Assignment- 3

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.

Team Member	Assignment Part	Contribution(%)
Anudeep Balagam	Part 1	100%
Divya Sharvani Kandukuri	Part 2	100%
Anudeep Balagam	Part 3	100%
Divya Sharvani Kandukuri	Part 4	100%

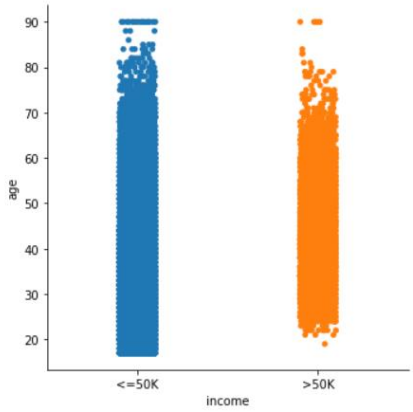
Part 1

Building a Basic NN

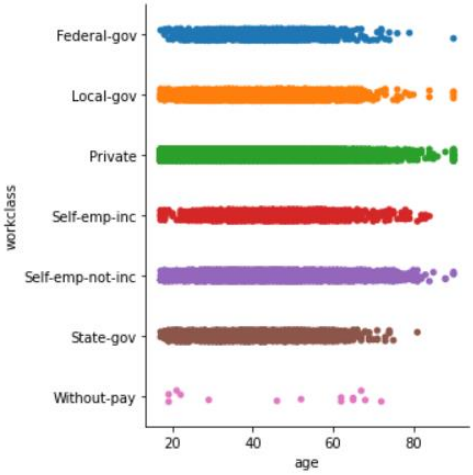
1. The given dataset describes how different factors influence the income of an individual like age, education, country, occupation etc. The details include their age,Workclass, fnlwgt, education, marital status, occupation, relationship, capital gain, capital loss, hours per week, native country. We have different types of data in this dataset like string, float64 and int64. There are 32561 entries and 11 variables in the dataset.

```
In [419]: indf.describe

Out[419]: <bound method NDFrame.describe of            age  workclass    fnlwgt  education.num  marital_status  \
1      0.890411          2  0.117131              9               6
3      0.506849          2  0.086061              4               0
4      0.328767          2  0.170568             10               5
5      0.232877          2  0.138072              9               0
6      0.287671          2  0.093024              6               5
...         ...        ...       ...            ...             ...
32556  0.068493          2  0.201493             10               4
32557  0.136986          2  0.165563             12               2
32558  0.315068          2  0.095589              9               2
32559  0.561644          2  0.093914              9               6
32560  0.068493          2  0.127620              9               4

       occupation  relationship  capital.gain  capital.loss  hours.per.week  \
1               3             1             0      1.000000        0.173469
3               6             4             0      0.895317        0.397959
4               9             3             0      0.895317        0.397959
5               7             4             0      0.865473        0.448980
6               0             4             0      0.865473        0.397959
...           ...           ...           ...           ...             ...
32556          10             1             0      0.000000        0.397959
32557          12             5             0      0.000000        0.377551
32558           6             0             0      0.000000        0.397959
32559           0             4             0      0.000000        0.397959
32560           0             3             0      0.000000        0.193878

       native_country
1                  38
3                  38
4                  38
5                  38
6                  38
...               ...
32556              38
32557              38
32558              38
32559              38
32560              38

[30162 rows x 11 columns]>
```

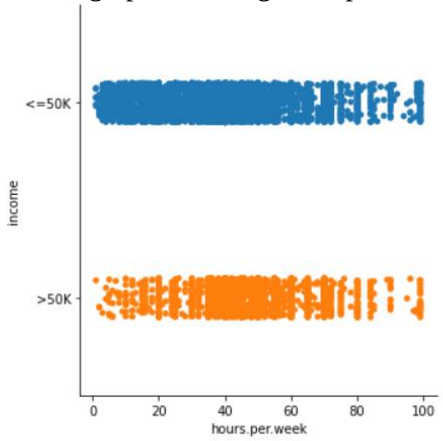
2. Graphs:



The above graph is a categorical plot between income and age. We can observe that age is not really playing an important role in determining income as a lot of people in the same age group earn less than 50k and almost same number of people earn more than 50k as well.



The above graph is a categorical plot between age and working class.



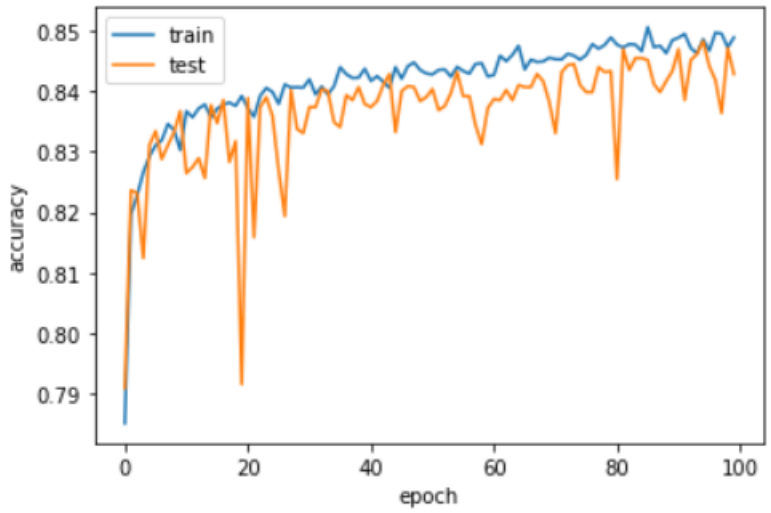
The above graph is a categorical plot between hours per week and income.

3. We used various methods to make our dataset the best possible input to the neural network. They are as follows:

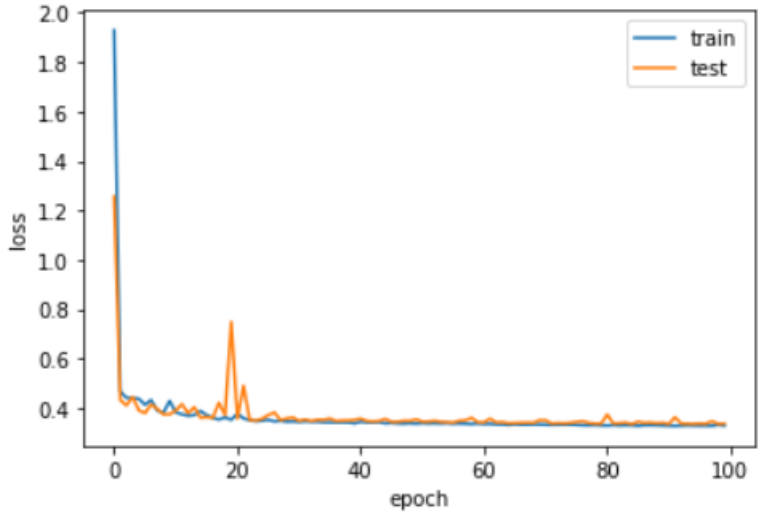
a. Encoding the variables using LabelEncoder from scikit-learn – as the categorical variables cannot be processed mathematically by the model, we converted them into numerical by encoding them.
b. Normalized the dataset – we normalized the variables which have large values to values between 0 and 1
c. Splitting into testing and training data – we used train_test_split from scikit-learn to divide the data into training(80%) and testing(20%) datasets.

4. The neural network consists of 3 hidden layers with 50 nodes in the first hidden layer, 25 nodes in the second hidden layer and 10 nodes in the third. We used selu activation function in 3 hidden layers. The output layer has 1 node and sigmoid activation function for the output.

5. Accuracy graph:



Loss graph:



Part 2

Optimizing NN

1. Initial model from step 1 uses:

   Activation function: selu

   Loss: binary cross entropy

   Optimizer: Adam

Table 1:

Hyperparameter	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Activation function	selu	Training = 84.22% Testing = 83.75%	selu	Training =77.52 % Testing = 76.77%	selu	Training = 84.24% Testing = 83.82%
Loss	Binary crossentropy		Binary crossentropy		Binary crossentropy	
Optimizer	Adamax		Adadelta		Ftrl	

Table 2:

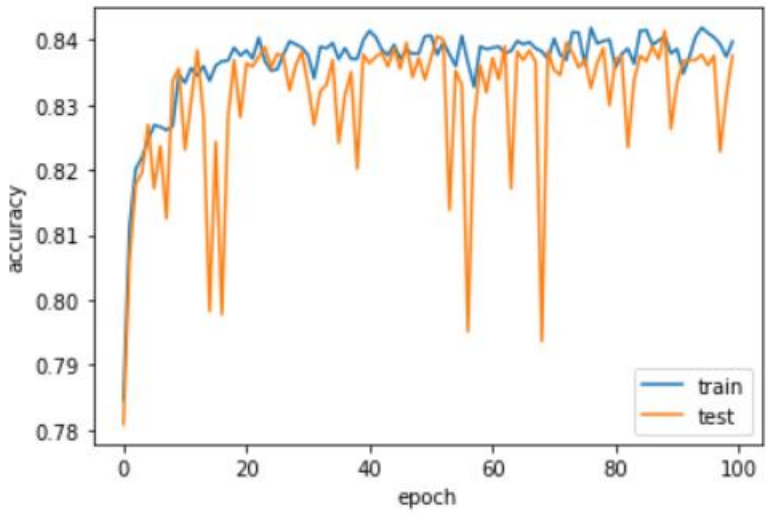
Hyperparameter	Setup 4	Accuracy	Setup 5	Accuracy	Setup 6	Accuracy
Activation function	Relu	Training = 84.52% Testing = 83.83%	elu	Training = 84.19% Testing = 83.88%	softsign	Training = 84.56% Testing = 83.93%
Loss	Binary crossentropy		Binary crossentropy		Binary crossentropy	
Optimizer	Adamax		Adamax		Adamax	

Table 3:

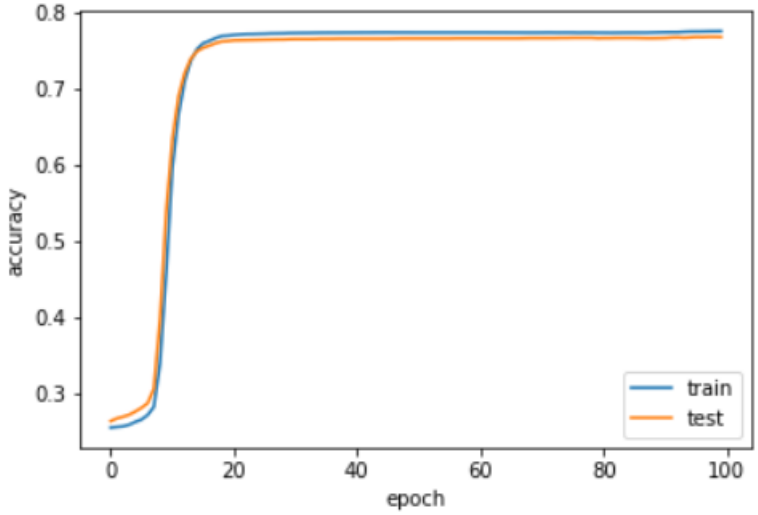
Hyperparameter	Setup 7	Accuracy	Setup 8	Accuracy	Setup 9	Accuracy
Activation function	relu	Training = 82.34% Testing = 81.68%	relu	Training = 75.28% Testing = 74.39%	relu	Training = 24.71% Testing = 25.60%
Loss	poisson		Categorical crossentropy		Kl_divergence	
Optimizer	Adamax		Adamax		Adamax	

2. Accuracy graphs for all the setups:

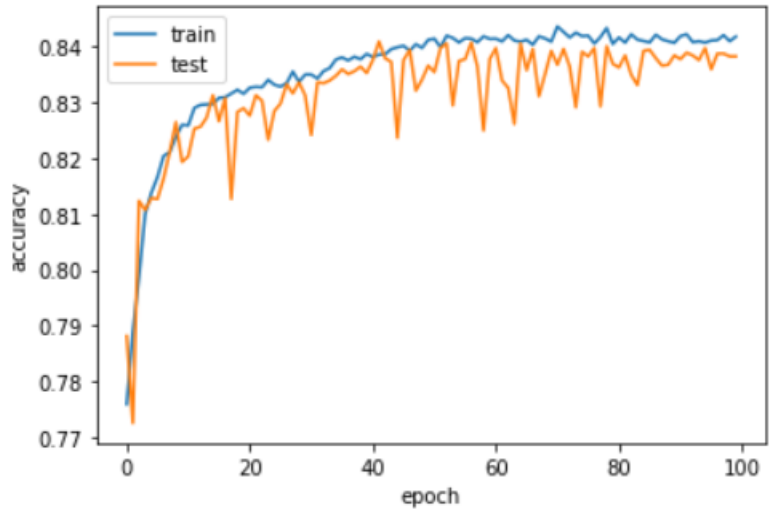
   Setup 1:



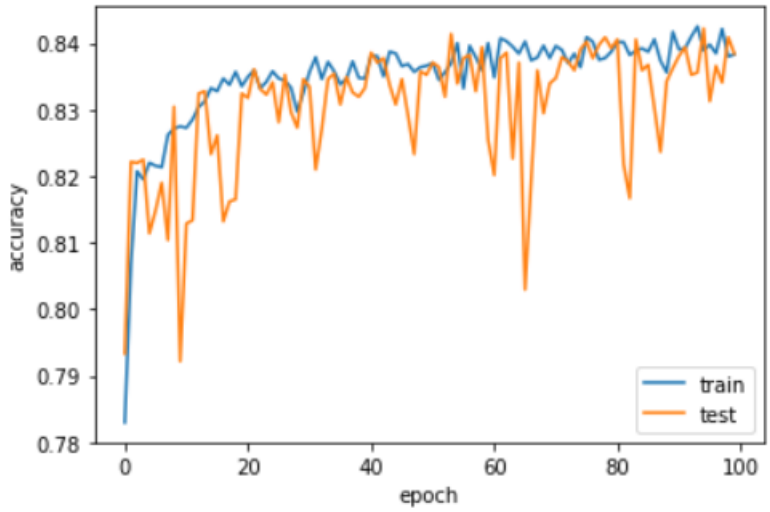
Setup 2:



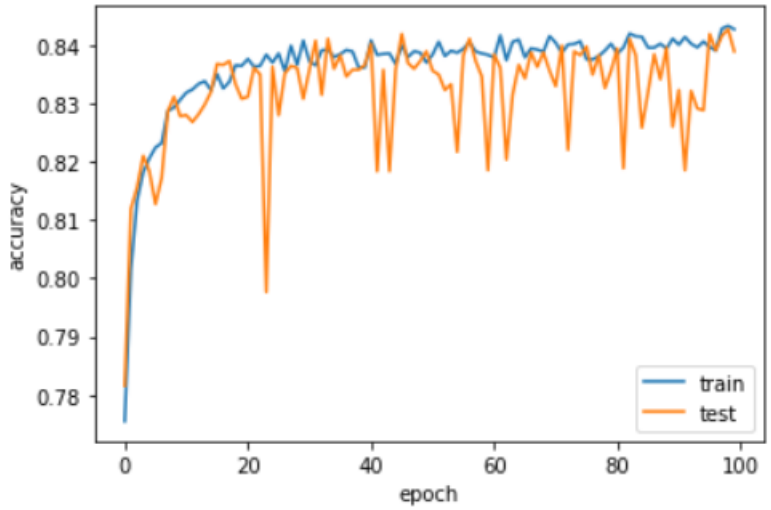
Setup 3:



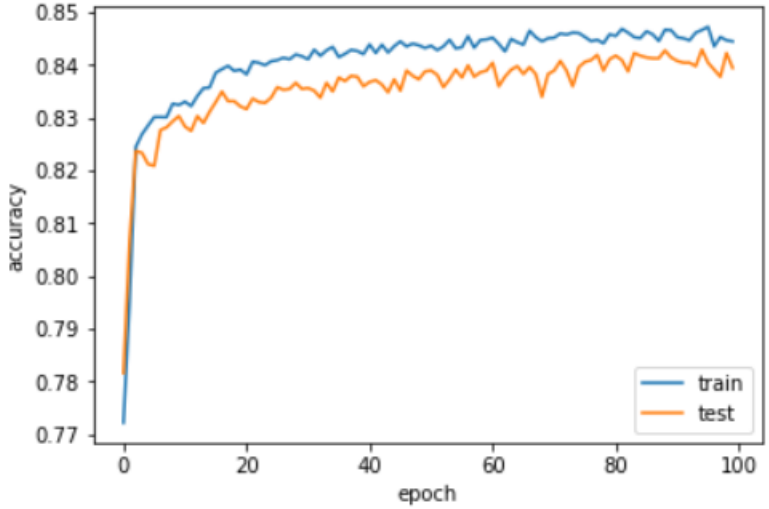
Setup 4:



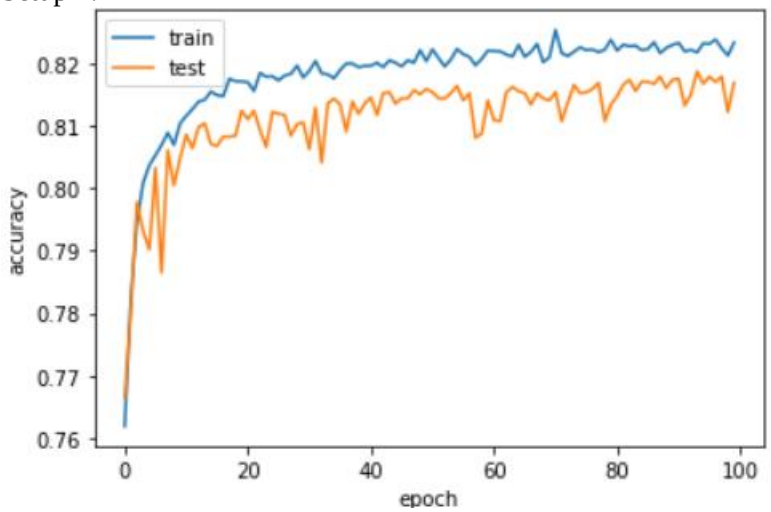
Setup 5:



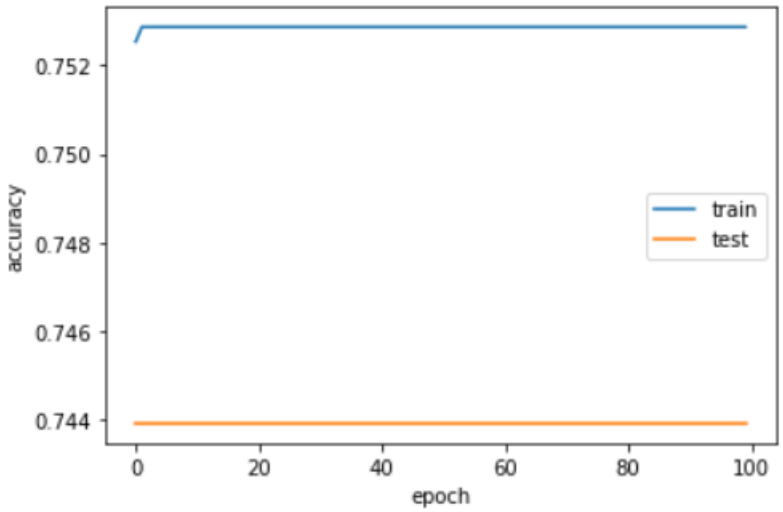
Setup 6:



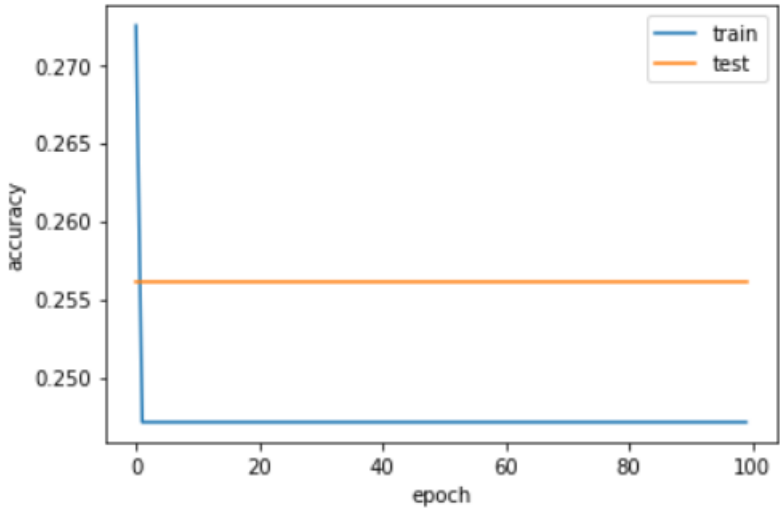
Setup 7:



Setup 8:



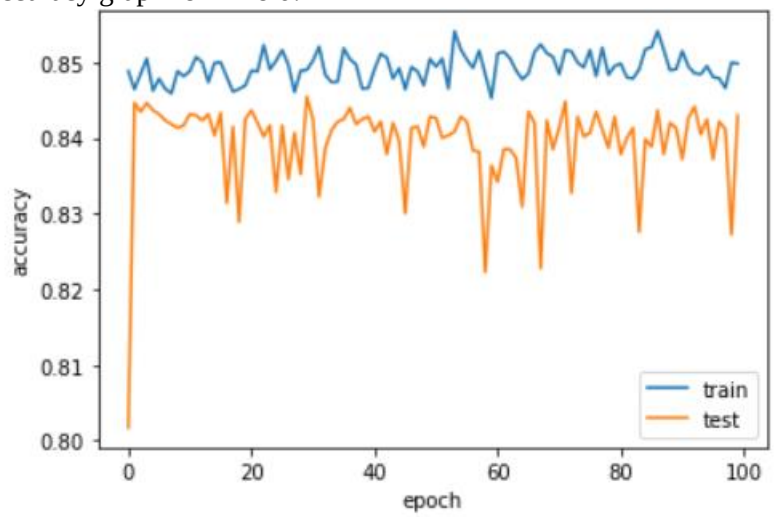
Setup 9:



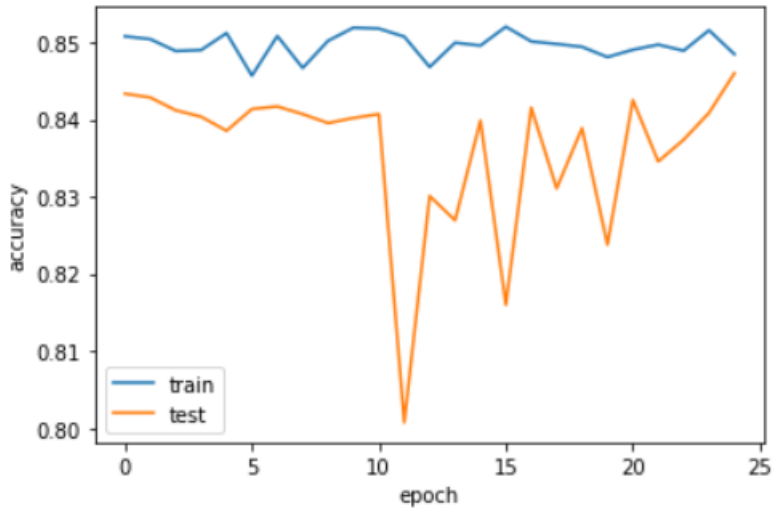
3. Table 1: In this table, we have changed the optimizer for the Neural Network setups. We used Adamax, Adadelta and Ftrl. Ftrl gave the highest testing accuracy in this run.
   Table 2: In this table, we have changed the activation function for the Neural Network setups. We used relu, elu and softsign. Softsign gave the highest testing accuracy in this run.
   Table 3: In this table, we have changed the loss for the Neural Network setups. We used poisson, categorical cross entropy and KL Divegence. Poisson gave the highest testing accuracy in this run.

4. The methods which we used are:
   K-fold
   Early stopping
   Dropout
   Stratified K-fold
   Among these methods, early stopping yielded the highest testing accuracy of 84.59%.

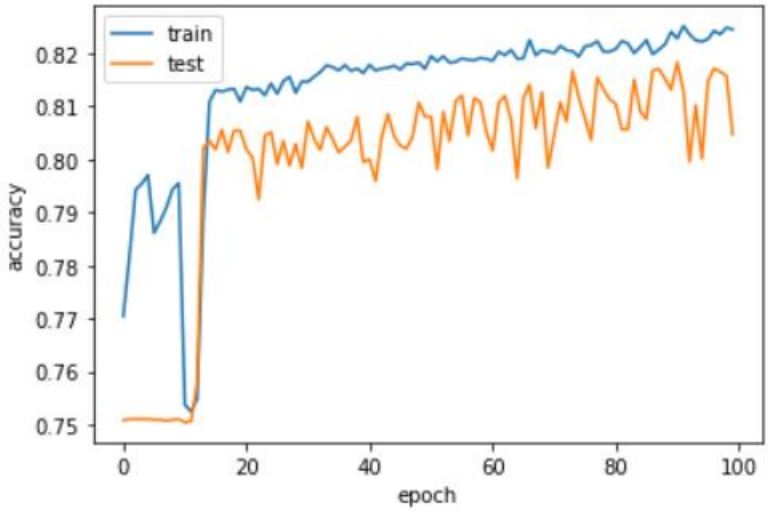
Accuracy graph for K-fold:



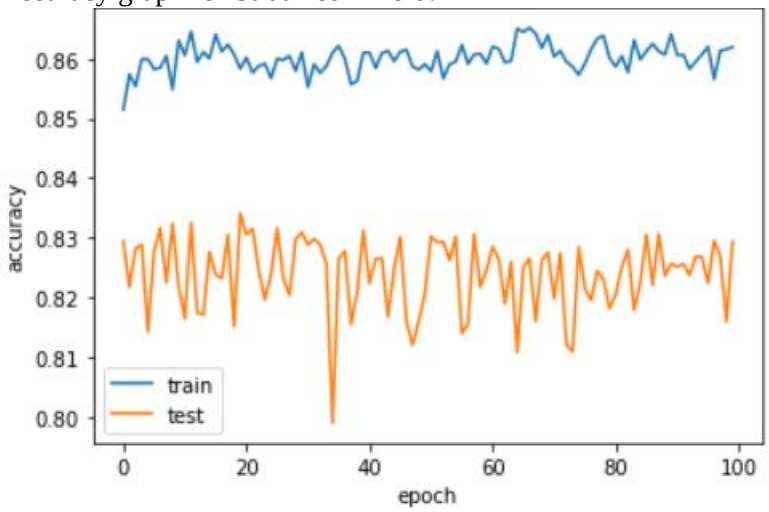
Accuracy graph for Early stopping:



Accuracy graph for Dropout:



Accuracy graph for Stratified K-fold:



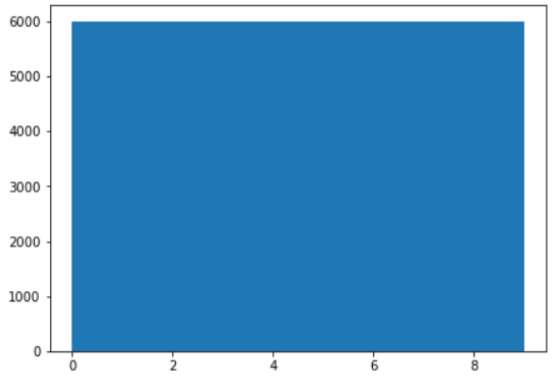
Part 3

Building a CNN

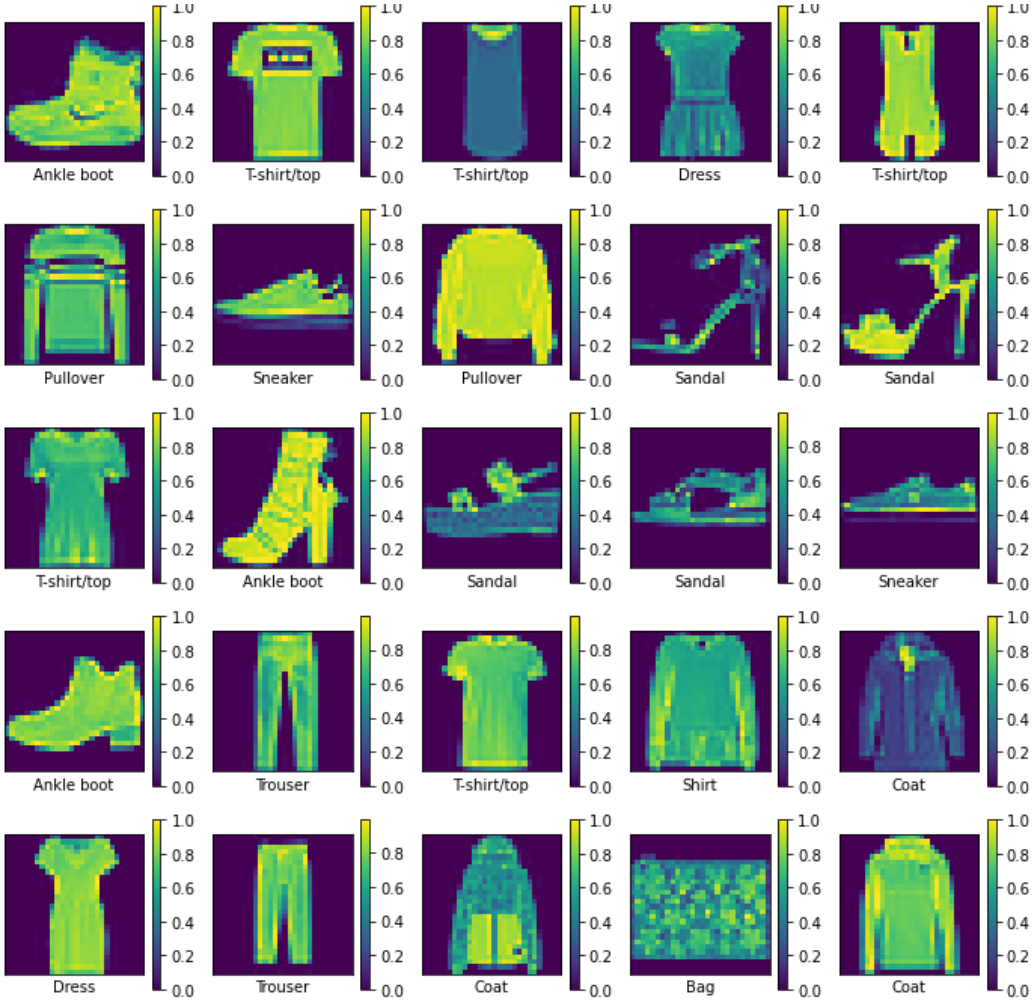
1. The given dataset contains a total of 70000 images of different types of clothes and accessories. The whole data is divided into 10 classes which are T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot in the dataset. We use 60000 images for training purposes and remaining 10000 for testing our model.
   Statistics:

```
In [4]: X_train.shape
Out[4]: (60000, 28, 28)

In [5]: X_test.shape
Out[5]: (10000, 28, 28)

In [6]: y_train.shape
Out[6]: (60000,)

In [7]: y_test.shape
Out[7]: (10000,)
```

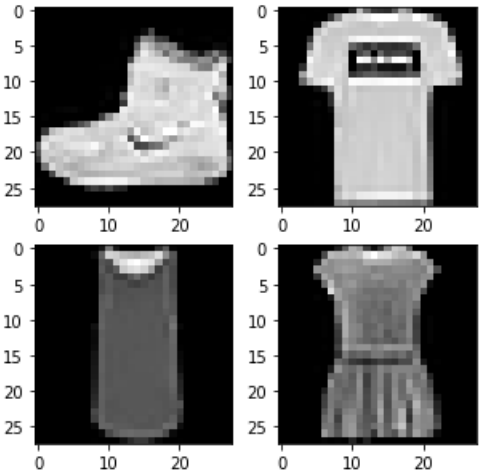
2. 3 visualization graphs



   The above graph represents how many images are present in every category we have. 0-9 on the X-axis are the encoded categories (shoe, trousers etc) and Y-axis has number of images in the category. We can see that in training dataset there are 6000 images in all 10 categories.



The above graph displays the processed images and their heatmaps with their label names.



The above graph is a representation of the gray scale images which were given as input to the neural network.

3. Architecture of CNN:
   - CNN consists of 3 hidden layers.
   - Two of them are Convolutional 2D layers. They have the following initial parameters:

Parameter	Layer 1	Layer 2
Nodes	64	128
Activation function	Relu	Relu
Padding	Same	Same
Kernel size	3x3	3x3

   - Note: same value for padding means that the image will be padded evenly with 0's on all four sides.
   - The third hidden layer is a Dense layer with 128 nodes and activation function as relu.
   - The output layer for this network is a dense layer with 10 nodes and softmax activation function.

4. The methods which we used are:
   K-fold
   Dropout
   Early stopping
   Among these methods, K-Fold when used to split only the training data for both training and testing yielded the highest testing accuracy of 95%.

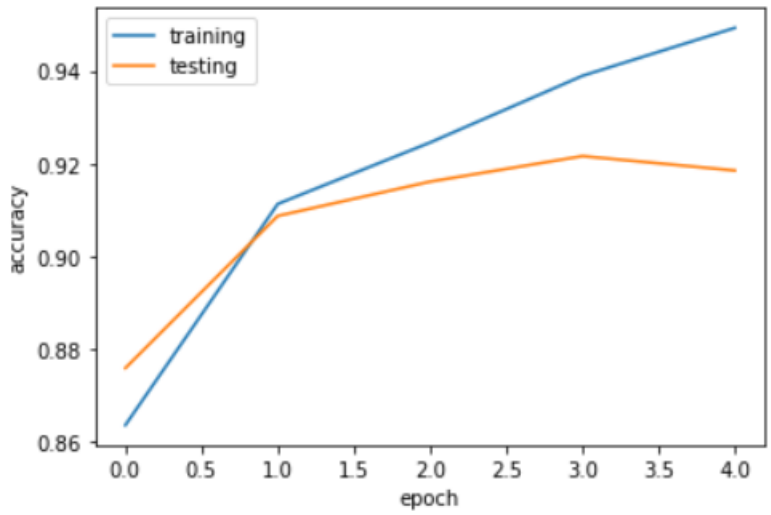
5. Training accuracy : 95.72%

```
In [137]: _, accuracy = model.evaluate(X_train,y_train)
          print("Training accuracy: ", accuracy*100)

          1875/1875 [==============================] - 26s 13ms/step - loss: 0.1127 - accuracy: 0.9572
          Training accuracy:  95.72333097457886
```

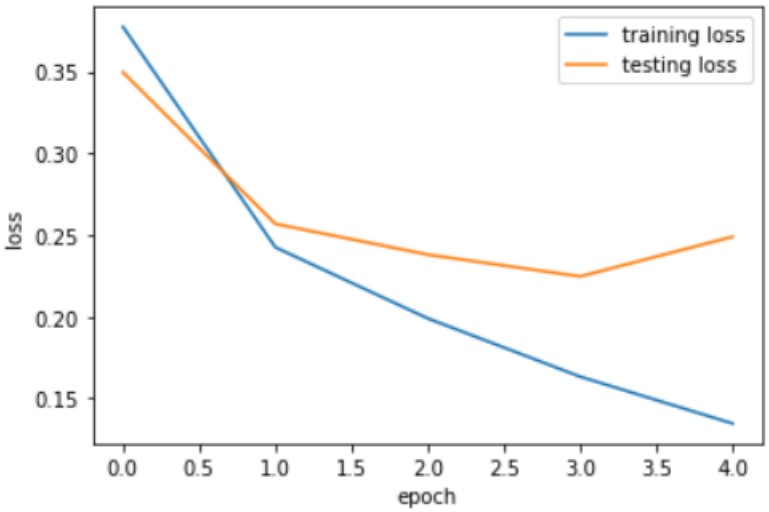
   Testing accuracy: 91.85%

```
In [138]: _, accuracy = model.evaluate(X_test,y_test)
          print("Testing accuracy: ", accuracy*100)

          313/313 [==============================] - 4s 13ms/step - loss: 0.2489 - accuracy: 0.9185
          Testing accuracy:  91.8500006198883
```

   Accuracy graph:



Loss graph:



Part 4

Optimizing NN

1. Initial model from step 1 uses:

Activation function: relu

Kernel size : 3x3

Optimizer: Adam

Table 1:

Hyperparameter	Setup 1	Accuracy	Setup 2	Accuracy	Setup 3	Accuracy
Activation function	selu	Training = 96.23% Testing =90.83%	elu	Training =96.36 % Testing = 91.06%	tanh	Training = 95.70% Testing = 91.28%
Optimizer	Adam		Adam		Adam	
Kernel Size	3x3		3x3		3x3	

Table 2:

Hyperparameter	Setup 4	Accuracy	Setup 5	Accuracy	Setup 6	Accuracy
Activation function	relu	Training = 96.13% Testing = 92.00%	relu	Training =95.77 % Testing = 91.36%	relu	Training = 95.27% Testing = 91.21%
Optimizer	adam		adam		adam	
Kernel Size	5x5		7x7		9x9	

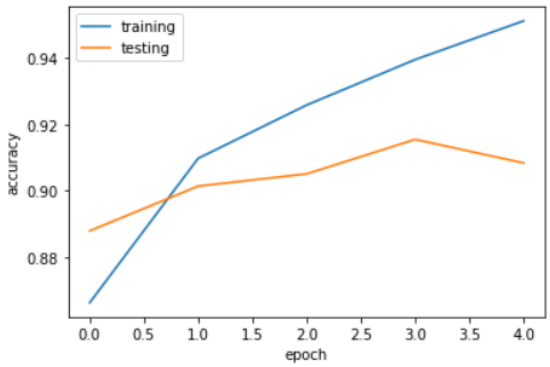
Table 3:

Hyperparameter	Setup 7	Accuracy	Setup 8	Accuracy	Setup 9	Accuracy
Activation function	relu	Training = 93.91% Testing = 91.51%	relu	Training =71.86 % Testing = 70.92%	relu	Training = 10% Testing = 10%
Optimizer	Adamax		adadelta		ftrl	
Kernel Size	5x5		5x5		5x5	

2. Setup 1:

```
In [31]: accuracy(model1,hist1)
```

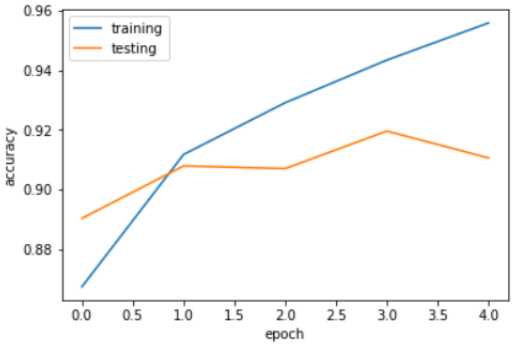
```
1875/1875 [==============================] - 26s 14ms/step - loss: 0.1065 - accuracy: 0.9624
Training accuracy:  96.23666405677795
313/313 [==============================] - 4s 14ms/step - loss: 0.2917 - accuracy: 0.9084
Testing accuracy:  90.83999991416931
```



Setup 2:

```
In [34]: accuracy(model2,hist2)
```

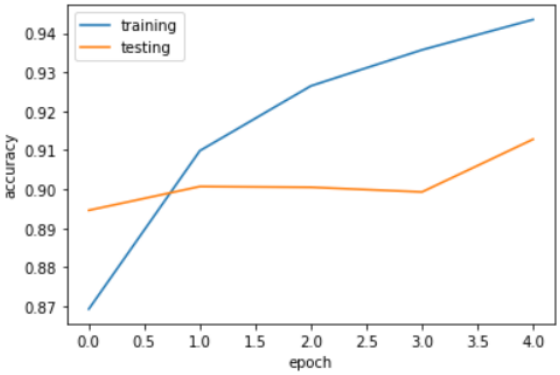
```
1875/1875 [==============================] - 32s 17ms/step - loss: 0.0956 - accuracy: 0.9637
Training accuracy:  96.36666774749756
313/313 [==============================] - 6s 18ms/step - loss: 0.2847 - accuracy: 0.9106
Testing accuracy:  91.06000065803528
```



Setup 3:

```
In [36]: accuracy(model3,hist3)
```

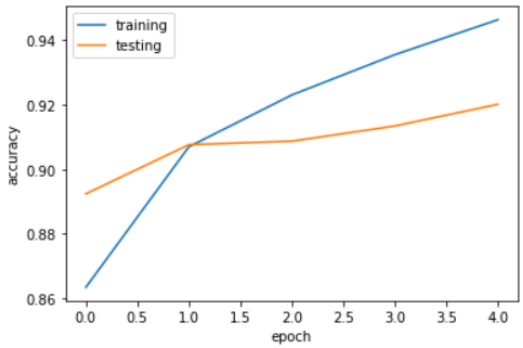
```
1875/1875 [==============================] - 34s 18ms/step - loss: 0.1213 - accuracy: 0.9571
Training accuracy:  95.70833444595337
313/313 [==============================] - 5s 16ms/step - loss: 0.2503 - accuracy: 0.9128
Testing accuracy:  91.28000140190125
```



Setup 4:

```
In [39]: accuracy(model4,hist4)
```

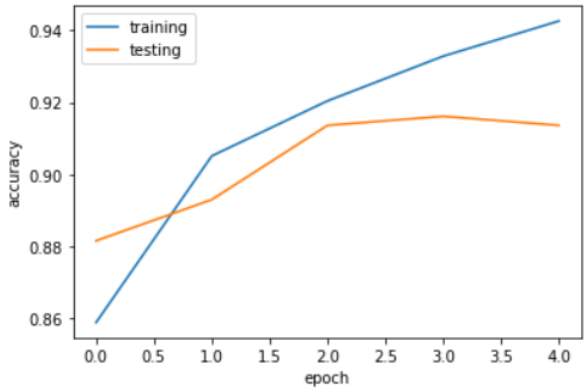
```
1875/1875 [==============================] - 43s 23ms/step - loss: 0.1071 - accuracy: 0.9614
Training accuracy:  96.13500237464905
313/313 [==============================] - 7s 23ms/step - loss: 0.2364 - accuracy: 0.9201
Testing accuracy:  92.00999736785889
```



Setup 5:

```
In [41]: accuracy(model5,hist5)
```

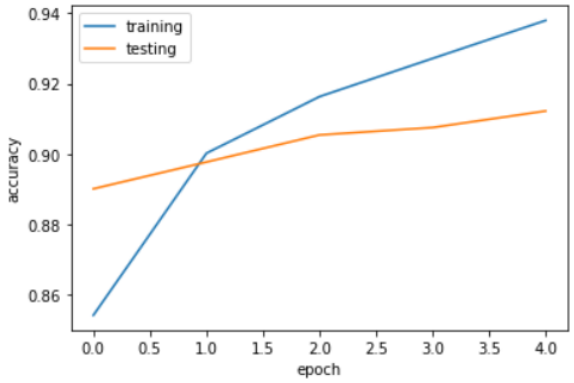
```
1875/1875 [==============================] - 72s 38ms/step - loss: 0.1195 - accuracy: 0.9578
Training accuracy:  95.77500224113464
313/313 [==============================] - 12s 39ms/step - loss: 0.2375 - accuracy: 0.9136
Testing accuracy:  91.36000275611877
```



Setup 6:

```
In [43]: accuracy(model6,hist6)
```

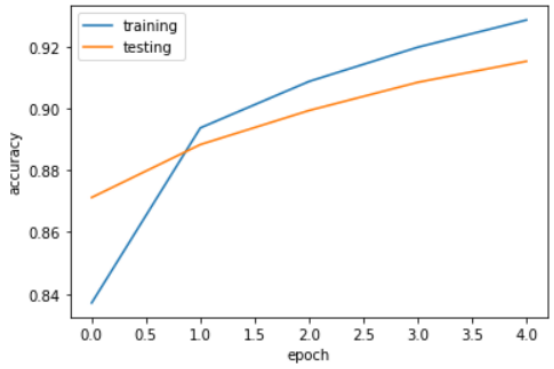
```
1875/1875 [==============================] - 100s 53ms/step - loss: 0.1240 - accuracy: 0.9527
Training accuracy:  95.27166485786438
313/313 [==============================] - 17s 55ms/step - loss: 0.2563 - accuracy: 0.9122
Testing accuracy:  91.21999740600586
```



Setup 7:

```
In [45]: accuracy(model7,hist7)
```

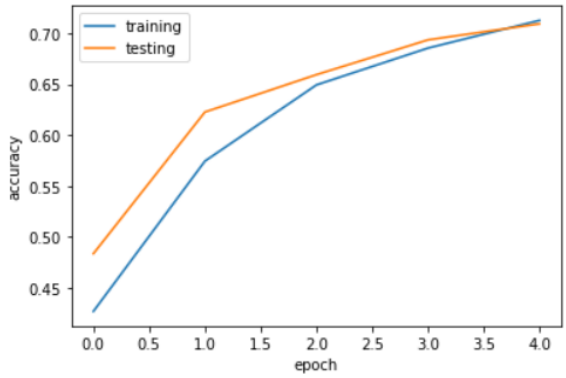
```
1875/1875 [==============================] - 40s 21ms/step - loss: 0.1683 - accuracy: 0.9391
Training accuracy:  93.91166567802429
313/313 [==============================] - 7s 23ms/step - loss: 0.2382 - accuracy: 0.9152
Testing accuracy:  91.51999950408936
```



Setup 8:

```
In [47]: accuracy(model8,hist8)
```

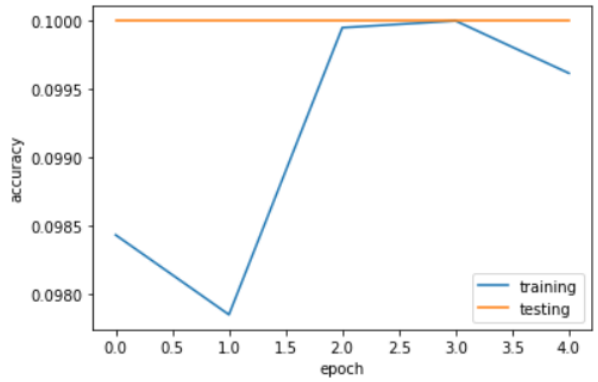
```
1875/1875 [==============================] - 41s 22ms/step - loss: 0.7974 - accuracy: 0.7186
Training accuracy:  71.86333537101746
313/313 [==============================] - 7s 22ms/step - loss: 0.8127 - accuracy: 0.7093
Testing accuracy:  70.92999815940857
```



Setup 9:

```
In [49]: accuracy(model9,hist9)
```

```
1875/1875 [==============================] - 43s 23ms/step - loss: 2.3026 - accuracy: 0.1000
Training accuracy:  10.000000149011612
313/313 [==============================] - 7s 22ms/step - loss: 2.3026 - accuracy: 0.1000
Testing accuracy:  10.000000149011612
```



3. Table 1: In this table, we have changed the activation function for the Neural Network setups. We used selu, elu and tanh. tanh gave the highest testing accuracy (91.28%) in this run.
   Table 2: In this table, we have changed the kernel size for the Neural Network setups. We used 5x5, 7x7 and 9x9. 5x5 gave the highest testing accuracy (92.00%) in this run.
   Table 3: In this table, we have changed the optmizer for the Neural Network setups. We used adamax, adadelta and ftrl. adamax gave the highest testing accuracy (91.51%) in this run.

4. We used Random Flip, Random Rotation, Random Zoom and Random Translation for data augmentation for our model. We included it as a layer in our model in the beginning.