CSE 574 Introduction to Machine Learning

Assignment 3
Building Neural Networks and CNN

Team Members:
Naman Tejaswi - namantej@buffalo.edu (50435697)
Sachin Sarsambi - ssarsamb@buffalo.edu (50419943)

Academic Integrity Statement

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.

Part I – Building a Basic NN

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.
    - The dataset being analyzed contains Income details of people across different countries and working class. The following are the columns and their corresponding datatypes:

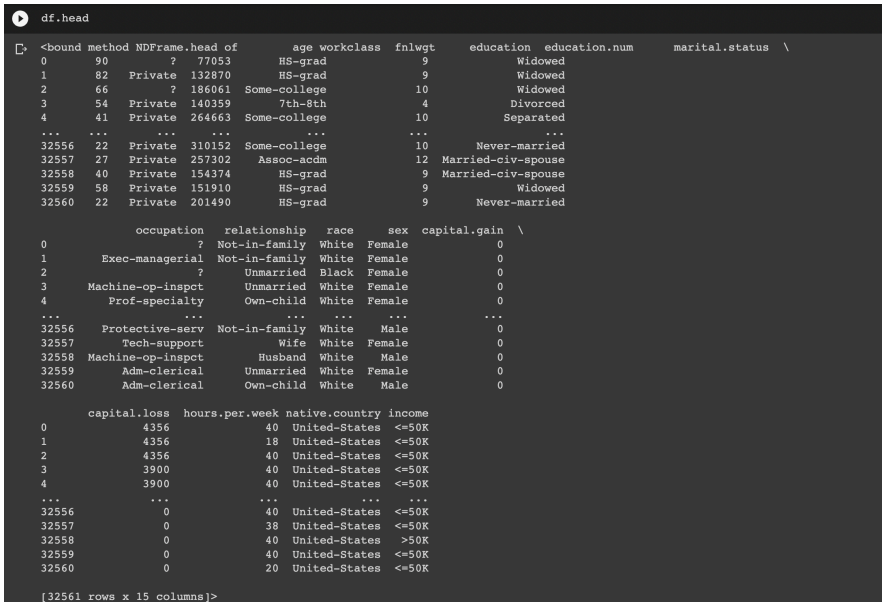
Column Names	Data Type
age	int64
workclass	object
fnlwgt	int64
education	object
education.num	int64
marital.status	object
occupation	object
relationship	object
race	object
sex	object
capital.gain	int64
capital.loss	int64
hours.per.week	int64
native.country	object
income	object

    - Here, there are 15 columns and 32561 data samples. The columns *age, fnlwgt, education.num, capital.gain and capital.loss* are of the type int whereas the columns of *workclass, education, marital.status, occupation, relationship, race, sex, native.country* and *income* are of the type *object*. The columns of the dtype *object* are later converted to *category* type. Using the correlation matrix it can be observed that there is a direct relationship between *income* and *hours per week*.

- The salient features of the dataset:

index	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country	income
count	32561.0	32561	32561.0	32561	32561.0	32561	32561	32561	32561	32561	32561.0	32561.0	32561.0	32561	32561
unique	NaN	9	NaN	16	NaN	7	15	6	5	2	NaN	NaN	NaN	42	2
top	NaN	Private	NaN	HS-grad	NaN	Married-civ-spouse	Prof-specialty	Husband	White	Male	NaN	NaN	NaN	United-States	<=50K
freq	NaN	22696	NaN	10501	NaN	14976	4140	13193	27816	21790	NaN	NaN	NaN	29170	24720
mean	38.58164675532080	NaN	189778.36651208500	NaN	10.0806793403151	NaN	NaN	NaN	NaN	NaN	1077.6488437087300	87.303829734959	40.437455852093000	NaN	NaN
std	13.640432553581100	NaN	105549.97769702200	NaN	2.5727203320673400	NaN	NaN	NaN	NaN	NaN	7385.292084836110	402.9602186495170	12.347428681730800	NaN	NaN
min	17.0	NaN	12285.0	NaN	1.0	NaN	NaN	NaN	NaN	NaN	0.0	0.0	1.0	NaN	NaN
25%	28.0	NaN	117827.0	NaN	9.0	NaN	NaN	NaN	NaN	NaN	0.0	0.0	40.0	NaN	NaN
50%	37.0	NaN	178356.0	NaN	10.0	NaN	NaN	NaN	NaN	NaN	0.0	0.0	40.0	NaN	NaN
75%	48.0	NaN	237051.0	NaN	12.0	NaN	NaN	NaN	NaN	NaN	0.0	0.0	45.0	NaN	NaN
max	90.0	NaN	1484705.0	NaN	16.0	NaN	NaN	NaN	NaN	NaN	99999.0	4356.0	99.0	NaN	NaN

- The head data of the income dataset:



```
df.head

<bound method NDFrame.head of        age workclass  fnlwgt     education  education.num      marital.status  \
0       90         ?   77053       HS-grad              9            Widowed
1       82   Private  132870       HS-grad              9            Widowed
2       66         ?  186061  Some-college             10            Widowed
3       54   Private  140359       7th-8th              4           Divorced
4       41   Private  264663  Some-college             10          Separated
...    ...       ...     ...           ...            ...                ...
32556   22   Private  310152  Some-college             10      Never-married
32557   27   Private  257302    Assoc-acdm             12  Married-civ-spouse
32558   40   Private  154374       HS-grad              9  Married-civ-spouse
32559   58   Private  151910       HS-grad              9            Widowed
32560   22   Private  201490       HS-grad              9      Never-married

              occupation   relationship   race     sex  capital.gain  \
0                      ?  Not-in-family  White  Female             0
1        Exec-managerial  Not-in-family  White  Female             0
2                      ?      Unmarried  Black  Female             0
3      Machine-op-inspct      Unmarried  White  Female             0
4         Prof-specialty      Own-child  White  Female             0
...                  ...            ...    ...     ...           ...
32556     Protective-serv  Not-in-family  White    Male             0
32557        Tech-support           Wife  White  Female             0
32558  Machine-op-inspct        Husband  White    Male             0
32559        Adm-clerical      Unmarried  White  Female             0
32560        Adm-clerical      Own-child  White    Male             0

       capital.loss  hours.per.week native.country income
0              4356              40  United-States  <=50K
1              4356              18  United-States  <=50K
2              4356              40  United-States  <=50K
3              3900              40  United-States  <=50K
4              3900              40  United-States  <=50K
...             ...             ...            ...    ...
32556             0              40  United-States  <=50K
32557             0              38  United-States  <=50K
32558             0              40  United-States   >50K
32559             0              40  United-States  <=50K
32560             0              20  United-States  <=50K

[32561 rows x 15 columns]>
```

- The column names and their corresponding information:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64
 11  capital.loss    32561 non-null  int64
 12  hours.per.week  32561 non-null  int64
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

- The tail records of the income dataset:

```
df.tail
```

```
<bound method NDFrame.tail of        age workclass  fnlwgt     education  education.num      marital.status  \
0       90         ?   77053       HS-grad              9             Widowed
1       82   Private  132870       HS-grad              9             Widowed
2       66         ?  186061  Some-college             10             Widowed
3       54   Private  140359       7th-8th              4            Divorced
4       41   Private  264663  Some-college             10           Separated
...    ...       ...     ...           ...            ...                 ...
32556   22   Private  310152  Some-college             10       Never-married
32557   27   Private  257302    Assoc-acdm             12  Married-civ-spouse
32558   40   Private  154374       HS-grad              9  Married-civ-spouse
32559   58   Private  151910       HS-grad              9             Widowed
32560   22   Private  201490       HS-grad              9       Never-married

              occupation   relationship   race     sex  capital.gain  \
0                      ?  Not-in-family  White  Female             0
1        Exec-managerial  Not-in-family  White  Female             0
2                      ?      Unmarried  Black  Female             0
3      Machine-op-inspct      Unmarried  White  Female             0
4         Prof-specialty      Own-child  White  Female             0
...                  ...            ...    ...     ...           ...
32556    Protective-serv  Not-in-family  White    Male             0
32557       Tech-support           Wife  White  Female             0
32558  Machine-op-inspct        Husband  White    Male             0
32559       Adm-clerical      Unmarried  White  Female             0
32560       Adm-clerical      Own-child  White    Male             0

       capital.loss  hours.per.week native.country income
0              4356              40  United-States  <=50K
1              4356              18  United-States  <=50K
2              4356              40  United-States  <=50K
3              3900              40  United-States  <=50K
4              3900              40  United-States  <=50K
...             ...             ...            ...    ...
32556             0              40  United-States  <=50K
32557             0              38  United-States  <=50K
32558             0              40  United-States   >50K
32559             0              40  United-States  <=50K
32560             0              20  United-States  <=50K

[32561 rows x 15 columns]>
```

- The shape of the dataset is:

```
df.shape
```

```
(32561, 15)
```

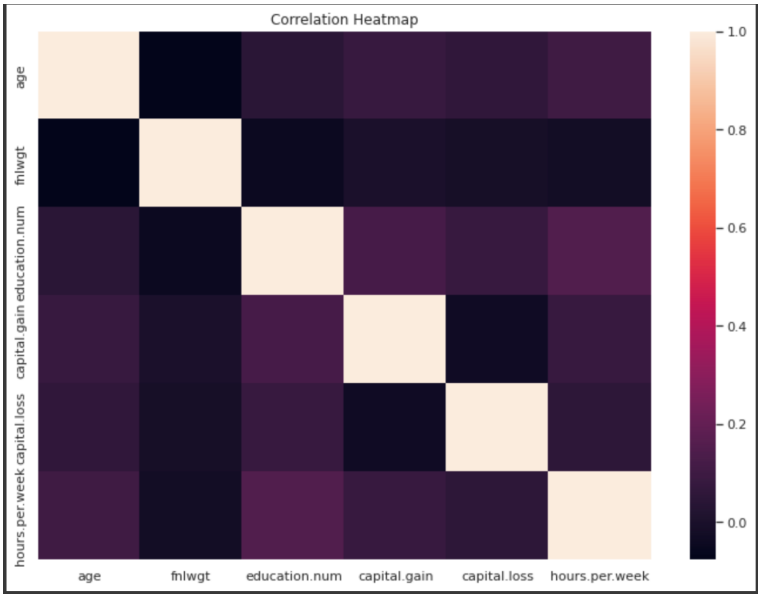
- The count of “?” character across various columns:

```
df[df=='?'].count()
```

```
age                  0
workclass         1836
fnlwgt               0
education            0
education.num        0
marital.status       0
occupation        1843
relationship         0
race                 0
sex                  0
capital.gain         0
capital.loss         0
hours.per.week       0
native.country     583
income               0
dtype: int64
```

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

- *Correlation Matrix*:

