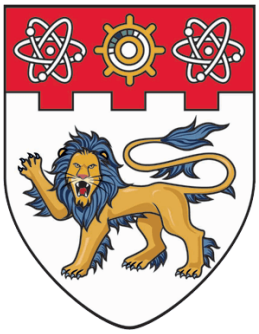


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

**School of Computer Science and Engineering**

# CZ4042 - Neural Network & Deep Learning

## Project 1


| Name | Matriculation Number |
|---|---|
| Quek Chin Wei | U1620394D |
| Ching Jia Chin | U1620237E |

## Introduction

For this project, we are doing 2 tasks, classification & regression. Classification requires us to train a neural network to classify fetal heart rate (FHR) and uterine contraction (UC) according to 3 classes (Normal, Suspect & Pathologic). The classification in the dataset have been manually done by experts & we are training the model based on this dataset.

For regression, we are training a neural network to predict the chances of a graduate getting into a Master Program. We have parameters such as GRE score, Letter of recommendation, etc that we can use to make our prediction. The dataset also contains the predicted chance of entering a particular university.

## Methods

In this project, we use Keras which is TensorFlow's high level API for building and training deep learning models. It's fast in prototyping, easy to use and extend and give insightful visualization on training history.

The default weight initialization used in Keras is Glorot uniform initializer. It draws samples from a uniform distribution within [-limit, limit] where limit is sqrt( 6/ (fan_in + fan_out)) where fan_in is the number of input units in the weight tensor and fan_out is the number of output units in the weight tensor. We use it throughout the project. Different weight initialization method might give different training results. We used zero bias initialization in the project. Our analysis on the experiments and results is based on the outcome of the model with this initialization.

We wrote a python script _createTrain&TestDatasets.py to split the given data into training and testing data and write it as two csv files. The given data was shuffled and then split into 70:30 training to testing ratio. We further split out the 'training data' to do 5-fold cross validation in some of the questions. 4 folds of the 'training data' will be used for training and 1 fold will be used for validation. Eventually every data will be used for training and validation. Each models/experiments with different hyperparameter (batch size, number of neurons in hidden layers, weight decay choices etc ) will be trained and validated 5 times with 5-fold cross validation. The outcomes of each models are then averaged out to select the best model.

**Experiments and Results**

Part A

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.

   The normalization method used in part A is min-max scaling. The input features will have range in [0,1].

$$X' = \frac{xi - x\ min}{x\ \max - x\ min}$$

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

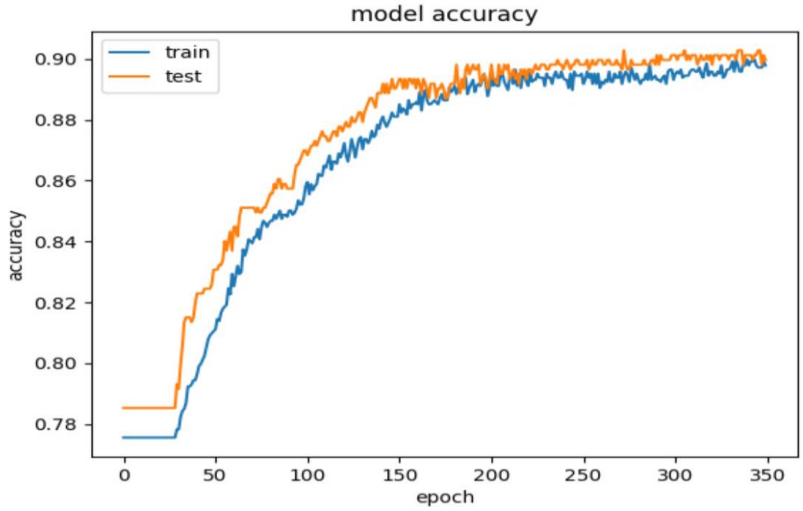



   Figure 1a shows training accuracy and test accuracy of the model against epoch.

b) State the approximate number of epochs where the test error converges.

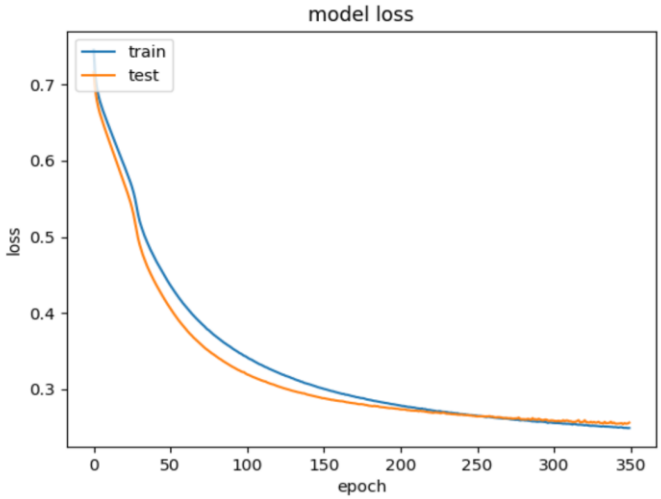



Figure 1b shows training loss and test loss of the model against epoch

From figure 1b, the test error converges at epoch 200 approximately. Total epochs for the whole training is 350. We evaluate the test error of the model at the end of every epoch.

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 sizes.

