CZ4042 NEURAL NETWORK AND DEEP LEARNING

ASSIGNMENT 1

LIM MING WEI	U1521477H
GENEVIEVE LAM WEN QI	U1521863H

PART A – CLASSIFICATION PROBLEM

Lim Ming Wei U1521477H

OVERVIEW

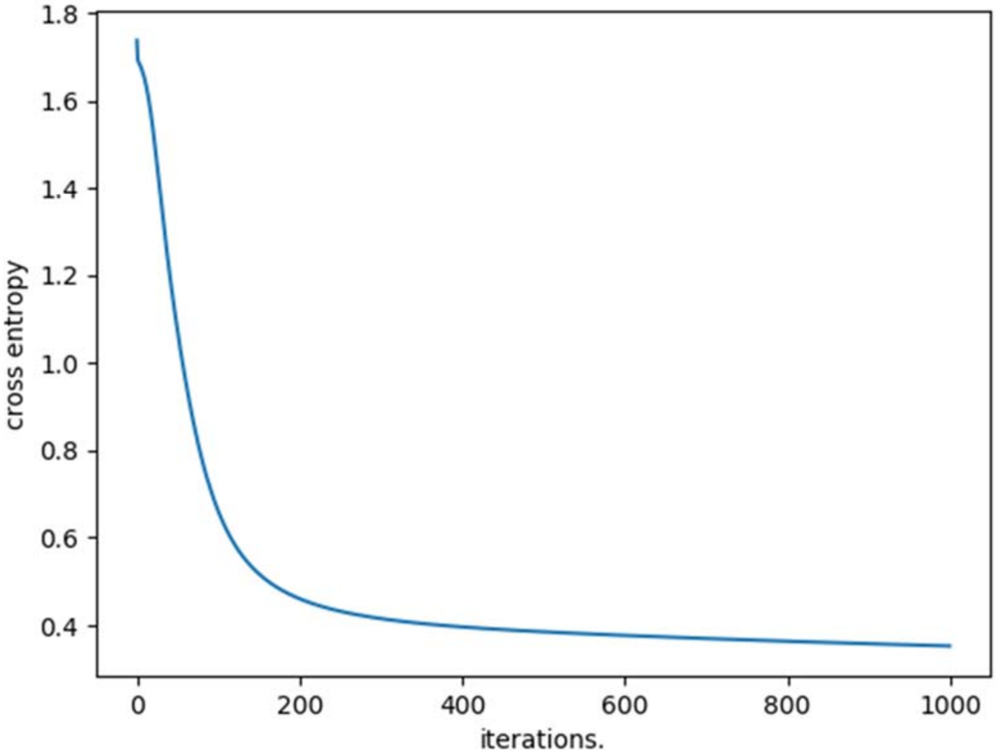
The code was implemented with the libraries found in the starter code file with no additional libraries. The code was run on a standard laptop with no threading and GPU implementation. While there was consideration to optimize the code and implement threading, it was not implemented in the end due to time constraint.

Most of the experiments are conducted using a feedforward neural network with hidden perceptron layer(s) and a output softmax layer, with L2 regularisation implemented. For weight training, minibatch gradient descent is used as per the question requirement.

Sequential mini batch training is used, where if there is a final small batch, we sample the 32 samples of the dataset again. Thus, one epoch may sample more than 4435 rows of data.

RESULTS AND EXPERIMENTS

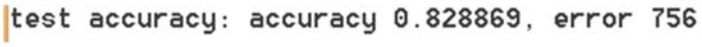
1.Design a feedforward neural network which consists of: an input layer, one hidden perceptron layer of 10 neurons and an output softmax layer. Assume a learning rate α = 0.01, L2 regularization with weight decay parameter β = 10^{-6}, and batch size = 32. Use appropriate scaling of input features.



For all experiments, a we train the model by a default of 1000 epochs, although there was some experimentation with higher epoch counts.

With 1000 epochs the above model gave a train accuracy of approximately 0.86 and cross entropy of 0.35. In a previous run conducted, a training accuracy of approximately 0.95 could be achieved with 30000 epochs.

It is observed that test accuracy is always slightly lower than train accuracy. In this run, we managed a test accuracy of 82.88%, or 756 classification errors.

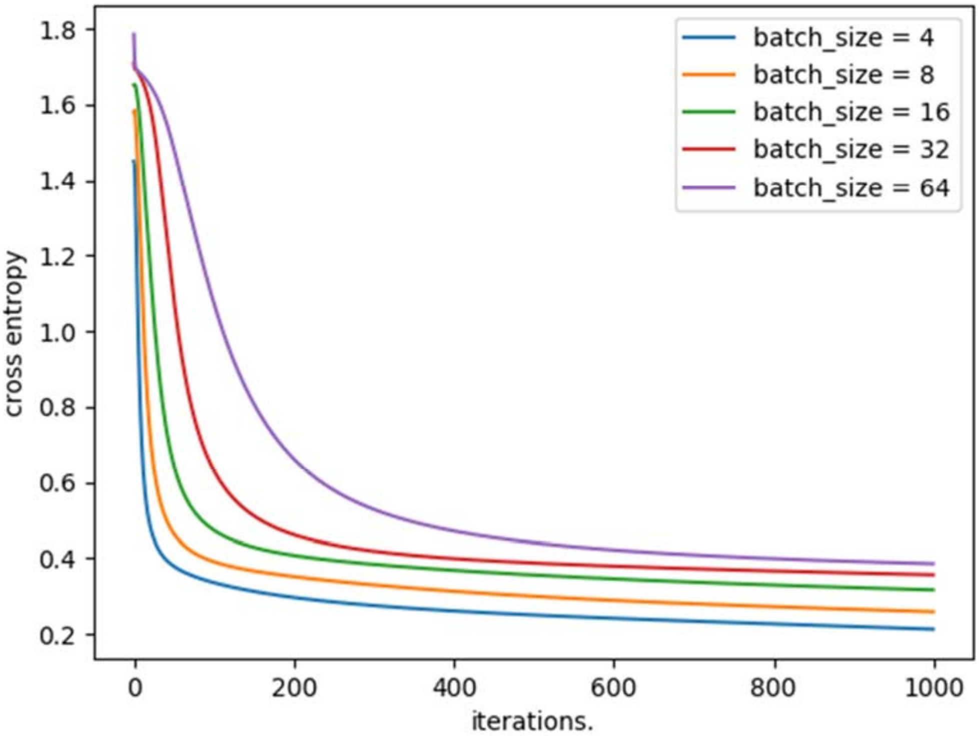


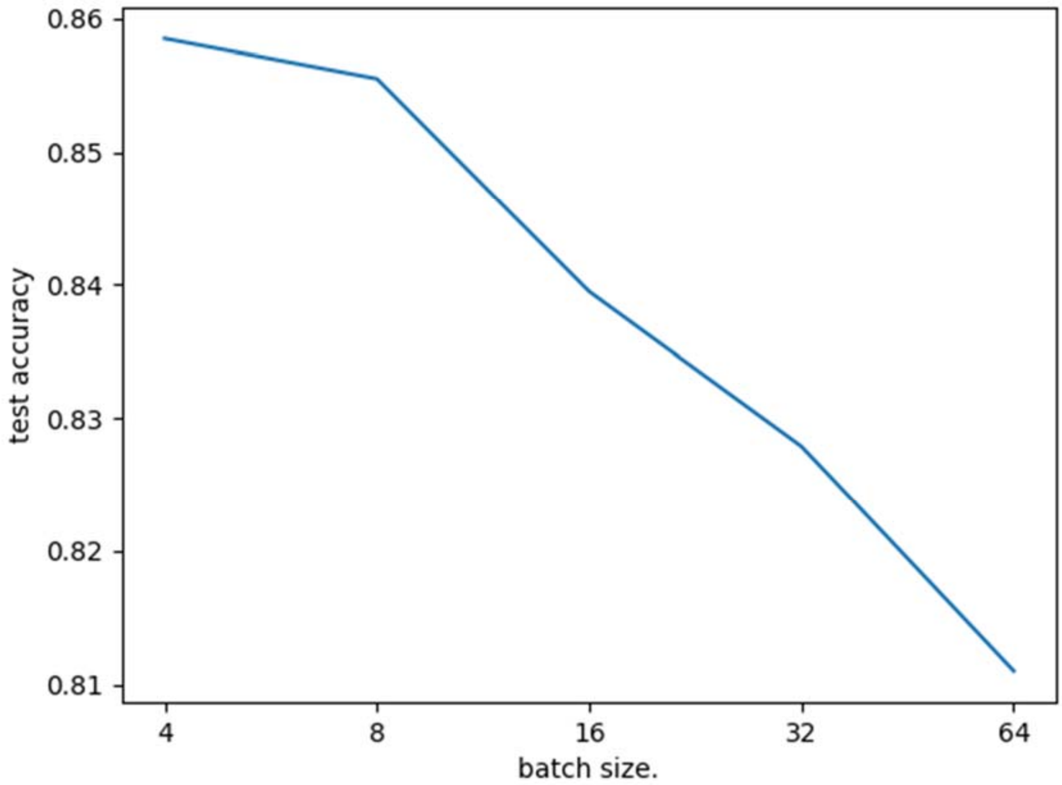
```
test accuracy: accuracy 0.828869, error 756
```