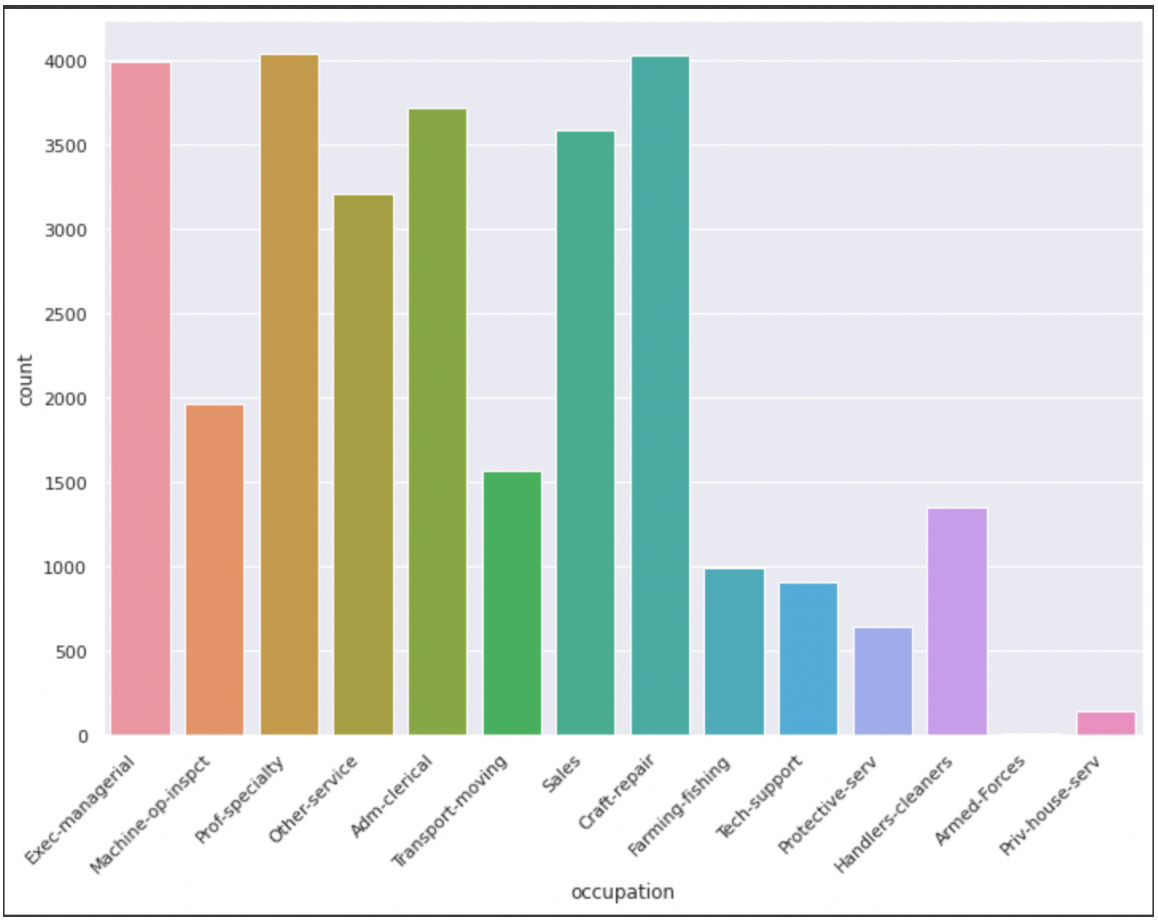



```
df.corr()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
age	1.000000	-0.076511	0.043526	0.080154	0.060165	0.101599
fnlwgt	-0.076511	1.000000	-0.044992	0.000422	-0.009750	-0.022886
education.num	0.043526	-0.044992	1.000000	0.124416	0.079646	0.152522
capital.gain	0.080154	0.000422	0.124416	1.000000	-0.032229	0.080432
capital.loss	0.060165	-0.009750	0.079646	-0.032229	1.000000	0.052417
hours.per.week	0.101599	-0.022886	0.152522	0.080432	0.052417	1.000000

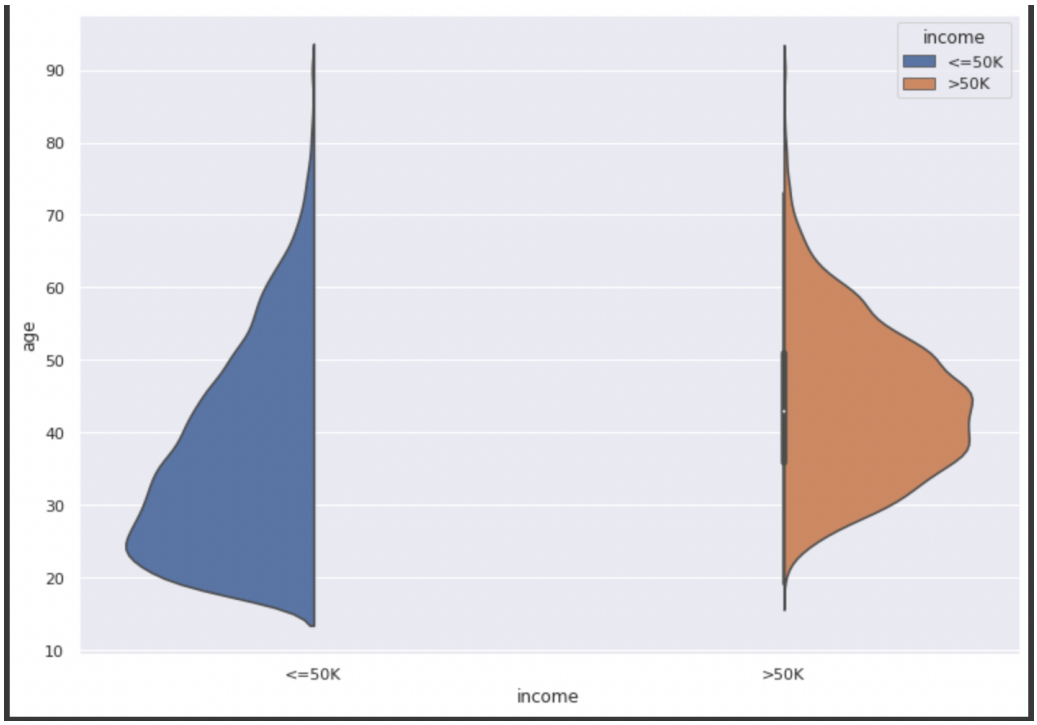
- We are using a heatmap to represent a correlation matrix between features *age, fnlwgt, education.num, capital.gain, capital.loss* and *hours.per.week*. A correlation matrix indicates relationship between various features.
- Here, the closest relation is between *hours per week* and *education column* with a correlation value of 0.15 and then followed by *education* column and *capital gain*.

- *Occupation Distribution*:



- The occupation with least number of employees is *Armed-Forces* (9 Employees) and the occupation with the greatest number of employees is *Prof-Specialty* (4140 Employees).

- *Violin Graph*:



- The above violin graph shows the distribution of income against employee age.
- The density of employee with income <=50K is highest in the age group 20-30 years and the density of employees with income >50K is highest in the age group 35-45 years.

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

- *One-Hot Encoder*: To improve the accuracy, we include as many features as possible. To achieve this, we need to convert the category type data to numerical ones using the one-hot encode process. To apply one-hot encoding on the feature dataset, we use pandas in-built function *get_dummies* api.
- *Min-Max-Normalization*: The *sklearn.preprocessing.MinMaxScaler* performs min-max normalization on the dataset so that the input features have the similar range of magnitude.

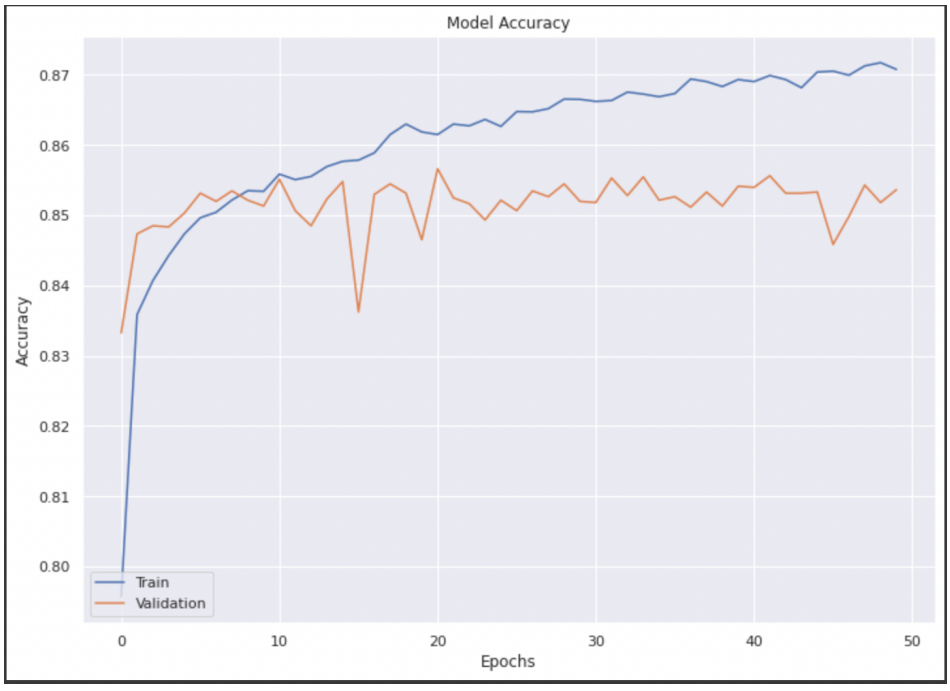
4. Provide the architecture structure of your NN

    - The following is the model summary:

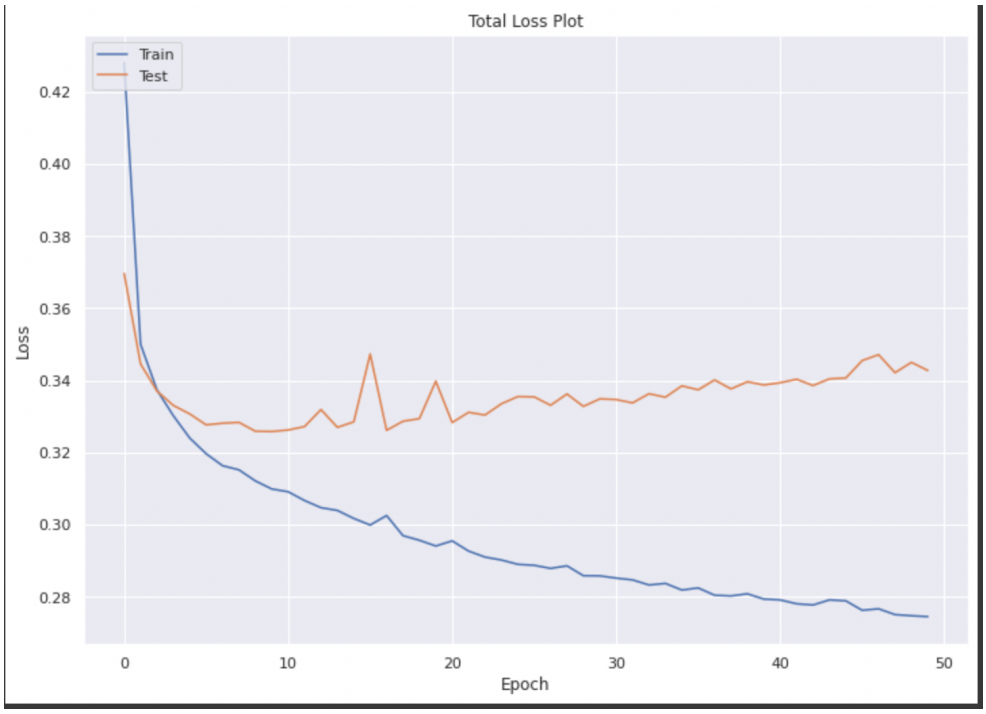
```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_33 (Dense)            (None, 48)                4656

