

**CZ4042 Neural Networks**

**Project 1 Report**

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

**Zeng Jinpo (U1620575J)**

Table of Contents

[Part A - Classification Problem 4](#_Toc527232576)

[1. Introduction 4](#_Toc527232577)

[2. Method 5](#_Toc527232578)

[2.1 Three-Way Data Splits Method 5](#_Toc527232579)

[2.2 Data Pre-processing: Normalization of inputs 5](#_Toc527232580)

[2.3 Model Development 5](#_Toc527232581)

[2.3.1 Architecture 5](#_Toc527232582)

[2.3.2 Learning Goal 6](#_Toc527232583)

[2.3.3 Weights Initialisation - Truncated Normal Distribution 6](#_Toc527232584)

[2.3.4 Mini-batch gradient descent 7](#_Toc527232585)

[2.3.5 Optimising Hyper Parameters 7](#_Toc527232586)

[2.3.6 Selection Criteria 7](#_Toc527232587)

[2.3.7 Early Stopping 8](#_Toc527232588)

[3. Experiments and Results 8](#_Toc527232589)

[3.1 Optimal Batch Size = 4 9](#_Toc527232590)

[3.2 Optimal Number of hidden neurons = 20 10](#_Toc527232591)

[3.3 Optimal Decay Parameter (L2) = 1 x 10-6 10](#_Toc527232592)

[3.4 Optimal Number of Layers = 2 11](#_Toc527232593)

[4. Conclusion 12](#_Toc527232594)

[Part B - Regression Problem 20](#_Toc527232595)

[1. Introduction 20](#_Toc527232596)

[2. Method 20](#_Toc527232597)

[2.1 Data Pre-processing: Train Test Split & Normalization 20](#_Toc527232598)

[2.2 Model Development 20](#_Toc527232599)

[2.2.1 Architecture 20](#_Toc527232600)

[2.2.2 Learning Goal 21](#_Toc527232601)

[2.2.3 Weights Initialisation - Truncated Normal Distribution 21](#_Toc527232602)

[2.2.4 Optimising Hyper Parameters 22](#_Toc527232603)

[2.2.5 Selection Criteria 22](#_Toc527232604)

[2.2.6 Train-Test Data Split 22](#_Toc527232605)

[2.2.7 5-fold cross-validation with mini-batch gradient descent: 22](#_Toc527232606)

[3. Experiments and Results 23](#_Toc527232607)

[3.1 The optimal learning rate = 0.5 x 10-6 23](#_Toc527232608)

[3.2 Optimal number of hidden neurons = 20 23](#_Toc527232609)

[3.3 Comparison of models with different layers (with / without dropouts) 24](#_Toc527232610)

[4. Conclusion 25](#_Toc527232611)

[Appendix 29](#_Toc527232612)

[Classification report with precision/recall/f1 score 29](#_Toc527232613)

[Classification report for different batch sizes with Early Stopping 29](#_Toc527232614)

[Classification report for different L2-regularized term with early stopping 31](#_Toc527232615)

[Classification report for different number of layers with early stopping 32](#_Toc527232616)

[Part A Conclusion Figures 33](#_Toc527232617)

[FigA.Q2a.1 33](#_Toc527232618)

[FigA.Q2a.2 33](#_Toc527232619)

[FigA.Q2b.1 34](#_Toc527232620)

[FigA.Q2c.1 34](#_Toc527232621)

[FigA.Q2c.2 35](#_Toc527232622)

[FigA.Q3a.1 35](#_Toc527232623)

[FigA.Q3a.2 36](#_Toc527232624)

[FigA.Q3b.1 37](#_Toc527232625)

[FigA.Q3c.1 37](#_Toc527232626)

[FigA.Q3c.2 37](#_Toc527232627)

[FigA.Q4a.1 38](#_Toc527232628)

[FigA.Q4b.1 38](#_Toc527232629)

[FigA.Q4b.2 39](#_Toc527232630)

[FigA.Q4b.3 39](#_Toc527232631)

[FigA.Q4b.4 40](#_Toc527232632)

[FigA.Q5a.1 40](#_Toc527232633)

[FigA.Q5a.2 41](#_Toc527232634)

[FigA.Q5b.1 41](#_Toc527232635)

[FigA.Q5b.2 42](#_Toc527232636)

[Part B Conclusion Figures 43](#_Toc527232637)

[FigB.Q1a 43](#_Toc527232638)

[FigB.Q2b.1 44](#_Toc527232639)

[FigB.Q2b.2 44](#_Toc527232640)

[FigB.Q3a 45](#_Toc527232641)

[FigB.Q3b 45](#_Toc527232642)

[FigB.Q3c 46](#_Toc527232643)

[FigB.Q4 46](#_Toc527232644)

[Reference 47](#_Toc527232645)

# 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.

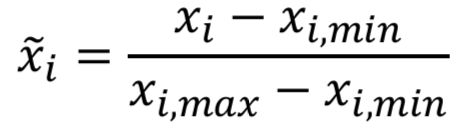
# 2. Method

## 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](#_2.3.7_Early_Stopping). 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](#_2.3.6_Selection_Criteria).

## 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:



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](#_ucu279vkxfaj).

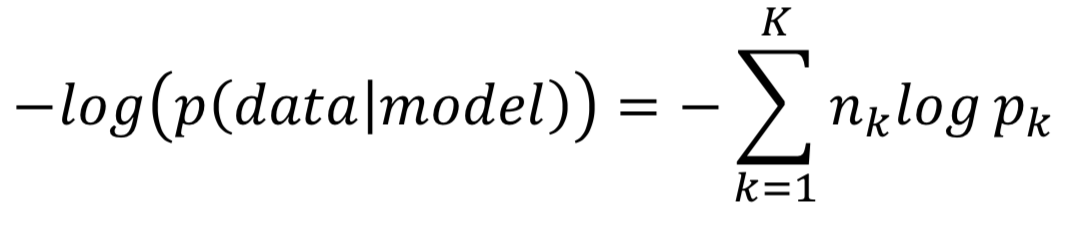
For Q5 of part A, we developed another 4-layer feedforward neural network, with the following architecture:

* 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.

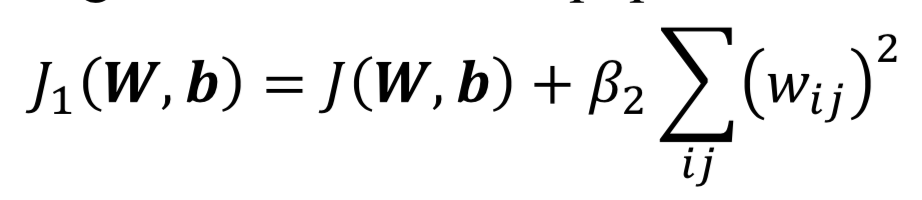
### 2.3.2 Learning Goal

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:



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:



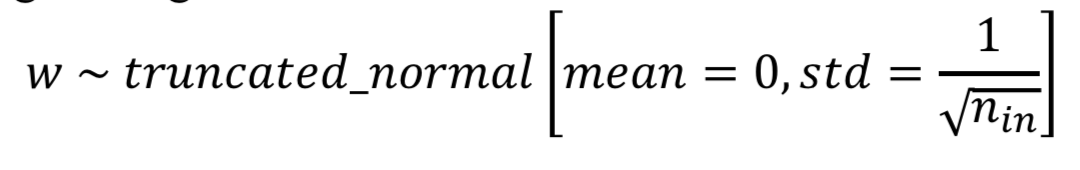
In this assignment, we had experimented with a list of different L2 regularization term, and the final results are further discussed in [Section 3.3](#_mpw93w8jnpy9).

### 2.3.3 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:



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](#_fpe53a5odnhl).

### 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:

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]

### 2.3.6 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:

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 Robustness (ability to generalize):**

In order to improve the model robustness and prevent overfitting[[1]](#footnote-1), we would limit the capacity of the neural network, by controlling:

* 1. **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 more costly 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. **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]](#footnote-2), we could foresee smoother mappings, and improve the model’s ability to generalize. Hence, the model with higher weight decay is generally better.

* 1. **Batch Size:**

Large-batch methods tend to converge to sharp minimizers of the training function, and tend to generalize less well[[3]](#footnote-3), while small-batch methods converge to flat minimizers characterized by having numerous small eigenvalues of . 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:

1. Patience (default: 20) - Number of epochs with no min\_delta improvement after which training will be stopped.
2. 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.

# 

# 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](#_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. 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](#_Classification_report_with) to further strengthen the support for the selection of optimal hyperparameter.

## 3.1 Optimal Batch Size = 4

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 | Time per Epoch (ms) | Total Time (ms) |
| **4** | **0.868** | **241.125** | **65586.071** |
| 8 | 0.856 | 125.916 | 41048.765 |
| 16 | 0.836 | 65.068 | 12883.430 |
| 32 | 0.831 | 34.806 | 10789.832 |
| 64 | 0.827 | 20.350 | 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. [Converged Test Accuracy] The converged 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](#_Classification_report_for), 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

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 | Time / Epoch (ms) | Total Time (ms) |
| 5 | 0.847 | 223.908 | 24405.936 |
| 10 | 0.868 | 241.241 | 65617.532 |
| 15 | 0.858 | 248.127 | 56324.790 |
| **20** | **0.875** | **254.776** | **58088.956** |
| 25 | 0.862 | 253.728 | 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. [Converged Test Accuracy] The converged 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

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 | Time / Epoch (ms) | Total Time (ms) |
| 0 | 0.876 | 235.092 | 53601.049 |
| 1 x 10-12 | 0.876 | 242.662 | 55327.047 |
| 1 x 10-9 | 0.875 | 243.503 | 55518.714 |
| **1 x 10-6** | **0.875** | **241.574** | **55078.953** |
| 1 x 10-3 | 0.851 | 242.892 | 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. [Converged Test Accuracy] Model with decay parameter 1 x 10-3 has the lowest test accuracy (0.851). While the converged 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 worthful. 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. 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

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 | Time / Epoch (ms) | Total Time (ms) |
| 1 | 0.831 | 36.413 | 11288 |
| **2** | **0.848** | **43.690** | **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. [Converged Test Accuracy] The model with 2 hidden layers has a higher converged 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 2 hidden layers is still robust.

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](#_3.1_Optimal_Batch) |
| Number of hidden neurons | 20 | [Refer to Section 3.2](#_3.2_Optimal_Number) |
| Decay Parameter | 1 x 10-6 | [Refer to Section 3.3](#_3.3_Optimal_Decay) |
| Number of Hidden Layers | 2 | [Refer to Section 3.4](#_3.4_Optimal_Number) |

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](#_Part_A_Conclusion) 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 |  |
|  | (refer to appendix FigA.Q2a.2 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) |
| 2(c) the total time taken to train the network against different batch sizes |  |
|  | (refer to appendix FigA.Q2c.1 for larger figure) |
| 2(c) the converged test accuracy against different batch sizes |  |
|  | (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) |
| 3(a) the test accuracy against the number of epochs |  |
|  | (refer to appendix FigA.Q3a.2 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) |
| 3(c) the total time taken to train the network against the number of epochs |  |
|  | (refer to appendix FigA.Q3c.1 for larger figure) |
| 3(c) the converged test accuracy against different number of epochs |  |
|  | (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) |
| 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) |
| 4(b) the converged test accuracy against different values of decay parameter |  |
|  | (refer to appendix FigA.Q4b.4 for larger figure) |
| 5(a) the train error of 4-layer network |  |
|  | (refer to appendix FigA.Q5a.1 for larger figure) |
| 5(b) the test accuracy of 4-layer network |  |
|  | (refer to appendix FigA.Q5a.2 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) |
| 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

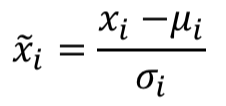
# 1. Introduction

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 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 .

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:

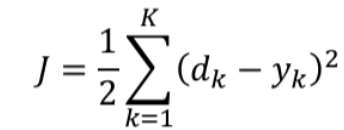
* an input layer of dimension 8 (corresponding to the input feature dimensions),
* a hidden discrete perceptron layer of *optimal\_number* perceptrons with ReLu activation function (optimal number of neurons is discussed in [Section 3.2](#_octtai3225uc)),
* a hidden discrete perceptron layers of 20 perceptrons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

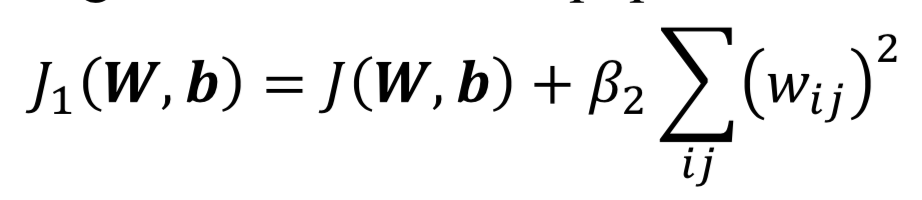
In this assignment, the above 3 neural models mentioned in 2.2.1 aim to minimize L2-regularized loss.

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.

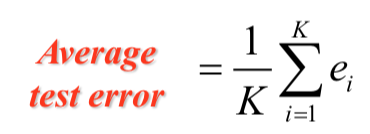
~~As validation data is also used to select in the process, the error estimate of the final model on validation data will be biased. Hence, the separation of test and validation sets is necessary. The test data is used to estimate the performance of the final model after 5-fold cross-validation. There 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) |

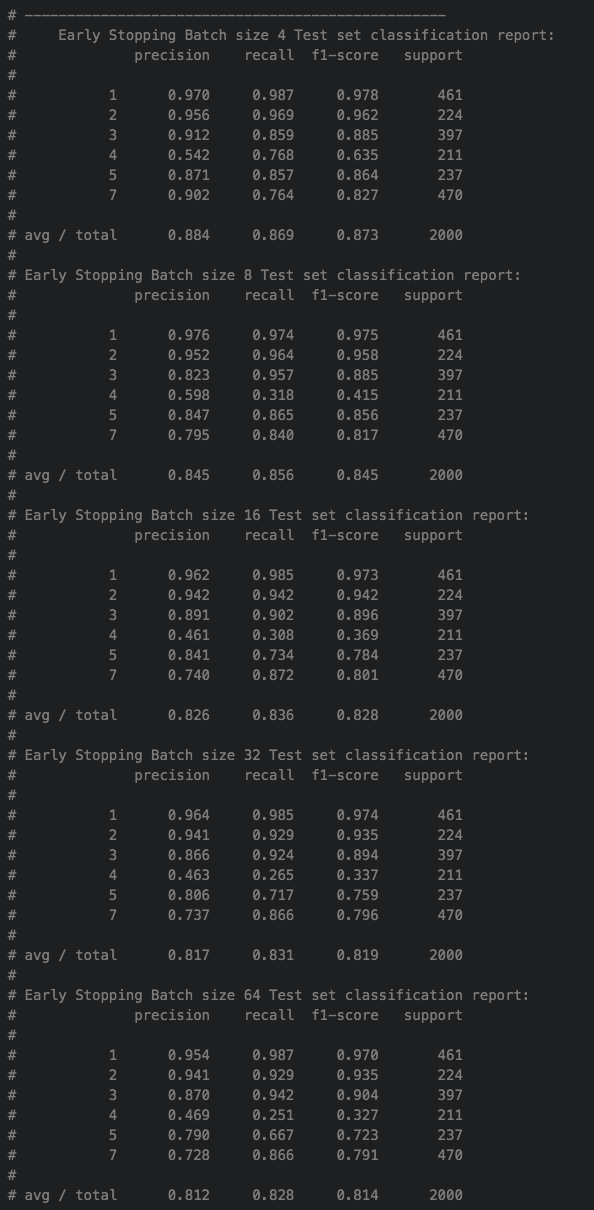
### 

## 

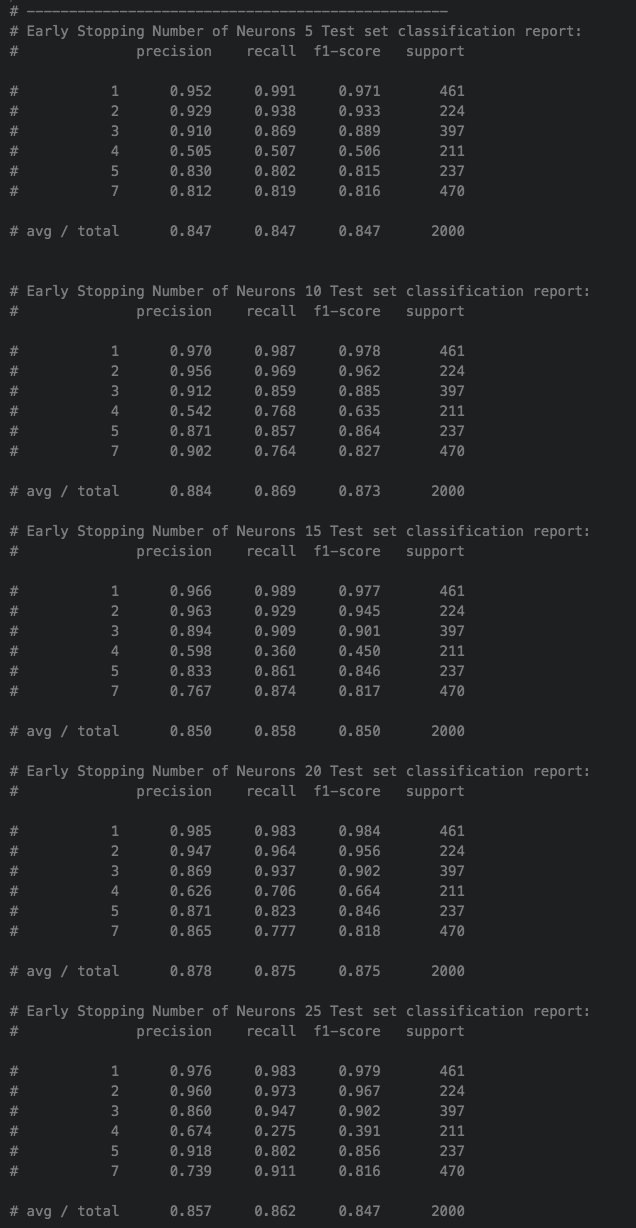
# 

# Appendix

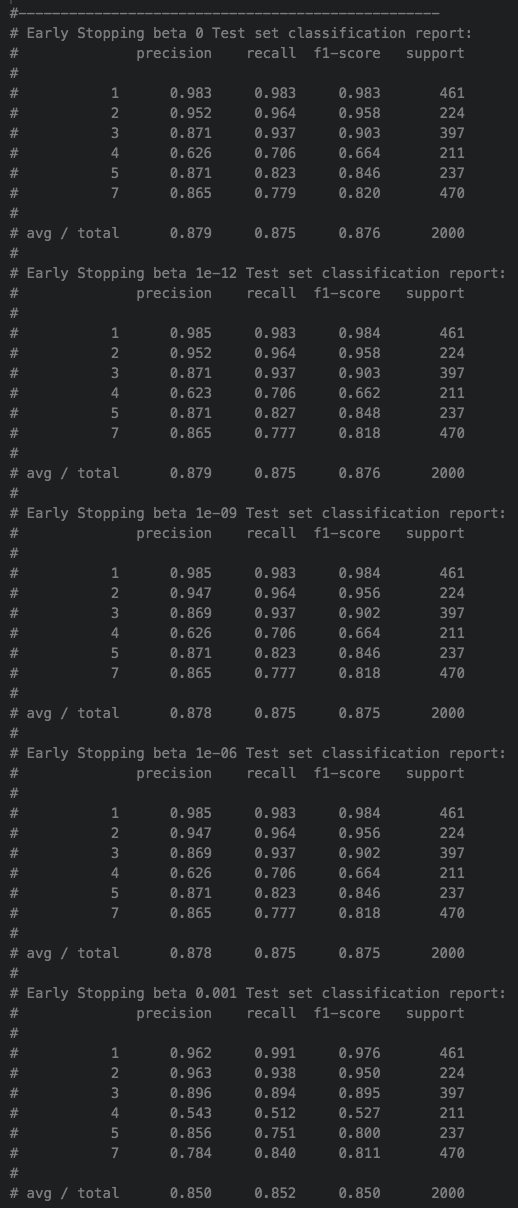
## Classification report with precision/recall/f1 score



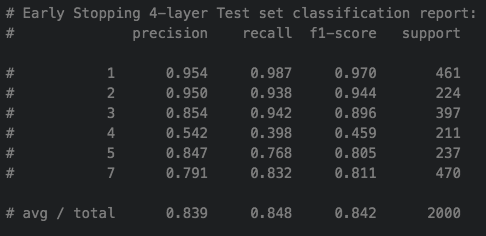
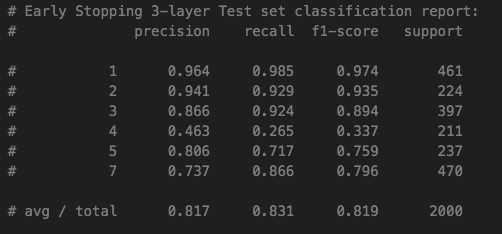
### Classification report for different batch sizes with Early Stopping



Classification report for different number of neurons with Early Stopping



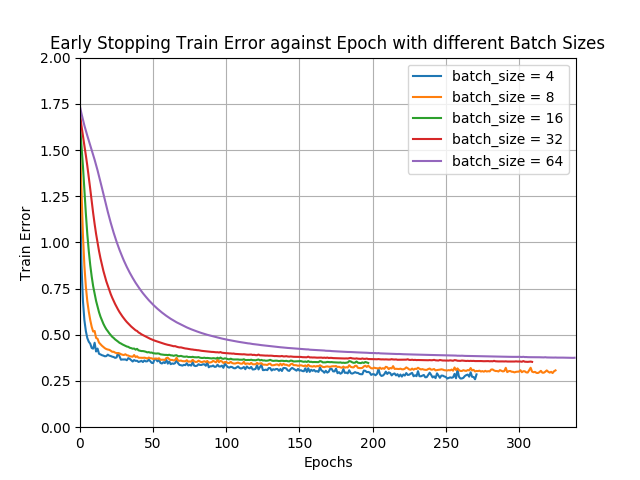
### Classification report for different L2-regularized term with early stopping



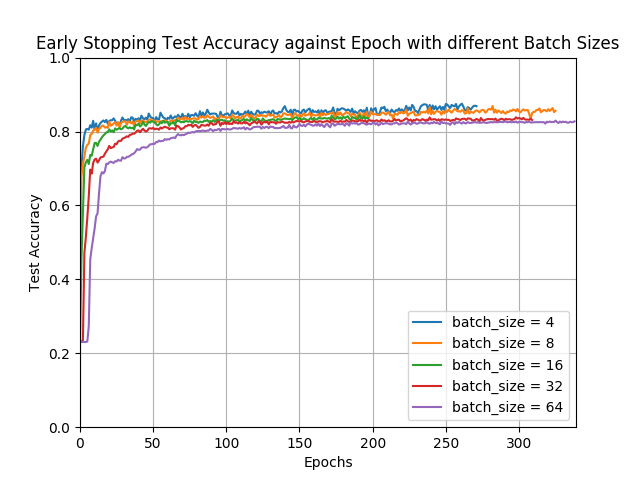
### Classification report for different number of layers with early stopping

## 

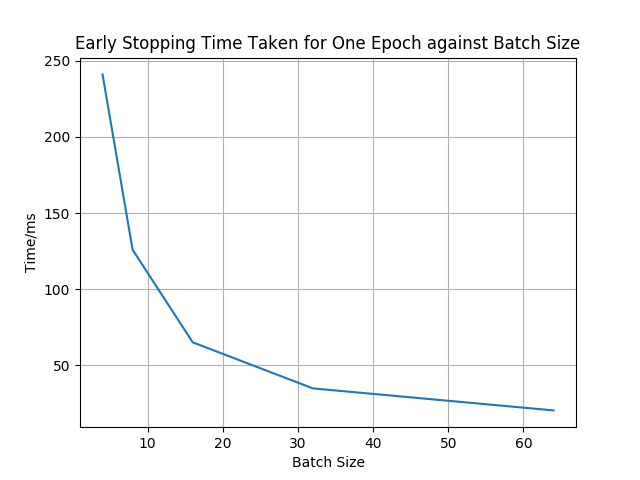
## Part A Conclusion Figures



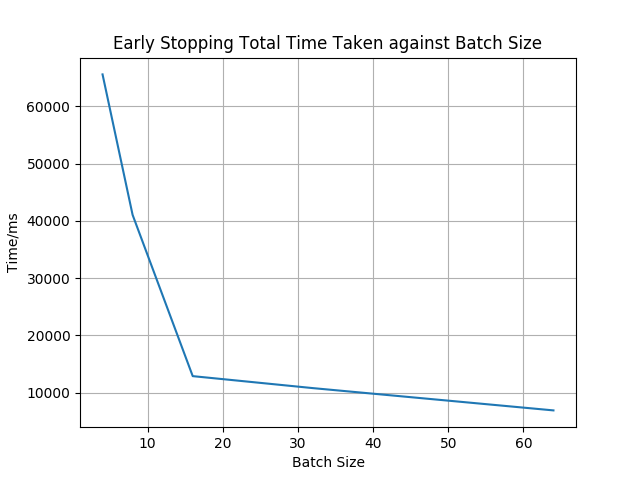
### FigA.Q2a.1



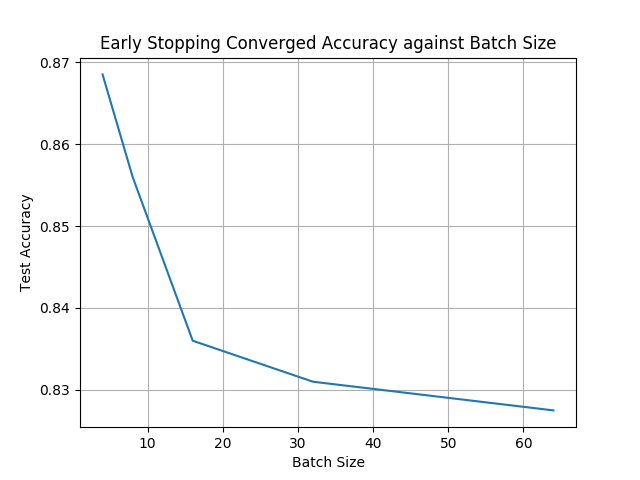
### FigA.Q2a.2



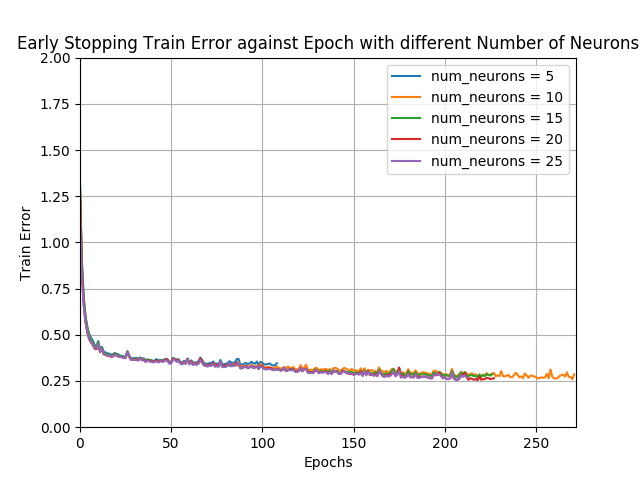
### FigA.Q2b.1



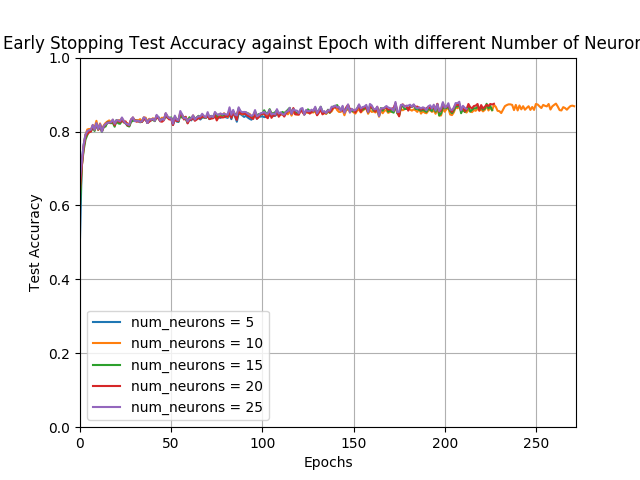
### FigA.Q2c.1



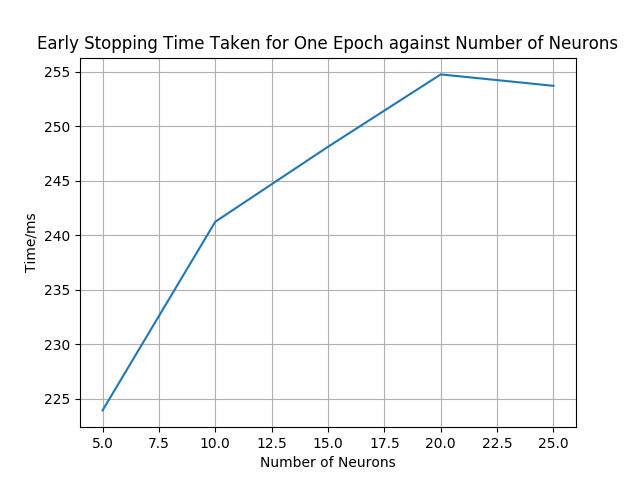
### FigA.Q2c.2



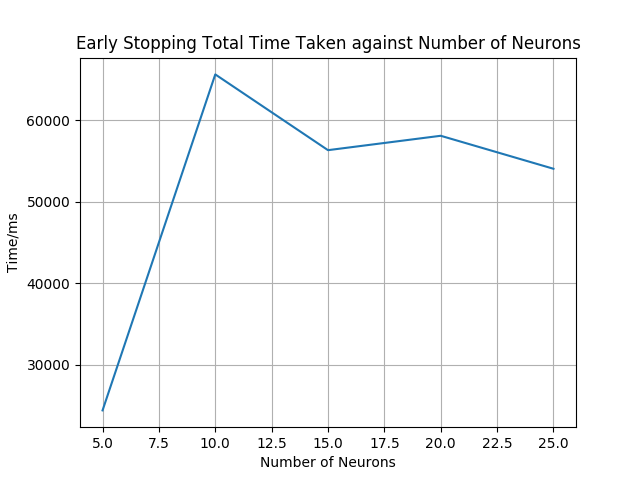
### FigA.Q3a.1



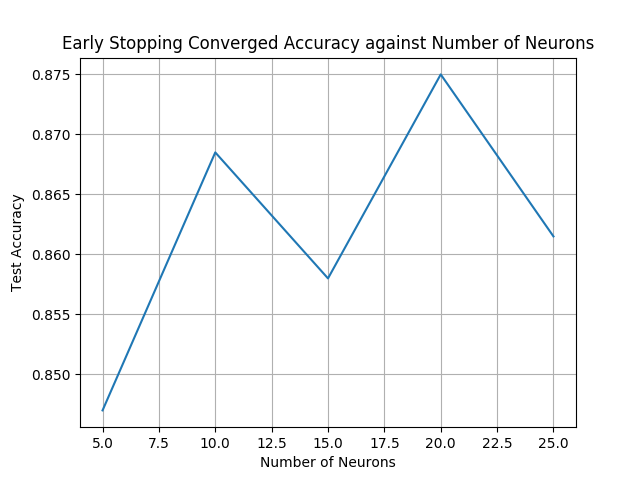
### FigA.Q3a.2



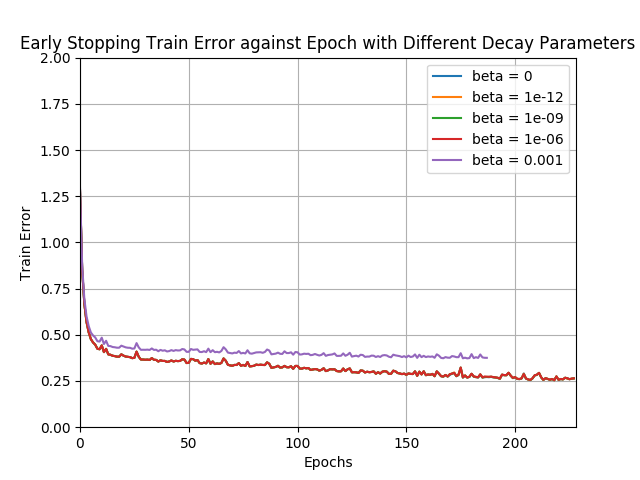
### FigA.Q3b.1



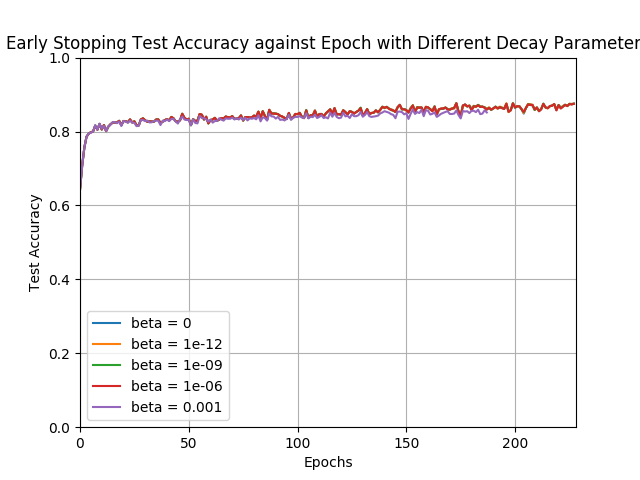
### FigA.Q3c.1



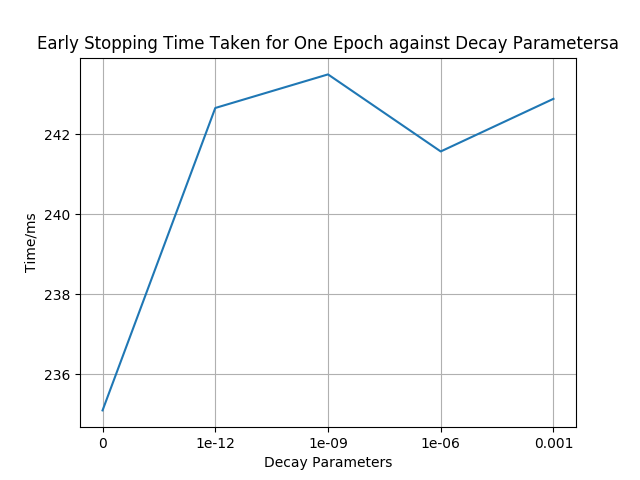
### FigA.Q3c.2



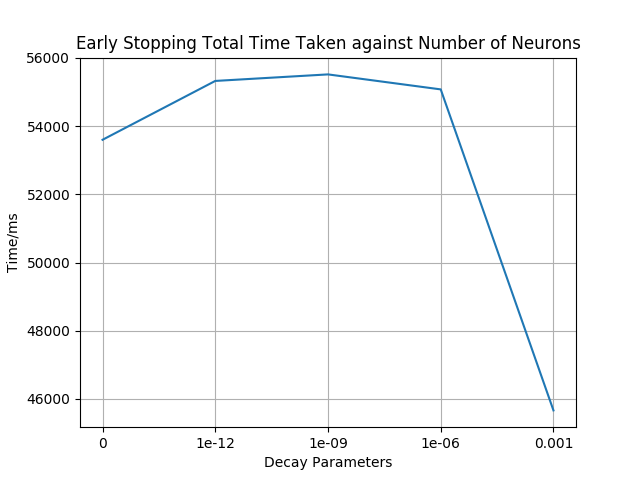
### FigA.Q4a.1



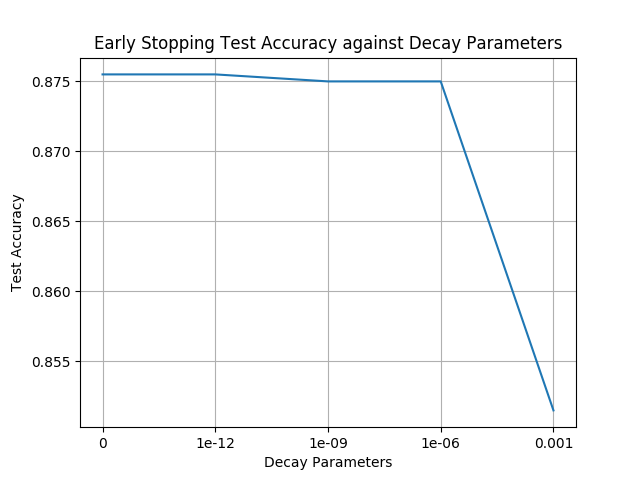
### FigA.Q4b.1



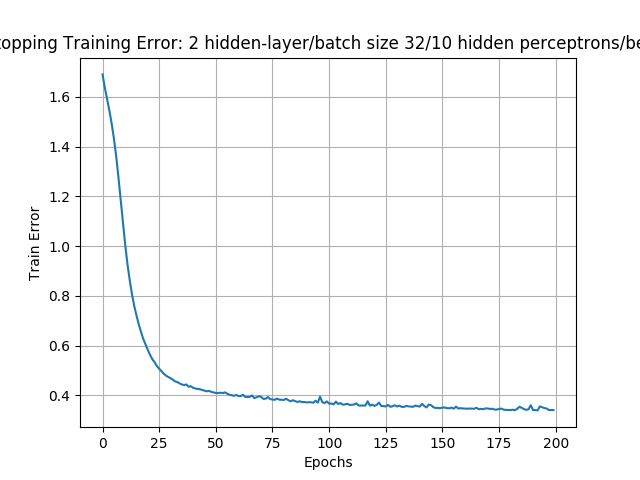
### FigA.Q4b.2



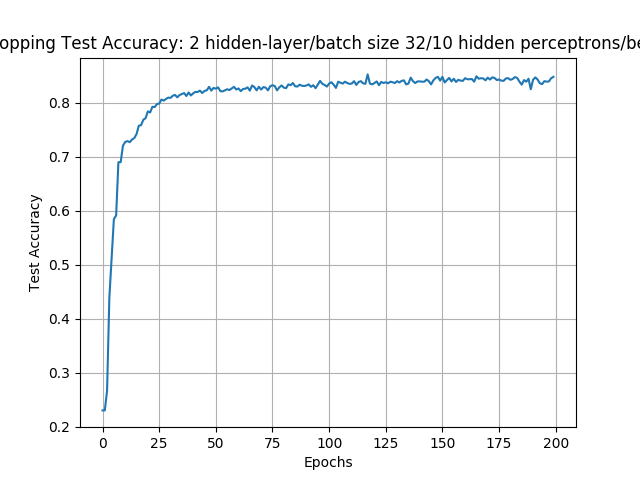
### FigA.Q4b.3



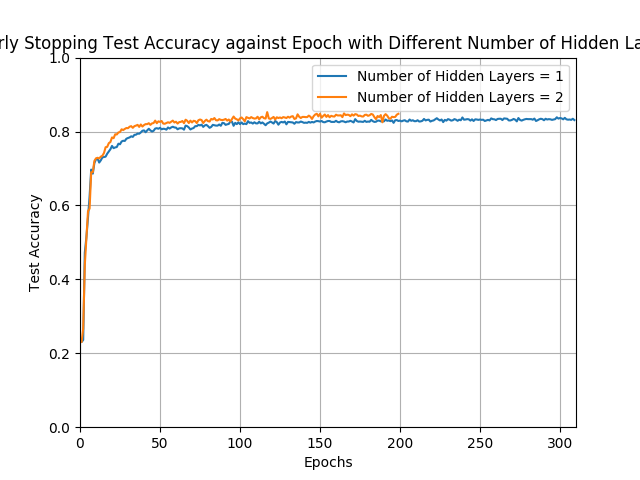
### FigA.Q4b.4



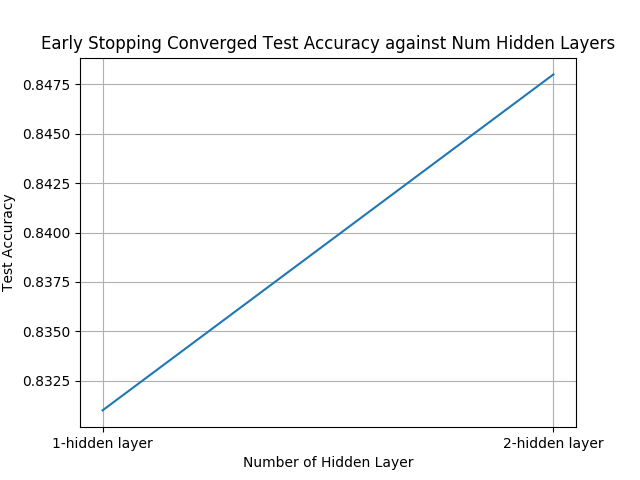
### FigA.Q5a.1



### FigA.Q5a.2



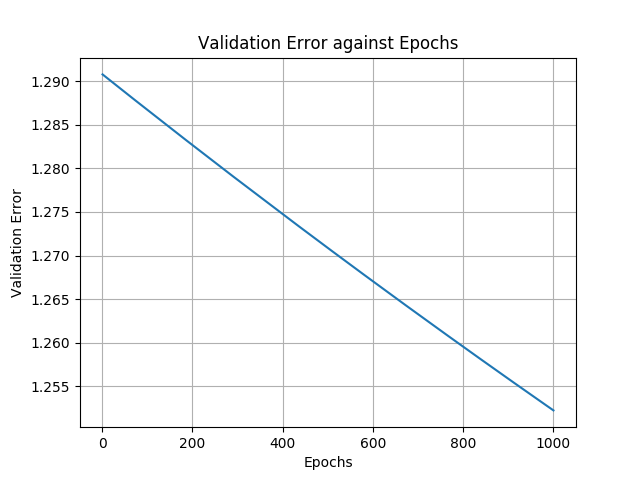
### FigA.Q5b.1



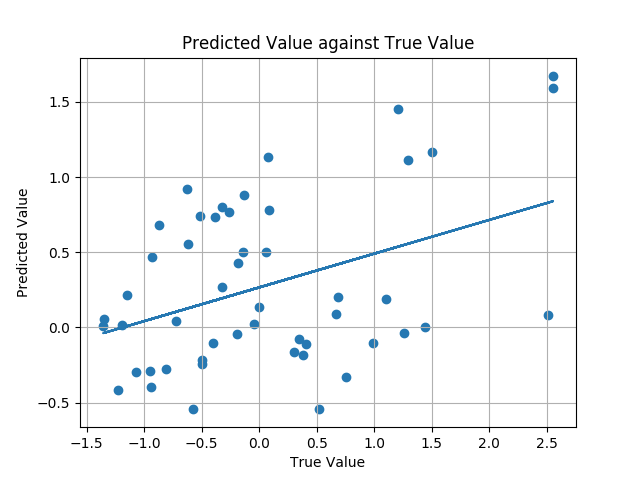
### FigA.Q5b.2

## 

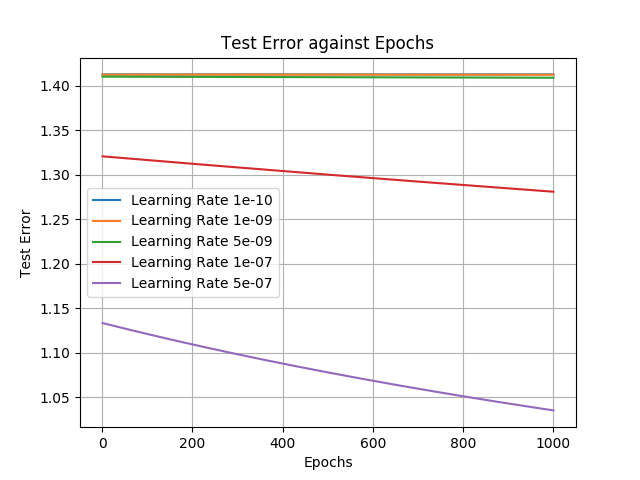
## Part B Conclusion Figures



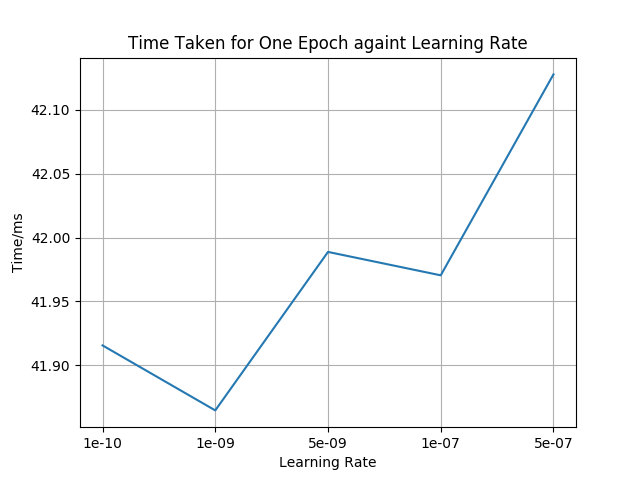
### FigB.Q1a



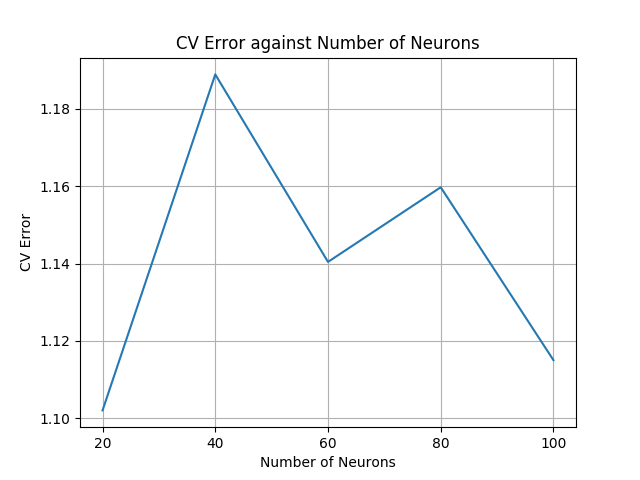
FigB.Q2a



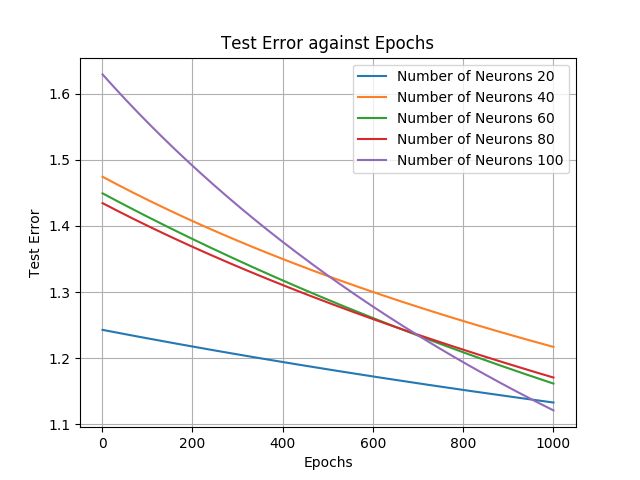
### FigB.Q2b.1



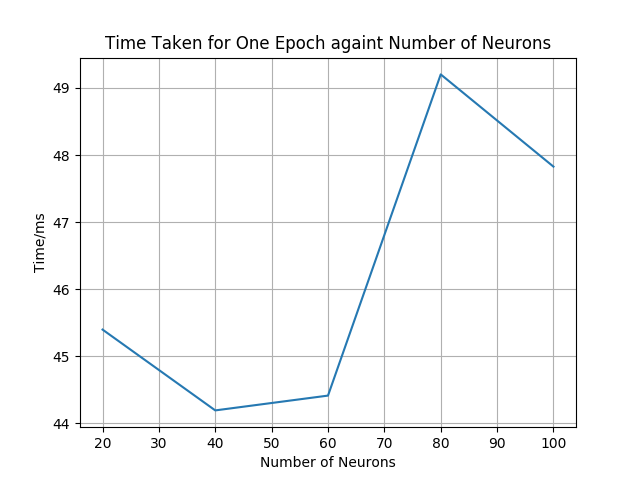
### FigB.Q2b.2



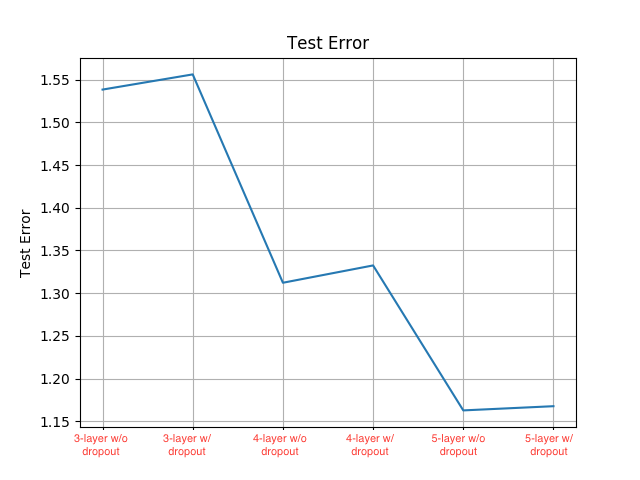
### FigB.Q3a



### FigB.Q3b



### FigB.Q3c



### FigB.Q4

# 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>

1. https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture\_slides\_lec9.pdf [↑](#footnote-ref-1)
2. http://www.cs.bham.ac.uk/~jxb/INC/l10.pdf [↑](#footnote-ref-2)
3. https://openreview.net/pdf?id=H1oyRlYgg [↑](#footnote-ref-3)