2. Find the optimal batch size by training the neural network by evaluating the performances for different batch sizes.

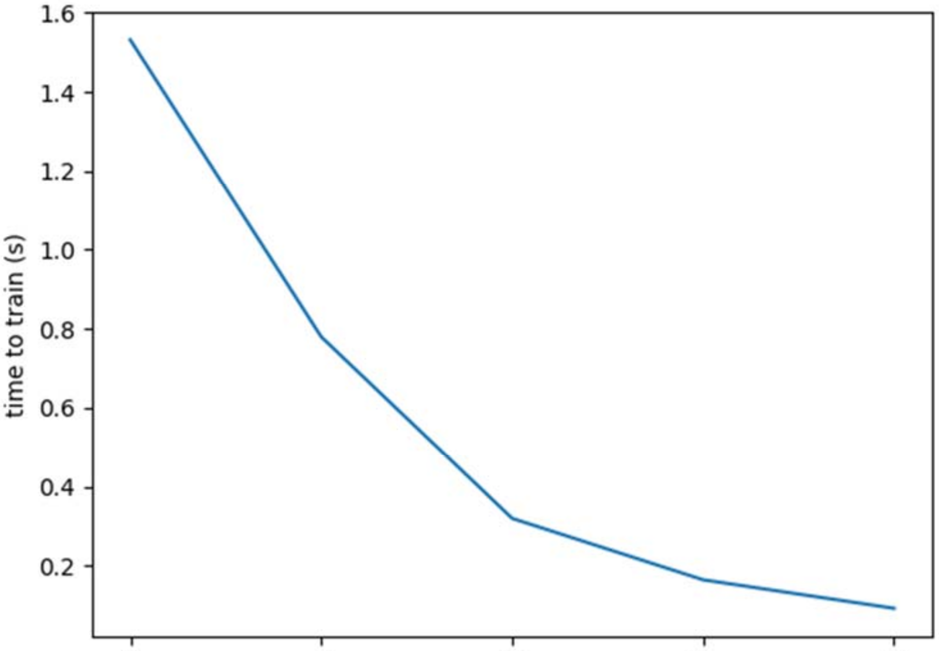
Due to the random implementation of weights, different test results were obtained. Generally it can be observed that entropy near converges is higher when the model is trained at a larger batch size; also, test accuracy is lower with larger batch sizes. However, as seen in graph below for single epoch run time, it is extremely costly (in terms of time). Considering relative accuracy across the batch sizes and the time taken, 16 was decided as the optimal batch size.

a) Plot the training errors and test accuracies against the number of epochs for the 3-layer network for different batch sizes. Limit search space to $S = \{4, 8, 16, 32, 64\}$.





b) Plot the time taken to train the network for one epoch against different batch sizes.



c) State the rationale for selecting the optimal batch size.

As batch size increases, it is able to take advantage of parallelism and matrix computation, resulting in a faster runtime. However, there is less updating of weights and biases per epoch and may result in a loss of accuracy. Hence, in finding the optimal batch size, we obtain the best cost-runtime combination.

3. Find the optimal number of hidden neurons for the 3-layer network designed in part (2).

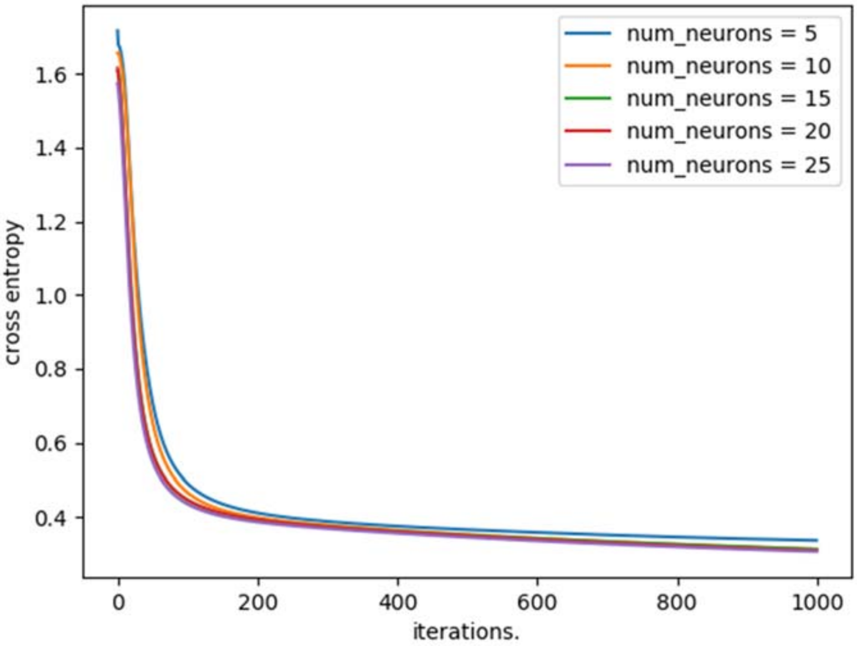
Batch size 16 was used for the subsequent experiments.

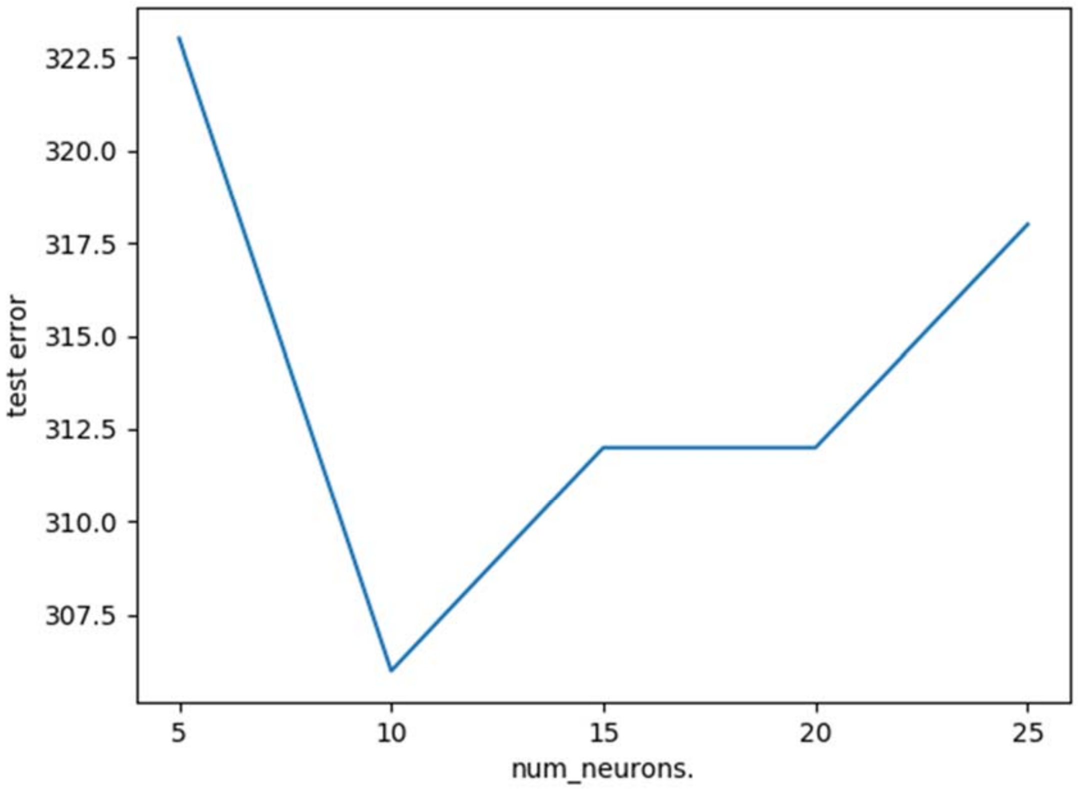
While it is hard to determine the relationship between the number of hidden neurons and cost, the model with 5 neurons had a visibly larger entropy compared to the other models.

