CZ4042 Neural Networks

Project 1 Report

Chen Zhiwei (U1620741J)

Zeng Jinpo (U1620575J)

Table of Contents

Part A - Classification Problem

1. Introduction

The project aims at building neural networks to classify the Landsat satellite dataset. The provided dataset contains 36 input attributes (4 spectral bands * 9 pixels in the neighbourhood) and the class label that belongs to {1,2,3,4,5,7}. There are 4435 training data and 2000 test data.

2. Method

2.1 Three-Way Data Splits Method

For this assignment, we have three sets of data: training set, validation set and test set. Original training data is split into training set and validation set in the ratio 3:1. We kept track of the validation error for the application of early stopping, further discussed in Section 2.3.7. In terms of the optimal model selection, instead of selecting the model with the minimum validation error, we would take more metrics into consideration, further discussed in Section 2.3.6.

2.2 Data Pre-processing: Normalization of inputs

Initially, we scaled all input attributes of training set, validation set and test set into [0, 1] by the following formula:

$$\tilde{x}_i = \frac{x_i - x_{i,min}}{x_{i,max} - x_{i,min}}$$

Here, the maximum and the minimum were only calculated over the training set, as the model would not “know” the validation set and test set in advance. Hence, both validation set and test set are scaled using the training data’s maximum and minimum.

This scaling step was introduced to improve the model performance and convergence.

2.3 Model Development

For this assignment, we used mini-batch gradient descent in training the 3-layer feedforward model and the 4-layer feedforward model. The goal of the training is to optimise the L2-regularized cross-entropy cost function.

In this assignment, we had tried to determine the optimal hyper-parameters for the 3-layer feedforward model. We had conducted exhaustive controlled experiments, where each time only one hyper-parameter is explored to determine the optimal value of the hyper-parameter. The hyper-parameter we had experimented for the 3-layer feedforward neural network are:

- batch size,
- number of hidden neurons, and
- weight decay parameter (L2-regularization).

Moreover, we had also experimented with the model performance with early stopping for the 3-layer feedforward neural network, so as to prevent model overfitting and to assess the impact of different hyper-parameters on model convergence time.

2.3.1 Architecture

For Q1 to Q4 of part A, we developed a 3-layer feedforward neural network, with the following architecture:

- an input layer of dimension 36 (corresponding to the input feature dimensions),
- a hidden discrete perceptron layer of n perceptrons with ReLu activation function, and
- an output softmax neuron layer with 6 logistic neurons.

In this assignment, we had experimented with different number of perceptron n in the hidden layer, and the final results are further discussed in Section 3.2.

For Q5 of part A, we developed another 4-layer feedforward neural network, with the following architecture:

- an input layer of dimension 36 (corresponding to the input feature dimensions),
- a hidden discrete perceptron layer of 10 perceptrons with ReLu activation function,
- a hidden discrete perceptron layer of 10 perceptrons with ReLu activation function, and
- an output softmax neuron layer with 6 logistic neurons.

2.3.2 Learning Goal

In this assignment, the above-mentioned neural models aim to minimize L2-regularized cross-entropy loss.

The cross-entropy is the cost function for neural network models learning classification tasks, it is the negative likelihood of the data given by the model:

$$-log\big(p(data|model)\big) = -\sum_{k=1}^{K} n_k log\, p_k$$

The L2-regularization is introduced to penalise the learned weights, so as to improve the generalising ability of the models. During overfitting, some weights attain large values so as to reduce the training error, jeopardizing the model’s ability to generalise. In order to avoid this, the penalty term i.e. regularization term is added to the above cross-entropy cost function:

$$J_1(\boldsymbol{W}, \boldsymbol{b}) = J(\boldsymbol{W}, \boldsymbol{b}) + \beta_2 \sum_{ij} \left(w_{ij}\right)^2$$

In this assignment, we had experimented with a list of different L2 regularization term, and the final results are further discussed in Section 3.3.

2.3.3 Weights Initialisation - Truncated Normal Distribution

Random initialisation is inefficient, and it is desirable that the weights

- are small and near zero to operate in the linear region of the activation function.
- preserve the variance of activation and feedback gradients.

In this assignment, the weights vectors for the two above mentioned neural models are initialised from a truncated normal distribution:

$$w \sim truncated\_normal \left[mean = 0, std = \frac{1}{\sqrt{n_{in}}} \right]$$

In this assignment, we had set the seed when applying tf.truncated_normal, so as to ensure the same random weights initialisation for all experiments. This will ensure a fair weights initialisation among different experiments, and therefore a fair comparison of results.

2.3.4 Mini-batch gradient descent

Mini-batch gradient descent seeks to find a balance between the robustness of **stochastic gradient descent**, with the introduction of the random shuffling of the pairs of the inputs and outputs in each epoch, and the efficiency of **batch gradient descent**. We trained the models batch by batch in an effort to minimize the loss function.

In this assignment, we had experimented with a list of different batch size, and the final results are further discussed in Section 3.1.

2.3.5 Optimising Hyper Parameters

In order to optimize the hyper parameters, we have designed controlled experiments, by holding all other variables constant, while changing one of the following hyper parameters at a time:

(a) Batch Size - [4, 8, 16, 32, 64]
(b) Number of Hidden Neurons - [5,10,15,20,25]
(c) Weight Decay Parameter - [0, 1e-12, 1e-9, 1e-6, 1e-3]

2.3.6 Selection Criteria

To avoid overfitting and improve the generalization of the neural network model we developed, while also ensuring the high performance of the model in terms of model test accuracy, we have set the following criteria in determining the optimal hyper parameter:

(a) Test Accuracy:
 (i) The model with the higher test accuracy is generally better.

(b) Convergence Time:
 (i) The model with shorter convergence time is generally less costly to train, and thus better.

(c) Model Robustness (ability to generalize):
In order to improve the model robustness and prevent overfitting[1], we would limit the capacity of the neural network, by controlling:
 (i) Model complexity:
 We will limit the number of hidden layers and number of units per layer to control the model complexity. The more complex model tends to overfit and reduces the model's ability to generalize. Moreover, it is costlier to train the more complex model. Hence, generally if no under-fitting signal is shown, the less complex model with fewer number of hidden layers or fewer number of units per layer is generally better.

[1] https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec9.pdf

(ii) **Weight decay:**
We will penalize large weights using penalties or constraints. While producing over-fitted mappings requires high curvature and large weights, by applying weight decay to keep the weights small[2], we could foresee smoother mappings, and improve the model's ability to generalize. Hence, the model with higher weight decay is generally better.

(iii) **Batch Size:**
Large-batch methods tend to converge to sharp minimizers of the training function, and tend to generalize less well[3], while small-batch methods converge to flat minimizers characterized by having numerous small eigenvalues of $\nabla^2 f(x)$. Thus, the model with smaller batch size during training is generally better.

2.3.7 Early Stopping

By default, this assignment has set the number of training epochs as 1000. However, it is a common practice in the industry to employ early stopping, so as to:

- prevent overfitting and to improve generalization of the model,
- reduce training costs by avoiding unnecessary training epochs that will not bring significant improvements after the weights have converged.

Thus, in order to prevent overfitting and to improve generalization of the model, as well as to assess the impact of different hyper-parameters on model convergence time, we had decided to introduce early stopping.

When early stopping is applied, 25% of the original training data was randomly sampled as the validation data (the validation data will not be trained) before training. At the end of each training epoch, we kept track of the validation error using the validation data. To decide when to early stop, we introduced another 2 parameters:

(a) Patience (default: 20) - Number of epochs with no min_delta improvement after which training will be stopped.
(b) Min_delta (default: 0.001) - Minimum improvement in the monitored quantity to qualify as an improvement.

For example, if the validation error did not improve by min_delta of 0.001 for consecutive 20 epochs (patience), the training will be terminated early.

[2] http://www.cs.bham.ac.uk/~jxb/INC/l10.pdf
[3] https://openreview.net/pdf?id=H1oyRlYgg

3. Experiments and Results

In this section, we present the experiment findings for different hyper-parameters. The default hyper-parameters, unless otherwise stated, are:

- Batch size = 32
- Number of Neurons in Hidden Layer = 10
- L2-regularized term = 1 x 10^{-6}
- Learning Rate = 0.01
- Patience = 20
- Min_delta = 0.001

In this section, we present the experimental results with early-stopping applied only. For results and plots without early-stopping (i.e. 1000 training epochs), please refer to Section 4. The reasons we had chosen to present early-stopped results only are that

- we found that the test accuracies of early-stopped models are mostly comparable with the non-early-stopped models (subject to all other hyper-parameters hold the same),
- and the early-stopped models are able to provide extra information on how different hyper-parameters affects the model convergence time.

It is worth noting that with early stopping applied, the training time for each experiment had significantly reduced by 8-9 folds.

Also, when assessing the optimal parameters, we will not look at train error, because the models were trained to fit the train data, and thus optimised for the train error. As such, train error is a biased metric to look at. Therefore, we will be focusing on test accuracy to assess the performance of different hyper-parameters.

In addition to test accuracy, we had also computed the **precision/recall/f1** score for the early-stopped models, which is an alternative metric to test accuracy. The results of precision/recall/f1 score largely coincides with the results of test accuracy. In some cases, we would also refer to Appendix Classification report to further strengthen the support for the selection of optimal hyper-parameter.

3.1 Optimal Batch Size = 4

This section answers Q2 of part A.

The experimental results can be summarised into the following table (for the plots required, they can be found in Section 4):

Batch Size	Test Accuracy	Time per Epoch (ms)	Epochs	Total Time (ms)
4	**0.868**	**241.125**	**272**	**65586.071**
8	0.856	125.916	326	41048.765
16	0.836	65.068	198	12883.430
32	0.831	34.806	310	10789.832

64	0.827	20.350	339	6898.703

Apply the 3 criteria for optimal hyper-parameter:

1. [Convergence Time] The total time taken by the model with batch size 4 is the longest (65.6s), while the shortest time is the model with batch size 64 (6.9s).
2. [Test Accuracy] The test accuracy for the model with batch size 4 is the highest (0.868), significantly higher than the rest. Moreover, with reference to Classification report for different batch sizes without Early Stopping, model with batch size 4 has the highest precision, recall, f1-score, significantly higher than the rest.
3. [Model Robustness] The model robustness is affected by batch size. The model with the smallest batch size 4 is more able to generalize than the rest, as shown by the highest test accuracy, and flatter train error curve.

While model with batch size 4 has the greatest training cost, as shown by the longest training time, the model performance is the best, and significantly better than the rest. As we concern more about the model performance with a relatively small data set, it is determined that the **optimal batch size is 4**.

3.2 Optimal Number of hidden neurons = 20

This section answers Q3 of part A.

The experimental results can be summarised into the following table (for the plots required, they can be found in section 4):

Number of Hidden Neurons	Test Accuracy	Time / Epoch (ms)	Epochs	Total Time (ms)
5	0.847	223.908	109	24405.936
10	0.868	241.241	272	65617.532
15	0.858	248.127	227	56324.790
20	**0.875**	**254.776**	**228**	**58088.956**
25	0.862	253.728	213	54044.010

Apply the 3 criteria for optimal hyper-parameter:

1. [Convergence Time] When early stopping is applied, the total time taken is the shortest for the model with 5 hidden neurons (24.4s), and the longest for the model with 10 hidden neurons (65.6s).
2. [Test Accuracy] The test accuracy for the model with 20 hidden neurons is the highest (0.875), significantly higher than the rest.
3. [Model Robustness] In terms of model robustness, intuitively, we will prefer model with fewer number of hidden neurons (5 / 10), as the model will be less complex. However, with reference to test accuracy, it is suggesting the model with 5/10/15 hidden neurons may be under-fitting, while the model with 25 hidden neurons may be over-fitting.

The highest model performance for model with 20 hidden neurons deserves the relatively higher training cost in terms of longer training time. Hence, it is determined that the **optimal number of hidden neurons is 20**.

3.3 Optimal Decay Parameter (L2) = 1 x 10^{-6}

This section answers Q4 of part A.

The experimental results can be summarised into the following table (for the plots required, they can be found in Section 4):

Decay Parameter	Test Accuracy	Time / Epoch (ms)	Epochs	Total Time (ms)
0	0.876	235.092	228	53601.049
1 x 10^{-12}	0.876	242.662	228	55327.047
1 x 10^{-9}	0.875	243.503	228	55518.714
1 x 10^{-6}	**0.875**	**241.574**	**228**	**55078.953**
1 x 10^{-3}	0.851	242.892	188	45663.671

Apply the 3 criteria for optimal hyper-parameter:

1. [Convergence Time] The total time for model with decay parameter 1 x 10^{-3} is the shortest (45.7s), while the total time taken by the other models are relatively close to each other, this means the training cost is about the same for different decay parameters when early stopping is applied.
2. [Test Accuracy] Model with decay parameter 1 x 10^{-3} has the lowest test accuracy (0.851). While the test accuracies for the other models are relatively close to each other (within 0.1%). While model with decay parameter 1 x 10^{-3} has the lowest training cost in terms of the training time, the trade-off of a comparatively lower test accuracy is not worthy. Thus, we will consider the other decay parameters as the potential candidates for optimal decay parameter.
3. [Model Robustness] Model with the higher decay parameter generally has the greater ability to generalize when handling unseen data. Hence, we will prefer model with decay parameter 1 x 10^{-6} over the rest, when the model performances, in terms of test accuracy and training cost in terms of total training time, are similar.

Hence, it is determined that the **optimal decay parameter is 1 x 10^{-6}**.

3.4 Optimal Number of Layers = 2

This section answers Q5 of part A.

The experimental results can be summarised into the following table (for the plots required, they can be found in Section 4):

Number of Hidden Layers	Test Accuracy	Time / Epoch (ms)	Epochs	Total Time (ms)
1	0.831	36.413	310	11288
2	**0.848**	**43.690**	**200**	**8738.03**

Apply the 3 criteria for optimal hyper-parameter:

1. [Convergence Time] The model with 1 hidden layer (11.3s) has a longer training time than the model with 2 hidden layers (8.7s), as the model 2 hidden layers requires much fewer training epochs for early stopping, despite it having longer training time per epoch.
2. [Test Accuracy] The model with 2 hidden layers has a higher test accuracy (0.848), than the model with 1 hidden layer (0.831).
3. [Model Robustness] In general, we would prefer the model with fewer number of hidden layers to prevent over-fitting and improve the model's ability to generalize. However, the test accuracy improves from the model with 1 hidden layer to the model with 2 hidden layers, showing that the model with 1 hidden layer is less robust than model with 2 hidden layers.

Hence, in consideration of both training cost and model performance, it is determined that the **optimal number of hidden layer is 2**.