 dense_34 (Dense)            (None, 24)                1176

 dense_35 (Dense)            (None, 2)                 50

=================================================================
Total params: 5,882
Trainable params: 5,882
Non-trainable params: 0
_________________________________________________________________
```

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 graphs with a clear labeling.

    - *Model Accuracy Plot*



- *Total Loss Plot*:



Part II – Optimizing NN

1. Include all 3 tables with different NN setups.

Dropout Tuning-

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.2	84.24%	0.4	82.20%	0.8	75.55%
Optimizer	Adam		Adam		Adam	
Activation Function	relu		relu		relu	
Initializer	-		-		-	

Optimizer Tuning-

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.2		0.2		0.2	
Optimizer	RMSprop	84.47%	SGD	83.31%	Adagrad	82.10%
Activation Function	relu		relu		relu	
Initializer	-		-		-	

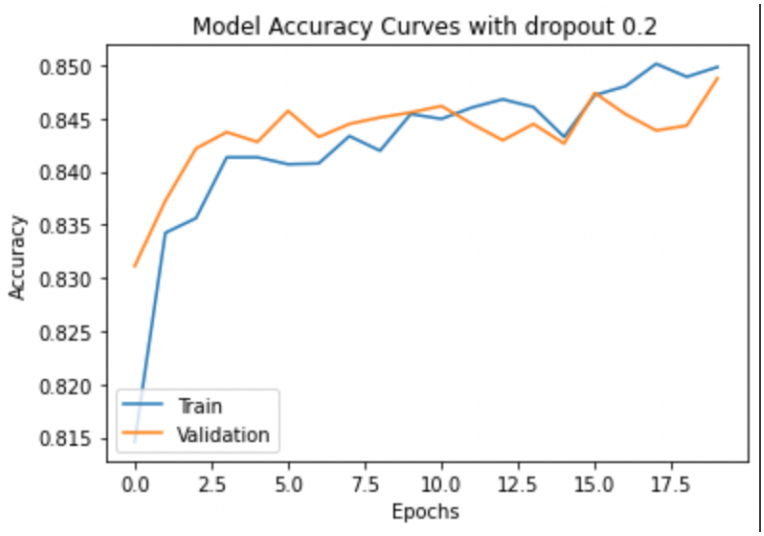
Activation Function Tuning-

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.2		0.2			
Optimizer	SGD		SGD			
Activation Function	Sigmoid	83.01	tanh	83.27	Relu	24.48
Initializer	-		-		-	

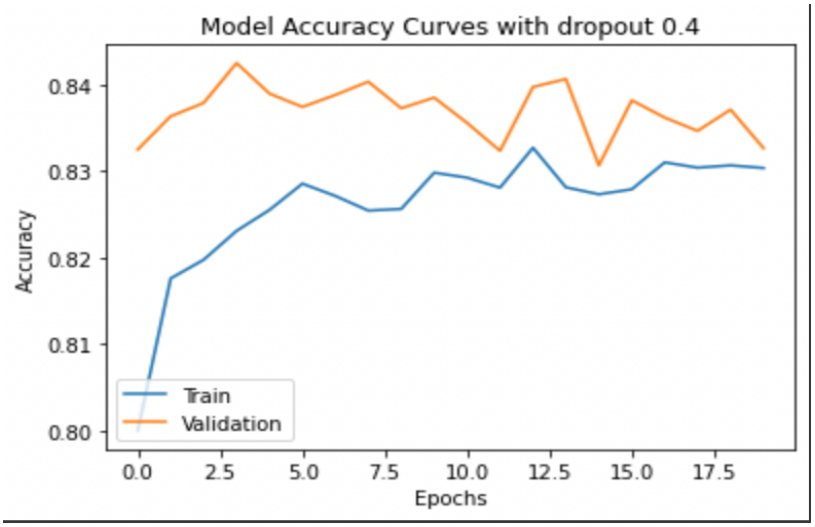
Initializer Tuning-

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.2		0.2		0.2	
Optimizer	SGD		SGD		SGD	
Activation Function	relu		relu		relu	
Initializer	Random normal	75.51%	Zeros weight	75.51%	ones	75.51

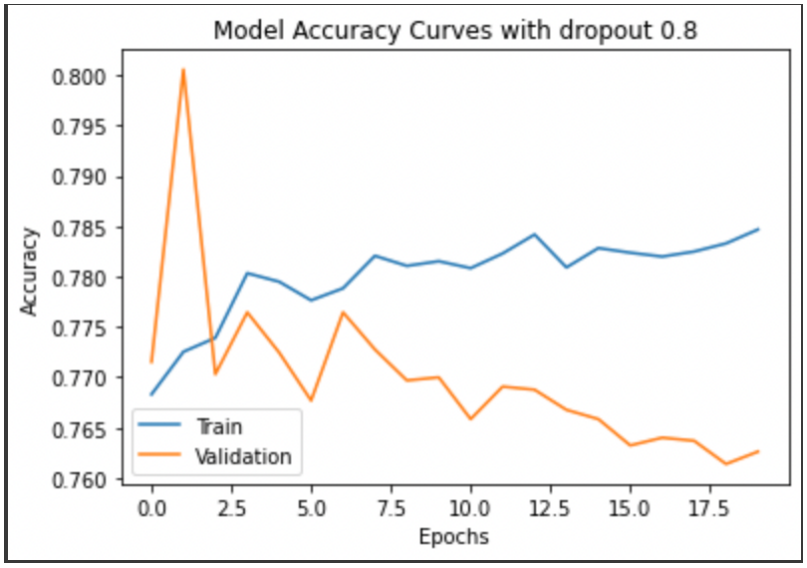
2. Provide graphs that compares test and training accuracy on the same plot for all your setups and add a short description for each graph.
    - Dropout Tuning
        1. Dropout Value=0.2 – Here for a dropout value of 0.2, the accuracy obtained is 84.24%.



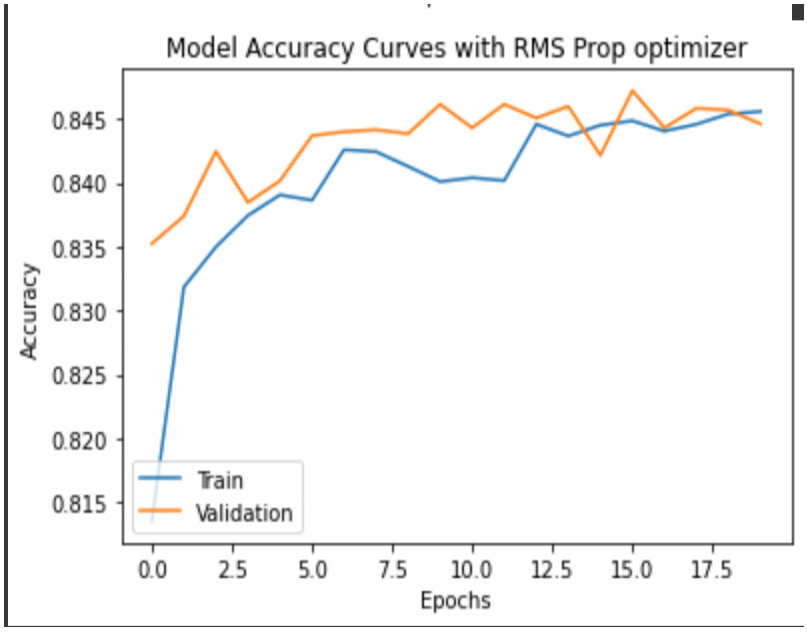
        2. Dropout value=0.4 – Here for a dropout value of 0.4, the accuracy is still significant at 82.20%



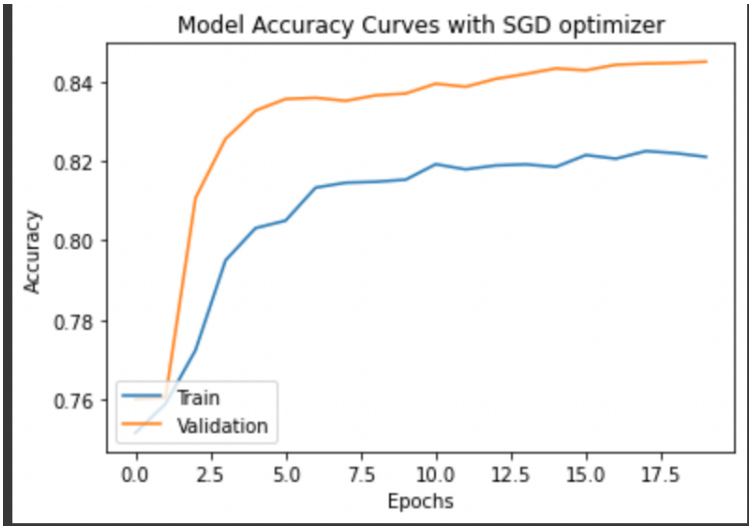
3. Dropout value=0.8 - If we dop 80% of the neurons the complexity of neural network decreases a lot and as a result performance takes a massive hit



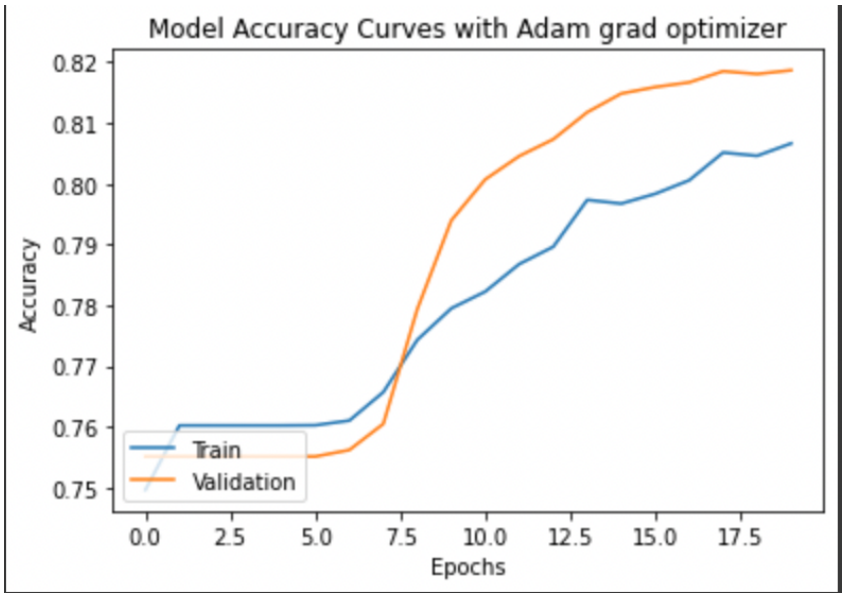
- Optimizer Tuning
    1. RMSprop – With the RMSprop optimizer we achieve a maximum accuracy of 85.62%.



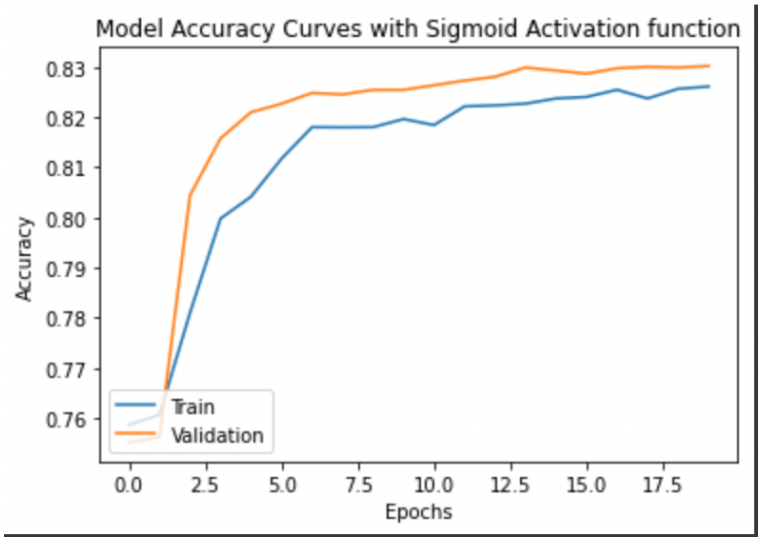
2. SGD - With Stochastic Gradient Descent optimizer the loss and accuracy graph are less jagged and asymptotic in terms of convergence



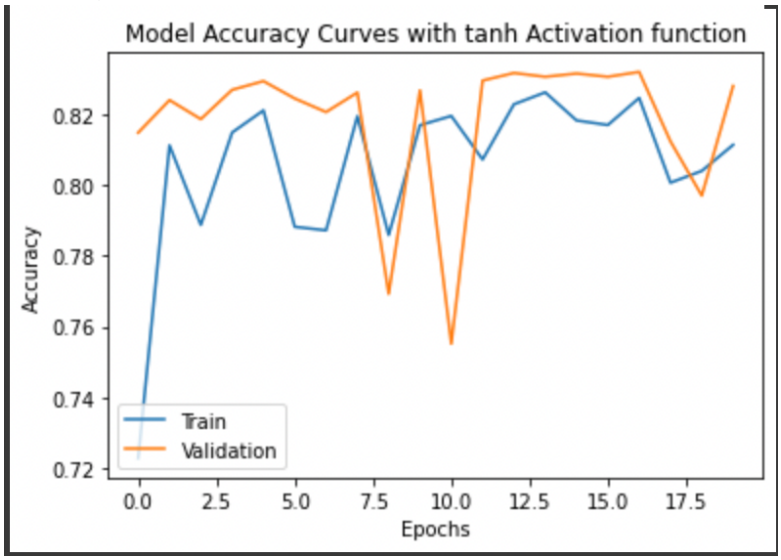
3. Adagrad - Here with Adagrad we have exponential decay of loss or convergence



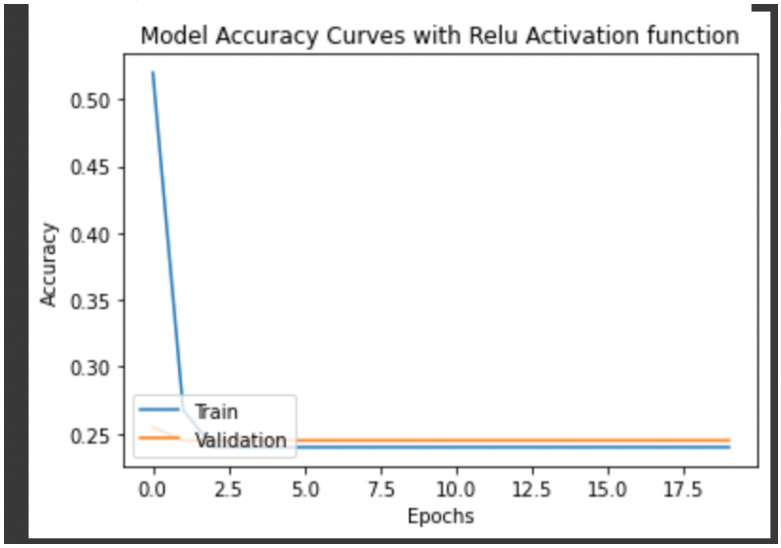
- Activation Function Tuning
    1. Sigmoid – With the sigmoid activation function we have an accuracy of around ~84.44%



