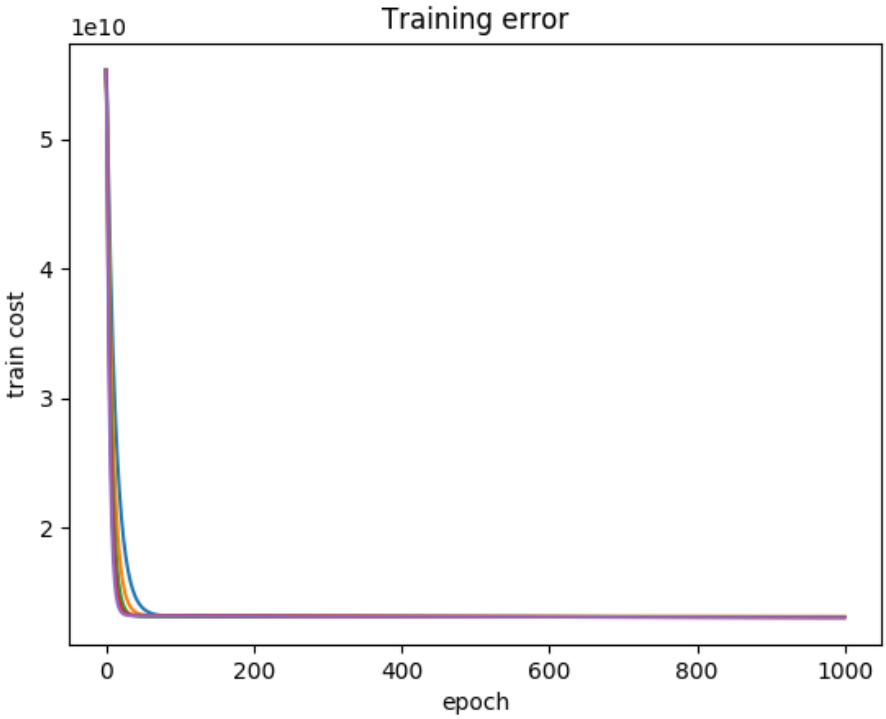


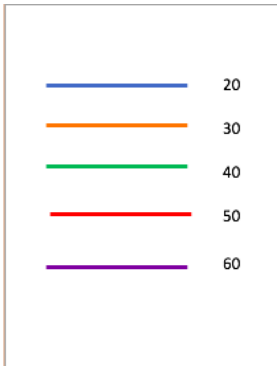


Task 3: Evaluating the 3-layer network with different number of hidden layer neurons