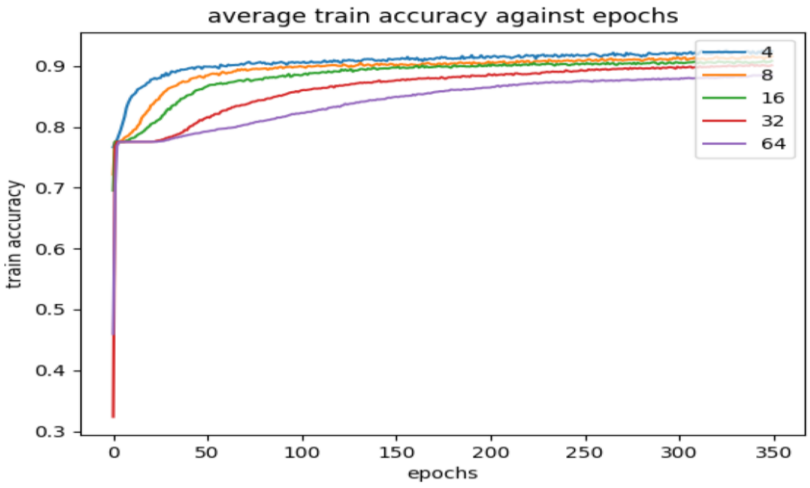



Figure 2a

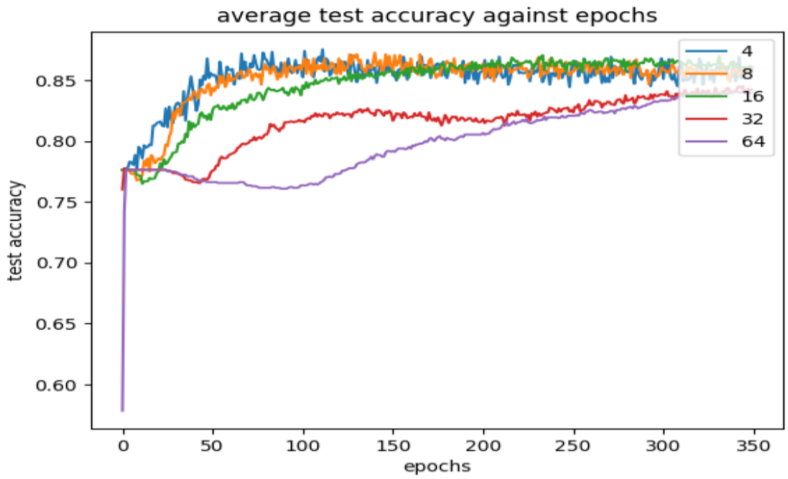



Figure 2b

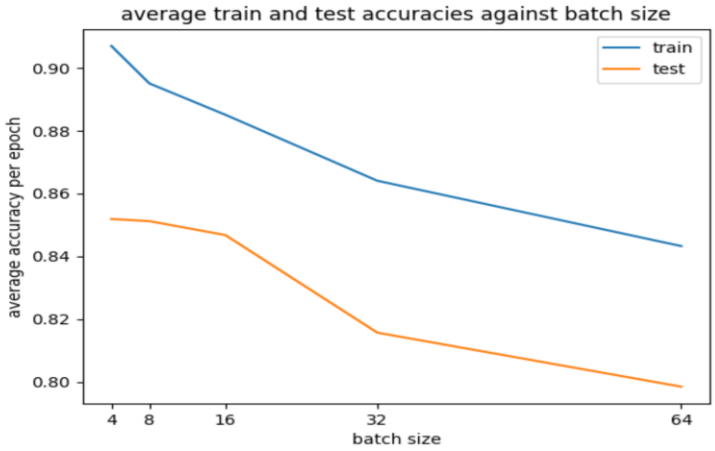



Figure 2c

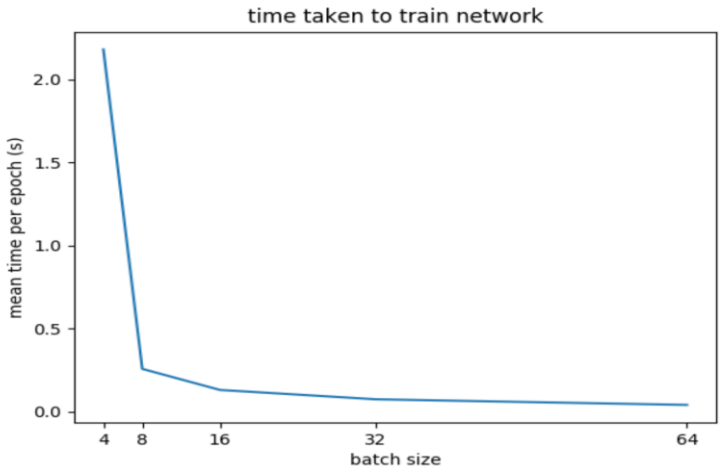



Figure 2d

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

| Batch size | Average Train Accuracy | Average Test Accuracy | % Increase Average Test Accuracy |
|---|---|---|---|
| 4 | 0.9070173 | 0.8518865 | 0.00079 |
| 8 | 0.8950745 | 0.8512113 | 0.0053 |
| 16 | 0.88509923 | 0.8467609 | 0.038 |
| 32 | 0.864072 | 0.81567293 | 0.022 |
| 64 | 0.8432582 | 0.7984626 | - |

