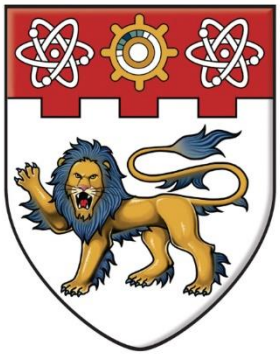


NANYANG TECHNOLOGICAL UNIVERSITY

# CZ 4042 Neural Networks & Deep Learning

Assignment 1

**Phua Jia Sheng**

---

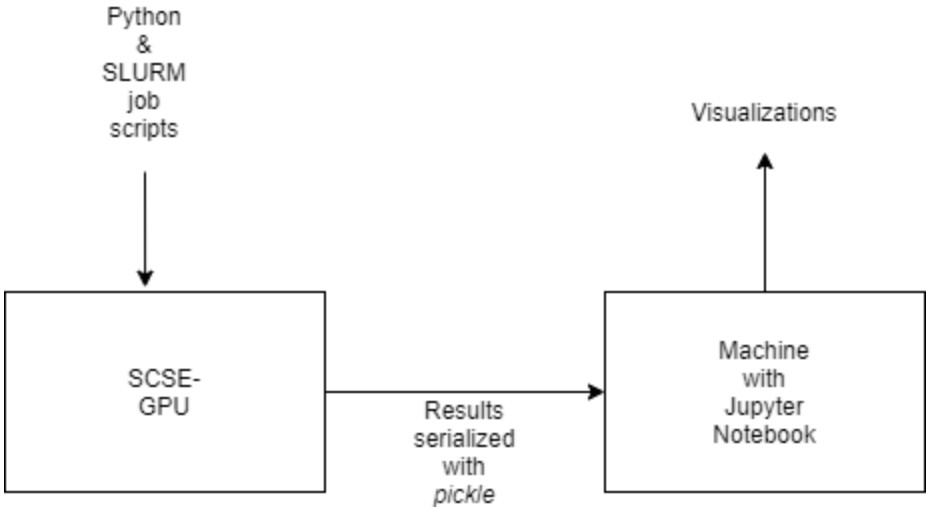
**October 2020**

# 1 Introduction

We are given 2 problems, a classification problem to classify the Cardiotocography dataset containing measurements of fetal heart rate (FHR) and uterine contraction (UC) features on cardiotocograms, and a regression problem to predict the probability of University admission based on some parameters.

# 2 Methods

SCSE GPU cluster has GPUs, which are optimized for training a neural network, compared to CPUs. However, the drawback is that it does not have interactive python, which is necessary for visualizing the training data, such as time taken and accuracy. The following workflow makes use of *pickle* to serialize and pass the training data to a machine capable of running Jupyter Notebook, which in this case is my personal machine. Another benefit of running the visualization on my home machine is eliminating the latency that comes with connecting to another machine remotely, especially since GUI is required for visualization.