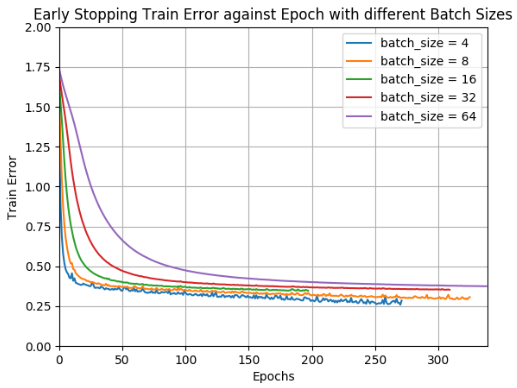
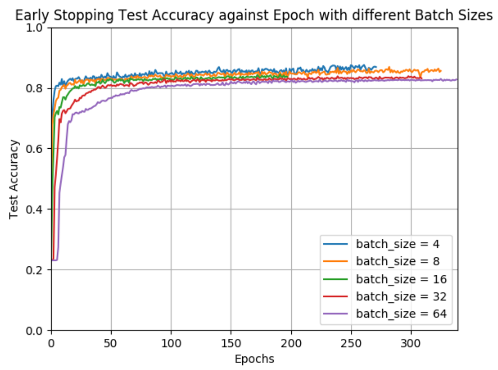
4. Conclusion

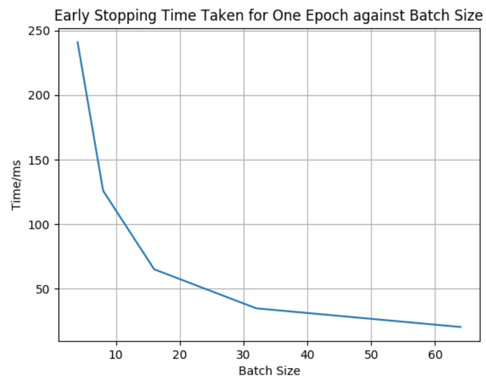
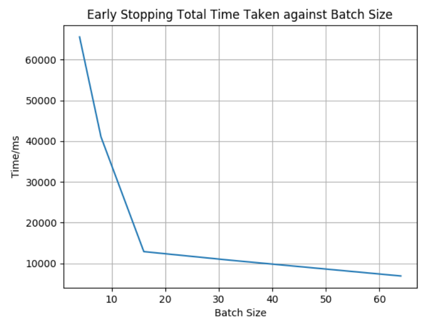
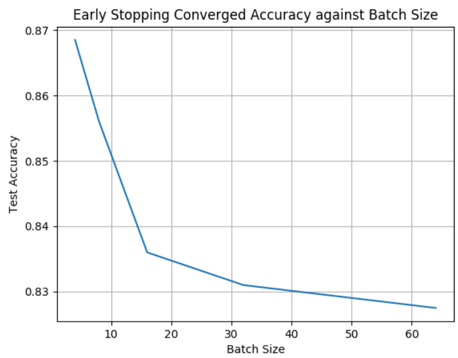
In conclusion, the optimal hyper parameters should be:

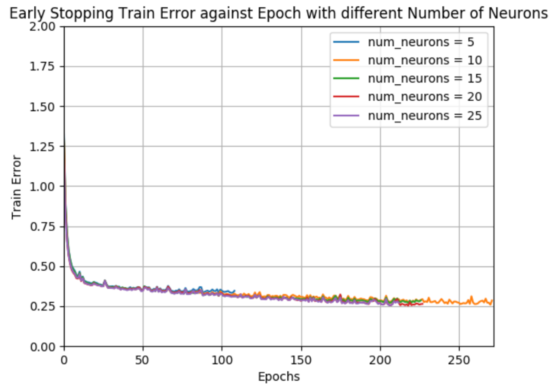
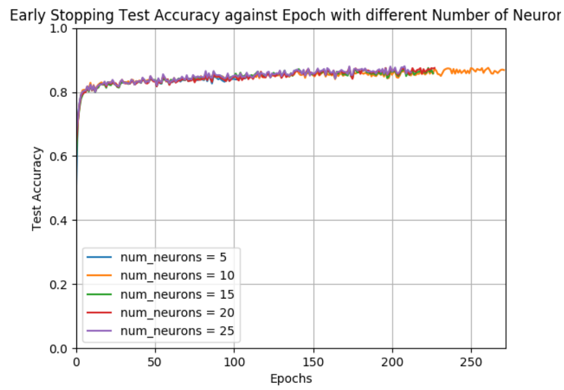
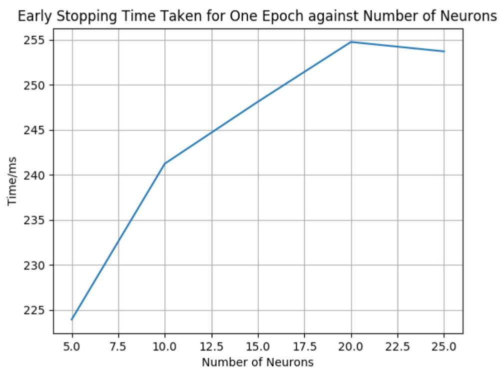
Optimal Hyper-Parameter	Value	Rationale
Batch Size	4	Refer to Section 3.1
Number of hidden neurons	20	Refer to Section 3.2
Decay Parameter	1×10^{-6}	Refer to Section 3.3
Number of Hidden Layers	2	Refer to Section 3.4

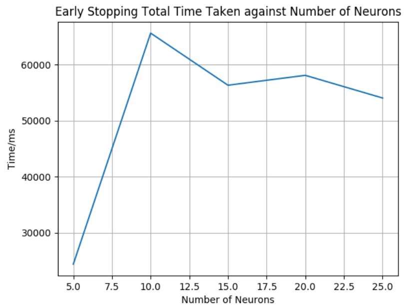
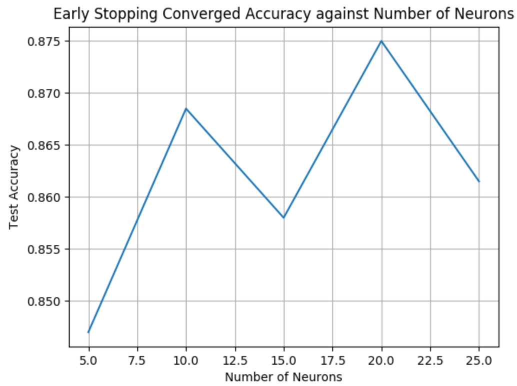
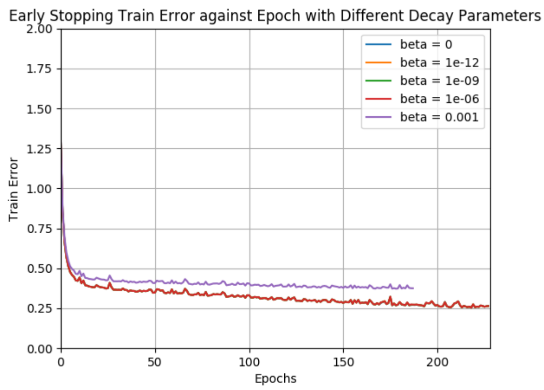
The early stopping significantly improves the network training time. While it is inevitable that with less training data, and also the fact that there will be multiple local minima, and the converged test accuracy for early stopping models will be lower than that for normal models. In our experiments, we prefer the use of early stopping in order to reduce the likelihood of overfitting and improve the generalization of the models.

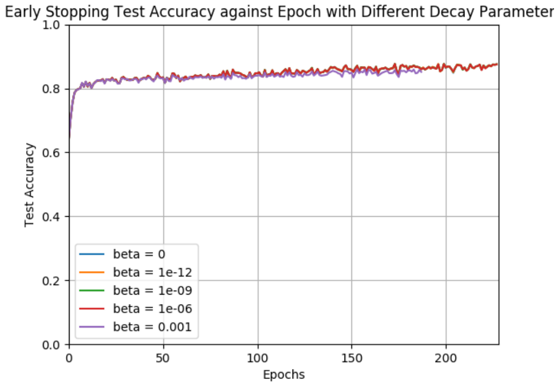
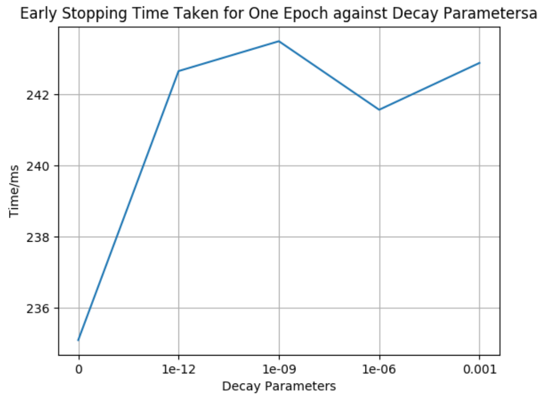
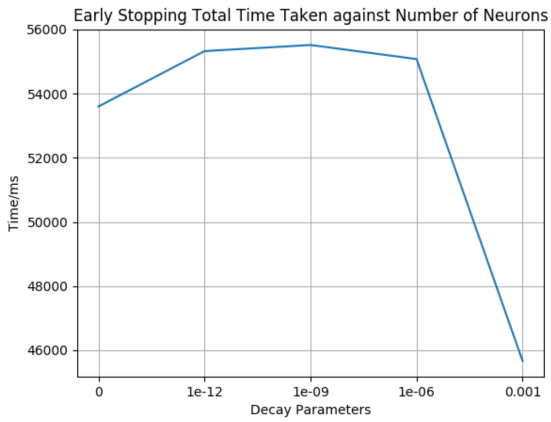
The requested plots for Part A with and without early stopping (see Appendix Part A for larger figures):

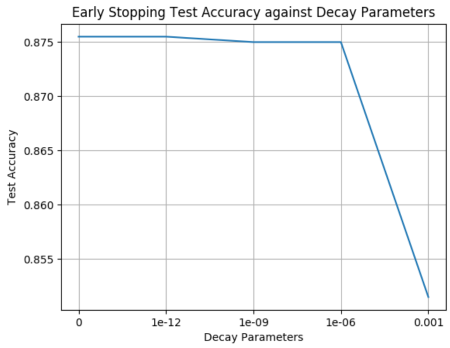
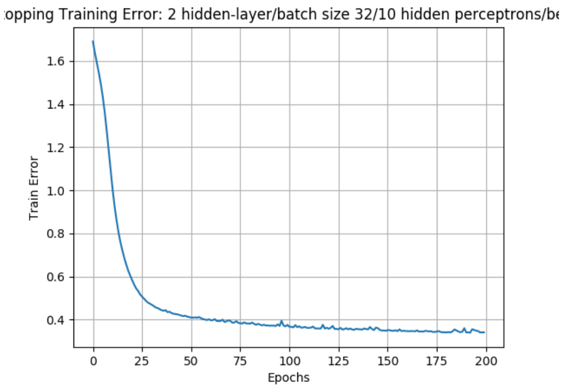
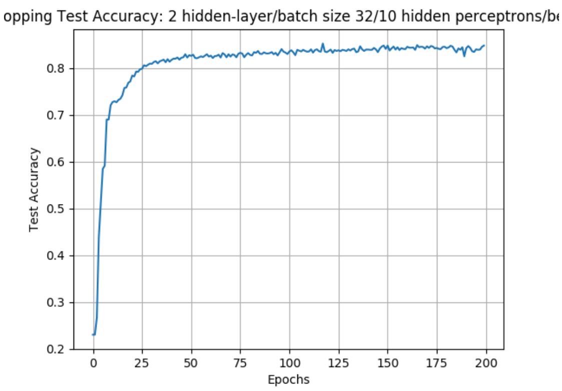
	With early stopping
2(a) the training error against the number of epochs	
	(refer to appendix FigA.Q2a.1 for larger figure)
2(a) the test accuracy against the number of epochs	

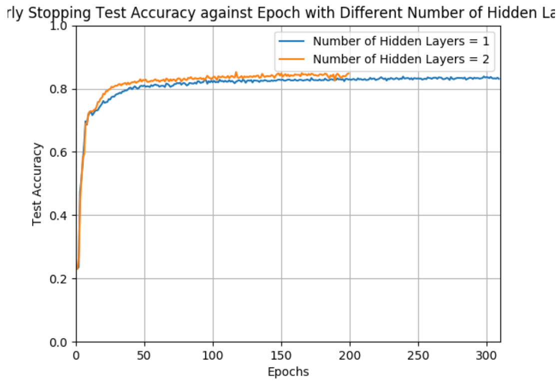
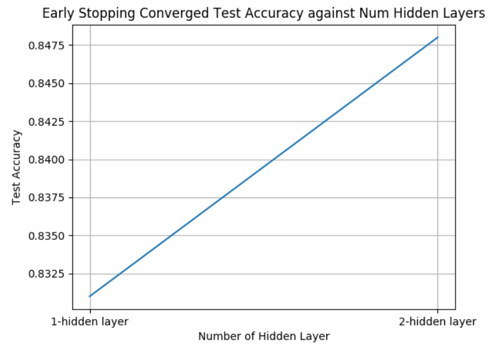
<table>
<tr><td></td><td>(refer to appendix FigA.Q2a.2 for larger figure)</td></tr>
<tr><td rowspan="2">2(b) the time taken to train the network for one epoch against different batch sizes</td><td>
</td></tr>
<tr><td>(refer to appendix FigA.Q2b.1 for larger figure)</td></tr>
<tr><td rowspan="2">2(c) the total time taken to train the network against different batch sizes</td><td>
</td></tr>
<tr><td>(refer to appendix FigA.Q2c.1 for larger figure)</td></tr>
<tr><td rowspan="2">2(c) the converged test accuracy against different batch sizes</td><td>
</td></tr>
<tr><td>(refer to appendix FigA.Q2c.2 for larger figure)</td></tr>
</table>

<table>
<tr><td rowspan="2">3(a) the training error against the number of epochs</td><td>
</td></tr>
<tr><td>(refer to appendix FigA.Q3a.1 for larger figure)</td></tr>
<tr><td rowspan="2">3(a) the test accuracy against the number of epochs</td><td>
</td></tr>
<tr><td>(refer to appendix FigA.Q3a.2 for larger figure)</td></tr>
<tr><td rowspan="2">3(b) the time taken to train the network for one epoch against the number of epochs</td><td>
</td></tr>
<tr><td>(refer to appendix FigA.Q3b.1 for larger figure)</td></tr>
</table>

<table>
<tr>
<td rowspan="2">3(c) the total time taken to train the network against the number of epochs</td>
<td>
</td>
</tr>
<tr>
<td>(refer to appendix FigA.Q3c.1 for larger figure)</td>
</tr>
<tr>
<td rowspan="2">3(c) the converged test accuracy against different number of epochs</td>
<td>
</td>
</tr>
<tr>
<td>(refer to appendix FigA.Q3c.2 for larger figure)</td>
</tr>
<tr>
<td rowspan="2">4(a) the training error against the number of epochs</td>
<td>
</td>
</tr>
<tr>
<td>(refer to appendix FigA.Q4a.1 for larger figure)</td>
</tr>
</table>

4(b) the test accuracy against the number of epochs	
	(refer to appendix FigA.Q4b.1 for larger figure)
4(b) the time taken to train the network for one epoch against the number of epochs	
	(refer to appendix FigA.Q4b.2 for larger figure)
4(b) the total time taken to train the network against the number of epochs	
	(refer to appendix FigA.Q4b.3 for larger figure)

<table>
<tr><td rowspan="2">4(b) the converged test accuracy against different values of decay parameter</td><td>
</td></tr>
<tr><td>(refer to appendix FigA.Q4b.4 for larger figure)</td></tr>
<tr><td rowspan="2">5(a) the train error of 4-layer network</td><td>
</td></tr>
<tr><td>(refer to appendix FigA.Q5a.1 for larger figure)</td></tr>
<tr><td rowspan="2">5(b) the test accuracy of 4-layer network</td><td>
</td></tr>
<tr><td>(refer to appendix FigA.Q5a.2 for larger figure)</td></tr>
</table>

5(b) Comparison on the performances on the 3-layer and 4-layer networks.	 (refer to appendix FigA.Q5b.1 for larger figure)
5(b) the converged test accuracy against different values of decay parameter	 (refer to appendix FigA.Q5b.2 for larger figure)

Part B - Regression Problem

1. Introduction

The project aims at predicting the median housing prices from the 8 input attributes (e.g. median income, housing median age etc). We will be developing a regression model to predict the median housing price.