However, test error actually increases past a certain number of neurons. We deduce that this may have been caused by overfitting issues.

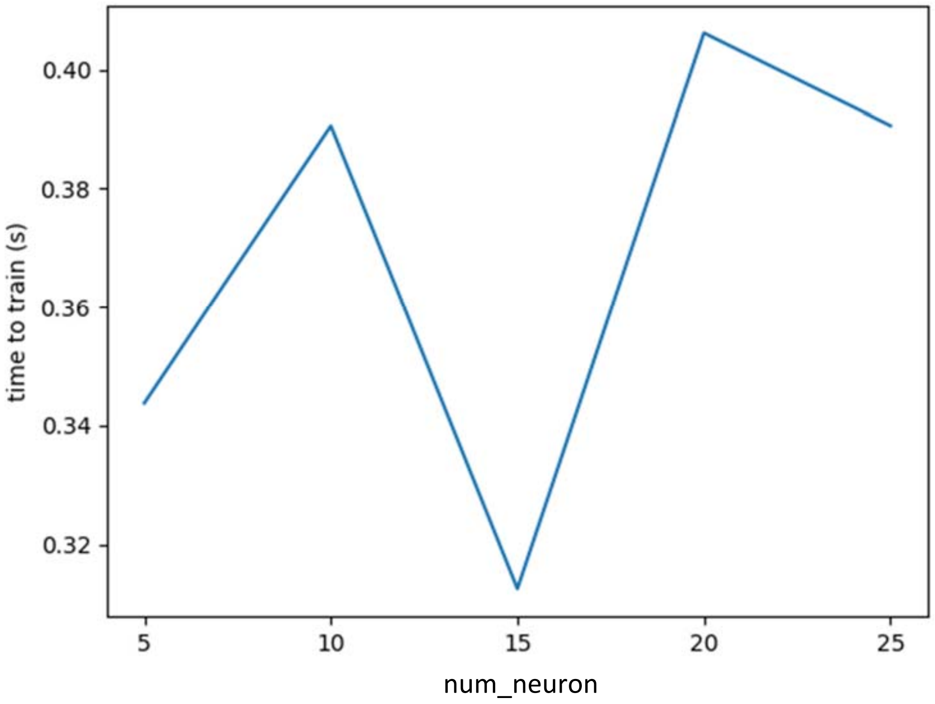
Different test accuracies were derived from multiple runs of the model, with 10 and 15 being the most suitable candidate as they are the most common values to achieve the lowest test error. In this run of the model, 10 was decided as the optimal number of neurons.

a) Plot the training errors and test accuracies against the number of epochs for 3- layer network at hidden-layer neurons. Limit the search space to the number of hidden neurons to $S = \{5,10,15,20,25\}$.





b) Plot the time to train the network for one epoch for different number of hidden-layer neurons.



c) State the rationale for selecting the optimal number of hidden neurons.

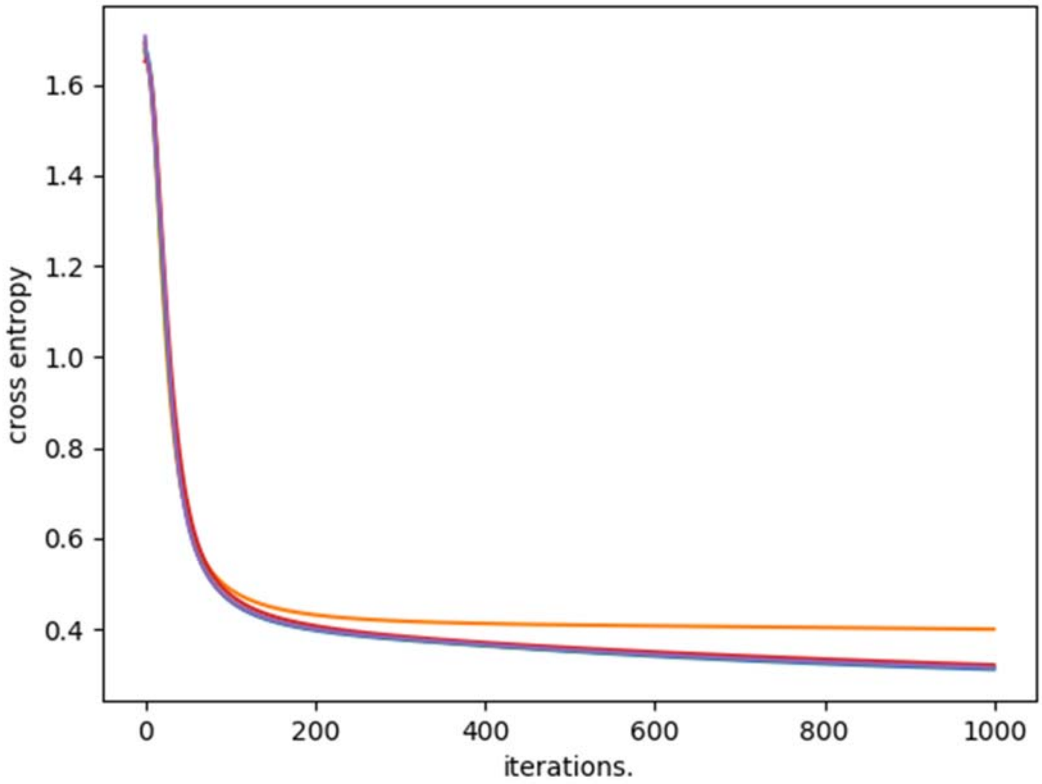
The number of neurons determine the complexity of the decision boundary. With increasing number of hidden neurons, we enable it to better remember training patterns, at the cost of some generalization ability. Selecting an optimal number of hidden neurons helps avoid underfitting and overfitting issues to achieve a better test accuracy.

4) Find the optimal decay parameter for the 3-layer network designed with optimal hidden neurons in part (3).

Batch size 16 and 10 hidden neurons are used.

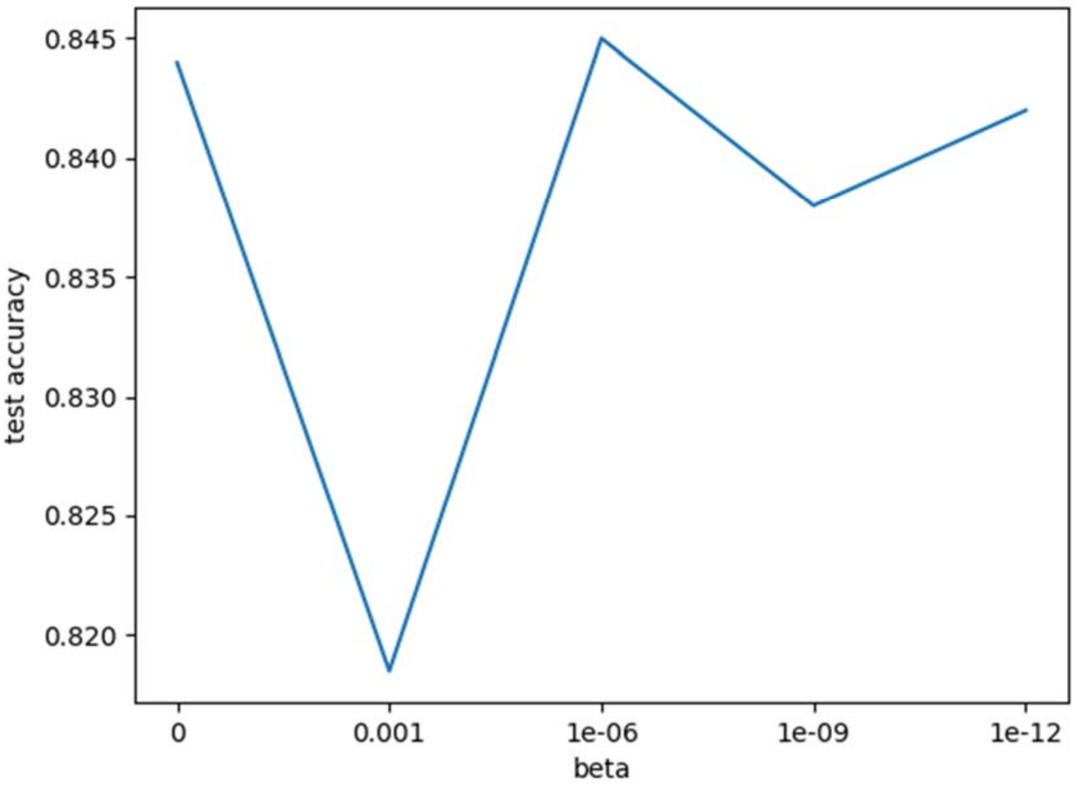
Cross entropy is roughly similar across all decay parameters with the except for β = 0.001. Testing results echo the findings, with β = 0.001 giving significantly higher test errors.
Test errors show similar test errors for all other β-values. β = 10^{-6} is chosen as the optimal decay parameter as it has the lowest error in testing.