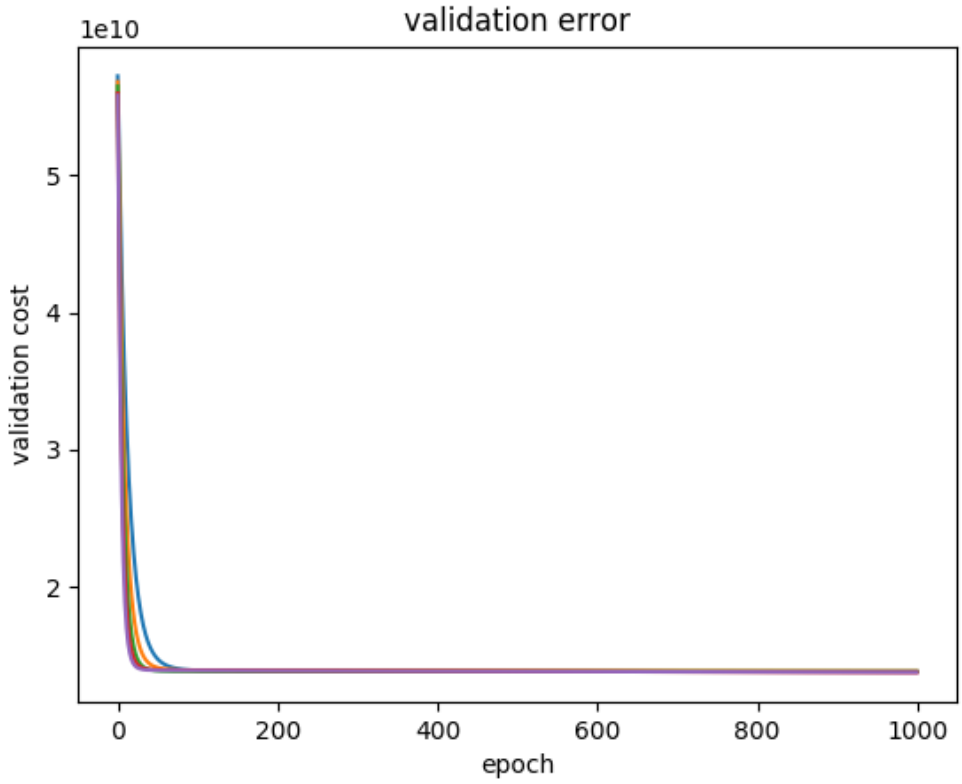
The 3-layer network designed in the stage above was trained with different neurons in the hidden layer using 5-fold cross validation in order to find the optimal hidden neurons. The different number of neurons that were tried were [20,30,40,50,60]. The optimal learning rate chosen in the previous problem was used for determining the optimal number of hidden neurons. We found that the performance improved with the increased number of hidden neurons. **Therefore the optimal number of neurons is 60**

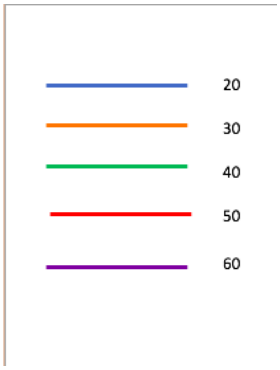
<u>Training error of the different number of layer neurons against the number of epochs</u>





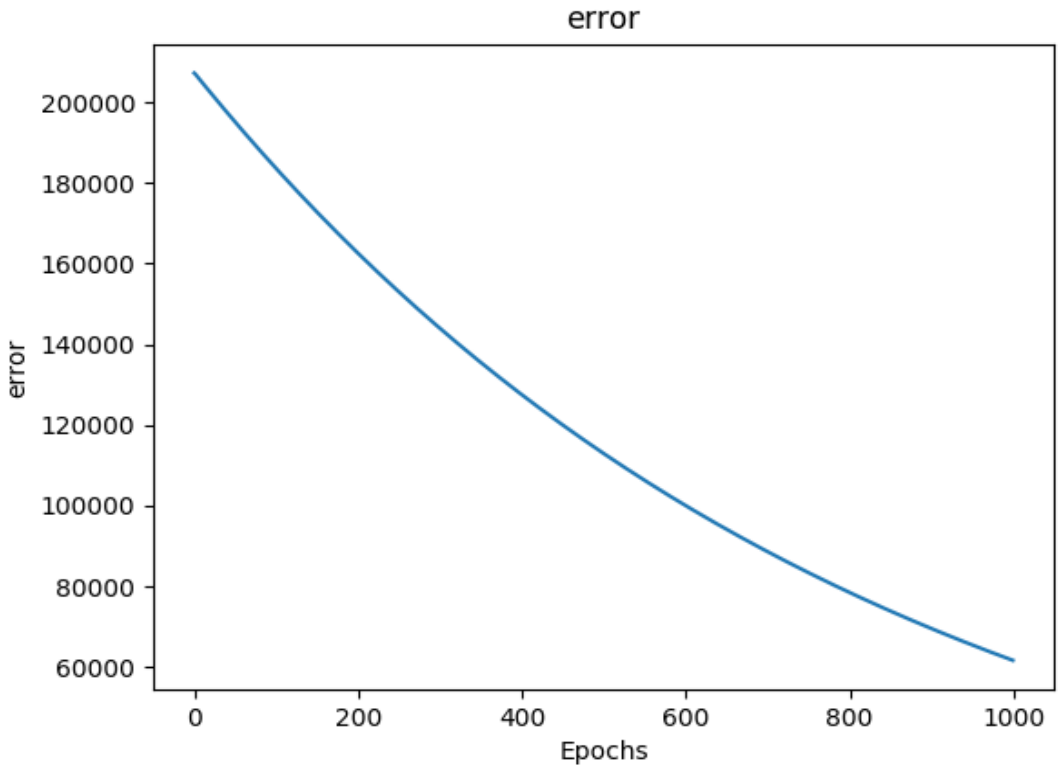
Validation error of the different number of layer neurons against the number of epochs



