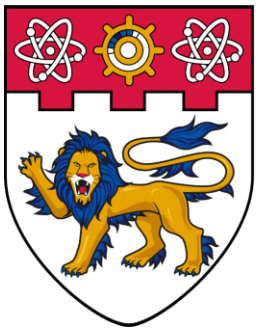


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4042 NEURAL NETWORK & DEEP LEARNING

ASSIGNMENT 1

ASSIGNMENT REPORT

Shearman Chua Wei Jie (U1820058D)

LAB GROUP: CS4

Part A: Classification Problem

This project aims at building neural networks to classify the Cardiotocography dataset containing measurements of fetal heart rate (FHR) and uterine contraction (UC) features on 2126 cardiotocograms classified by expert obstetricians.

Question 1

For question 1, we first obtain the dataset from the ctg_data_cleaned.csv and normalize the data using the **scale()** function. The data is then split into X, the input features, and Y, the labels to perform the classification task on. The data is then divided in 70:30 ratio for training and testing. A seed is used for the random_state of the train_test_split() function to ensure that we get the same shuffled dataset all the time, and that we get the same classification results all the time when the code is run.

```
# create training and testing vars
X_train, X_test, Y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=seed)
print (X_train.shape, Y_train.shape)
print (X_test.shape, y_test.shape)
```

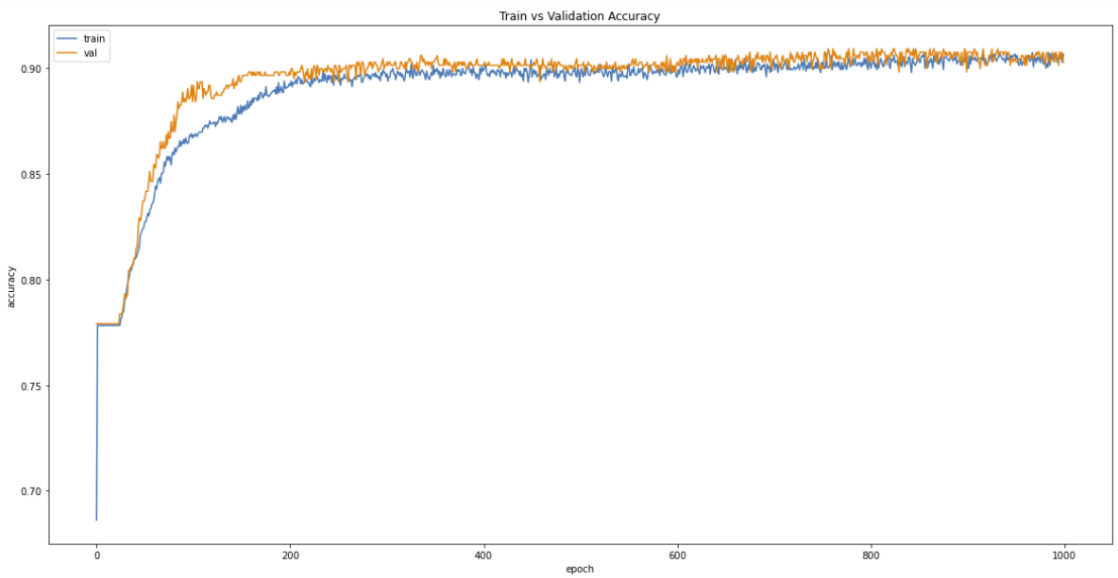
```
(1488, 21) (1488,)
(638, 21) (638,)
```

a) For this part of the question, we first create a sequential neural network that is composed of 3 layers, the input layer, a hidden layer of 10 neurons with ReLU activation function, and an output softmax layer. We then instantiate an optimizer with learning rate α=0.01, L2 regularization with weight decay parameter β=10−6.

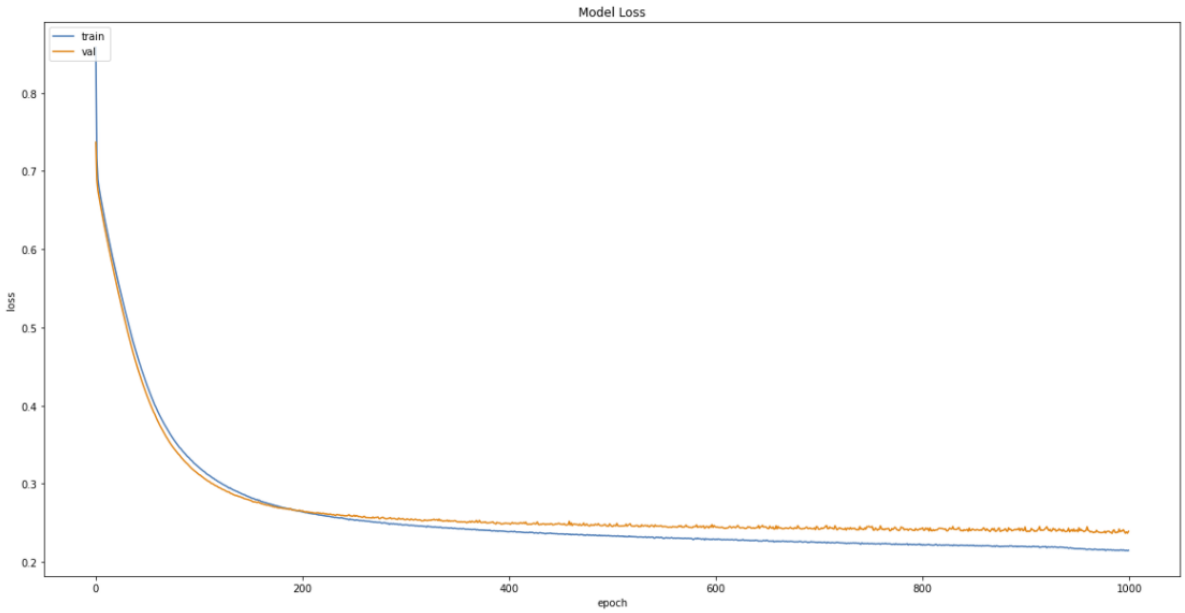
```
l2 = tf.keras.regularizers.l2(l2=1e-6)

model = Sequential([
    Dense(num_neurons, activation='relu', kernel_regularizer=l2),
    Dense(NUM_CLASSES, activation='softmax')
])

opt = tf.keras.optimizers.SGD(lr=0.01)

model.compile(optimizer=opt,
              loss= 'sparse_categorical_crossentropy',
              metrics=['accuracy'])

# train the model
histories['qn1'] =model.fit(X_train, Y_train,
                            epochs=epochs,
                            verbose = 2,
                            batch_size=batch_size,
                            validation_data=(X_test, y_test))
```

The model was trained for 1000 epochs. Using the training accuracies and test accuracies for each epoch, we plot the graph of accuracies on training and testing data against training epochs. From the graph, we can observe that the training and test accuracies start to stabilize after 200 epochs and the training accuracy starts to catch up with the test accuracy at around 1000 epochs. Training the model after 1000 epochs will cause the model to overfit on the training data. The test accuracy obtained for the final epoch was 0.9028 as shown below.

```
Epoch 1000/1000
47/47 - 0s - loss: 0.2147 - accuracy: 0.9066 - val_loss: 0.2391 - val_accuracy: 0.9028
```



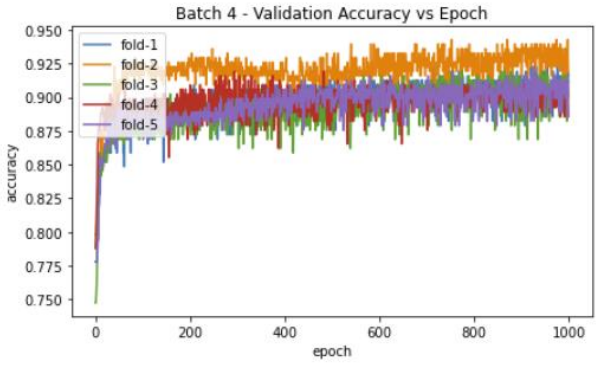
b) For the graph below, the model's training and test losses were plotted against the training epochs. From the graph, we can observe that the test loss of the model starts to converge at about 200 epochs, with significantly less improvement of test loss after 200 epochs.

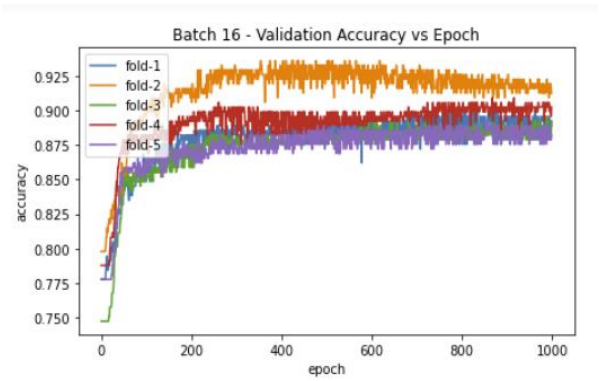


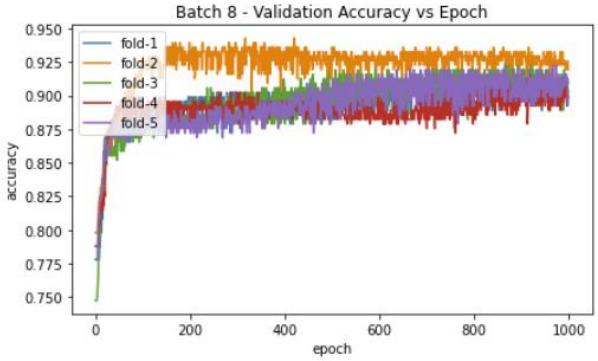
Question 2

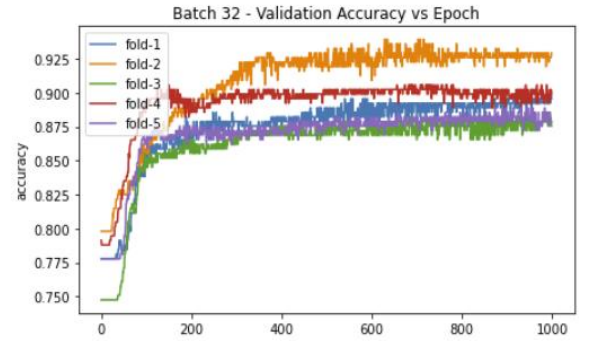
a) Plot cross-validation accuracies against the number of epochs for different batch sizes.

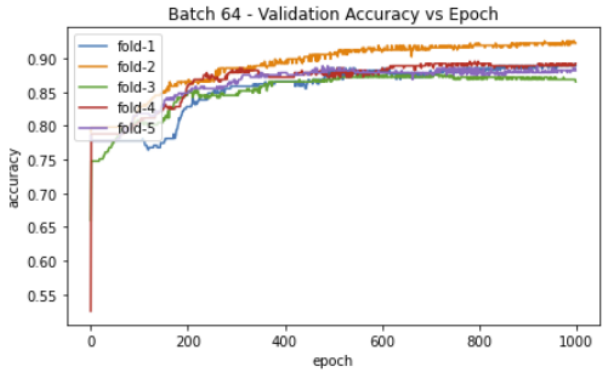
Below are the plots of the validation accuracy against epoch for each of the 5 folds during training for each of the batch size that are to be tested on. From the graphs, we can see that for bigger batch sizes, the validation accuracy plots for each of the fold is smoother and more convergent. The amount of "jitter" in the validation curve will depend to some extend on minibatch size; larger minibatches will result in a smoother curve, while smaller minibatches will result in more "jitter". Using larger minibatch sizes can however under some circumstances contribute to overfitting.





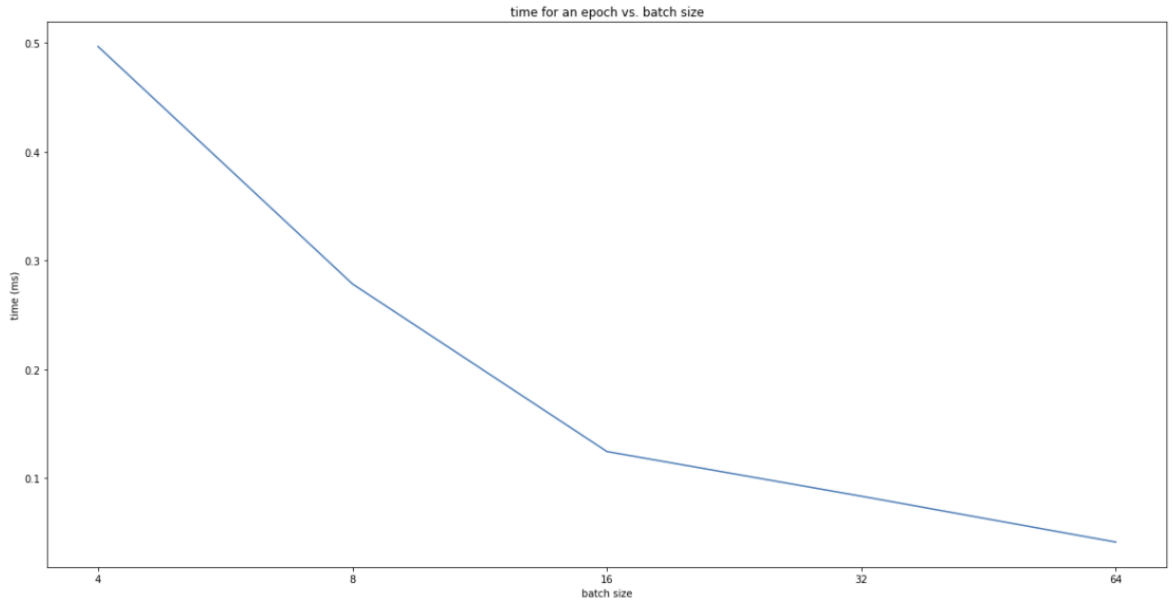






Plot the time taken to train the network for one epoch against different batch sizes.

From the graph below, we can observe that a smaller batch size will result in a longer time for each epoch which is intuitive as a smaller batch size means that more batches will have to be run for each epoch, resulting in a longer time taken.



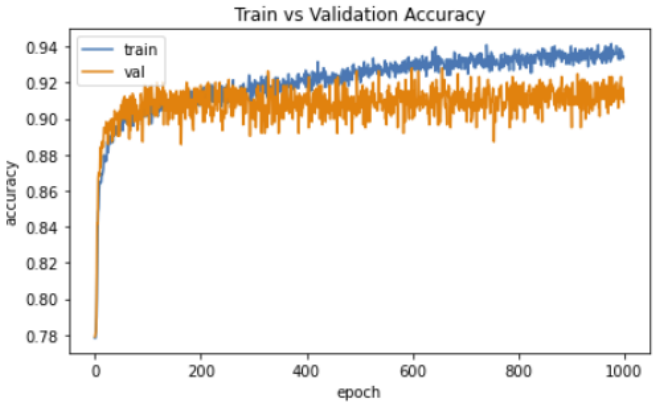
b) Select the optimal batch size and state reasons for your selection.

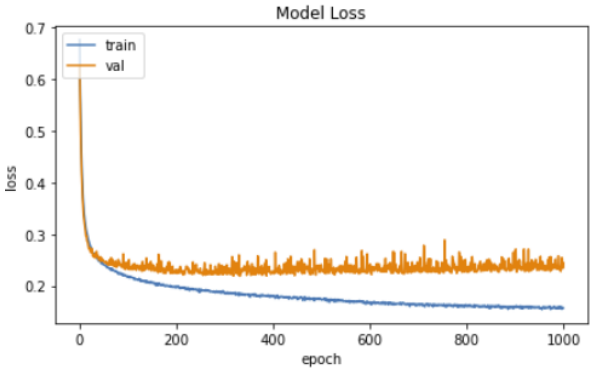
For each batch size, we first create a sequential neural network that is composed of 3 layers, the input layer, a hidden layer of 10 neurons with ReLU activation function, and an output softmax layer. We then instantiate an optimizer with learning rate α=0.01, L2 regularization with weight decay parameter β=10−6. The training dataset that was obtained in the previous parts of the question was then split into 5-fold for cross-validation. The models are trained for 1000 epochs for each fold and each batch size. From there, we are able to obtain the mean validation accuracy for each batch size by taking the average validation accuracy of the 5 folds. The results obtain is shown in the figure below. From the results, we can see that the batch size that is the most optimal is the batch size of 4, as amongst the batch sizes that were trained, the batch size of 4 gives the best mean 5-fold cross-validation accuracy.

```
For batch 4 mean validation accuracy is: 0.909764289855957
For batch 8 mean validation accuracy is: 0.9057239174842835
For batch 16 mean validation accuracy is: 0.8956228971481324
For batch 32 mean validation accuracy is: 0.8956228971481324
For batch 64 mean validation accuracy is: 0.8902356863021851
Best mean validation accuracy: 0.909764289855957 from batch size 4
```

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

Using the optimal batch size, we train a sequential neural network that is composed of 3 layers, the input layer, a hidden layer of 10 neurons with ReLU activation function, and an output softmax layer. We then instantiate an optimizer with learning rate α=0.01, L2 regularization with weight decay parameter β=10−6 for 1000 epochs. The train and test accuracies against epochs graph is plotted as shown below.

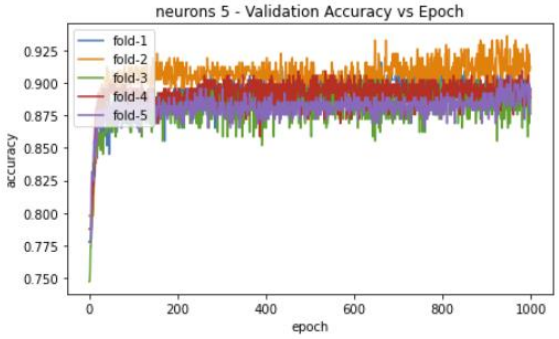


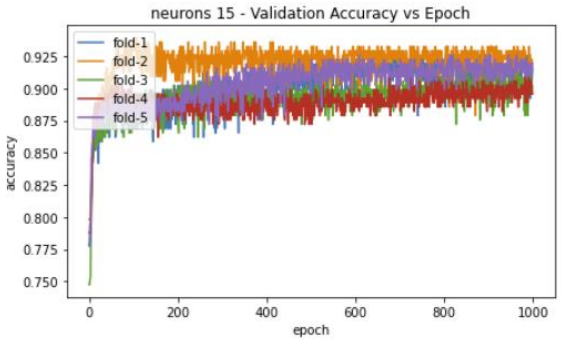


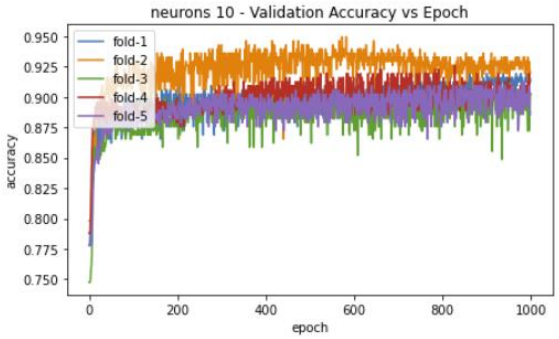
From the graph above, we can observe that for an optimal batch size, we are able to achieve a higher test accuracy, of 0.9091, when compared to the baseline model in Question 1 and that the test accuracy converges faster than the baseline model.

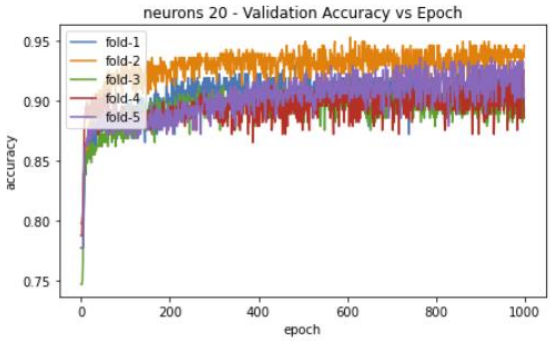
Question 3

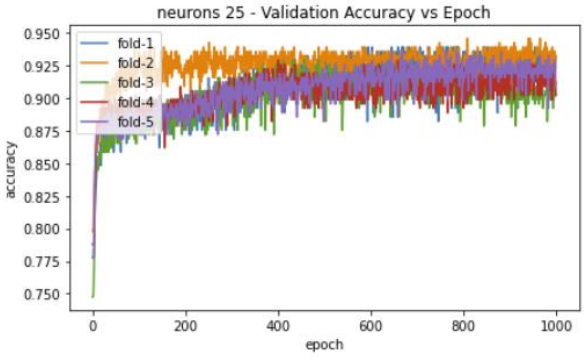
a) Plot the cross-validation accuracies against the number of epochs for different number of hidden-layer neurons.











b) Select the optimal number of neurons for the hidden layer. State the rationale for your selection.

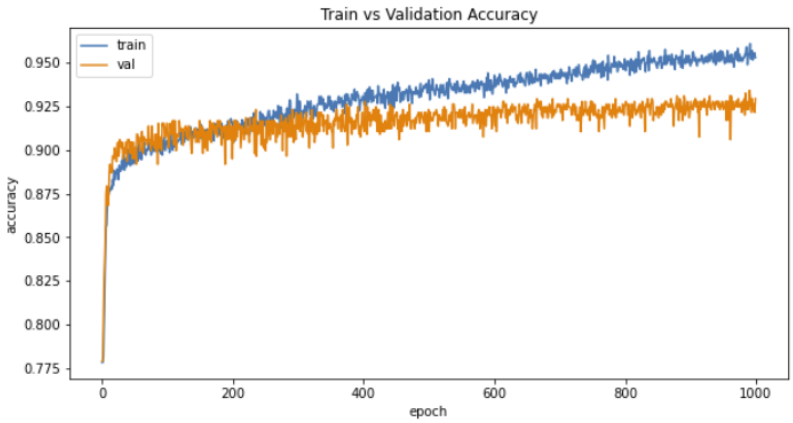
For each number of neurons, we first create a sequential neural network that is composed of 3 layers, the input layer, a hidden layer with ReLU activation function, and an output softmax layer. We then instantiate an optimizer with learning rate α=0.01, L2 regularization with weight decay parameter β=10−6. The training dataset that was obtained in the previous parts of the question was then split into 5-fold for cross-validation. The models are trained for 1000 epochs for each fold and each number of neurons. From there, we are able to obtain the mean validation accuracy for each number of neurons by taking the average validation accuracy of the 5 folds. The results obtain is shown in the figure below. From the results, we can see that the number of neurons that is the most optimal is 25 neurons for the hidden layer, as amongst the number of neurons that were trained, 25 neurons in the hidden layer gives the best mean 5-fold cross-validation accuracy.

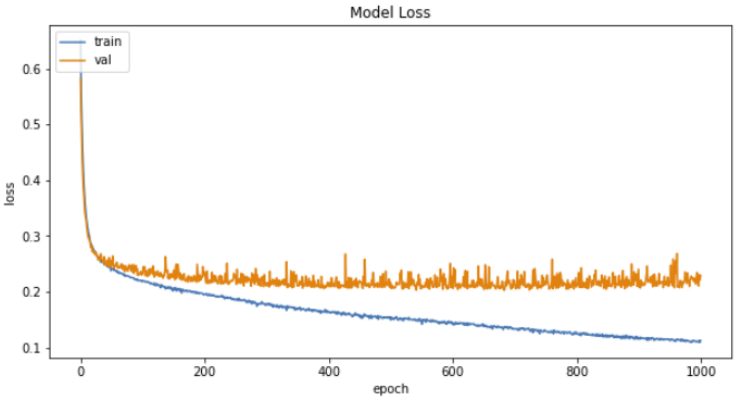
```
For 5 neurons mean validation accuracy is: 0.8962962985038757
For 10 neurons mean validation accuracy is: 0.9043771028518677
For 15 neurons mean validation accuracy is: 0.9104377031326294
For 20 neurons mean validation accuracy is: 0.9070707201957703
For 25 neurons mean validation accuracy is: 0.91649831533432
Best mean validation accuracy: 0.91649831533432 from neuron size 25
```

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

Using the optimal number of neurons, we train a sequential neural network that is composed of 3 layers, the input layer, a hidden layer of 25 neurons with ReLU activation function, and an output softmax layer. We then instantiate an optimizer with learning rate α=0.01, L2 regularization with weight decay parameter β=10−6 for 1000 epochs. The train and test accuracies against epochs graph is plotted as shown below. We can also observe that the test accuracy is higher than that of the baseline model at 0.9295 for the 1000th epoch.

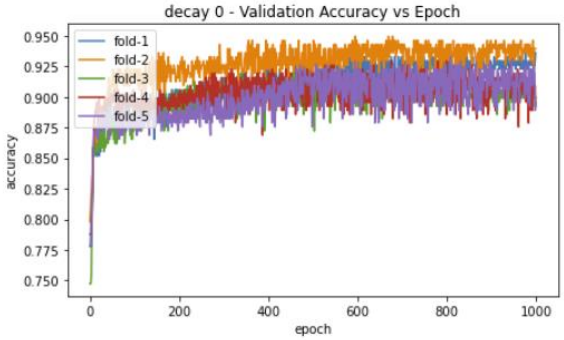
```
Epoch 1000/1000
372/372 - 0s - loss: 0.1121 - accuracy: 0.9530 - val_loss: 0.2292 - val
_accuracy: 0.9295
```

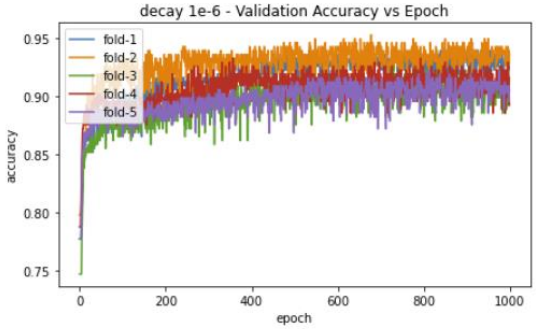


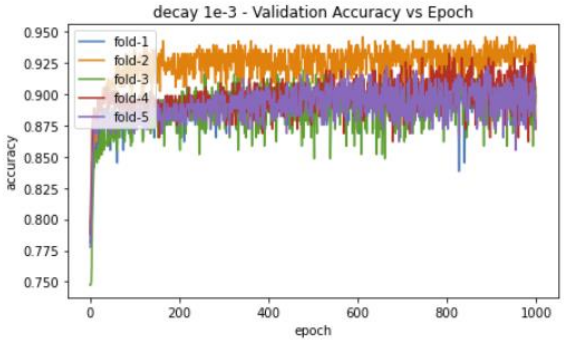


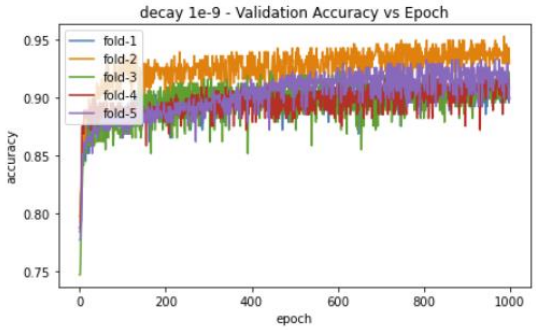
Question 4

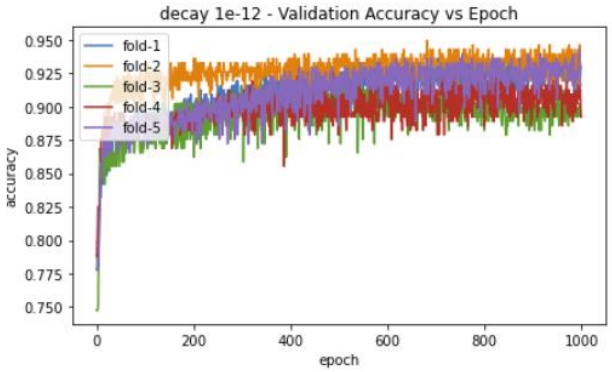
a) Plot cross-validation accuracies against the number of epochs for the 3-layer network for different values of decay parameters.











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

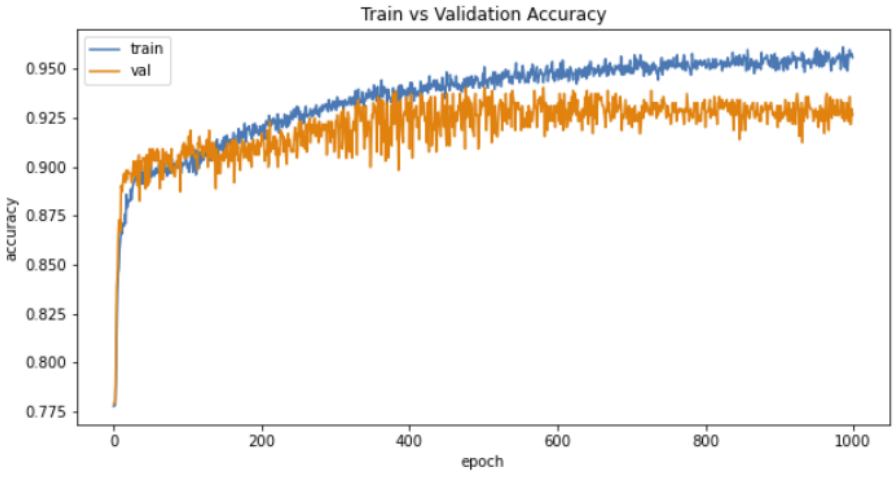
For each decay parameter, we first create a sequential neural network that is composed of 3 layers, the input layer, a hidden layer with ReLU activation function, and an output softmax layer. We then instantiate an optimizer with learning rate α=0.01. The training dataset that was obtained in the previous parts of the question was then split into 5-fold for cross-validation. The models are trained for 1000 epochs for each fold and each decay parameter. From there, we are able to obtain the mean validation accuracy for each decay parameter by taking the average validation accuracy of the 5 folds. The results obtain is shown in the figure below. From the results, we can see that the decay parameter that is the most optimal is the decay parameter of 1e-6, as amongst the decay parameters that were trained, the decay parameter of 1e-6 gives the best mean 5-fold cross-validation accuracy.

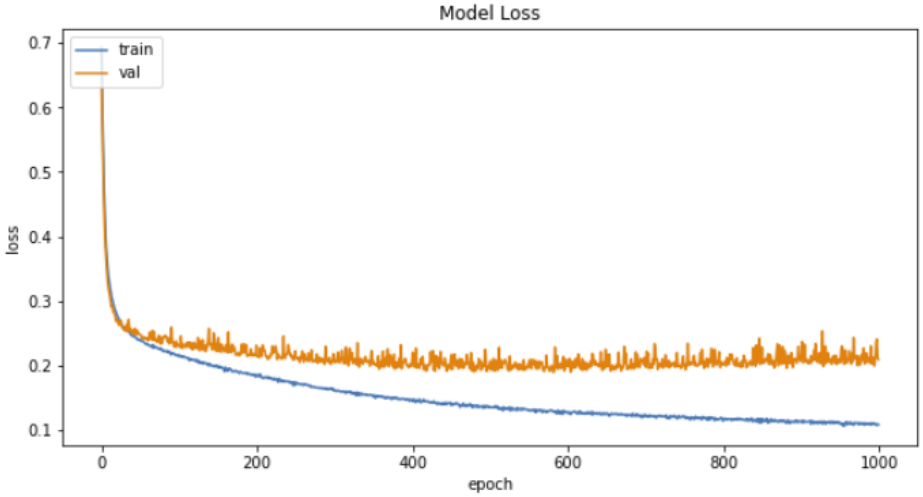
```
For 0 decay rate mean validation accuracy is: 0.9124579191207886
For 0.001 decay rate mean validation accuracy is: 0.8962962985038757
For 1e-06 decay rate mean validation accuracy is: 0.9158249139785767
For 1e-09 decay rate mean validation accuracy is: 0.9070707201957703
For 1e-12 decay rate mean validation accuracy is: 0.9117845058441162
Best mean validation accuracy: 0.9158249139785767 from decay rate 1e-06
```

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

Using the optimal decay parameter, we train a sequential neural network that is composed of 3 layers, the input layer, a hidden layer of 25 neurons with ReLU activation function, and an output softmax layer. We then instantiate an optimizer with learning rate α=0.01, L2 regularization with weight decay parameter β=10−6 for 1000 epochs. The train and test accuracies against epochs graph is plotted as shown below. We can also observe that the test accuracy is higher than that of the baseline model at 0.9263 for the 1000th epoch.

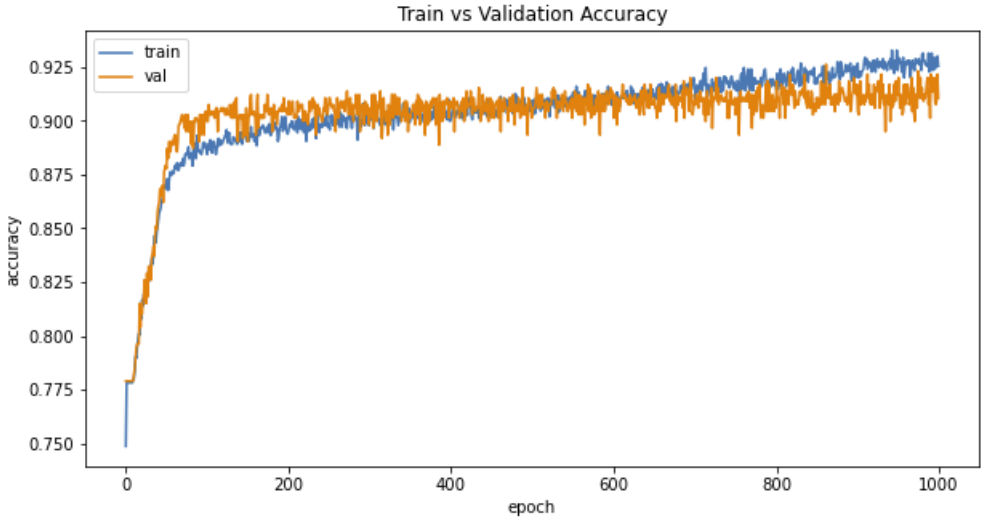
```
Epoch 1000/1000
372/372 - 0s - loss: 0.1086 - accuracy: 0.9556 - val_loss: 0.2096 - val_accuracy: 0.9263
```

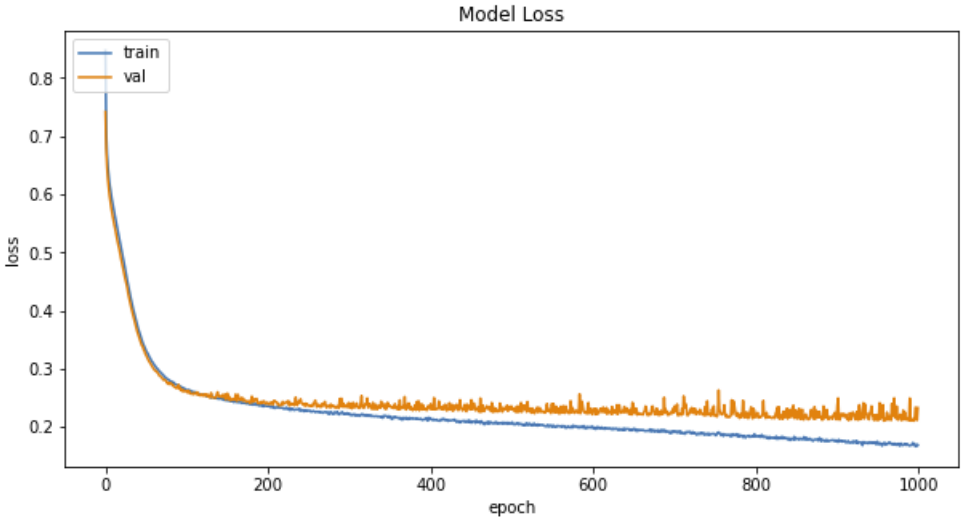




Question 5

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





b) Compare and comment on the performances of the optimal 3-layer and 4-layer networks.
For the optimal 3-layer network, we obtained a final test accuracy of 0.9263 and a test loss of 0.2096, whereas for the 4-layer network, , we obtained a final test accuracy of 0.9107 and a test loss of 0.2328. This shows that a 3-layer with optimal parameters can perform better than a 4-layer "deeper" network that does not have the optimal parameters. This tells us that tuning of network parameters is more important than utilizing a deeper neural network. Therefore, we must always tune the parameters of our neural networks for each task on hand to ensure that the performance of the network is the most optimal and we are able to achieve the best possible accuracy for a network of that particular depth.
We can also observe from the train and test accuracy graphs of the optimal 3-layer and 4-layer networks that an optimized network is able to learn faster and converge faster. Form the graphs we can see that the test loss for the 3-layer network converges at close to the 0^{th} epoch whereas the test loss for the 4-layer network converges at close to the 100^{th} epoch.

Conclusions

From the results obtained from the experiments conducted above, we can conclude that tuning of the various parameters of a neural network is very important. For the baseline model, we have a sequential neural network that is composed of 3 layers, the input layer, a hidden layer of 10 neurons with ReLU activation function, and an output softmax layer, instantiated with an optimizer with learning rate α=0.01, L2 regularization with weight decay parameter β=10−6. For this baseline model, we were able to only obtain test accuracies around 0.9, whereas after we used the optimal batch size, number of neurons in the hidden layer as well as the optimal weight decay, we were able to achieve test accuracies around 0.92.

We can also observe that a 3-layer with optimal parameters can perform better than a 4-layer "deeper" network that does not have the optimal parameters. This tells us that tuning of network parameters is more important than utilizing a deeper neural network. Therefore, we must always tune the parameters of our neural networks for each task on hand to ensure that the performance of the network is the most optimal and we are able to achieve the best possible accuracy for a network of that particular depth.

Neural Network	Test Accuracy at 1000^{th} Epoch
Baseline	0.9028
Optimized Batch Size	0.9091
Optimized Number of Neurons	0.9295
Optimized Weight Decay	0.9263
4-Layered	0.9107

Part B: Regression Problem

Question 1

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

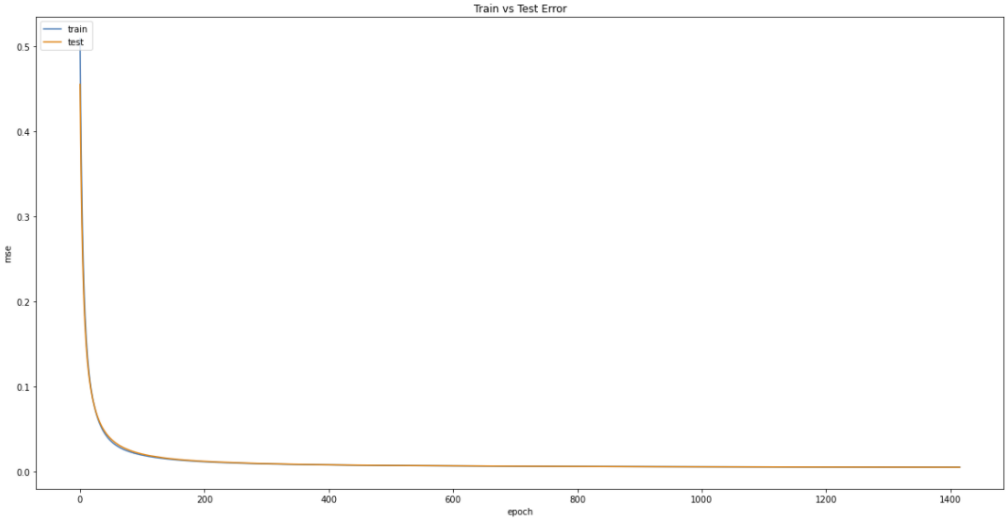
For this part of the question, we first extract the data from the **admission_predict.csv**, excluding the serial number, and take the first 7 columns of the data as the input features and the last column as the output to be predicted. We then normalize the data and split the data into 70:30 ratio for training and testing.

```
X_data = (X_data- np.mean(X_data, axis=0))/ np.std(X_data, axis=0)
X_train, X_test, Y_train, y_test = train_test_split(X_data, Y_data, test_size=0.3, shuffle = False)
```

For the first model, we designed 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 β=10−3 and a learning rate α=10−3 to train the network. In order to limit the number of epochs for training, we used an early stopping function to stop the training of the model if the improvement of the validation m.s.e for the current epoch is less than 1e-9 from the previous epoch validation m.s.e.

```
callback = tf.keras.callbacks.EarlyStopping(monitor='val_mse',min_delta=1e-9, patience=0)

l2 = tf.keras.regularizers.l2(1e-3)

print(l2)

# create a network
model = Sequential([
        Dense(num_neurons, activation='relu',kernel_regularizer=l2),
        Dense(1)
])

opt = tf.keras.optimizers.SGD(lr=0.001)

model.compile(optimizer=opt,
              loss="mean_squared_error",
              metrics=['mse'])

# learn the network
histories['qn1'] = model.fit(X_train, Y_train,
                             epochs=epochs,
                             batch_size=batch_size,
                             verbose = 2,
                             callbacks=[callback],
                             validation_data=(X_test, y_test))
```

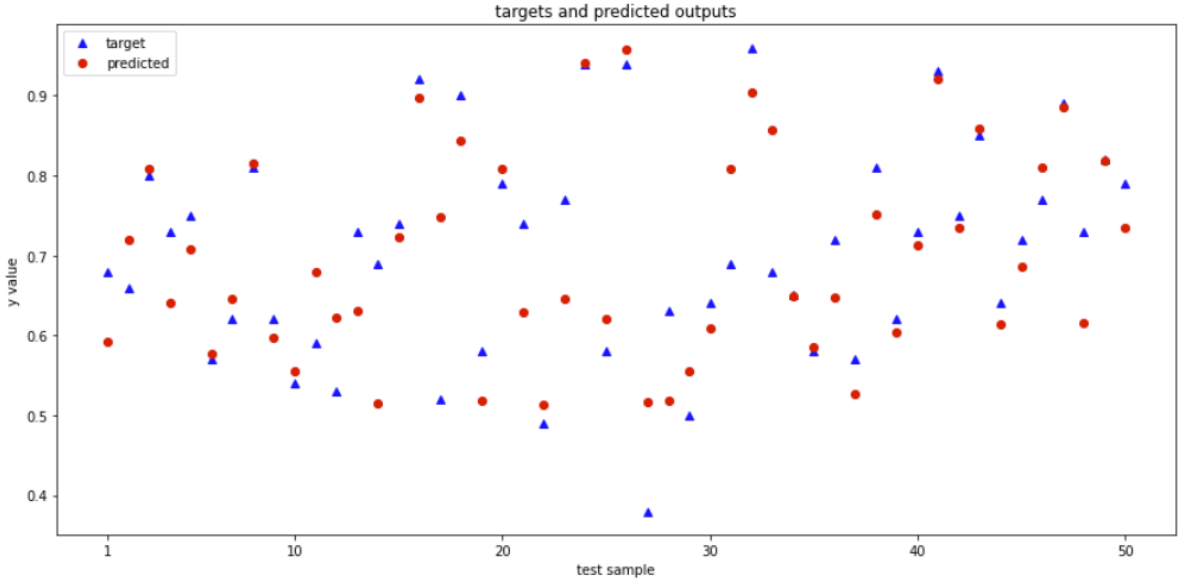
After training the neural network, the early stopping was called at Epoch 1416. The graph of the train and test errors against epochs is shown below. As can be seen from the graph, the training m.s.e and test m.s.e are fairly similar for epochs after 200.



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

As mentioned previously, an early stopping function is used to stop the training of the model if the improvement of the validation m.s.e for the current epoch is less than 1e-9 from the previous epoch's validation m.s.e. This is due to the fact that although the validation m.s.e of the model is quite convergent after 200 epochs; it is still going down. Therefore, an early stopping function is used, and the test error is deemed minimum when of the validation m.s.e for the current epoch is less than 1e-9 from the previous epoch's validation m.s.e, which is at Epoch 1416.

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



Question 2

For this question, we are supposed to use Recursive feature elimination (RFE) to remove unnecessary features from the inputs. The initial dataset contains 7 input features. We start by running the 3-layer feedforward neural network which consists of an input layer, a hidden-layer of 10 neurons having ReLU activation functions, and a linear output layer, on the different combinations of 6 input features. We then select the 6 input features combination that gives the best test m.s.e. The 6 input features combinations are created as follows:

```
X_data, Y_data = admit_data[1:,[2,3,4,5,6,7]], admit_data[1:,-1]
Y_data = Y_data.reshape(Y_data.shape[0], 1)
X_data, Y_data = X_data[idx], Y_data[idx]
print(X_data[idx])
x_sets.append(X_data)
y_sets.append(Y_data)

X_data, Y_data = admit_data[1:,[1,3,4,5,6,7]], admit_data[1:,-1]
Y_data = Y_data.reshape(Y_data.shape[0], 1)
X_data, Y_data = X_data[idx], Y_data[idx]
print(X_data[idx])
x_sets.append(X_data)
y_sets.append(Y_data)

X_data, Y_data = admit_data[1:,[1,2,4,5,6,7]], admit_data[1:,-1]
Y_data = Y_data.reshape(Y_data.shape[0], 1)
X_data, Y_data = X_data[idx], Y_data[idx]
print(X_data[idx])
x_sets.append(X_data)
y_sets.append(Y_data)

X_data, Y_data = admit_data[1:,[1,2,3,5,6,7]], admit_data[1:,-1]
Y_data = Y_data.reshape(Y_data.shape[0], 1)
X_data, Y_data = X_data[idx], Y_data[idx]
print(X_data[idx])
x_sets.append(X_data)
y_sets.append(Y_data)

X_data, Y_data = admit_data[1:,[1,2,3,4,6,7]], admit_data[1:,-1]
Y_data = Y_data.reshape(Y_data.shape[0], 1)
idx = np.arange(X_data.shape[0])
print(X_data[idx])
x_sets.append(X_data)
y_sets.append(Y_data)
final_x_set.append(X_data)
final_y_set.append(Y_data)

X_data, Y_data = admit_data[1:,[1,2,3,4,5,7]], admit_data[1:,-1]
Y_data = Y_data.reshape(Y_data.shape[0], 1)
idx = np.arange(X_data.shape[0])
print(X_data[idx])
x_sets.append(X_data)
y_sets.append(Y_data)

X_data, Y_data = admit_data[1:,[1,2,3,4,5,6]], admit_data[1:,-1]
Y_data = Y_data.reshape(Y_data.shape[0], 1)
idx = np.arange(X_data.shape[0])
print(X_data[idx])
x_sets.append(X_data)
y_sets.append(Y_data)
```

We then train neural network models on these created 6 input features datasets for 1500 epochs and obtained the results as shown below.

```
For x_data set 1 m.s.e is: 0.005857745185494423
For x_data set 2 m.s.e is: 0.004573947284370661
For x_data set 3 m.s.e is: 0.006292017642408609
For x_data set 4 m.s.e is: 0.005873831454664469
For x_data set 5 m.s.e is: 0.0038928338326513767
For x_data set 6 m.s.e is: 0.005230068229138851
For x_data set 7 m.s.e is: 0.004204415716230869
Best m.s.e: 0.0038928338326513767 from x_data set 5
```

We can see that the 5th combination for 6 input features gives the best test m.s.e. The combination contains the GRE Score, TOEFL Score, University Rating, SOP, CGPA and Research columns of the initial dataset.

Next, we run the 3-layer feedforward neural network which consists of an input layer, a hidden-layer of 10 neurons having ReLU activation functions, and a linear output layer, on the different combinations of 5 input features. We then select the 5 input features combination that gives the best test m.s.e.

We then train neural network models on these created 6 input features datasets for 1500 epochs and obtained the results as shown below.

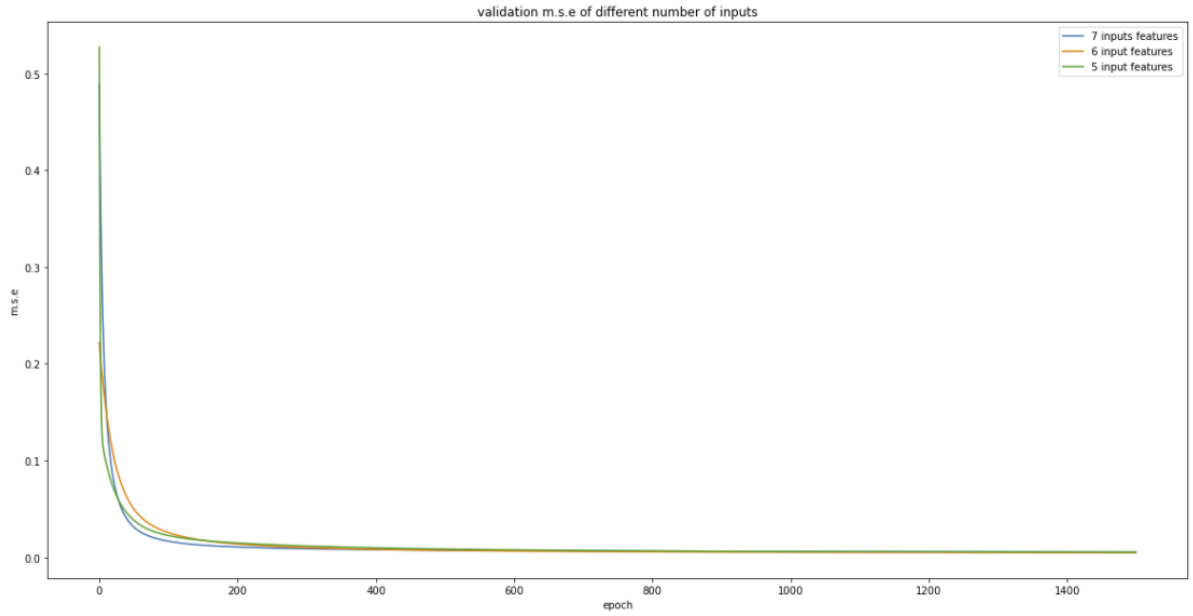
```
For x_data set 0 m.s.e is: 0.006046404130756855
For x_data set 1 m.s.e is: 0.005025614984333515
For x_data set 2 m.s.e is: 0.0052968887612223625
For x_data set 3 m.s.e is: 0.005141674540936947
For x_data set 4 m.s.e is: 0.006478298921138048
For x_data set 5 m.s.e is: 0.004788455553352833
Best m.s.e: 0.004788455553352833 from x_data set 5
```

We can see that the 5th combination for 5 input features gives the best test m.s.e. The combination contains the GRE Score, TOEFL Score, University Rating, SOP and CGPA columns of the initial dataset.

After obtaining the best 6 input features and 5 input features datasets, we compare the accuracy of the model with all input features, with models using 6 input features and 5 input features selected using RFE. We train the 3-layer feedforward neural network on the 3 different datasets for 1500 epochs and obtain the findings below.

```
For data set with 7 features, m.s.e is: 0.005311346147209406
For data set with 6 features, m.s.e is: 0.004719463177025318
For data set with 5 features, m.s.e is: 0.0052022626623511314
Best m.s.e: 0.004719463177025318 from data set with 6 features
```

From the above results, we can see that the dataset with 6 input features perform the best amongst the datasets with all input features, 6 input features and 5 input features, having the lowest test m.s.e when trained for the same number of epochs.



The graph above shows that for all the 3 datasets, the test m.s.e converges at around 200 epochs and that the test m.s.e of the 3 datasets are relatively similar to each other with very minute differences in the test m.s.e values.

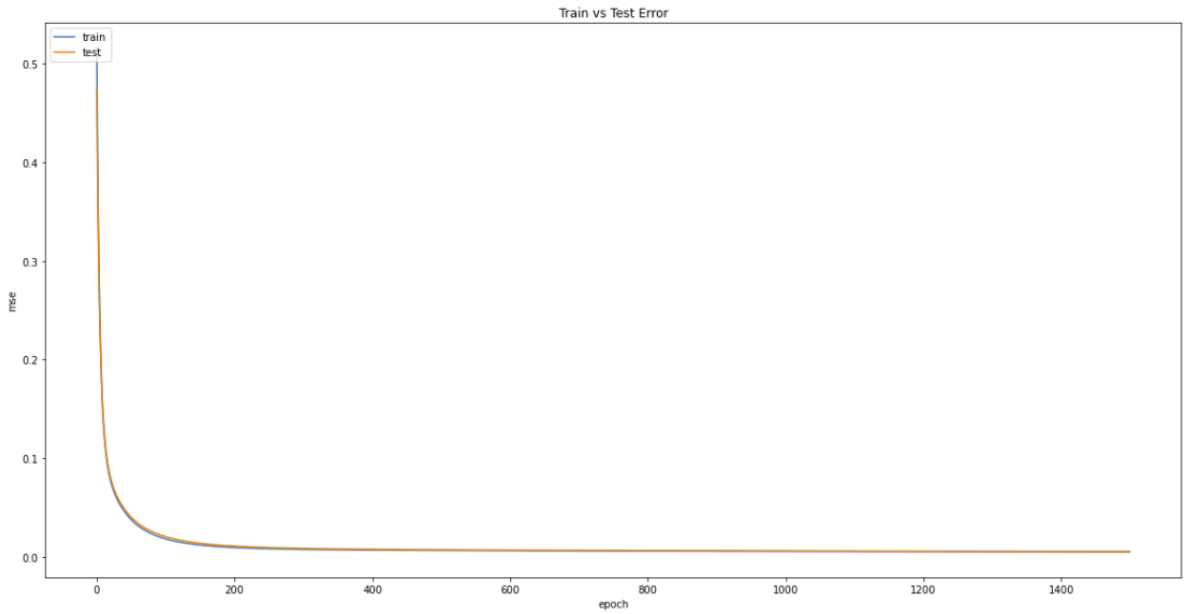
Question 3

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).

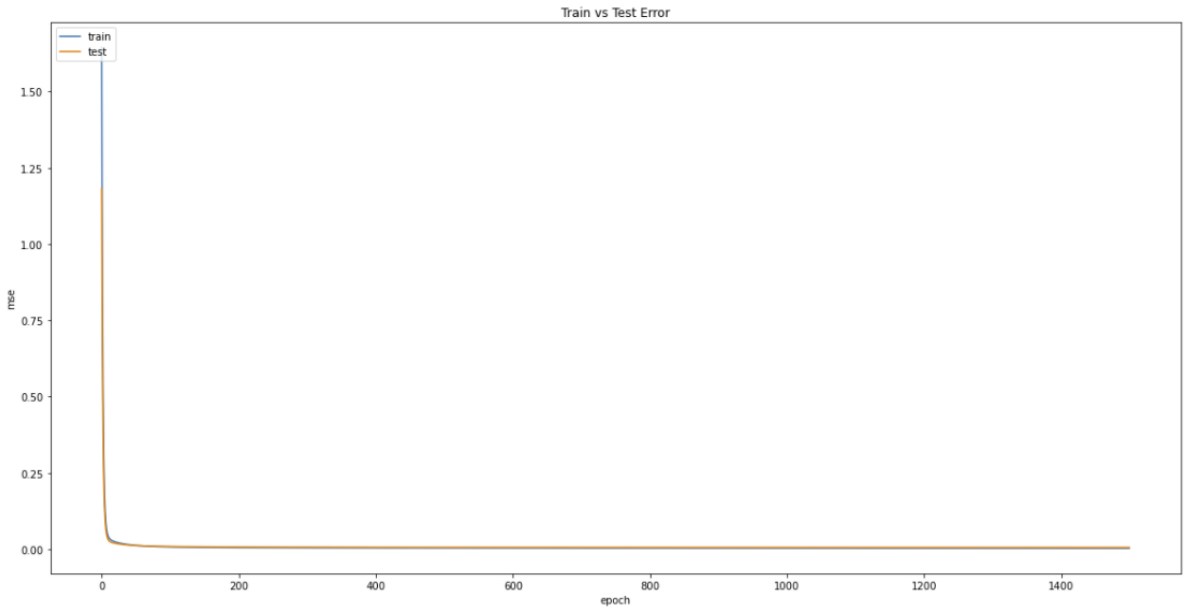
For all the neural networks for this question, we use a *L*2 regularization of weight decay parameter $\beta=10^{-3}$ and train all of them on the same number of epochs (1500 epochs). The most optimal dataset with 6 features containing the GRE Score, TOEFL Score, University Rating, SOP, CGPA and Research columns was used to train the models. We then plot the train and test errors against epochs graphs and predicted values and target values for any 50 test samples graphs for each of the neural network.

4-Layered Neural Network with No Dropouts

From the train and test errors against epochs graphs of the 3-Layered Neural Network and 4-Layered Neural Network with no dropouts, we can observe that both the train and test m.s.e for the and 4-Layered Neural Network converges faster than the 3-Layered Neural Network, with the values converging near the 0^{th} epoch whereas the m.s.e values for the 3-Layered Neural Network converges near the 200^{th} epoch. The graphs tells us that a 4-Layered Neural Network learns faster than the 3-Layered Neural Network.



3-Layered Neural Network m.s.e vs. epoch



4-Layered Neural Network m.s.e vs. epoch

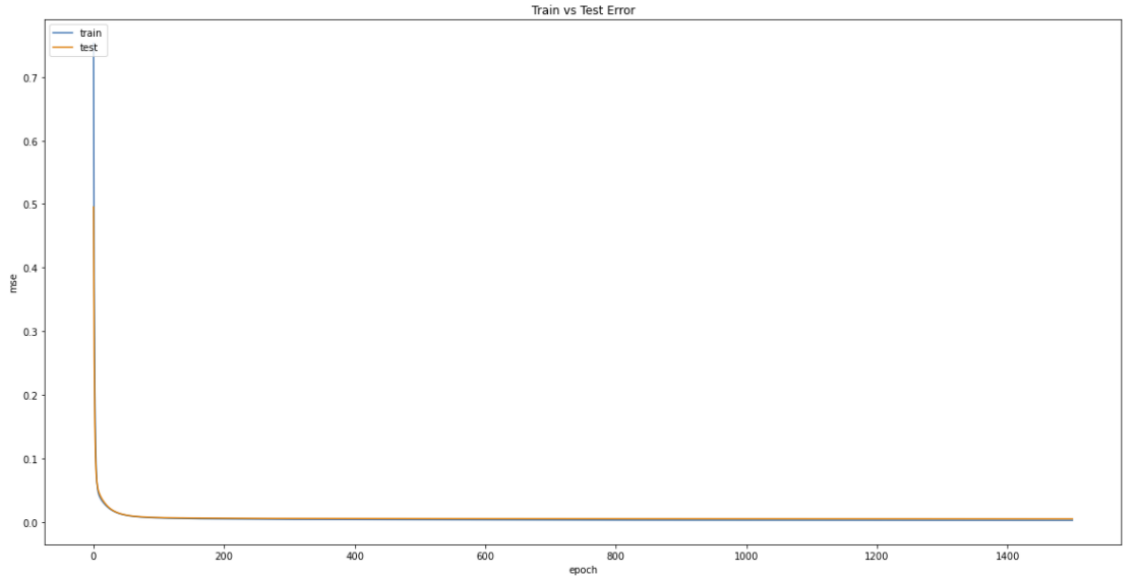
However, the test m.s.e for the 4-Layered Neural Network with no dropouts for the last epoch was worse than for the 3-Layered Neural Network as can be seen from the results below.

```
Validation accuracy of 3-layered network: 0.0054005468264222145
Validation accuracy of 4-layered network without dropouts: 0.005758028011769056
```

5-Layered Neural Network with No Dropouts

The test m.s.e. of the 5-Layered Neural Network with No Dropouts fair better than both the 3-Layered Neural Network as well as the 4-Layered Neural Network with no dropouts. From the train and test errors against epochs graph we can observe that both the train and test m.s.e for the and 5-Layered Neural Network converges faster than the 3-Layered Neural Network, with the values converging near the 0^{th} epoch, similar to the 4-Layered Neural Network. From this we can conclude that a deeper Neural Network learns faster as compared to a shallow Neural Network.

```
Validation accuracy of 5-layered network with no dropouts: 0.005155117250978947
```

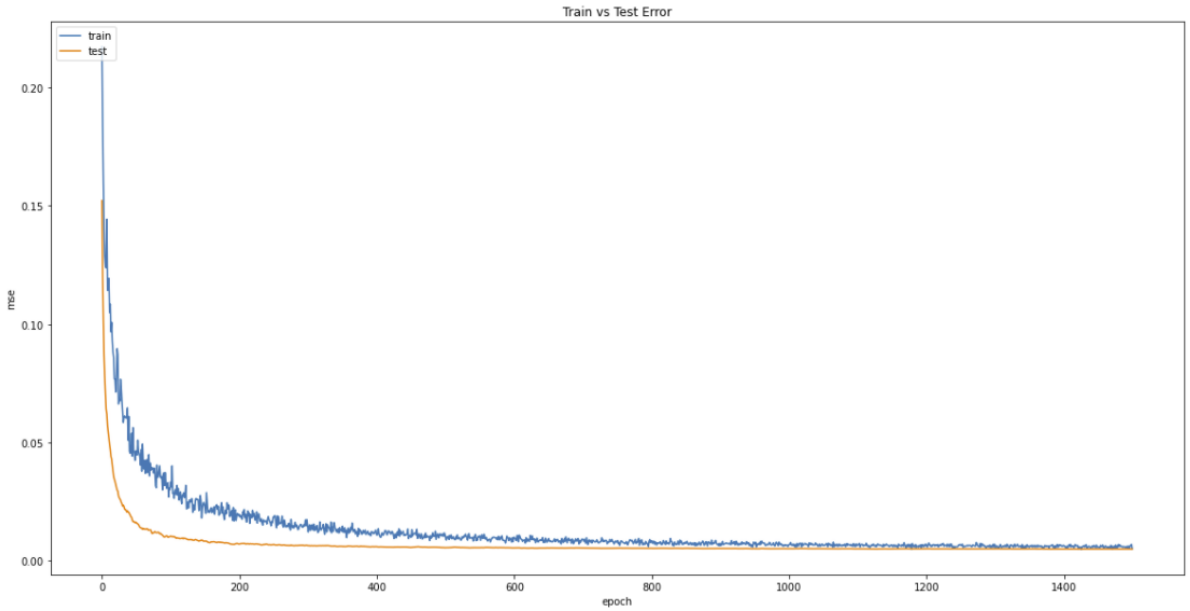


5-Layered Neural Network m.s.e vs. epoch

4-Layered Neural Network with Dropouts

The test m.s.e. of the 4-Layered Neural Network with Dropouts fair better than the 3-Layered Neural Network, the 4-Layered Neural Network with no dropouts as well as the 5-Layered Neural Network with no dropouts. Dropout is a technique used to prevent a model from overfitting. For this assignment, we used a dropout rate of 0.2 to randomly drop neurons (along with their connections) from the networks during training. This prevents neurons from co-adapting and thereby reduces overfitting.

From the graph, we can see that although the test m.s.e for the 4-Layered Neural Network with Dropouts converges very quickly near the 0^{th} epoch, the training m.s.e for the 4-Layered Neural Network with Dropouts converges only at about the 1000^{th} epoch and the curve is less "smooth". This is due to the fact that with 0.2 probability of the neurons in the network being dropped, the neurons learns the training data slower and is not able to co-adapt between one another. Although the train m.s.e converges slower, the final test m.s.e is better than a similar depth neural network with no dropouts and we prevent the model from overfitting.




```
Validation accuracy of 4-layered network with dropouts: 0.004995846189558506
```

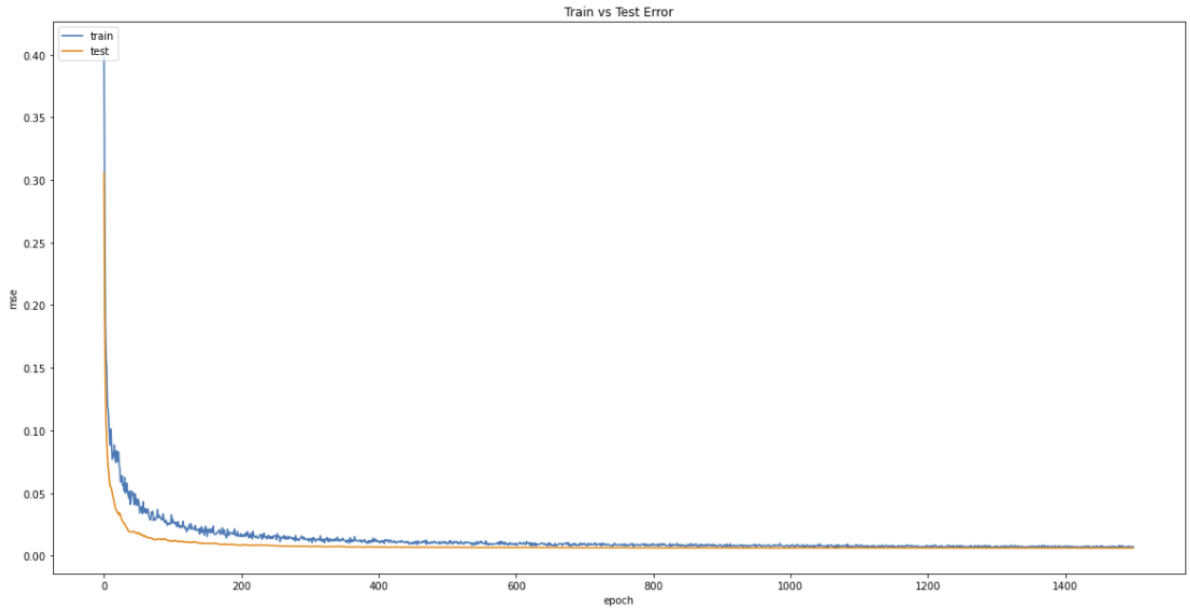
5-Layered Neural Network with Dropouts

From the results below, we can see that the 5-Layered Neural Network with Dropouts faired the worst among all the Neural Networks that we trained for 1500 epochs in terms of the test m.s.e value. This can be due to the fact that a 5-Layered Neural Network may not be suitable for this particular task or that due to the dropouts, the 5-Layered Neural Network with Dropouts may have to be trained for more epochs due to the complexity of the network.

```
Validation accuracy of 3-layered network: 0.0054005468264222145
Validation accuracy of 4-layered network without dropouts: 0.005758028011769056
Validation accuracy of 5-layered network without dropouts: 0.005155117250978947
Validation accuracy of 4-layered network with dropouts: 0.004995846189558506
Validation accuracy of 5-layered network with dropouts: 0.006081616040319204
```

Similar to the 4-Layered Neural Network with Dropouts, we can see that for the 5-Layered Neural Network with Dropouts, the test m.s.e converges relatively quickly whereas the train m.s.e converges

slower due to the drop out of the neurons in the network. The train m.s.e converges at around the 600th epoch.



Conclusion

From the questions answered in Part B of the assignment, we can see that reducing the number of input features for a neural network helps to reduce the m.s.e. However, that does not necessary mean that the lesser the number the input features, the better the results obtained. From the results above, we can observe that for this particular dataset, the most optimal dataset has 6 features from the original 7 and that the dataset with 6 features produces better results than the dataset with 5 input features.

From the questions, we can also conclude that it is not always true that a deeper neural network will produce better results as can be seen from the comparison of the different neural networks created and trained above. The depth required for a neural network differs from dataset to dataset and sometimes a neural network that is too complex may end up being worse off.

Also, we can see an improvement in performance of a neural network by introducing dropouts of neurons during each epoch of training. This can be seen from the results where the m.s.e for the neural network with 4-Layers with dropouts is better than the m.s.e of the neural network with 4-Layers with no dropouts.