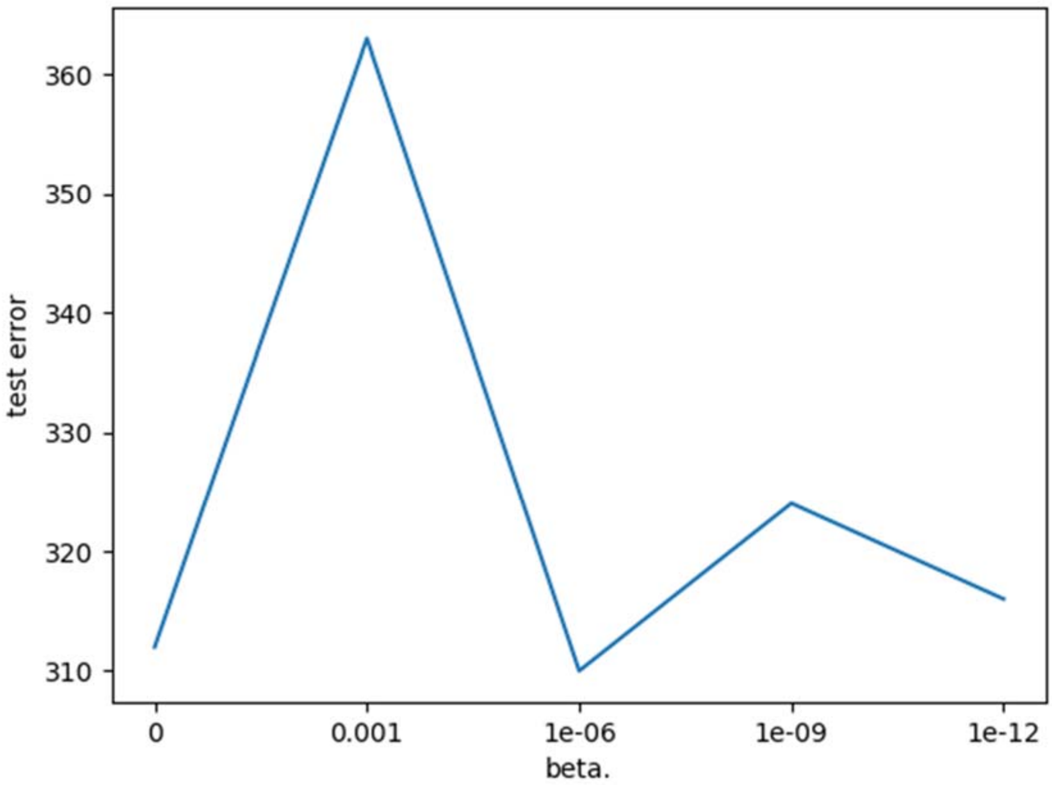
a) Plot the training errors against the number of epochs for the 3-layer network for different values of decay parameters in search space $S=\{0,10^{-3},10^{-6},10^{-9},10^{-12}\}$.



b) Plot the test accuracies against the different values of decay parameter.

Both accuracy and test errors are plotted in graphs below



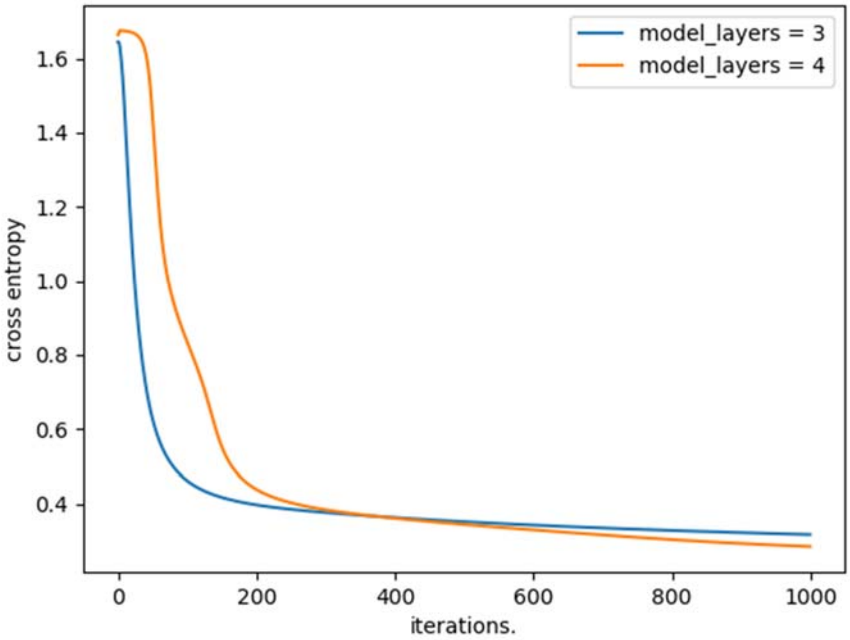


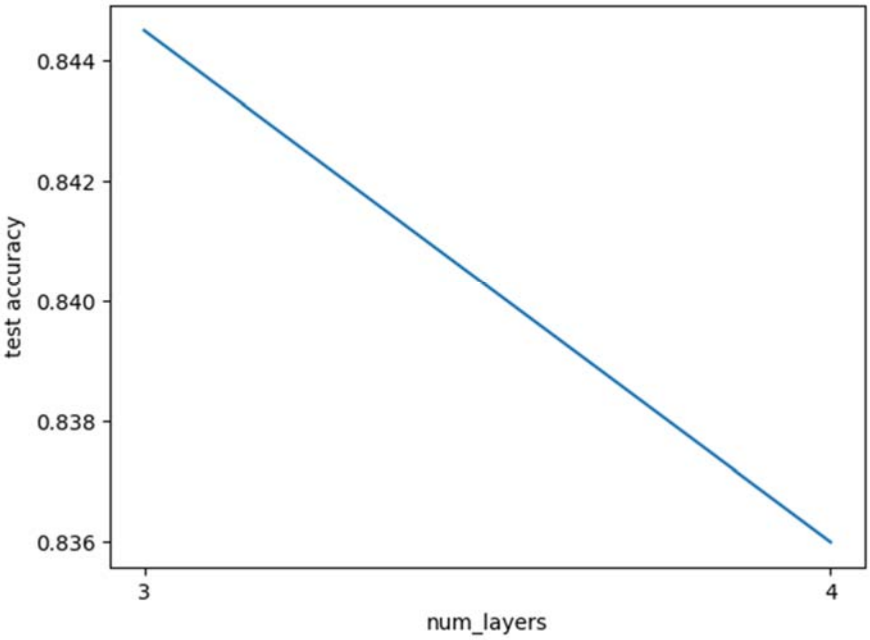
State the rationale for selecting the optimal decay parameter.

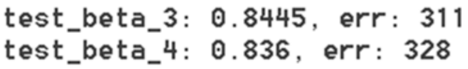
During overfitting, some weights attain large values to reduce training error, jeopardizing its ability to generalizing. To avoid this, a penalty is introduced to the cost function that penalizes large weights.

5) After you are done with the 3-layer network, design a 4-layer network with two hidden-layers, each consisting of 10 perceptrons, trained with a batch size of 32 and decay parameter 10^{-6}.