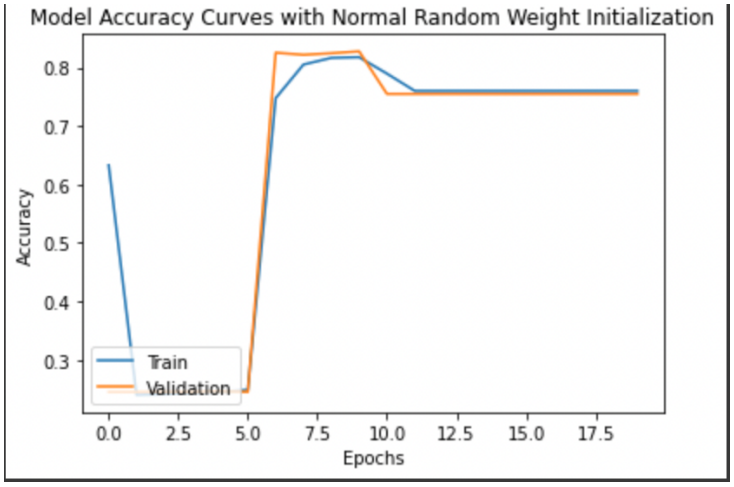
    2. Tanh – With tanh activation function we get a jagged plot with a maximum accuracy of 83.4%



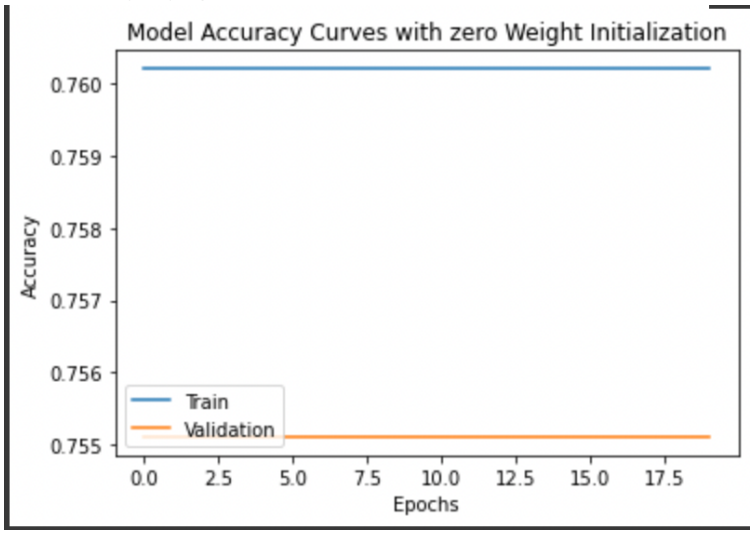
3. Relu – We get a very low accuracy of 23.5% with Relu Activation function



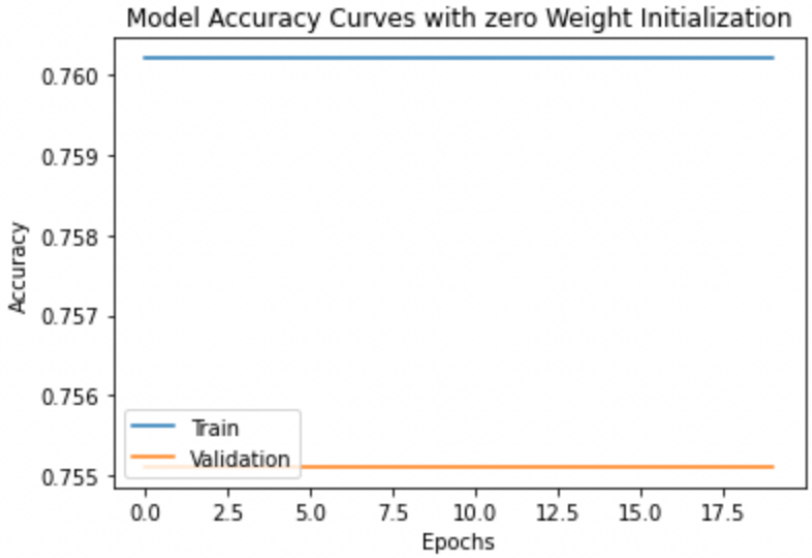
- Initializer Tuning
    1. Random normal – We are getting an accuracy of 76% with random weight initialization



2. Zeros weight - With zero initialization there is no learning as there is no gradient to be backpropagated

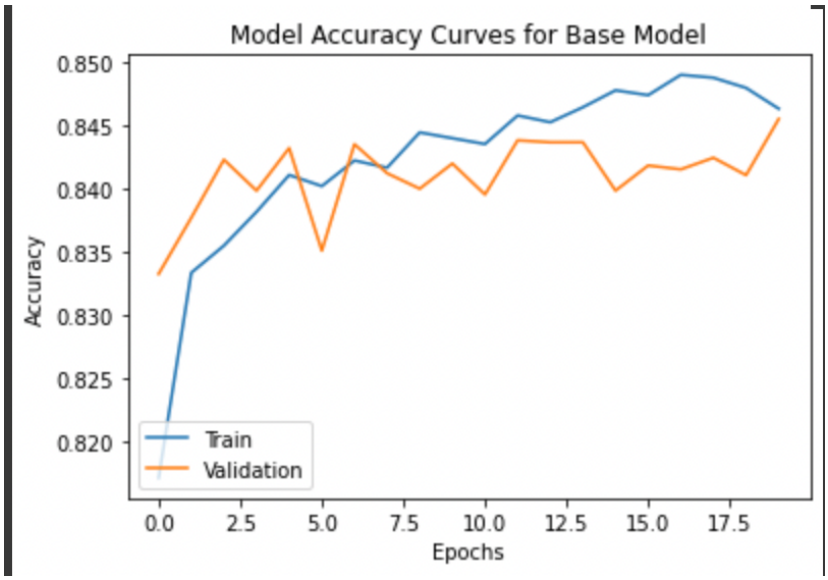


3. Ones

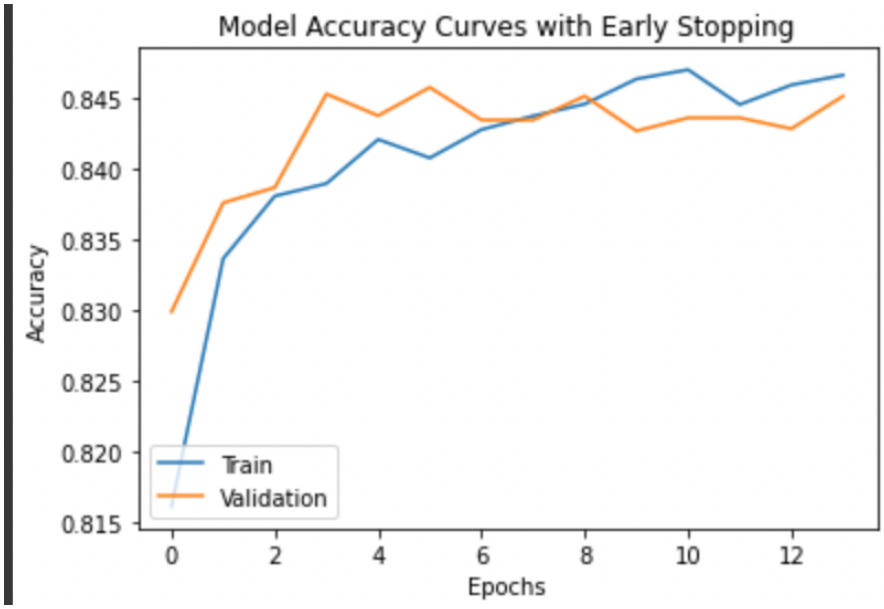


3. Provide a detailed analysis and reasoning about the NN setups that you tried.
    - Hidden Layers: As the requirement was to generate a model with hidden layers no more than 3, we decided to have 2 hidden layers. The reason being with 3 hidden layers there was a performance drop without any improvement in the accuracy as compared with a model with 2 hidden layers.
    - Dropout Tuning: The dropout tuning offers an effective regularization method to reduce overfitting and improve generalization error in the deep neural networks. Here in our model, we are using dropouts to have a few different network architectures by randomly dropping out nodes during the training. It is common practice to have a dropout value in the range 0-0.5 to achieve an effective model. However, as the dropout value crosses 0.5 most of the neurons would be dropped from the network and as a result the accuracy would low.
        1. The dropout values we tried are 0.2, 0.4 and 0.8. The accuracies obtained are as 84.44%, 82.22% and 72% respectively.
        2. Here we can observe a huge drop in the accuracy with dropout value of 0.8
    - Optimizer Tuning: Optimizers are the classes that are used to change the attributes of our model such as weights and learning rate to reduce the losses. Optimizers help to get results faster.
        1. In our setups, we are using three different optimizers namely RMSProp, Stochastic Gradient Descent and Adagrad.
        2. For each of the optimizers the accuracy obtained are: 84%, 82% and 83% respectively. Hence, the RMSProp provides us the highest accuracy values.
    - Activation Function Tuning: The activation function is a link between the between the input of the neuron and its output to the next layer. Our activation functions have direct impact on the accuracies and the loss obtained.
        1. Here we are utilizing the following activation functions: Sigmoid, tanh and relu.
        2. We are obtaining a maximum accuracy for the model using the sigmoid function which ranges between 83-85%.