2. Method

2.1 Data Pre-processing: Train Test Split & Normalization

Initially, we randomly split the data into validation and test set in ratio of 7:3.
During the 5-fold cross-validation training (see section 2.2.6 for more detailed discussion), 4/5 of the validation data will then be used to train the model while the rest for validation at each fold. Thus, at each fold, the data's inputs and output by the following formula:

$$\tilde{x}_i = \frac{x_i - \mu_i}{\sigma_i}$$

where, μ_i is the mean, and σ_i is the standard deviation of each feature, and they were only calculated over the 4/5 training data at each fold, as the model would not know the other 1/5 validation data in advance.

At testing stage, the test data will be scaled using mean and standard deviation of the validation data, which was used to train the final model after selecting the optimal hyper-parameter.

This scaling step was introduced to improve the model performance and convergence.

2.2 Model Development

For this assignment, we applied mini-batch gradient descent, 3-way data split and 5-fold cross-validation to train the models, as well as to determine the optimal hyper-parameters. We had conducted exhaustive controlled experiments, where each time only one hyper-parameter is changed to determine the optimal value of the hyper-parameter. The hyper-parameter we had experimented for 3-layer feedforward neural network are

- learning rate,
- number of hidden neurons

After determining the optimal hyper-parameter, we than combine the train and validation data to train the final model and plot the mean square error for test data against number of epochs.

Moreover, for Q4 of part B, we had also experimented with the model with different number of layers with and without dropouts.

2.2.1 Architecture

For Q1 to Q3 of part B, we developed a 3-layer feedforward neural network, with the following architecture:

- an input layer of dimension 8 (corresponding to the input feature dimensions),
- a hidden discrete perceptron layer of 30 perceptrons with ReLu activation function, and
- an output layer with one linear neuron.

In this assignment, we had experimented with different numbers of perceptron n in the hidden layer, which is further discussed in Section 3.2.

For Q4 of part B, we developed another 2 feedforward neural networks, with 4 and 5 layers respectively:

4-layer:

- an input layer of dimension 8 (corresponding to the input feature dimensions),
- a hidden discrete perceptron layer of *optimal_number* perceptrons with ReLu activation function (optimal number of neurons is discussed in Section 3.2),
- a hidden discrete perceptron layer of 20 perceptrons with ReLu activation function, and
- an output layer with one linear neuron.

5-layer:

- an input layer of dimension 8 (corresponding to the input feature dimensions),
- a hidden discrete perceptron layer of *optimal_number* perceptrons with ReLu activation function (optimal number of neurons is discussed in Section 3.2),
- a hidden discrete perceptron layers of 20 perceptrons with ReLu activation function,
- a hidden discrete perceptron layers of 20 perceptrons with ReLu activation function, and
- an output layer with one linear neuron.

To answer Q4 of Part B, we had also experimented the above-mentioned models with the introduction of dropouts, and the final results are further discussed in Section 3.3.

2.2.2 Learning Goal

In this assignment, the above 3 neural models mentioned in 2.2.1 aim to minimize L2-regularized loss while training.

The square-error cost is the cost function for neural network models learning regression tasks:

$$J = \frac{1}{2}\sum_{k=1}^{K}(d_k - y_k)^2$$

In order to avoid overfitting and enhance generalising ability, we introduced:

(1) L2-regularization is introduced to penalise the learned weights, i.e. to the above square-error cost function.

$$J_1(\boldsymbol{W}, \boldsymbol{b}) = J(\boldsymbol{W}, \boldsymbol{b}) + \beta_2 \sum_{ij} \left(w_{ij}\right)^2$$

(2) Dropout:

Dropout is introduced to randomly drop neurons from the networks during training. This prevents neurons from co-adapting and thereby reduces overfitting, as we only train a fraction of weights in each iteration. At test time, the weights are always present and presented to the network with weights multiplied by keep_rate p. The output at the test time is same as the expected output at the training time. Applying dropouts result in a 'thinned network' that consists of only neurons that survived.

2.2.3 Weights Initialisation - Truncated Normal Distribution

Weights Initialisation used in Part B is the same as that in Part A Section 2.2.3.

2.2.4 Optimising Hyper Parameters

In order to optimize the hyper parameters, we have designed controlled experiments, by holding all other variables constant, while changing one of the following hyper parameters at a time:

(a) Learning rate - [10^{-10}, 10^{-9}, $0.5 * 10^{-8}$, 10^{-7}, $0.5 * 10^{-6}$]
(b) Number of Hidden Neurons - [20,40,60,80,100]

2.2.5 Selection Criteria

We have set the following selection criteria in determining the optimal hyper parameter:

(a) Cross-validation Error:
   (i) The model with the lower cross-validation error is generally better.

(b) Training Time per Epoch:
   (i) The model with shorter training time is less costly to train, and thus better.

(c) Model Robustness (ability to generalize):
In order to improve the model robustness and prevent overfitting[4], we would limit the capacity of the neural network, by controlling:

   (i) Model complexity:
   We will limit the number of hidden layers and number of units per layer to control the model complexity. The more complex model tends to overfit and reduces the model's ability to generalize. Moreover, it is costlier to train the more complex model. Hence, generally if no under-fitting signal is shown, the less complex model with fewer number of hidden layers or fewer number of units per layer is generally better.

   (ii) Dropout
   Dropout is a technique used to prevent overfitting (as explained in section 2.2.2), and thereby improving the model's robustness.

2.2.6 5-fold Cross-validation with 3-way Data Split:

5-fold cross-validation with 3-way data split is experimented in resulting in a less biased model predictions. The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Create a 5-fold partition of the validation data.
3. For each of 5 experiments:
   (1) use 4 folds for training and the remaining one-fold for validation.

---

[4] https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec9.pdf

(2) The 4 folds will be further split into mini-batches and used to train the network.

We will keep track of the validation error at the end of each i_{th} fold, so as to calculate cross-validation error using the following formula:

$$\text{Cross-validation error} = \frac{1}{K}\sum_{i=1}^{K} e_i$$

The optimal hyper-parameter will then be decided based on the criteria specified in section 2.2.5. Afterwards, all the data used in cross-validation training will be used to train the final model, whereby the mean-squared error of the test set will be computed on.

3. Experiments and Results

In this section, we present the experiment findings for different hyper-parameters. The default hyper-parameters, unless otherwise stated, are:

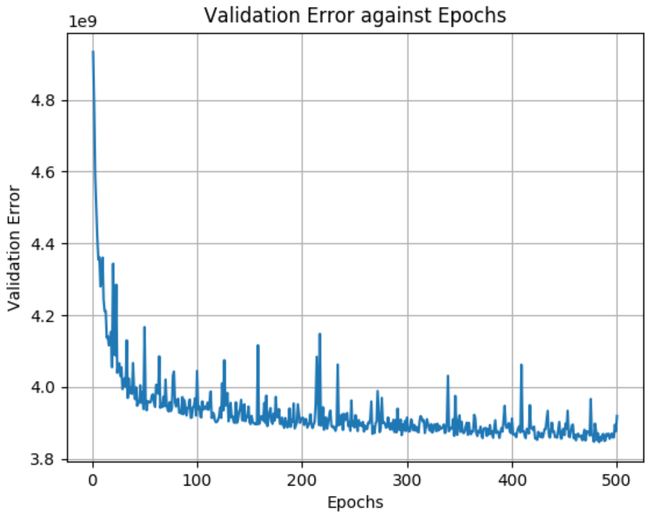
- Batch size = 32
- Number of Neurons in Hidden Layer = 30
- L2-regularized term = 1 x 10^{-3}
- Learning Rate = 1 x 10^{-7}
- Epochs = 500 for each fold

When assessing the optimal parameters, we will not look at cross-validation error, which is biased as it is calculated from the validation data which had also been used in training the model. Therefore, we will be focusing on test error to assess the performance of different hyper-parameters. Moreover, time and model complexity will be taken into consideration in assessing the performance of different hyper-parameters.

3.1 Validation Error, and Predicted Value Against True Value

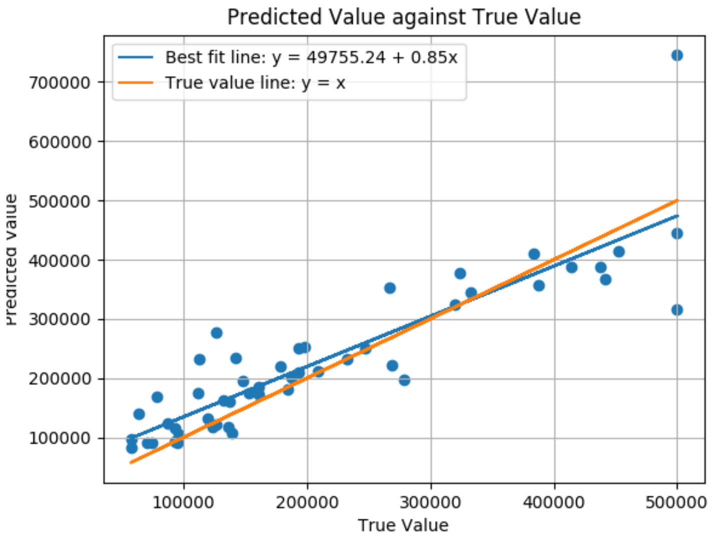
This section answers Q1 of Part B.

a) We used the validation set to train the model and plotted the validation error against epochs below.



Validation Error Against Epochs

b) We randomly sampled 50 data from the test set and computed the predicted value with the model trained in a). The result is presented below, with y-axis as the predicted value, and x-axis as the true value. The orange line is the true value line i.e. the errorless line, and the blue line is the best-fit line for the scatter plot of predicted value against true value.



3.2 The optimal learning rate = 10^{-7}

This section answers Q2 of Part B.

The experimental results can be summarised into the following table (for the plots required, they can be found in Section 4):

Learning rate	Cross-validation Error / 10^9	Time / Epoch (ms)
10^{-10}	30.830	102.264
10^{-9}	4.589	102.473
$0.5 * 10^{-8}$	4.247	99.825
10^{-7}	**3.947**	**99.897**
$0.5 * 10^{-6}$	4.367	100.578

Apply the 3 criteria for optimal hyper-parameter:

1. [CV Error] The cv error is the lowest for the model with learning rate 10^{-7}, which is significantly lower than the others. This indicates that 10^{-7} is the learning rate, among the candidates, that converges the model to the better local minima.
2. [Time per Epoch] The time taken per epoch for all the models are relatively close to each other. Thus, the training cost for different learning rates are invariant.
3. [Model Robustness] The model complexity will not be affected by the learning rate.

Hence, it is determined that the **optimal learning rate is 10^{-7}**.

The final model is then trained with learning rate 10^{-7}, and the plot of test error against number of epochs can be found Section 4.

3.3 Optimal number of hidden neurons = 100

This section answers Q3 of Part B.

The experimental results can be summarised into the following table (for the plots required, they can be found in Section 4):

Number of hidden neurons	Cross-validation Error / 10^9	Time per Epoch / ms
20	4.086	102.018
40	4.057	98.344
60	4.011	97.787
80	4.021	100.363
100	**3.935**	**106.269**

Apply the 3 criteria for optimal hyper-parameter:

1. [CV Error] The cv error for the model with 100 hidden neurons is the lowest, and it's significantly lower than the other candidates.
2. [Time per Epoch] The time taken per epoch for all the models are relatively the close to each other.
3. [Model Robustness] While we generally prefer models with less complexity, our ultimate interest still lies in the model performance. In this case, 100 neurons better fulfilled our fundamental concern than the rest as compared to other candidates.

Hence, it is determined that the **optimal number of hidden neurons is 100**.

The final model is then trained with number of hidden neurons 100, and the plot of test error against number of epochs can be found Section 4.

3.4 Comparison of models with different layers (with / without dropouts)

Q4 of part B specifies that the following models are to be trained on the validation data (without employing cross-validation as Q4 specifies to use validation data to train each model, and compare their test errors) with the following hyper-parameters:

- First hidden layer with optimal number of neurons found in section 3.2 = 100
- Number of hidden neurons in other hidden layers = 20
- Learning rate = 10^{-9}
- Drop out keep probability = 0.9

In the implementation of the code, we had turned off drop out by setting keep_prob to 1.0.

Model	Dropout	Test Error / 10^9	Time per Epoch / ms
3-layer	False	4.524	137.844
	True	4.527	171.212
4-layer	False	3.021	162.190

	True	3.602	206.296
5-layer	**False**	**2.649**	**183.605**
	True	3.053	239.463

Number of Layers

It is observed that the test error generally decreases when number of layers increases. In this case, the reported test error on the 5-layer without dropout is the lowest. This shows that a 5-layer model is better able to handle the California Housing Price dataset than fewer-layer models. It is important to note that even though we often value simplicity of the model, it shall not come at the expense of performance.

Dropout

For each number of layer, we can see that the models without dropout reported lower test error than the model with dropout. While dropout is generally introduced to improve the robustness of the model, it may not always workout. For this particular dataset, California Housing Price, drop out with keep probability of 0.9 had backfired and hurt the model performance.

It should also be noted that the training time per epoch significantly increased when dropout employed than when it is not employed. Thus, the usage of dropout will increase the training cost.

Possible Reason for the Observations

Since making the model architecture more complex (by adding more layers) is improving the test error, and the regularization technique (dropout) is hurting the model performance on test data, it is highly likely that the capacity of current model architecture is still low, such that the model has yet to overfit the training data. When the model capacity is already low, further lowering it with dropout will hurt the performance.