Plot the train and test accuracy of the 4-layer network.







```
test_beta_3: 0.8445, err: 311
test_beta_4: 0.836, err: 328
```

(typo; test_layers instead of **test_beta)**

Compare and comment on the performances on 3-layer and 4-layer networks.

The 4-layer network takes longer to achieve convergence as compared to the 3-layer network. While the 3-layer network had a higher cross entropy after 1000 epochs, it performed better with the test dataset as compared to the 4-layer network.

This may or may not have been an overfitting issue. Given the tendency for the 4-layer network to achieve a better minima and convergence, it may be worthwhile to explore training the models with larger datasets and practice early stoppage to gauge its generalization ability.

CONCLUSION

Due to random initialization of weights, different results were obtained with each run. Each model for each question part had to be run multiple times in order to confirm our empirical conclusions.
In some cases, such as batch size comparison, there was a general trend of increased accuracy and train time with a smaller batch size. However, for some other parameters, the trend was not apparent and changed with each run.
It can be surmised that the choice of hyperparameters and initial weights do play a significant role in the model performance and convergence, and multiple runs are usually required to tweak them through trial and error.

PART B : REGRESSION PROBLEM

Genevieve Lam Wen Qi U1521863H

INTRODUCTION

Given the California Housing Price Dataset which contains attributes of housing complexes in California such as location, size of house, etc, together with their corresponding prices. We are supposed to build a model that estimate the pricing given its attributes.

ARCHITECTURE AND PRE-PROCESSING

A simple neural network model was constructed based on specifications given in the project. The basic model is as follows: an input layer with 30 neurons, one hidden layer with ReLU activation function and a linear function. Learning is implemented via mini batch gradient descent of batch size 32.

Two additional major benefits for using ReLU are sparsity and a reduced likelihood of vanishing gradient. An L2 regularization on weights prevent overfitting.

HYPERPARAMETER AND MODEL SELECTION

Initial training was conducted on a two-way split of 70:30 ratio to validation and test datasets. Data in each segment was randomly shuffled, and various combinations of the model hyperparameters were trained on the training/validation set.

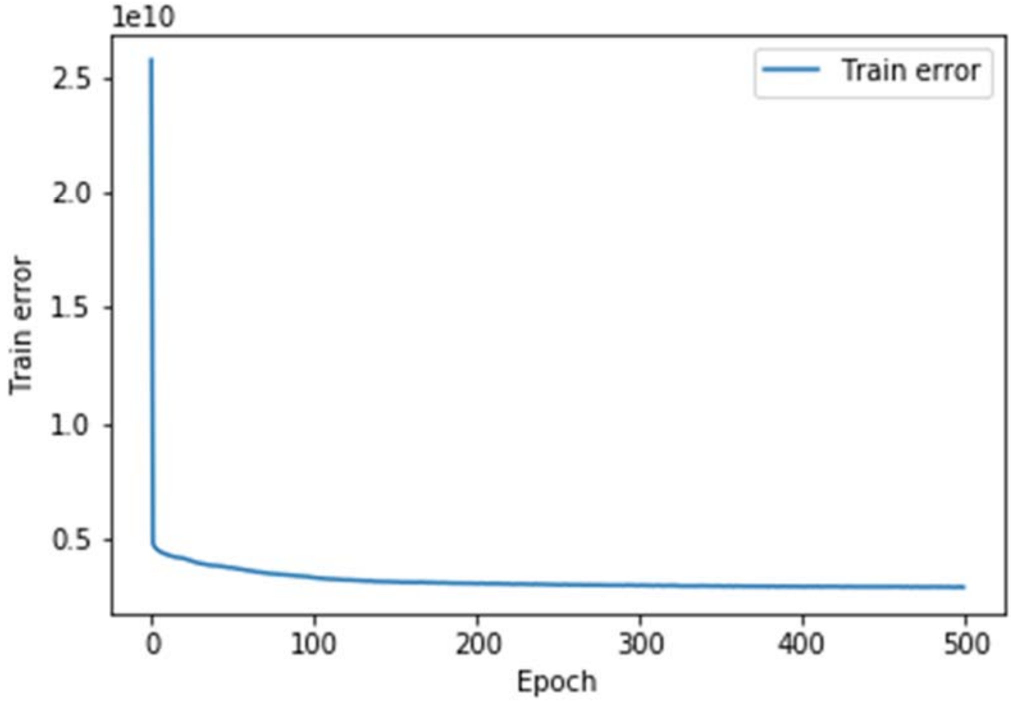
5-fold cross validation is then performed on the validation dataset and each fold is internally shuffled randomly. Various combinations of hyperparameters were set for each fold model and trained in order to find the best performing model.

1. Design a 3-layer feedforward neural network

This model is composed of a 3-layer feedforward neural network defined by given default hyperparameters: weight decay parameter β = 10^{-3} , learning rate α = 10^{-7} and 30 neurons in the hidden layer. 32 is the batch size for the mini-batch gradient descent.

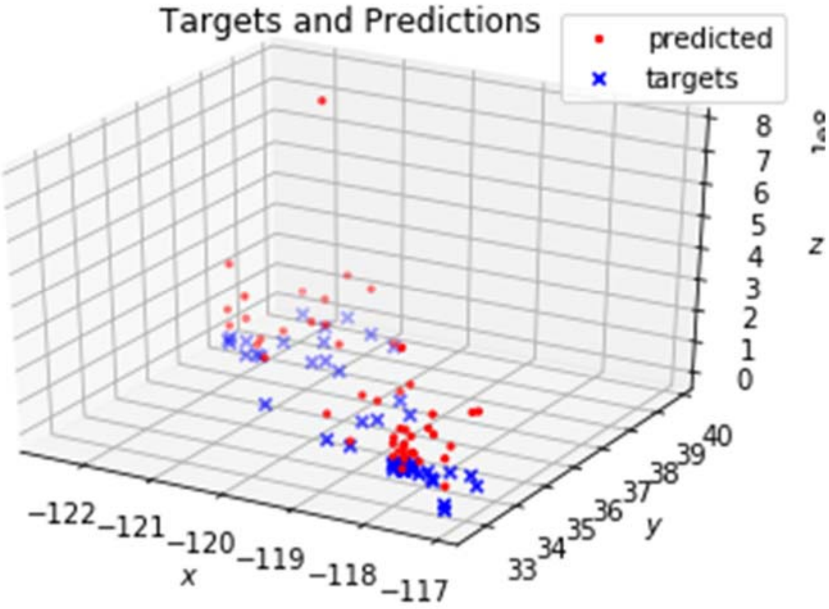
a) Use the validation dataset to train the model and plot validation errors against epochs.

The graph shows loss given by Mean Squared Error(MSE) over 500 epochs.



b) Plot the predicted values and target values for any 50 test samples.

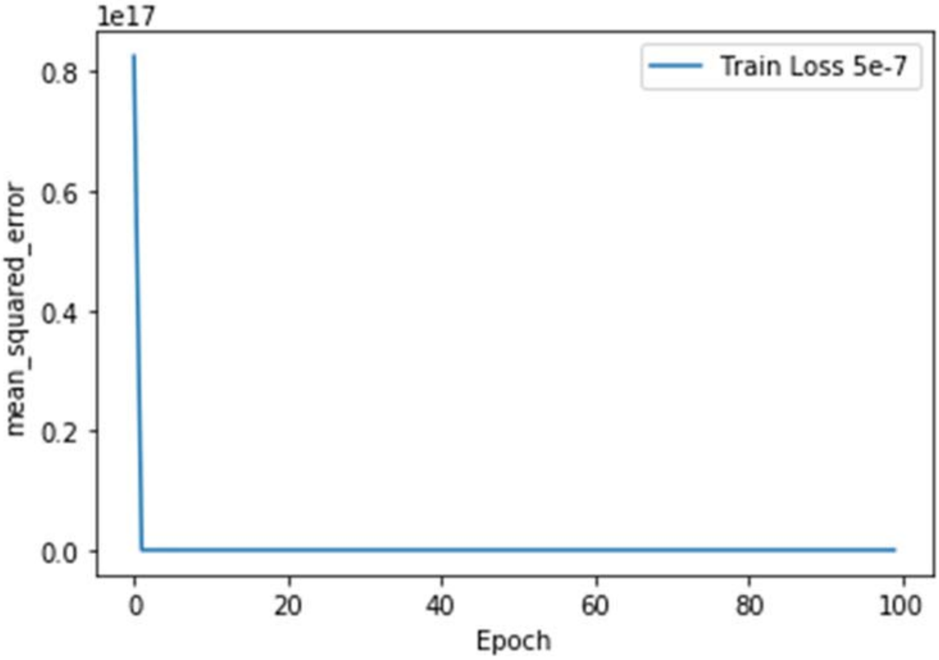
The graph below is the predicted values and the targeted values. As shown, the model is not predicting the values well as it is all over the place. Hence more training or hyperparameter tuning might be needed to obtain a better performing model. We will discuss more in detail in the following sections.