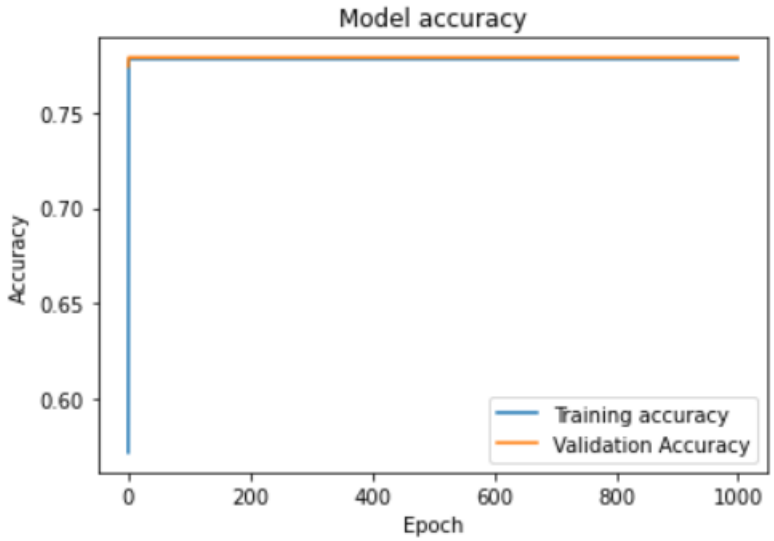

# 3 Experiments and Results

## 3.1 Classification Problem

Read the data from the file: ctg_data_cleaned.csv. Each data sample is a row of 23 values: 21 input attributes and 2 class labels (use the NSP label with values 1, 2 and 3 and ignore the other). First, divide the dataset in 70:30 ratio for training and testing. Use 5-fold cross-validation on the training dataset for selecting the optimal model, and test it on the testing data.

### 3.1.1 Design a feedforward neural network which consists of an input layer, one hidden layer of 10 neurons with ReLU activation function, and an output softmax layer. Assume a learning rate $\alpha$ = 0.01, L2 regularization with weight decay parameter $\beta$ = 10−6 , and batch size = 32. Use appropriate scaling of input features.

a) Use the training dataset to train the model and plot accuracies on training and testing data against training epochs.

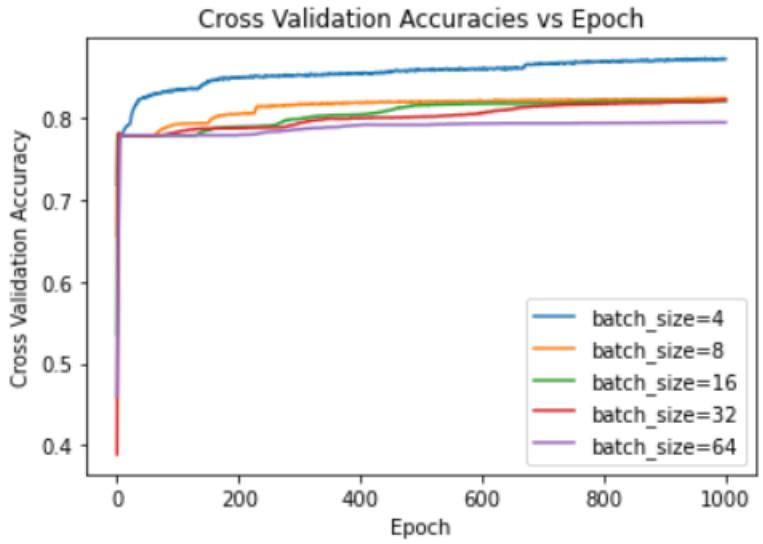


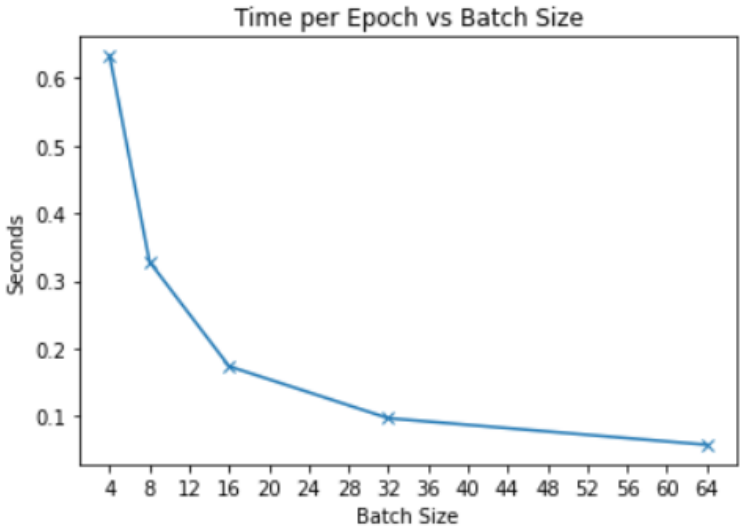

b) State the approximate number of epochs where the test error begin to converge.