Thus, any regularization technique introduced will hurt the performance.

4. Conclusion

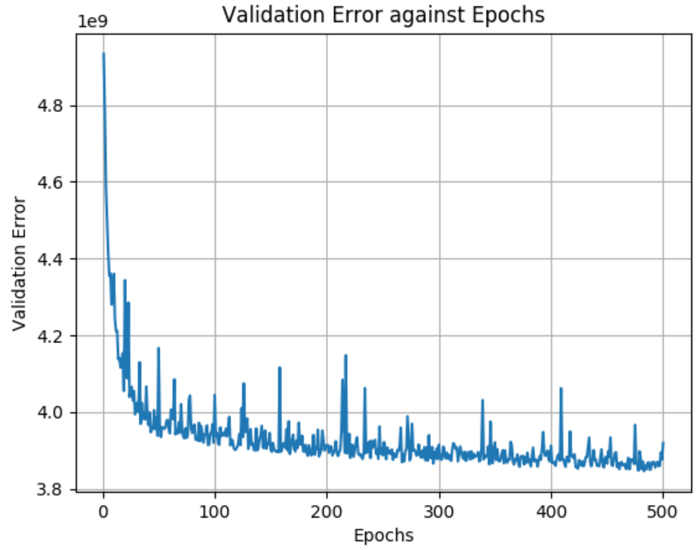
In conclusion, the optimal hyper parameters should be:

Optimal Hyper-Parameter	Value	Rationale
Learning rate	10^{-7}	Refer to section 3.2
Number of hidden neurons	100	Refer to section 3.3

Dropout in general can improve the generalization of the model, as stated in Section 2.2.2. However, from the experiment, it is observed that the model with dropout has a higher test error than the model without dropout. Hence, dropout is not preferred in this case.

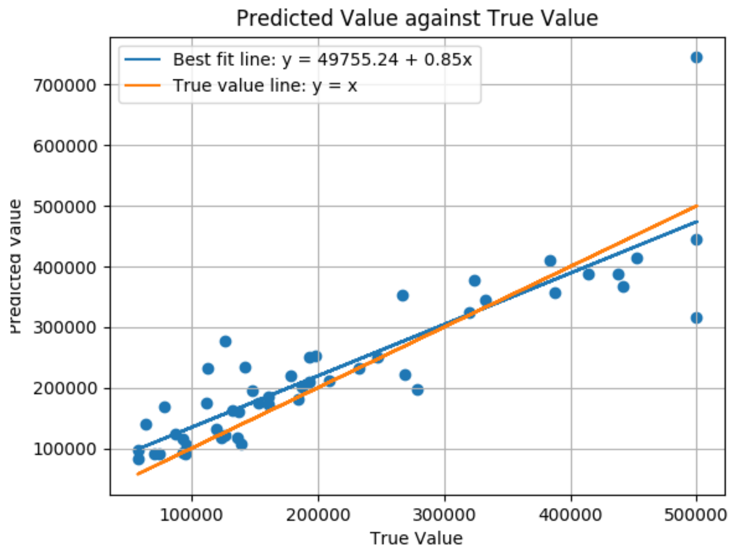
The requested plots for Part B (refer to Appendix Part B for larger figures):

1(a) Validation errors against epochs

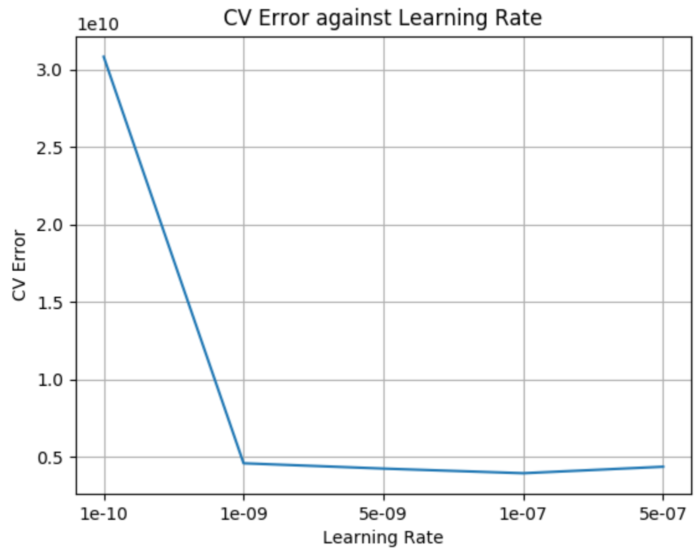
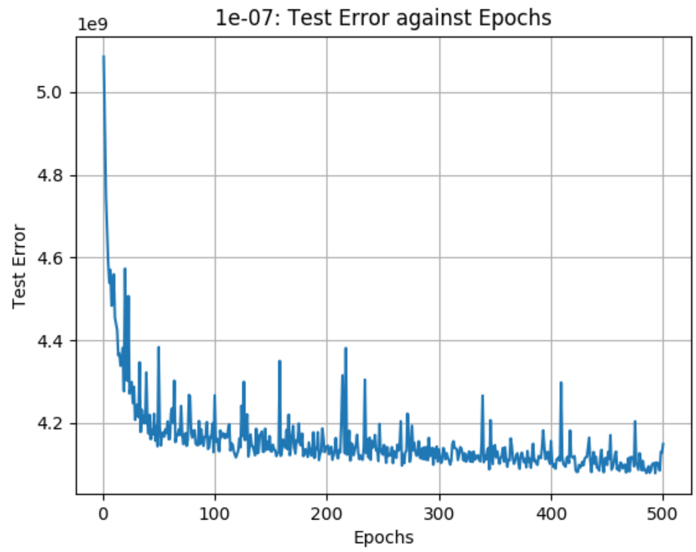


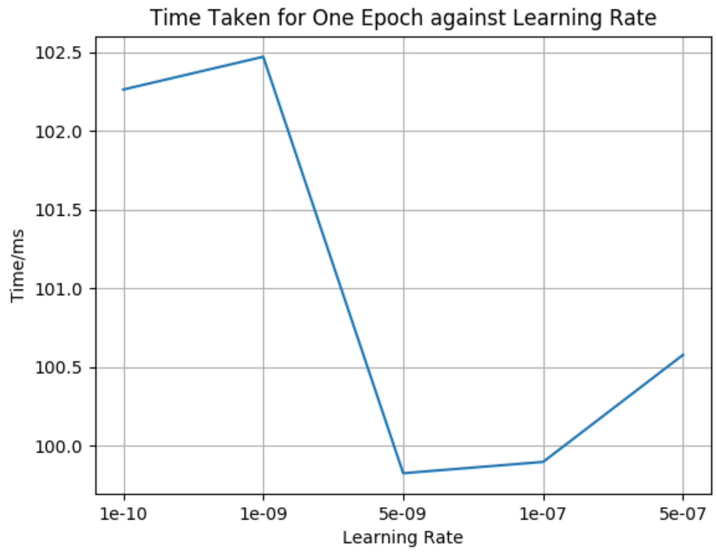
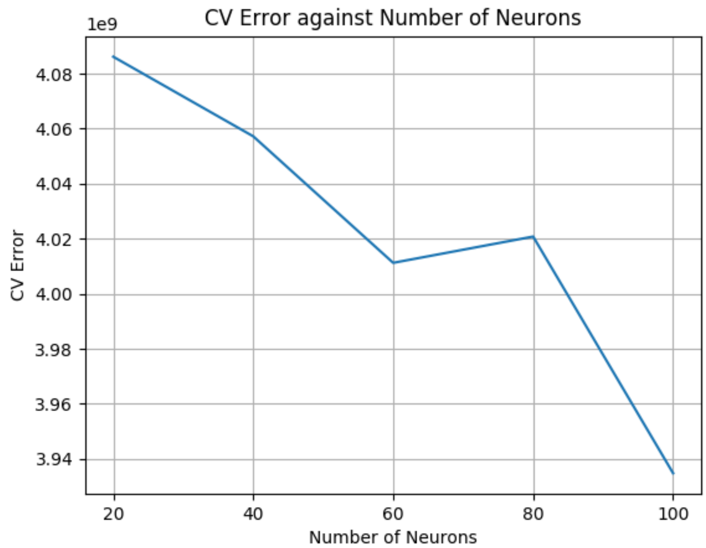
(refer to appendix FigB.Q1a for larger figure)

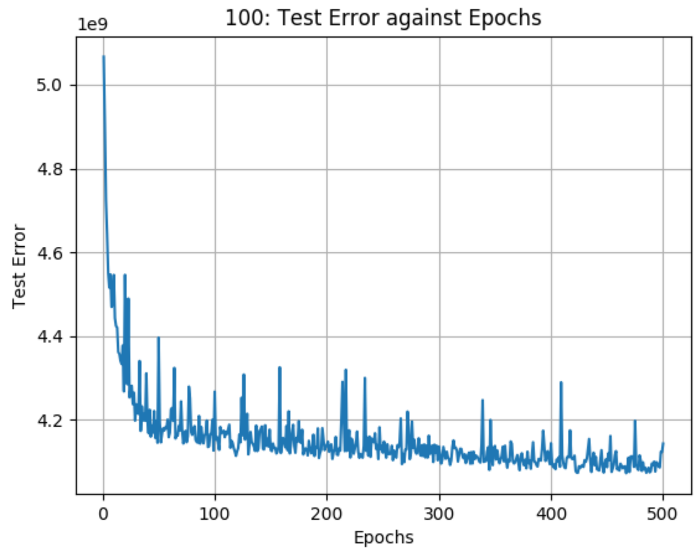
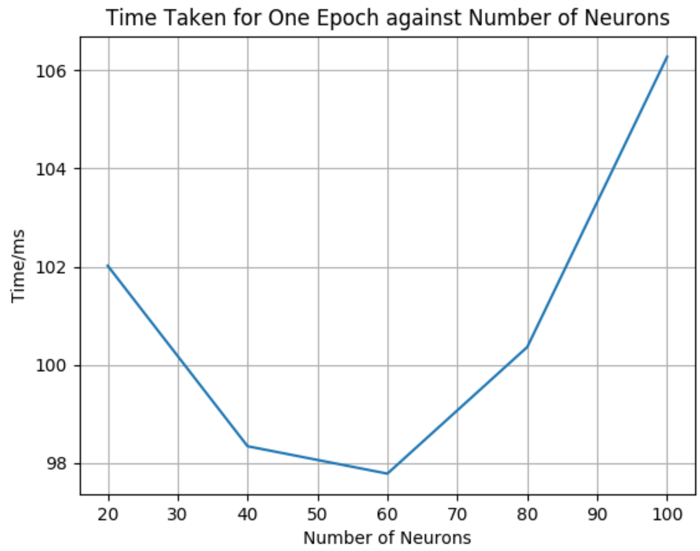
1(b) Predicted values against target values for any 50 test samples

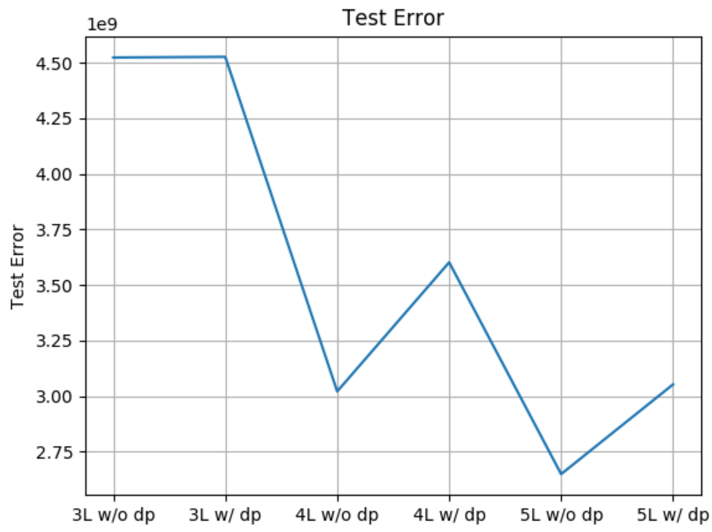
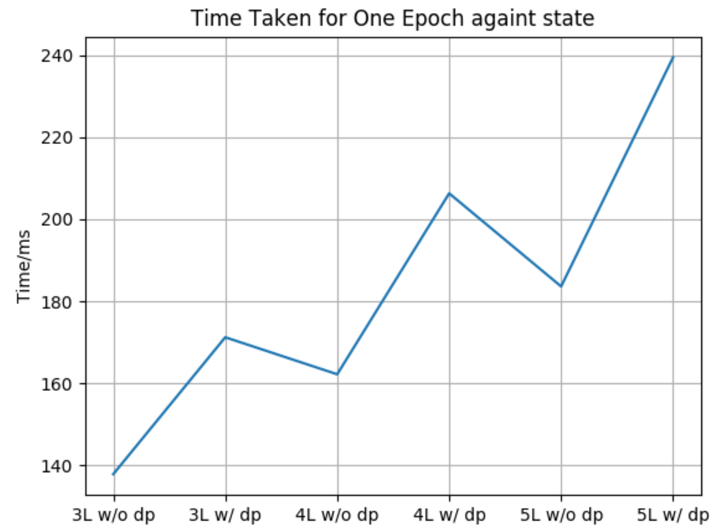


(refer to appendix FigB.Q1b for larger figure)

2(a) Cross-validation errors against learning rate	 (refer to appendix FigB.Q2a for larger figure)
2(b) Test error against training epoch	 (refer to appendix FigB.Q2b.1 for larger figure)

2(b) Time taken per epoch against learning rate	 (refer to appendix FigB.Q2b.2 for larger figure)
3(a) Cross-validation errors against the number of hidden-layer neurons	 (refer to appendix FigB.Q3a for larger figure)

3(b) Test errors against number of epochs	 (refer to appendix FigB.Q3b for larger figure)
3(c) Time taken for one epoch against number of neurons	 (refer to appendix FigB.Q3c for larger figure)

<table>
<tr><td rowspan="2">4 Test error against the different models with different number of neurons, and with / without dropout</td><td>
</td></tr>
<tr><td>(refer to appendix FigB.Q4 for larger figure)</td></tr>
<tr><td rowspan="2">4 Time per epoch for different models</td><td>
</td></tr>
<tr><td>(refer to appendix FigB.Q4.1 for larger figure)</td></tr>
</table>

