Assignment 2

CSE 574 - D
Fall 2023

Submitted by:

Akshobhya Sharma
UBIT : Akshobhy

Nazmus Saquib
UBIT: nsaquib2

Team Member	Assignment Part	Contribution(%)
Akshobhya Sharma	1,2,3,4	50
Nazmus Saquib	1,2,3,4	50

Report for Part I

Answer to Ques 1:

The dataset is numerical by nature having 766 entries and 8 variables (Including the target). Among the variables, 5 of them carry integer values, 2 with floats, and the target column holds binary data. From this we understand that it's an appropriate dataset for binary classification.

For main statistics, we see that every variable has 766 entries, with f2 having the highest mean of 120.90 and f7 having the lowest mean of 0.47 (We excluded "target" as that is binary). Column f5 has the highest standard deviation, indicating that the data points are spread out over a wider range and are more dispersed from the mean.

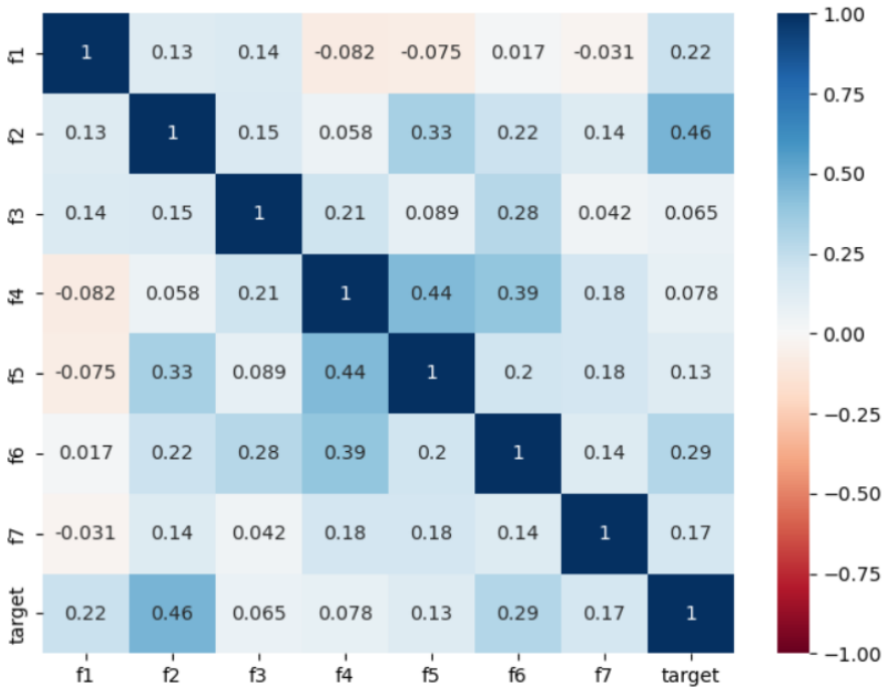
	f1	f2	f3	f4	f5	f6
count	766.000000	766.000000	766.000000	766.000000	766.000000	766.000000
mean	3.849673	120.909804	69.118799	20.542484	80.091503	31.998170
std	3.371490	31.927057	19.376901	15.950080	115.298950	7.893111
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.500000	0.000000	0.000000	27.300000
50%	3.000000	117.000000	72.000000	23.000000	36.000000	32.000000
75%	6.000000	140.000000	80.000000	32.000000	127.750000	36.600000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000

	f7	target
count	766.000000	766.000000
mean	0.472128	0.349869
std	0.331328	0.477240
min	0.078000	0.000000
25%	0.244000	0.000000
50%	0.374500	0.000000
75%	0.625500	1.000000
max	2.420000	1.000000

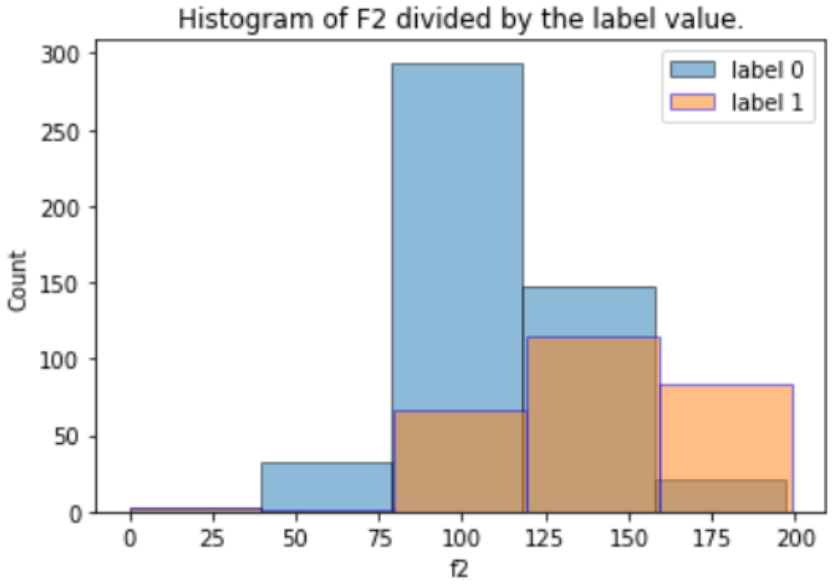
[1 0]

Answer to Ques 2:

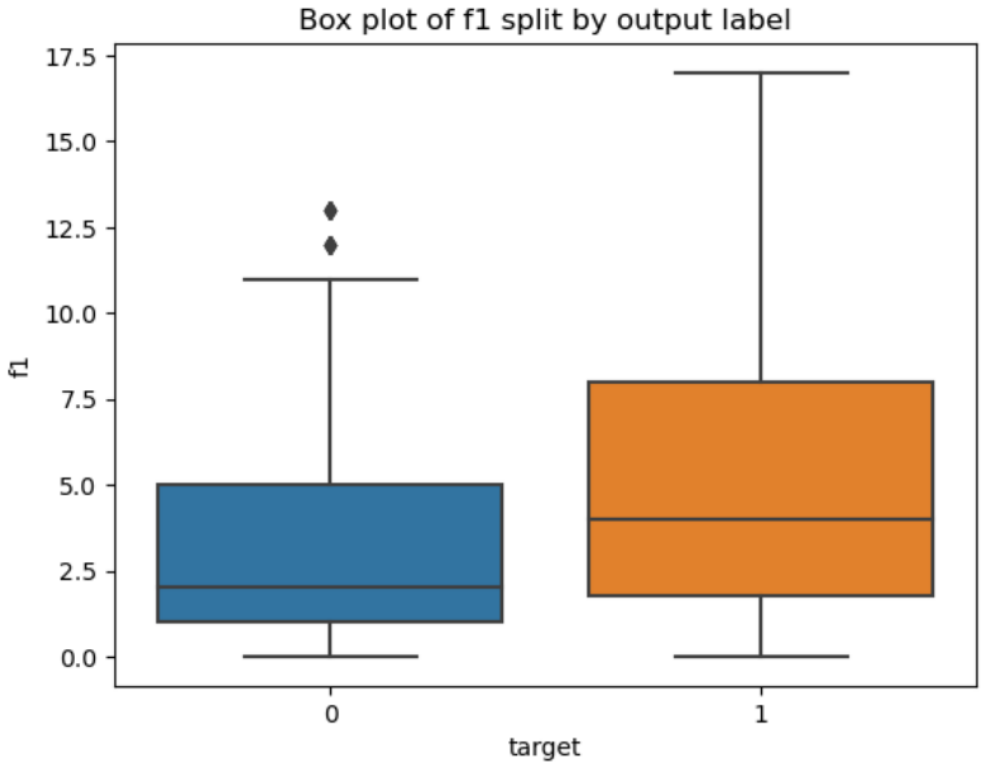
Correlation matrix:
From the correlation we found out that f3 and f2 (descending order) have good relation with the target compared to the other ones. On the other hand, f4, f5, and f7 (ascending order) have less impact on the target.



Histogram:
The histogram depicts that when f2 has values between 80-120 it most likely classifies the data as label 0 in the target. On the other hand, for f2 values from 160 to 200, they most likely label it as 1.



Box Plot:
From the box plot we understand that when the f1 value is high, it is more likely to classify the record as label 1, and the opposite for low values.



Answer to Ques 3:

For preprocessing, we used StandardScaler() from Scikit-learn. It transformed the data to have a standard normal distribution, which helped to ensure that different features had a similar impact (as much as possible) on the model and contributed to more stable and accurate model training, resulting in a better model accuracy.

Answer to Ques 4:

To summarise the model, it took 5 input parameters. The model consists of 3 hidden linear layers and 2 dropout layers, and 17025 trainable parameters.. While deeper networks are more prone to overfitting, dropout layers can help mitigate this problem. During training, dropout layers deactivate a subset of neurons at random, driving the network to acquire more robust and generalizable features. This can lead to improved performance on previously unknown data.

```
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
CustomNeuralNet                          --                        --
├─Linear: 1-1                            [1, 64]                   384
├─Dropout: 1-2                           [1, 64]                   --
├─Linear: 1-3                            [1, 128]                  8,320
├─Dropout: 1-4                           [1, 128]                  --
├─Linear: 1-5                            [1, 64]                   8,256
├─Linear: 1-6                            [1, 1]                    65
==========================================================================================
Total params: 17,025
Trainable params: 17,025
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 0.02
==========================================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.07
Estimated Total Size (MB): 0.07
```

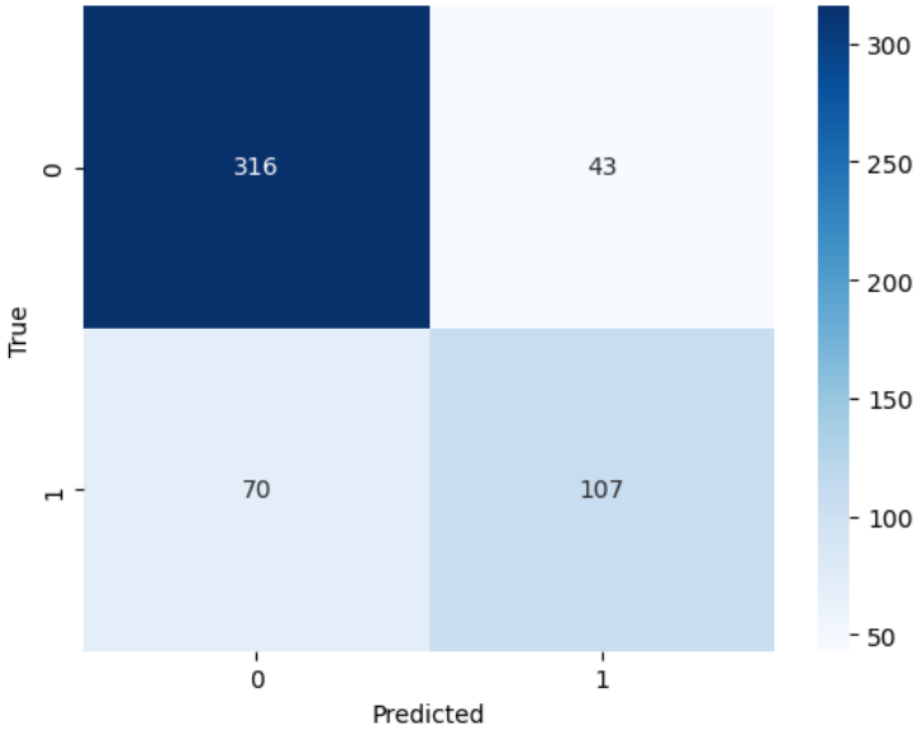
Answer to Ques 5:

Our analysis achieved the highest accuracy of 78.91% with the current model, with a precision, recall, and f1 score of 0.71, 0.60, and 0.65 consecutively.

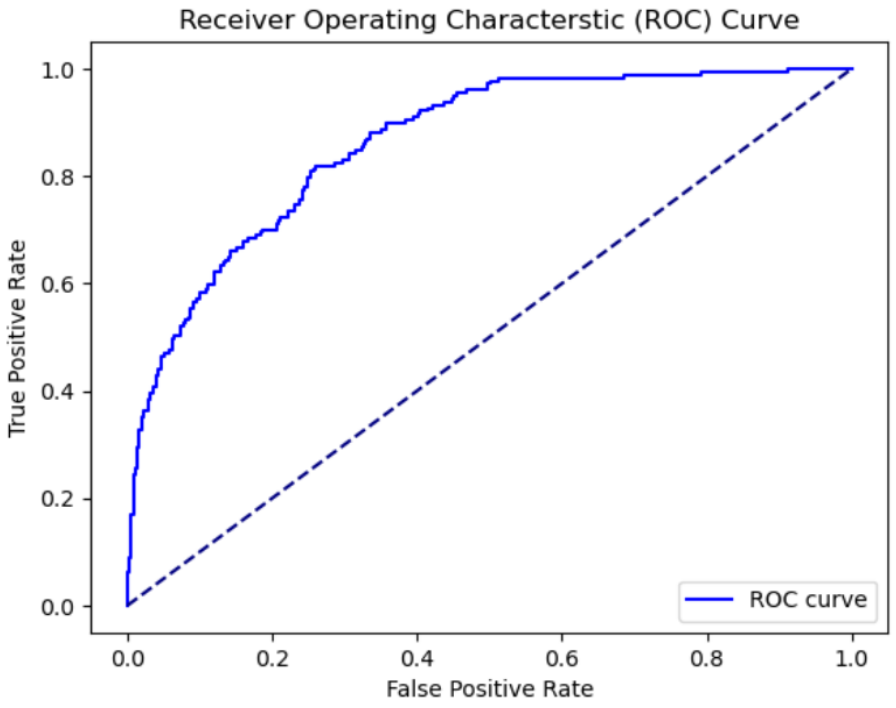
```
Testing Accuracy 0.789179104477612
Precision Score 0.7133333333333334
Recall Score 0.6045197740112994
f1 score 0.6544342507645261
Training Time 4.424584150314331
```

Answer to Ques 6:

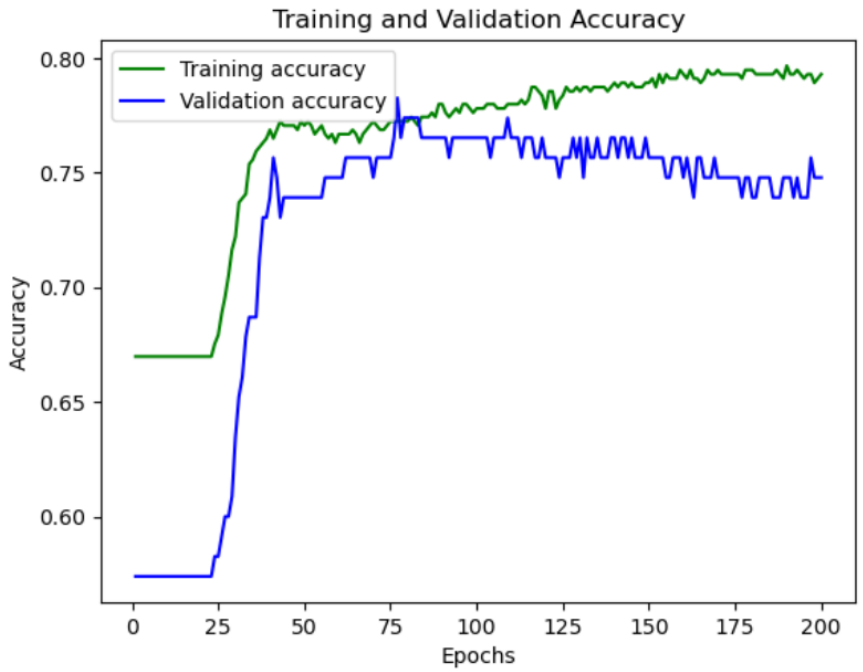
The below heatmap visualisation shows that our model predicts a good amount of true values.



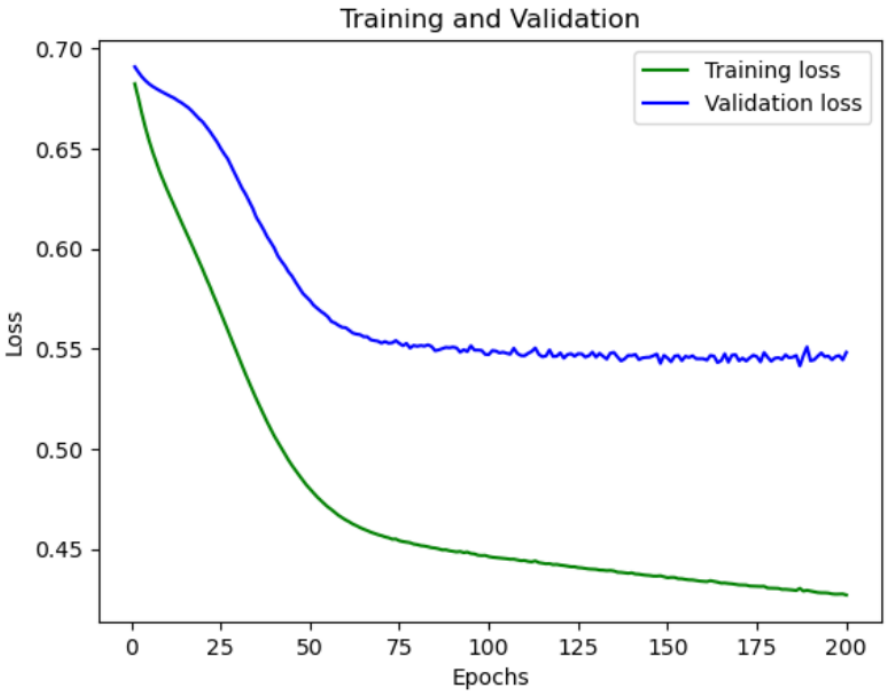
The below ROC curve depicts that our model is providing a high true positive value.



The validation accuracy measures how well a model adapts to unseen data, which is separate from the data used for training. The below graph shows that although our validation accuracy is a little bit less accurate than the training one, it is still good and close to 75% after 200 iterations.



The validation loss in the below graph measured how well the model performed on a separate dataset (not the training one). Here, our training loss decreased significantly but the validation loss remained high. It indicates overfitting, meaning that our model has learned to fit the training data closely than we wanted it to.



Report for Part II

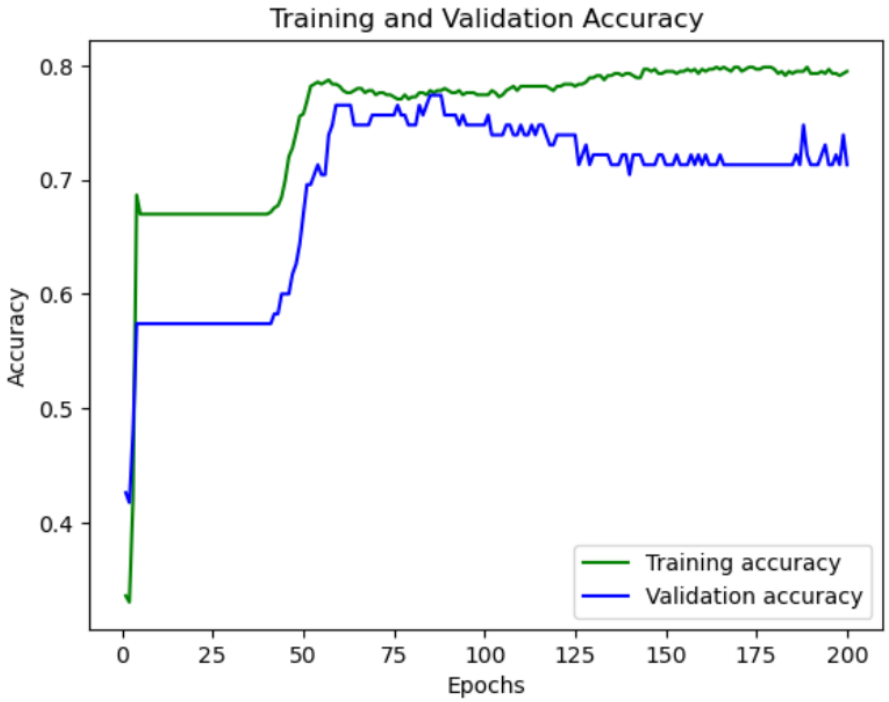
Answer to Ques 1:

	Setup 1	Test Accuracy	Setup 2	Test Accuracy	Setup 3	Test Accuracy
Dropout	0.5	0.791	0.2	0.794	0.2	0.8003
Optimizer	SGD		SGD		SGD	
Activation Function	ReLU		PReLU		ReLU	
Initializer	default		default		xavier_uniform	

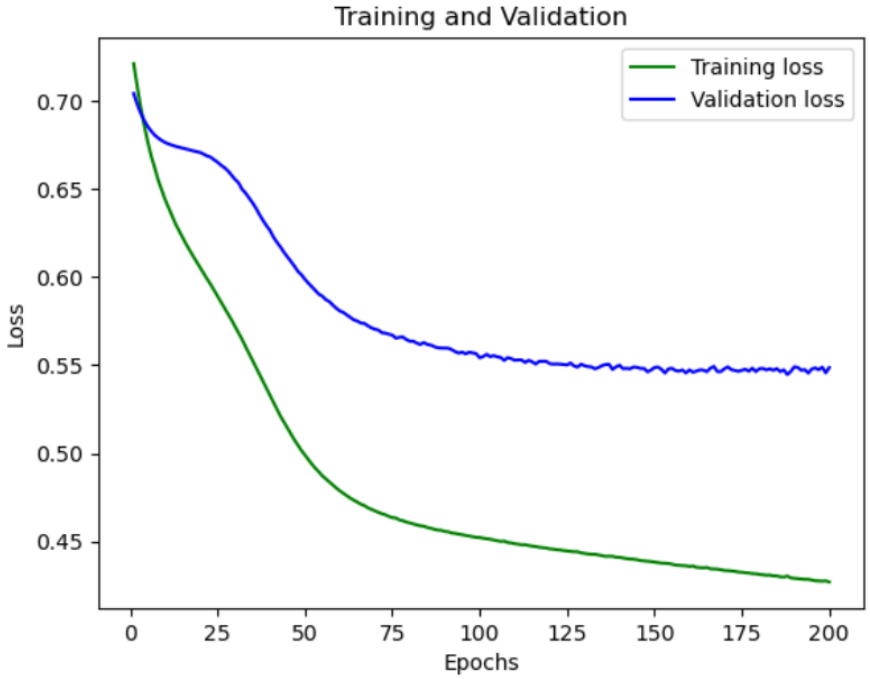
Answer to Ques 2:

Set up 1:

The below Training vs Validation graph shows that although our validation accuracy is less accurate than the training one, it is still fine and ranging between 0.70- 0.75 after 50 iterations.

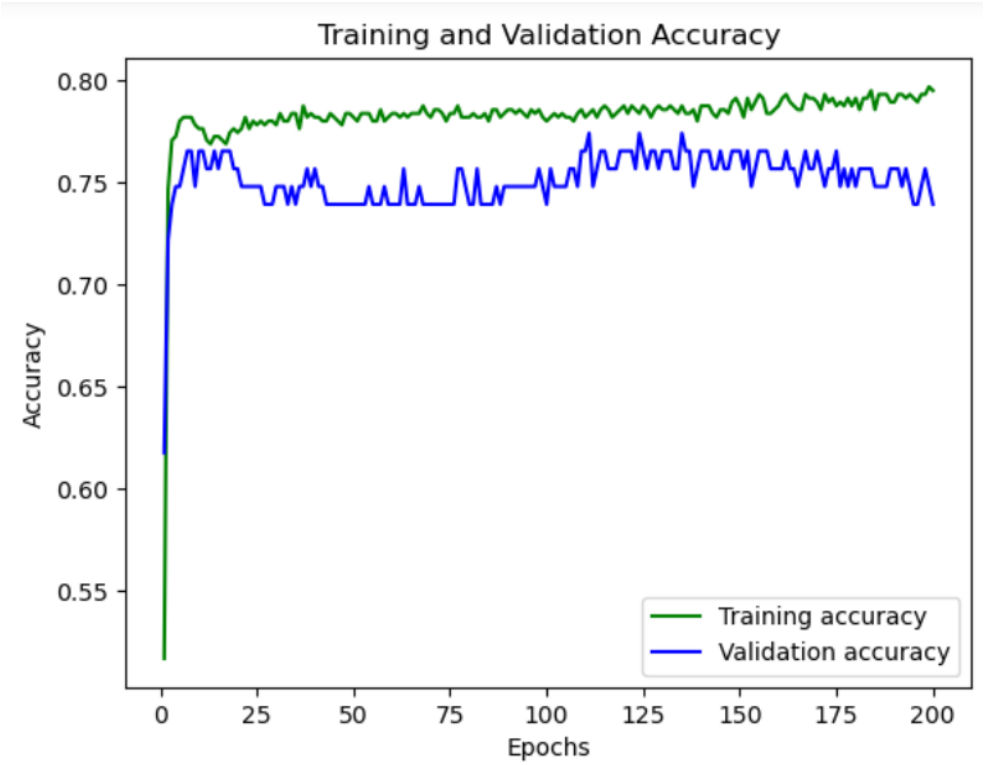


The training vs validation loss in the below graph, our training loss decreased significantly but the validation loss remained high. It indicates overfitting, meaning that our model has learned to fit the training data closely than we wanted it to.

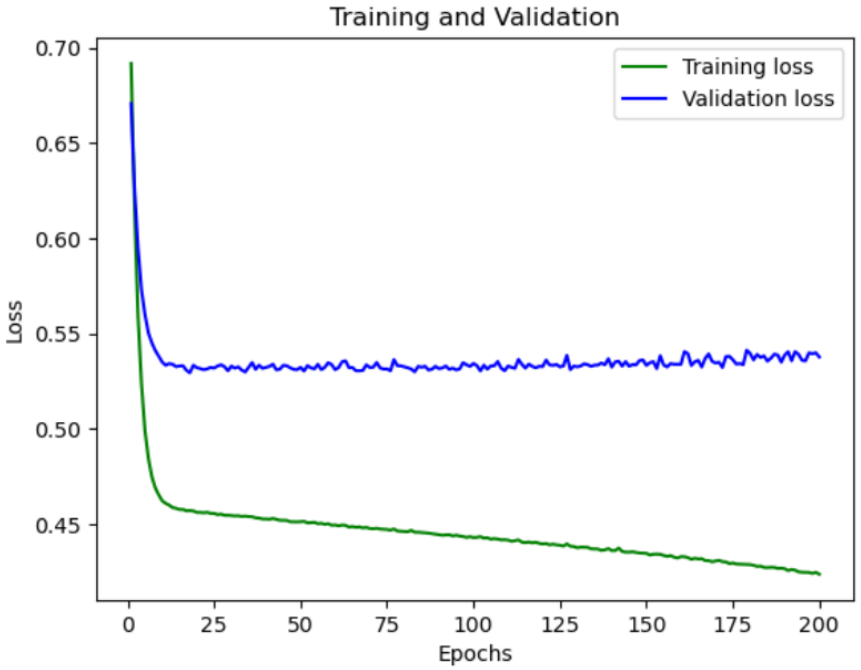


Set up 2:

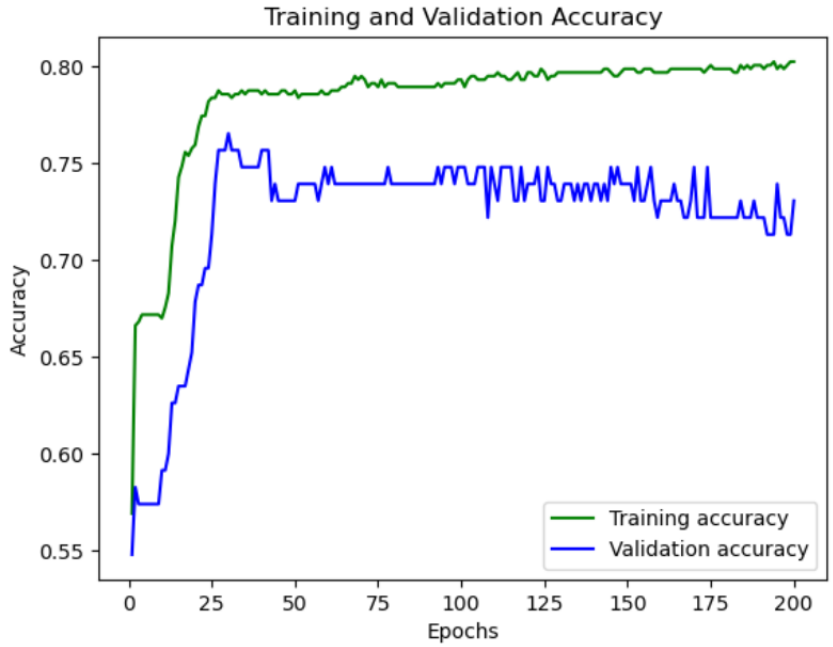
The below Training vs Validation accuracy graph shows that although our validation accuracy is less accurate than the training one, it is still fine and fluctuating near 0.75 almost from the beginning of iterations, also the consistency is good.



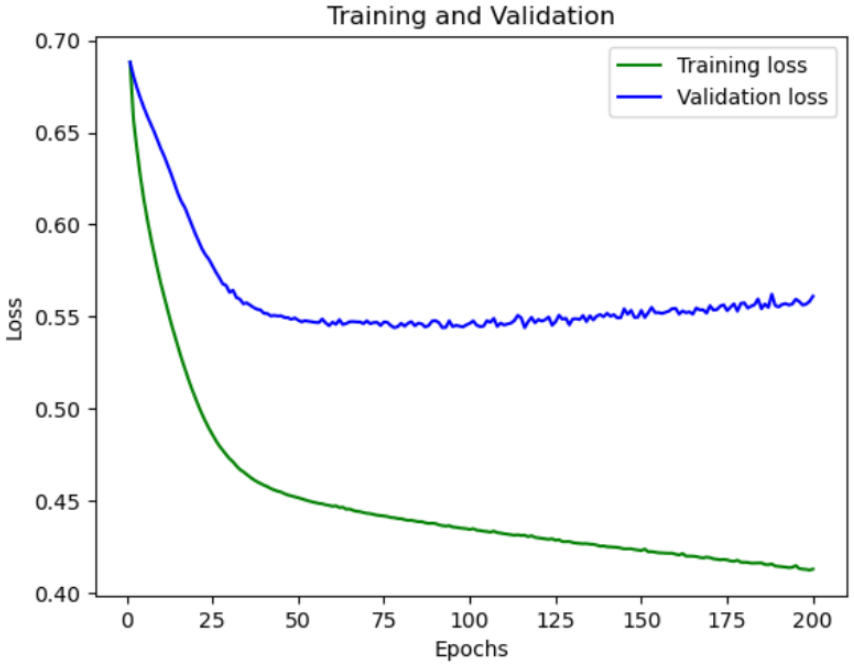
The training vs validation loss in the below graph, our training loss decreased significantly but the validation loss remained high. It indicates overfitting just as the previous set up. After a point almost at the beginning, the loss value is not decreasing unlike the training one.



Set up 3:
The below Training vs Validation accuracy graph shows that although our validation accuracy is less accurate than the training one, it is still fine and fluctuating between 70 to 75. However it's not improving as the iterations go forward unlike the training accuracy. Also the consistency is not very good.



The training vs validation loss in the below graph, our training loss decreased significantly but the validation loss remained high. It indicates overfitting just as the previous two set ups. After 25-50 iterations, the loss value is not decreasing unlike the training one.



Answer to Ques 3:

We have tried 3 setups in total:

Setup 1
Setup 1 uses 0.5 dropout and SGD as the optimization algorithm. It uses ReLU as the activation function and uses the default initializer for ReLU which is the kaiming_uniform initializer. The dropout of 0.5 causes the neural network to set 50% of the neurons to 0 during each forward pass of the training. This increases the dropout rate in the previous model. The idea is to turn off a higher fraction of neurons in the training to prevent any single neuron from being overly specialised or reliant on other neurons thus increasing the chance of the model to overfit. Due to increased dropout, our models become more specialised.

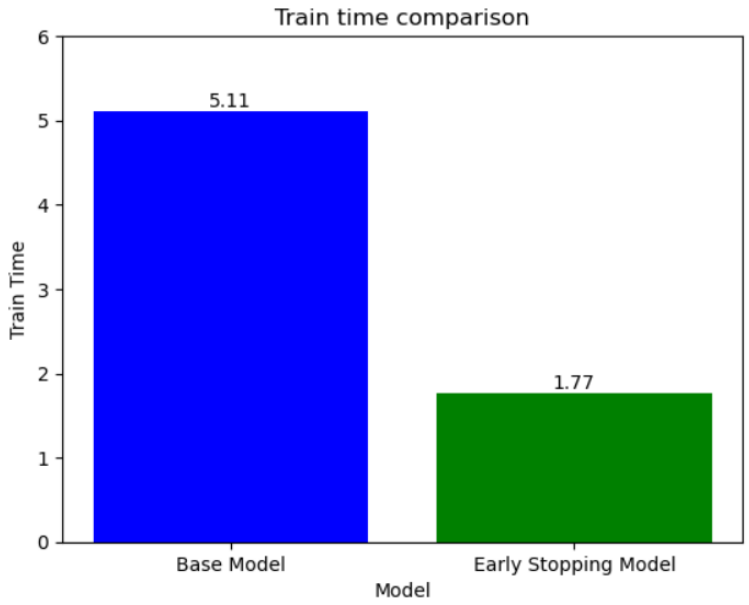
Setup 2
Setup 2 uses 0.2 dropout rate, SGD as optimization algorithm. It uses PReLU as the activation function and uses the default initializer for PReLU activation function. Compared ro ReLU activation function, in which negative values are converted to 0 after activation, PReLU has a learnable parameter that adjusts the slope in the negative part of the input space. This provides more adaptability to the model and can foster more effective learning.

Setup 3
Setup 3 uses 0.2 dropout rate, SGD as optimization algorithm. It uses ReLU as the activation function and uses the Xavier_uniform initializer for initialising the weight values. This initialization method aims to keep the scale of the gradients roughly the same in all the layers of the network. This balanced scaling of the gradient prevents exploding or vanishing gradient problems found in early stages of the neural network training.

Answer to Ques 4:

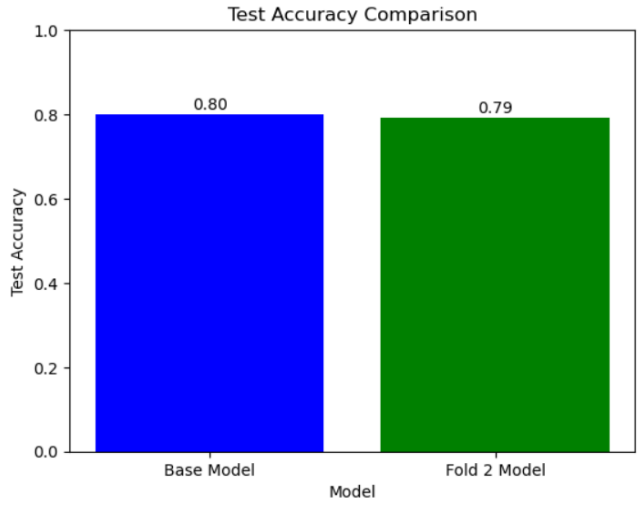
Methods used to improve the accuracy:

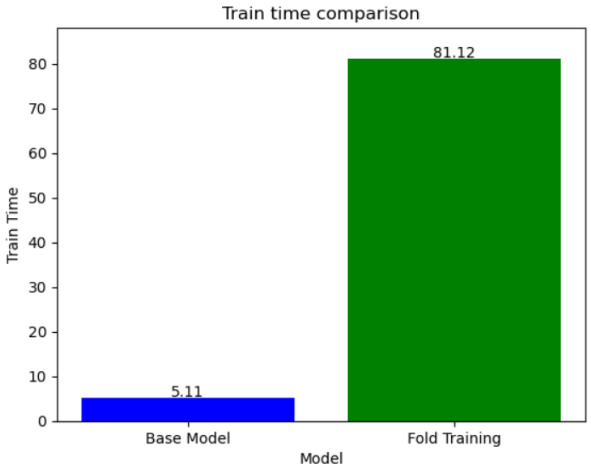
1. Early stopping
The early stopping method essentially has a buffer for validation error. It keeps track of the minimum validation error and a counter for the number of times the validation error has increased. The idea is that when the models are decreasing in testing error but increasing in validation error, the model is overfitting. Early stopping prevents this by prematurely ending the number of iterations of the model. This is the first method used to decrease the training time.



2. K-fold training

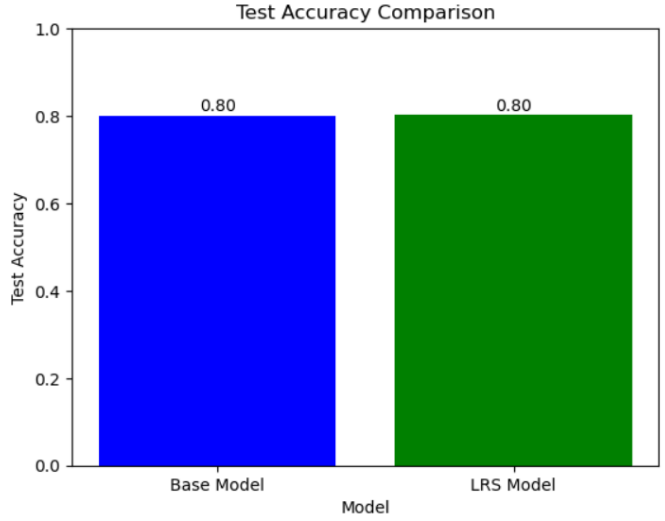
K-fold cross-validation is employed to ensure that a machine learning model performs consistently across various subsets of the dataset. It is useful when you have a limited dataset to split into training and testing and also to give the performance of a model on a more holistic dataset. It can reduce bias and give better overall performance, but in our case it has failed to do so. However, increasing the number of folds has the disadvantage of increasing the training time of the model as the iteration for training has to be done for each model for each number of folds.

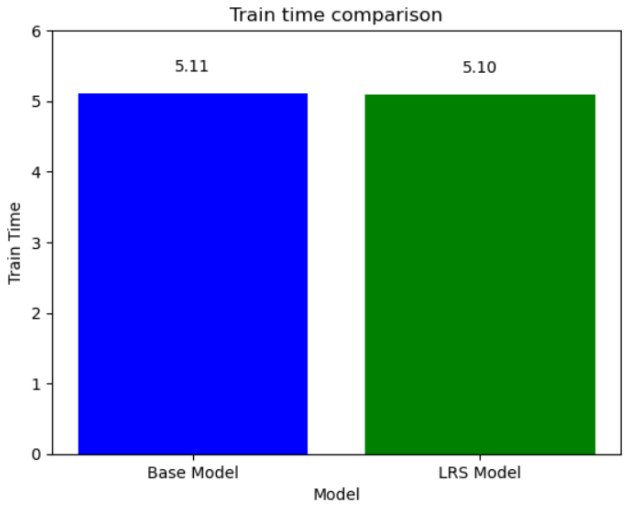




3. Learning rate scheduler

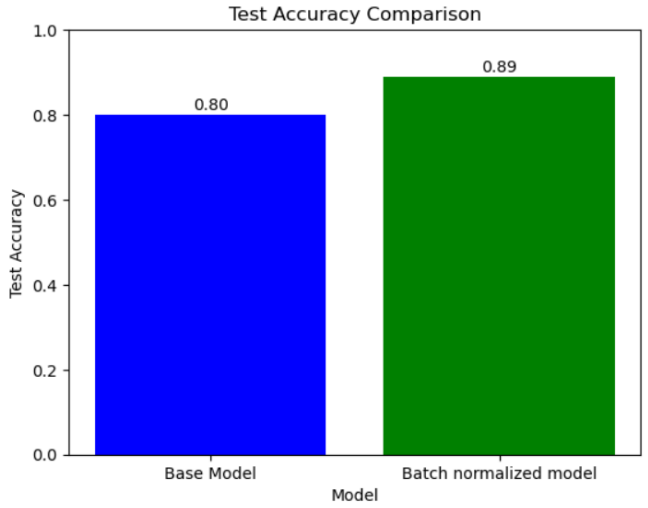
Learning rate scheduler basically decreases the size of the learning rate the closer it gets to the minimum of the function, this causes the learning rate to identify more accurately the minima of the function and can lead to faster convergence. In our case although it doesn't improve accuracy that much, we can see a very slight increase in the training time.

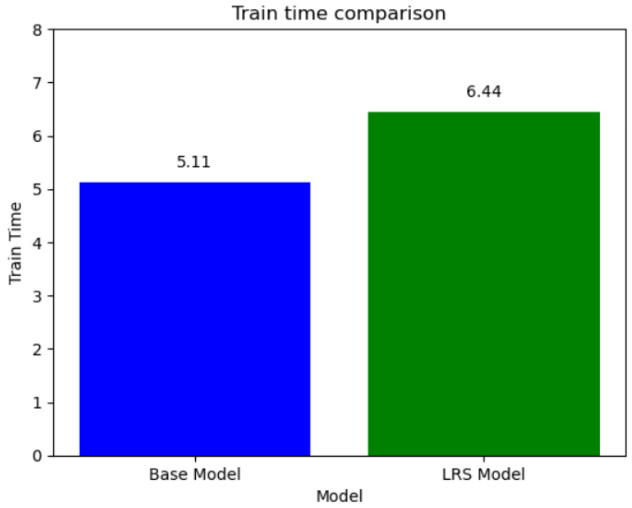




4. Batch normalisation

Batch normalisation by standardising the inputs to a layer during training, helps in achieving faster convergence and reduces the sensitivity to the initial weight's initialization. In our model, when batch normalisation was added we can see a good improvement in the model performance in the training dataset as well.





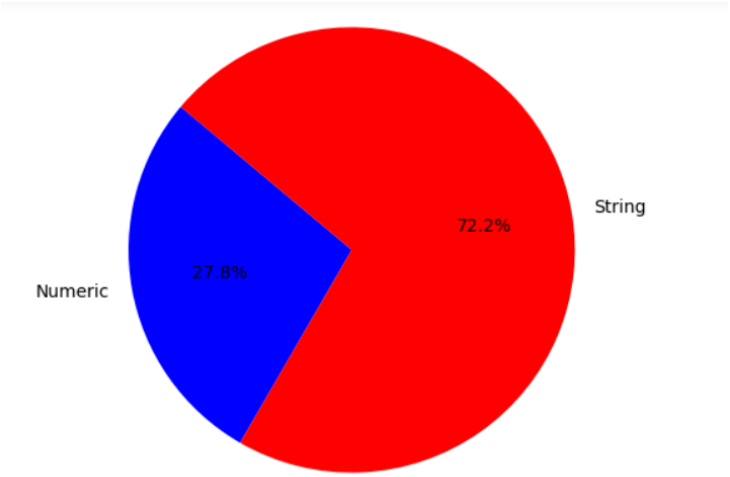
Report for Part III

Answer to Ques 1:

The CNN dataset consists of 36 categories of English alphabets and digits with 2800 examples for each category. The number of samples in total is 100800. The dataset we encountered is made of image based data with each example having a 28x28 image. The mean and standard deviation of pixel values of the entire dataset are 0.1758 and 0.3296 respectively.

Answer to Ques 2:

The dataset we have comprises images of 27.8% digits and 72.2% strings (letters).



Answer to Ques 3:

The model is built with 4 types of layers. Below are the details of each layer.

Convolutional Layers:
conv1: 3 input channels (assumed RGB), 32 output channels, kernel size (3, 3), padding (1, 1)
conv2: 32 input channels, 64 output channels, kernel size (3, 3), padding (1, 1)
conv3: 64 input channels, 128 output channels, kernel size (3, 3), padding (1, 1)

Pooling Layer:
pool: Max pooling with a kernel size of 2x2 and a stride of 2.

Fully Connected Layers:
fc1: Input features = 6272, Output features = 512
fc2: Input features = 512, Output features = 128
fc3: Input features = 128, Output features = 36 (assuming 36 classes for classification)

Dropout Layer:
dropout: Dropout with a probability of 0.5
This model is for a classification task with 36 classes. The input size is expected to be (batch_size, 3, 28, 28), assuming 28x28 pixel images with three colour channels (RGB).

Answer to Ques 4:

For our model the batch normalisation operated by normalising the input of each layer in a mini-batch to have zero mean and unit variance. This normalisation is applied independently to each feature in the mini-batch.

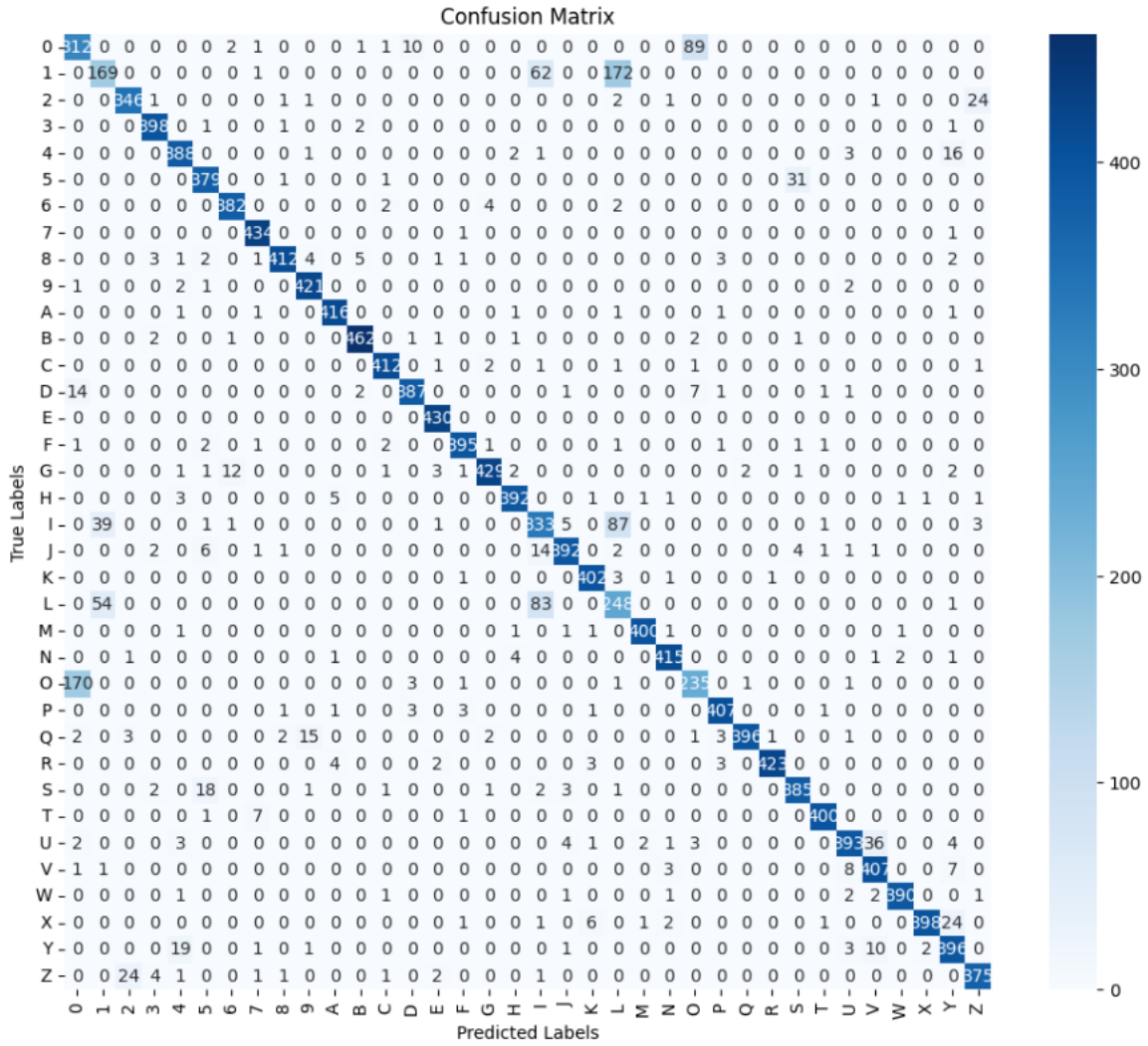
Answer to Ques 5:

The accuracy of the model is: 91%
The training time of the model is: 504.213 seconds

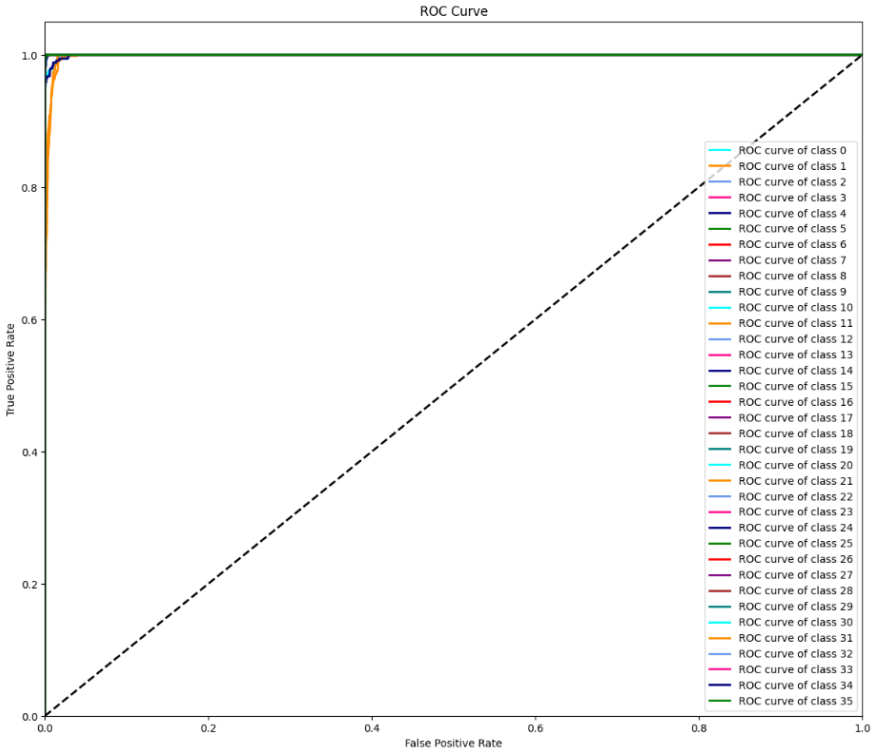
The precision of the model is: 0.9130113886070095
The recall of the model is: 0.9099867724867725
The f1 of the model is: 0.9099567496508928

Answer to Ques 6:

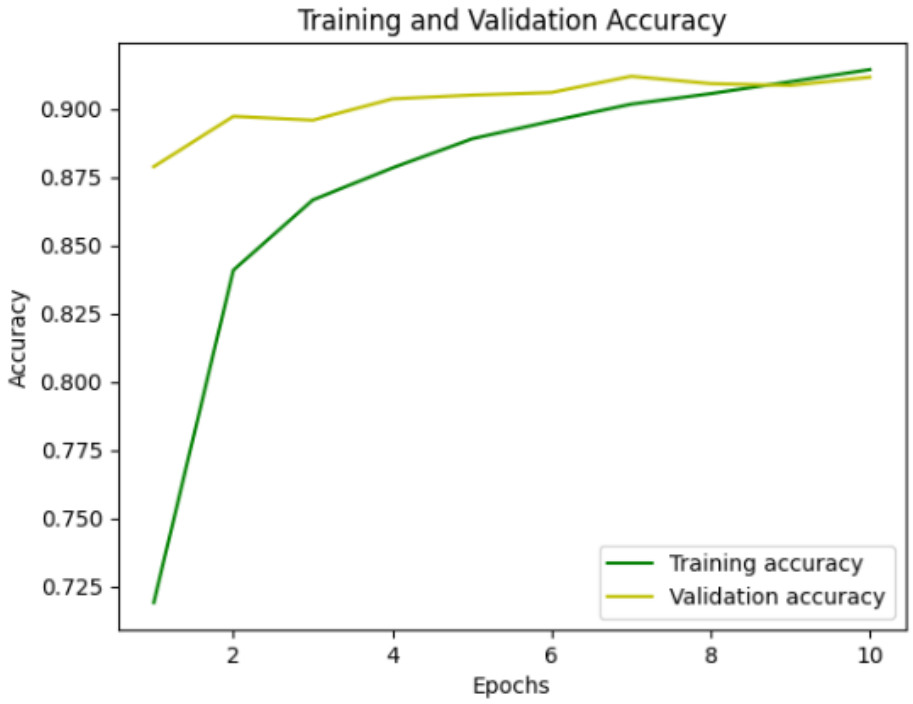
From the confusion matrix we get to know that our model is giving pretty accurate predictions with very low errors. The noticeable error is that sometimes the model is predicting “1” to be “L” and “O” to be “0” (Zero).



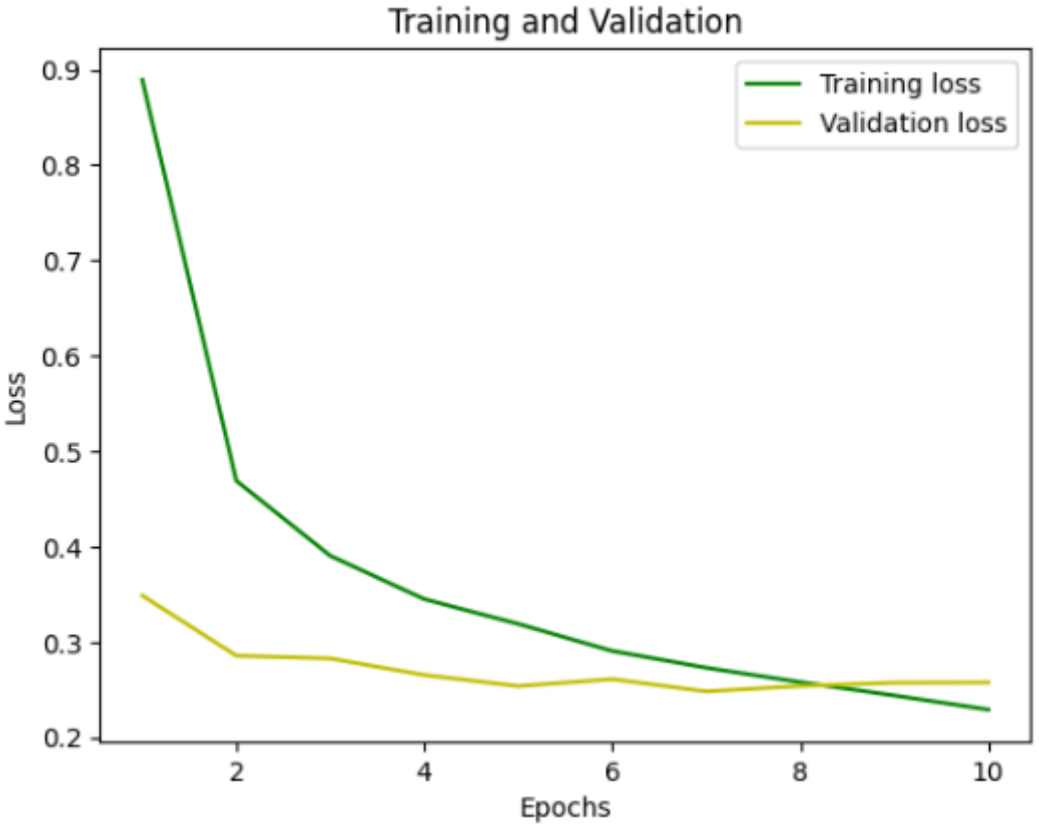
We used the ROC curve to evaluate the performance of our model. For all the classes it is over 0.9 which are pretty good scores.



In the “training vs validation accuracy” graph we see that with every epoch our accuracy is going up. The validation accuracy is fluctuating a little bit but that is negligible.



In the “Training vs validation loss” graph, both the training loss and validation loss are going down with iterations. Although the validation loss is fluctuating a little bit, that is negligible because the loss value is in our favour.



Report for Part IV

Answer to Ques 1:

The model closely resembles the VGG-11 architecture described in the instruction. It uses a series of 33 convolutional layers followed by 22 max pooling. The convolution layer uses 1 padding and 1 stride to prevent loss of resolution. The VGG net architecture was originally meant for 224224 size images. However, the input that we are using is 2828, because of this using the five pooling layers as described in the manual would cause loss of image resolution to 11 by the fourth pooling layer itself, due to this reason we have removed 3 pooling layers from our model so that we can have 77 feature map from the last pooling layer. Following the pooling layers, the fully connected layers are implemented according to the architecture provided.

Answer to Ques 2:

In our VGG-11 architecture, we have more nodes in fully connected layers and also there are more convolutional layers than part 3. For this reason, the VGG-11 model took much more time than it took in the model of part 3.

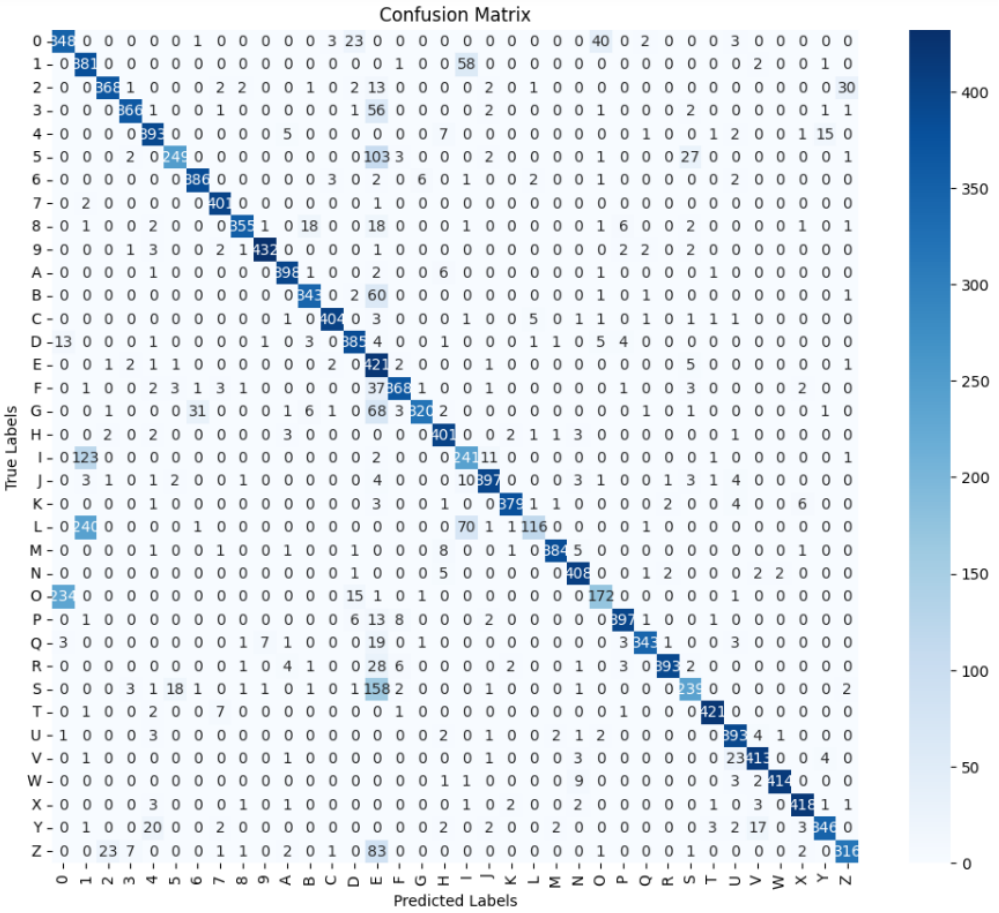
Answer to Ques 3:

	Part 3	Part 4
Accuracy %	91%	85.37%
Training time	504.21 seconds	1210.17 seconds
Precision	0.913	0.8879
Recall	0.9099	0.8537
F1	0.9099	0.8561

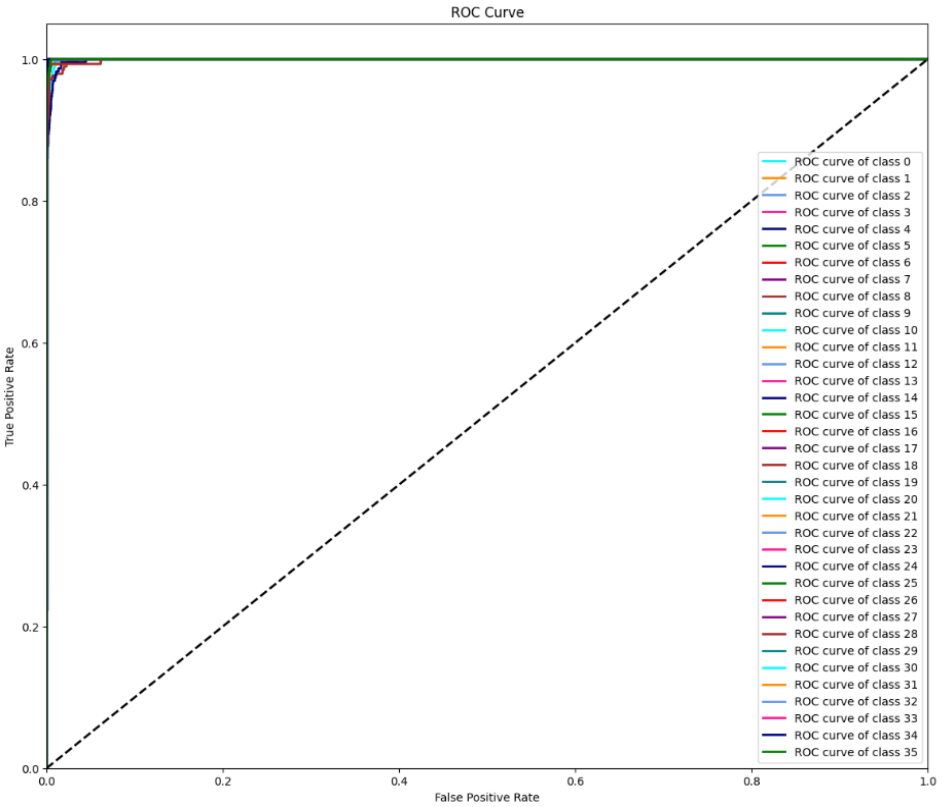
We can see that we have better accuracy in part 3 than in part 4, because we trained the model in part 3 for more number of iterations.

Answer to Ques 4:

From the confusion matrix we get to know that our model is giving pretty accurate predictions with very low errors. The noticeable error is that sometimes the model is predicting “1” to be “L”, “O” to be “0” (Zero), and “S, 5, Z” to be “E”.



We used the ROC curve to evaluate the performance of our model. For all the classes it is over 0.9 which are pretty good scores.



In the “training vs validation accuracy” graph we see that with every epoch our accuracy is going up. The validation accuracy is fluctuating a little bit but it is getting better with higher epochs.

In the “Training vs validation loss” graph, both the training loss and validation loss are going down with iterations. Although the validation loss is fluctuating a little bit, that is negligible because the loss value is in our favour.