    - Initializer Tuning: Initializers are used to define initial weight values in the keras models.
        1. In our setups we have used the following initializers: Random Weights, Zero Weights and One weights.
        2. We are achieving minimum accuracy in case of zero and one initialers as there is no gradient to be back propagated.

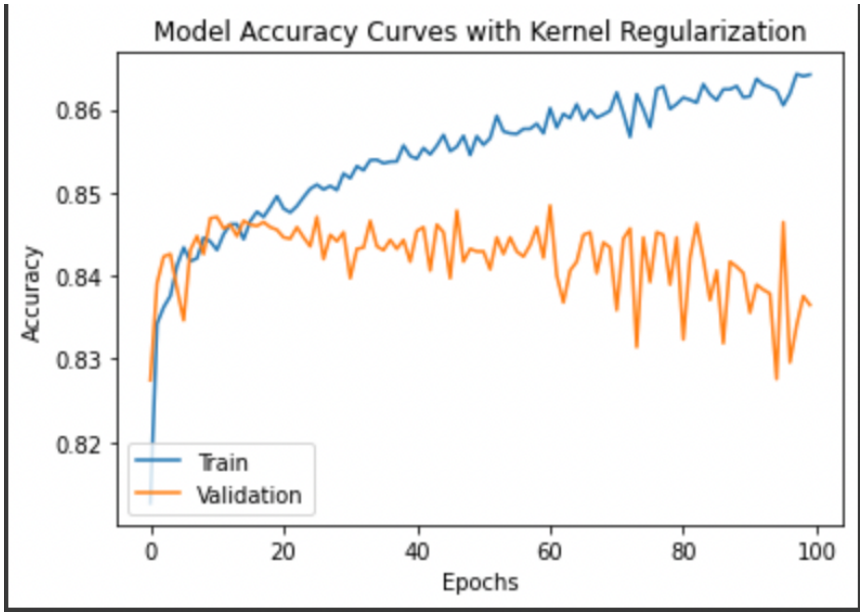
4. Briefly discuss all the methods you used that help to improve the accuracy or training time. Provide accuracy graphs and your short descriptions.
    - Base Model – We are choosing the model with dropout value = 0.2 as our base model



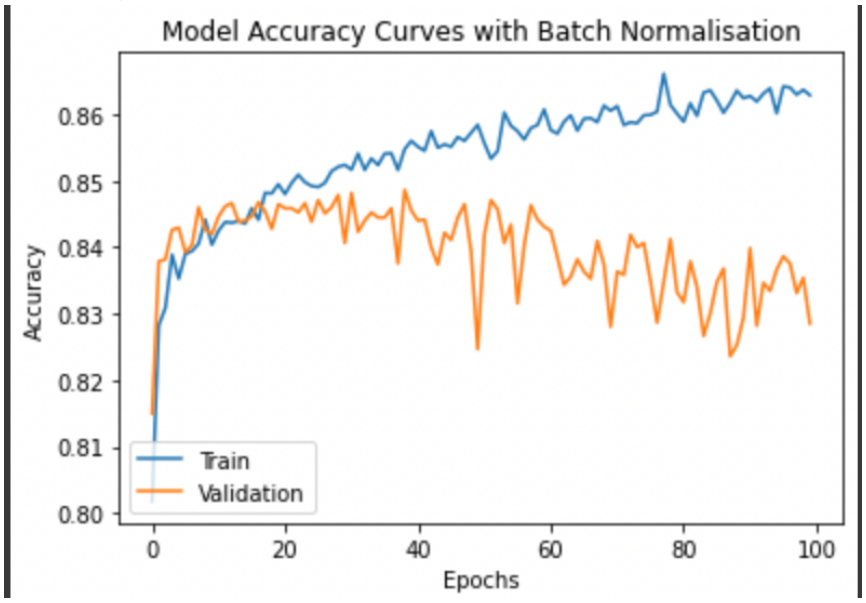
- Early Stopping: Earlystopping is a regularization technique that helps with overfitting. The Earlystopping feature stops the training when a specific condition has been met. In our case, we are stopping the model training when the '*val_loss*' parameter is maximum for 5 continuous epochs.



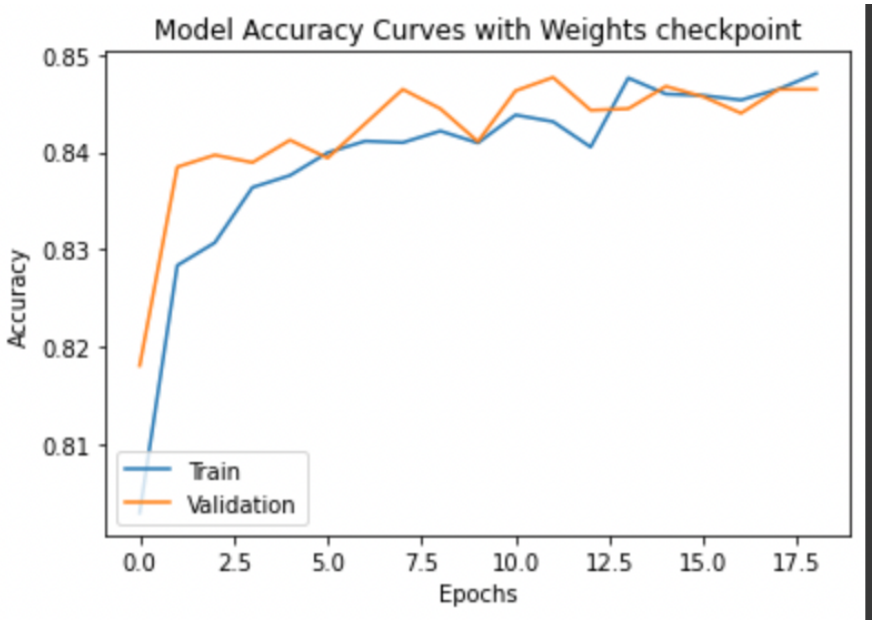
- Regularization L1 and L2- The regularization applies penalties on the layer's kernel during optimization. The penalties are then summed into the loss function that is optimized. Here we are using both L1 and L2 class regularizes.



1. Batch Normalization – The batch normalization is a layer that normalizes the input data. The layer essentially applies a transformation that maintains the mean output near to 0 and standard deviation near to 1.



2. Weights Checkpoint – The checkpoint function is used to store the weights values in a temporary location when a specific condition is met. In our case we are storing the weights when the val_accuracy is maximum. Once the weights have been saved, they would be used during the evaluation to achieve a maximum accuracy.



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.

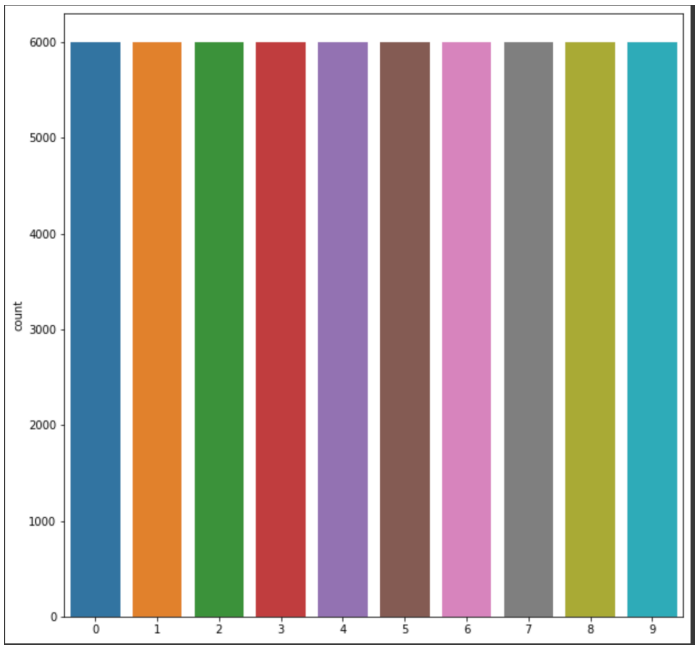
    - Fashion-MNIST is a dataset of clothing images. The dataset consists of a training set of 60,000 images and a test set of 10,000 images. Each of the images has a dimension of 28x28 and the is linked with one of the 10 classes. The 10 classes have a label associated as below:

Class Label Number	Class Label Value
0	T-shirt
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle Boot

    - The fashion-mnist is a suitable and challenging replacement for MNIST dataset and it can be used to build models with different neural architecture setup.

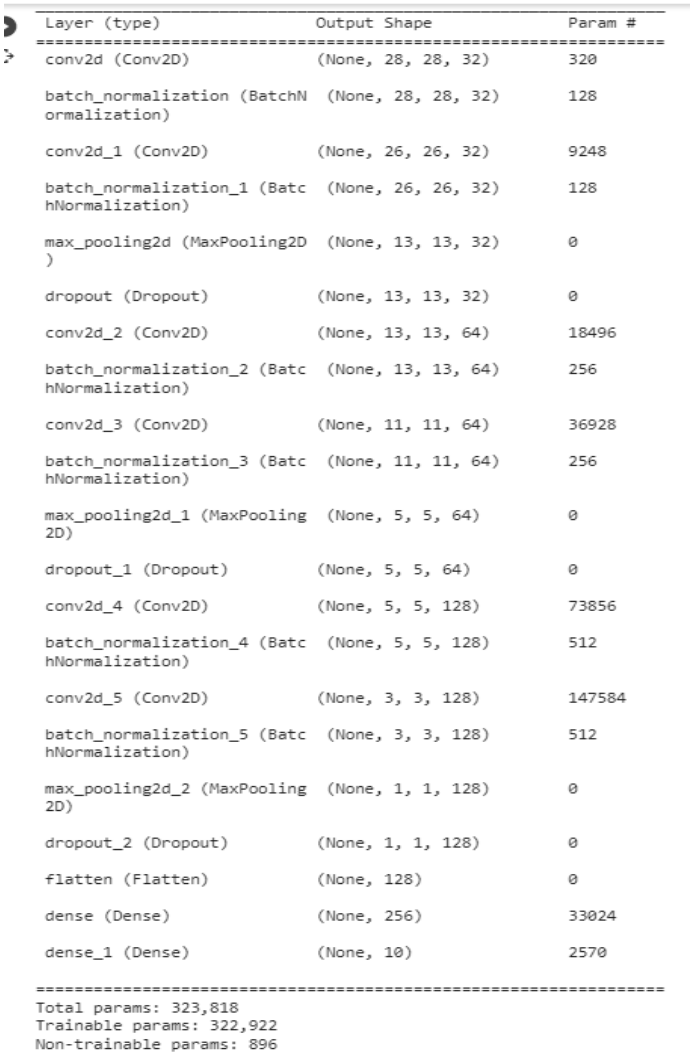
2. Provide at least 3 visualization graphs with short description for each graph.
    - The below visualization displays the grey-scale images of the fashion-mnist dataset before preprocessing.