Appendix

Classification report with precision/recall/f1 score

Classification report for different batch sizes with Early Stopping

```
# ------------------------------------------------
#     Early Stopping Batch size 4 Test set classification report:
#              precision    recall  f1-score   support
#
#           1      0.970     0.987     0.978       461
#           2      0.956     0.969     0.962       224
#           3      0.912     0.859     0.885       397
#           4      0.542     0.768     0.635       211
#           5      0.871     0.857     0.864       237
#           7      0.902     0.764     0.827       470
#
# avg / total      0.884     0.869     0.873      2000
#
# Early Stopping Batch size 8 Test set classification report:
#              precision    recall  f1-score   support
#
#           1      0.976     0.974     0.975       461
#           2      0.952     0.964     0.958       224
#           3      0.823     0.957     0.885       397
#           4      0.598     0.318     0.415       211
#           5      0.847     0.865     0.856       237
#           7      0.795     0.840     0.817       470
#
# avg / total      0.845     0.856     0.845      2000
#
# Early Stopping Batch size 16 Test set classification report:
#              precision    recall  f1-score   support
#
#           1      0.962     0.985     0.973       461
#           2      0.942     0.942     0.942       224
#           3      0.891     0.902     0.896       397
#           4      0.461     0.308     0.369       211
#           5      0.841     0.734     0.784       237
#           7      0.740     0.872     0.801       470
#
# avg / total      0.826     0.836     0.828      2000
#
# Early Stopping Batch size 32 Test set classification report:
#              precision    recall  f1-score   support
#
#           1      0.964     0.985     0.974       461
#           2      0.941     0.929     0.935       224
#           3      0.866     0.924     0.894       397
#           4      0.463     0.265     0.337       211
#           5      0.806     0.717     0.759       237
#           7      0.737     0.866     0.796       470
#
# avg / total      0.817     0.831     0.819      2000
#
# Early Stopping Batch size 64 Test set classification report:
#              precision    recall  f1-score   support
#
#           1      0.954     0.987     0.970       461
#           2      0.941     0.929     0.935       224
#           3      0.870     0.942     0.904       397
#           4      0.469     0.251     0.327       211
#           5      0.790     0.667     0.723       237
#           7      0.728     0.866     0.791       470
#
# avg / total      0.812     0.828     0.814      2000
```

Classification report for different number of neurons with Early Stopping

```
# -----------------------------------------------------
# Early Stopping Number of Neurons 5 Test set classification report:
#               precision    recall  f1-score   support

#           1      0.952     0.991     0.971       461
#           2      0.929     0.938     0.933       224
#           3      0.910     0.869     0.889       397
#           4      0.505     0.507     0.506       211
#           5      0.830     0.802     0.815       237
#           7      0.812     0.819     0.816       470

# avg / total      0.847     0.847     0.847      2000


# Early Stopping Number of Neurons 10 Test set classification report:
#               precision    recall  f1-score   support

#           1      0.970     0.987     0.978       461
#           2      0.956     0.969     0.962       224
#           3      0.912     0.859     0.885       397
#           4      0.542     0.768     0.635       211
#           5      0.871     0.857     0.864       237
#           7      0.902     0.764     0.827       470

# avg / total      0.884     0.869     0.873      2000

# Early Stopping Number of Neurons 15 Test set classification report:
#               precision    recall  f1-score   support

#           1      0.966     0.989     0.977       461
#           2      0.963     0.929     0.945       224
#           3      0.894     0.909     0.901       397
#           4      0.598     0.360     0.450       211
#           5      0.833     0.861     0.846       237
#           7      0.767     0.874     0.817       470

# avg / total      0.850     0.858     0.850      2000

# Early Stopping Number of Neurons 20 Test set classification report:
#               precision    recall  f1-score   support

#           1      0.985     0.983     0.984       461
#           2      0.947     0.964     0.956       224
#           3      0.869     0.937     0.902       397
#           4      0.626     0.706     0.664       211
#           5      0.871     0.823     0.846       237
#           7      0.865     0.777     0.818       470

# avg / total      0.878     0.875     0.875      2000

# Early Stopping Number of Neurons 25 Test set classification report:
#               precision    recall  f1-score   support

#           1      0.976     0.983     0.979       461
#           2      0.960     0.973     0.967       224
#           3      0.860     0.947     0.902       397
#           4      0.674     0.275     0.391       211
#           5      0.918     0.802     0.856       237
#           7      0.739     0.911     0.816       470

# avg / total      0.857     0.862     0.847      2000
```

Classification report for different L2-regularized term with early stopping

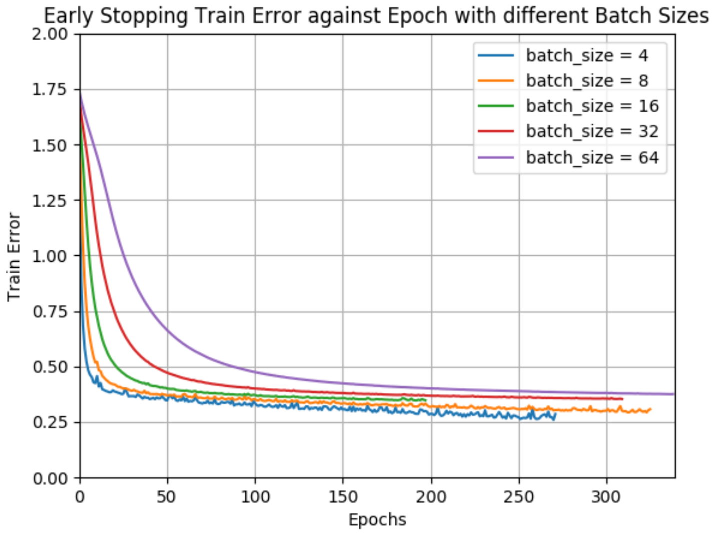
```
#-----------------------------------------------
# Early Stopping beta 0 Test set classification report:
#              precision    recall  f1-score   support
#
#           1      0.983     0.983     0.983       461
#           2      0.952     0.964     0.958       224
#           3      0.871     0.937     0.903       397
#           4      0.626     0.706     0.664       211
#           5      0.871     0.823     0.846       237
#           7      0.865     0.779     0.820       470
#
# avg / total      0.879     0.875     0.876      2000
#
# Early Stopping beta 1e-12 Test set classification report:
#              precision    recall  f1-score   support
#
#           1      0.985     0.983     0.984       461
#           2      0.952     0.964     0.958       224
#           3      0.871     0.937     0.903       397
#           4      0.623     0.706     0.662       211
#           5      0.871     0.827     0.848       237
#           7      0.865     0.777     0.818       470
#
# avg / total      0.879     0.875     0.876      2000
#
# Early Stopping beta 1e-09 Test set classification report:
#              precision    recall  f1-score   support
#
#           1      0.985     0.983     0.984       461
#           2      0.947     0.964     0.956       224
#           3      0.869     0.937     0.902       397
#           4      0.626     0.706     0.664       211
#           5      0.871     0.823     0.846       237
#           7      0.865     0.777     0.818       470
#
# avg / total      0.878     0.875     0.875      2000
#
# Early Stopping beta 1e-06 Test set classification report:
#              precision    recall  f1-score   support
#
#           1      0.985     0.983     0.984       461
#           2      0.947     0.964     0.956       224
#           3      0.869     0.937     0.902       397
#           4      0.626     0.706     0.664       211
#           5      0.871     0.823     0.846       237
#           7      0.865     0.777     0.818       470
#
# avg / total      0.878     0.875     0.875      2000
#
# Early Stopping beta 0.001 Test set classification report:
#              precision    recall  f1-score   support
#
#           1      0.962     0.991     0.976       461
#           2      0.963     0.938     0.950       224
#           3      0.896     0.894     0.895       397
#           4      0.543     0.512     0.527       211
#           5      0.856     0.751     0.800       237
#           7      0.784     0.840     0.811       470
#
# avg / total      0.850     0.852     0.850      2000
```

Classification report for different number of layers with early stopping

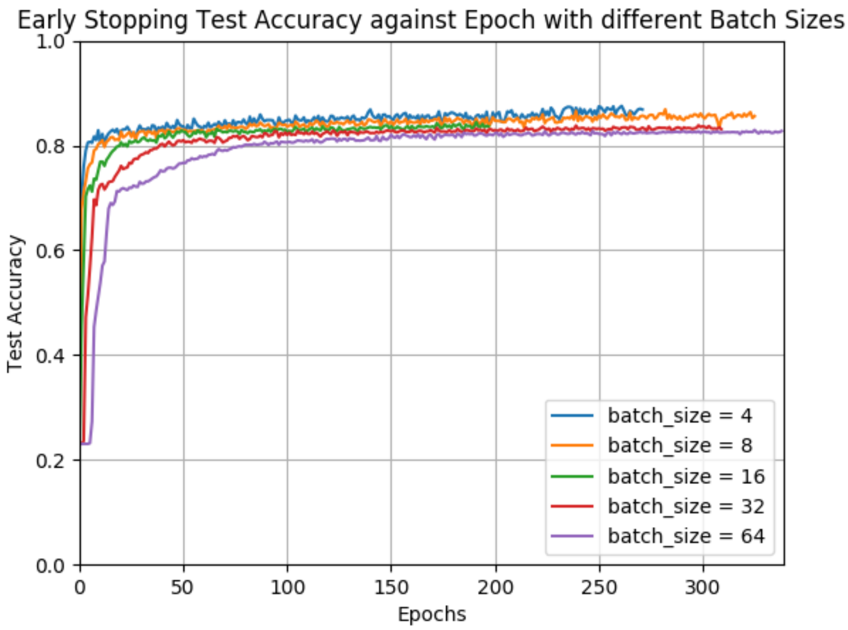
```
# Early Stopping 3-layer Test set classification report:
#              precision    recall  f1-score   support

#           1      0.964     0.985     0.974       461
#           2      0.941     0.929     0.935       224
#           3      0.866     0.924     0.894       397
#           4      0.463     0.265     0.337       211
#           5      0.806     0.717     0.759       237
#           7      0.737     0.866     0.796       470

# avg / total      0.817     0.831     0.819      2000
```

```
# Early Stopping 4-layer Test set classification report:
#              precision    recall  f1-score   support

#           1      0.954     0.987     0.970       461
#           2      0.950     0.938     0.944       224
#           3      0.854     0.942     0.896       397
#           4      0.542     0.398     0.459       211
#           5      0.847     0.768     0.805       237
#           7      0.791     0.832     0.811       470

# avg / total      0.839     0.848     0.842      2000
```

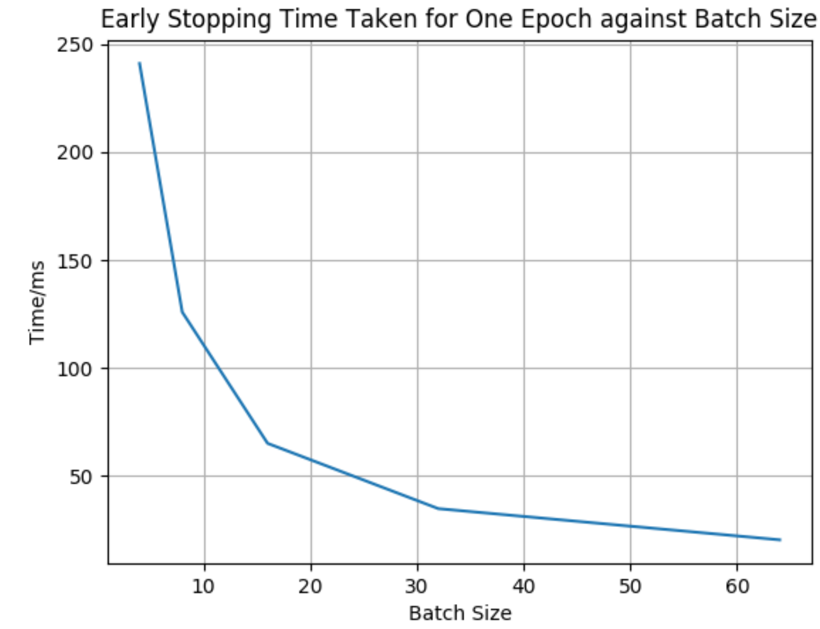
Part A Conclusion Figures



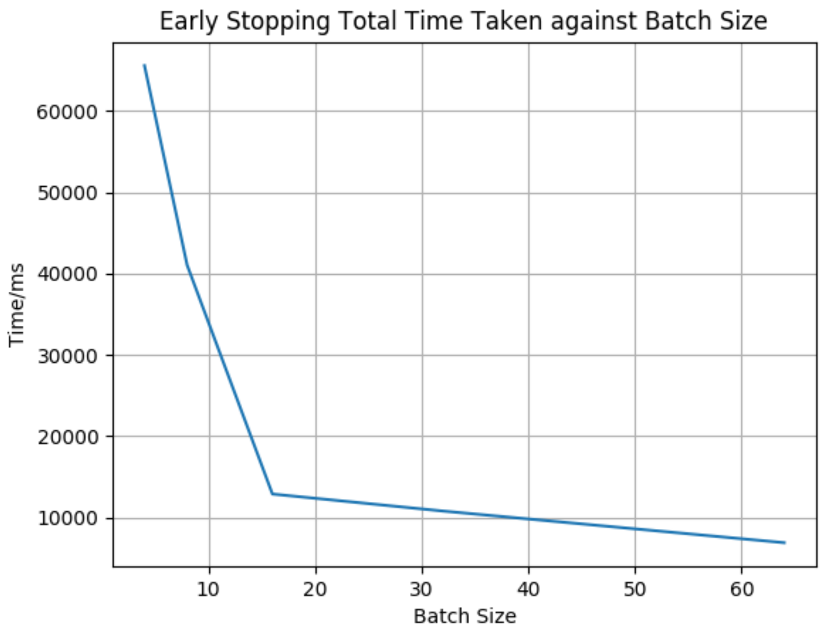
FigA.Q2a.1



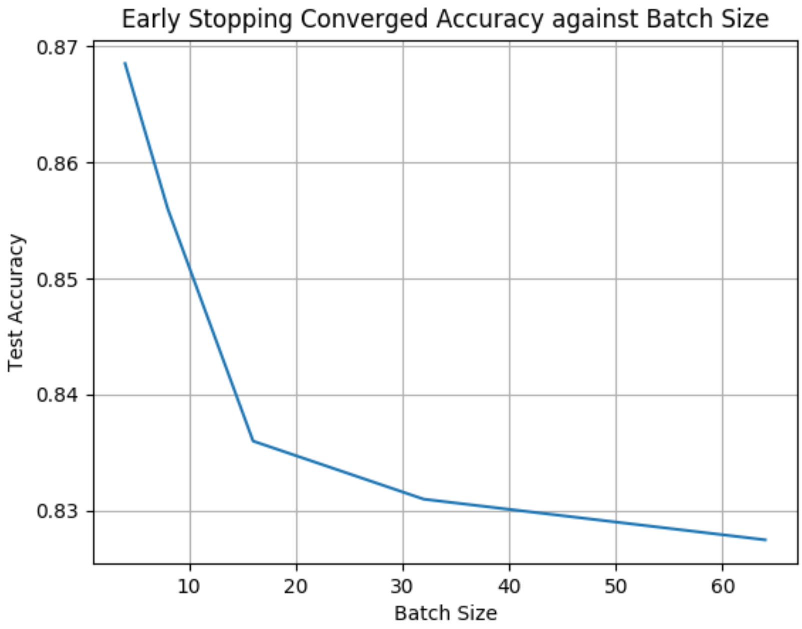
FigA.Q2a.2



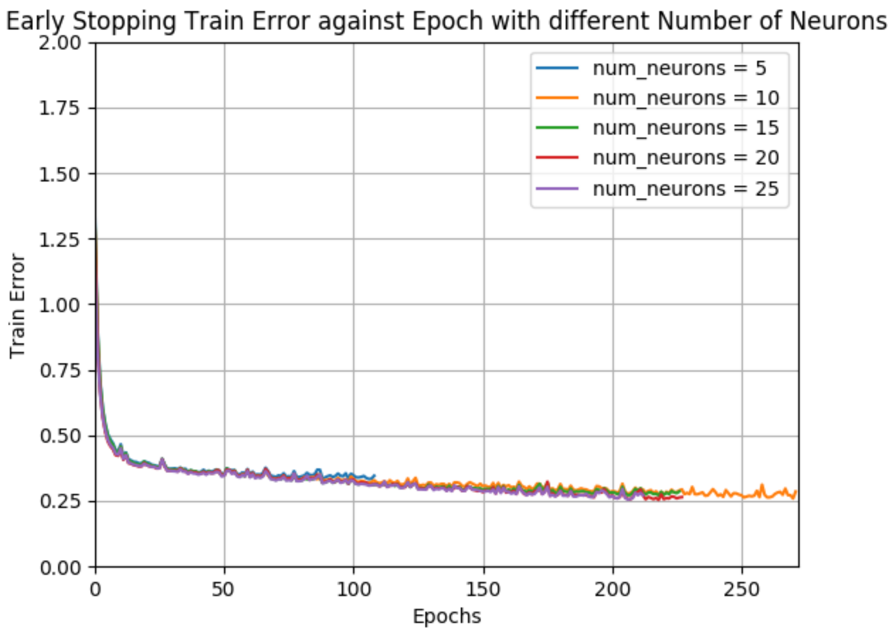
FigA.Q2b.1



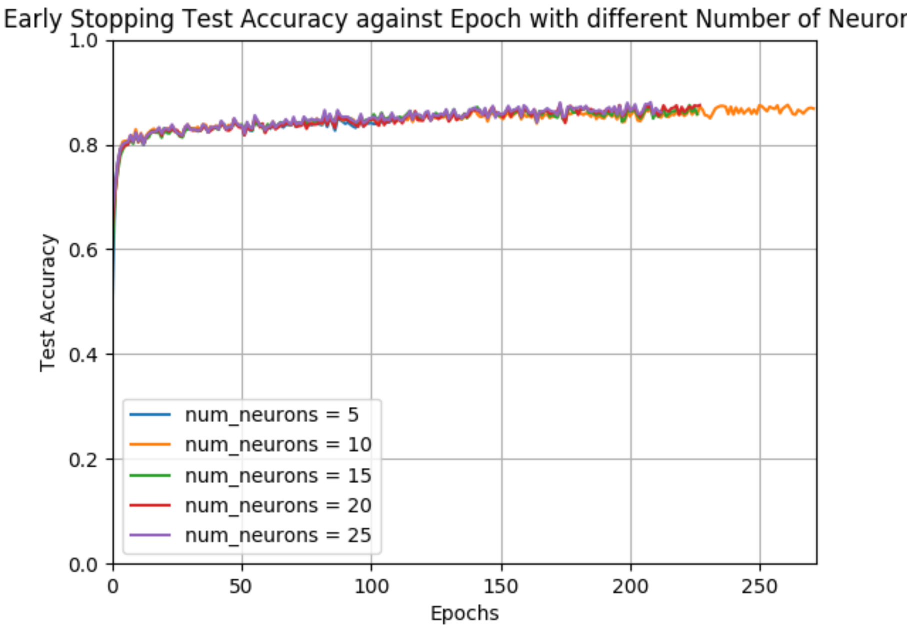
FigA.Q2c.1



FigA.Q2c.2

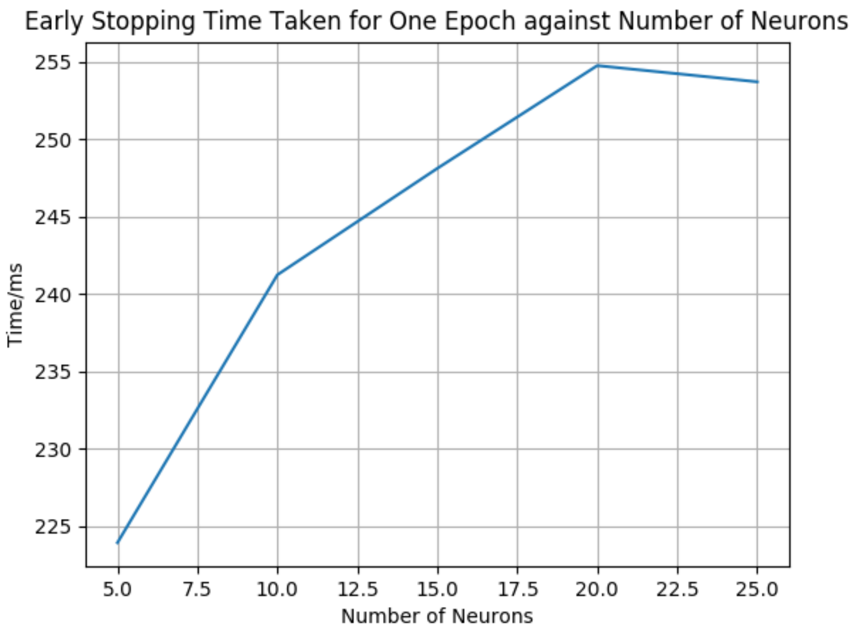


FigA.Q3a.1



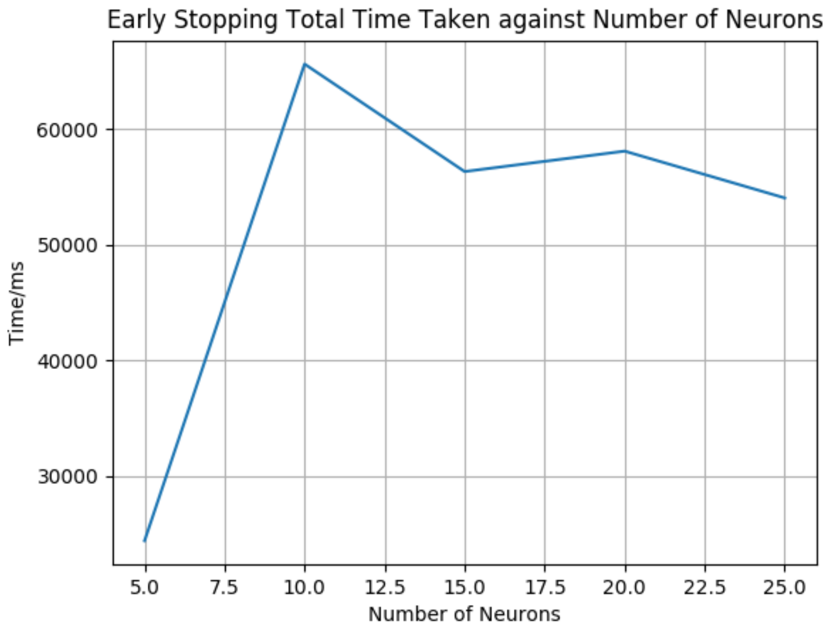
Early Stopping Test Accuracy against Epoch with different Number of Neuror
Test Accuracy
Epochs
num_neurons = 5
num_neurons = 10
num_neurons = 15
num_neurons = 20
num_neurons = 25
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200
250

FigA.Q3a.2

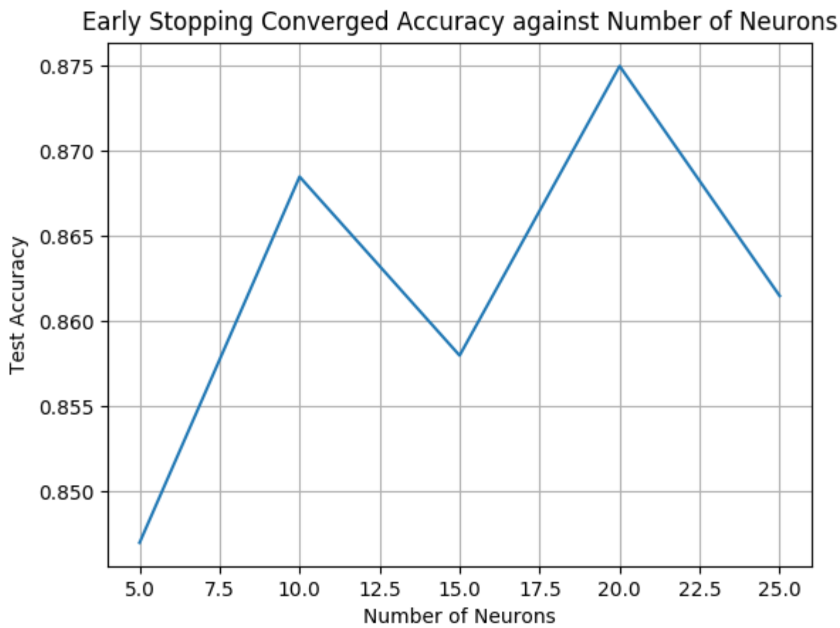


Early Stopping Time Taken for One Epoch against Number of Neurons
Time/ms
Number of Neurons
225
230
235
240
245
250
255
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0

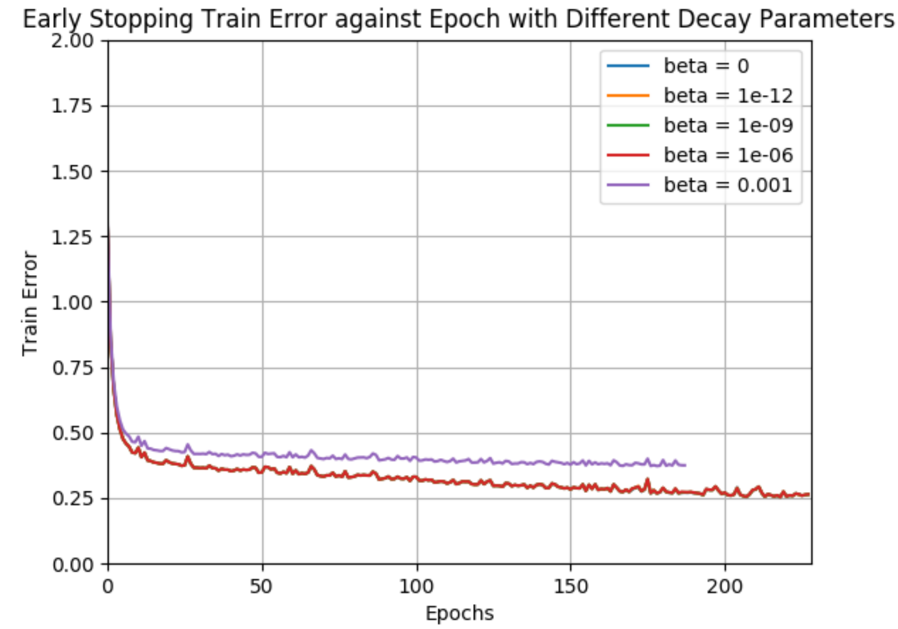
FigA.Q3b.1



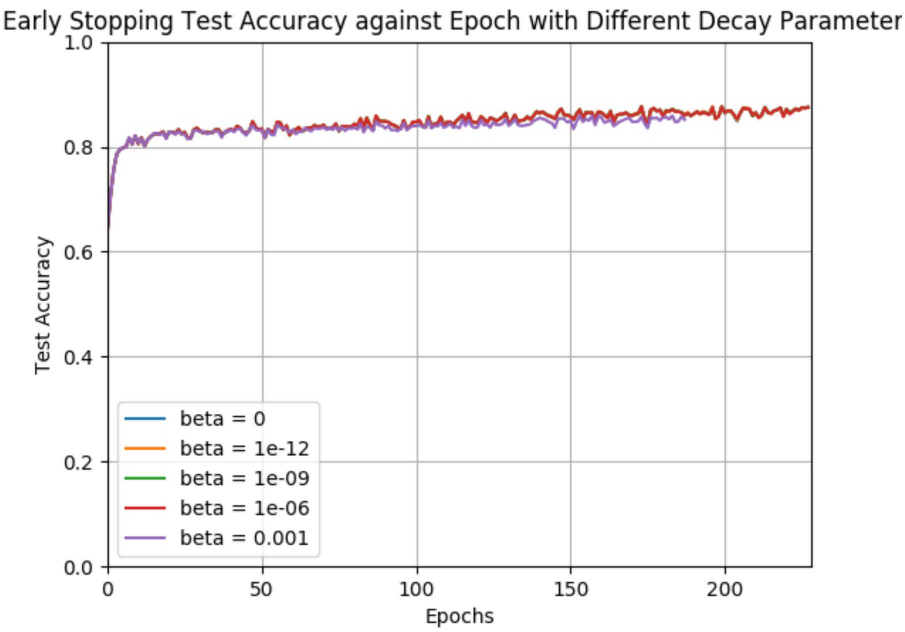
FigA.Q3c.1



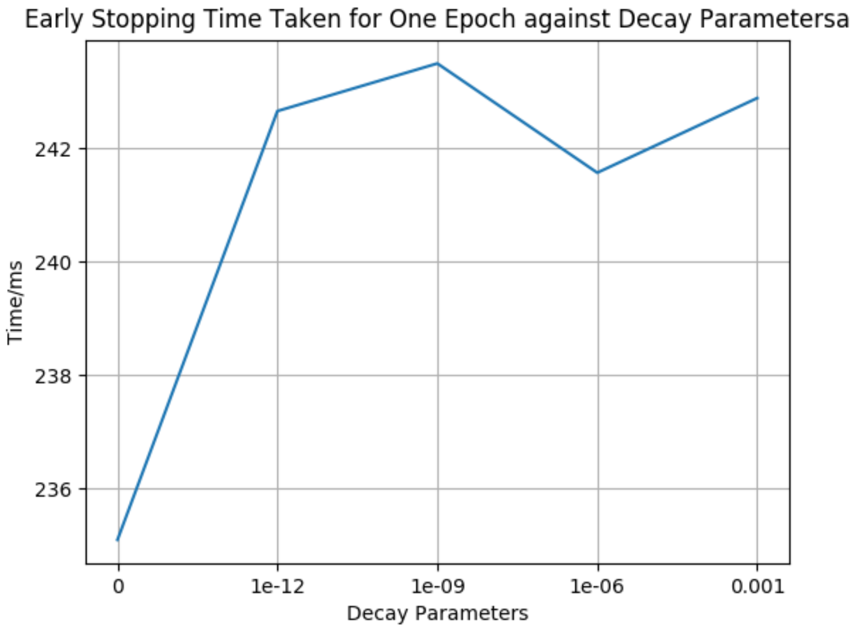
FigA.Q3c.2



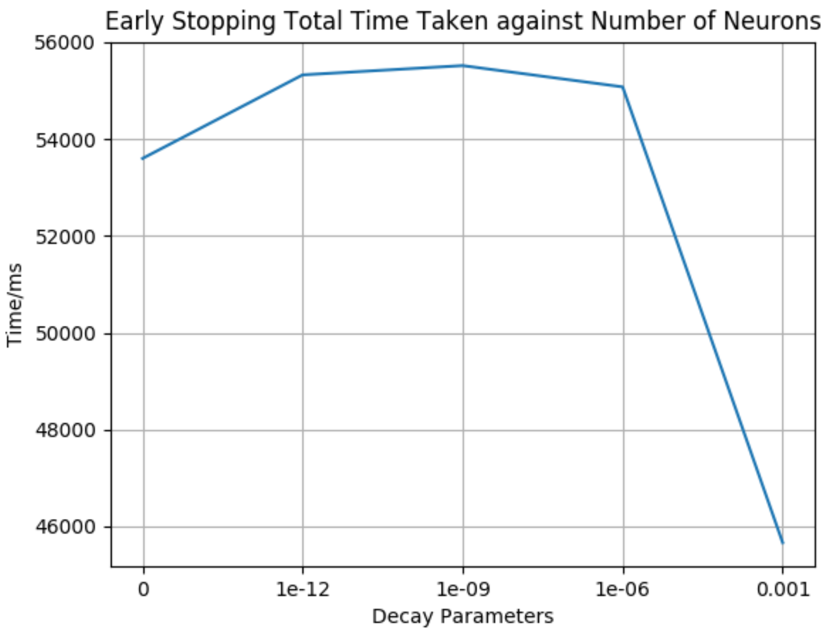
FigA.Q4a.1



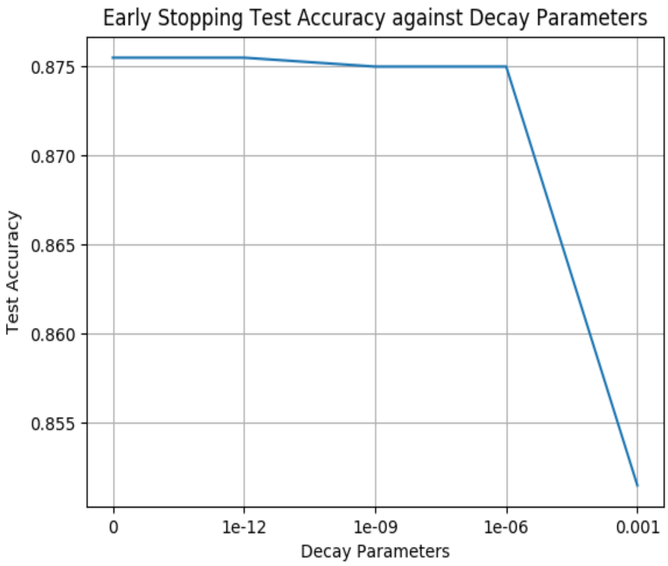
FigA.Q4b.1



FigA.Q4b.2

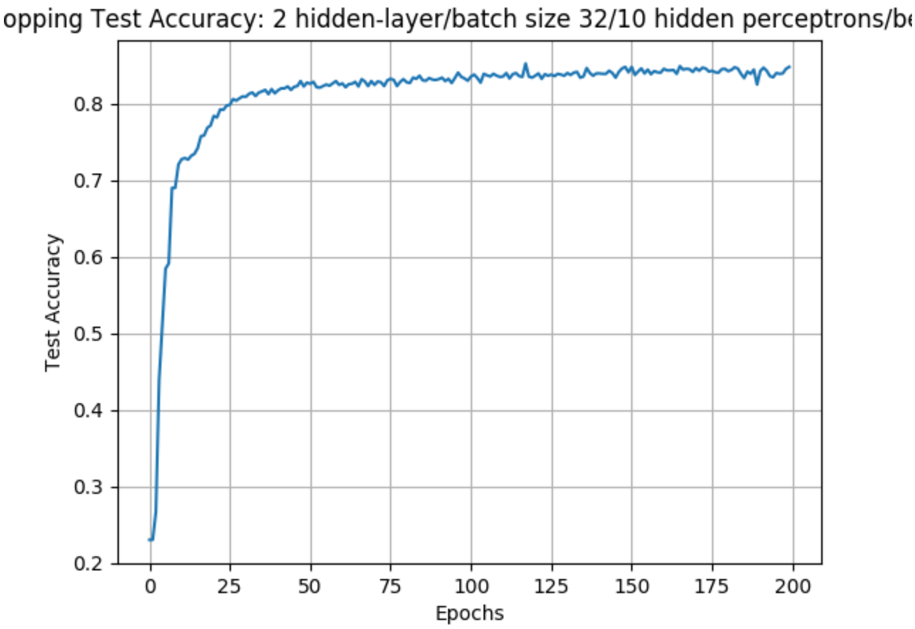


FigA.Q4b.3

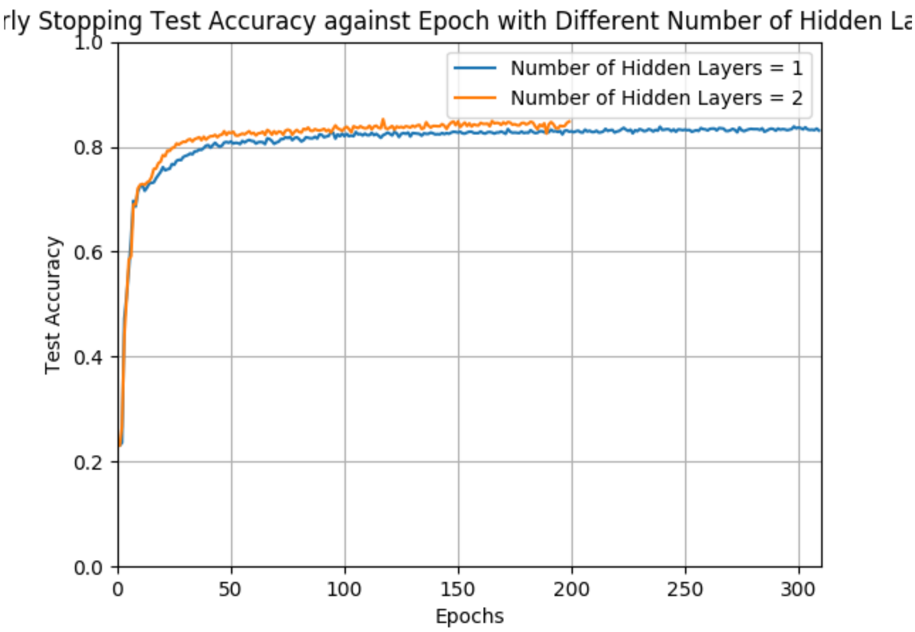


FigA.Q4b.4

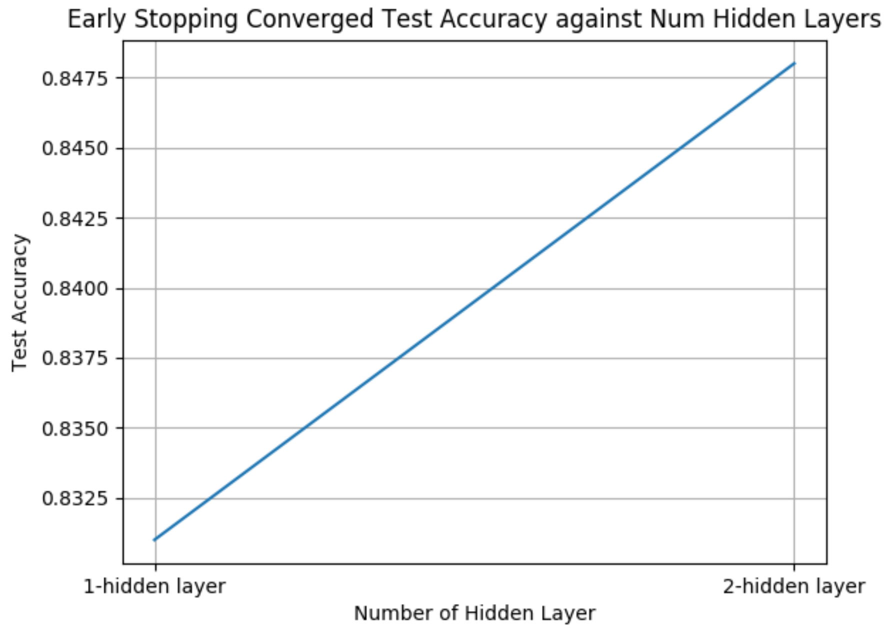
FigA.Q5a.1



FigA.Q5a.2

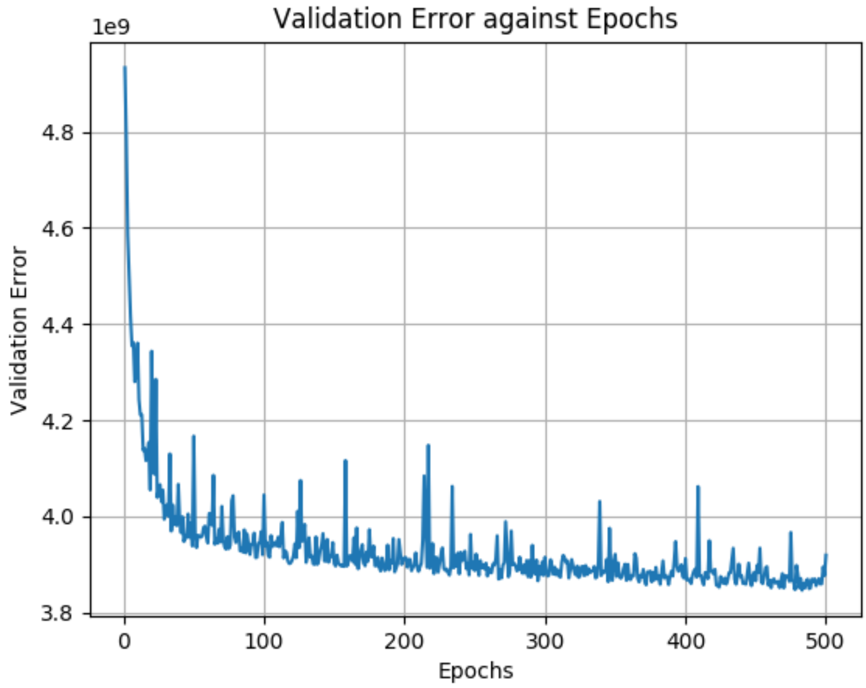


FigA.Q5b.1

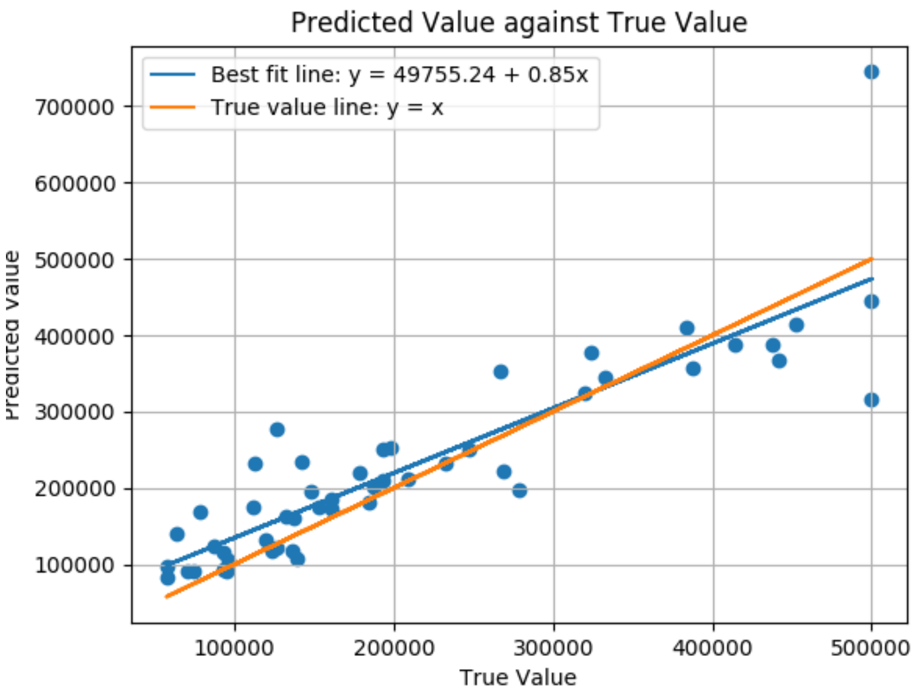

Early Stopping Converged Test Accuracy against Num Hidden Layers
0.8475
0.8450
0.8425
0.8400
0.8375
0.8350
0.8325
Test Accuracy
1-hidden layer
2-hidden layer
Number of Hidden Layer

