# Exploring Neural Networks with Fashion MNIST Clothing Dataset

Manisha Biswas
mbiswas2@buffalo.edu

## Abstract

This paper presents an approach to the development and the simulation of a machine learning model to implement neural network and convolutional neural network for the task of classification. The dataset used to create the classifier is the Fashion-MNIST dataset of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000 examples.The first part of the project is dedicated to pre-process the data.To achieve this we examine the data, then the next section is important to understand what are the issues that will need to be processed while preparing the data to create the classifier. Then we partition the train dataset as: 80% for training, 20% of the training data for validation dataset.. The next step is to implement Single Layer Neural Network, Multi Layer Neural Network and Convolutional Neural Network (CNN) to compare and visualize the predictions. We also select a set of hyperparameters for the learning process and estimate the best model parameters for **Single Layer Neural Network** which gives us **69%** of accuracy, **Multi Layer Neural Network** which gives us an accuracy of **85%** and **Convolutional Neural Network** which gives an accuracy of **85%** .

## 1. Introduction

We live in the age of Instagram, YouTube, and Twitter. Images and video (a sequence of images) dominate the way millennials and other weirdos consume information.Having models that understand what images show can be crucial for understanding your emotional state,location, interests and social group.

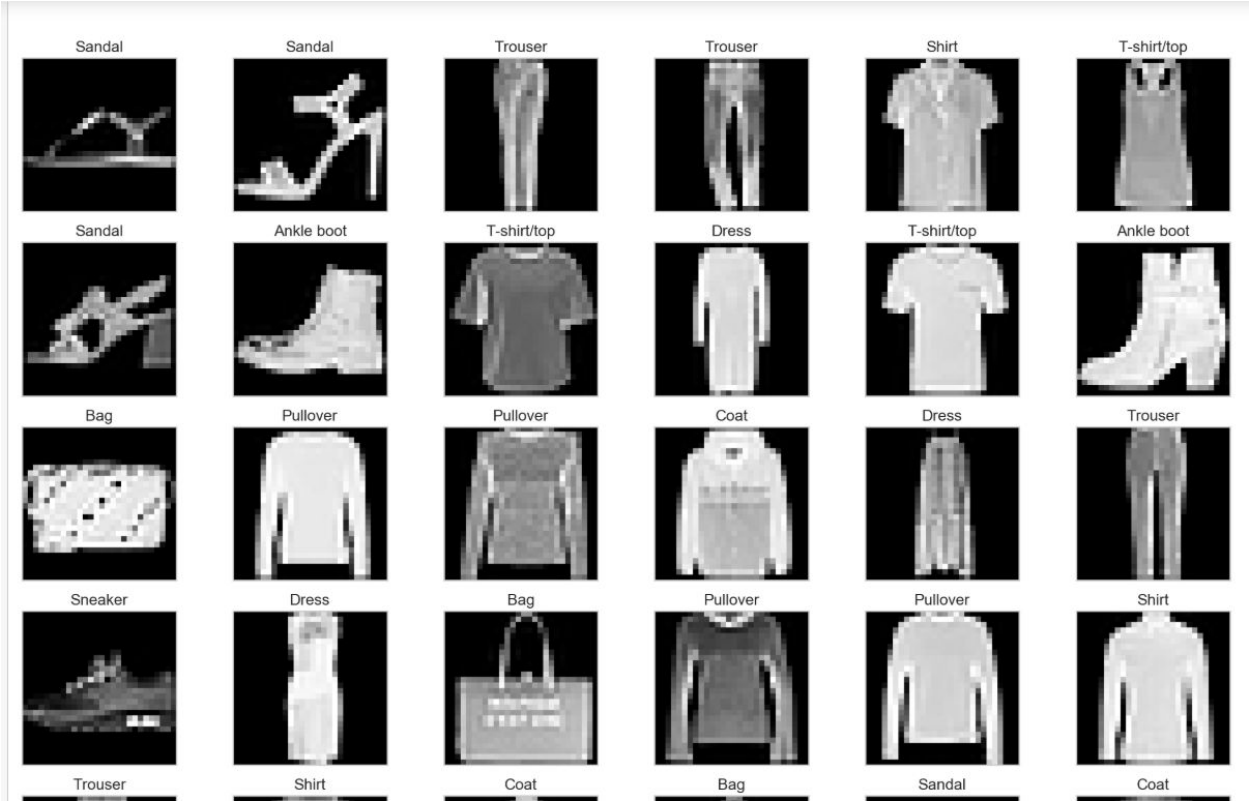
Images of clothes are great factors in driving the clothing industry today and trends found in the current fashion styles drive the online clothing economy. This study attempts to create a clothing classification tool with the use of neural networks.

This paper presents three different approaches, namely Neural Network with one hidden layer, multi-layer Neural Network, Convolutional Neural Network (CNN) for the task of classification. For training and testing of our classifiers, we will use the Fashion-MNIST dataset

## 2. Dataset Definition

The 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 image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels. 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. 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.




The training and test data sets have 785 columns. The first column consists of the class labels and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image.

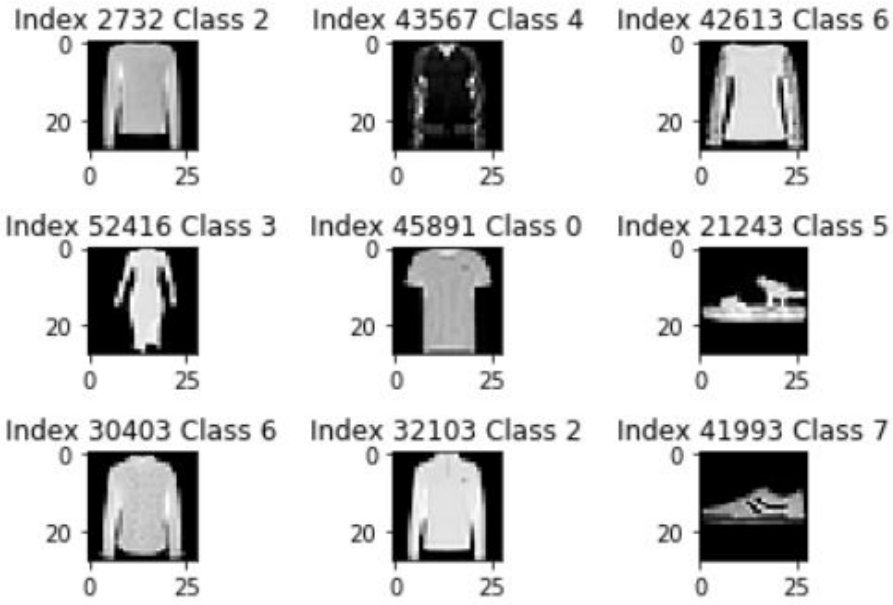
Each training and test example are assigned to one of the labels as shown in table 1.

| | |
|---|---|
| 1 | T-shirt/top |
| 2 | Trouser |
| 3 | Pullover |
| 4 | Dress |
| 5 | Coat |
| 6 | Sandal |
| 7 | Shirt |
| 8 | Sneaker |
| 9 | Bag |
| 10 | Ankle Boot |

Table 1: Labels for Fashion-MNIST dataset

## 3. Data Preprocessing

We are using Jupyter notebook to work on this dataset. We will first import the necessary packages and load the train set and test set which are given in two separate datasets. We can visualize the dataset by fetching randomly 4 examples from training data. Visualization of data is an imperative aspect of data science. It helps to understand data and also to explain the data to another person.




