

**CZ4042 Neural Networks**

**Project 1 Report**

**Chen Zhiwei (\_\_)**

**Zeng Jinpo (U1620575J)**

Table of Contents

[Part A - Classification Problem 4](#_Toc526634444)

[1. Introduction 4](#_Toc526634445)

[2. Method 4](#_Toc526634446)

[2.1 Data preprocessing: Normalization of inputs 4](#_Toc526634447)

[2.2 Model Development 4](#_Toc526634448)

[2.2.1 Architecture 4](#_Toc526634449)

[2.2.2 Learning Goal 5](#_Toc526634450)

[2.2.3 Weights Initialisation - Truncated Normal Distribution 5](#_Toc526634451)

[2.2.4 Mini-batch gradient descent 6](#_Toc526634452)

[2.2.5 Optimising Hyper Parameters and Selection Criteria 6](#_Toc526634453)

[2.2.6 Early Stopping 6](#_Toc526634454)

[3. Experiments and Results 7](#_Toc526634455)

[3.1 Optimal Batch Size = 16 7](#_Toc526634456)

[3.2 Optimal Number of hidden neurons = 10 8](#_Toc526634457)

[3.3 Optimal Decay Parameter (L2) = 1 x 10-6 9](#_Toc526634458)

[3.4 Optimal Number of Layers = 1 9](#_Toc526634459)

[4. Conclusion 10](#_Toc526634460)

[Part B - Regression Problem 15](#_Toc526634461)

[1. Introduction 15](#_Toc526634462)

[2. Method 15](#_Toc526634463)

[2.1 Data preprocessing : Train Test Split & Normalization 15](#_Toc526634464)

[2.2 Model Development 15](#_Toc526634465)

[2.2.1 Architecture 15](#_Toc526634466)

[2.2.2 Learning Goal 16](#_Toc526634467)

[2.2.3 Weights Initialisation - Truncated Normal Distribution 16](#_Toc526634468)

[2.2.4 Optimising Hyper Parameters and Selection Criteria 17](#_Toc526634469)

[2.2.5 Three-Way Data Splits 17](#_Toc526634470)

[2.2.6 5-fold cross-validation with mini-batch gradient descent: 17](#_Toc526634471)

[3. Experiments and Results 18](#_Toc526634472)

[3.1 The optimal learning rate = 0.5 x 10-6 18](#_Toc526634473)

[3.2 Optimal number of hidden neurons = 20 18](#_Toc526634474)

[3.3 Comparison of models with different layers (with / without dropouts) 19](#_Toc526634475)

[4. Conclusion 20](#_Toc526634476)

[Appendix 24](#_Toc526634477)

[Classification report with precision/recall/f1 score 24](#_Toc526634478)

[Classification report for different batch sizes without Early Stopping 24](#_Toc526634479)

[Classification report for different L2-regularized term with early stopping 28](#_Toc526634480)

[Classification report for different number of layers with early stopping 29](#_Toc526634481)

[Part A Conclusion Figures 29](#_Toc526634482)

[FigA.Q2a.2 30](#_Toc526634483)

[Part B Conclusion Figures 43](#_Toc526634484)

[FigB.Q1a 43](#_Toc526634485)

[FigB.Q2b.1 44](#_Toc526634486)

[FigB.Q2b.2 44](#_Toc526634487)

[FigB.Q3a 45](#_Toc526634488)

[FigB.Q3b 45](#_Toc526634489)

[FigB.Q3c 46](#_Toc526634490)

[FigB.Q4 46](#_Toc526634491)

# Part A - Classification Problem

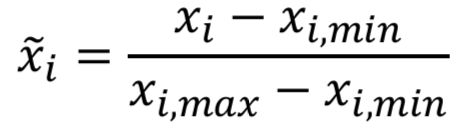
# 1. Introduction

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 Data Pre-processing: Normalization of inputs

Initially, we scaled all input attributes of both train and test data into [0, 1] by the following formula:



Here, the maximum and the minimum were only calculated over the training data, as the model would not know the test data in advance. So the test data are also scaled using the train data’s maximum and minimum.

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

## 2.2 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.

### 2.2.1 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](#_ucu279vkxfaj).

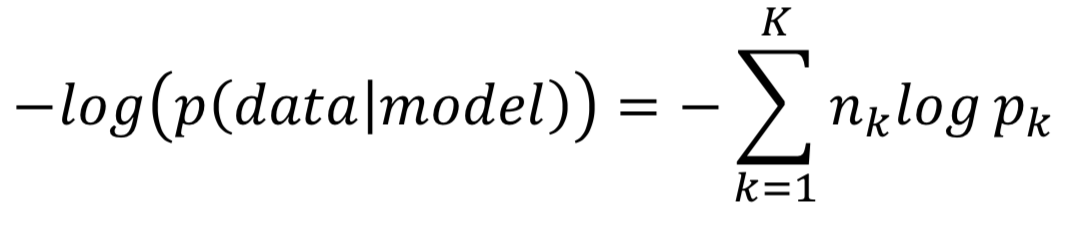
For Q5 of part A, we developed another 4-layer feedforward neural network, with the following architecture:

* 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.

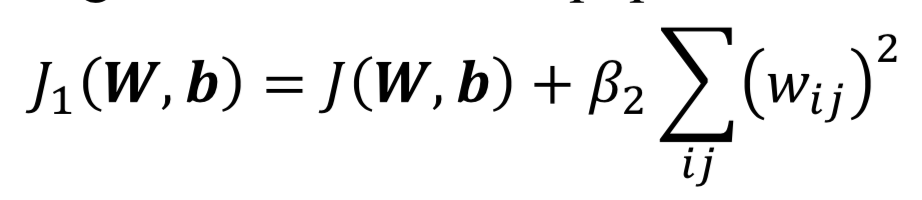
### 2.2.2 Learning Goal

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:



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:



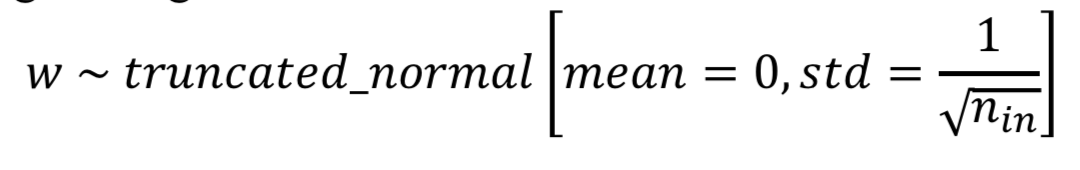
In this assignment, we had experimented with a list of different L2 regularization term, and the final results are further discussed in [section 3.3](#_mpw93w8jnpy9).

### 2.2.3 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:



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.

### 2.2.4 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](#_fpe53a5odnhl).

### 2.2.5 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:

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

### 2.2.6 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:

1. Converged Test Accuracy:
   1. The model with the higher final converged test accuracy is generally better.
2. Convergence Time:
   1. The model with shorter convergence time is generally less costly to train, and thus better.
3. Model Complexity:
   1. The less complex model is better able to generalise, and generally less costly to train, and thus generally better.

### 2.2.7 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:

1. Patience (default: 20) - Number of epochs with no min\_delta improvement after which training will be stopped.
2. Min\_delta (default: 0.005) - Minimum improvement in the monitored quantity to qualify as an improvement.

For example, if the validation error did not improve by min\_delta of 0.005 for consecutive 20 epochs (patience), the training will be terminated early.

# 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 = 0.005

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](#_ovti7g91595o). 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. Since the results of precision/recall/f1 score largely coincides with the results of test accuracy, we will not discuss them in this section, but rather they can be found in Appendix [Classification report](#_Classification_report_with).

## 3.1 Optimal Batch Size = 16

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Batch Size | Converged Test Accuracy | No. of Epochs | Time per Epoch (ms) | Convergence Time (ms) |
| 4 | 0.8368 | 151 | 271 | 40,921 |
| 8 | 0.8320 | 94 | 143 | 13,442 |
| **16** | **0.8323** | **108** | **73** | **7884** |
| 32 | 0.8280 | 149 | 38 | 5662 |
| 64 | 0.8270 | 233 | 21 | 4893 |

Apply the 3 criteria for optimal hyper-parameter:

1. [Convergence Time] The total time taken by models with batch size 4/8 are significantly longer than the rest, while the differences in time among other models are similar.
2. [Converged Test Accuracy] Also, the improvement in test accuracy for batch size 4/8 are less than 1.5%, which is not very significant, and thus not worth the significantly more amount of training time required. Therefore, batch size in 16/32/64 were the candidates that remained.
3. [Converged Test Accuracy] Next, among 16/32/64, the batch size 16 gives the highest test accuracy.
4. [Model Complexity] The model complexity is not affected by batch size.

Hence, it is determined that the optimal batch size is 16.

