CSE 574 - Intro Machine Learning

Assignment 3 Report

Instructor: Alina Vereshchaka

Name: Chaitanya Desai & Shahrukh Khan

Team: 110

Date: 04/20/2022

Academic Integrity

**"I (We) certify that the code and data in this assignment were generated independently, using
only the tools and resources defined in the course and that I (we) did not receive any external
help, coaching or contributions during the production of this work."**

Assignment 3

I (We) certify that the code and data in this assignment were generated independently, using only the tools and resources defined in the course and that I (we) did not receive any external help, coaching or contributions during the production of this work.

Chaitanya Desai – 50373880 (cdesai@buffalo.edu)
Sharukh Khan – 50425020 (skhan36@buffalo.edu)

Part I: Building a Basic NN

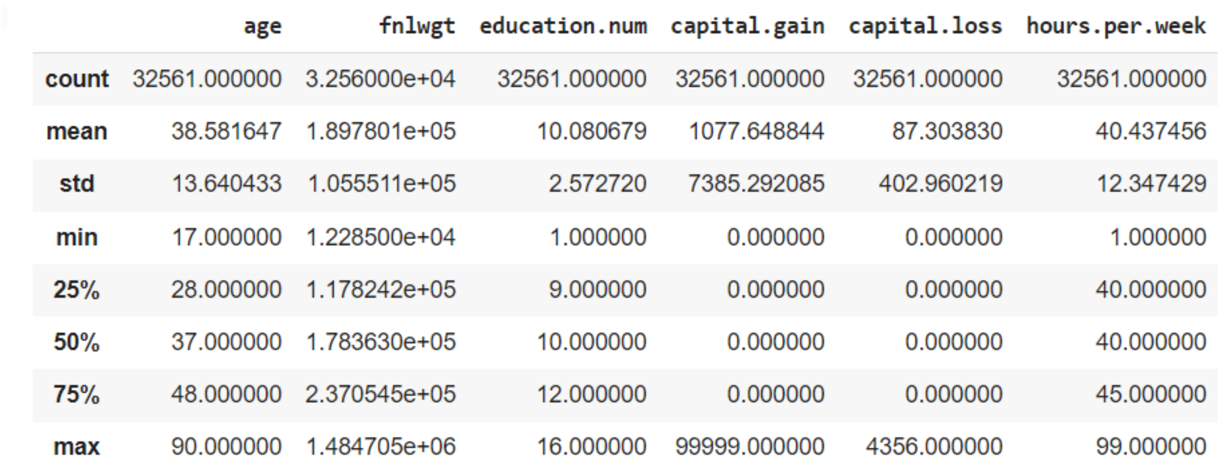
Q1) **Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? How many entries and variables does the dataset comprise? Provide the main statistics about the entries of the dataset.**

Income dataset give us the details about an individual's age, education, marital status etc. Additionally, it will give us details about hours worked by him/her per week, their country, and its capital income etc. Finally, it will give us the detail of the income. Is it greater than 50k or less than 50k.

Here the data we encounter includes integer values, string data type. Adding to it the data set is not optimized as it includes lot of Nan values and " ? ". This tells us the data set has some errors or the values are missed or not present. For this purpose, we must carry out preprocessing. We have used various steps as mentioned in our file(code) for preprocessing. This helps us optimize our dataset and increase the accuracy of our data set.

The Dataset comprises of 32561 rows (entries) and 13 columns (variables)

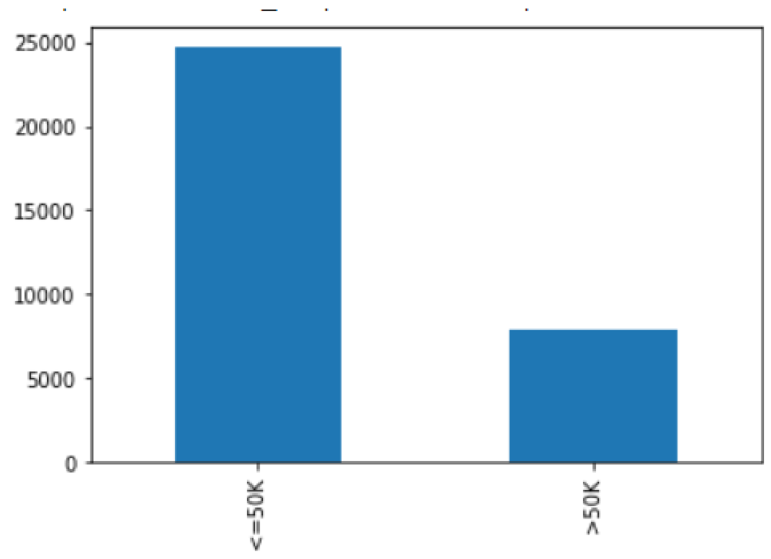
The main statistics of our dataset are shown below



	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	32561.000000	3.256000e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897801e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055511e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178242e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783630e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370545e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

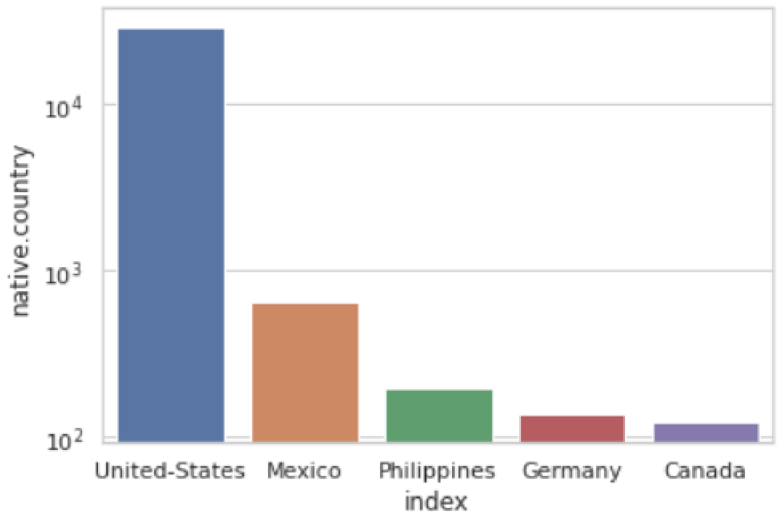
Q2) **Provide at least 3 visualization graphs with short description for each graph.**

a) Graph1: The following graph gives us some information about the dataset



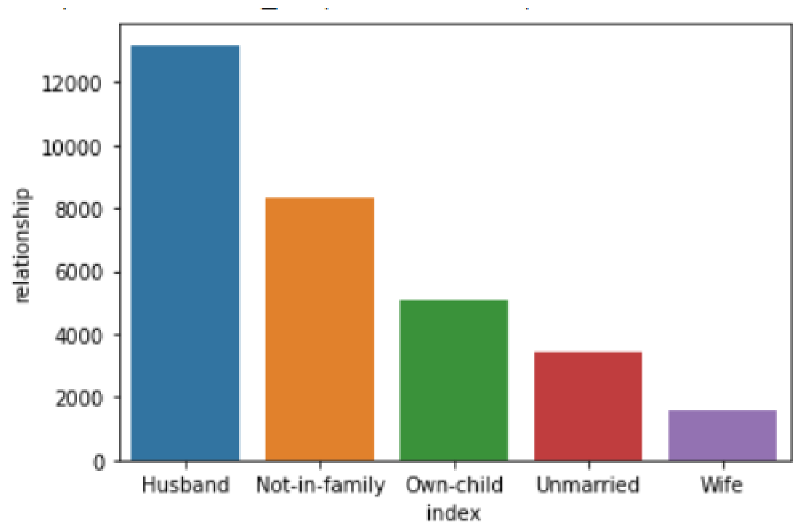
The above graph shows us the total number of people having income greater than or equal to50K and less than 50K.

b) Graph2:



The above graph gives us the information about top 5 countries and number of people respectively. As the number of people in United States were far more compared to other countries, we selected the logarithmic conversion of the values.

c) Garph3:

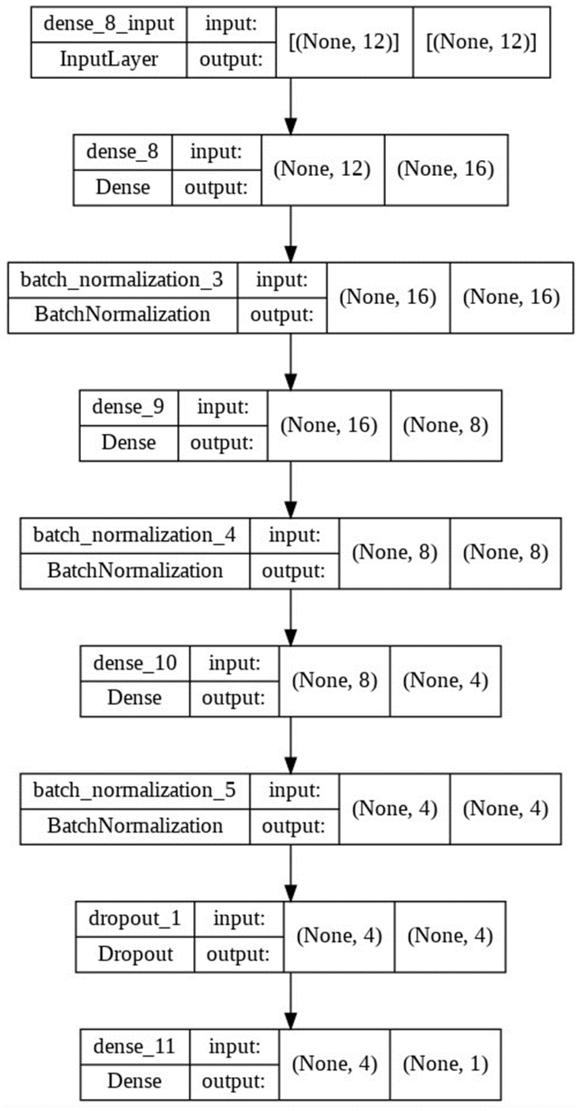


The above graph tells us the number of people in respective relationship. It tells us how many are related as husband, how many are unmarried etc.

Q3) **For the preprocessing part, discuss if you use any preprocessing tools, that helps to increase the accuracy of your model.**

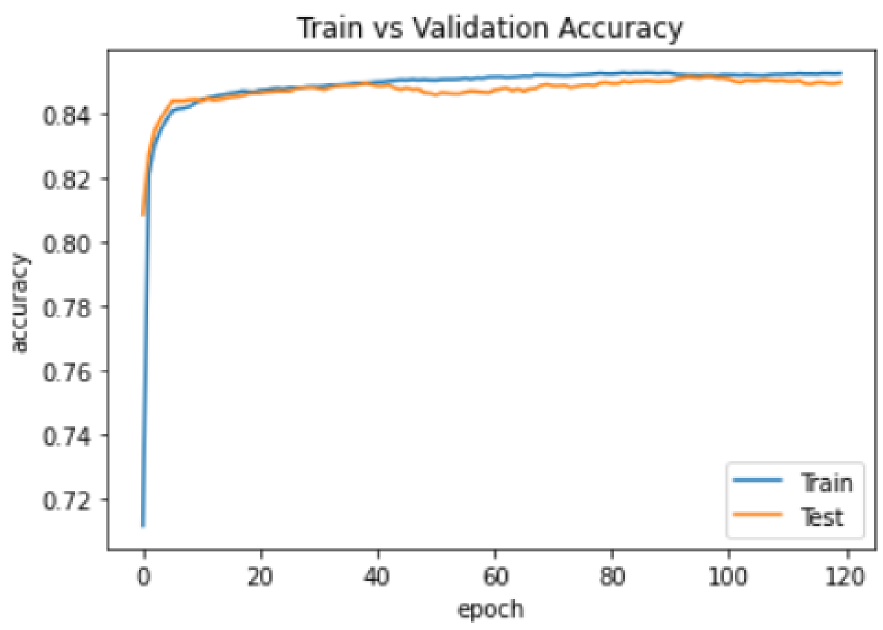
For our preprocessing part we have carried out removal of nan/ unidentified values. Later we normalize the larger values. The preprocessing parameters are necessary to optimize the dataset and our accuracy has increased after carrying out the preprocessing.

Q4) **Provide the architecture structure of your NN.**

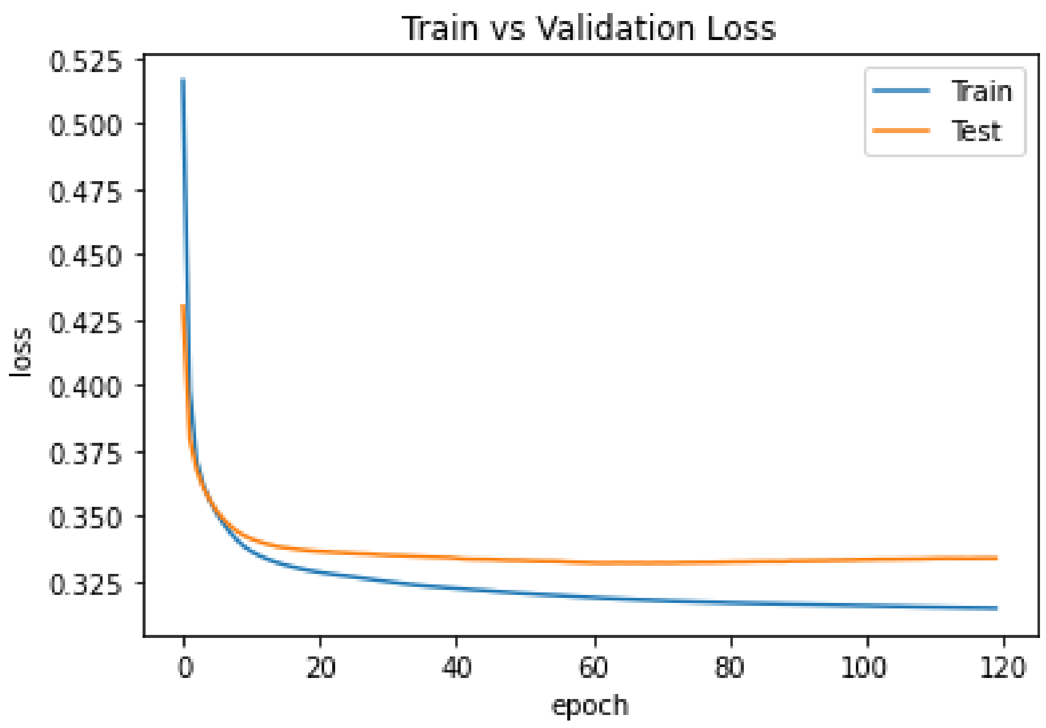


Q5) **Provide graphs that compares test and training accuracy on the same plot, test and training loss on the same plot. Thus in total two graphs with a clear labeling.**

Train vs test graph for accuracy



Train and test graph for loss



Part II: Optimizing NN

Q1) Include all 3 tables with different NN setups.

Here we add dropout considering values 0.2, 0.3 and 0.4. From this we conclude dropout value of 0.4 is best suited for our model.

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.2	85.38%	0.3	85.27%	0.4	85.15%
Optimizer	adam		adam		adam	
Activation Function	relu		relu		relu	

In this part keeping best dropout value found in above step we change the parameters like activation function etc and note accuracy for every iteration.

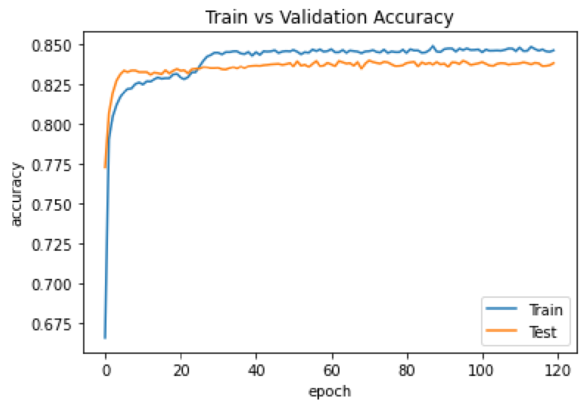
Here we get increased accuracy of 85.38% by using adam optimizer, tanh activation function and MSE loss function.

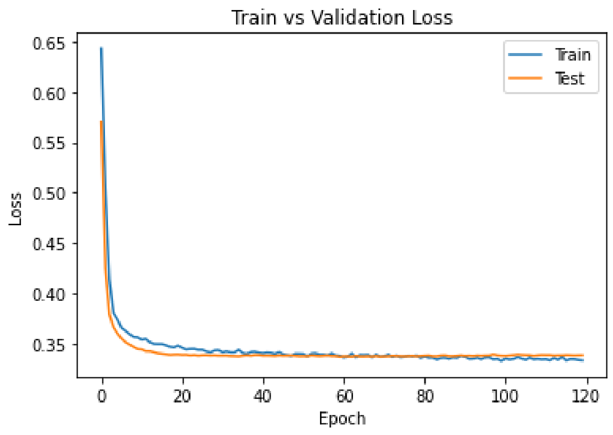
	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.2		0.2		0.2	
Optimizer	RMSprop	85.16%	Adadelta	76.95%	Adagrad	80.63%
Activation Function	relu		relu		relu	

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.2		0.3		0.4	
Optimizer	RMSprop		RMSprop		RMSprop	
Activation Function	relu	85.37%	softplus	85.87%	softsign	85.3%

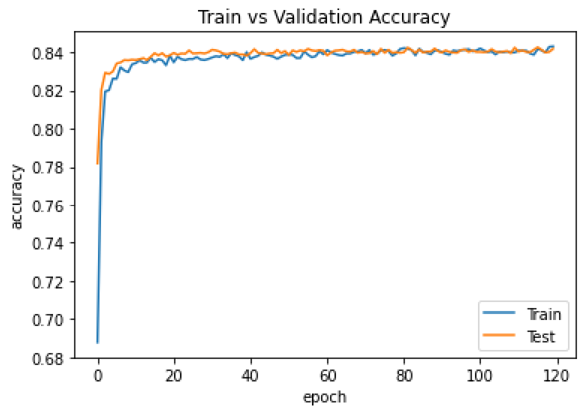
Q2) **Provide graphs that compares test and training accuracy on the same plot for all your setups and add a short description for each graph.**

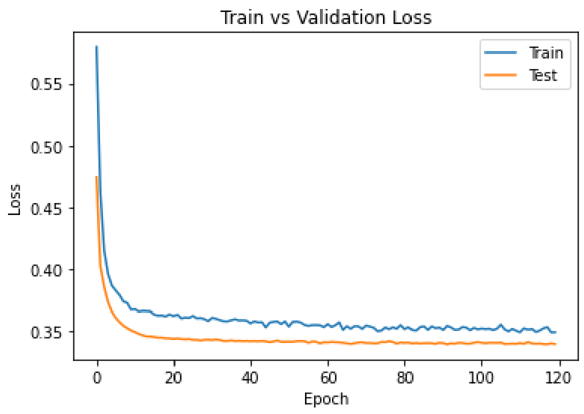
Setup1: Dropout = 0.2



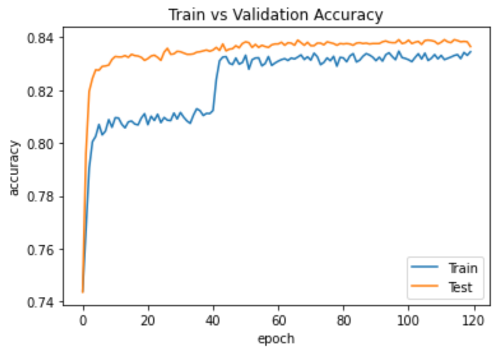


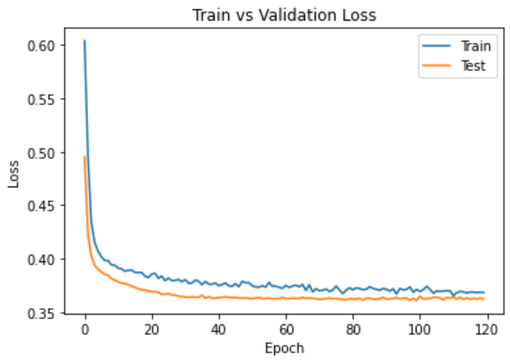
Setup2: Dropout = 0.3



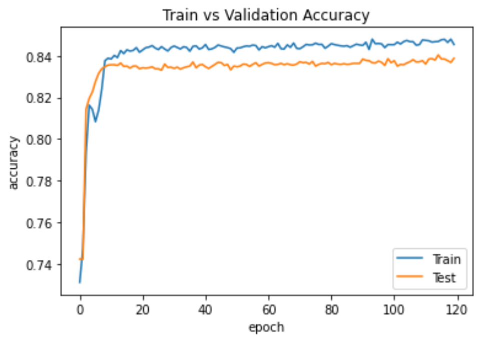


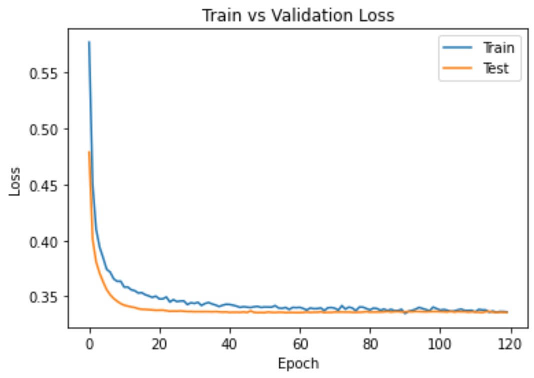
Setup3: Dropout = 0.4



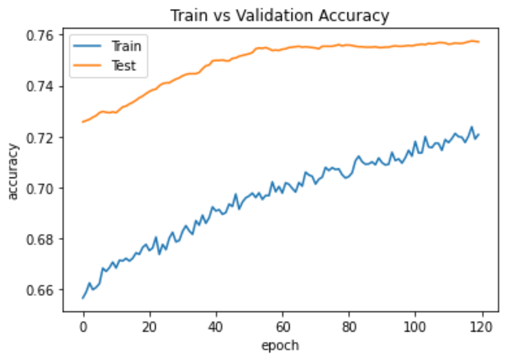


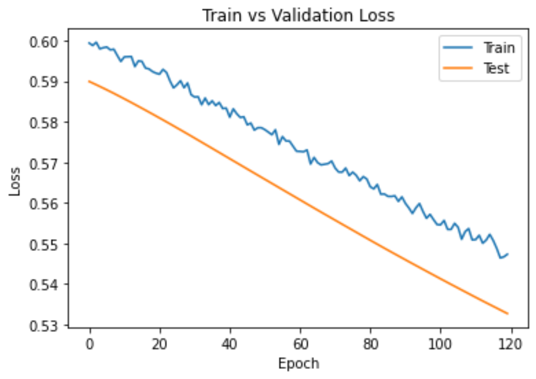
Setup1: Optimizer – RMSprop



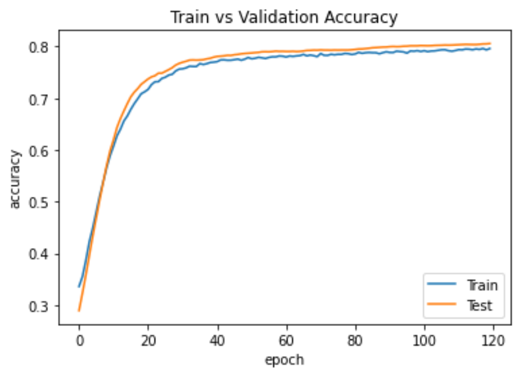


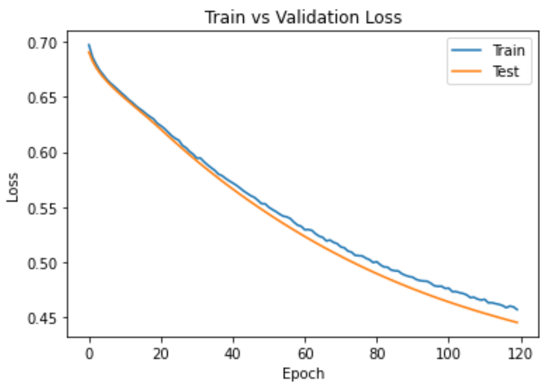
Setup2: Optimizer – Adadelta



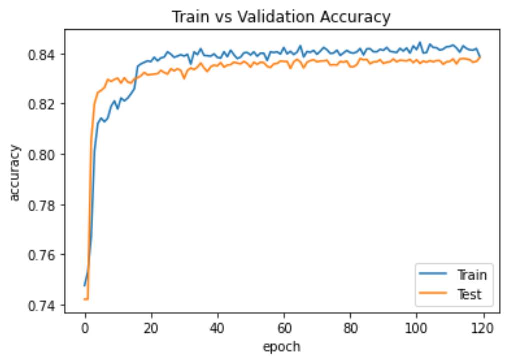


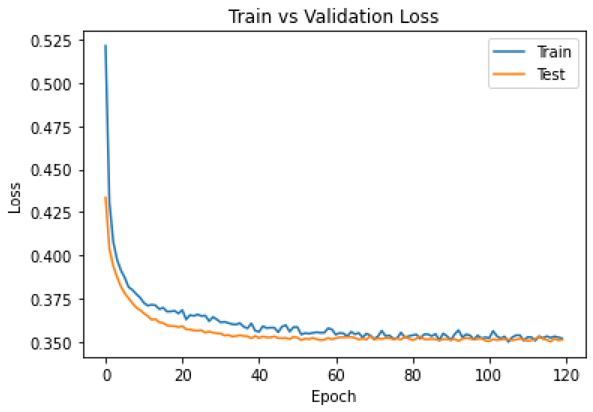
Setup3: Optimizer – Adagrad



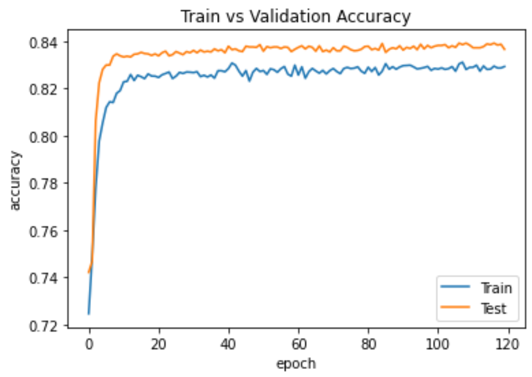


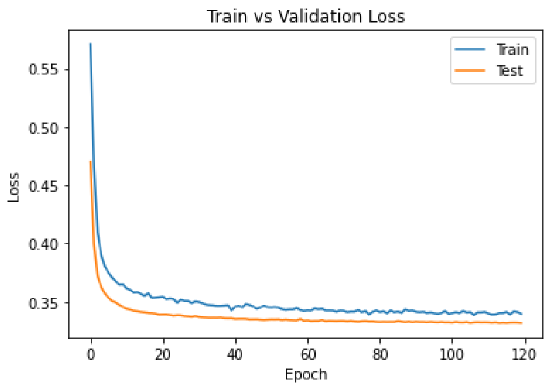
Setup1: Activation Function – Relu



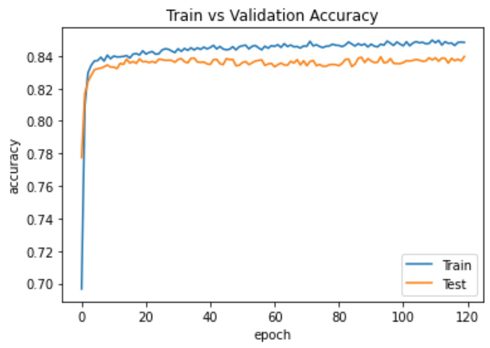


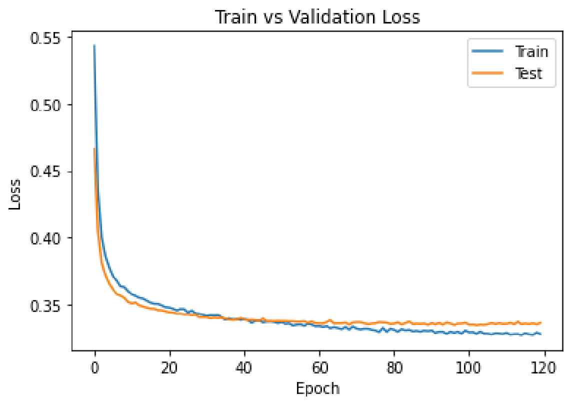
Setup2: Activation Function – Softplus



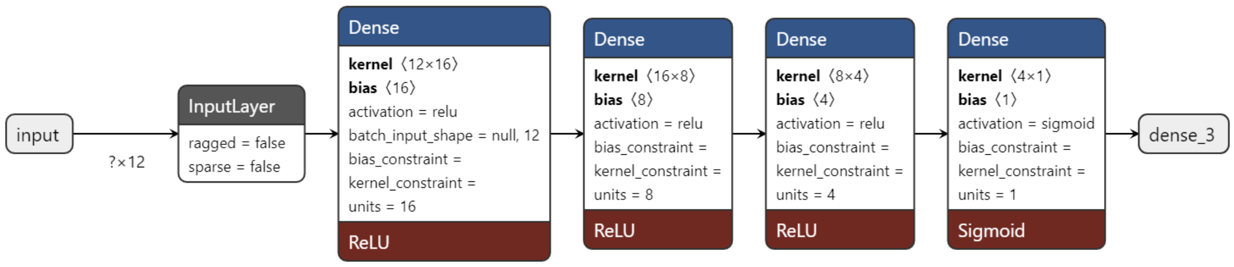


Setup3: Activation Function – Softsign





Q3) **Provide a detailed analysis and reasoning about the NN setups that you tried.**



Architectural structure of NN model.

Firstly, we use 80% values of dataset for training purpose and 20% values for testing purpose.

For first hidden layer we use 16 Neurons with input dimensions as 12 as we have 12 features for consideration. Activation function used is relu.

For second hidden layer we use 8 Neurons with input dimensions as 12 (same for all). Activation function used is relu.

For third hidden layer we use 8 Neurons with input dimensions as 12 (same for all). Activation function used is relu.

For final layer we have activation function used is relu.

The baseline model which was selected has following contents

Dropout = 0.2

Optimizer = RMSprop

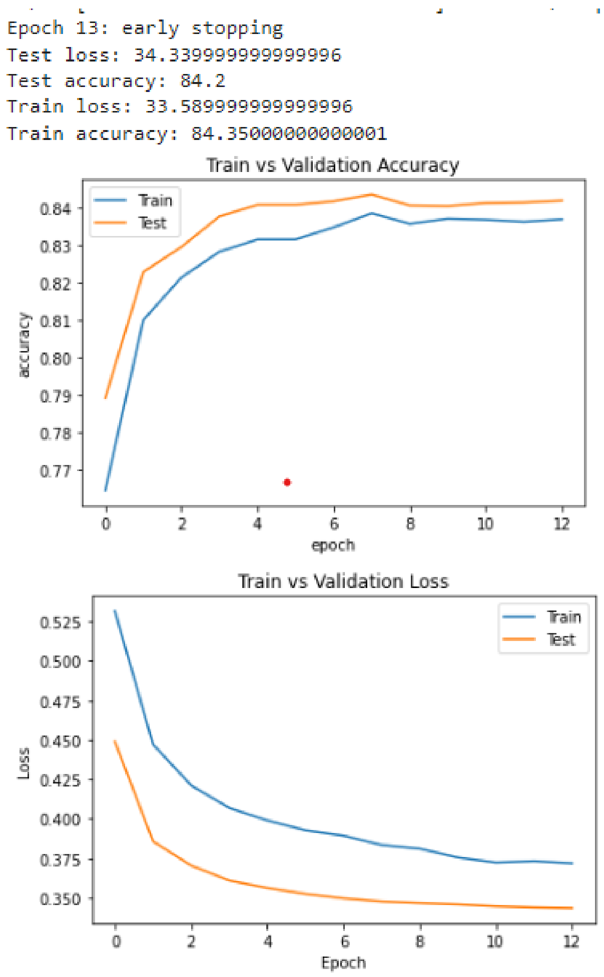
Activation Function = Relu

Loss function = Binarycrossentropy

Output activation function = sigmoid

Q4) Briefly discuss all the methods you used that help to improve the accuracy or training time. Provide accuracy graphs and your short descriptions.

Method1: Early stopping

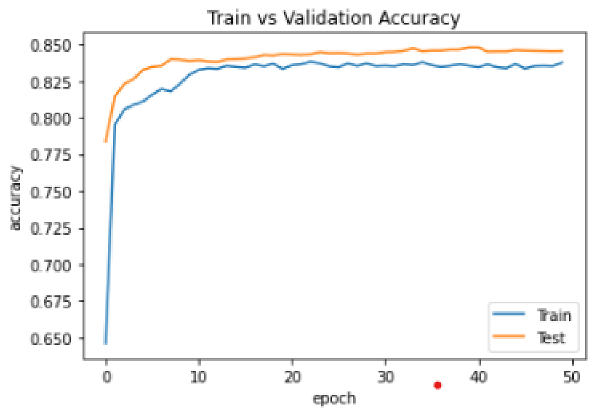


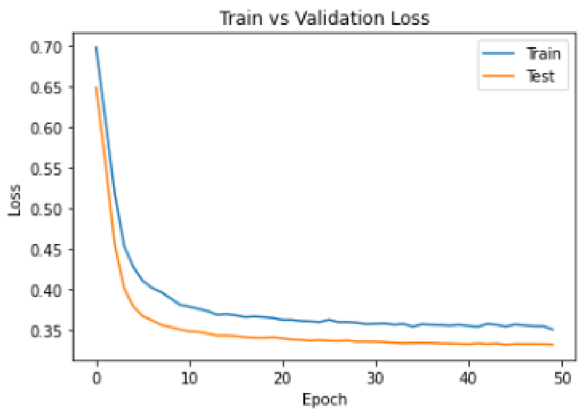
```
Epoch 13: early stopping
Test loss: 34.339999999999996
Test accuracy: 84.2
Train loss: 33.589999999999996
Train accuracy: 84.35000000000001
```

Early Stopping: Using early stopping, you may define an arbitrarily high number of training epochs and have the training process end as soon as the model performance on the validation dataset stops improving.

Method2: K fold : Number of Folds - 5

```
Test loss: 33.17
Test accuracy: 84.55
Train loss: 32.31
Train accuracy: 84.68
```

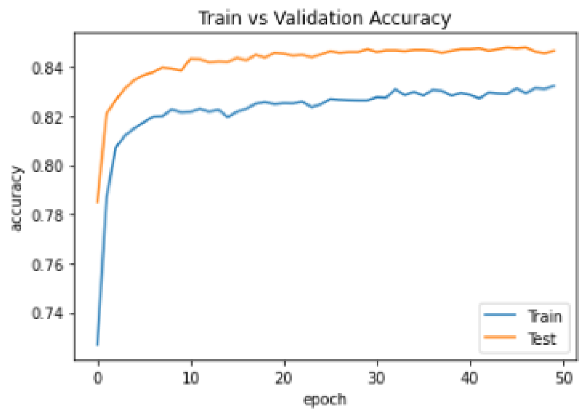


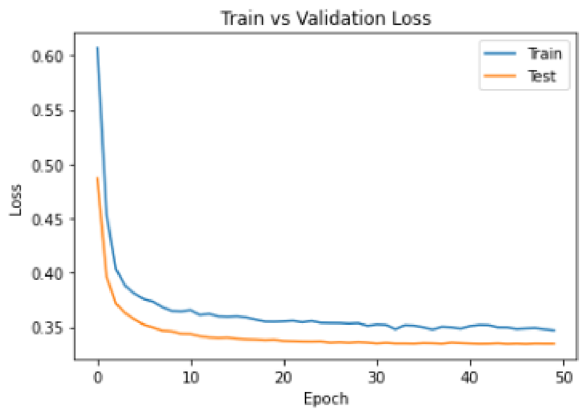


K Fold: If K=5, it signifies that we are partitioning the dataset into 5 folds and conducting the Train and Test. During each run, one fold will be considered for testing, while the remainder will be used for training and carrying on with iterations. The diagram below depicts the flow of the fold-defined size.

Method3: ReduceLRonPlateau

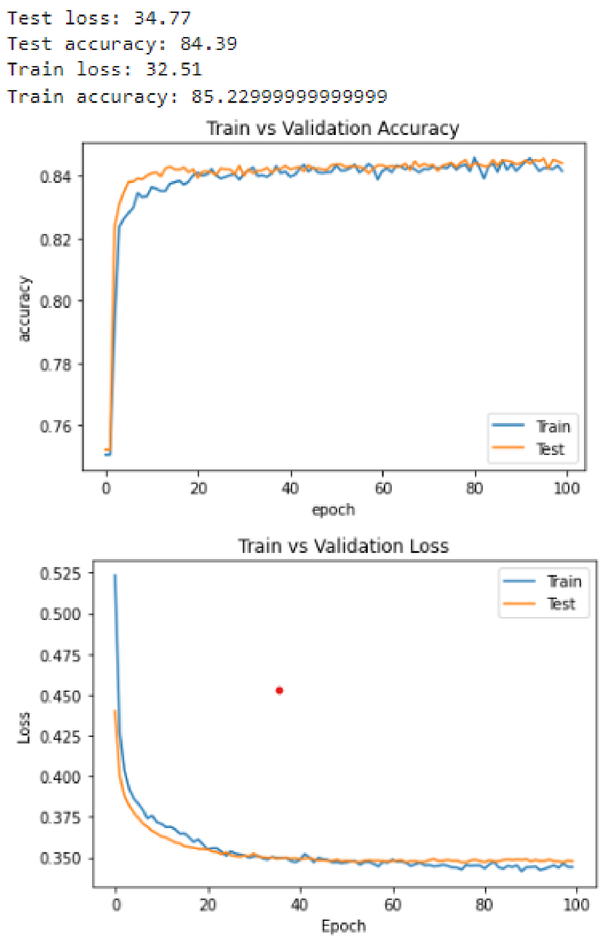
```
Test loss: 33.45
Test accuracy: 84.67
Train loss: 32.18
Train accuracy: 85.02
```





ReduceLRonPlateau: Keras also has a callback called ReduceLROnPlateau that reduces the learning rate by a certain amount when the learning stops. It is thought that sometimes our model will be better off if it slows down the rate at which it learns when it is stuck in a plateau.

Method4: LearningRateScheduler



```
Test loss: 34.77
Test accuracy: 84.39
Train loss: 32.51
Train accuracy: 85.22999999999999
```

LearningRateScheduler: Techniques that gradually reduce the learning rate during training are the most straightforward and, in many cases, the most often used modification of learning rate during training. In addition to making large changes at the beginning of the training procedure when larger learning rate values are used, this method has the advantage of decreasing the learning rate so that a smaller rate and, consequently, smaller training updates to weights are made later in the training procedure when smaller learning rate values are used.

Part III: Building a CNN

1. **Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? How many entries and variables does the dataset comprise? Provide the main statistics about the entries of the dataset**

   Fashion-MNIST is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples.

   Each example is a 28x28 grayscale image, associated with a label from 10 classes.

   Each training and test example is assigned to one of the following labels:

   0 T-shirt/top

   1 Trouser

   2 Pullover

   3 Dress

   4 Coat

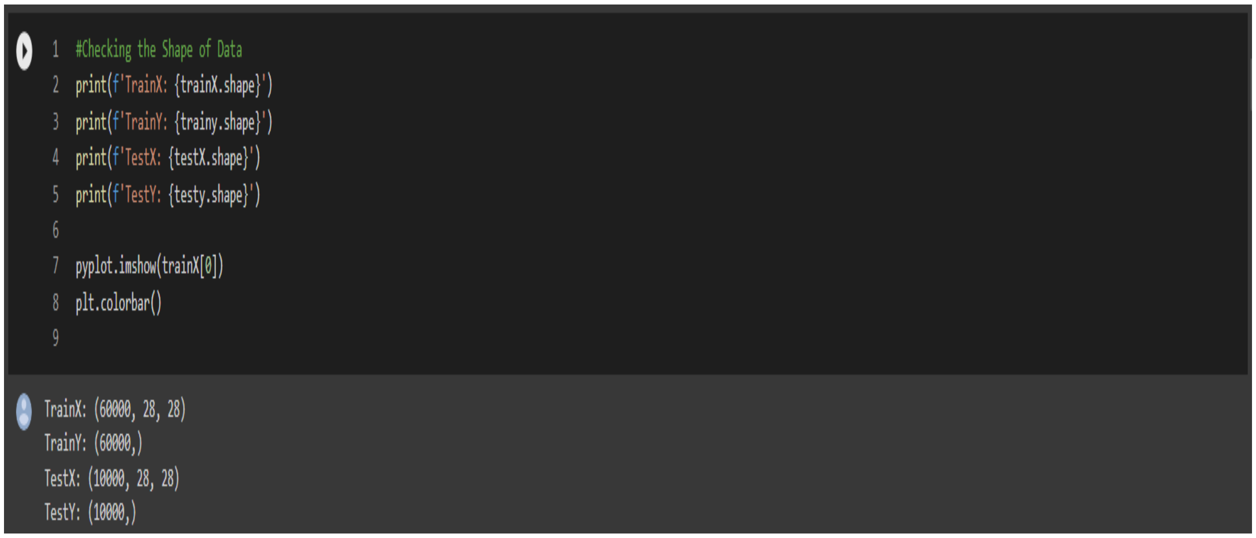
   5 Sandal

   6 Shirt

   7 Sneaker

   8 Bag

   9 Ankle boot

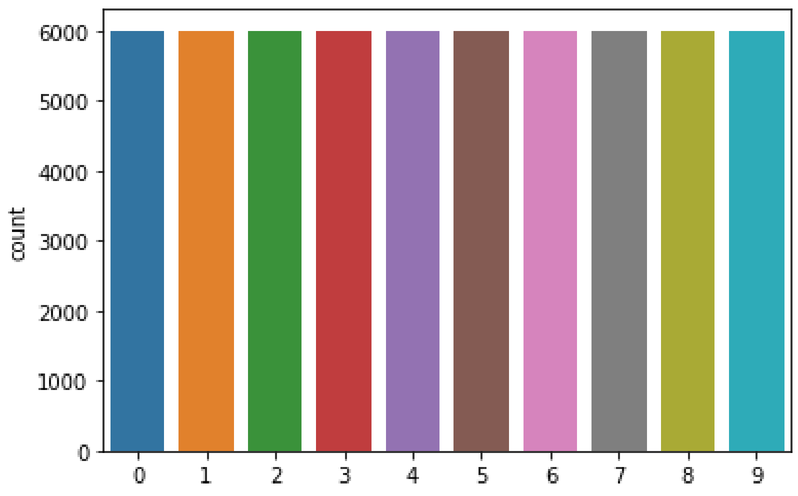


```
#Checking the Shape of Data
print(f'TrainX: {trainX.shape}')
print(f'TrainY: {trainy.shape}')
print(f'TestX: {testX.shape}')
print(f'TestY: {testy.shape}')

pyplot.imshow(trainX[0])
plt.colorbar()

```

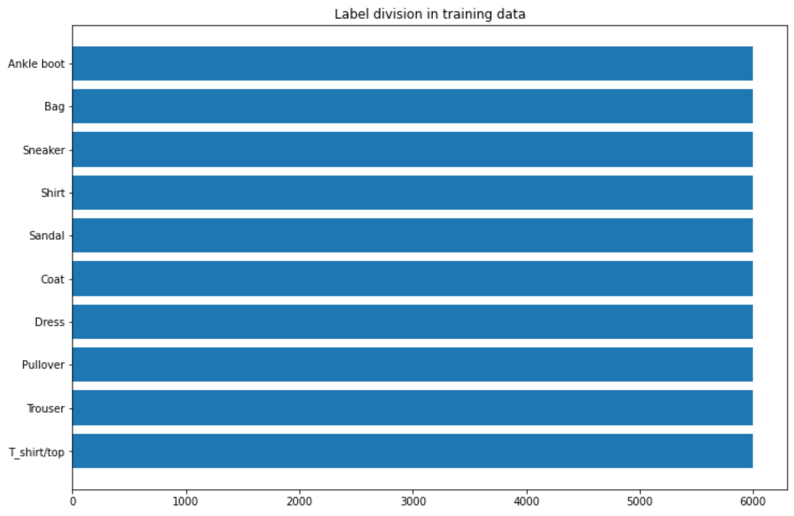
```
TrainX: (60000, 28, 28)
TrainY: (60000,)
TestX: (10000, 28, 28)
TestY: (10000,)
```

2. Provide at least 3 visualization graphs with short description for each graph.

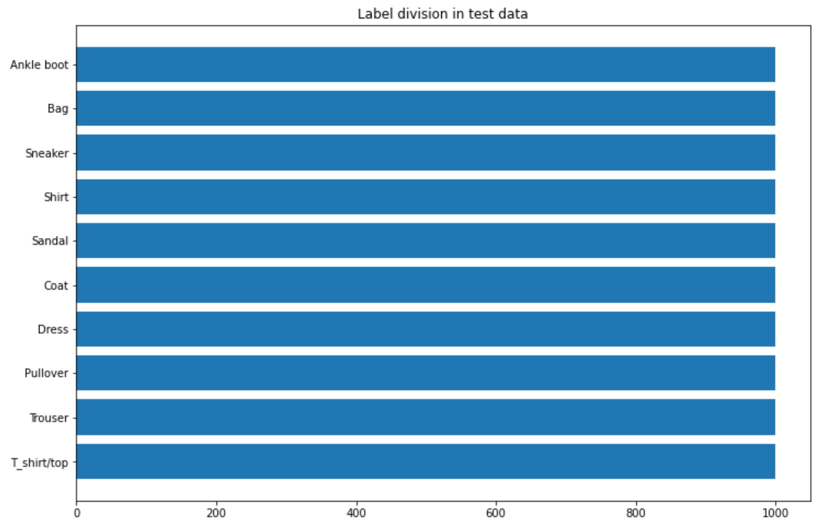
Graph 1 : Plot of Count of Train Data



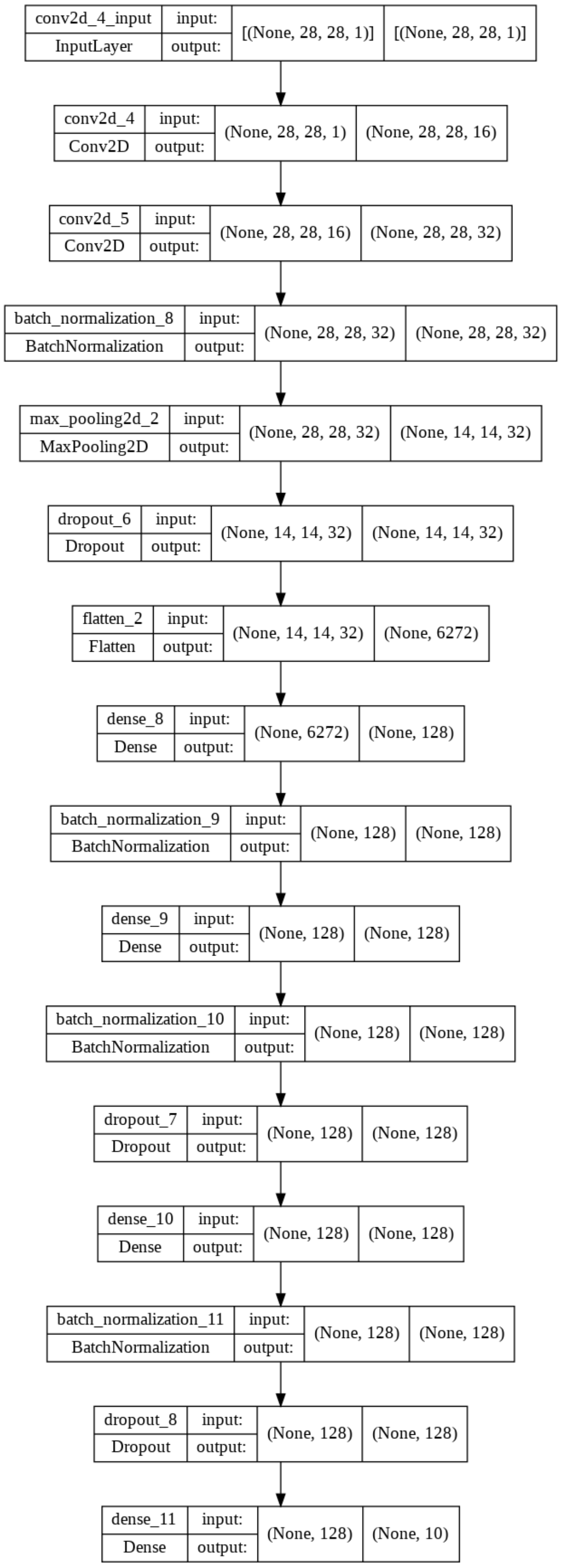
Graph 2 : Label division in training data



Graph 3 : Label division in test data



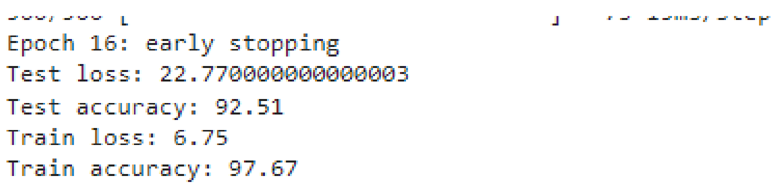
3. Provide the architecture structure of your CNN.


conv2d_4_input | InputLayer | input: [(None, 28, 28, 1)] | output: [(None, 28, 28, 1)]
conv2d_4 | Conv2D | input: (None, 28, 28, 1) | output: (None, 28, 28, 16)
conv2d_5 | Conv2D | input: (None, 28, 28, 16) | output: (None, 28, 28, 32)
batch_normalization_8 | BatchNormalization | input: (None, 28, 28, 32) | output: (None, 28, 28, 32)
max_pooling2d_2 | MaxPooling2D | input: (None, 28, 28, 32) | output: (None, 14, 14, 32)
dropout_6 | Dropout | input: (None, 14, 14, 32) | output: (None, 14, 14, 32)
flatten_2 | Flatten | input: (None, 14, 14, 32) | output: (None, 6272)
dense_8 | Dense | input: (None, 6272) | output: (None, 128)
batch_normalization_9 | BatchNormalization | input: (None, 128) | output: (None, 128)
dense_9 | Dense | input: (None, 128) | output: (None, 128)
batch_normalization_10 | BatchNormalization | input: (None, 128) | output: (None, 128)
dropout_7 | Dropout | input: (None, 128) | output: (None, 128)
dense_10 | Dense | input: (None, 128) | output: (None, 128)
batch_normalization_11 | BatchNormalization | input: (None, 128) | output: (None, 128)
dropout_8 | Dropout | input: (None, 128) | output: (None, 128)
dense_11 | Dense | input: (None, 128) | output: (None, 10)

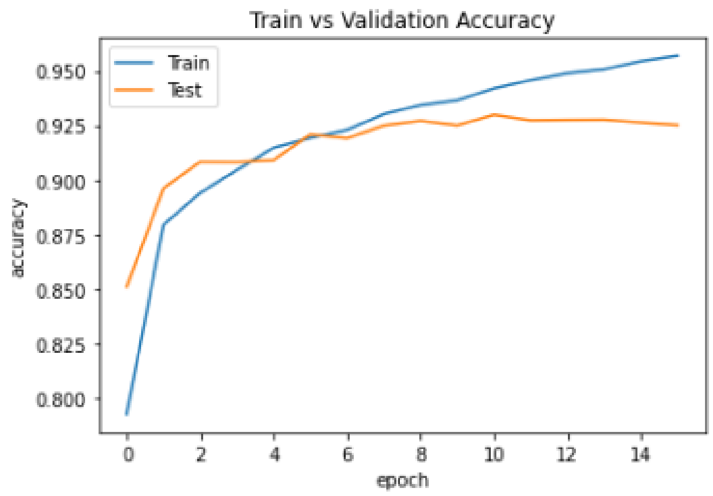
4. Discuss how the improvement tools works on CNN architectures.

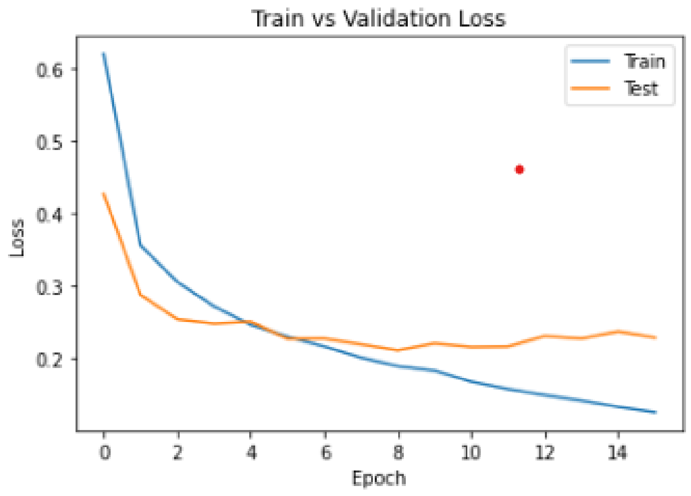
Improvements tried for CNN architecture :

1. Early folding :



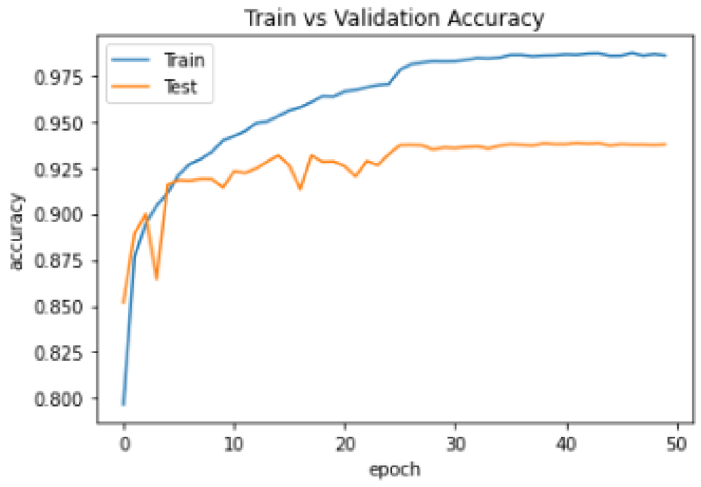
```
Epoch 16: early stopping
Test loss: 22.770000000000003
Test accuracy: 92.51
Train loss: 6.75
Train accuracy: 97.67
```

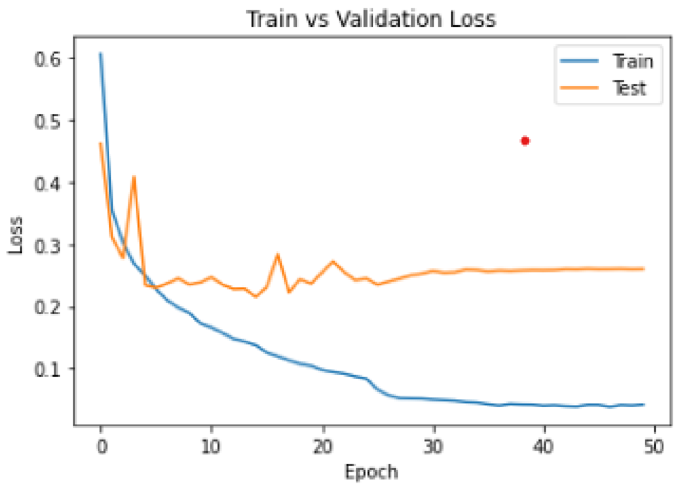




2. ReduceLROnPlateau :

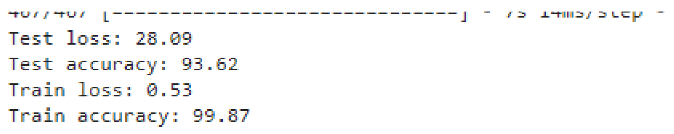
```
Test loss: 26.009999999999998
Test accuracy: 93.76
Train loss: 0.63
Train accuracy: 99.85000000000001
```



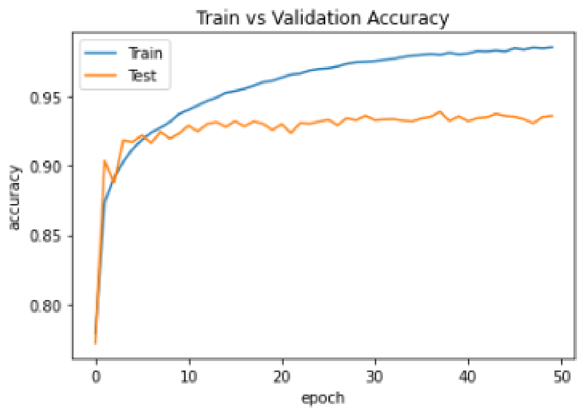


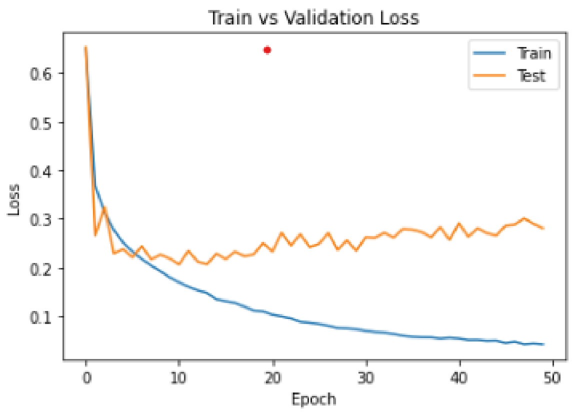
3. K-Fold : Number of fold – 5

Fold 1:



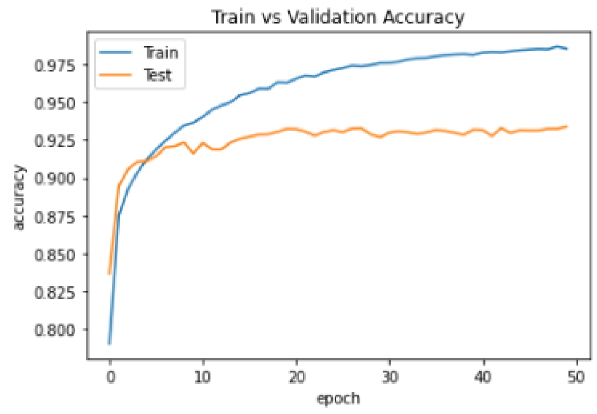
```
407/407 [==============================] - 7s 14ms/step -
Test loss: 28.09
Test accuracy: 93.62
Train loss: 0.53
Train accuracy: 99.87
```

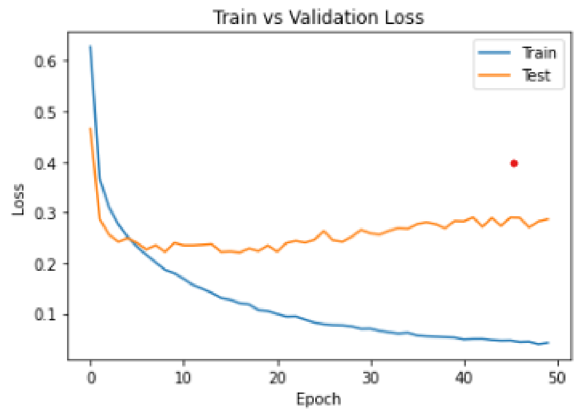




Fold 2:

```
Test loss: 28.689999999999998
Test accuracy: 93.39
Train loss: 0.38999999999999996
Train accuracy: 99.92999999999999
```

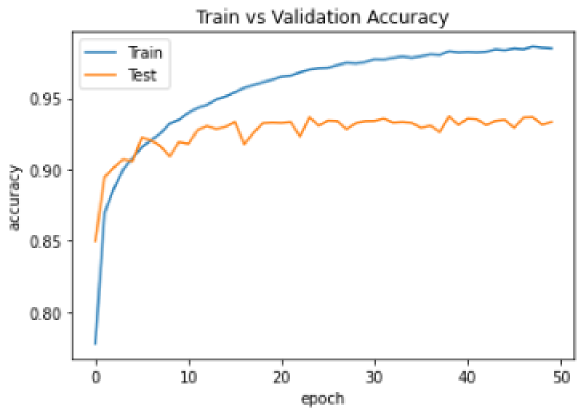


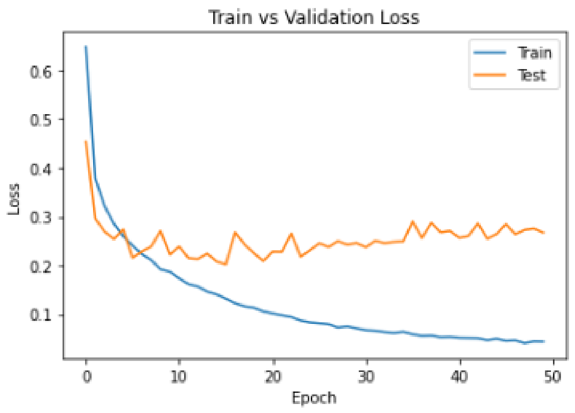


```
----------------------------------------------------------
```

Fold 3:

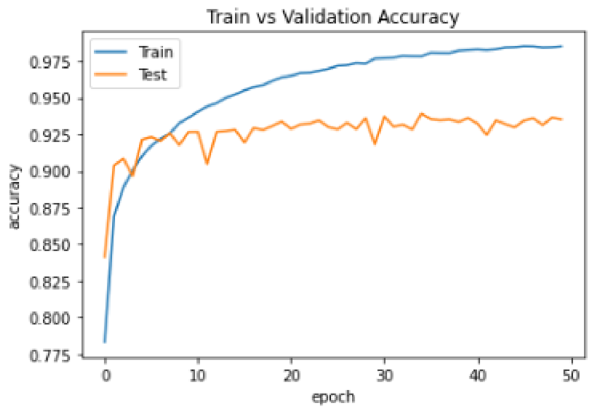
```
Test loss: 26.740000000000002
Test accuracy: 93.33
Train loss: 0.5
Train accuracy: 99.89
```

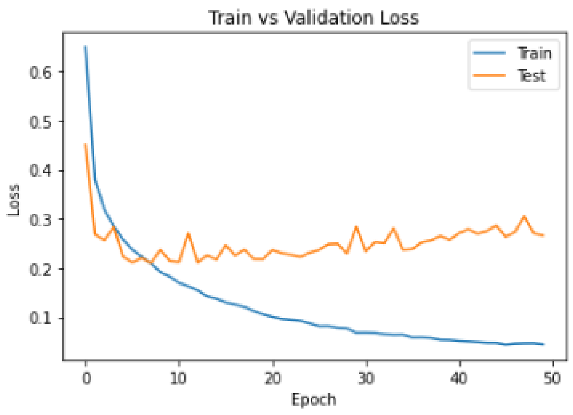




Fold 4:

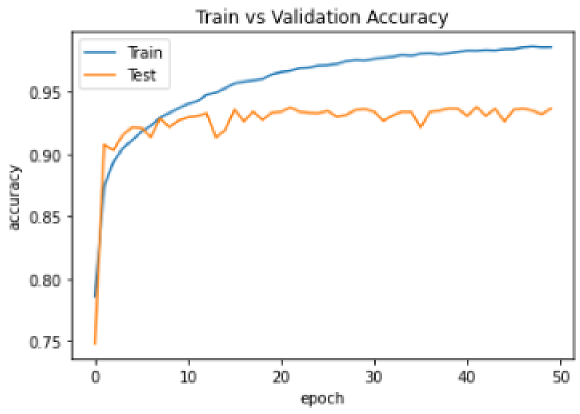
```
407/407 [==============================] - 7s 15ms/step
Test loss: 26.619999999999997
Test accuracy: 93.52000000000001
Train loss: 0.5499999999999999
Train accuracy: 99.86
```

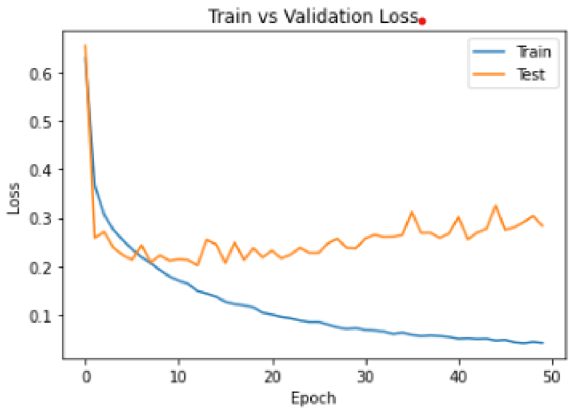




Fold 5:

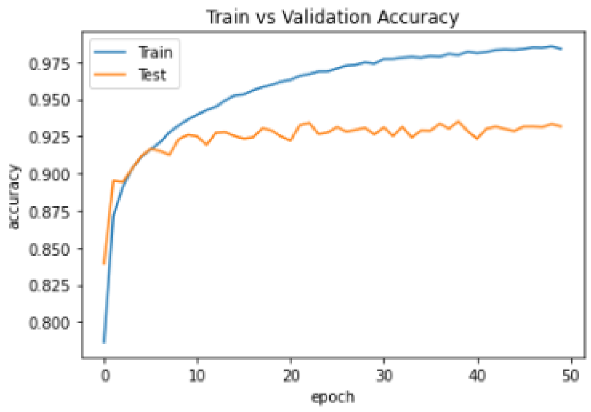
```
Test loss: 28.38
Test accuracy: 93.64
Train loss: 0.47000000000000003
Train accuracy: 99.86
```

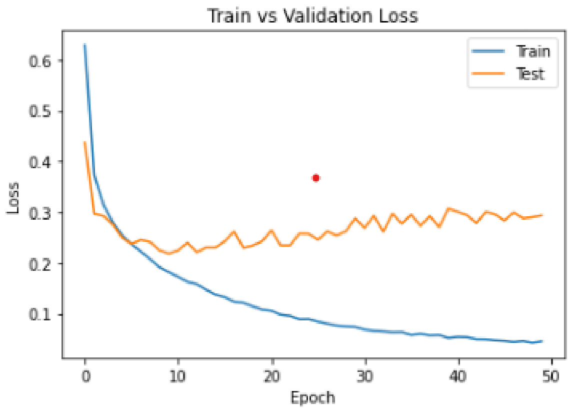




4. LearningRateScheduler :

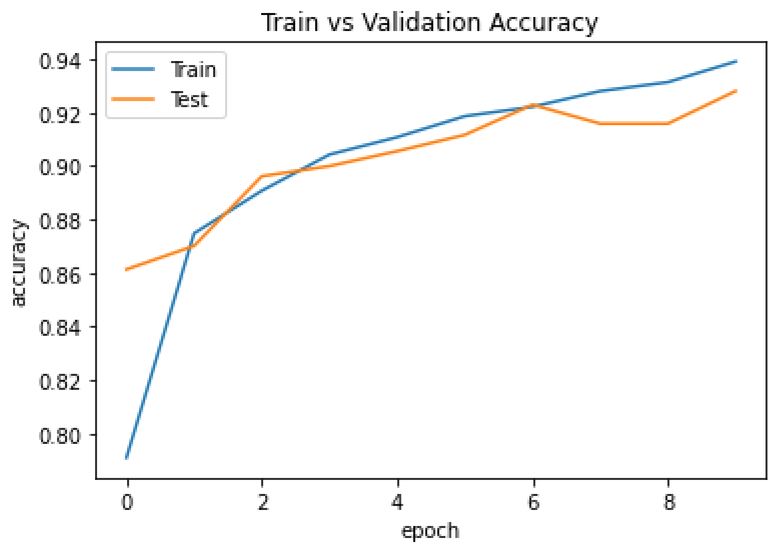
```
Test loss: 29.360000000000003
Test accuracy: 93.17
Train loss: 0.62
Train accuracy: 99.8
```



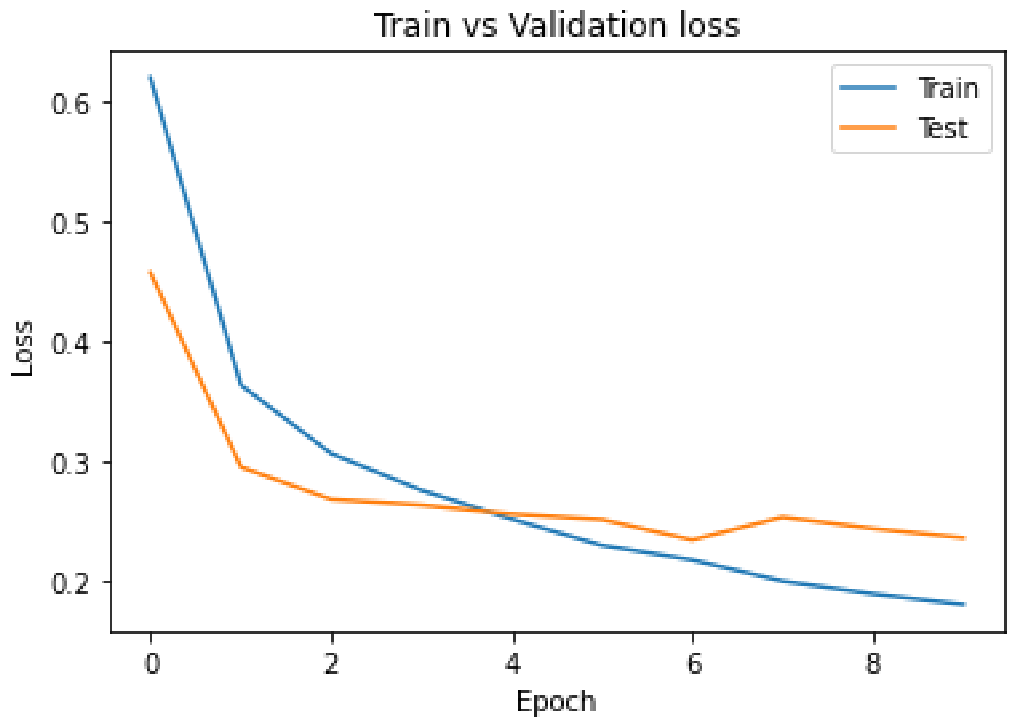


.

5. **Provide graphs that compares test and training accuracy on the same plot, test and training loss on the same plot. Thus, in total two graph with a clear labeling.**



#Loss function graph :



Part IV: Optimizing CNN + Data Argumentation

Q1. Include all 3 tables with different CNN setups.

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.1	98.58%	0.3	97.15%	0.7	93.73%
Optimizer	adam		adam		adam	
Activation Function	relu		relu		relu	
Initializer						
Kernel	(3,3)		(3,3)		(3,3)	

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.1		0.1		0.1	
Optimizer	RMSprop	97.65%	Adadelta	67.02%	Adagrad	87.97%
Activation Function	relu		relu		relu	
Initializer						
Kernel	(3,3)		(3,3)		(3,3)	

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.1		0.1		0.1	
Optimizer	RMSprop		RMSprop		RMSprop	
Activation Function	Softplus	97.65%	Softsign	67.02%	tanh	87.97%
Initializer						
Kernel	(3,3)		(3,3)		(3,3)	

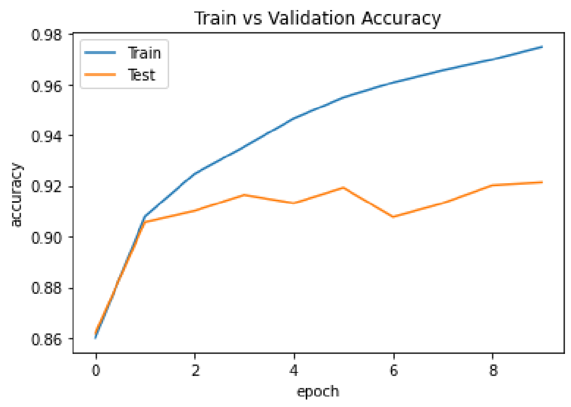
	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.1		0.1		0.1	
Optimizer	RMSprop		RMSprop		RMSprop	
Activation Function	relu		relu		relu	
Initializer	glorot_uniform	97.09%	he_normal	97.09%	he_uniform	97.399%
Kernel	(3,3)		(3,3)		(3,3)	

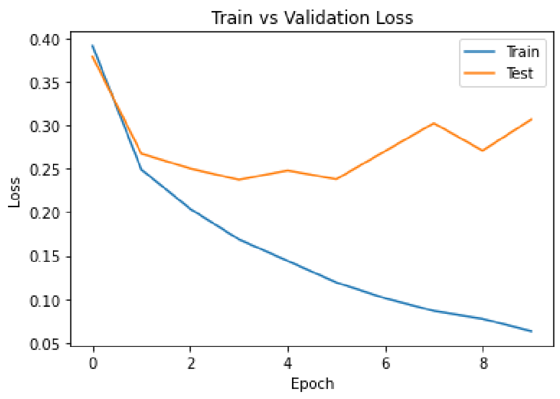
	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.1		0.1		0.1	
Optimizer	RMSprop		RMSprop		RMSprop	
Activation Function	relu		relu		relu	
Initializer						
Kernel	(5,5)	95.309%	(7,7)	96.0%	(9,9)	95.28%

Q2. Provide graphs that compares test and training accuracy on the same plot for all your setups and add a short description for each graph.

Dropout = 0.1

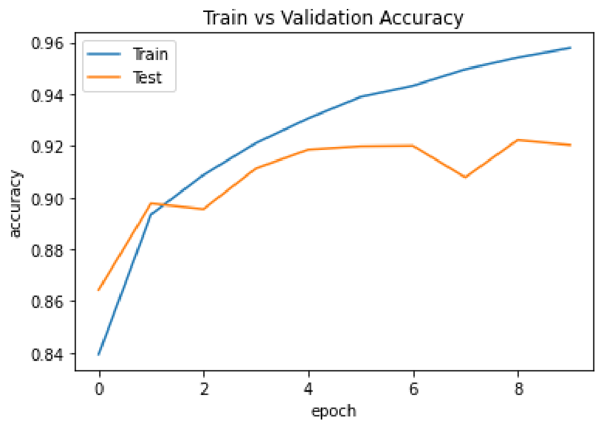
Training / Testing Accuracy Training / Testing Loss

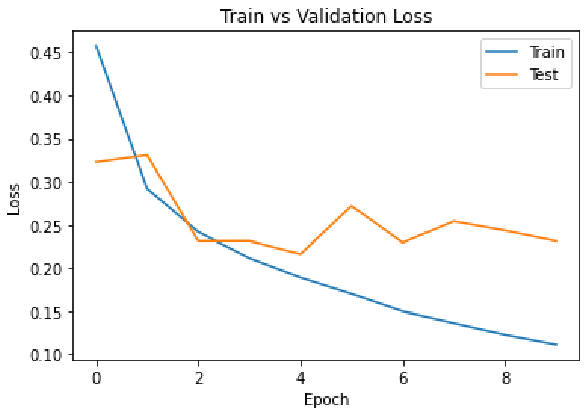




Dropout = 0.2

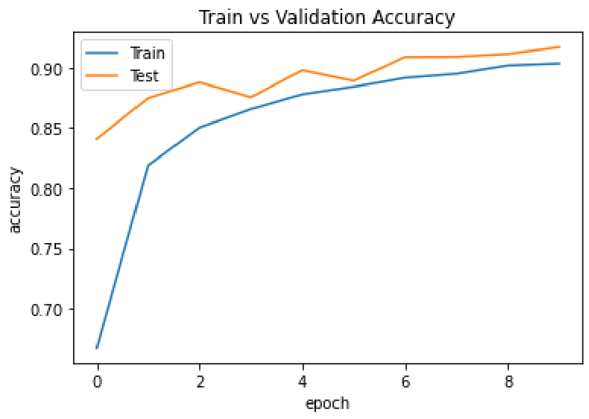
Training / Testing Accuracy Training / Testing Loss

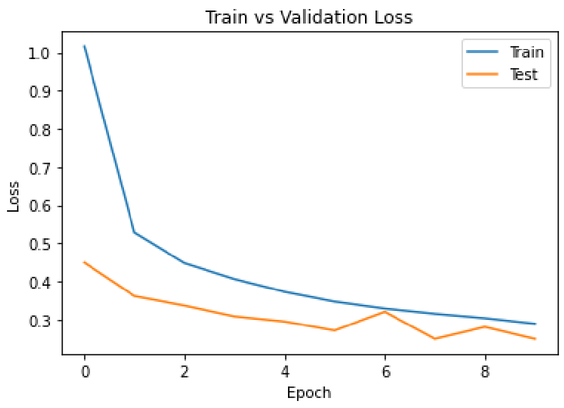




Dropout = 0.3

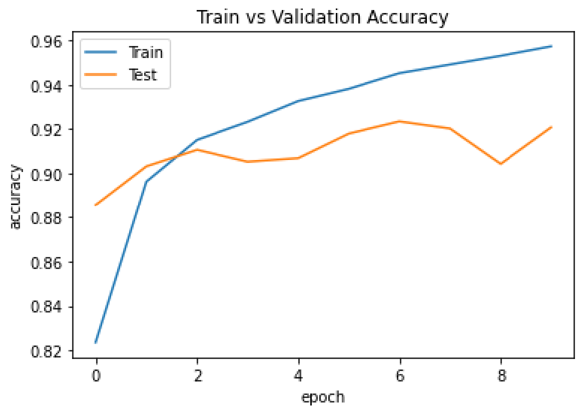
Training / Testing Accuracy Training / Testing Loss

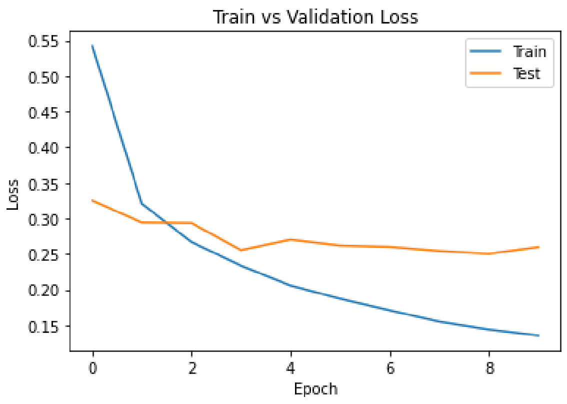




Optimizer: RMSprop

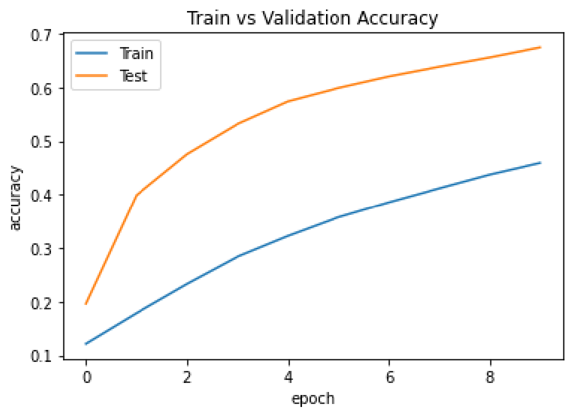
Training / Testing Accuracy Training / Testing Loss

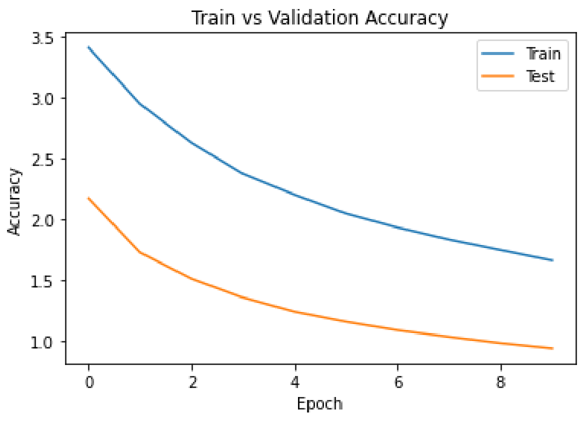




Optimizer: Adadelta

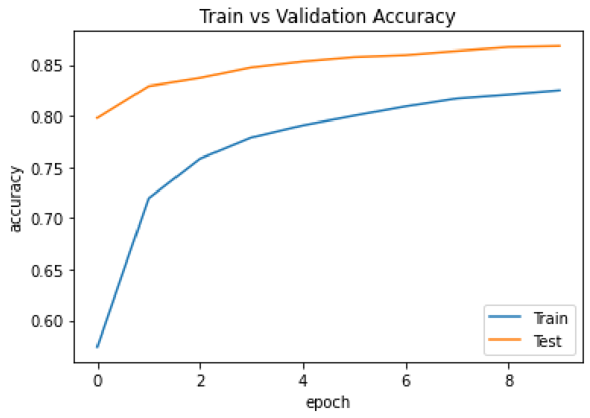
Training / Testing Accuracy　　　　　　Training / Testing Loss

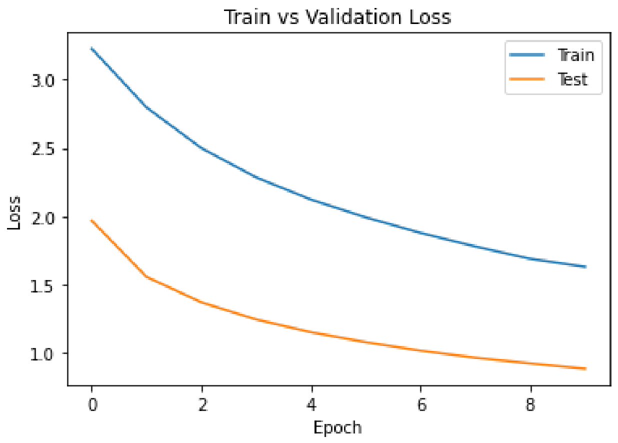




Optimizer: Adagrad

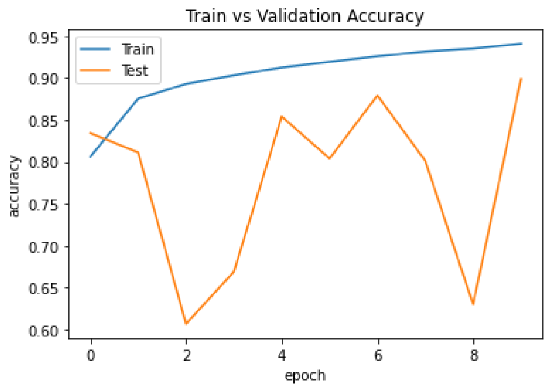
Training / Testing Accuracy　　　　　　Training / Testing Loss

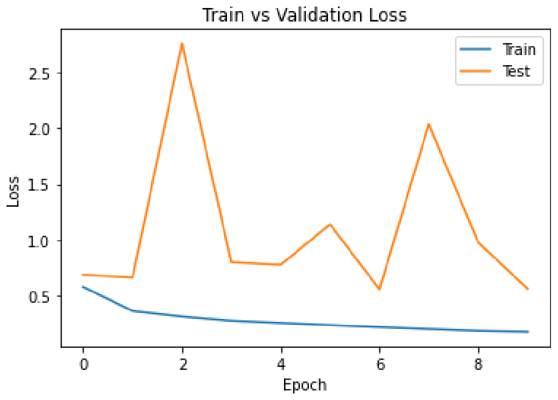




Activation Function: Softplus

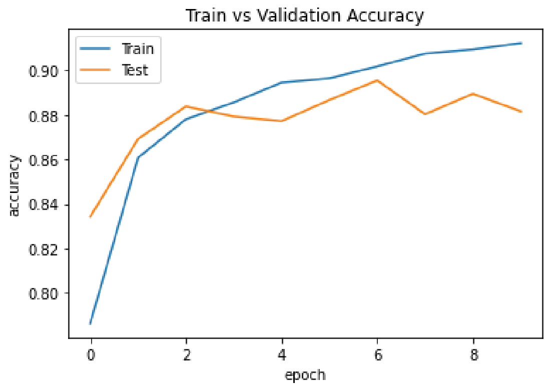
Training / Testing Accuracy Training / Testing Loss

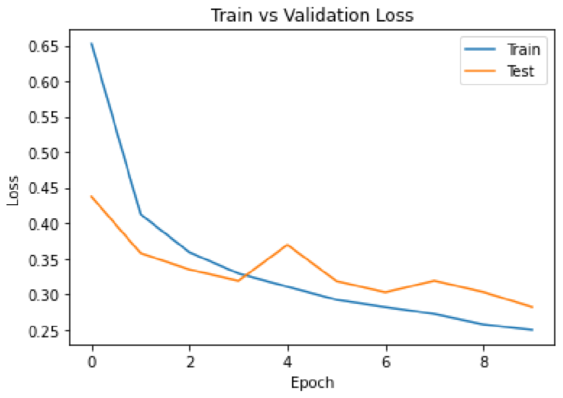




Activation Function: Softsign

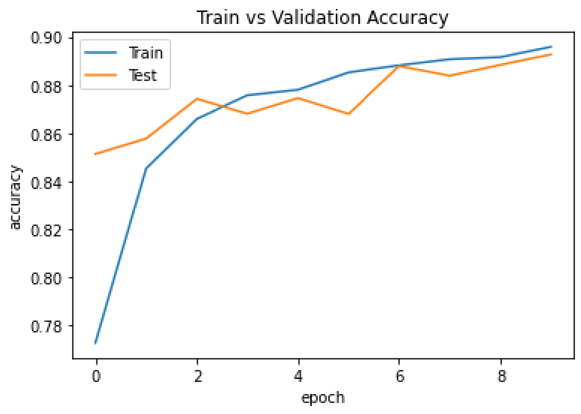
Training / Testing Accuracy Training / Testing Loss

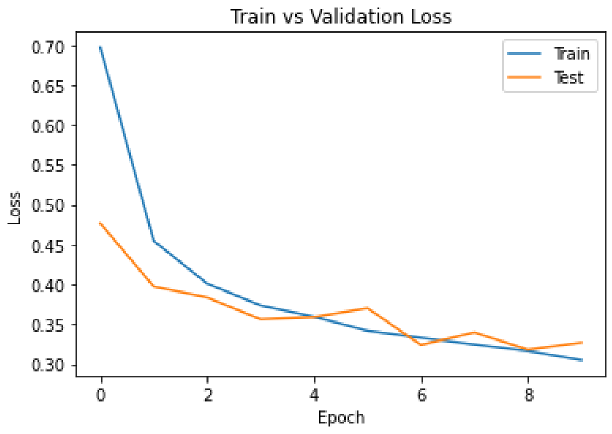




Activation Function: Tanh

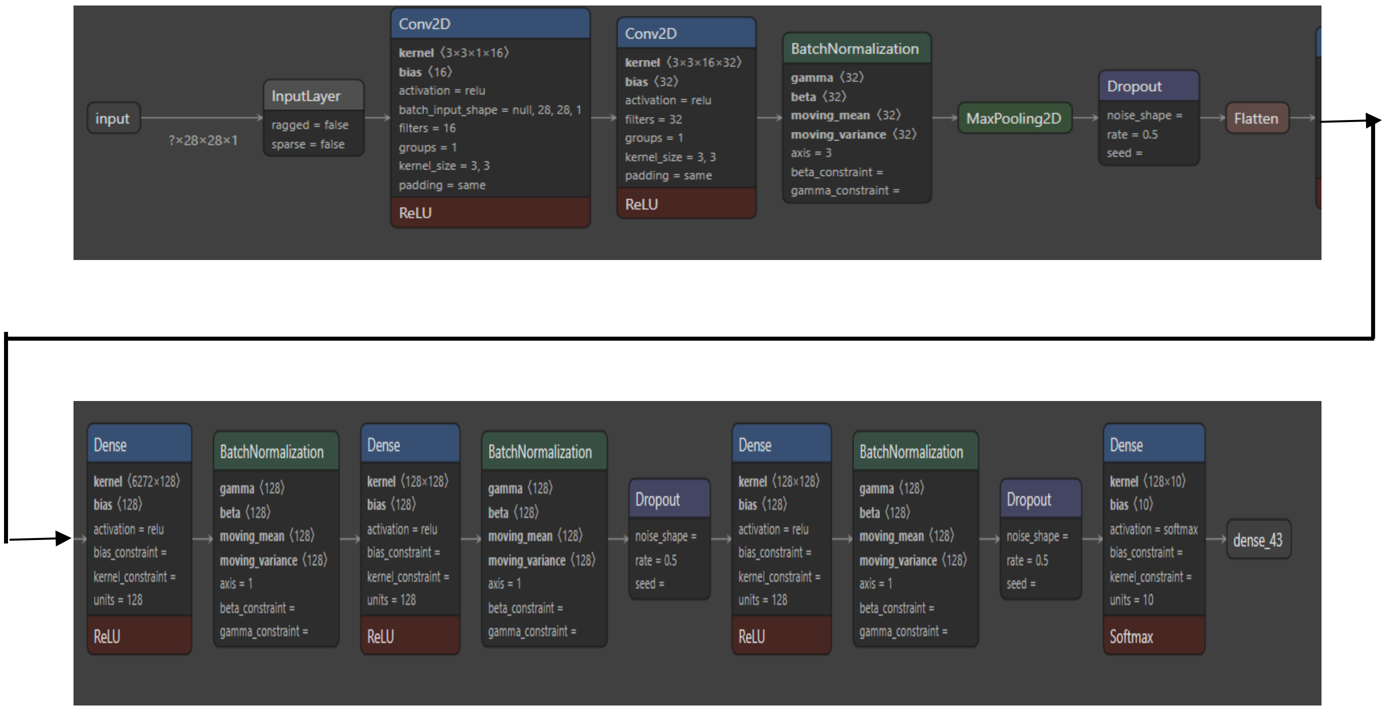
Training / Testing Accuracy Training / Testing Loss





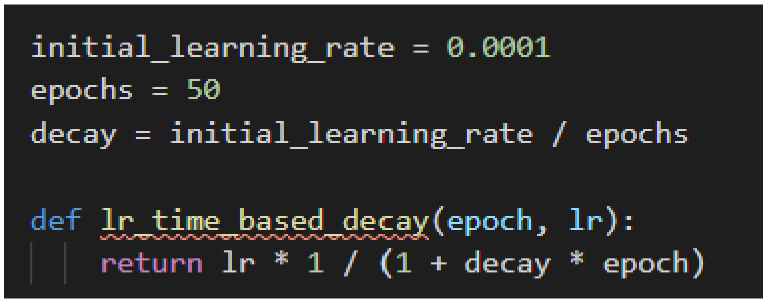
Q3. Provide a detailed analysis and reasoning about the CNN setups that you tried

We used Netron to generate the CNN model which is shown below.



For optimizing our model we tried Learning Rate Scheduler using Time – Base Decay.

Where time based decay is given as :



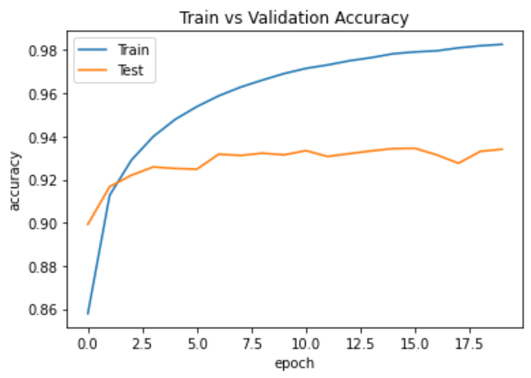
```
initial_learning_rate = 0.0001
epochs = 50
decay = initial_learning_rate / epochs

def lr_time_based_decay(epoch, lr):
    return lr * 1 / (1 + decay * epoch)
```

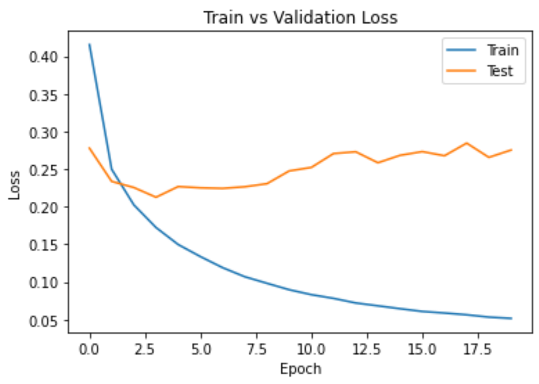
where lr is the previous learning rate, decay represents a hyperparameter, and epoch represents the number of iterations. When the decay is zero, there is no effect on the learning rate. When the decay parameter is set, the learning rate from the previous epoch will be decreased by the stated fixed amount from that point on.

Q4. Discuss briefly methos that you used for data augmentation and reason how it helped to increase the accuracy of the model.

We used Image generator for augmenting the data. So, we applied horizontal flip to our dataset and increased the dataset by x4. Augmenting the dataset gave the better accuracy while applied on learning rate scheduler model.

```
Test loss: 27.51

Test accuracy: 93.41000000000001
Train loss: 0.72
Train accuracy: 99.82
```





References:

1. https://www.tensorflow.org/tutorials/images/data_augmentation#data_augmentation_2
2. https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/LearningRateScheduler
3. https://www.youtube.com/watch?v=2C5uGteyhS4&ab_channel=KGPTalkie
4. https://towardsdatascience.com/building-your-own-artificial-neural-network-from-scratch-on-churn-modeling-dataset-using-keras-in-690782f7d051

Name	Part Number	Contribution
Chaitanya Desai	Part 2, Part 3, Project Report	100%, 100%, 50%
Sharukh Khan	Part 1, Part 4, Project Report	100%, 100%, 50%