    - The below visualization displays the count of each of the class in the fashion-mnsit dataset which indicates a cumulative value of 60000 class of images in the training dataset.



- The below image displays the fashion-mnsit images after preprocessing and normalization.

3. Provide the architecture structure of your CNN.

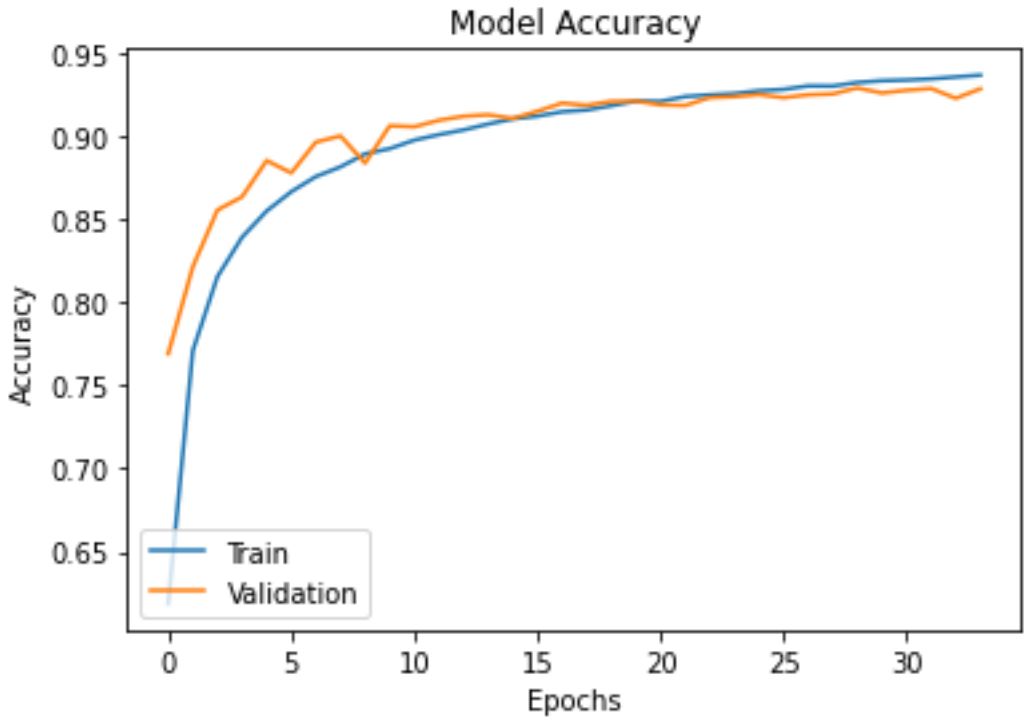


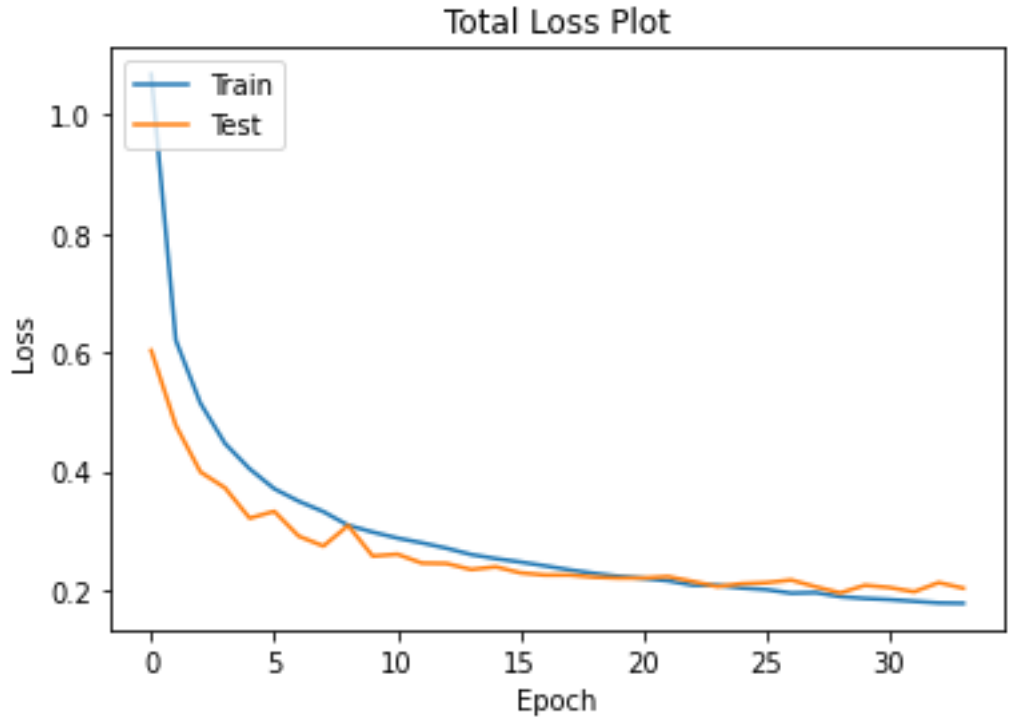
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 32)	320
batch_normalization (BatchNormalization)	(None, 28, 28, 32)	128
conv2d_1 (Conv2D)	(None, 26, 26, 32)	9248
batch_normalization_1 (BatchNormalization)	(None, 26, 26, 32)	128
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
dropout (Dropout)	(None, 13, 13, 32)	0
conv2d_2 (Conv2D)	(None, 13, 13, 64)	18496
batch_normalization_2 (BatchNormalization)	(None, 13, 13, 64)	256
conv2d_3 (Conv2D)	(None, 11, 11, 64)	36928
batch_normalization_3 (BatchNormalization)	(None, 11, 11, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
dropout_1 (Dropout)	(None, 5, 5, 64)	0
conv2d_4 (Conv2D)	(None, 5, 5, 128)	73856
batch_normalization_4 (BatchNormalization)	(None, 5, 5, 128)	512
conv2d_5 (Conv2D)	(None, 3, 3, 128)	147584
batch_normalization_5 (BatchNormalization)	(None, 3, 3, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 1, 1, 128)	0
dropout_2 (Dropout)	(None, 1, 1, 128)	0
flatten (Flatten)	(None, 128)	0
dense (Dense)	(None, 256)	33024
dense_1 (Dense)	(None, 10)	2570

Total params: 323,818
Trainable params: 322,922
Non-trainable params: 896

4. Discuss how the improvement tools works on CNN architectures.
    - *Earlystopping*: The earlystopping option which we utilized here stops the model training when the '*val_loss'* is minimum for 5 consecutive epochs. The model won't stop training if the loss value is reducing further, however as soon as the loss stabilizes i.e., there is no decrease in the loss value the model stops training. The earlystopping is used in combination with the weights checkpoint tool.
    - *Weights Checkpoint*: The purpose of the weights checkpoint tool is that it saves the weight values in a temporary location whenever a maximum val_accuracy value is obtained during the training. As soon as the training stops due to the earlystopping tool, the saved weights are used for the evaluation.

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.





Part IV: Optimizing CNN + Data Augmentation

1. Include all 3 tables with different CNN setups.
    - Dropout Tuning:

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.15,0.20,0.25	93.01%	0.10,0.20,0.30	92.10%	0.1,0.1,0.1	92.43%
Optimizer	adam		adam		adam	
Activation Function	relu		relu		relu	
Initializer	Glorot Normal		Glorot Normal		Glorot Normal	
Kernel size	(3, 3)		(3, 3)		(3, 3)	