From python terminal or figure 2c,

Average_train_accuracy_per_epoch

[0.9070173 0.8950745 0.88509923 0.864072 0.8432582 ]

Average_test_accuracy_per_epoch

[0.8518865 0.85121113 0.8467609 0.81567293 0.7984626 ]

From figure 2c, we can see that the training and test accuracy is the highest when the batch size is smallest as smaller batch size gives randomness to the model during gradient descent. From figure 2d, as the batch size decreases, the training time per epoch increases. The optimal batch size that we choose is 16 because it gives the highest increase in test accuracy when we decrease the batch size from 32 to 16. Further decrease in batch size doesn't help much in the test accuracy however we increase the training time significantly. From figure 2d, the training time per epoch for batch size 4, 8, 16 are 2.2 , 0.25, 0.1 respectively. We can then compute the time to complete one training.

Batch size 4 – 2.2 seconds * 350 epochs = 770 seconds / 12.8 minutes

Batch size 8 – 0.25 seconds * 350 epochs = 87.5 seconds / 1.46 minutes

Batch size 16 – 0.1 seconds * 350 epochs = 35 seconds / 0.58 minutes

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

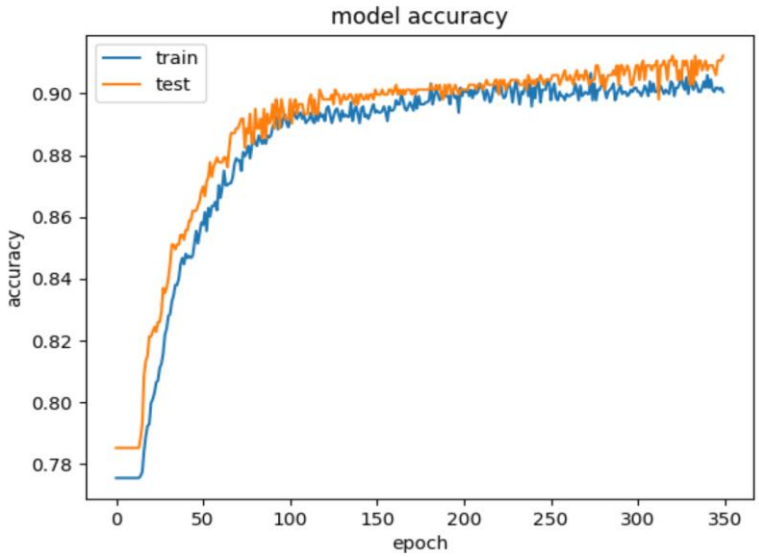



Figure 2e

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}.

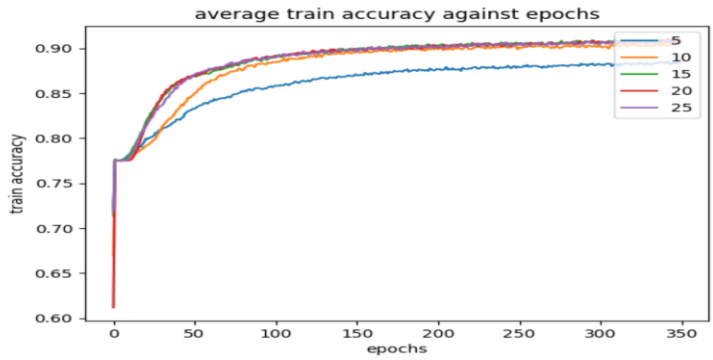



Figure 3a

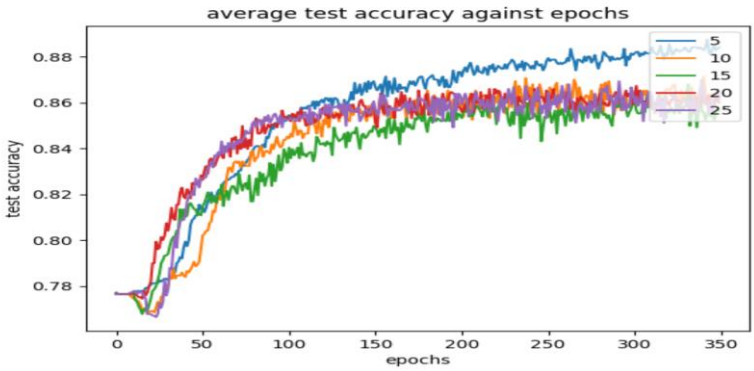



Figure 3b

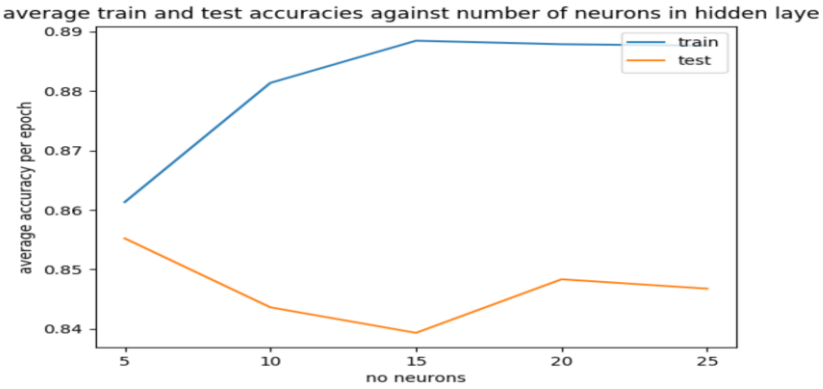



Figure 3c

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

From python terminal or figure 3c,

| | | | | | |
|---|---|---|---|---|---|
| Avg_train_acc | 0.86133146 | 0.8813838 | 0.88847536 | 0.8878852 | 0.88764447 |
| Avg_test_acc | 0.8552227 | 0.8436249 | 0.83933043 | 0.8483425 | 0.8467648 |

The optimum number of neurons for the hidden layer would be 5. From figure 3c, it gives the highest test accuracy. When the number of neurons in the hidden layer increases from 5 to 10 and 15, the training accuracy increases while the test accuracy decreases which implies that the model is overfitting. Theoretically, the network attempts to remember the training patterns with increasing number of parameters or hidden neurons, minimizing the training error at the expense of its generalization ability on unseen data. Although the test accuracy started to increase when number of neurons in the hidden layer increases from 15 to 20 and 25, by trial and error, the optimum number of neurons still is 5.

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

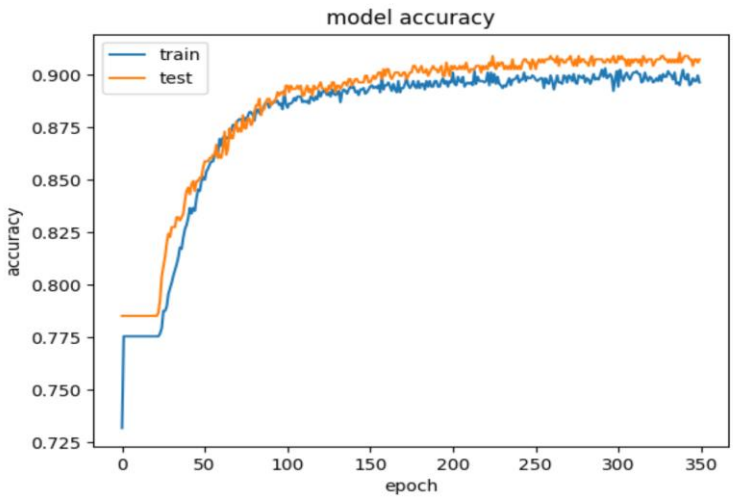



Figure 3d

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 $\{0, 10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}\}$.

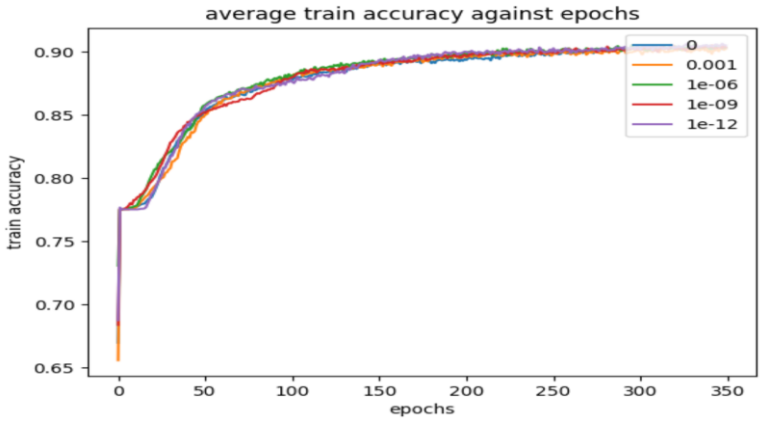



Figure 4a

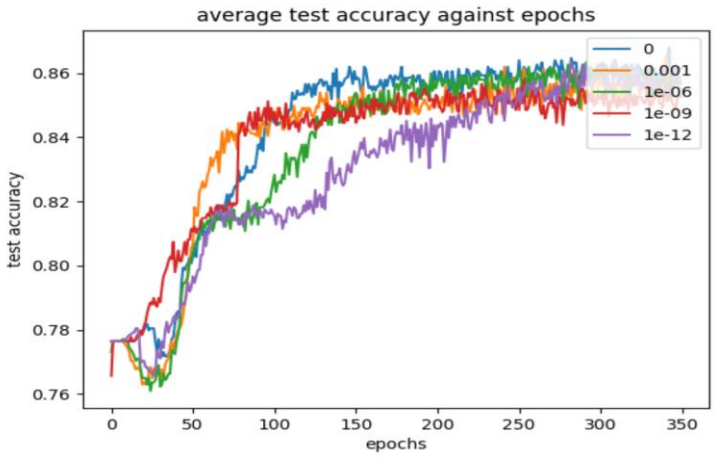



Figure 4b

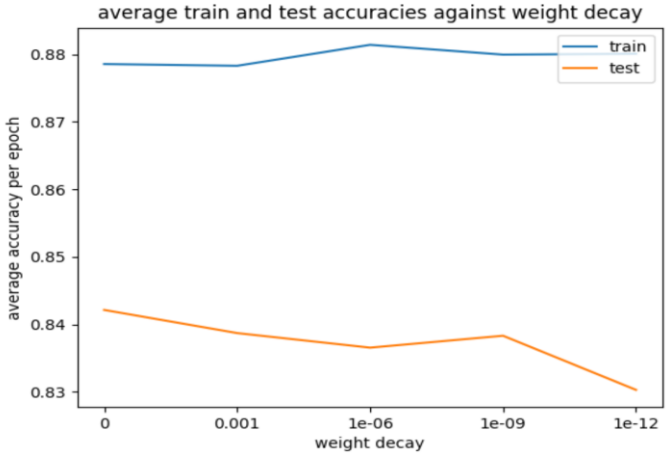



Figure 4c

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

From python terminal or figure 4c,

Average_train_accuracy_per_epoch

[0.87854624 0.8782872 0.8813934 0.8799444 0.8800691 ]

Average_test_accuracy_per_epoch

[0.84214914 0.83872247 0.8365637 0.83833 0.8303165 ]

The optimum decay parameter would be 0 based on the test accuracy. Weight decay is an regularization approach used to prevent overfitting. It gives penalty to the cost function. The choice of decay parameter here is only based on which decay parameter gives the best average test accuracy.

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

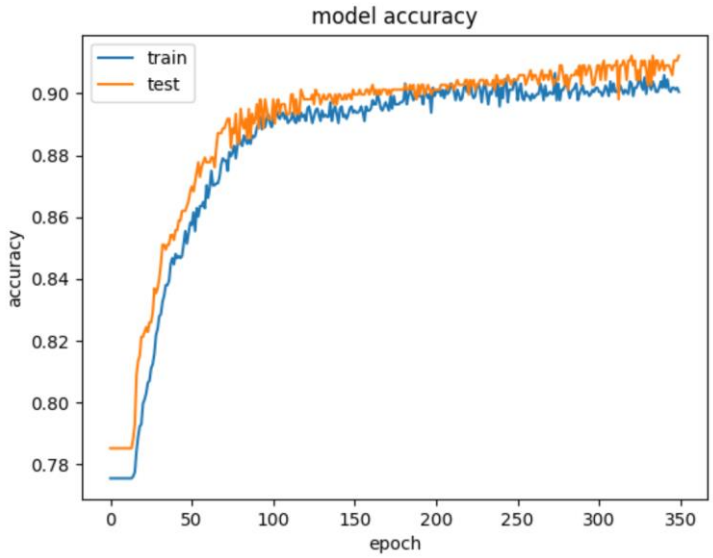



Figure 4d

5. After you are done with the 3-layer network, design a 4-layer network with two hidden- layers, each consisting 10 neurons, and train it with a batch size of 32 and decay parameter $10^{-6}$.

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

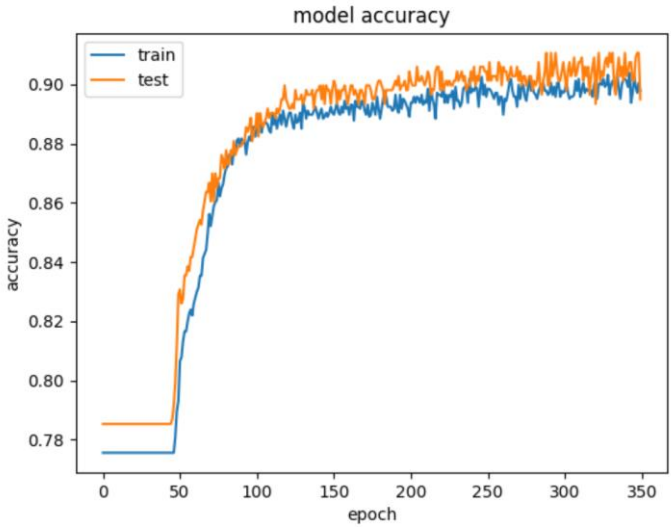
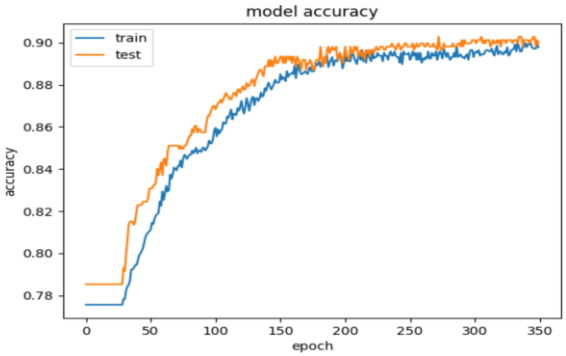


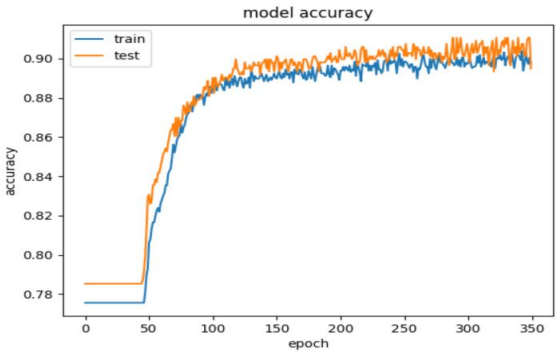

Figure 5a

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

*3-layer network*




*4-layer network*




3 layer

Average train accuracy - 0.86648804

Average test accuracy - 0.8741917

4 layer

Average train accuracy - 0.87216395

Average test accuracy - 0.8790372

*Conclusion : 4-layer network performs slightly better than 3-layer network. The increase in performance is not significant. Theoretically, higher number of layers would perform better if the problem we were trying to solve is complex and provided we have enough amount of training data to fit the model well.*

Part B

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.

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

a)

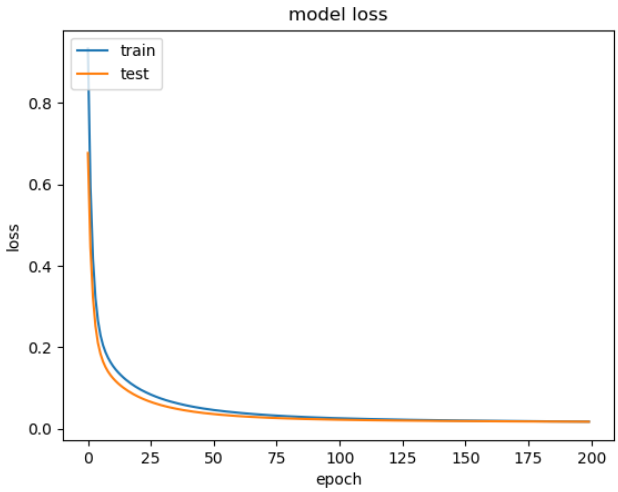



Figure 1a

b) Based on figure 1a, the optimal number of epochs is 100.

c)

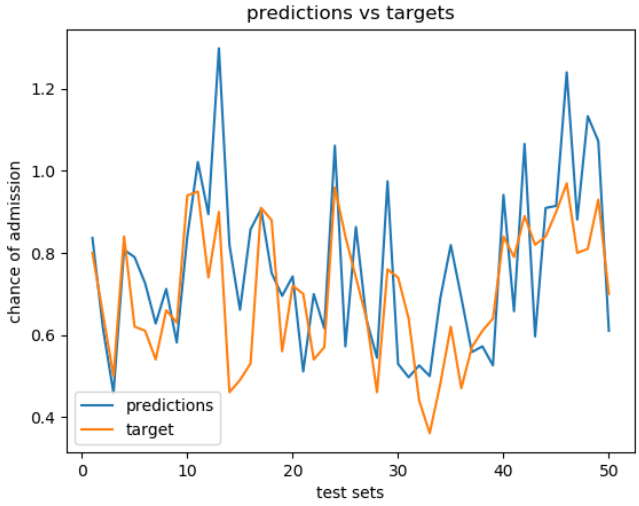
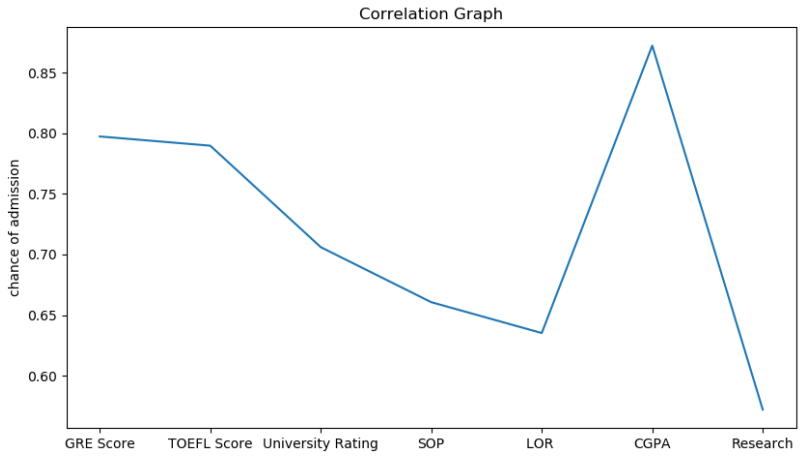



Figure 1c

2. Use the train data to compute (and plot) an 8X8 correlation matrix between the different feature scores and the corresponding chances of admit.
a) Which features are most correlated to each other? Is it justifiable?
b) What features have the highest correlations with the chances of admit?

Correlation Matrix

| | GRE Score | TOEFL Score | University Rating | SOP | LOR | CGPA | Research | Chance of admission |
|---|---|---|---|---|---|---|---|---|
| GRE Score | 1 | 0.820288 | 0.63166 | 0.565743 | 0.499807 | 0.811529 | 0.578914 | 0.797415 |
| TOEFL Score | 0.820288 | 1 | 0.657618 | 0.622333 | 0.53029 | 0.807247 | 0.474875 | 0.789813 |
| University Rating | 0.63166 | 0.657618 | 1 | 0.729212 | 0.604471 | 0.722389 | 0.459666 | 0.706071 |
| SOP | 0.565743 | 0.622333 | 0.729212 | 1 | 0.660102 | 0.706401 | 0.400011 | 0.660664 |
| LOR | 0.499807 | 0.53029 | 0.604471 | 0.660102 | 1 | 0.63742 | 0.382613 | 0.635199 |
| CGPA | 0.811529 | 0.807247 | 0.722389 | 0.706401 | 0.63742 | 1 | 0.506941 | 0.872403 |
| Research | 0.578914 | 0.474875 | 0.459666 | 0.400011 | 0.382613 | 0.506941 | 1 | 0.572018 |
| Chance of admission | 0.797415 | 0.789813 | 0.706071 | 0.660664 | 0.635199 | 0.872403 | 0.572018 | 1 |




a) Examinations-based features and chance of admission have high correlation (>0.78). It is justifiable because examinations require a lot of effort & are very competitive. Therefore, applications with high examination scores are selected.
Examinations-based features have a high correlation to each other (>0.80). It is justifiable because if a student is good at one examination, he should also be good at other examinations.

b) Examinations-based features like GRE Score, TOEFL Score & CGPA.

3. Recursive feature elimination (RFE) is a feature selection method that removes unnecessary features from the inputs. Start by removing one input feature the 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.

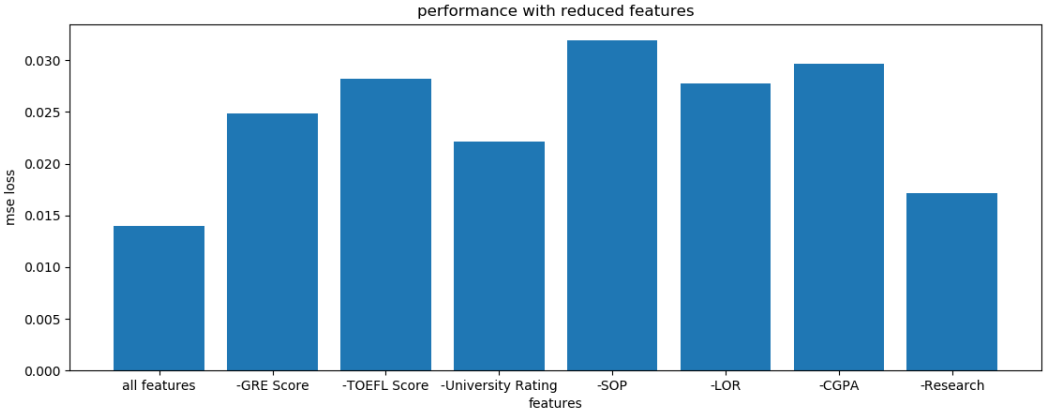
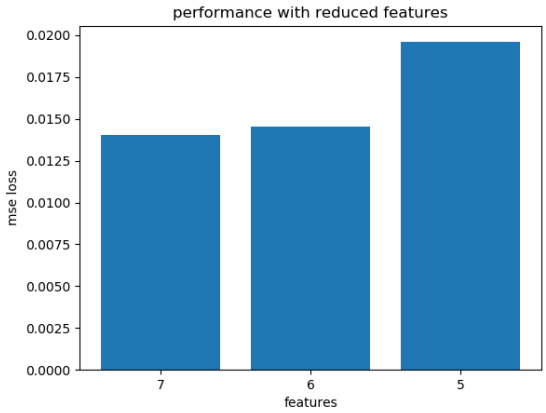



Figure 3

The reduced models with the best performance are "-Research" followed by "-University Rating". This coincides with Research and University Rating having low correlation as mentioned in Question 2. However, removing SOP and LOR also caused a big increase in loss despite having one of the least correlations.

Instead, what is common between Research & University Rating is that they have the fewest possible input space. Research is a binary input & University Rating is an integer between 1 to 5. Meanwhile SOP and LOR have twice as much input space as University Rating as they are a multiple of 0.5 between 1 to 5. This suggests that useful features have bigger input space and that input space is more important than correlation.

Therefore, our 6-feature model will remove Research feature & our 5-feature model will remove both Research and University Rating feature. After training the models, the performance for each model is shown below.




From the above figure, the model with the best performance has 7 features. However, 6-features model has comparable results. This suggests that Research is not important for admission.
Finally, the optimal feature set includes all 7 features.

4. 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.

We have trained 4 models for this question, where each model has 7 features. The figure below shows their performance according to 'mean square error loss'. Note that the 3-layer network have 10 neurons in the hidden layer.

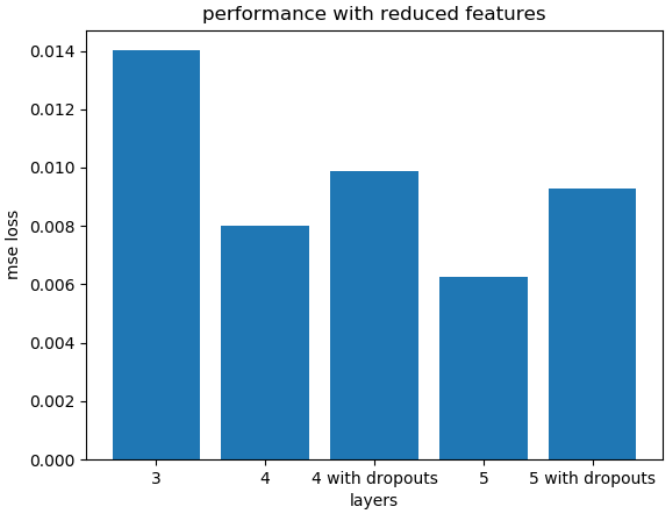



Figure 4a

The above figure shows the performance of models with different number of layers. Models with more layers have lower 'mse loss' and better performance. Unfortunately, models with dropouts have a lower performance despite reducing overfitting. We will attempt to explain why in the next section.

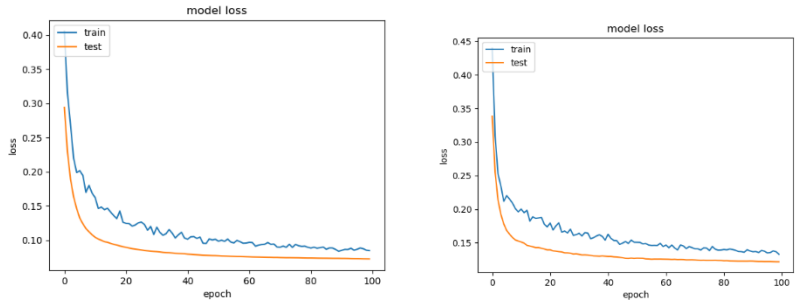



Figure 4b & 4c

Figure 4b and 4c shows the training and test loss during the training of the model. Compared to figure 1a, we can see that train loss has not reached its minimum. As such, this suggests that training is incomplete, and we must increase the number of epochs during training. We decided to try an epoch of 1000 instead of 100

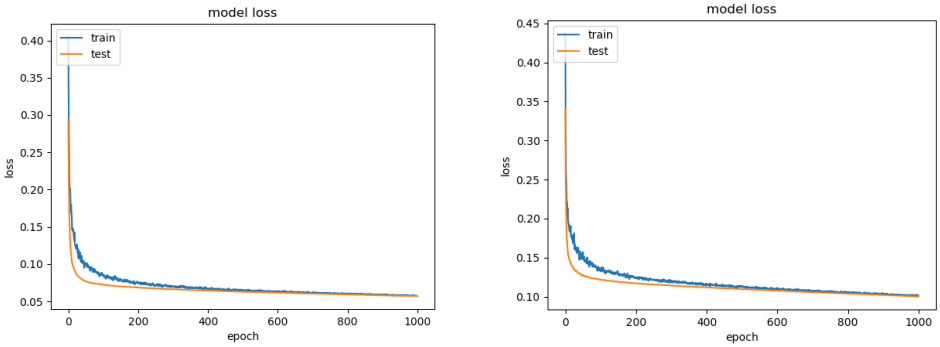



Figure 4d & 4c

Figure 4d & 4d shows the new training and test loss during training. We can see that loss has reached a minimum.

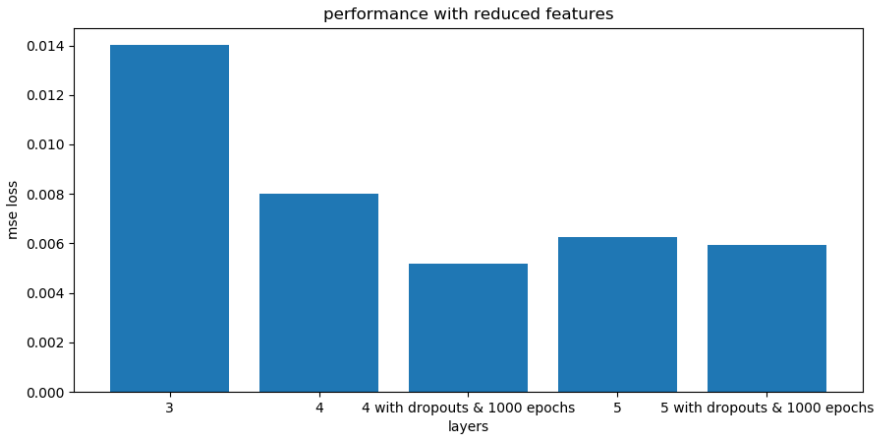



Figure 4e

Finally, Figure 4e shows the new comparison graph between models. Our 3,4,5-layer models without dropouts are unchanged from figure 4a. We can see that 4-layer model with dropouts and finished training has the best performance. Therefore, we must increase the number of epochs when using dropouts to ensure a thorough training. We should also note that trained models with 1000 epochs have slightly better performance than trained models with 100 epochs, but we decided to overlook this difference.

In conclusion, we should use a 4-layer model with 7 features & dropouts to predict admission chances.

## Conclusion

From the classification task, we find that the optimal decay parameter is 0. This could mean that the decay parameter is not suitable for this task. It could also mean that we were unlucky and our test dataset is unevenly distributed. We believe that we were unlucky as class 3 is quite rare and should have a big impact on accuracies in different datasets. We can observe this could be the case if we compare figure 3c & 4c. The test data has similar shape for both figures. Even if we were to shuffle every time we do k-fold validation, the issue persists as one dataset may end up biased due to luck.

From the regression task, we find that not all parameters are equally important. Some are even detrimental, as their effect on the output is not consistent. Parameters should occupy a bigger input space & binary parameters are less desirable.

Finally, although we have found that 4 layers is the optimal number of layers, we got to this number by trial & error. We do not know the reason why. If we were to make a guess, it is likely that shallow networks have too little space to process parameters while deep networks are difficult to train. We believe that each neural network requires a minimum depth to process all parameters. However, modern neural network techniques are unable to properly backpropagate weights to create deep neural networks effectively.