Test error begins to converge around **epoch 2**.

### 3.1.2 Find the optimal batch size by training the neural network and evaluating the performances for different batch sizes.

a) Plot cross-validation accuracies against the number of epochs for different batch sizes. Limit search space to batch sizes {4,8,16,32,64}. Plot the time taken to train the network for one epoch against different batch size.as



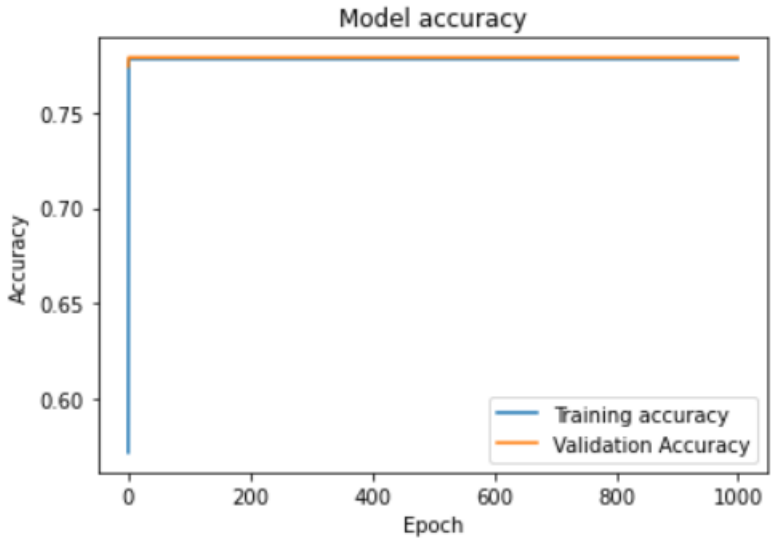




b) Select the optimal batch size and state reasons for your selection.

| | Batch size | Accuracy divided by time | Final Epoch Accuracy | Time Per Epoch (seconds) | Total Time(5-Fold Cross Validation) |
|---|---|---|---|---|---|
| **0** | 4 | 1.378281 | 0.871487 | 0.632300 | 0 days 00:52:41.500000 |
| **1** | 8 | 2.507241 | 0.823398 | 0.328408 | 0 days 00:27:22.040000 |
| **2** | 16 | 4.716913 | 0.819988 | 0.173840 | 0 days 00:14:29.200000 |
| **3** | 32 | 8.458746 | 0.821987 | 0.097176 | 0 days 00:08:05.880000 |
| **4** | 64 | 13.700384 | 0.794239 | 0.057972 | 0 days 00:04:49.860000 |

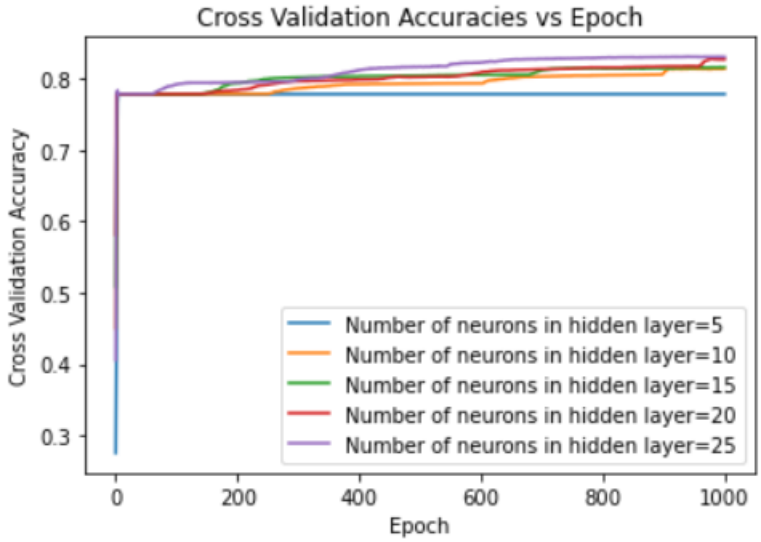
If I were to select the optimal batch size based solely on accuracy, I would go with a batch size of 4. However, that is too costly in terms of total time needed, which would be 52:41 (mm:ss) for 5-fold cross-validation. If the condition for optimal batch size was to be accuracy/time, I would go with a batch size of 64. However, the final epoch accuracy is too low at 0.794. Looking at batch sizes of 8,16 and 32, they all have similar final epoch accuracies of about 0.82, with batch size 32 having the lowest time per epoch cost of 0.0971, and a very tolerable total training time of 08:05. Therefore, I will choose **32** as the optimal batch sizes.