## 3.2 Optimal Number of hidden neurons = 10

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number of Hidden Neurons | Converged Test Accuracy | No. of Epochs | Time / Epoch (ms) | Total Time (ms) |
| 5 | 0.8240 | 100 | 69 | 6,900 |
| **10** | **0.8320** | **108** | **76** | **8,208** |
| 15 | 0.8280 | 102 | 76 | 7,752 |
| 20 | 0.8300 | 113 | 75 | 8,475 |
| 25 | 0.8295 | 108 | 77 | 8,316 |

Apply the 3 criteria for optimal hyper-parameter:

1. [Convergence Time] The total time taken by all models are relatively close to each other, this means the training cost is about the same for different number of neurons when early stopping is applied. Thus, all can be retained as candidates.
2. [Converged Test Accuracy] Their converged test accuracies are also relatively close to each other (within 1%). Thus, all can be retained as candidates.
3. [Model Complexity] Next, since we value simplicity of the model, the best candidate will be among number of neurons between 5/10.
4. [Model Complexity] Moreover, even though we values simplicity, it is also important to note that the input dimension is 36. Thus, 10 neurons may be able to better learn the task with 36 features than only 5 neurons.
5. [Converged Test Accuracy] Lastly, 10 hidden neurons give higher converged test accuracy, which is about 1% higher than 5 hidden neurons. While this improvement is not very significant, the training cost was not significantly greater too. Thus, 10 hidden neurons will be the best choice.

Hence, it is determined that the optimal number of hidden neurons is 10.

## 3.3 Optimal Decay Parameter (L2) = 1 x 10-6

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Decay Parameter | Converged Test Accuracy | No. of Epochs | Time / Epoch (ms) | Total Time (ms) |
| 0 | 0.8270 | 109 | 72 | 7,848 |
| 1 x 10-12 | 0.8270 | 109 | 74 | 8,066 |
| 1 x 10-9 | 0.8270 | 109 | 74 | 8,066 |
| **1 x 10-6** | **0.8320** | **108** | **76** | **8,208** |
| 1 x 10-3 | 0.8280 | 91 | 77 | 7,007 |

Apply the 3 criteria for optimal hyper-parameter:

1. [Convergence Time] The total time taken by all 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. Thus, all can be retained as candidates.
2. [Converged Test Accuracy] Their converged test accuracies are also relatively close to each other (within 1%). Thus, all can be retained as candidates.
3. [Converged Test Accuracy] Lastly, 1 x 10-6 gives the highest converged test accuracy among all candidates. While this improvement is not very significant, the training cost was not significantly greater too. Thus, decay parameter of 1 x 10-6 will be the best choice.

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

## 3.4 Optimal Number of Layers = 1

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number of Hidden Layers | Converged Test Accuracy | No. of Epochs | Time / Epoch (ms) | Total Time (ms) |
| **1** | **0.8280** | **149** | **39** | **5,811** |
| 2 | 0.8340 | 134 | 45 | 6,030 |

Apply the 3 criteria for optimal hyper-parameter:

1. [Convergence Time] The total time taken by both models are relatively close to each other, this means the training cost is about the same for different number of layers when early stopping is applied. Thus, all can be retained as candidates.
2. [Converged Test Accuracy] Their converged test accuracies are also relatively close to each other (within 1%). Thus, all can be retained as candidates.
3. Since we values simplicity of the model, 1 hidden layer is probably a better choice than 2 hidden layers. The reasoning is that if we can achieve about the same level of performance with a simpler model, there is little benefit and motive to employ a more complex model.

Hence, it is determined that the optimal number of hidden layer is 1.

# 4. Conclusion

In conclusion, the optimal hyper parameters should be:

|  |  |
| --- | --- |
| **Optimal Hyper-Parameter** | **Value** |
| Batch Size | 16 |
| Number of hidden neurons | 10 |
| Decay Parameter | 1 x 10-6 |
| Number of Hidden Layers | 1 |

From the experiments, it is found that 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, the converged test accuracy for early stopping models lower than that for normal models, the difference is usually within 3%. 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](#_Part_A_Conclusion) for larger figures):

|  |  |  |
| --- | --- | --- |
|  | **Without early stopping** | **With early stopping** |
| 2(a) the training error against the number of epochs |  |  |
|  | (refer to appendix FigA.Q2a.1for larger figure) | (refer to appendix FigA.Q2a.2 for larger figure) |
| 2(a) the test accuracy against the number of epochs |  |  |
|  | (refer to appendix FigA.Q2a.3 for larger figure) | (refer to appendix FigA.Q2a.4 for larger figure) |
| 2(b) the time taken to train the network for one epoch against different batch sizes |  |  |
|  | (refer to appendix FigA.Q2b.1 for larger figure) | (refer to appendix FigA.Q2b.2 for larger figure) |
| 2(c) the converged test accuracy against different batch sizes |  |  |
|  | (refer to appendix FigA.Q2c.1 for larger figure) | (refer to appendix FigA.Q2c.2 for larger figure) |
| 3(a) the training error against the number of epochs |  |  |
|  | (refer to appendix FigA.Q3a.1 for larger figure) | (refer to appendix FigA.Q3a.2 for larger figure) |
| 3(a) the test accuracy against the number of epochs |  |  |
|  | (refer to appendix FigA.Q3a.3 for larger figure) | (refer to appendix FigA.Q3a.4 for larger figure) |
| 3(b) the time taken to train the network for one epoch against the number of epochs |  |  |
|  | (refer to appendix FigA.Q3b.1 for larger figure) | (refer to appendix FigA.Q3b.1 for larger figure) |
| 3(c) the converged test accuracy against different number of epochs |  |  |
|  | (refer to appendix FigA.Q3c.1 for larger figure) | (refer to appendix FigA.Q3c.2 for larger figure) |
| 4(a) the training error against the number of epochs |  |  |
|  | (refer to appendix FigA.Q4a.1 for larger figure) | (refer to appendix FigA.Q4a.2 for larger figure) |
| 4(b) the converged test accuracy against different values of decay parameter |  |  |
|  | (refer to appendix FigA.Q4b.1 for larger figure) | (refer to appendix FigA.Q4b.2 for larger figure) |
| 5(a) the train error of 4-layer network |  |  |
|  | (refer to appendix FigA.Q5a.1 for larger figure) | (refer to appendix FigA.Q5a.2 for larger figure) |
| 5(b) the test accuracy of 4-layer network |  |  |
|  | (refer to appendix FigA.Q5a.3 for larger figure) | (refer to appendix FigA.Q5a.4 for larger figure) |
| 5(b) Comparison on the performances on the 3-layer and 4-layer networks. |  |  |
|  | (refer to appendix FigA.Q5b.1 for larger figure) | (refer to appendix FigA.Q5b.2 for larger figure) |
| 5(b) the converged test accuracy against different values of decay parameter |  |  |
|  | (refer to appendix FigA.Q5b.3 for larger figure) | (refer to appendix FigA.Q5b.4 for larger figure) |

# Part B - Regression Problem

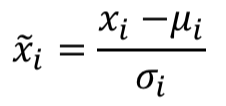
# 1. Introduction

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 train and test set in ratio of 7:3. Then scaled both inputs and output by the following formula:



Here, is the mean, and is the standard deviation of each feature, and they were only calculated over the training data, as the model would not know the test data in advance. So the test data are also scaled using mean and standard deviation of the train data.

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 and 5-fold cross-validation to train the models. In order 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

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
* a 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](#_octtai3225uc).

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](#_octtai3225uc)),
* a hidden discrete perceptron layer of 20 perceptrons with ReLu activation function, and
* a layer with one linear neuron.

5-layer:

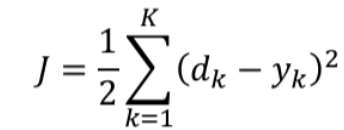
* an input layer of dimension 8 (corresponding to the input feature dimensions),
* a hidden discrete perceptron layer of *optimal\_number* perceptrons with ReLu activation function (optimal number of neurons is discussed in [section 3.2](#_octtai3225uc)),
* a hidden discrete perceptron layers of 20 perceptrons with ReLu activation function,
* a hidden discrete perceptron layers of 20 perceptrons with ReLu activation function, and
* a 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](#_m8zigiejkood).

### 2.2.2 Learning Goal

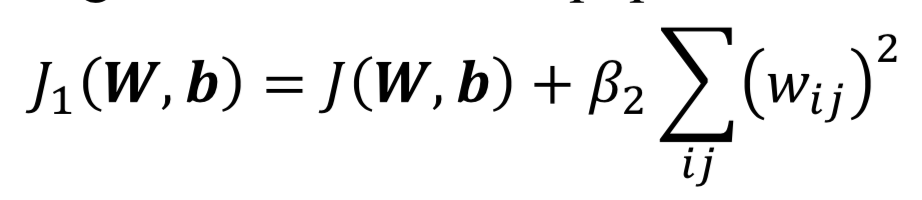
In this assignment, the above 3 neural models mentioned in 2.2.1 aim to minimize L2-regularized loss.

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



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.



1. dropout:

It 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](#_5nzcyc4t9bqm).

### 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:

1. Learning rate - [0.5e-6, 1e-7, 0.5e-8, 1e-9, 1e-10]
2. 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:

1. Converged Test Error:
   1. The model with the lower final converged test error is generally better.
2. Convergence Time
   1. The model with shorter convergence time is generally less costly to train, and thus better.
3. Model Complexity:
   1. The less complex model is better able to generalise, and generally less costly to train, and thus generally better.