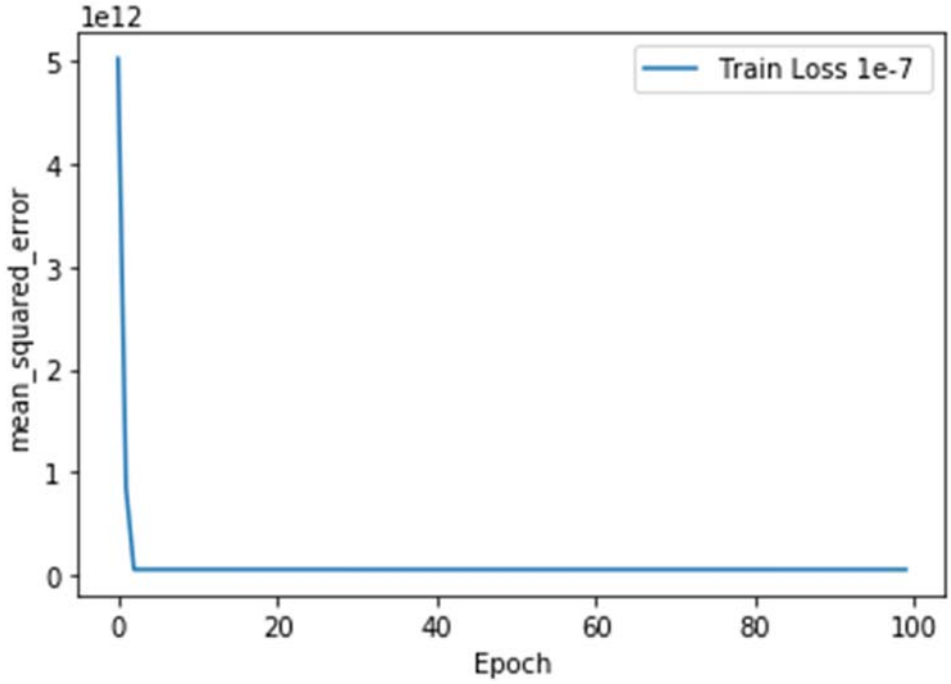
2. Find the optimal learning rate for the 3-layer network designed using 5-fold cross-validation on validation data. Let the search space be: {0.5×10^{-6}, 10$^{–7}$, 0.5×10$^{–8}$, 10$^{–9}$, 10$^{–10}$}.

The model from (1) was used with the defined learning rates above. The graphs was plotted by taking the average of the 5 folds across each learning rate per epoch.