Now we will Normalize the data using min-max scalar and define the labels.

```
labels = [ "T-shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal","Shirt", "Sneaker", "Bag", "Ankle boot"]
```

We further split the train set in train and validation set. The validation set will be 20% from the original train set, therefore the split will be train/validation of 0.8/0.2.

- **Training data** — used for training the model
- **Validation data** — used for tuning the hyperparameters and evaluate the models
- **Test data** — used to test the model after the model has gone through initial vetting by the validation set.

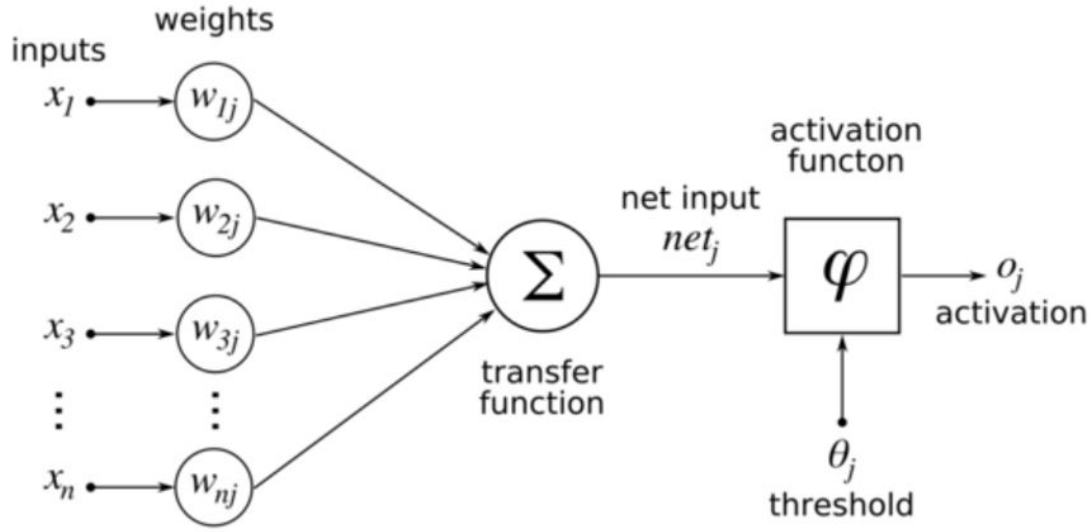
In [9]:
```
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
print(X_val.shape)
```

```
(48000, 784)
(48000,)
(10000, 784)
(10000,)
(12000, 784)
```

## 4. Model Architecture

- **Single Layer Neural Network:**

A single hidden layer neural network consists of 3 layers: input, hidden and output. The input layer has all the values form the input, in the hidden layer, where most of the calculations happens, every perceptron unit takes an input from the input layer, multiplies and add it to initially random values. The last and third layer is the output layer, it takes all the previous layer perceptrons as input and multiplies and add their outputs to initially random values. then gets activated by a Sigmoid Function.




To train the network we relies on gradient descent and backpropagation of the gradients. Here, the output values are compared with the correct answer to compute the value of the predefined error-function. The error compared to the expected final output is then fed back through the network. Using this information, the algorithm adjusts the initially random weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small. In this case, we would say that the network has learned a certain target function. To adjust weights properly, we applies nonlinear optimization that is called gradient descent. For this, the network calculates the derivative of the error function with respect to the network weights and changes the weights such that the error decreases (thus going downhill on the surface of the error function). For this reason, back-propagation can only be applied on networks with differentiable activation functions.

Mathematically, neural network has parameters (W, b) = ($W^{[1]}$, $b^{[1]}$, $W^{[2]}$, $b^{[2]}$), where W denotes the weight and b is the bias.

$Z^{[1]} = W^{[1]}x + b^{[1]}$

Activation function for the hidden layer:

$a^{[1]} = \sigma(Z^{[1]})$

$Z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$

Activation function for the output layer:

$a^{[2]} = \text{softmax}\ (Z^{[2]})$

Cost Function:

$L(a^{[2]}, y) = -\sum y \log a^{[2]}$

Derivatives of weights:

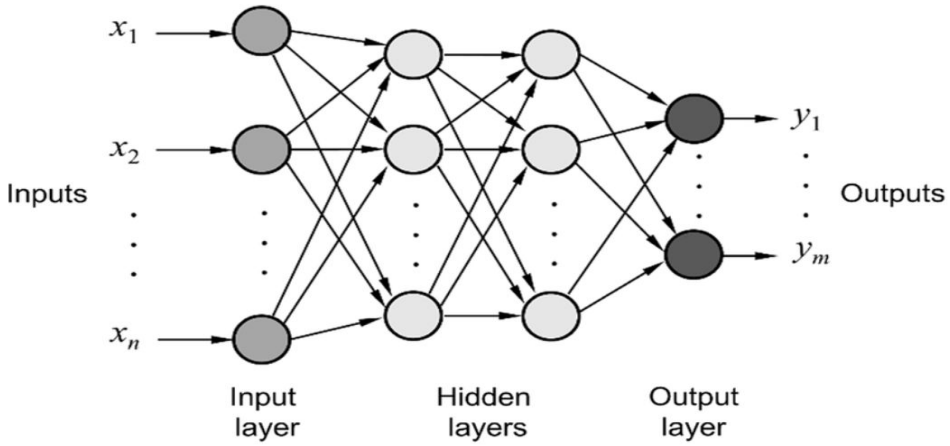
$$\Delta w^{[1]} = (a^{[2]} - y)\, a^{[1]}$$

$$\Delta w^{[1]} = (a^{[2]} - y) w^{[2]}\, a^{[1]}(1 - a^{[1]})\, x$$

In single layer neural network we have considered only one hyperparameter, i.e number of hidden nodes and we are tuning this model using the same in 4 stages using 10, 20, 30 and 40 hidden nodes.

- **Multi Layer Neural Network:**

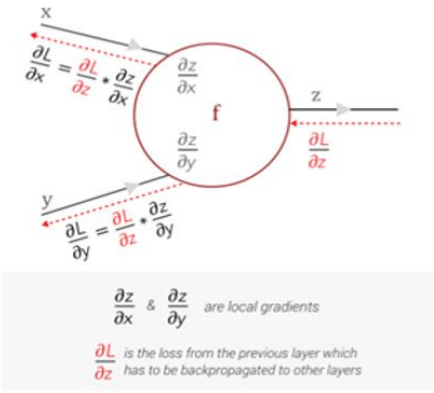
A multi-layer neural network contains more than one layer of artificial neurons or nodes. They differ widely in design. It is important to note that while single-layer neural networks were useful early in the evolution of AI, the vast majority of networks used today have a multi-layer model.




We will be using a Sequential model which is a linear stack of layers. It can be first initialized and then we add layers using add method.

- **Convolutional Neural Network:**

A Convolutional Neural Network is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms.




Computational Graph – Convolution Operation

We will be using a Sequential model which is a linear stack of layers. It can be first initialized and then we add layers using add method.

## 5. Results

The results obtained in this paper were very accurate with possibility of being better by usage of different techniques, methods of data pre-processing and including more data into the training. We discuss a few such methods below along with the results we obtained by applying them.

### 1. Single Layer Neural Network:

In Single layer neural network, we use tanh as the primary activation function in the single hidden layer and in the final output layer we use softmax activation function since we are doing multi-class classification. We are using vanilla gradient descent as our optimizer and categorical cross entropy as the loss function. .

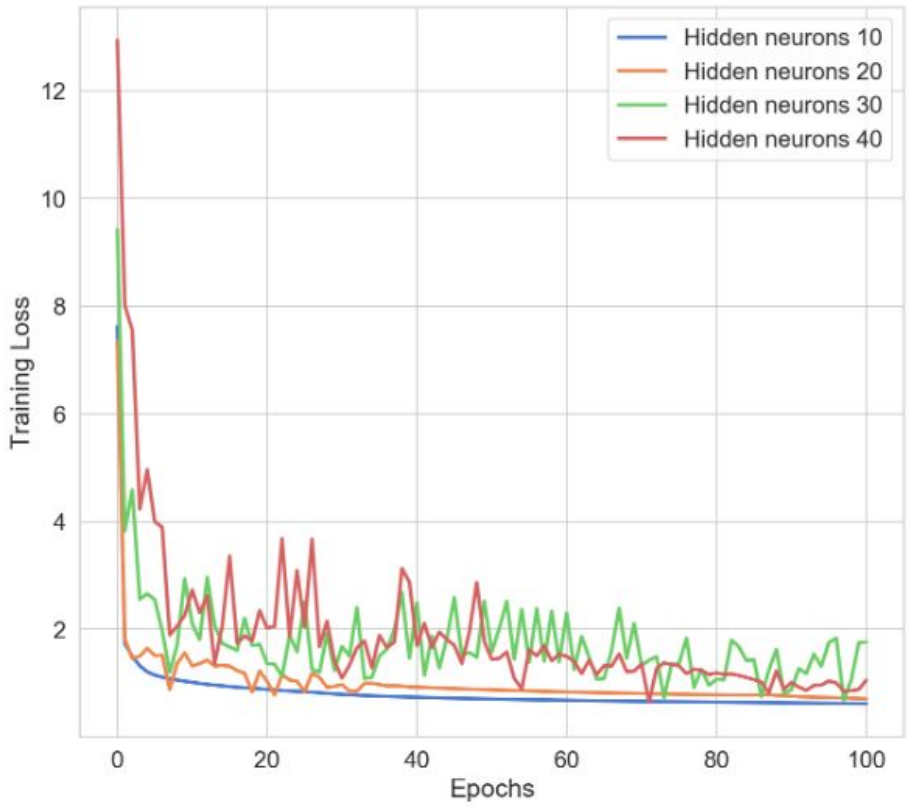
### Hyperparameter Tuning

***We consider one hyperparameter as number of hidden nodes in a single layer neural network with different values as 10, 20, 30, 40.***

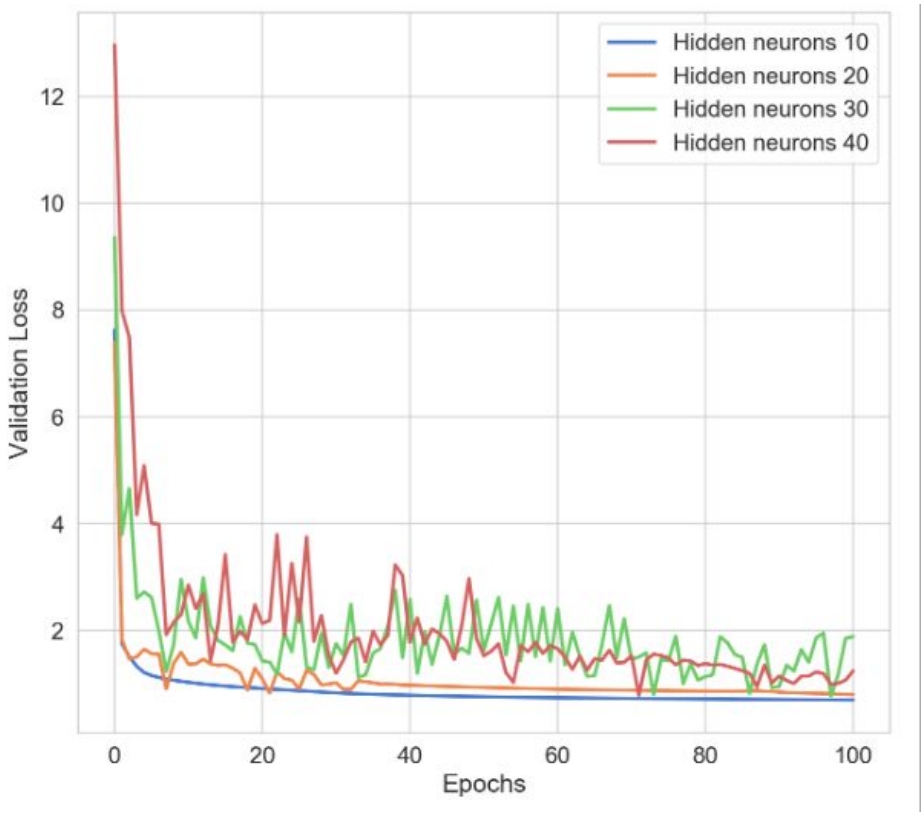
Apart from the above hyperparameters we have two other important hyperparameters that decide the learning curve of the model which is learning rate and number of epochs. In this case, our learning rate is set to 1e-4 and epochs set to 1000

We iteratively update the value of w1,b1,w2,b2 with respect to the Training and Validation data to estimate the best parameter for accuracy.

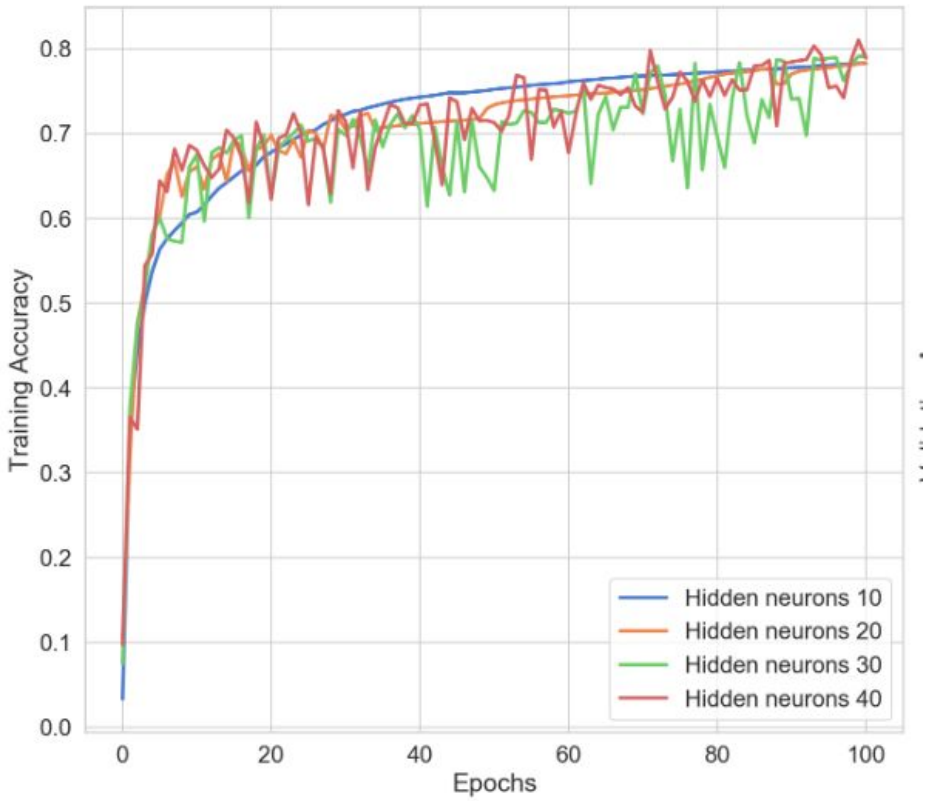
We record the Training loss with different hidden nodes as:



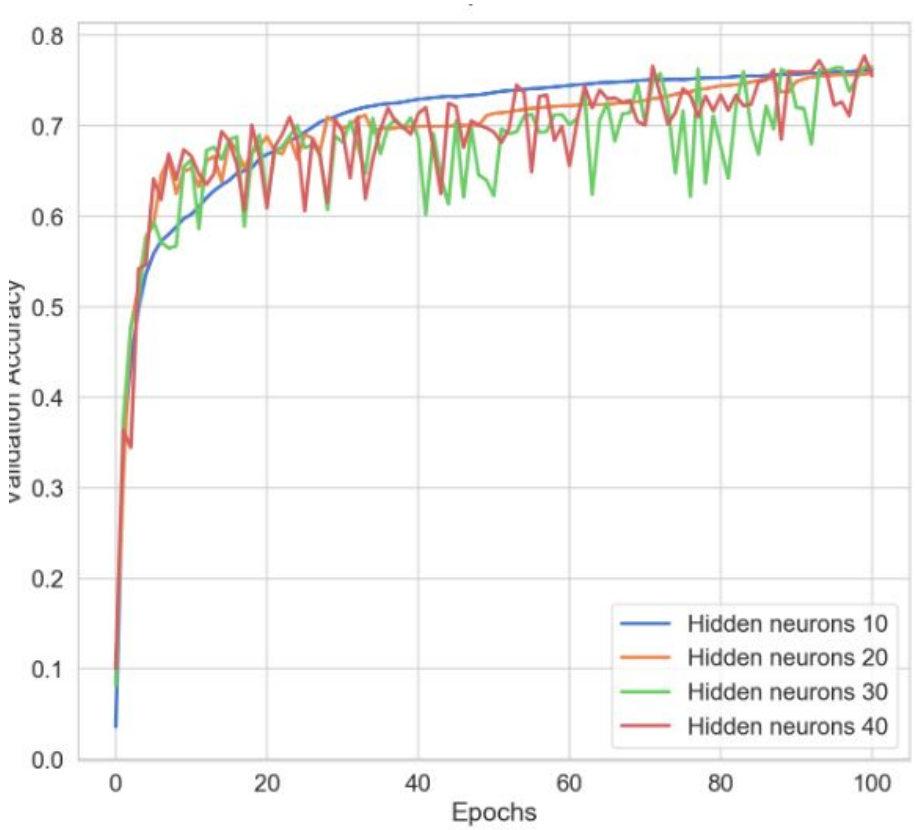

We record the Validation loss with different hidden nodes as:




We record the Training Accuracy with  hidden nodes as:




We record the Validation Accuracy with different nodes as:

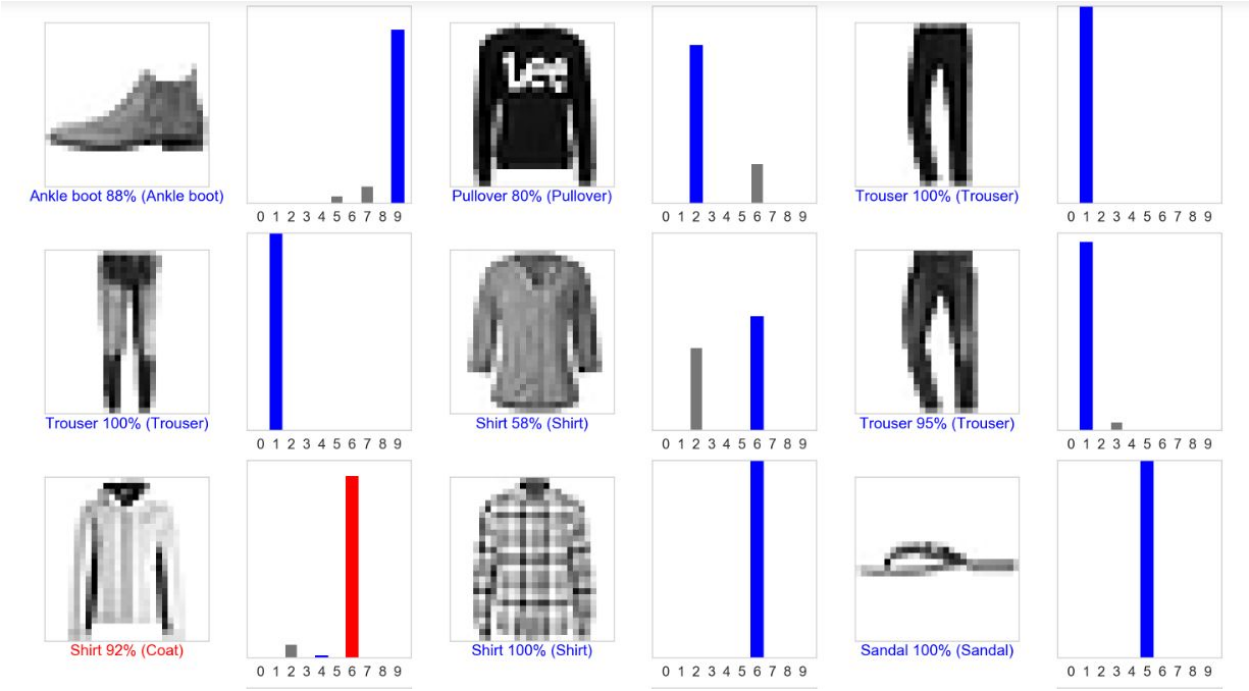


We estimate that increasing the number of hidden nodes gives us better performance so now we fix the number of hidden nodes to 40 and recall the trained parameters w1,b1,w2,b2 to predict the test data through forward neural network which gives us an **accuracy of 69%**

The overall performance is as follows:

| | Training Data | Validation Data | Test Data |
|---|---|---|---|
| **Accuracy** | **78.89 %** | **75.55%** | **69%** |

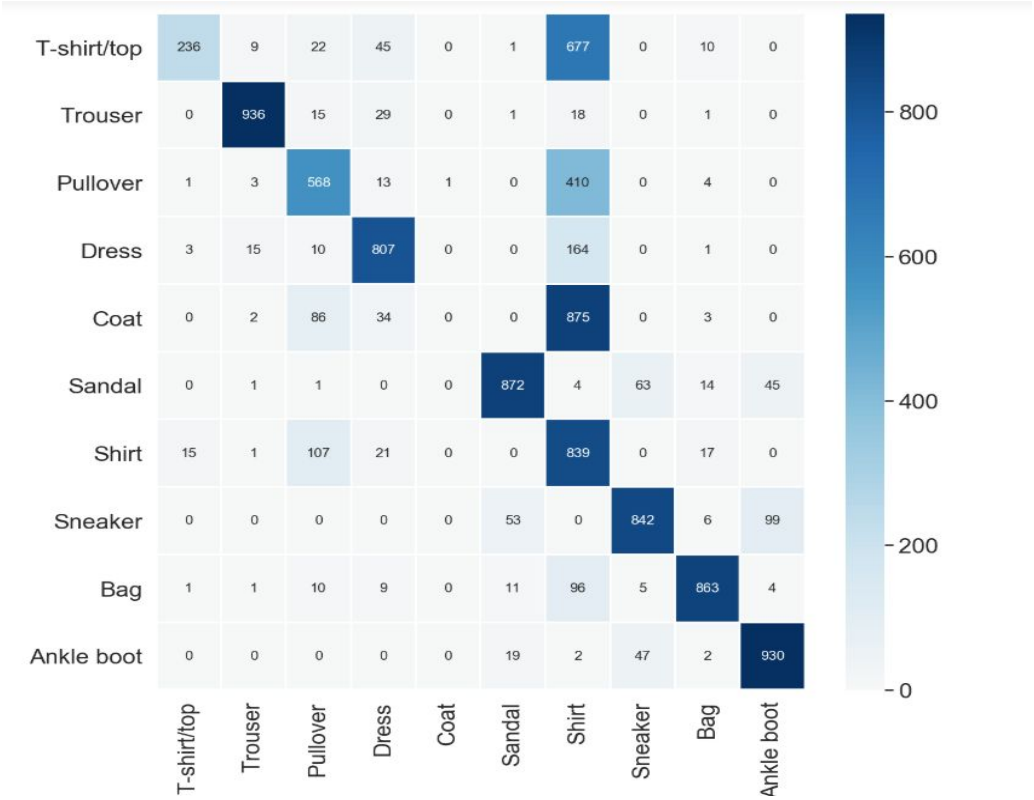
We plot the sample test images and check the full set of 10 class predictions as below:




## Confusion Matrix

To check the accuracy of test dataset we introduce a concept of confusion matrix which we import using the confusion_matrix method of sklearn metrics class. The confusion matrix is a way of tabulating the number of misclassifications, i.e., the number of predicted classes which ended up in a wrong classification bin based on the true classes. It is a table with four different combinations of predicted and actual values.

The confusion matrix in our case is as follows:




We will use the Confusion Matrix to calculate the value of Accuracy, Precision ,Recall and F1- score .

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 0.24 | 0.38 | 1000 |
| 1 | 0.97 | 0.94 | 0.95 | 1000 |
| 2 | 0.69 | 0.57 | 0.62 | 1000 |
| 3 | 0.84 | 0.81 | 0.82 | 1000 |
| 4 | 0.00 | 0.00 | 0.00 | 1000 |
| 5 | 0.91 | 0.87 | 0.89 | 1000 |
| 6 | 0.27 | 0.84 | 0.41 | 1000 |
| 7 | 0.88 | 0.84 | 0.86 | 1000 |
| 8 | 0.94 | 0.86 | 0.90 | 1000 |
| 9 | 0.86 | 0.93 | 0.90 | 1000 |
| accuracy | | | 0.69 | 10000 |
| macro avg | 0.73 | 0.69 | 0.67 | 10000 |
| weighted avg | 0.73 | 0.69 | 0.67 | 10000 |

## 2. Multi - Layer Neural Network

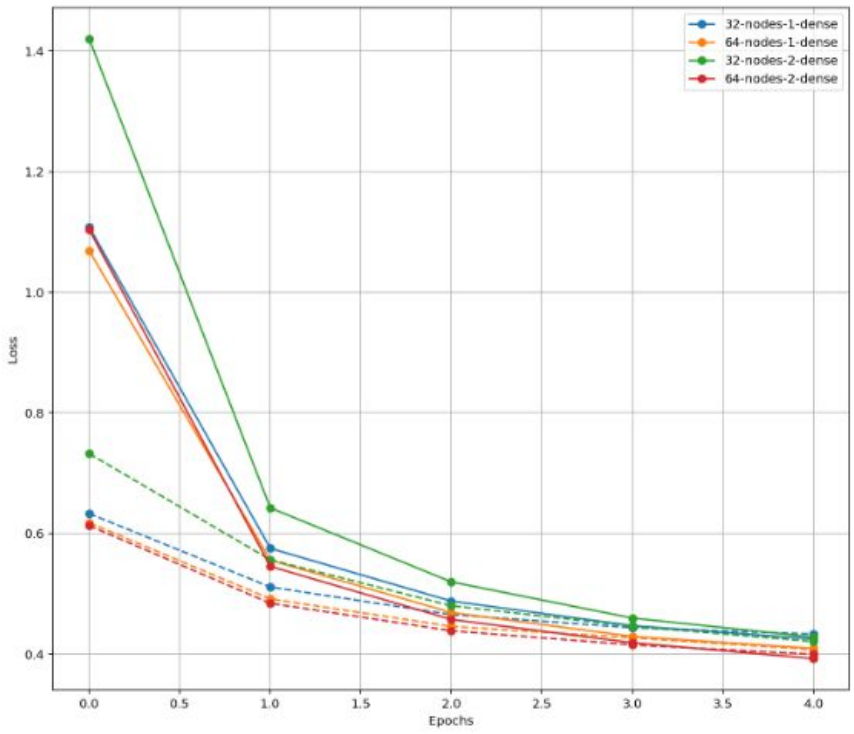
In Multi layer Neural Network, we use **relu as the primary activation function** in every hidden layer and in the final output layer we use **softmax activation function** since we are doing multi-class classification. We use adam as our optimizer and categorical cross entropy as the loss function and we evaluate the model on the accuracy metric.

**Hyperparameter Tuning**

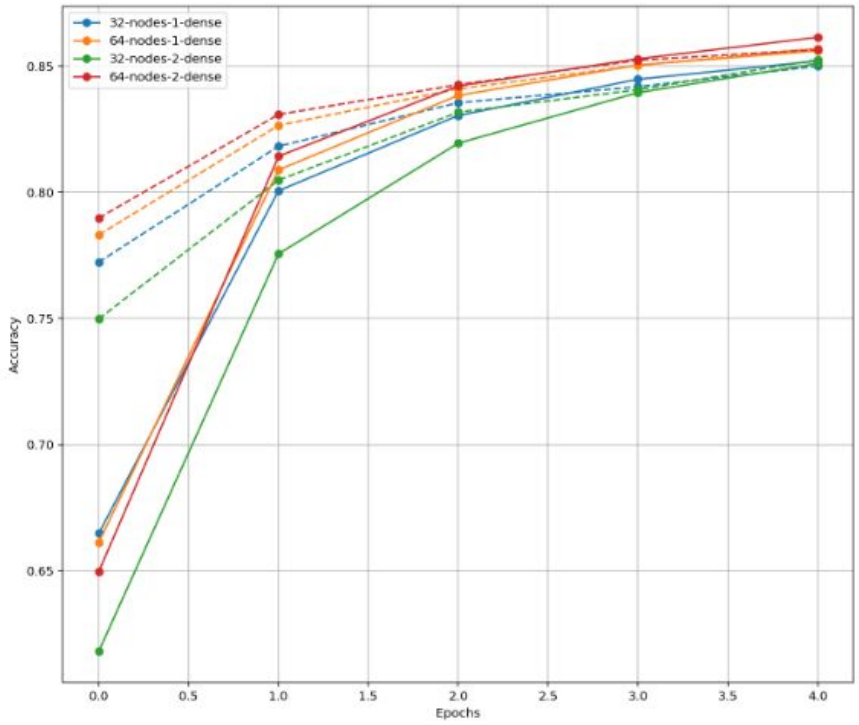
***We will consider two hyperparameter as the number of dense layers (1-layer, 2-layer) and the number of hidden nodes in each layer (32-nodes, 64-nodes) of the multi-layer neural network.***

Apart from the above hyperparameters we have three other important hyperparameters that decide the learning curve of the model which is learning rate, batch size and number of epochs. In this case, our **learning rate is set to 1e-3**, **batch size set to 2048** and **epochs set to 10**.

We record the Training and Validation loss with all the hyperparameters as:




We record the Training and Validation Accuracy with all the hyperparameters as:

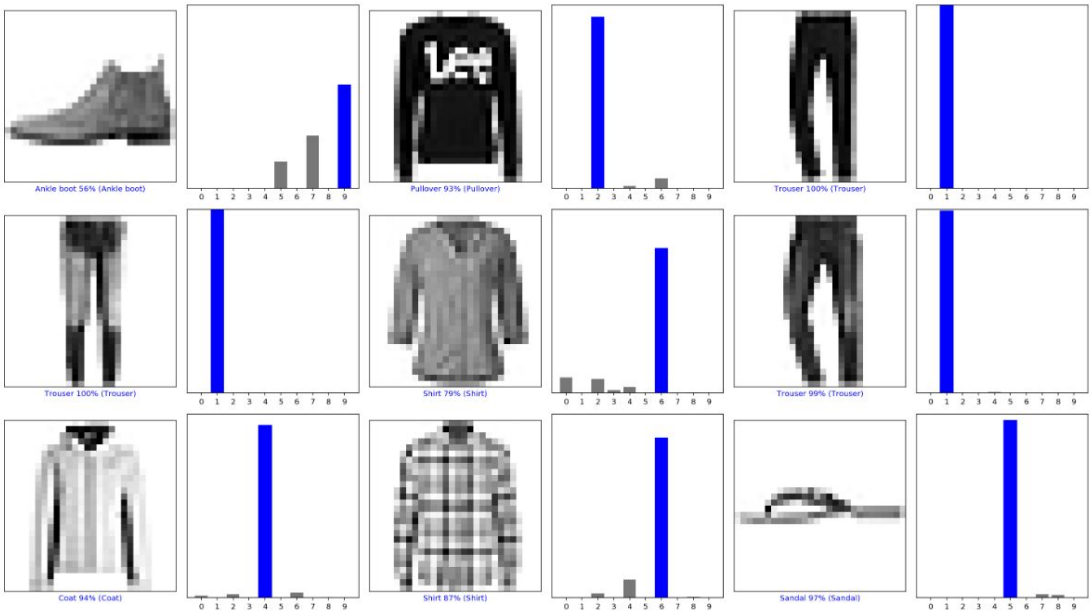



Now we estimate that the best model parameters which we get for 64 nodes in 2 dense layers and we predict the test data for this parameter to get an **accuracy of 84.84%**

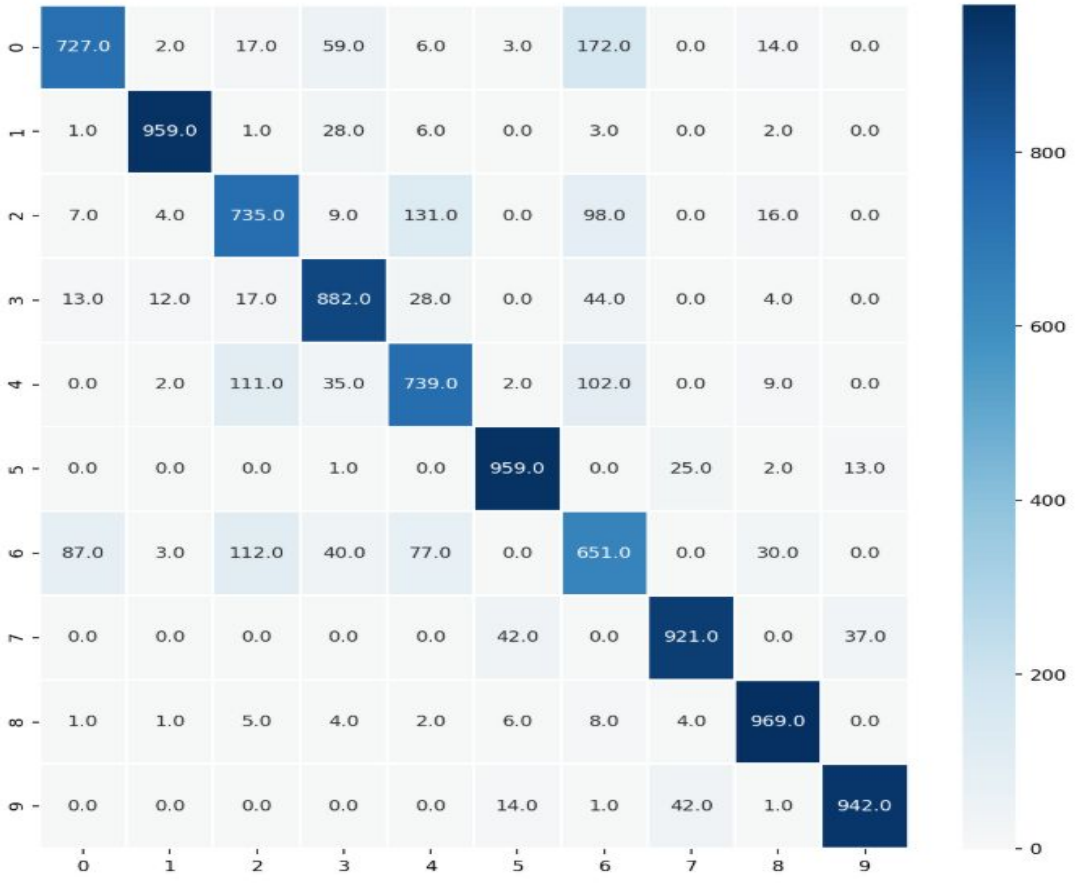
The overall performance is as follows:

| | Training Data | Validation Data | Test Data |
|---|---|---|---|
| **Accuracy** | **86%** | **85%** | **84.84%** |

We plot the sample test images and check the full set of 10 class predictions as below:




The confusion matrix in our case is as follows:



Now we will use the Confusion Matrix to calculate the value of Accuracy, Precision, Recall and F1 score.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.73 | 0.79 | 1000 |
| 1 | 0.98 | 0.96 | 0.97 | 1000 |
| 2 | 0.74 | 0.73 | 0.74 | 1000 |
| 3 | 0.83 | 0.88 | 0.86 | 1000 |
| 4 | 0.75 | 0.74 | 0.74 | 1000 |
| 5 | 0.93 | 0.96 | 0.95 | 1000 |
| 6 | 0.60 | 0.65 | 0.63 | 1000 |
| 7 | 0.93 | 0.92 | 0.92 | 1000 |
| 8 | 0.93 | 0.97 | 0.95 | 1000 |
| 9 | 0.95 | 0.94 | 0.95 | 1000 |
| accuracy | | | 0.85 | 10000 |
| macro avg | 0.85 | 0.85 | 0.85 | 10000 |
| weighted avg | 0.85 | 0.85 | 0.85 | 10000 |

### 3. Convolutional Neural Network:

In Convolutional Neural Network our kernel size is (3,3 ) and our max pooling layer divides the whole kernels height and width by 2 i.e. (2, 2).

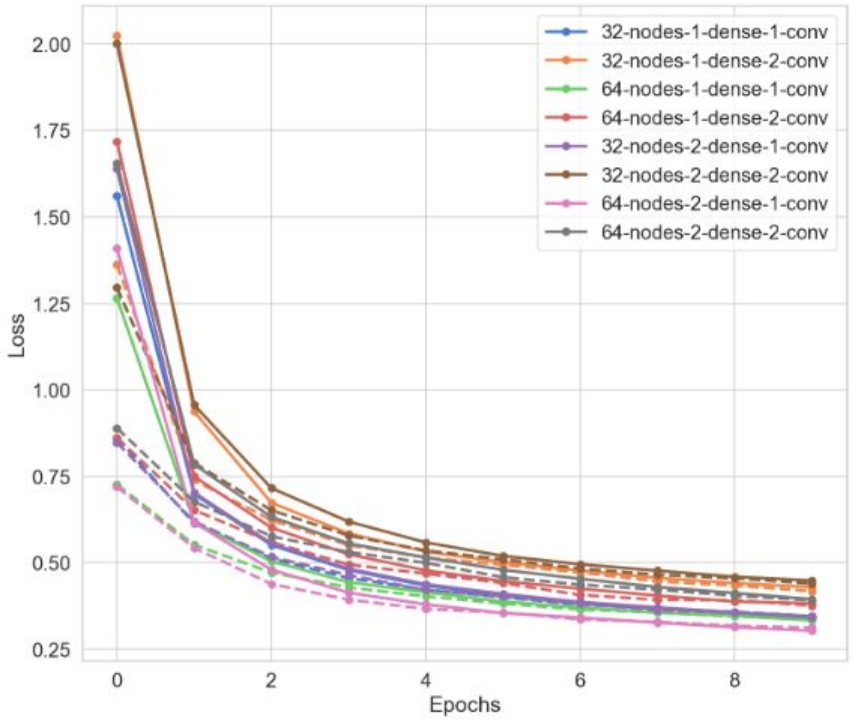
We use relu as the primary activation function in every hidden layer and in the final output layer we use softmax activation function since we are doing multi-class classification. We use adam as our optimizer and categorical cross entropy as the loss function and we evaluate the model on the accuracy metric.

**Hyperparameter Tuning**

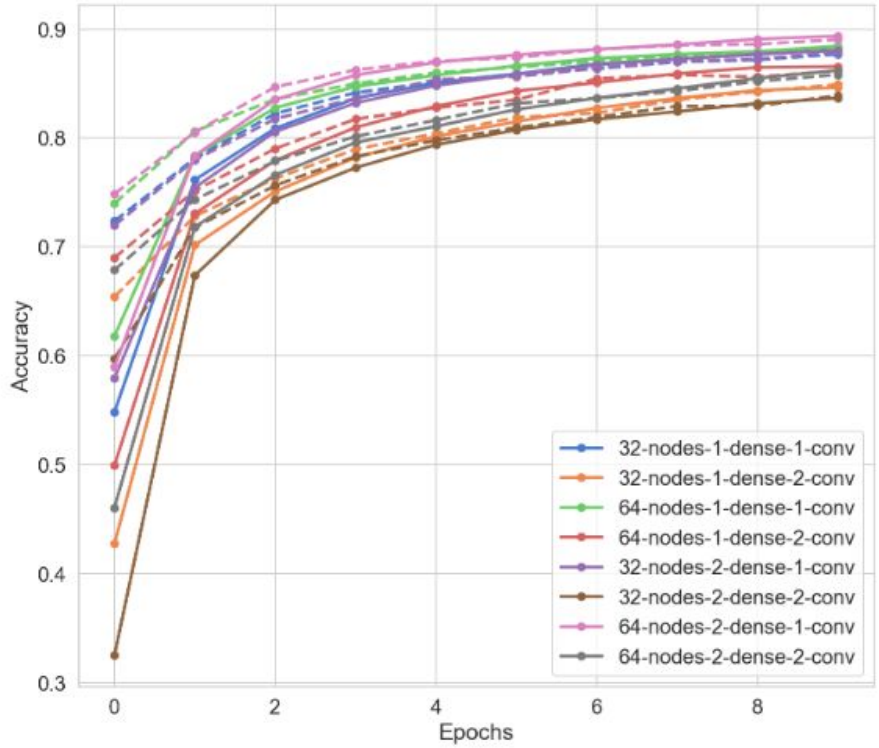
***We will consider three hyperparameter as the number of dense layers (2-layer, 3-layer) and the number of convolutional layer(1- layer, 2 layer) and number of hidden nodes in each layer (32-nodes, 64-nodes ) of convolutional neural networks.***

Apart from the above hyperparameters we have three other important hyperparameters that decide the learning curve of the model which is learning rate, batch size and number of epochs. In this case, our **learning rate is set to 1e-3**, **batch size set to 2048** and **epochs set to 10.**

We record the Training and Validation loss as:




We record the Training and Validation Accuracy as:



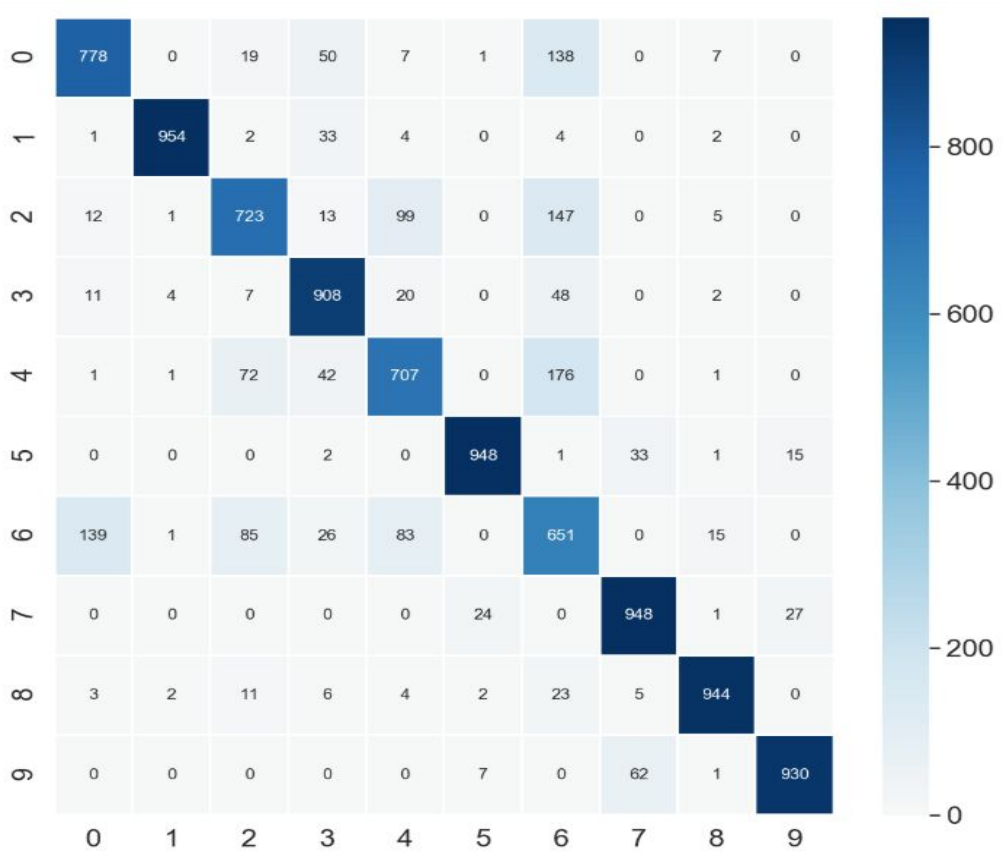
Now we estimate that the best model parameters are 64 nodes in 2 dense layers and we predict the test data for this parameter to get an **accuracy of 84.84%**

The overall performance is as follows:

| | Training Data | Validation Data | Test Data |
|---|---|---|---|
| **Accuracy** | **86%** | **85.%** | **85%** |

We plot the sample test images and check the full set of 10 class predictions as below:

The confusion matrix in our case is as follows:




Now we will use the Confusion Matrix to calculate the value of Accuracy, Precision, Recall and F1 score.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.82 | 0.78 | 0.80 | 1000 |
| 1 | 0.99 | 0.95 | 0.97 | 1000 |
| 2 | 0.79 | 0.72 | 0.75 | 1000 |
| 3 | 0.84 | 0.91 | 0.87 | 1000 |
| 4 | 0.77 | 0.71 | 0.73 | 1000 |
| 5 | 0.97 | 0.95 | 0.96 | 1000 |
| 6 | 0.55 | 0.65 | 0.60 | 1000 |
| 7 | 0.90 | 0.95 | 0.93 | 1000 |
| 8 | 0.96 | 0.94 | 0.95 | 1000 |
| 9 | 0.96 | 0.93 | 0.94 | 1000 |
| accuracy | | | 0.85 | 10000 |
| macro avg | 0.85 | 0.85 | 0.85 | 10000 |
| weighted avg | 0.85 | 0.85 | 0.85 | 10000 |

## 6. Conclusion

This paper presents the development and the simulation of a neural network and convolutional neural network machine learning model using Fashion-MNIST, a fashion product images dataset. The presented ML algorithm exhibited high performance on recognizing an image and identify it as one of the ten designated classes. Consequently, the statistical measures on the classification problem were also satisfactory.

## 7. Acknowledgments

We are extremely grateful to Professor Sargur Srihari and Mihir Chauhan for teaching all the necessary concepts related to Logistic Regression and helping in this project throughout.

## 8.References

- **https://www.nicolamanzini.com/single-hidden-layer-neural-network/**
- **https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist**
- **https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53**
- **https://www.apress.com/gp/book/9781484236727**