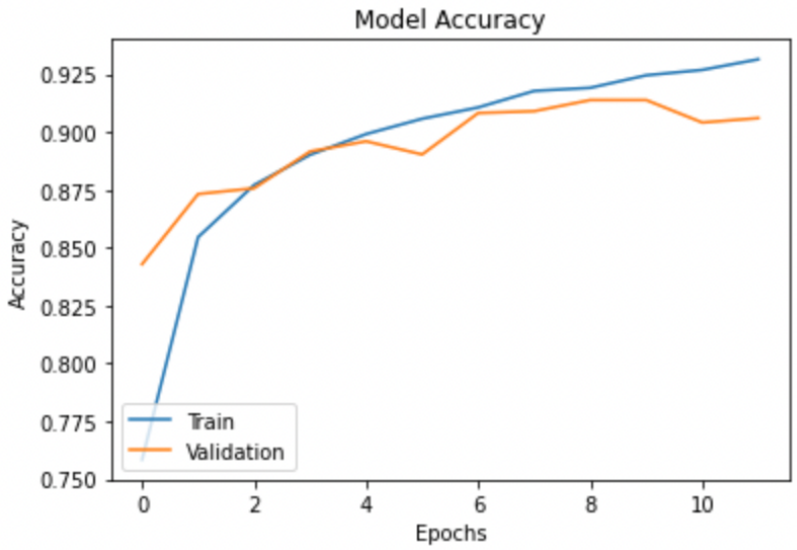
    - Optimizer Tuning:

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.15,0.20,0.25		0.15,0.20,0.25		0.15,0.20,0.25	
Optimizer	adam	92.57%	RMS prop	92.03%	Adagrad	89.86%
Activation Function	relu		relu		relu	
Initializer	Glorot Normal		Glorot Normal		Glorot Normal	
Kernel size	(3, 3)		(3, 3)		(3, 3)	

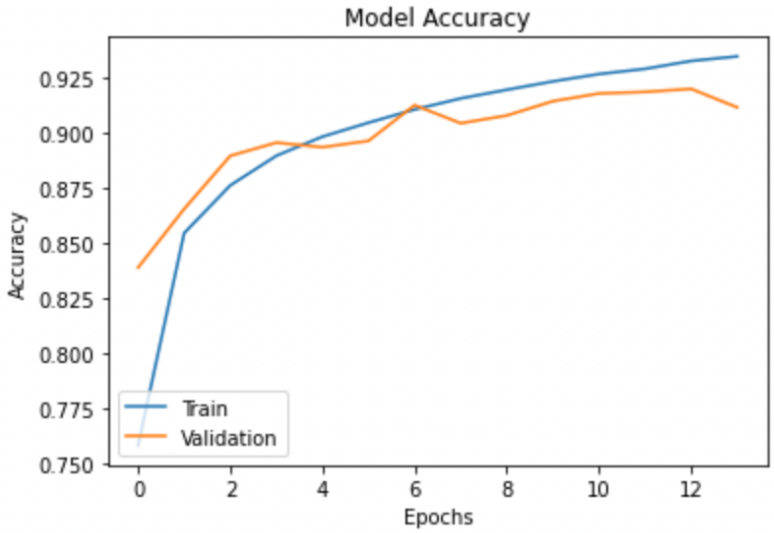
- Kernel Size Tuning:

	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Dropout	0.15,0.20,0.25		0.15,0.20,0.25		0.15,0.20,0.25	
Optimizer	adam		adam		adam	
Activation Function	relu		relu		relu	
Initializer	Glorot Normal		Glorot Normal		Glorot Normal	
Kernel size	(3, 3)	92.77%	(5, 5)	92.05%	(7, 7)	90%

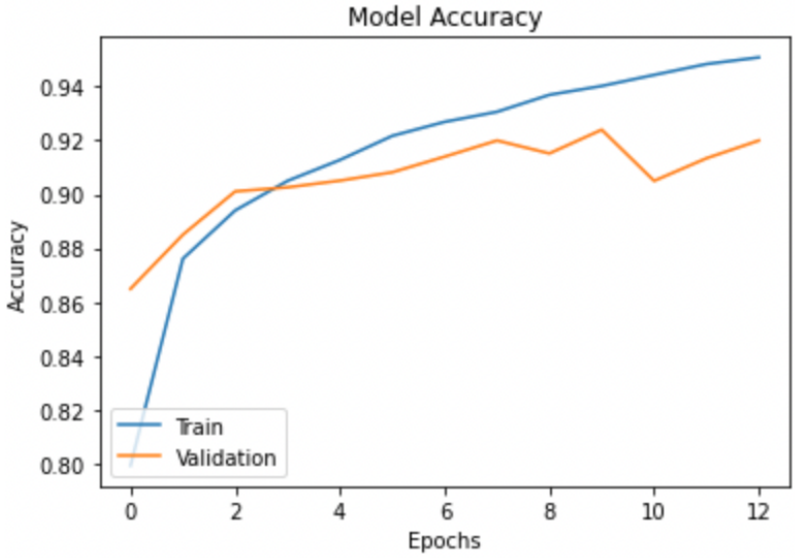
2. Provide graphs that compares test and training accuracy on the same plot for all your setups and add a short description for each graph.
    - Dropout Tuning (0.15, 0.20, 0.25) – The accuracy obtained with these dropout values is 90.06%



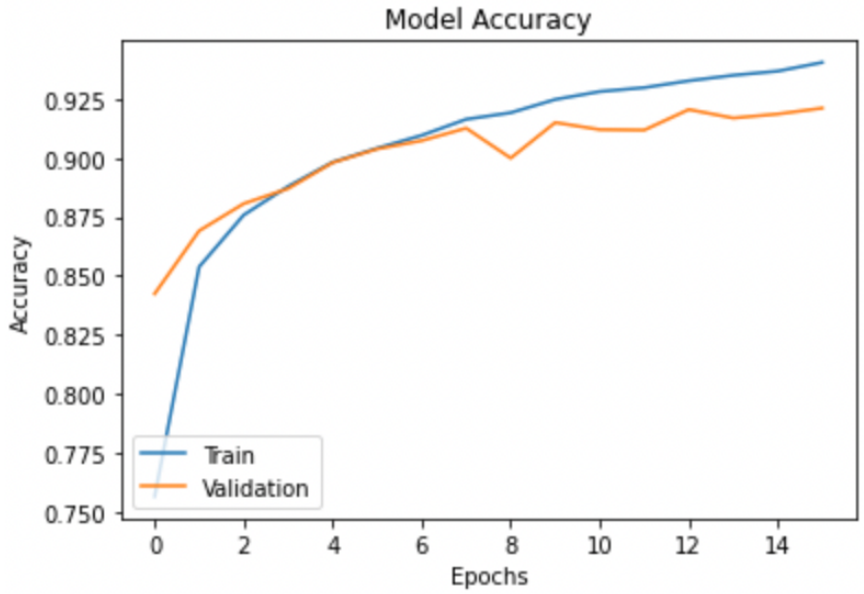
- Dropout Tuning (0.10, 0.20, 0.30) - The accuracy obtained with these dropout values is 91.17%



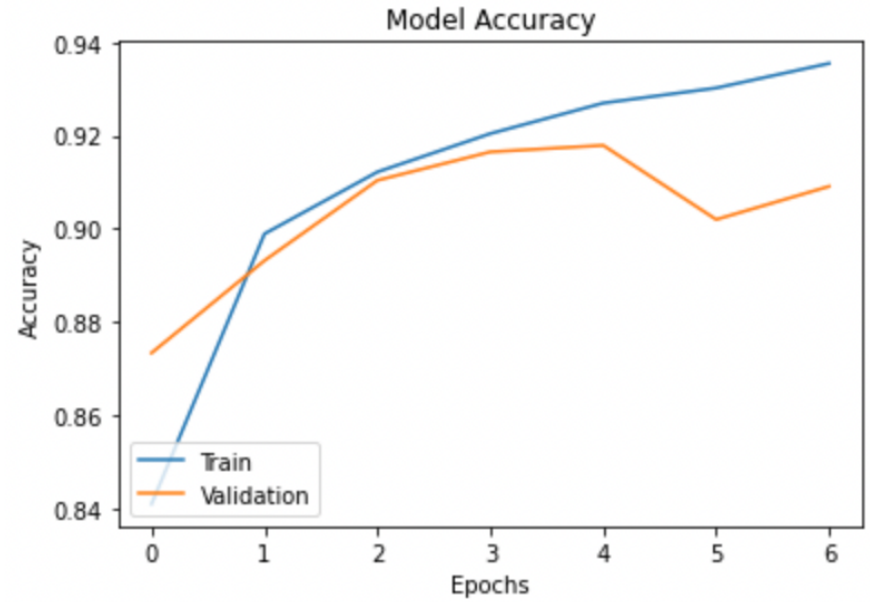
- Dropout Tuning (0.1, 0.1, 0.1) - The accuracy obtained with these dropout values is 92.0%



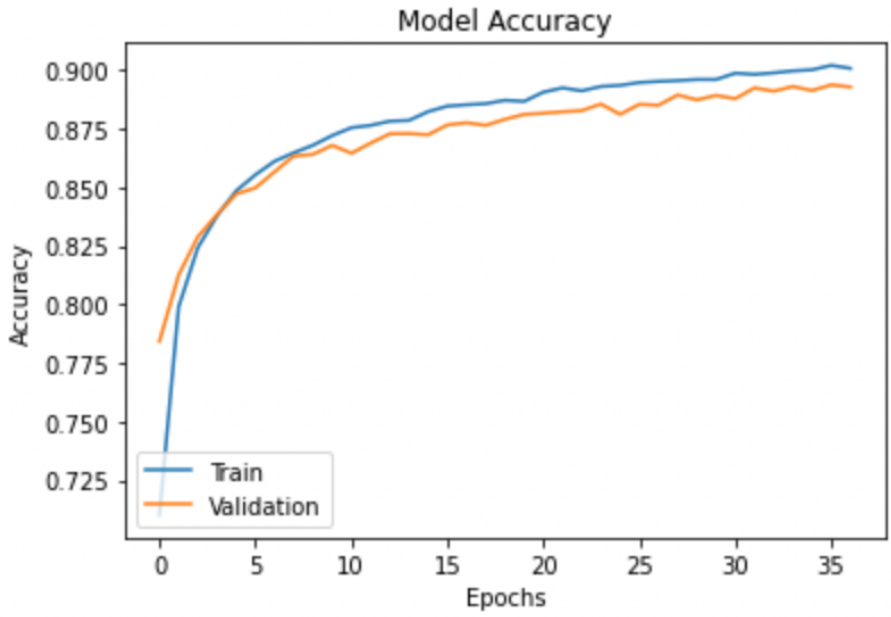
- Optimizer Tuning (Adam) - The accuracy obtained Adam optimizer is 92.1%



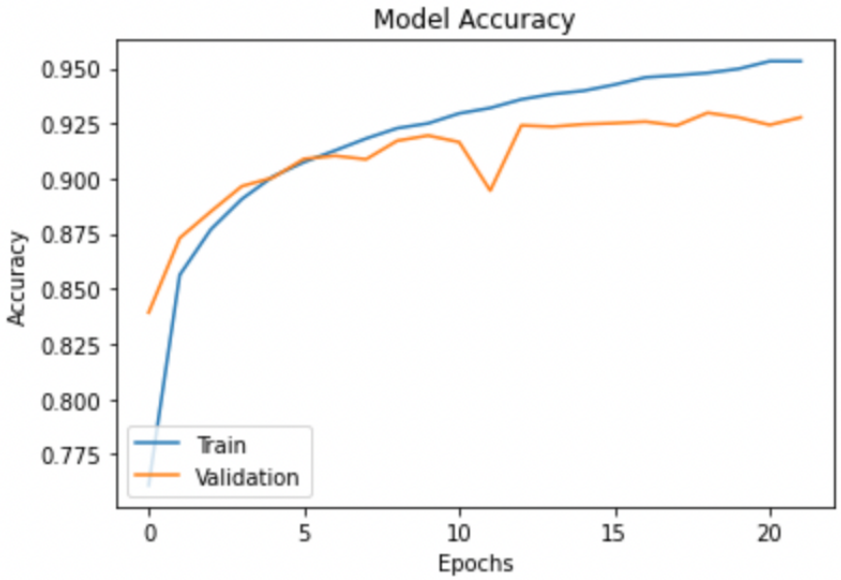
- Optimizer Tuning (RMSProp) - The accuracy obtained Adam optimizer is 90.91%



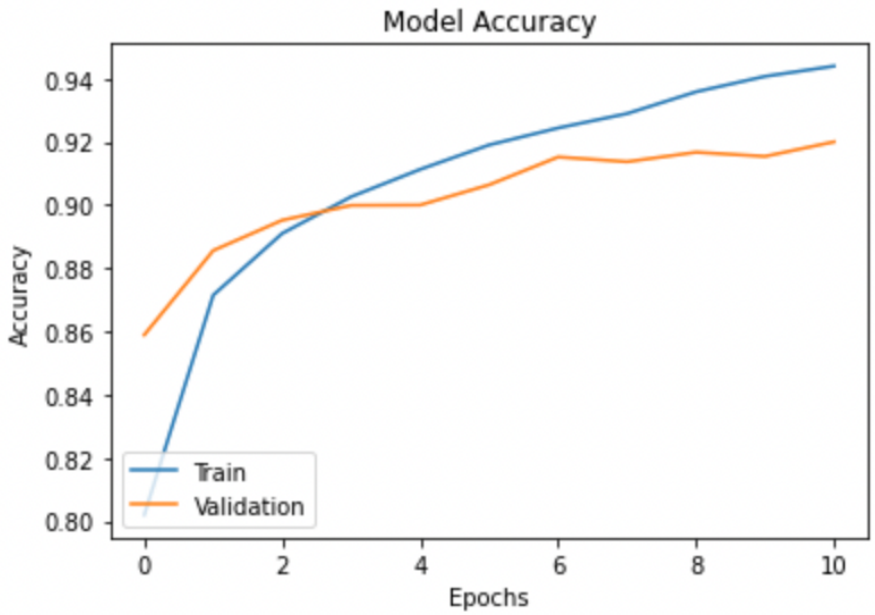
- Optimizer Tuning (Adagrad) - The accuracy obtained Adam optimizer is 89.29%



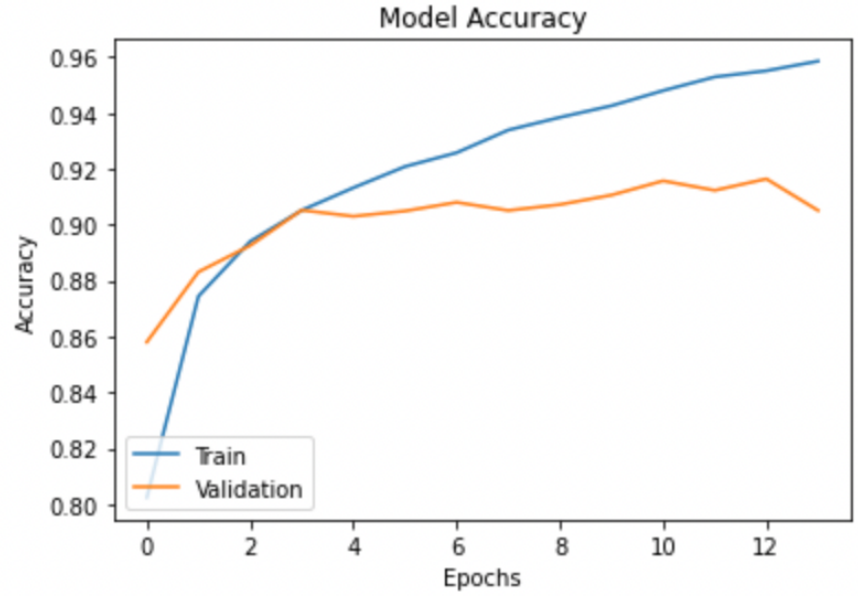
- Kernel Size Tuning (3, 3) - The accuracy obtained with kernel_size=(3,3) is 92.7%



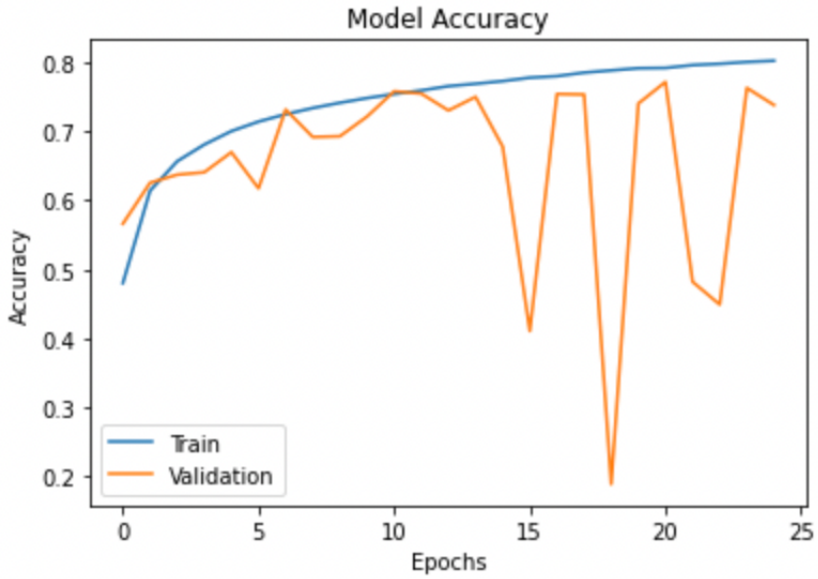
- Kernel Size Tuning (5, 5) - The accuracy obtained with kernel_size=(5,5) is 92.0%



- Kernel Size Tuning(7, 7) - The accuracy obtained with kernel_size=(7,7) is 90.5%



- Data Augmentation:



3. Provide a detailed analysis and reasoning about the CNN setups that you tried.
    - Dropout Tuning: The dropout tuning offers an effective regularization method to reduce overfitting and improve generalization error in the deep neural networks. Here in our model, we are using dropouts to have a few different network architectures by randomly dropping out nodes during the training. It is common practice to have a dropout value in the range 0-0.5 to achieve an effective model. However, as the dropout value crosses 0.5 most of the neurons would be dropped from the network and as a result the accuracy would low.
        1. In this setup we tried different set of dropout values for each layer.