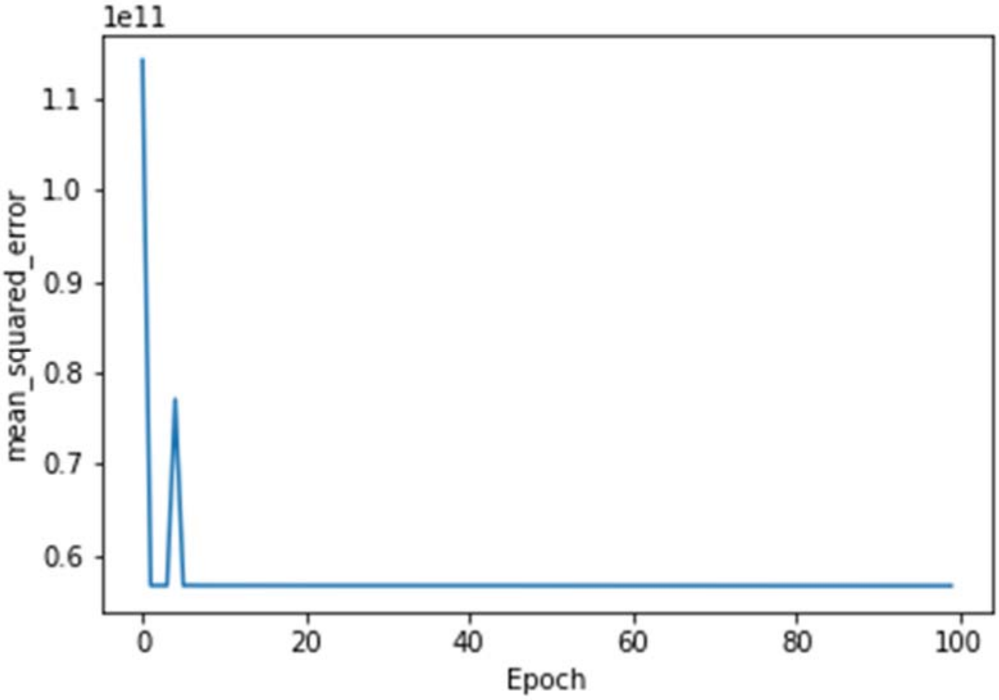
Learning Rate = 0.5x10^{-6}



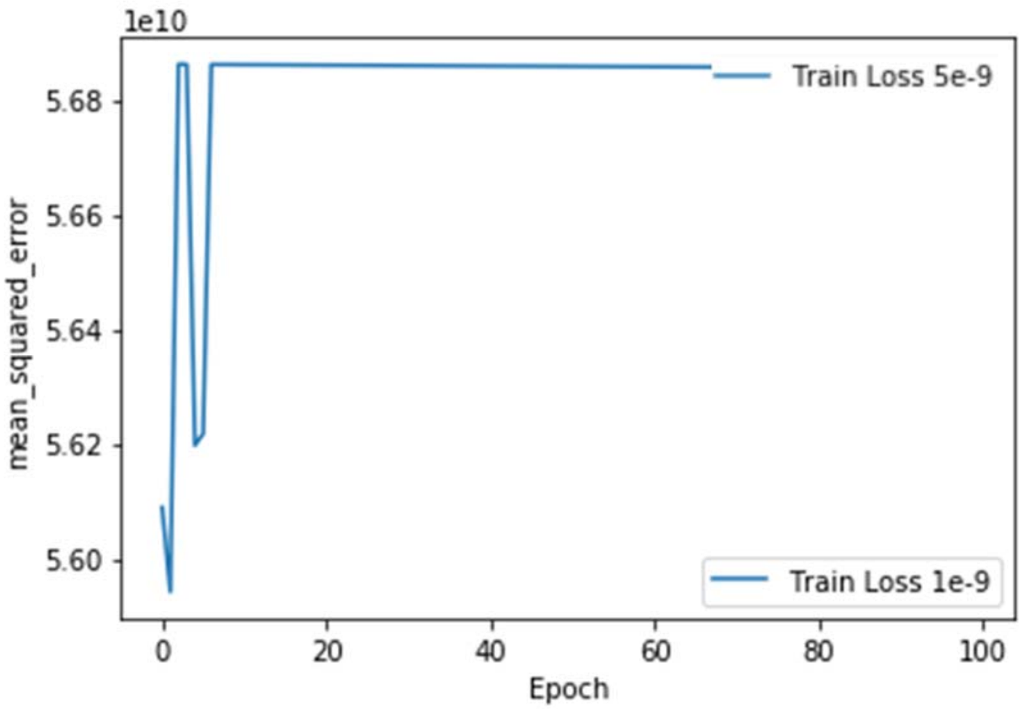
Learning Rate = 10^{-7}



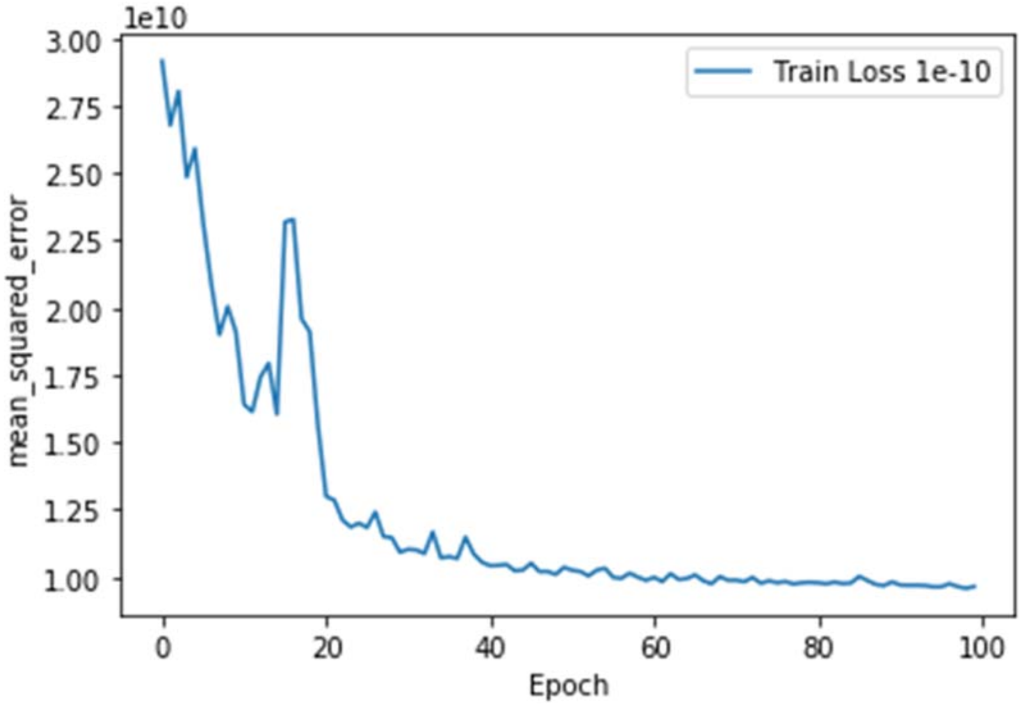
Learning Rate = $0.5x10^{-8}$



Learning Rate = 10^{-9}



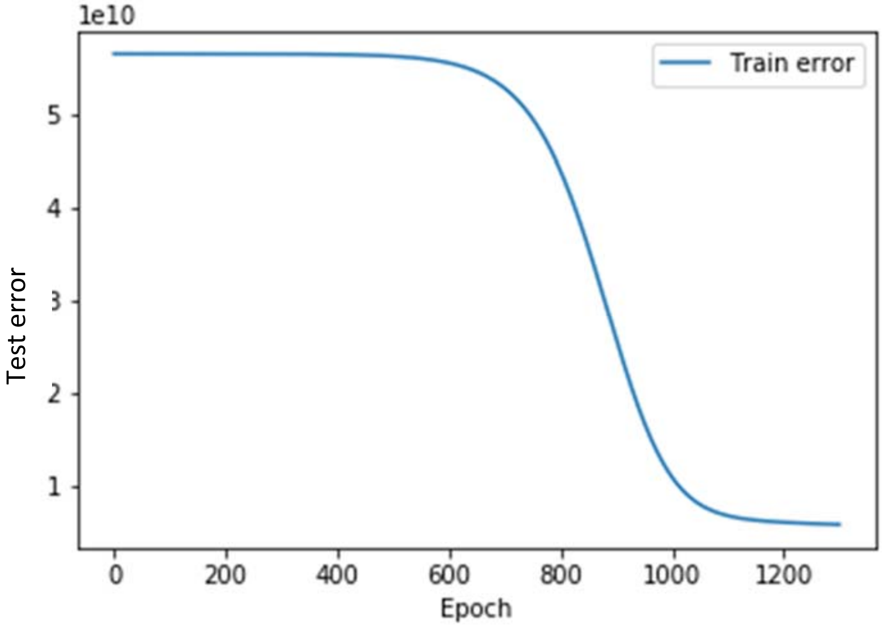
Learning Rate = 10^{-10}



From this run of the models, it is observed that learning rate = 10^{-10} has the lowest validation error given the low MSE derived compared to the other learning rate values. 10^{-10} is chosen as the optimal learning rate.

b) For the optimal learning rate, plot the test errors against training epochs.

The graph below is derived by training the model with the chosen optimal learning rate of 10^{-10} and plotting the test error(MSE) against the number of epochs trained.

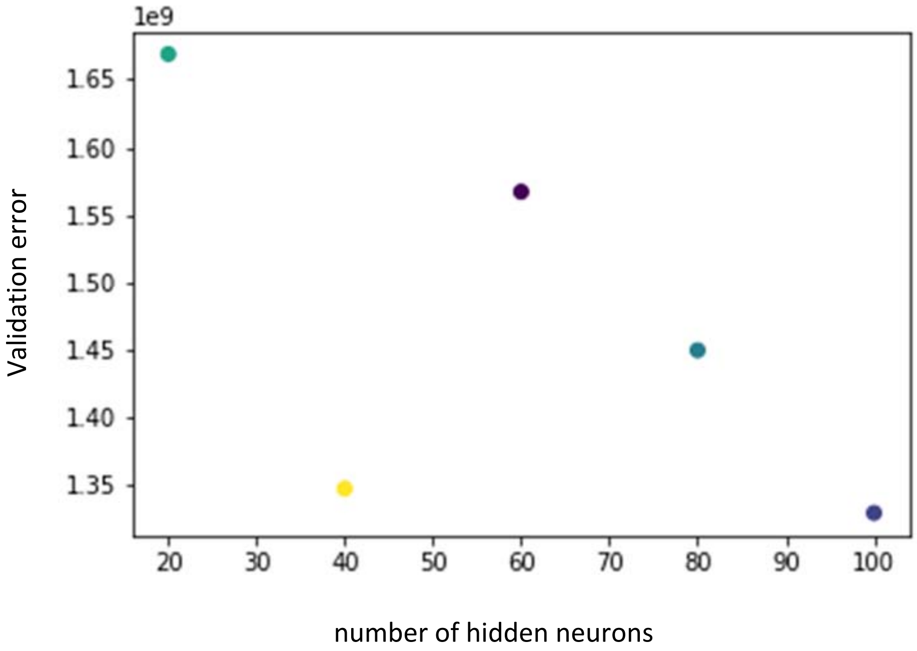


3. Find the optimal number of hidden neurons for the 3-layer network designed. Limit search space to: {20,40,60,80,100}. Use the learning rate from part (2).

The models are trained with an updated optimal learning rate of 10^{-10}. Five different models are trained with the given hidden neuron numbers in order to find the optimal number of hidden neurons.

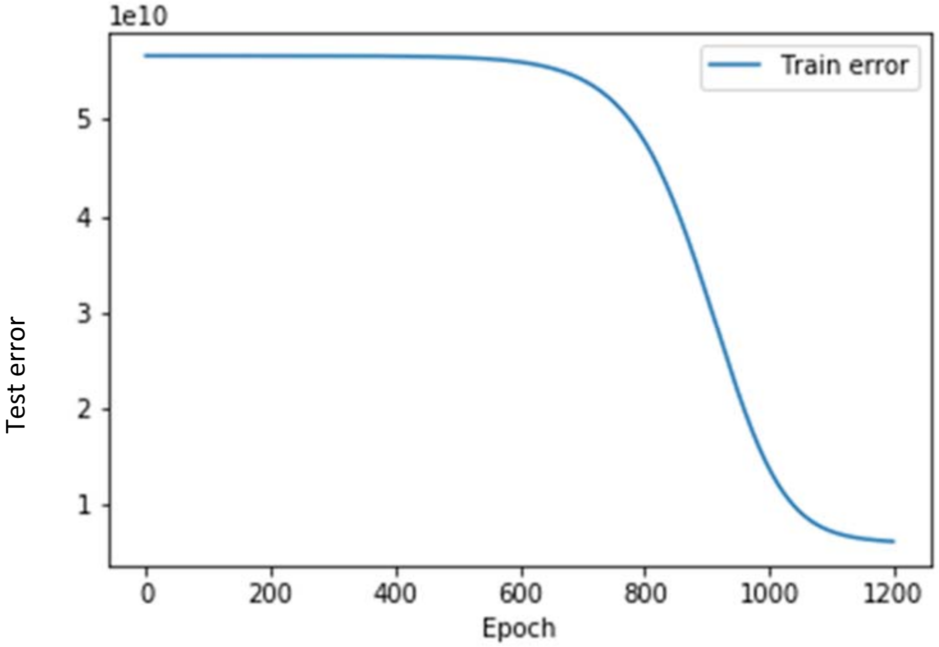
a) Plot the cross-validation errors against the number of hidden-layer neurons.

The scatter plot below is derived by plotting validation error(MSE) against the number of hidden layer neurons in the model. Based on the scatter plot, the optimal number of hidden neurons is 100, as it has the lowest MSE.



b) Plot the test errors against number of epochs for the network consisting of the optimal number of hidden neurons.

A model with 100 neurons preserving all previous parameters was trained with 1200 epochs, and in the graph below test errors are plotted against the number of epochs trained.



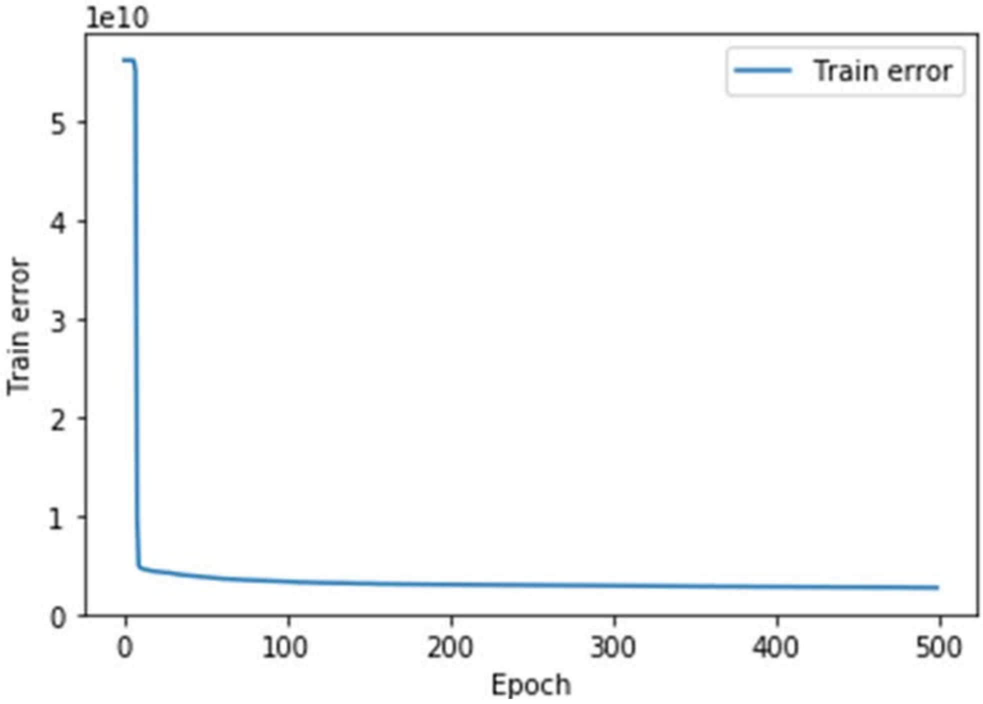
Selecting the optimal number of hidden neurons

As the number of hidden neurons, the network becomes a memory bank is able to recall training patterns but does not perform well on samples that was not part of the training set. Hence it is better to train the with a lower number of neurons. However, with too little neurons, the network might not be able to learn well. Hence, finding an optimal number of hidden neurons helps improves test accuracy by finding a good fit.

4. Design a four-layer neural network and a five-layer neural network, with the first hidden layer having the number of neurons found in step (3) and other hidden layers having 20 neurons each. Use a learning rate of α=10^{-9} for all layers. Train four-layer and five-layer networks on validation data and compare their test errors on test data with those on the three-layer networks.

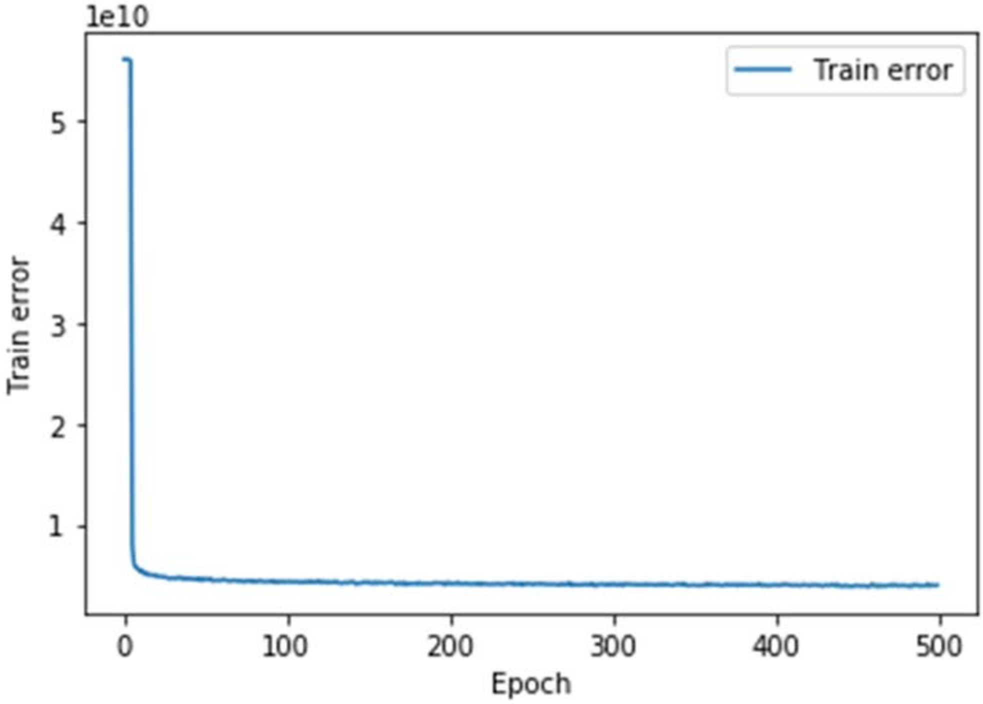
Four-layer neural network

It consists of an input layer, 2 hidden layers with ReLU activation and a linear output layer. The first hidden layer has 100 neurons with $L2$ regularization at weight decay parameter β = 10^{-3}. The second hidden layer consists of 20 neurons. It was trained with 500 epochs and obtained a MSE of 3000619986 (~3.00x10^{9}).



Five-layer neural network

It consists of an input layer, 3 hidden layers with ReLU activation and a linear output layer. The first hidden layer has 100 neurons with $L2$ regularization at weight decay parameter β = 10−3. The following two hidden layers have 20 hidden neurons each. Mini batch gradient descent with learning rate of 10^{-9} and batch size 32 was implemented. It was trained with 500 epoch, and obtained a MSE of 2701991674 (~2.70x10^{9}).



Comparison of 3, 4, 5-layer Neural Network

To keep the comparison fair for all of the models. They were trained with the same training set and validated on the same test set. Amongst the three different neural networks, the 5-layer neural network performed the best with the lowest MSE.

Layers	MSE
3	6444640482 (6.44x10^{9})
4	3000619986 (3.00x10^{9})
5	2701991674 (2.70x10^{9})

Dropouts

The purpose of having dropouts is to prevent model overfitting. Dropouts were placed in between each hidden layer and were done only with the 4-layer and 5-layer neuro network. We used a 0.1 dropout rate to preserve 0.9 probability. For the 4 and 5-layer neural network without dropouts it uses the same neural network presented previously. The following table shows the outcome of the various models.

Layers	Drop Out	MSE
4	No	3000619986 (3.00 x 10^9)
4	Yes	3322430388 (3.22 x 10^9)
5	No	2701991674 (2.70 x 10^9)
5	Yes	3035544581 (3.04 x 10^9)

As shown in the table, the dropouts increased error in the model. It can be deduced that the model has no overfitting issues and may be suffering from underfitting instead.

CONCLUSION

For the given dataset, it is learnt that 10^{-10} is the best performing learning rate, 100 is the optimal number of neuros in the first hidden layer.

Dropouts were not useful in this dataset as it increased error. Finally, the 5-layer neural network was the best performing model as compared to 3 and 4 layers given similar parameters.