### 2.2.6 Train-Test Data Split

We split the data into training data set and testing data set by a ratio of 7:3 during pre-processing. While the training data set is further split into training data and validation data during 5-fold cross-validation as shown in [section 2.2.6](#_ale4qw2y8zlk). The testing data set is used to estimate the error of the network on unseen data.

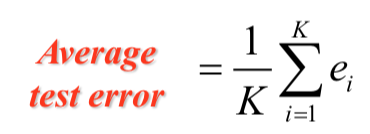
~~As validation data is also used to select in the process, the error estimate of the final model on validation data will be biased. Hence, the separation of test and validation sets is necessary. The test data is used to estimate the performance of the final model after 5-fold cross-validation. There will be no further tuning of the model when evaluating the model performance on the test data.~~

### 2.2.7 5-fold cross-validation with mini-batch gradient descent:

5-fold cross-validation with mini-batch gradient descent 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 dataset.
3. For each of 5 experiments:
4. use 4 folds for training and the remaining one-fold for testing.
5. 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 fold, so as to calculate cross-validation error using the following formula:



Cross-validation error

The advantage of 5-fold cross validation is that all data in the dataset are used for both training and testing.

# 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

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 The optimal learning rate = 0.5 x 10-6

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

|  |  |  |
| --- | --- | --- |
| Learning rate | Converged Test Error | Time / Epoch (ms) |
| **0.5 x 10-6** | **1.032** | **42.14** |
| 10-7 | 1.277 | 41.97 |
| 0.5 x 10-8 | 1.418 | 41.99 |
| 10-9 | 1.422 | 41.87 |
| 10-10 | 1.421 | 41.92 |

Apply the 3 criteria for optimal hyper-parameter:

1. [Converged Test Error] The test error is the lowest for the model with learning rate 0.5 x 10-6, which is significantly lower than the others, and worth the additional training time per epoch.
2. [Convergence Time] The time taken per epoch for the model with learning rate 0.5 x 10-6 is the longest, but of little difference to the rest (0.27 ms longer than the shortest time taken per epoch).
3. [Model Complexity] The model complexity will not be affected by the learning rate.

Hence, it is determined that learning rate is 0.5 x 10-6.

## 3.2 Optimal number of hidden neurons = 20

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

|  |  |  |
| --- | --- | --- |
| Number of hidden neurons | Converged Test Error | Time per Epoch (ms) |
| **20** | **1.122** | **45.23** |
| 40 | 1.212 | 44.17 |
| 60 | 1.170 | 41.45 |
| 80 | 1.173 | 49.20 |
| 100 | 1.118 | 47.92 |

Apply the 3 criteria for optimal hyper-parameter:

1. [Converged Test Error] The test error for the model with 20 hidden neurons is the second lowest (1.122), slightly greater than the model with 100 number of hidden neurons (1.118).
2. [Convergence Time] The time taken per epoch for the model with 20 hidden neurons is the third among the experimented models with different number of hidden neurons, longer than the model with 60 hidden neurons by 3.78 ms.
3. [Model Complexity] The model is the simplest for the model with 20 hidden neurons.
4. In this experiment, the model complexity is the primary concern, as model performance in terms of other two factors are similar for the experimented models.

Hence, it is determined that the optimal number of hidden neurons is 20.

## 3.3 Comparison of models with different layers (with / without dropouts)

|  |  |
| --- | --- |
| Model | Test Error |
| 3-layer w/o dropout | 1.5383 |
| 3-layer w/ dropout | 1.5561 |
| 4-layer w/o dropout | 1.3122 |
| 4-layer w/ dropout | 1.3325 |
| **5-layer w/o dropout** | **1.1628** |
| 5-layer w/ dropout | 1.1677 |

Table 1. models with different layers with / without dropout

The test error for the model with 5 layers without dropout (1.1628), is the lowest, significantly lower than the model with 3 or 4 layers.

It is also noteworthy that, in this experiment, models with dropout generally have higher test errors than models without dropout. Hence, dropout is not preferred in this experiment. Thus, model with 5 layers and without dropout is the best performing model among the 6 models trained.

# 

# 

# 4. Conclusion

In conclusion, the optimal hyper parameters should be:

|  |  |
| --- | --- |
| **Optimal Hyper-Parameter** | **Value** |
| The optimal learning rate | 0.5 x 10-6 |
| Number of hidden neurons | 20 |

While the dropout in general can improve the generalization of the model, as discussed in [section 2.2.2](#_4suvzmbexlyf), from the experiment, it is noted that the model with dropout usually has a higher test error than the model without dropout. Hence, dropout is not preferred in this case.

The requested graphs for Part B (refer to [Appendix Part B](#_Part_B_Conclusion)):

|  |  |
| --- | --- |
| 1(a) Validation errors against epochs |  |
|  | (refer to appendix FigB.Q1a) |
| 1(b) Predicted values against target values for any 50 test samples |  |
|  | (refer to appendix FigB.Q1b) |
| 2(a) Cross-validation errors against learning rate |  |
|  | (refer to appendix FigB.Q2a) |
| 2(b) Test error against training epoch |  |
|  | (refer to appendix FigB.Q2b.1) |
| 2(b) Time taken for one epoch against learning rate |  |
|  | (refer to appendix FigB.Q2b.2) |
| 3(a) Cross-validation errors against the number of hidden-layer neurons |  |
|  | (refer to appendix FigB.Q3a) |
| 3(b) Test errors against number of epochs |  |
|  | (refer to appendix FigB.Q3b) |
| 3(c) Time taken for one epoch against number of neurons |  |
|  | (refer to appendix FigB.Q3c) |
| 4 Test error against the different models with different number of neurons, and  with / without dropout |  |
|  | (refer to appendix FigB.Q4) |

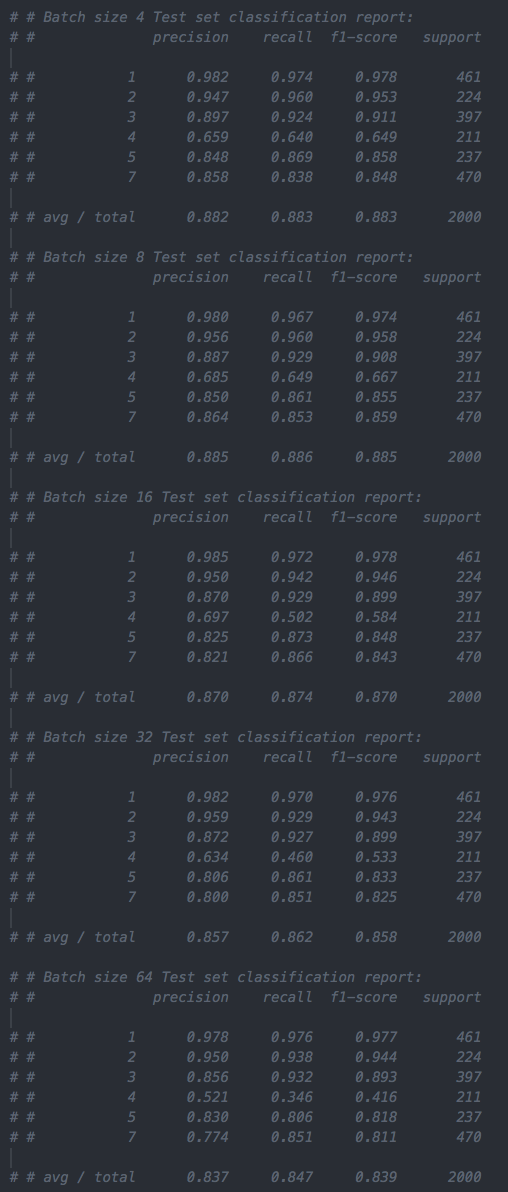
### 

## 

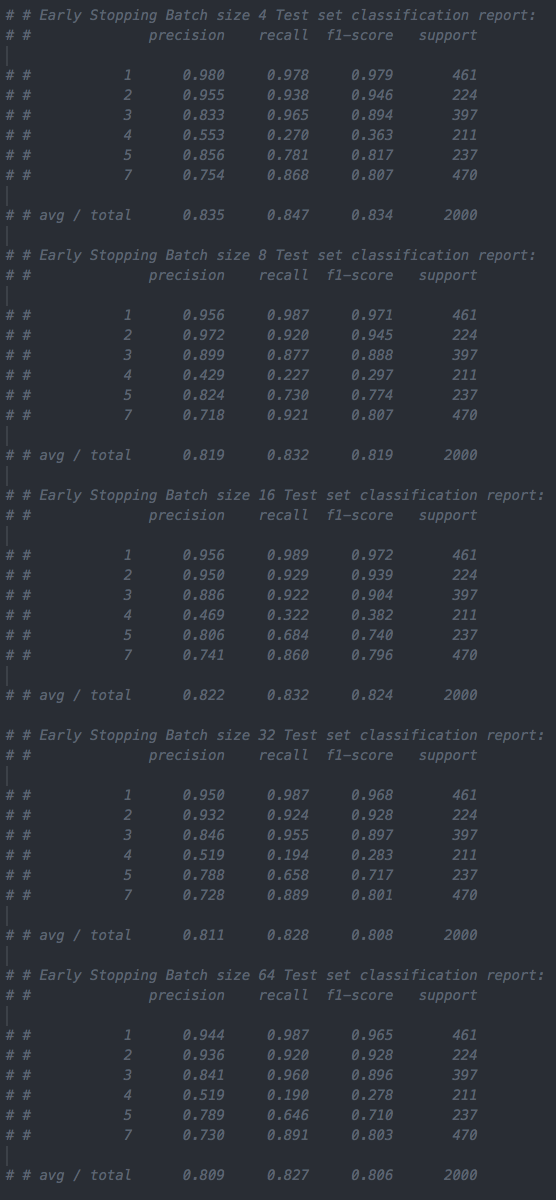
# 

# Appendix

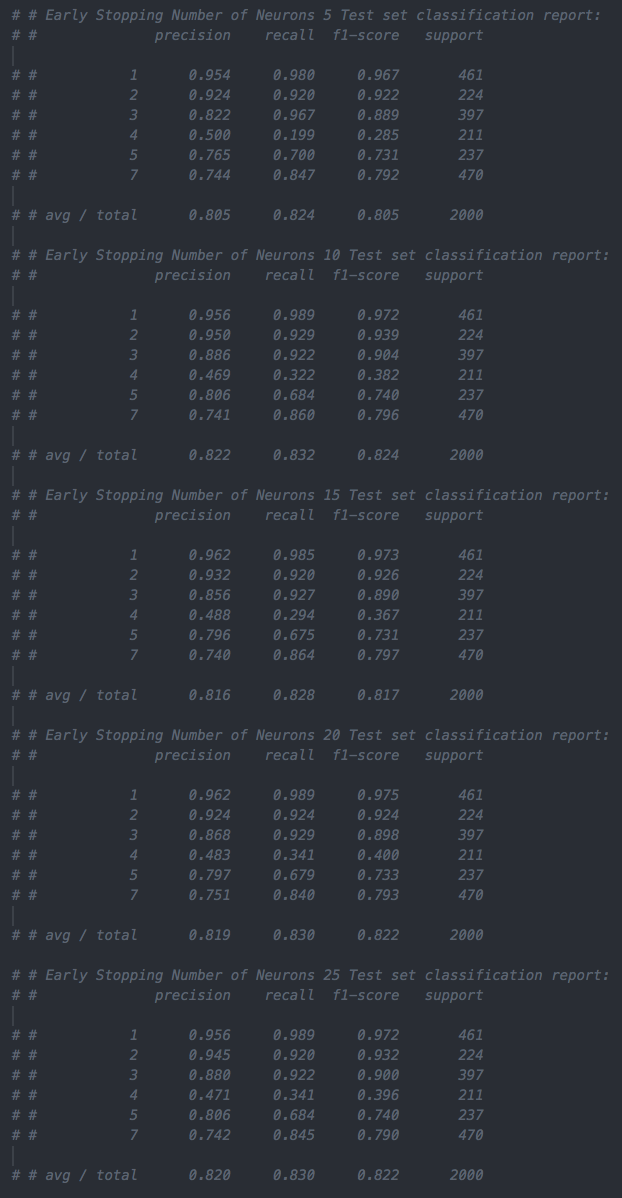
## Classification report with precision/recall/f1 score



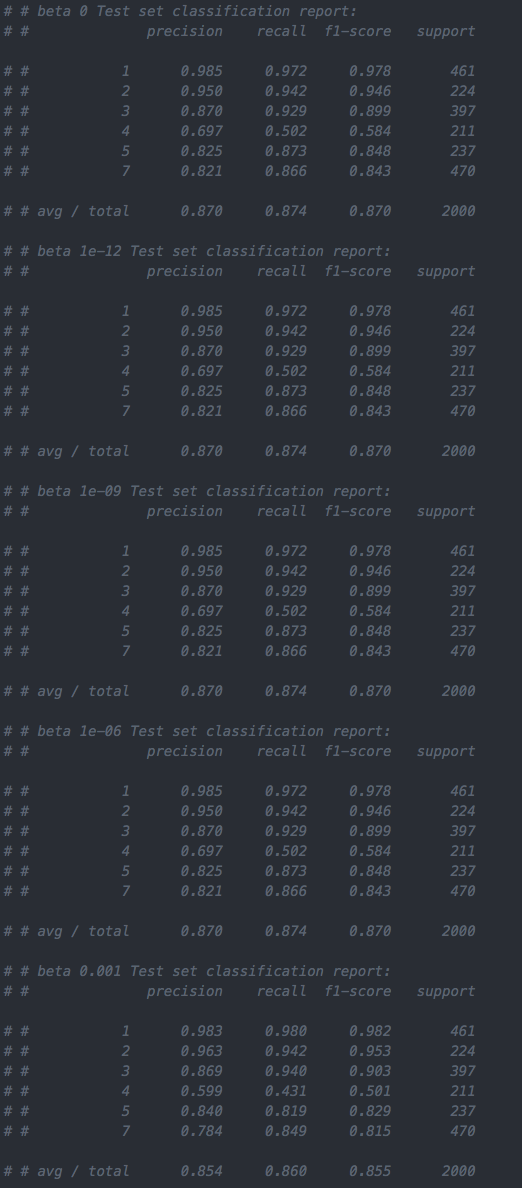
### Classification report for different batch sizes without Early Stopping



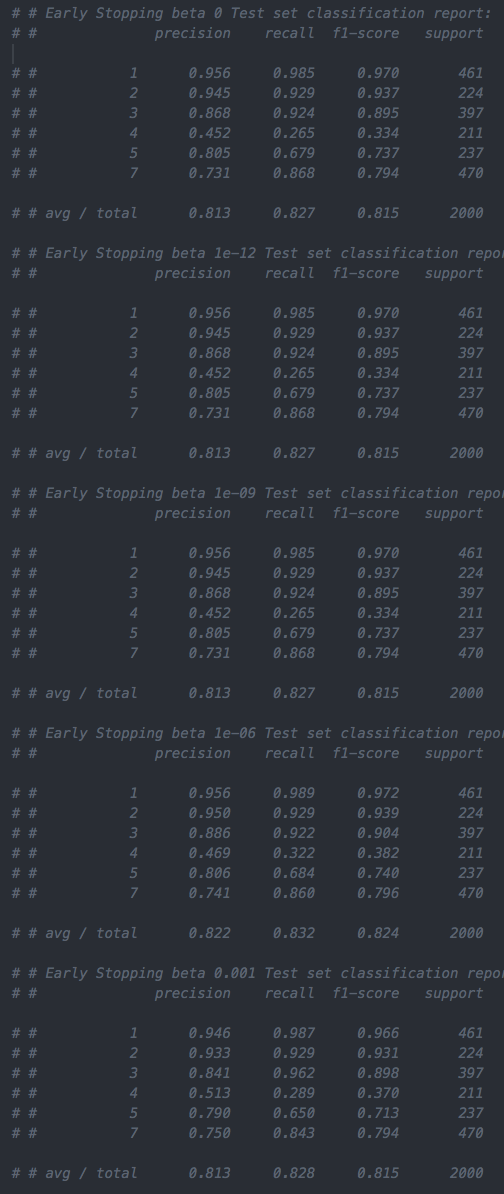
Classification report for different batch sizes with Early Stopping



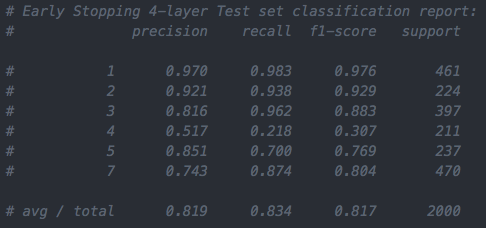
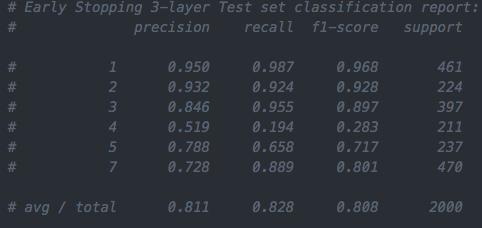
Classification report for different number of neurons with Early Stopping



Classification report for different L2-regularized term without early stopping

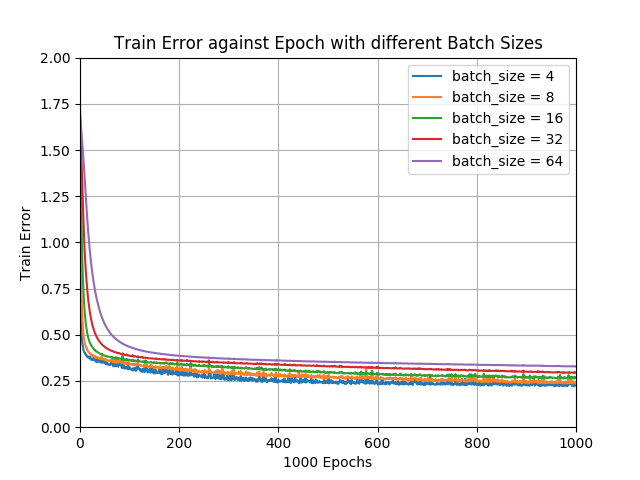


### Classification report for different L2-regularized term with early stopping

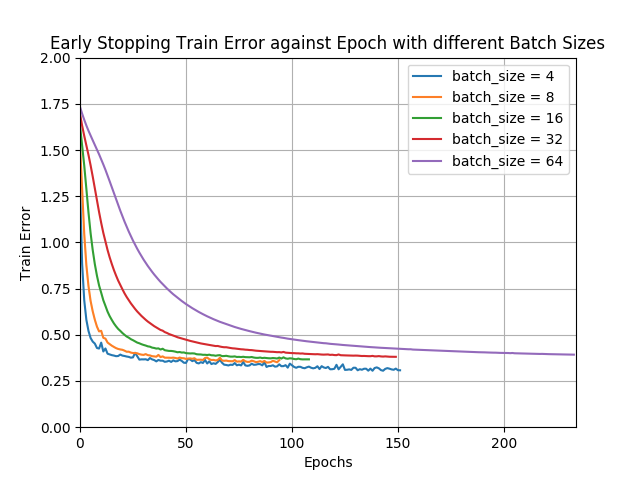


### Classification report for different number of layers with early stopping

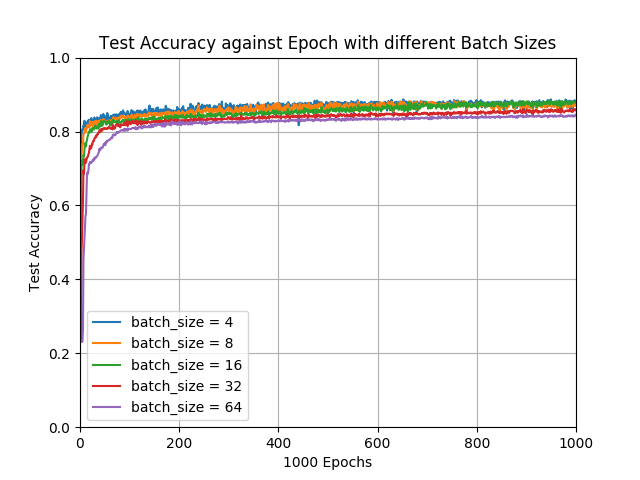
## Part A Conclusion Figures



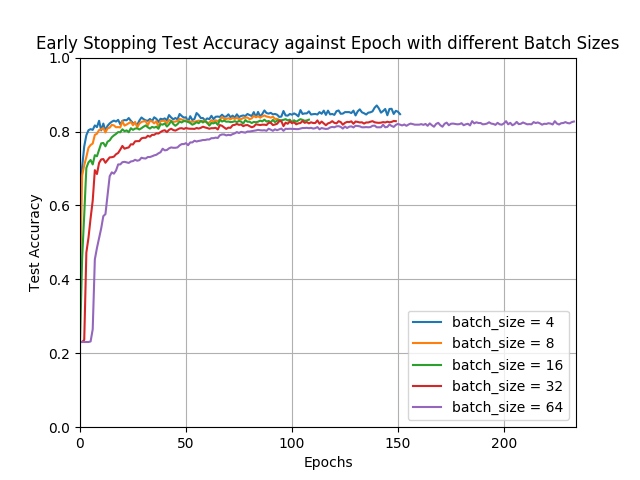
FigA.Q2a.1



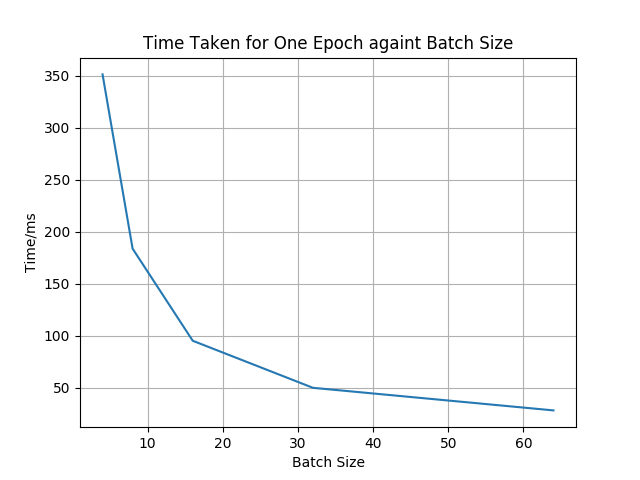
### FigA.Q2a.2



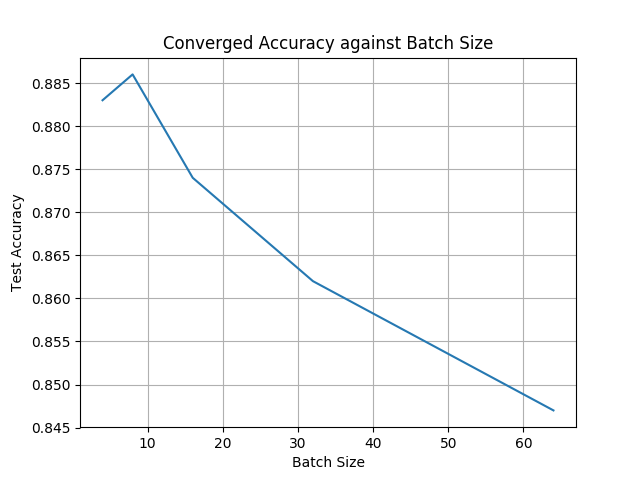
FigA.Q2a.3



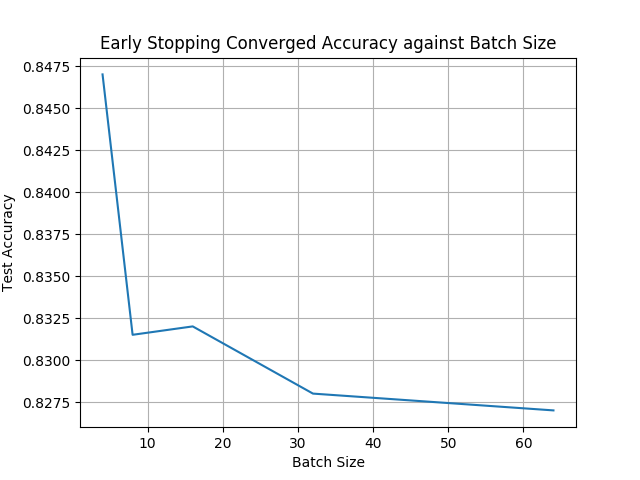
FigA.Q2a.4



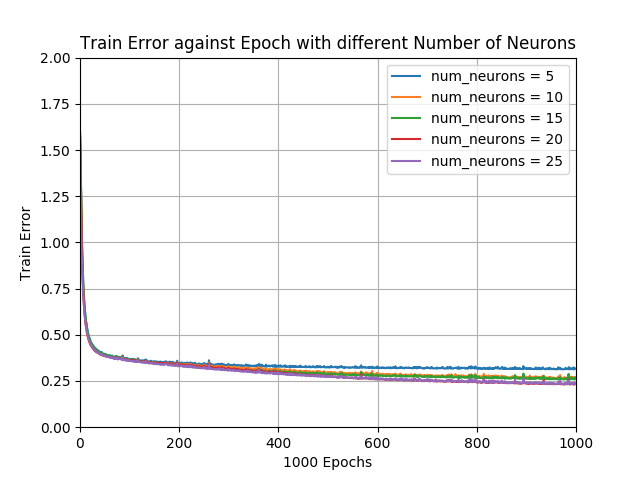
FigA.Q2b.2



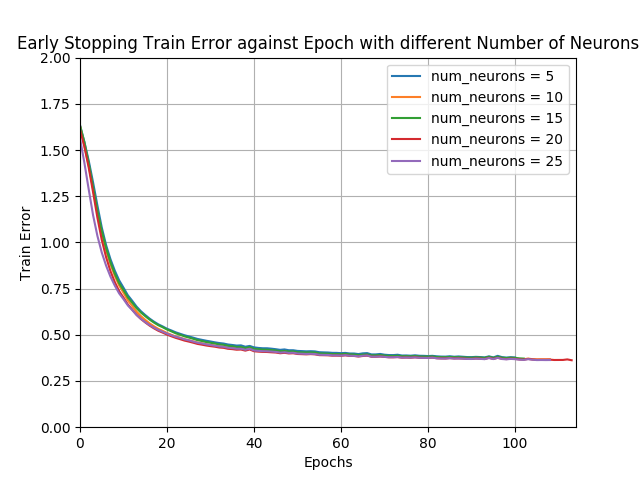
FigA.Q2c.1



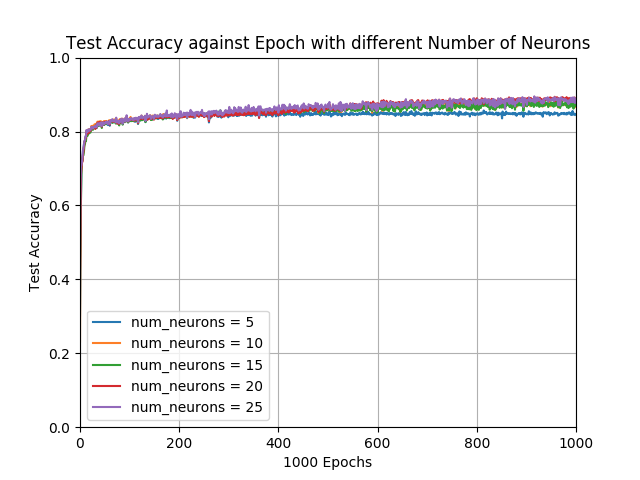
FigA.Q2c.2



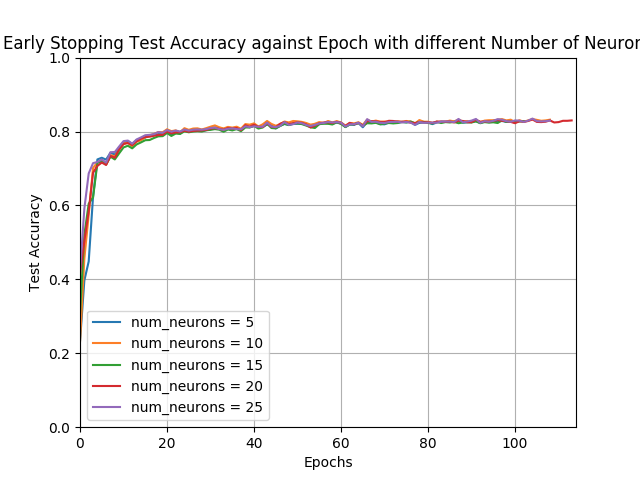
FigA.Q3a.1



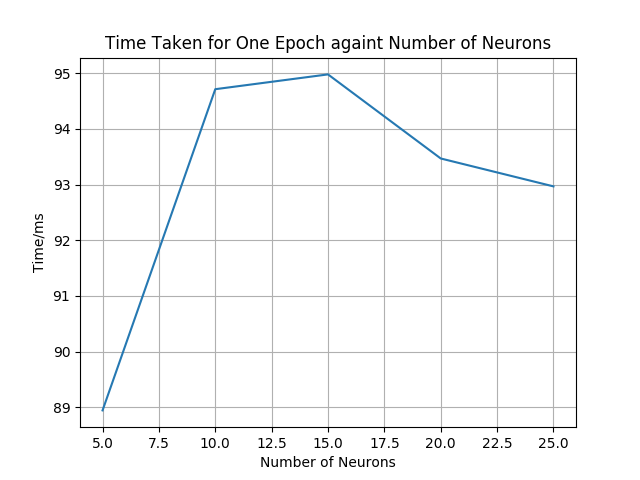
FigA.Q3a.2



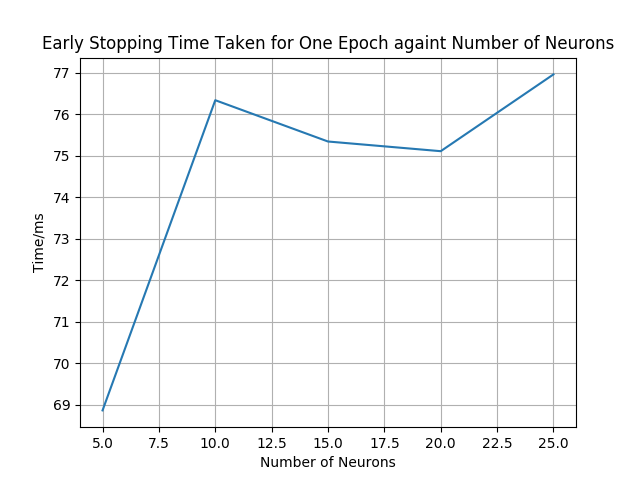
FigA.Q3a.3



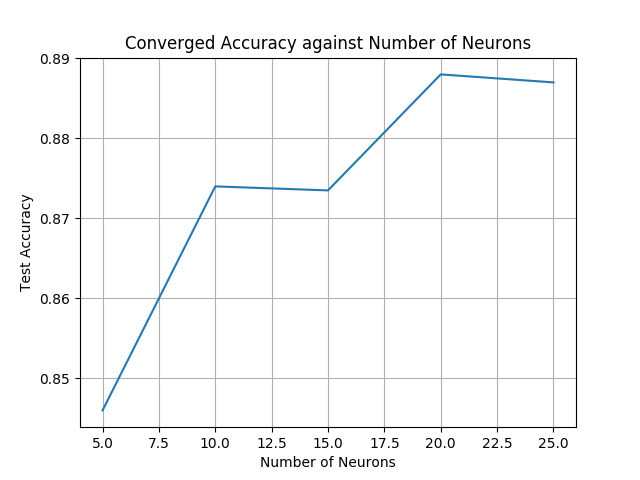
FigA.Q3a.4



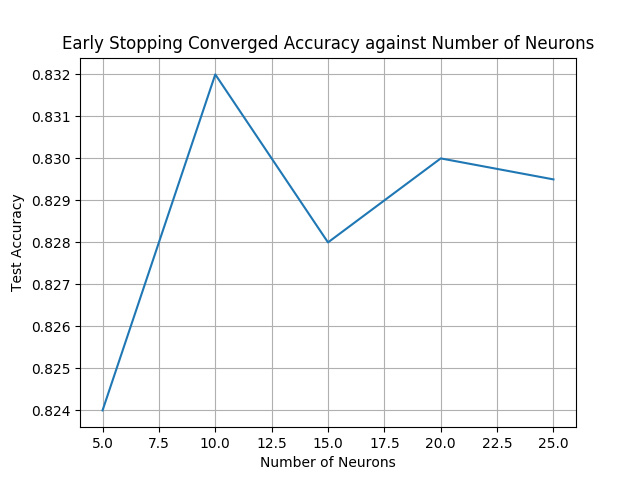
FigA.Q3b.1



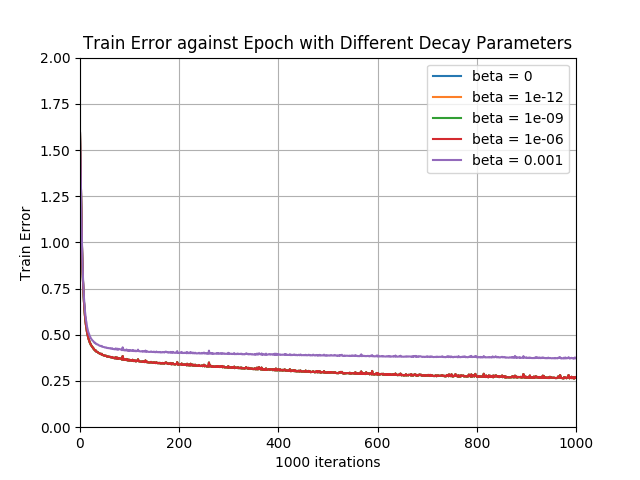
FigA.Q3b.2



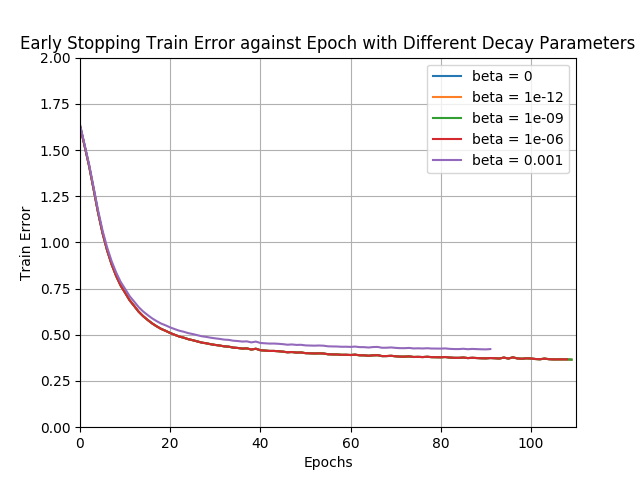
FigA.Q3c.1



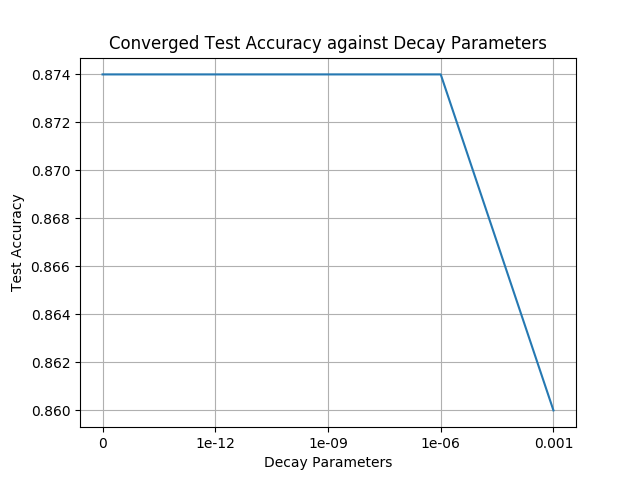
FigA.Q3c.2



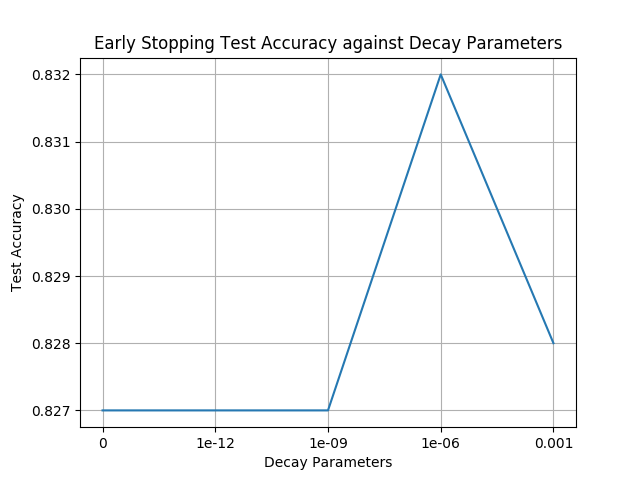
FigA.Q4a.1



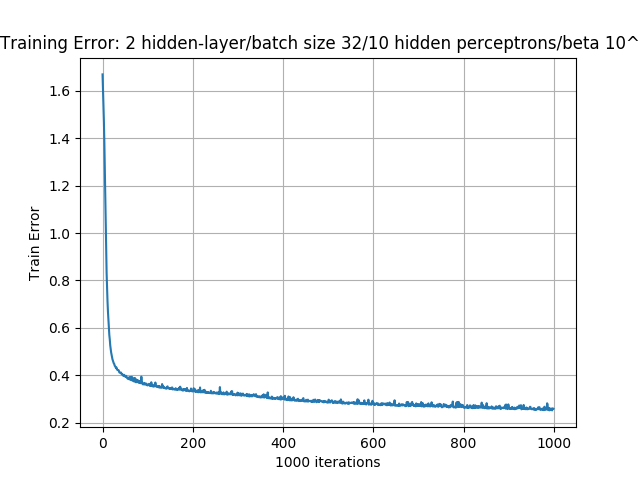
FigA.Q4a.2



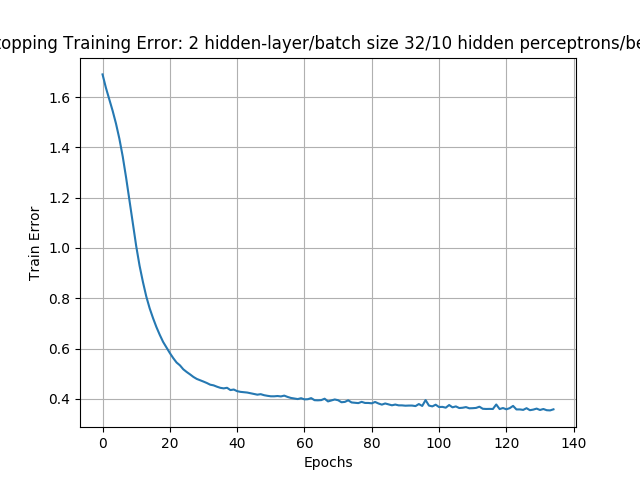
FigA.Q4b.1



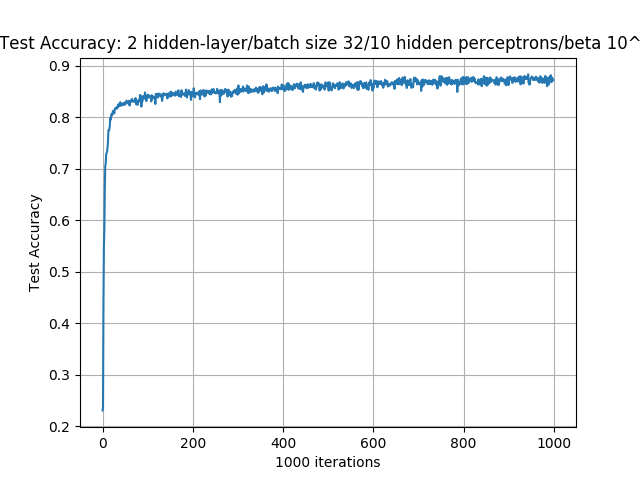
FigA.Q4b.2



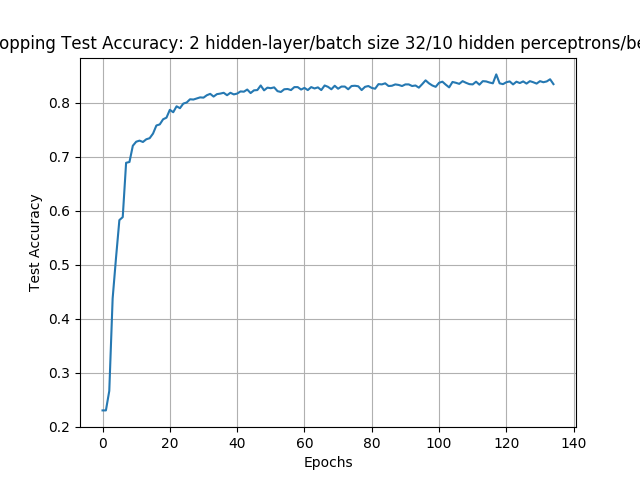
FigA.Q5a.1



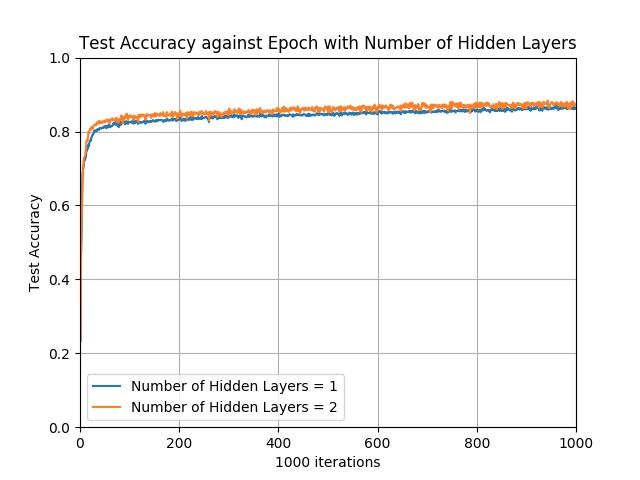
FigA.Q5a.2



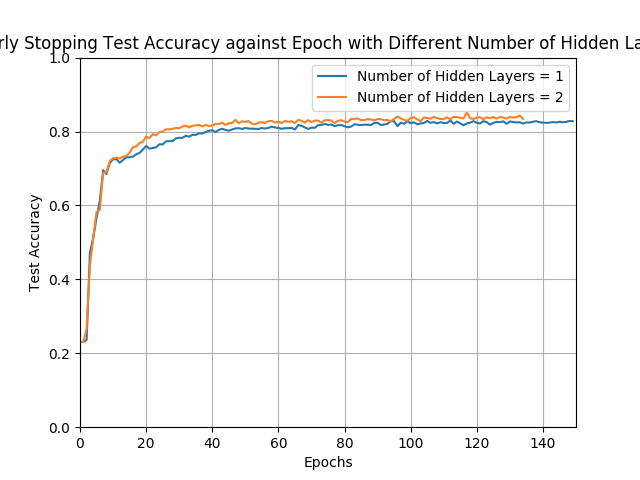
FigA.Q5a.3



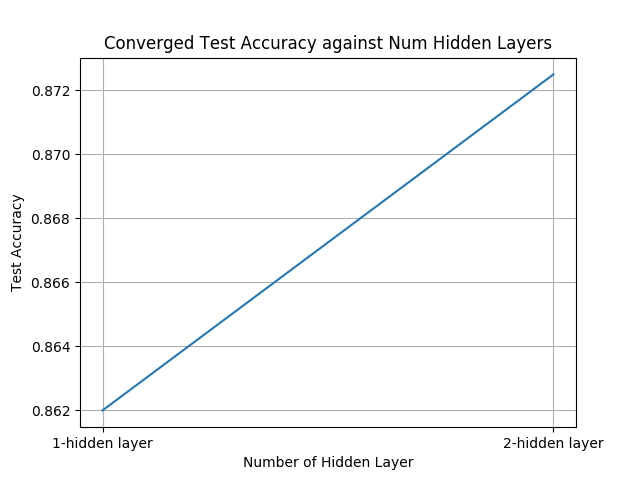
FigA.Q5a.4



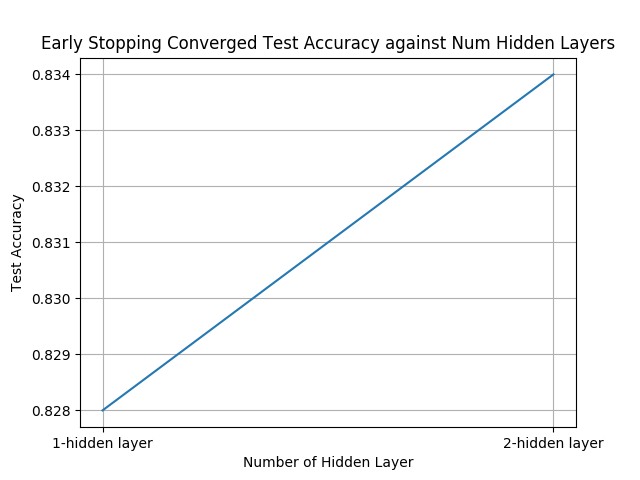
FigA.Q5b.1



FigA.Q5b.2

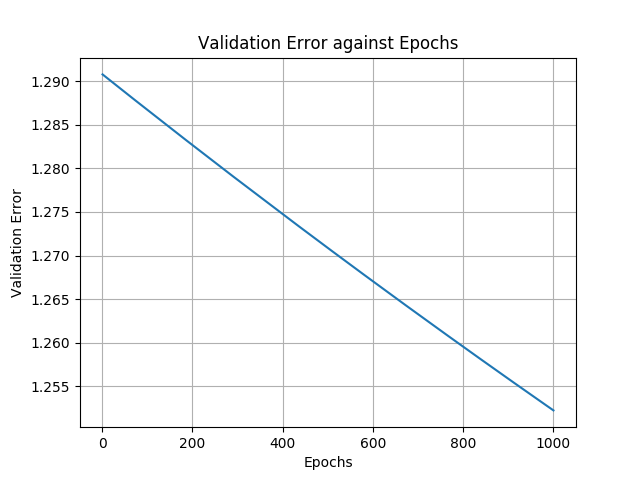


FigA.Q5b.3

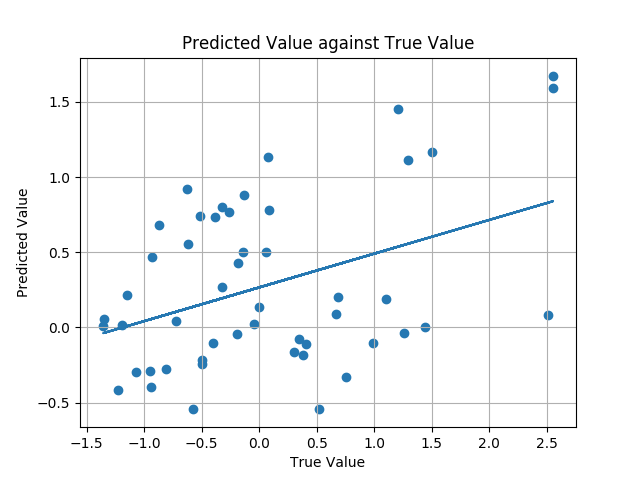


FigA.Q5b.4

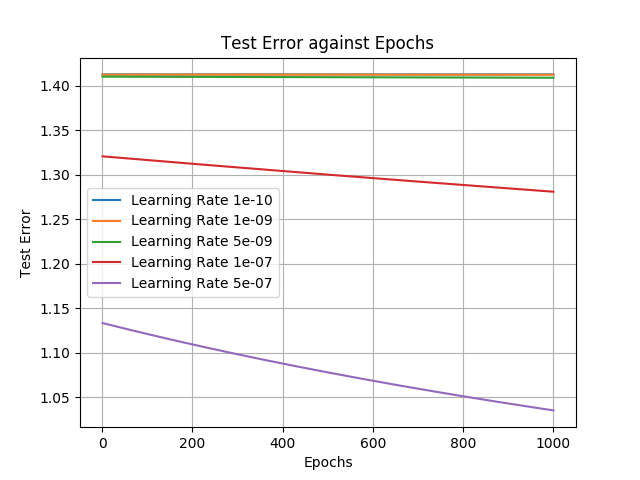
## Part B Conclusion Figures



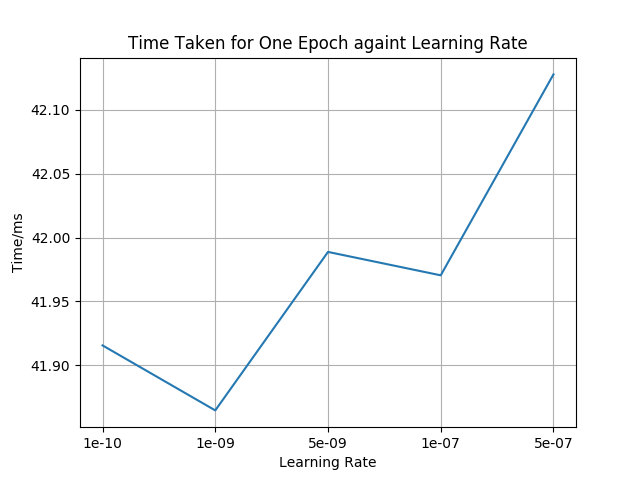
### FigB.Q1a



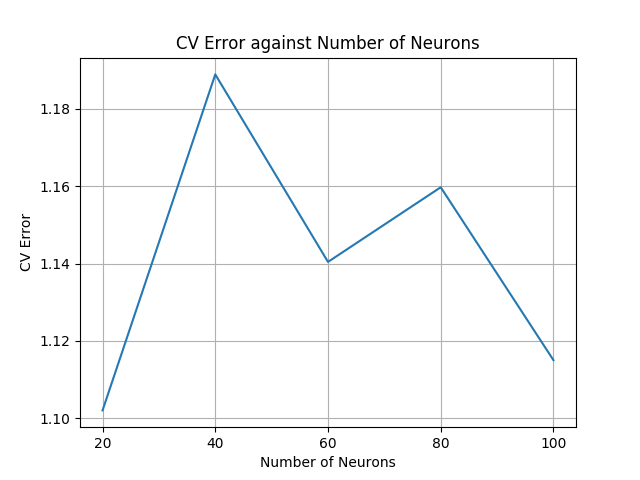
FigB.Q2a



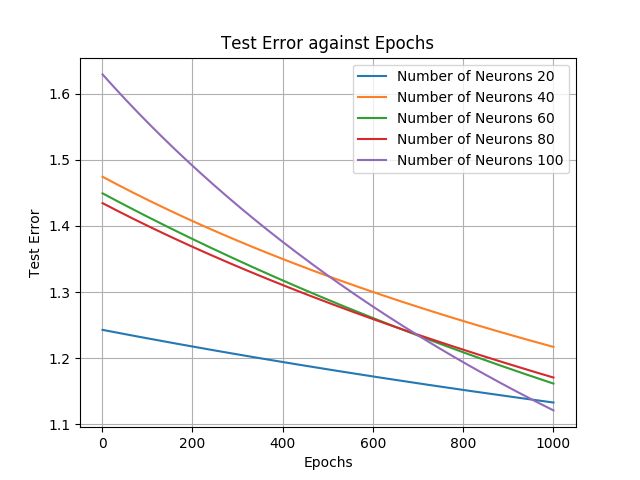
### FigB.Q2b.1



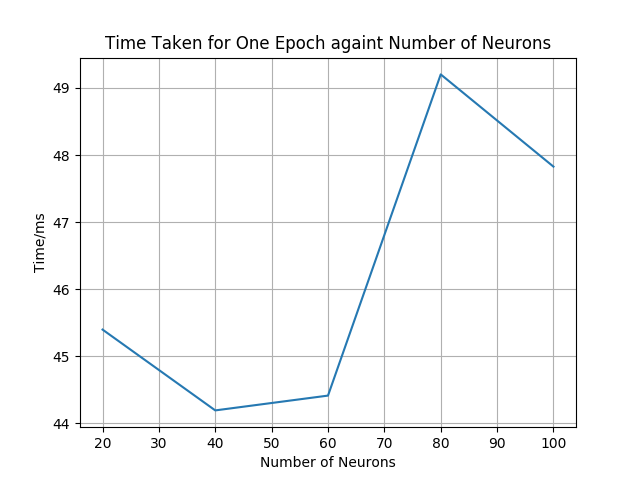
### FigB.Q2b.2



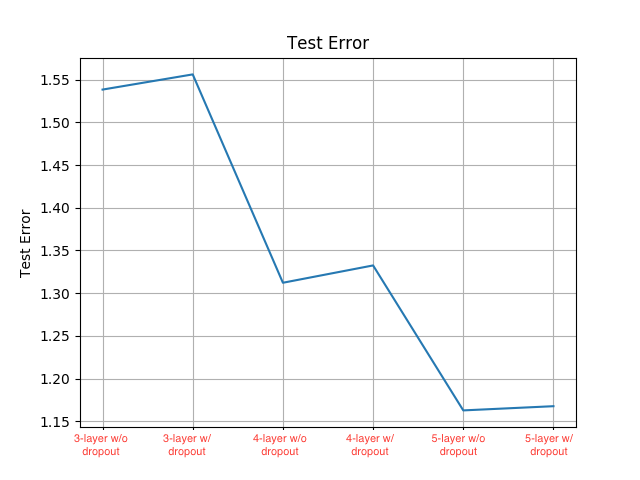
### FigB.Q3a



### FigB.Q3b



### FigB.Q3c



### FigB.Q4