        2. The values we tried are: (0.15, 0.20, 0.25), (0.10, 0.20, 0.30) and (0.1, 0.1, 0.1).
        3. We obtained a maximum accuracy of 92% when the dropout value was same and minimum across all the layers. When we tried a dropout value of 0.2 across all the layers even though we were achieving a maximum accuracy of 93%, it was not consistent during the runs.
    - Optimizer Tuning: Optimizers are the classes that are used to change the attributes of our model such as weights and learning rate to reduce the losses. Optimizers help to get results faster.

1. In our setups, we are using three different optimizers namely RMSProp, Adam and Adagrad.
2. For each of the optimizers the accuracy obtained are: 90.91%, 92.1% and 89.92% respectively. Hence, the adagrad optimizer provides us the lowest accuracy values whereas the Adam optimizer provides the highest.

- Kernel Size Tuning: The kernel size in a keras model indicates the height and width of the convolution model.
    1. In our setups, we are using three different kernel size values: (3,3), (5,5) and (7,7).
    2. Even though there is no direct relationship between kernel size and the accuracy. The kernel size setup which are trying here is to check if using a kernel of specific size covers majority of the features. A larger kernel size may detect the features missed by the smaller kernel size and vice-versa.
    3. Here we are achieving the accuracy of the values: 92.77%, 92.0% and 90.05%

4. Briefly discuss methods that you used for data augmentation and reason how it helped to increase the accuracy of the model.
    - We used the following data augmentation methods:

Horizontal Flip
Vertical Flip
90 Degree Rotation
Zoom

    - The data augmentation process is generally utilized when the input dataset size is insufficient for training the model, so using the augmentation process increases the size of the data.

BONUS QUESTION

1. The accuracy obtained using the perceptron built from scratch is 52.23%
2. The accuracy obtained using the Logistic Regression is 88.05%

CONTRIBUTION:

Team Member	Assignment Part	Contribution
Sachin Sarsambi	Part 1	50%
Naman Tejaswi		50%
Sachin Sarsambi	Part 2	50%
Naman Tejaswi		50%
Sachin Sarsambi	Part 3	50%
Naman Tejaswi		50%
Sachin Sarsambi	Part 4	50%
Naman Tejaswi		50%

References:

1) https://keras.io/api/
2) https://keras.io/api/models/sequential/
3) https://numpy.org/
4) https://keras.io/api/models/model_training_apis/#fit-method
5) https://keras.io/api/callbacks/model_checkpoint/
6) https://keras.io/api/optimizers/
7) https://keras.io/api/layers/activations/
8) https://keras.io/api/callbacks/early_stopping/
9)https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator