SJSU SAN JOSÉ STATE UNIVERSITY

# Fashion MNIST Image Classification using Neural Network

Neural Network Project Report

By Kashyap Mehta (ID: 014489891)
Project Partner (Mukund Vyas)

## Table of Contents

**1. Problem Statement** .......... 3

**2. Introduction** .......... 3

**3. Import Data** .......... 4

**4. Data Visualization** .......... 4

**5. Feature Scaling or Standardization** .......... 5

**6. Building Model** .......... 6

**6.1. Complexity** .......... 7

**6.2. Activation Function** .......... 13

**6.3. Complier** .......... 16

**7. Model Training** .......... 18

**8. Cross-Validation** .......... 19

**9. Performance Metrics** .......... 19

**10. Conclusion** .......... 21

## Figure of Contents

Figure 1: Data Visualization .......... 4
Figure 2: Output Data Distribution .......... 4
Figure 3: Scaled vs Unscaled Image Classification .......... 5
Figure 4: Training Accuracy with increase in No. of Neurons in Hidden Layer .......... 8
Figure 5: Training & Test Accuracy with increase in Number of Neurons in Hidden Layer .......... 8
Figure 6: Training Accuracy with change in no. of Hidden Layers .......... 10
Figure 7: Train & Test Accuracy with change in no. of Hidden Layers .......... 10
Figure 8:Training Accuracy with Change in Batch Size & No. of Epochs .......... 12
Figure 9:Accuracy vs Batch Size .......... 12
Figure 10: Activation Function .......... 14
Figure 11: Accuracy vs Activation Function of Hidden Layers .......... 14
Figure 12:Accuracy vs Activation Function in Output Layers .......... 15
Figure 13: Accuracy vs Change in Optimizer .......... 16
Figure 14: Accuracy vs Change in Metrics .......... 17
Figure 15: Training Accuracy & Loss Function .......... 18
Figure 16: Cross Validation .......... 19
Figure 17: Recall & Precision Graph .......... 20

## 1. Problem Statement

Implement neural networks to classify the 10 classes in the FASHION MNIST dataset namely T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle Boot.

## 2. Introduction

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. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 785 columns. The first column consists of the class labels (see above), and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.

**Features:**

28 x 28 pixels

**Labels:**

The output 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 |

## 3. Import Data

Fashion MNIST dataset requires to download the data from the server or it could also be loaded using pandas. For this project the data are imported directly from the server using keras library.

## 4. Data Visualization

The feature 28x28 pixel if represented in graph than it is represented as below:

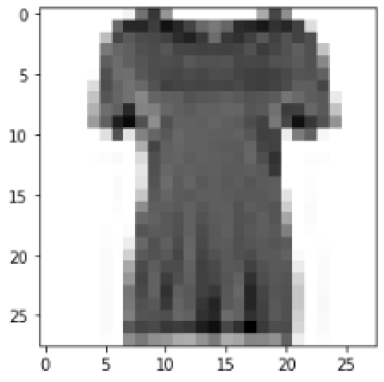


*Figure 1: Data Visualization*

The total number of training datasets are 60000 while test datasets are 10000. The input feature is reshaped to 28x28 = 784.

The distribution of output for training & test data is represented as below. It could be depicted from the graph that the fashion MNIST dataset is equally distributed among the output.

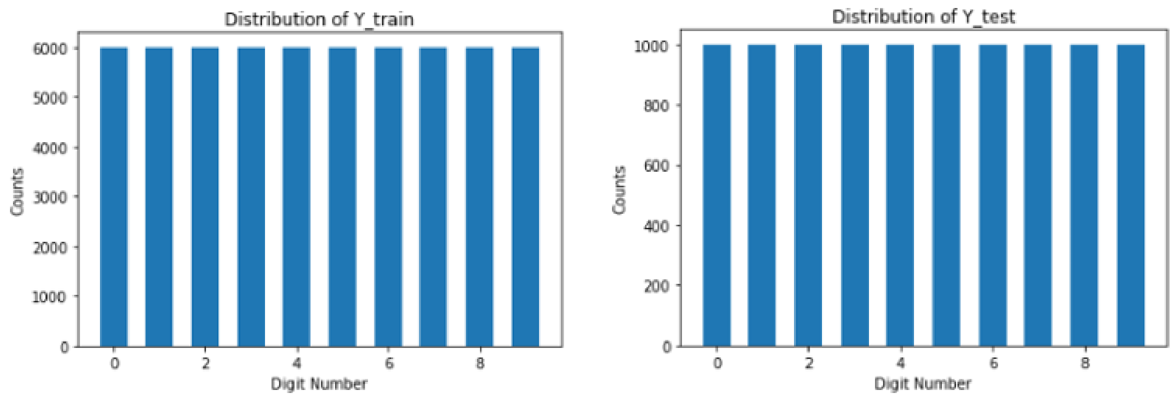


*Figure 2: Output Data Distribution*

## 5. Feature Scaling or Standardization

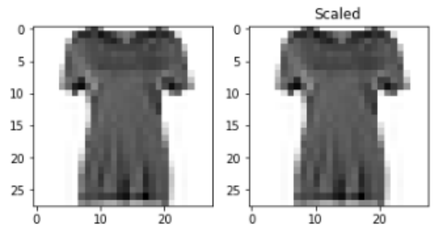
It is a step of Data Pre-Processing which is applied to independent variables or features of data. It basically helps to normalize the data within a particular range. It also helps in speeding up the calculations in an algorithm as if the scaling is not done than it might require large amount of iterations to converge to calculate the weights.

Scaling can be done in two ways:

i. **Min Max Scaling**: It converts the given features into the scale of 0 to 1. The standard formula to convert min max scaling is given below:

$$X_{new} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$

If given features are applied for min-max scaling than we cannot identify the scaling in images as the imshow function scales the input from 0-1 to 0-255 for representation of image. Hence we use Standardized Scaling method to scale.



ii. **Standardized Scaling:** It is a very effective technique which re-scales a feature value so that it has distribution with 0 mean value and variance equals to 1. The standard equation to scale the input is as given below:

$$X_{new} = \frac{X_i - X_{mean}}{\text{Standard Deviation}}$$

The image shown below represents the unscaled input and scaled input for each image classified.

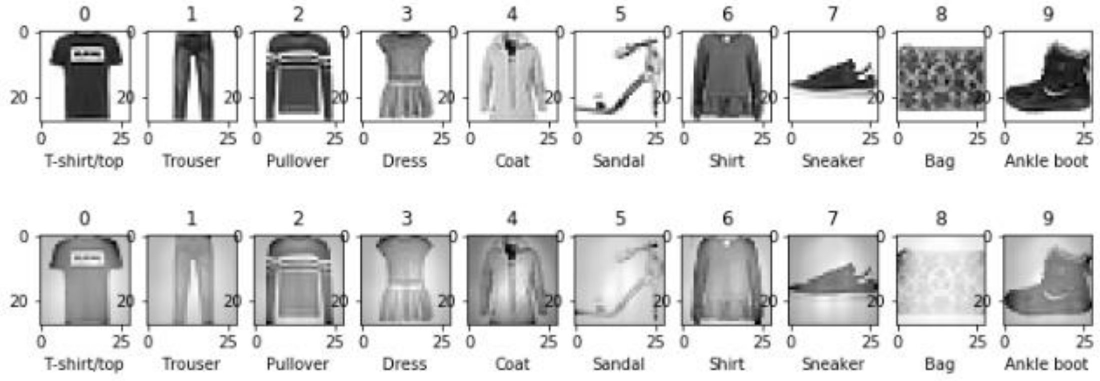


*Figure 3: Scaled vs Unscaled Image Classification*

## 6. Building Model

To develop a model following things shall be considered for creating a fully connected feedforward neural network:

1. Number of Neurons
2. Number of Hidden Layers
3. Batch Size
4. Number of Epochs
5. Activation Function of Hidden Layer
6. Activation Function of Output Layer
7. Compiler

To get the best test & train accuracy following are the parameters which should be considered to design a model.

1. Complexity
    a. Increase in No. of Neurons
    b. Increase in No. of Hidden Layers
    c. Increase in batch size & epochs
2. Activation Function
    a. Hidden Layer
    b. Output Layer
3. Compiler
    a. Optimizer
    b. Metrics

## 6.1. Complexity

### i. Training Accuracy with change in no. of Neurons

**Variable:**

- No. of neurons in hidden layer:
    a. 1 Neuron in Hidden Layer
    b. 60 Neurons in Hidden Layer
    c. 200 Neurons in Hidden Layer
    d. 3000 Neurons in Hidden Layer

**Constants:**

- Hidden Layer Activation Function: RELU
- Output Layer Activation Function: Sigmoid
- Hidden Layer: 1
- Batch Size: 1024
- Epochs: 5

**Observations:**

- Figure 1 represents training accuracy with increase in epoch for different number of neurons in hidden layer.
- With increase in number of neurons in the hidden layer, the complexity of the model increases and hence the training accuracy increases.
- Figure 2 represents the accuracy of training & test data for different number of neurons in hidden layer.
- For 1 neuron in hidden layer the accuracy of training & test data is very poor. For 60, 200 & 3000 neurons in hidden layer the accuracy is almost same.
- It could be depicted that with increase in number of neurons the accuracy increases but after particular number of neurons in hidden layer, the accuracy does not increase with increase in number of neurons.
- As the training & test accuracy is unable to increase beyond 60 neurons, hence 60 neurons in hidden layer is selected.

**Output:**

- 60 Neurons selected in hidden layer

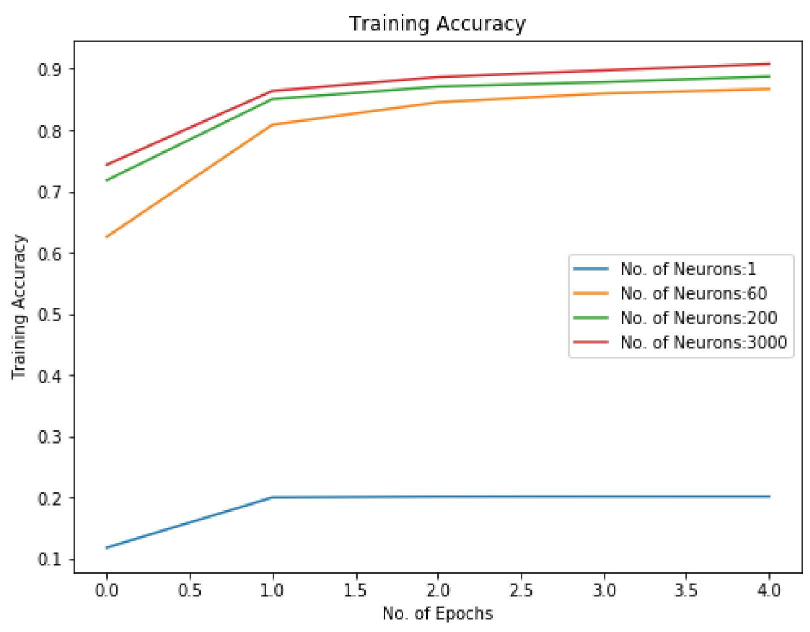


*Figure 4: Training Accuracy with increase in No. of Neurons in Hidden Layer*

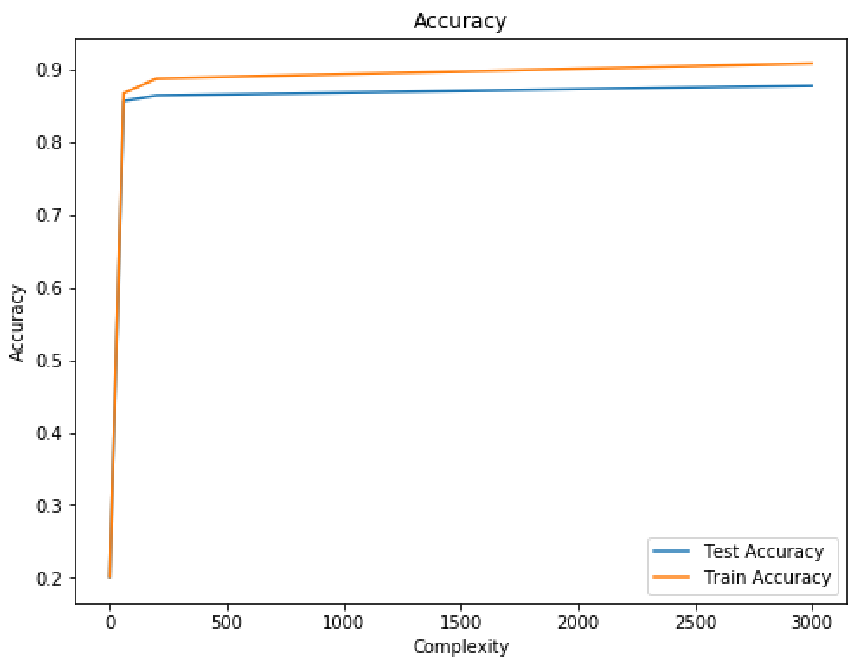


*Figure 5: Training & Test Accuracy with increase in Number of Neurons in Hidden Layer*

## ii. Increase in Number of Hidden Layer

**Variable:**

- No. of hidden layer:
    a. 1 Hidden Layer
    b. 2 Hidden Layers
    c. 5 Hidden Layers

**Constants:**

- Number of Neurons: 60
- Hidden Layer Activation Function: RELU
- Output Layer Activation Function: Sigmoid
- Batch Size: 128
- Epochs: 5

**Observations:**

- Figure 3 represents training accuracy with increase in epoch for different number of hidden layers.
- For single hidden layer the training accuracy is highest, while with increase in hidden layer the training accuracy decreases considering same number of epochs. Ideally the training accuracy shall increase with increase in complexity of the network but as the number of epochs is constant hence the training accuracy is higher for less hidden layer.
- Figure 4 represents the training & test accuracies with change in number of hidden layers. The graph represents the accuracy for 1, 2 & 5 hidden layers.
- As shown in figure, the test accuracy is highest for 2 hidden layer and it decreases afterwards as the model is overfitted with the complexity.

**Output:**

- 2 Hidden Layer is selected as the test accuracy is maximum

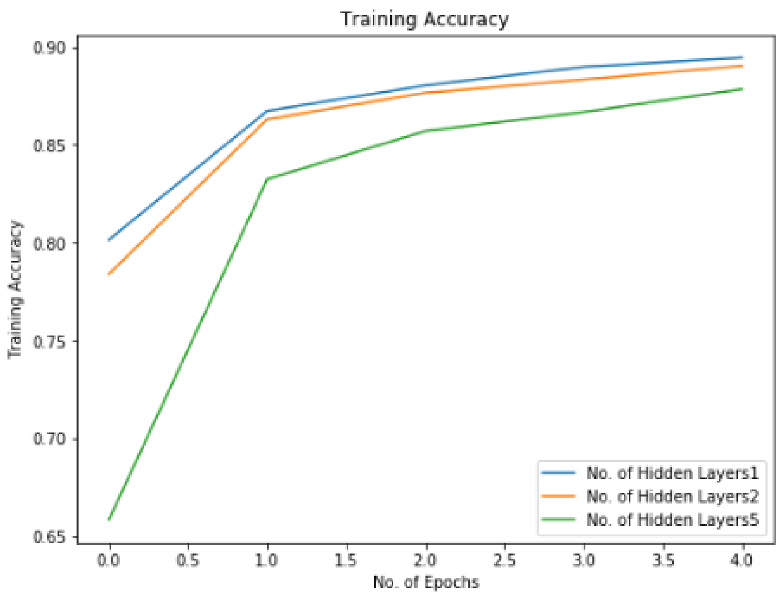


*Figure 6: Training Accuracy with change in no. of Hidden Layers*

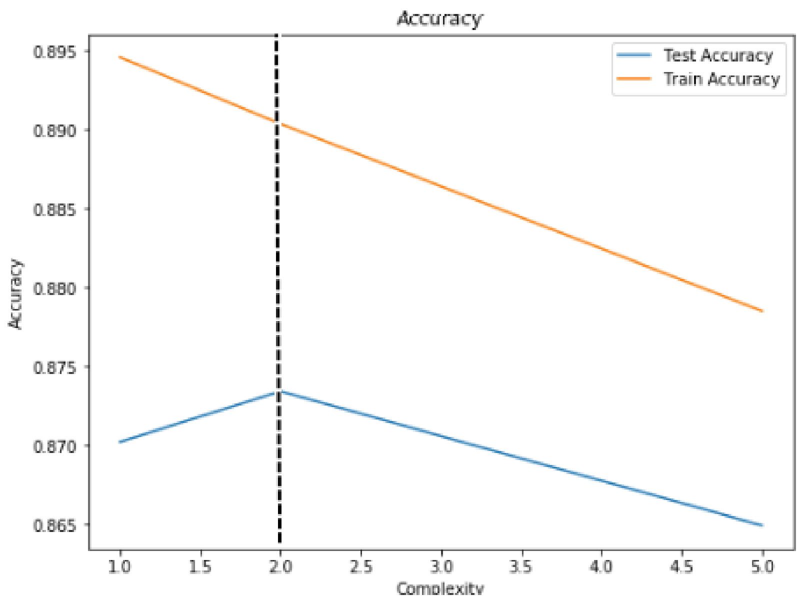


*Figure 7: Train & Test Accuracy with change in no. of Hidden Layers*

## iii. Change in Batch Size & Epochs

### Variable:

- No. of Batch Size & No. of Epochs:
    a. 64 Batch Size for 5 Epochs
    b. 128 Batch Size for 8 Epochs
    c. 1024 Batch Size for 10 Epochs
    d. 4096 Batch Size for 12 Epochs

### Constants:

- Number of Neurons: 60
- Hidden Layer Activation Function: RELU
- Output Layer Activation Function: Sigmoid
- Hidden Layer: 1

### Observations:

- Figure 5 represents training accuracy with increase in batch size for different epochs used to get the output.
- For large batch size the training accuracy is less for very first iteration. As the batch size is large, it requires less time to converge and hence we have to increase the number of epochs.
- With increase in batch size the training accuracy decreases as well as it requires a greater number of epochs to converge. The training accuracy is highest for 128 Batch size for 8 Epochs.
- Figure 6 represents training & test accuracy with respect to Batch Size. The accuracy is increases gradually for Batch size 64 to 128 than it decreases with increase in Batch.
- Hence, we could conclude that the best Batch size could be 128 with Epochs 8.

### Output:

- Batch Size & No. of Epochs is selected as 128 & 8 respectively for the model.

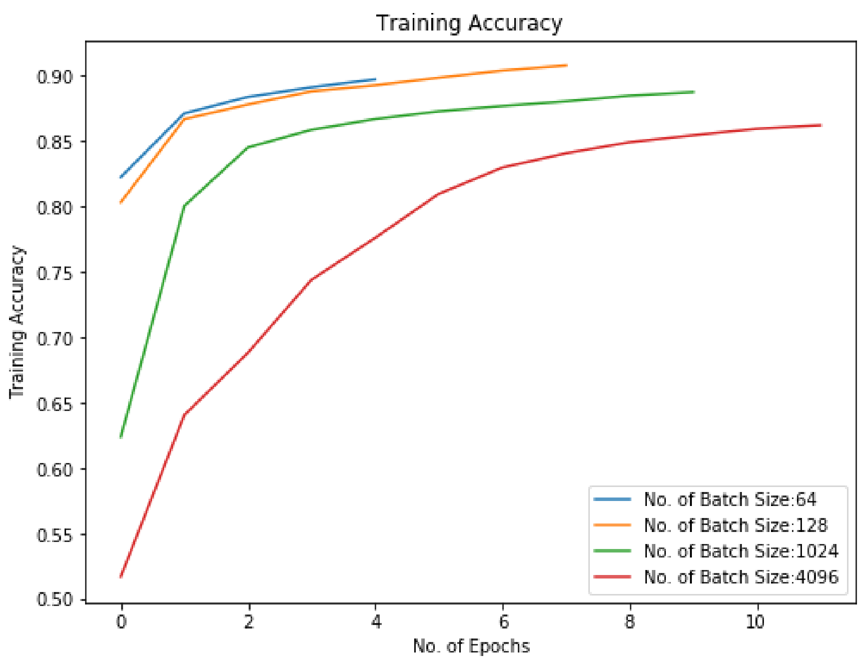


*Figure 8:Training Accuracy with Change in Batch Size & No. of Epochs*

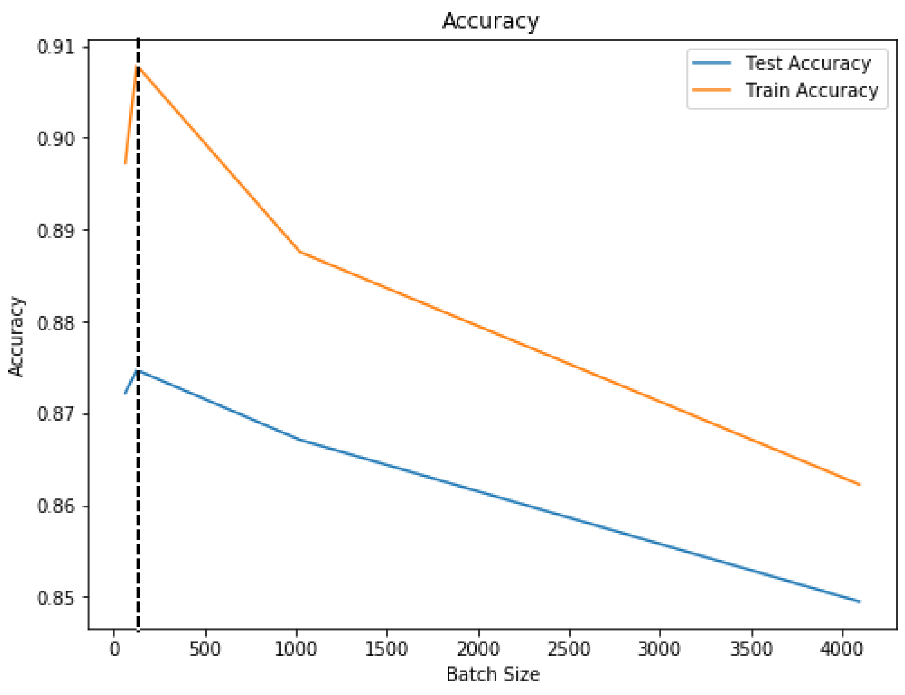


*Figure 9:Accuracy vs Batch Size*

## 6.2.Activation Function

### i. Change in Activation Function of Hidden Layer

**Variable:**

- Activation Function of Hidden Layer:
    a. Linear
    b. tanh
    c. exponential
    d. Hard sigmoid
    e. Sigmoid
    f. RELU
    g. Soft sign
    h. Soft plus

**Constants:**

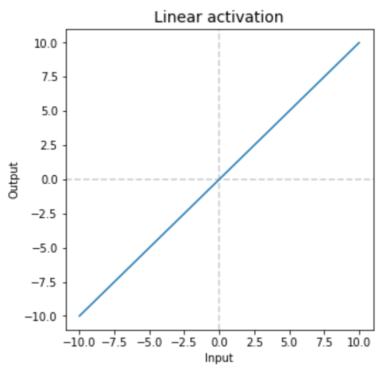
- Number of Neurons: 30 in each hidden layer
- Number of Hidden Layer: 3
- Batch Size: 128
- Epochs: 5
- Output Layer Activation Function: Sigmoid

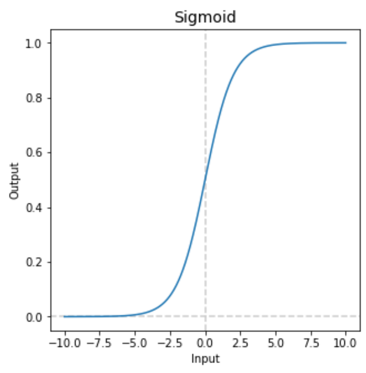
**Observations:**

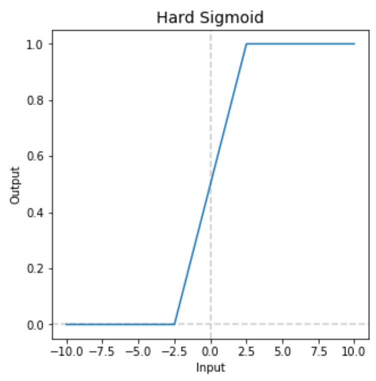
- Each Activation Function response is represented in figure as below
- Figure 7 represents the accuracy of training & test data with respect to change in activation function of hidden layer.
- It could be concluded that the accuracy is highest for linear, RELU & soft plus activation function.

**Output:**

- RELU is selected as hidden layer activation function.







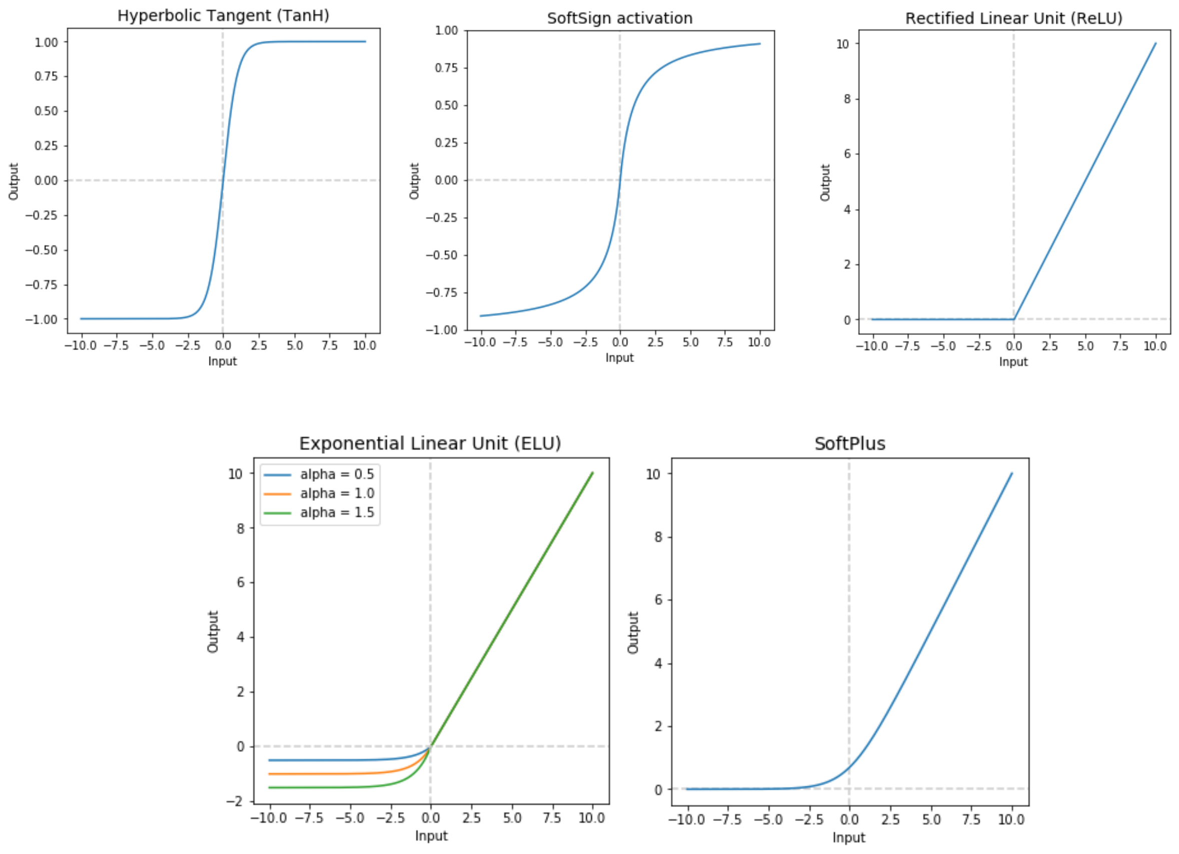


*Figure 10: Activation Function*

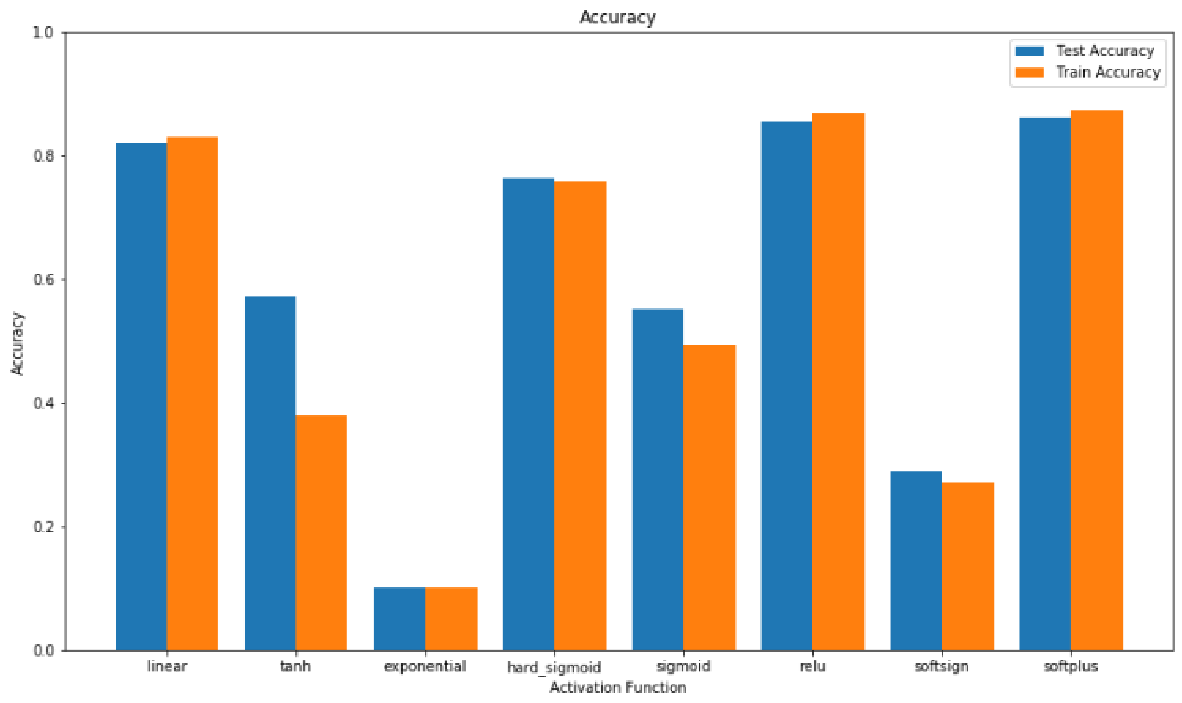


*Figure 11: Accuracy vs Activation Function of Hidden Layers*

ii. <u>Change in Activation Function of Output Layer</u>

<u>Variable:</u>

- Activation Function of Output Layer:
    a. Linear
    b. tanh
    c. Hard sigmoid
    d. Sigmoid
    e. Soft sign
    f. Soft plus
    g. Soft max

<u>Constants:</u>

- Number of Neurons: 60
- Number of Hidden Layer: 1
- Batch Size: 128
- Epochs: 5
- Activation Function of Hidden Layer: RELU

<u>Observations:</u>

- Figure 8 represents the train & test accuracy with change in output activation function.
- The graph represents that tanh, hard sigmoid & soft sign are having poor impact on the output and have poor accuracy while sigmoid, softplus & softmax have the best accuracy

<u>Output:</u>

- Sigmoid is selected as the hidden layer of output layer.

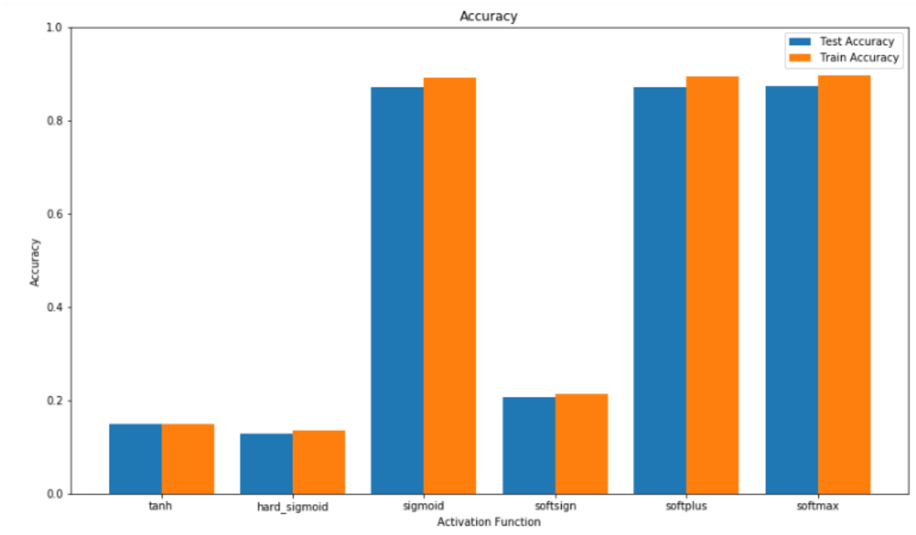


*Figure 12:Accuracy vs Activation Function in Output Layers*

## 6.3.Complier

The Complier consists of following major design parameters namely Loss function, Optimizer & Metrics.

- Loss function — These measures how accurate the model is during training. You want to minimize this function to "steer" the model in the right direction.
- Optimizer —This is how the model is updated based on the data it sees and its loss function.
- Metrics —Used to monitor the training and testing steps.

i. <u>Chane in Optimizer</u>

**Variable:**

- Optimizer:
  a. Stochastic Gradient Descent (SGD)
  b. RMSprop
  c. Adagrad
  d. Adadelta
  e. Adam
  f. Adamax
  g. Nadam

**Constants:**

- Number of Neurons: 60
- Number of Hidden Layer: 1
- Batch Size: 128
- Epochs: 5
- Activation Function of Hidden Layer: RELU
- Activation Function of Output Layer: Sigmoid

**Observations:**

- Figure 9 represents the train & test accuracy with change in optimizer.
- The graph represents that stochastic gradient descent is having less accuracy considering same number of epochs and hidden layers.

**Output:**

- Nadam is selected as the optimizer as it provides the best accuracy.

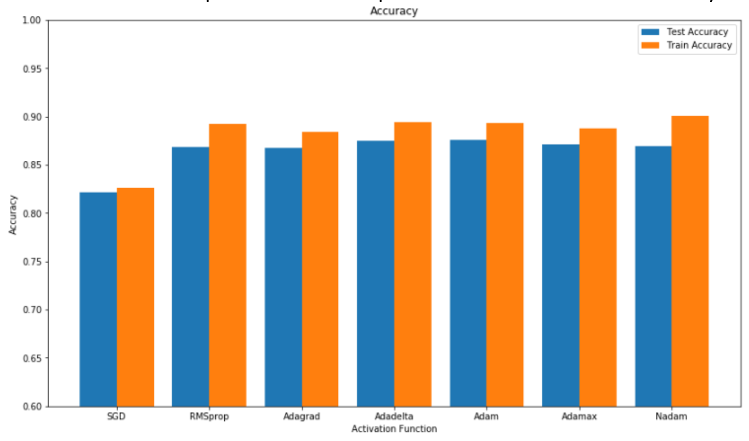


*Figure 13: Accuracy vs Change in Optimizer*

## ii. Chane in Metrics

**Variable:**

- Metrics:
    a. Binary Accuracy
    b. Accuracy
    c. Categorical Accuracy

**Constants:**

- Number of Neurons: 60
- Number of Hidden Layer: 1
- Batch Size: 128
- Epochs: 5
- Activation Function of Hidden Layer: RELU
- Activation Function of Output Layer: Sigmoid

**Observations:**

- Figure 10 represents the train & test accuracy with change in metrics.
- The graph represents that test accuracy is highest for binary accuracy than others.

**Output:**

- Accuracy is selected as the metrics.

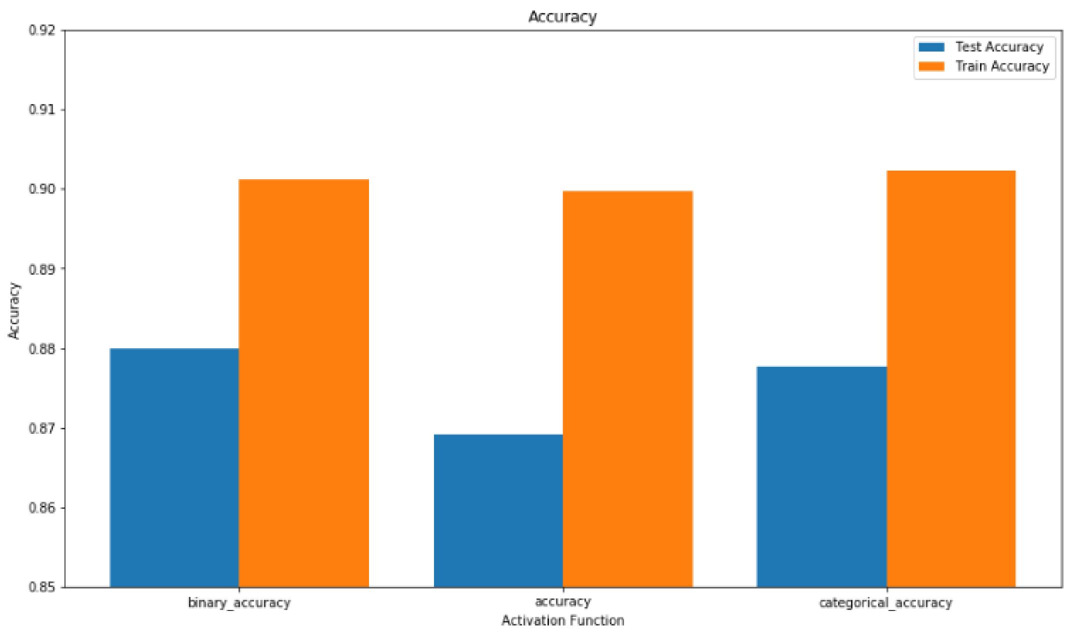


*Figure 14: Accuracy vs Change in Metrics*

## 7. Model Training

<u>Parameters:</u>

- _Number of Neurons: 60
- Number of Hidden Layer: 2
- Batch Size: 128
- Epochs: 8
- Activation Function of Hidden Layer: RELU
- Activation Function of Output Layer: Sigmoid
- Optimizer: Nadam
- Loss: Sparse Cross Entropy
- Metrics: Accuracy

After selecting design parameters for the model and iterating the model for multiple times the above parameters are fixed. The graph below represents the training accuracy and loss function with increase in number of epochs. As the number of epochs increases the training accuracy will increase as the model will try to overfit with reduction in loss function.

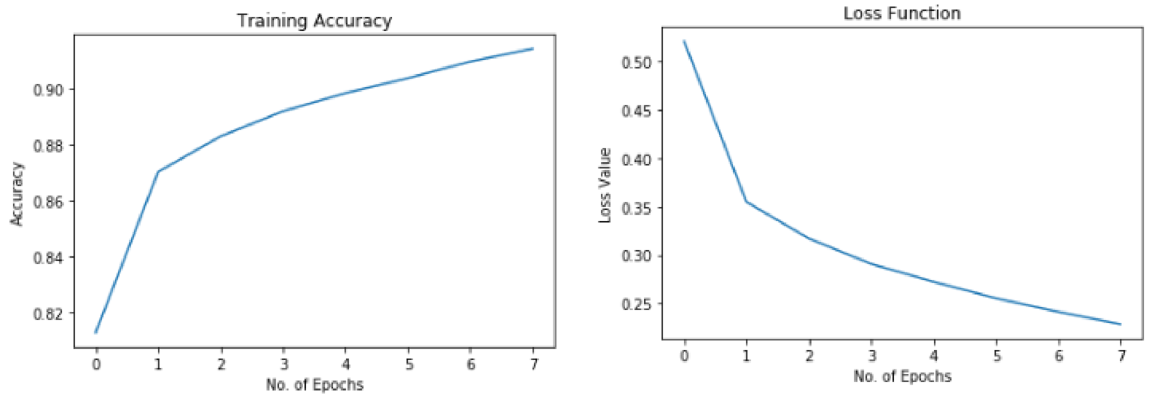


*Figure 15: Training Accuracy & Loss Function*

Now the model is trained and require to apply test data as features but before that the test features shall be scaled as scaling applied to train data. The test accuracy obtained for the model is **87.64%**.

## 8. Cross-Validation

As the input are already divided into test & train dataset hence we cannot use K-fold cross-validation. Normal method for cross-validation is to use the test data for validation or else divide the train data into validation data and test data and find out the accuracies for each.

The graph represents the accuracy of each output data after cross validation of the input. The graph represents that the accuracy for 10-fold cross validation of input. It could easily said that the accuracy does not change with the change in data slabs of input.

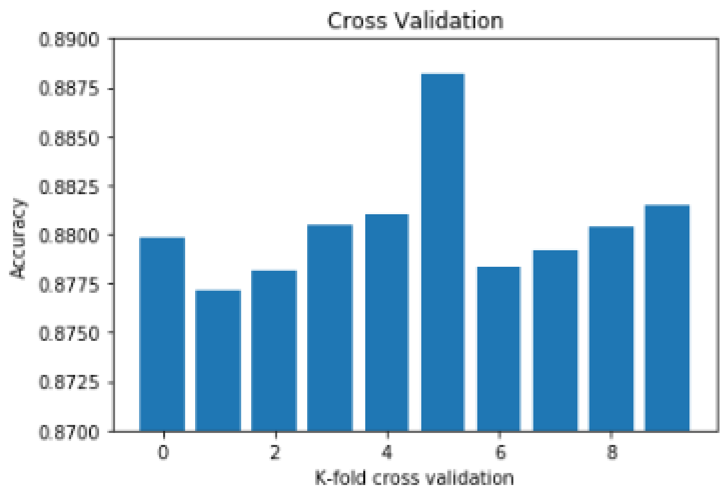
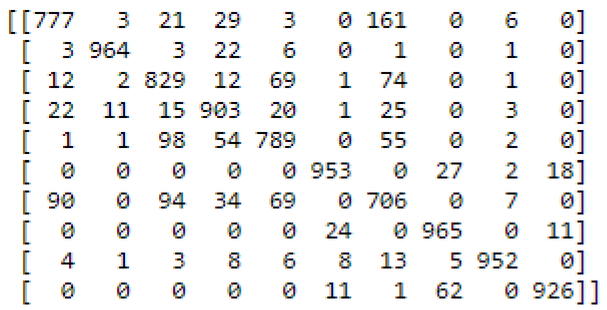


*Figure 16: Cross Validation*

## 9. Performance Metrics

To calculate the performance metrics of the model first we require confusion metrics to calculate. The table below shows the confusion metrics of the model:



```
[[777   3  21  29   3   0 161   0   6   0]
 [  3 964   3  22   6   0   1   0   1   0]
 [ 12   2 829  12  69   1  74   0   1   0]
 [ 22  11  15 903  20   1  25   0   3   0]
 [  1   1  98  54 789   0  55   0   2   0]
 [  0   0   0   0   0 953   0  27   2  18]
 [ 90   0  94  34  69   0 706   0   7   0]
 [  0   0   0   0   0  24   0 965   0  11]
 [  4   1   3   8   6   8  13   5 952   0]
 [  0   0   0   0   0  11   1  62   0 926]]
```

The model performance is measured by the following parameters:

i. **Accuracy:** The accuracy of the model is the sum of the diagonal of the confusion metrics. The Accuracy of the model is **87.6%.**

ii. **Precision:** The precision of each Output is represented in table as below:

```
Output   Precision (%)
0        85.0
1        98.0
2        78.0
3        85.0
4        82.0
5        95.0
6        68.0
7        91.0
8        98.0
9        97.0
```

iii. **Recall:** The Recall of each Output is represented in table as below:

```
Output   Recall (%)
0        78.0
1        96.0
2        83.0
3        90.0
4        79.0
5        95.0
6        71.0
7        96.0
8        95.0
9        93.0
```

The graph represents the Precision & Recall for each fashion output classification:

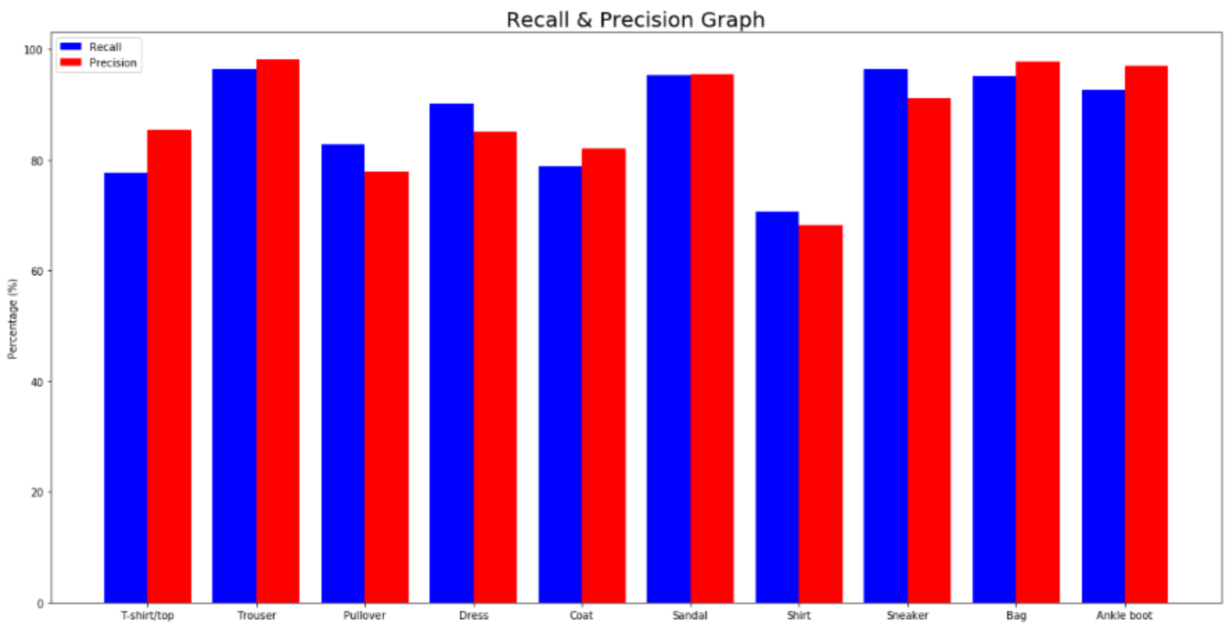


*Figure 17: Recall & Precision Graph*

## 10.Conclusion

In this project of classification of 28x28 pixel input features to fashion dataset as output using fully connected feedforward neural network, the image is classified with 88.7% test accuracy. The project uses keras and tensorflow library to design the neural network. The dataset is downloaded from the keras server which provides the test & train data separately. The dataset consists of 60000 training data and 10000 test data. The output consists of 10 classification of fashion items.

The model is designed for 2 hidden layers each consists of 60 neurons, the activation function of hidden layer is chosen as RELU and the activation function for output layer is Sigmoid. The batch size is taken as 128 with 8 epochs which provides the training accuracy as 91% and test accuracy as 89%.

The biggest challenge for the project was to experiment each activation function, number of hidden layers, number of neurons, epoch & batch size to get the best accuracy. This all are the design parameters of the model and requires to iterate the model to find the best.