c) Plot the train and test accuracies against epochs for the optimal batch size.




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

a) Plot the cross-validation accuracies against the number of epochs for different number of hidden-layer neurons. Limit the search space of number of neurons to {5,10,15,20,25}.



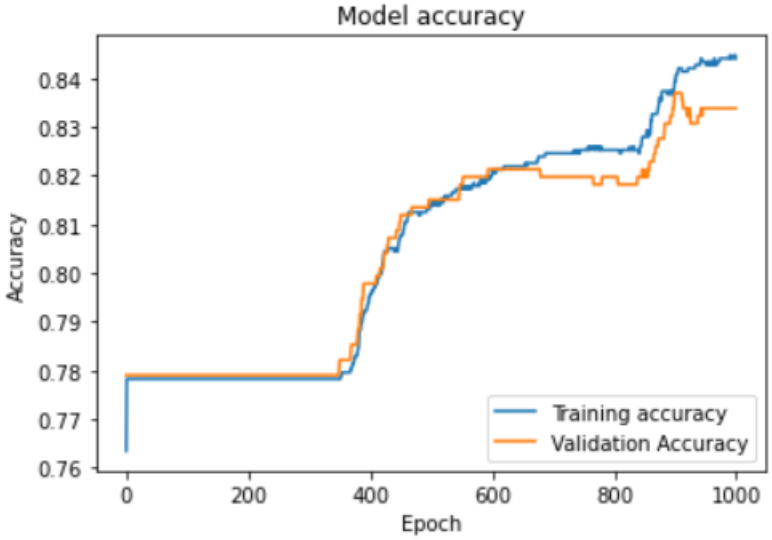

b)

| | Number of Neurons in Hidden Layer | Accuracy divided by time | Final Epoch Accuracy | Time Per Epoch (seconds) | Total Time(5-Fold Cross Validation) |
|---|---|---|---|---|---|
| 0 | 5 | 7.735788 | 0.778483 | 0.100634 | 0 days 00:08:23.170000 |
| 1 | 10 | 8.145386 | 0.813757 | 0.099904 | 0 days 00:08:19.520000 |
| 2 | 15 | 8.140603 | 0.816226 | 0.100266 | 0 days 00:08:21.330000 |
| 3 | 20 | 8.225027 | 0.827043 | 0.100552 | 0 days 00:08:22.760000 |
| 4 | 25 | 8.283668 | 0.830570 | 0.100266 | 0 days 00:08:21.330000 |

The model with 25 neurons in the hidden layer has the best performance, with a final epoch accuracy of 0.831. The model with 5 neurons in the hidden layer did significantly worse than the other models. Since there is no significant time penalty in increasing the number of

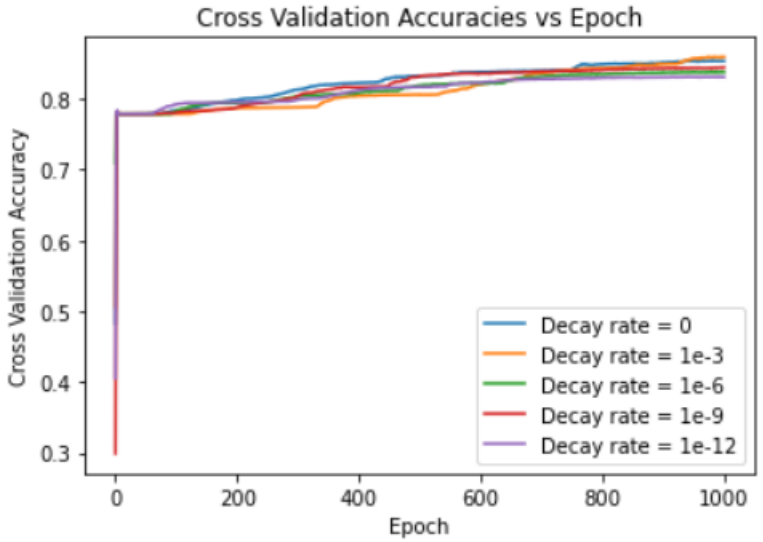
neurons in the hidden layer, I will choose the model with the best performance, which is the model with **25** neurons in the hidden layer.

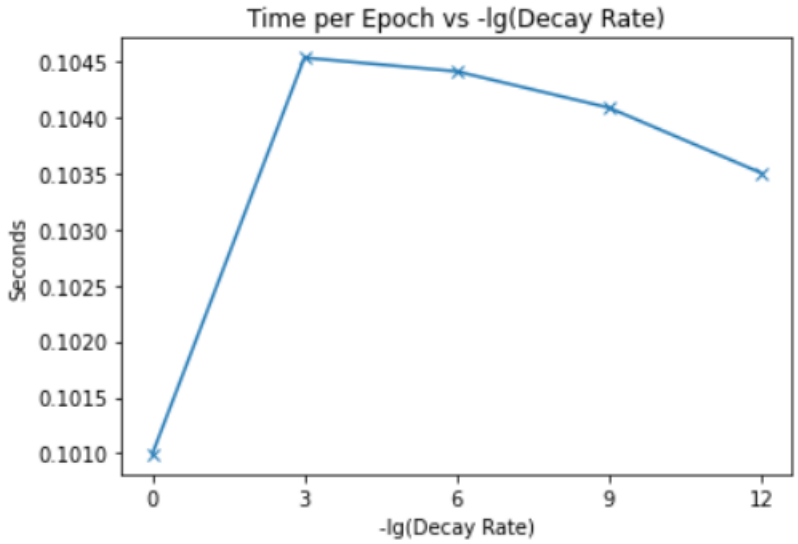
c) Plot the train and test accuracies against epochs with the optimal number of neurons.




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

a) Plot cross-validation accuracies against the number of epochs for the 3-layer network for different values of decay parameters. Limit the search space of decay parameters to {0, 10−3 , 10−6 , 10−9 , 10−12}.



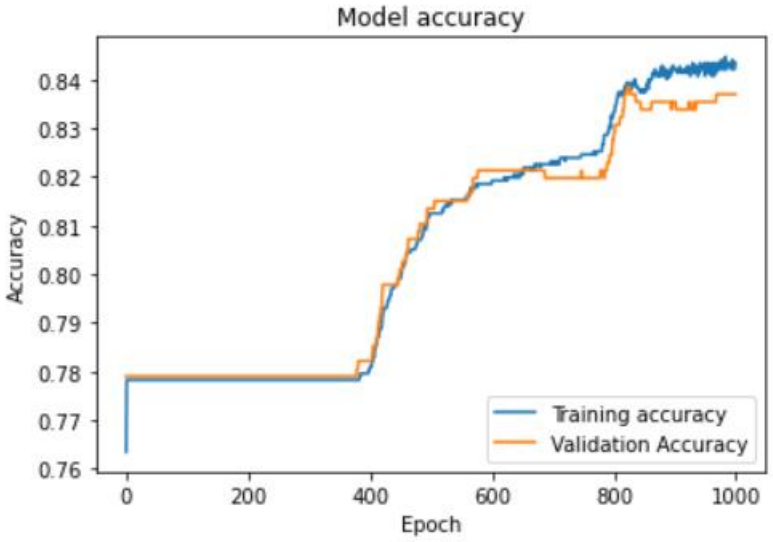



b) Select the optimal decay parameter. State the rationale for your selection.

| | Decay rate | Accuracy divided by time | Final Epoch Accuracy | Time Per Epoch (seconds) | Total Time(5-Fold Cross Validation) |
|---|---|---|---|---|---|
| **0** | 0 | 8.447651 | 0.853145 | 0.100992 | 0 days 00:08:24.960000 |
| **1** | 1e-3 | 8.216842 | 0.858907 | 0.104530 | 0 days 00:08:42.650000 |
| **2** | 1e-6 | 8.020360 | 0.837390 | 0.104408 | 0 days 00:08:42.040000 |
| **3** | 1e-9 | 8.110846 | 0.844209 | 0.104084 | 0 days 00:08:40.420000 |
| **4** | 1e-12 | 8.024678 | 0.830570 | 0.103502 | 0 days 00:08:37.510000 |

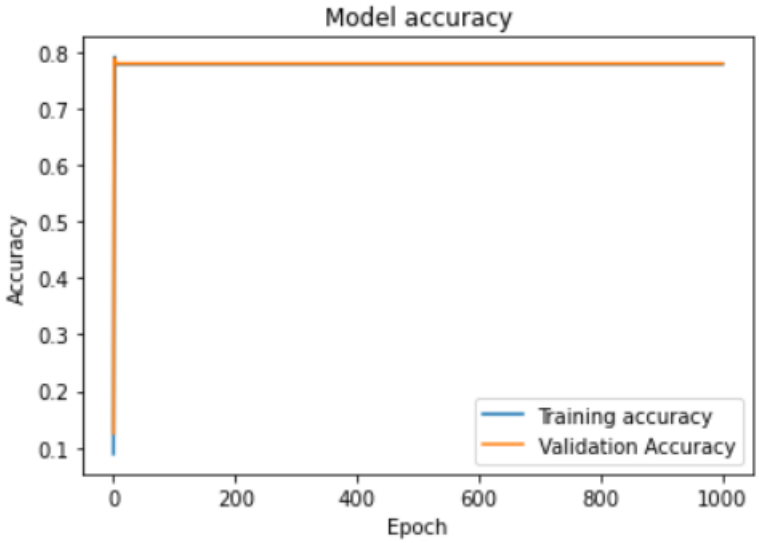
We select the decay parameter that gives the highest final epoch accuracy, **1e-3**.

c) Plot the train and test accuracies against epochs for the optimal decay parameter.




3.1.5 After you are done with the 3-layer network, design a 4-layer network with two hiddenlayers, each consisting 10 neurons, and train it with a batch size of 32 and decay parameter 10-6 .

a)




b) Compare and comment on the performances of the optimal 3-layer and 4-layer networks.

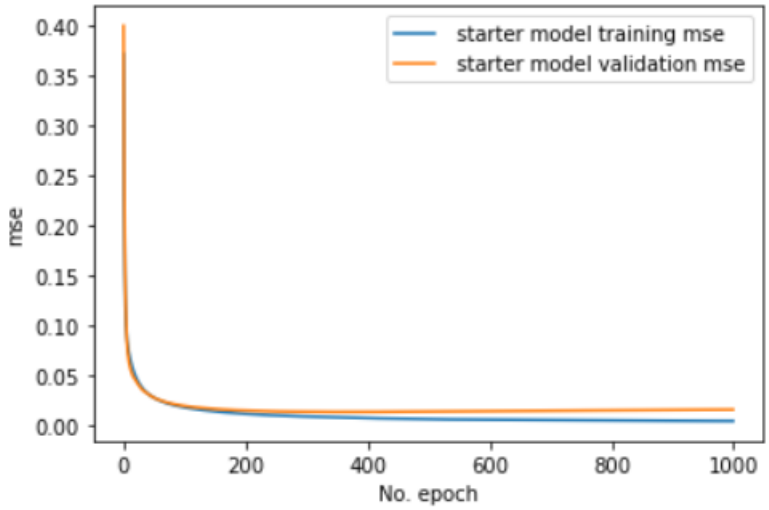
The un-optimized 3-layer and 4-layer networks have identical performances (same final epoch accuracy of 0.779), but the optimized 3-layer network has the best performance (final epoch testing accuracy of more than 0.83). The testing accuracy of the un-optimized 3-layer network stabilized at epoch 2, while the testing accuracy of the un-optimized 4-layer network stabilized at epoch 5.

## 3.2 Regression Problem

This assignment uses the data from the Graduate Admissions Predication [2]. The dataset contains several parameters like GRE score (out of 340), TOEFL score (out of 120), university Rating (out of 5), strengths of Statement of Purpose and Letter of Recommendation (out of 5), undergraduate GPA (out of 10), research experience (either 0 or 1), that are considered important during the application for Master Programs. The predicted parameter is the chance of getting an admit (ranging from 0 to 1). You can obtain the data from: https://www.kaggle.com/mohansacharya/graduate-admissions or from file 'admission_predict.csv'. Each data sample is a row of 9 values: 1 serial number (ignore), 7 input attributes and the probability of getting an admit as targets. Divide the dataset at 70:30 ratio for training and testing.

### 3.2.1 Design a 3-layer feedforward neural network consists of an input layer, a hidden-layer of 10 neurons having ReLU activation functions, and a linear output layer. Use mini-batch gradient descent with a batch size = 8, $L2$ regularization at weight decay parameter $\beta$ = 10−3 and a learning rate $\alpha$ = 10−3 to train the network.

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



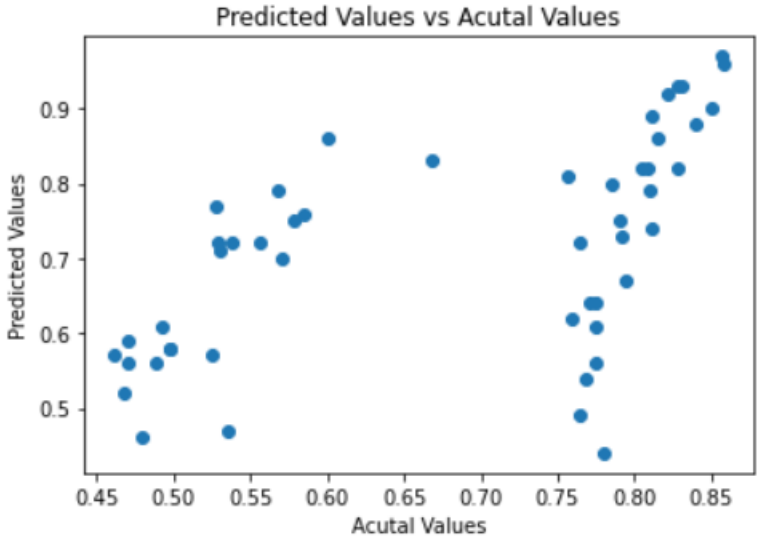

b) State the approximate number of epochs where the test error is minimum and use it to stop training.

```
np.argmin(histories['starter'].history['val_mse'])+1
```

```
371
```

Test error is a minimum at **epoch 371**.

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



3.2.2 Recursive feature elimination (RFE) is a feature selection method that removes unnecessary features from the inputs. Start by removing one input feature that causes the minimum drop (or maximum improvement) in performance. Repeat the procedure recursively on the reduced input set until the optimal number of input features is reached. Remove the features one at a time. Compare the accuracy of the model with all input features, with models using 6 input features and 5 input features selected using RFE. Comment on the observations.

*Round 1: Creating a 6-input model by removing 1 variable from the base model at a time*

Base MSE (no variables removed): 0.016361317

| Index | Variable Removed | Validation MSE at epoch 371 |
|---|---|---|
| **0** | GRE Score | 0.013298033 |
| **1** | TOEFL Score | 0.016361317 |
| **2** | University Rating | 0.018793227 |
| **3** | SOP | 0.01271763 |
| **4** | LOR | 0.01194668 |
| **5** | CGPA | 0.019737046 |
| **6** | Research | 0.0135234045 |

Removing LOR gives the greatest improvement in MSE, so we do that and move on to remove the second variable.

*Round 2: Creating a 5-input model by removing 1 variable from the optimal 6-input model at a time*

Optimal 6-input model MSE: 0.01194668

| Index | Variable Removed | Validation MSE at epoch 371 |
|---|---|---|
| **0** | GRE Score | 0.010364646 |
| **1** | TOEFL Score | 0.012940791 |
| **2** | University Rating | 0.0129352845 |
| **3** | SOP | 0.012305949 |
| **4** | CGPA | 0.020544486 |
| **5** | Research | 0.023111966 |

Removing GRE gives the greatest improvement in MSE, so we do that and move on to remove the second variable.

*Round 3: Creating a 4-input model by removing 1 variable from the optimal 5-input model at a time*

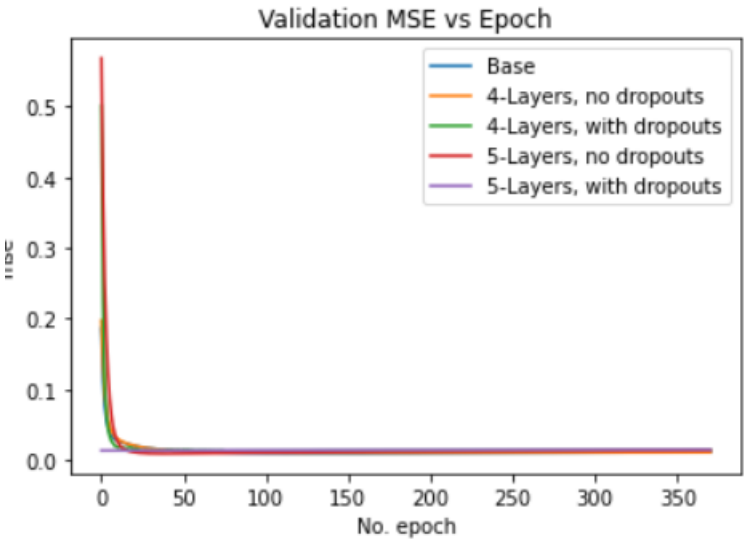
Optimal 5-input model MSE: 0.010364646

| Index | Variable Removed | Validation MSE at epoch 371 |
|---|---|---|
| **0** | TOEFL Score | 0.013608102 |
| **1** | University Rating | 0.015873507 |
| **2** | SOP | 0.012564289 |
| **3** | CGPA | 0.019414656 |
| **4** | Research | 0.012371924 |

None of the attributes give a better MSE when removed, so we conclude that the optimal number of inputs is 5, with the 5 most relevant inputs listed in the table above.

3.2.3 Design a four-layer neural network and a five-layer neural network, with the hidden layers having 50 neurons each. Use a learning rate of 10-3 for all layers and optimal feature set selected in part (3). Introduce dropouts (with a keep probability of 0.8) to the layers and report the accuracies. Compare the performances of all the networks (with and without dropouts) with each other and with the 3-layer network.

| | Model | Final Epoch MSE |
|---|---|---|
| **0** | Base | 0.011608 |
| **1** | 4-Layers, no dropouts | 0.011122 |
| **2** | 4-Layers, with dropouts | 0.014179 |
| **3** | 5-Layers, no dropouts | 0.013599 |
| **4** | 5-Layers, with dropouts | 0.014855 |




Optimal model: **4-layers, no dropouts**.

The introduction of dropouts did not improve the accuracy of the model, likely due to the fact that there is not much over-fitting present, since an optimal number of epochs (371) which is derived from 3.2.1 is used, which stops the training before over-fitting occurs.