FigA.Q5b.2

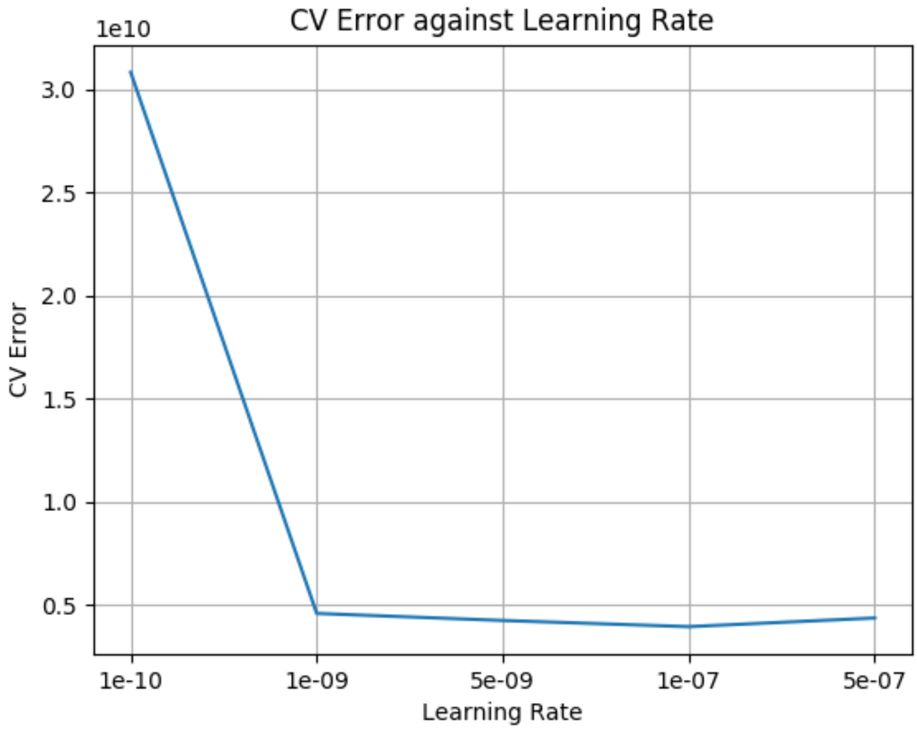
Part B Conclusion Figures



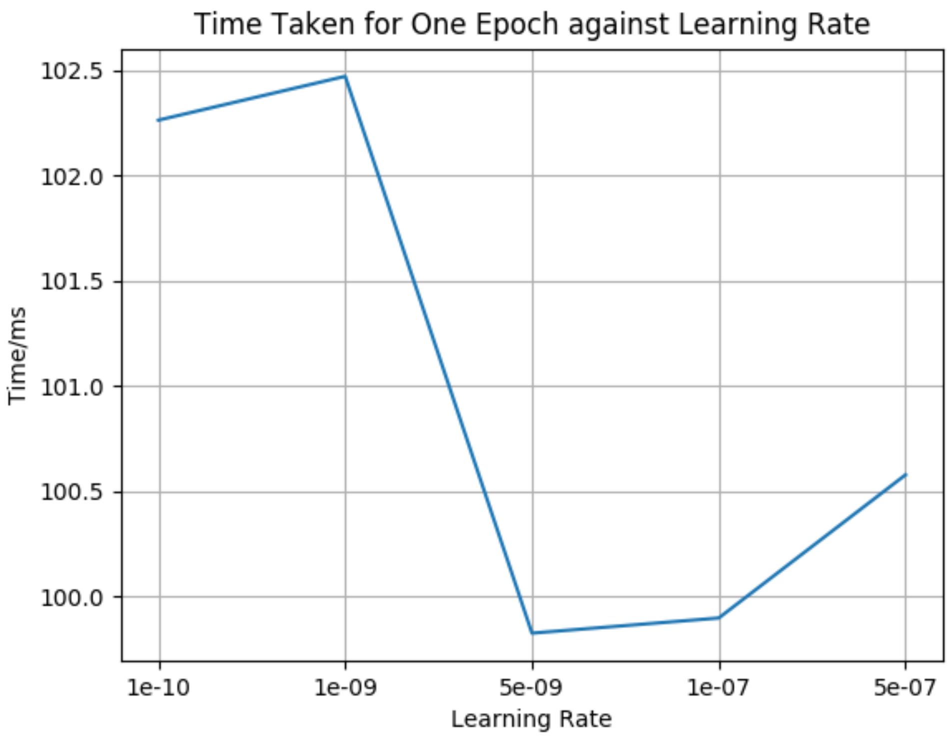
FigB.Q1a



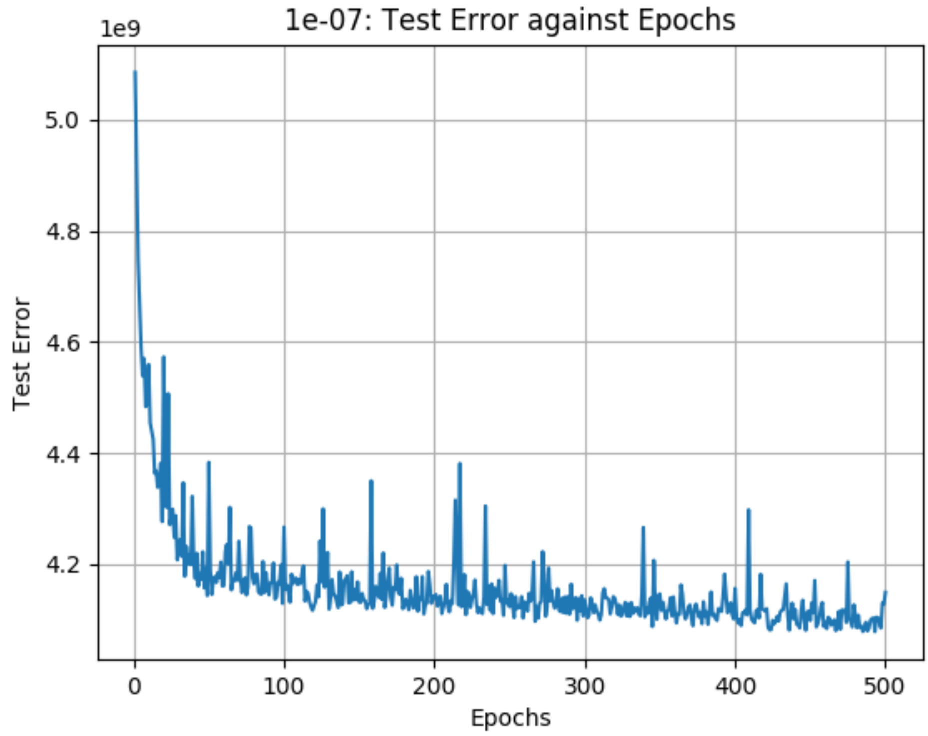
FigB.Q1b



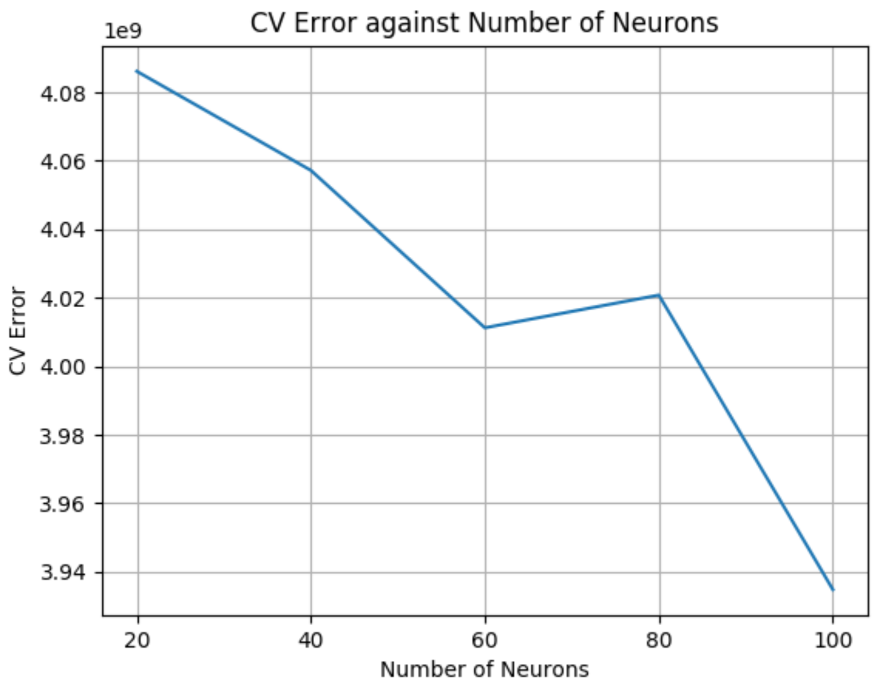
FigB.Q2a



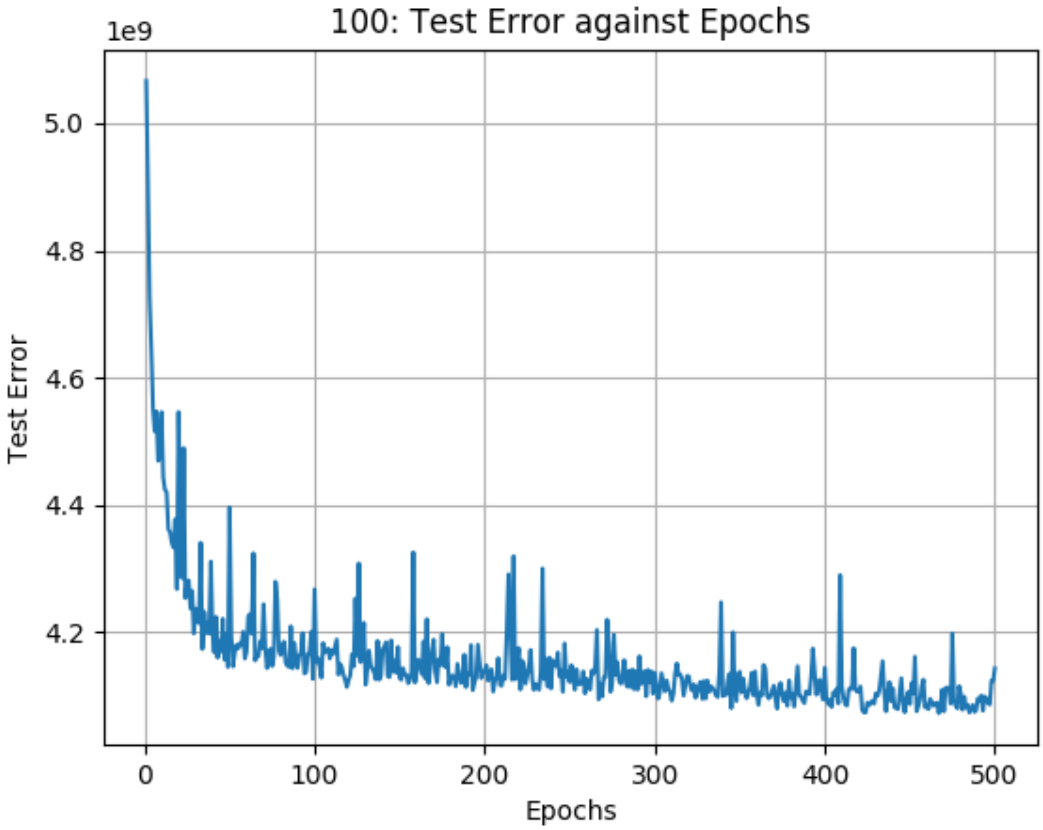
FigB.Q2b.1



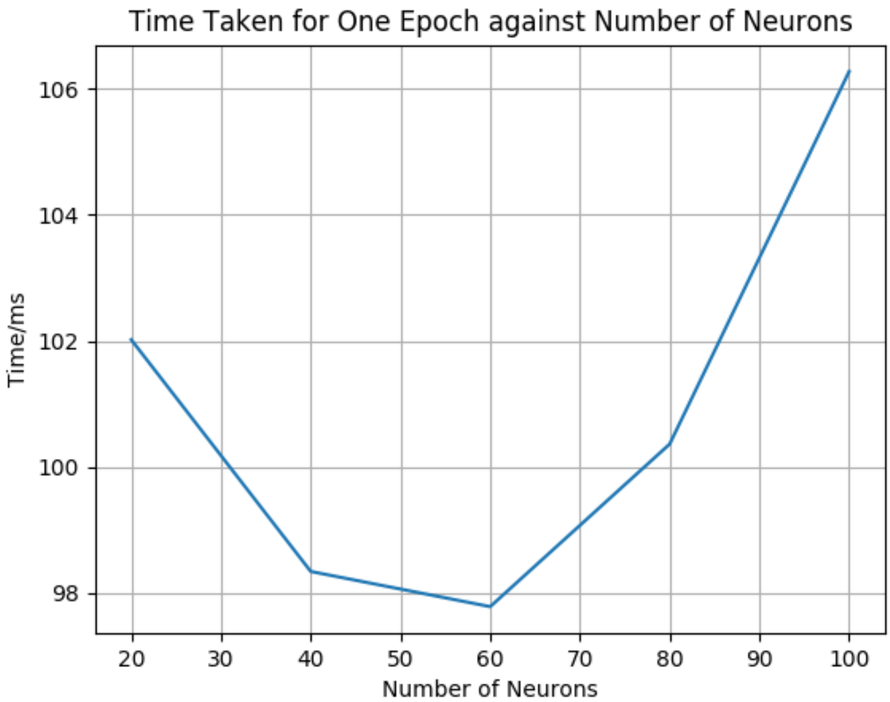
FigB.Q2b.2



FigB.Q3a

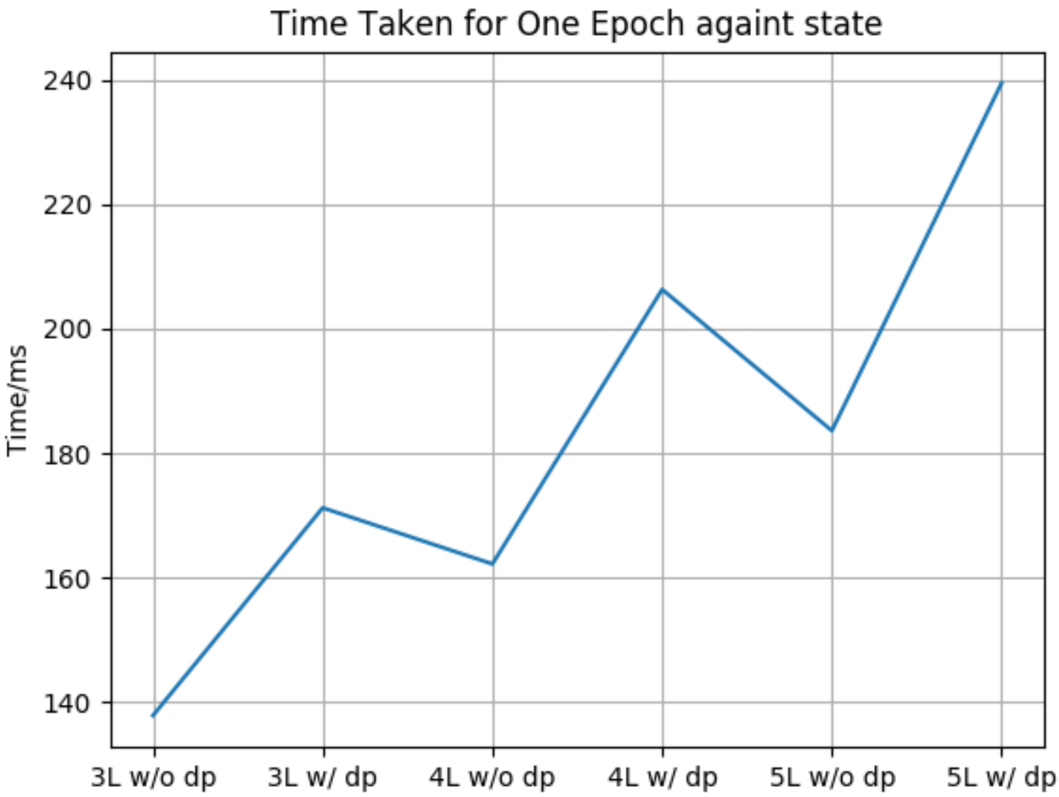


FigB.Q3b.1



FigB.Q3c

FigB.Q4



FigB.Q4.1

Reference

1. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec9.pdf
2. http://www.cs.bham.ac.uk/~jxb/INC/l10.pdf
3. https://openreview.net/pdf?id=H1oyRlYgg