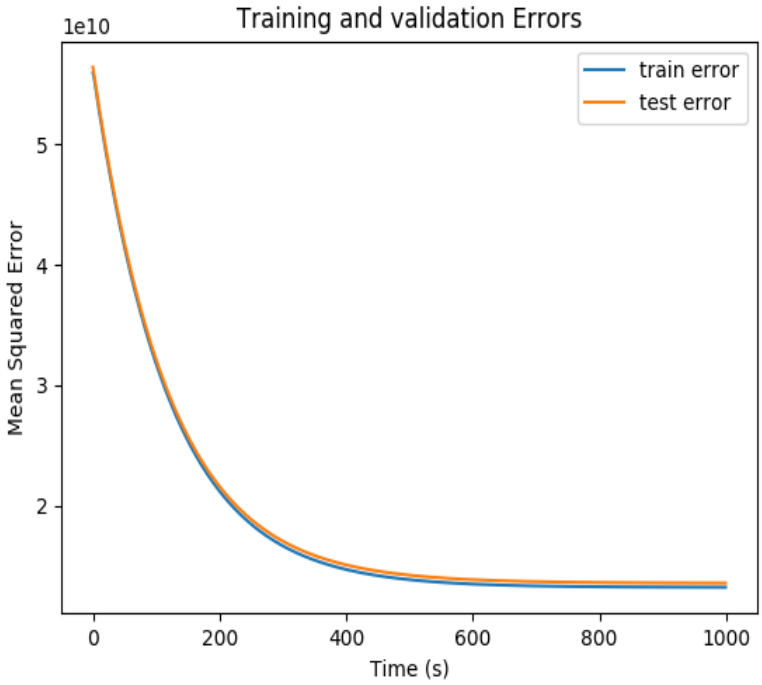


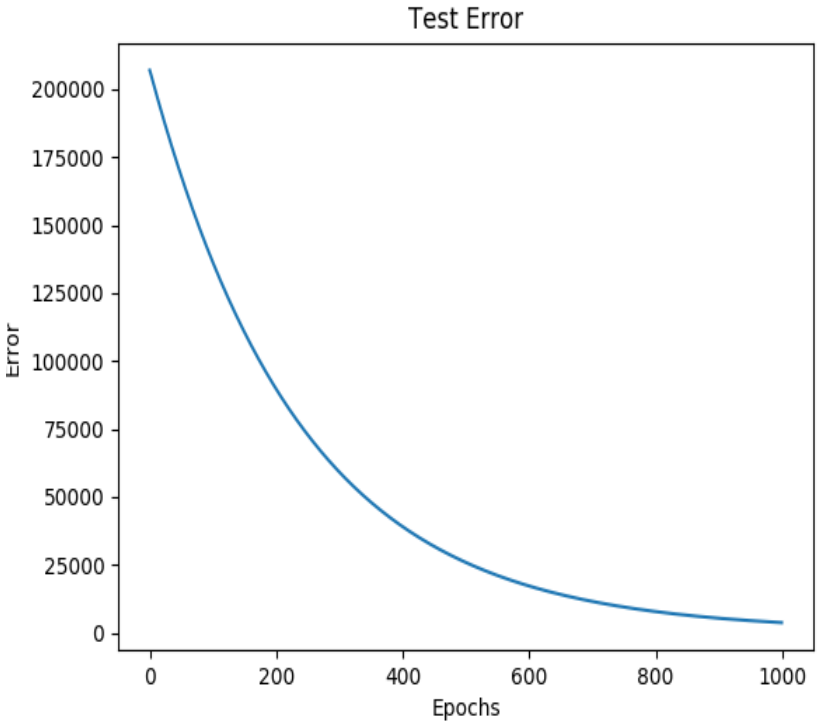
Task 4: Design 4-layer and 5-layer networks

We experimented with four and five layer neurons with optimal neurons found in the first layer i.e. 60 neurons and other layers having 20 neurons each. We used a learning rate of 10 raised to -4. We compared the three four and five layer networks to find the best model. We plotted the errors against the number of iterations. The performance improved from 3 to 4 layer but there was no considerable difference in the 4 and the 5 layers. The optimal number of layers is 4.**3-Layer: Best Model Chosen from previous experiments:**

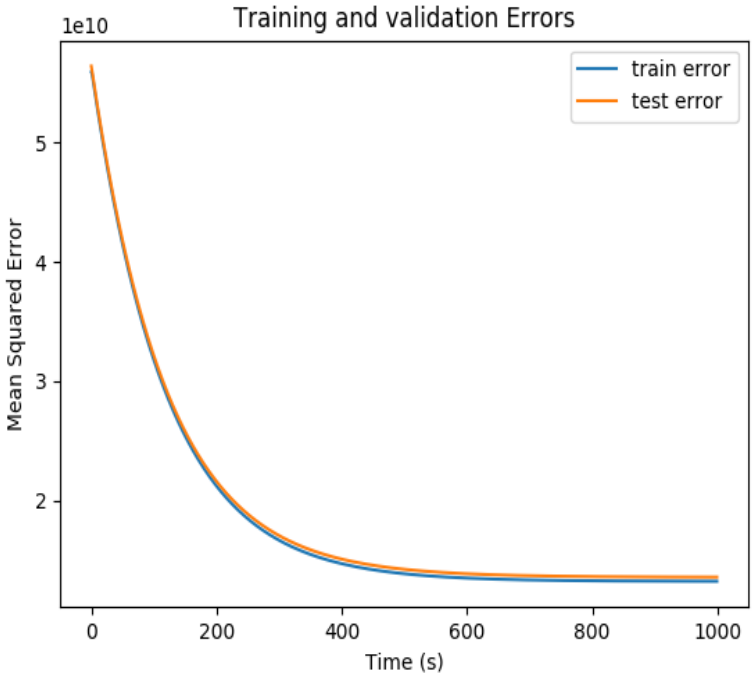


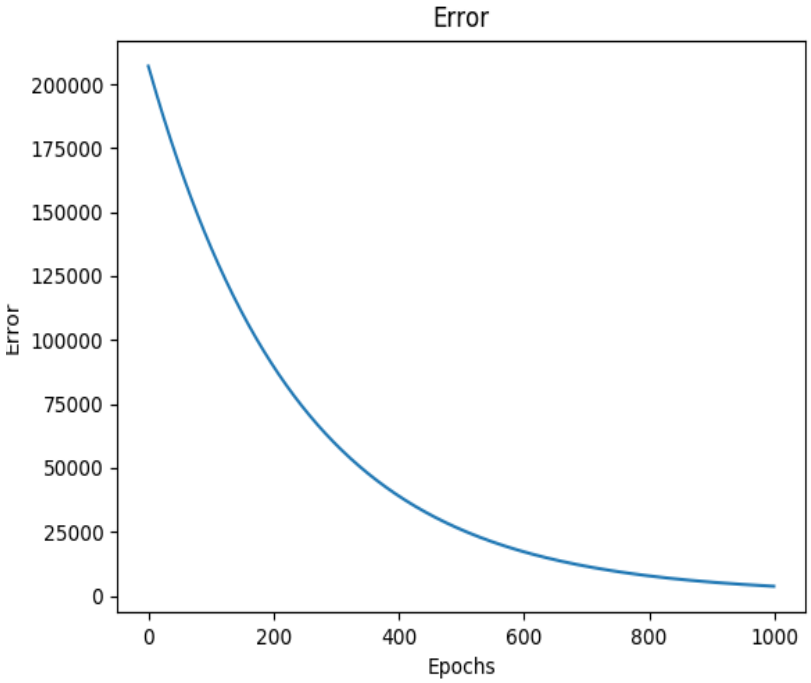
4 layer training and testing errors plotted against the number of epochs:





5-Layer: Training, validation and test errors versus epochs

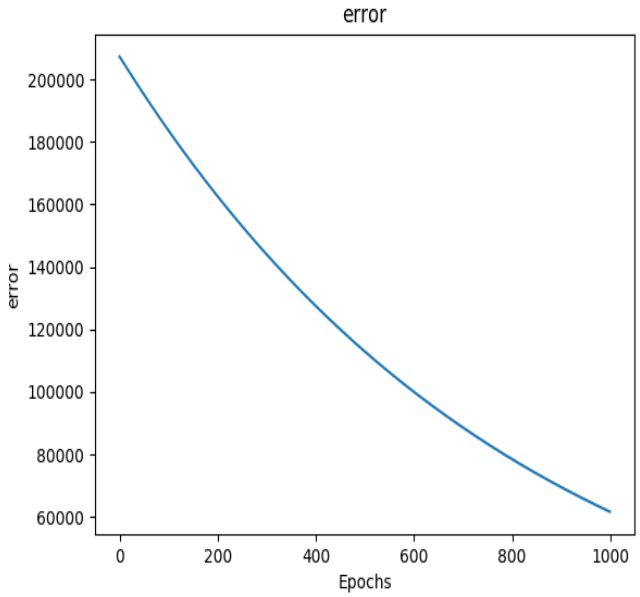


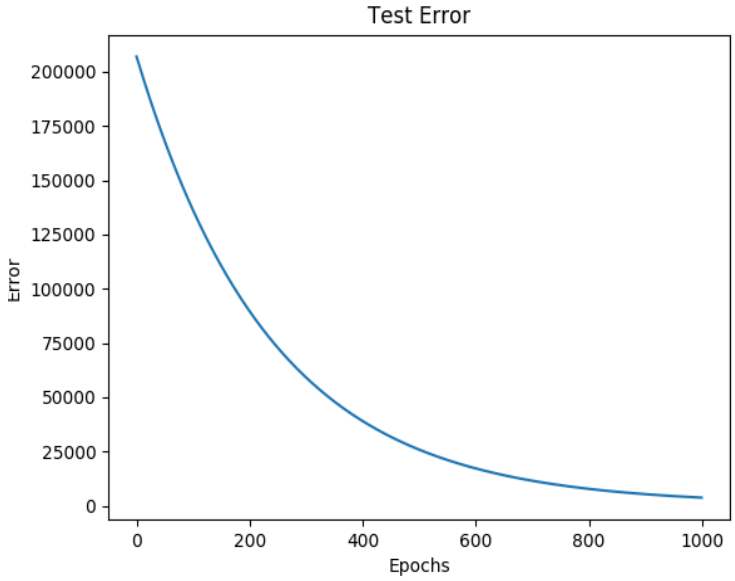


Comparison:

A large difference could be seen in the error function of a 3 layer and a 4-layer neural network but not between a 4 layer and a 5-layer neural network
So, according to the results, a 4 would be the optimum number of layer.
Comparison between 3 layers and 4 layer network.

Comparison between 3 and 4 layer network:





Final model:

Learning rate = 1e-4
Number of layers= 4
Number of neurons in the first hidden layer= 60
Number of neurons in the second hidden layer=20
Activation Function: Sigmoidal activation function in the hidden layers and linear function in the output function.