CSE574 Project2: Multi Class Classification using Neural Network and Convolutional Neural Network

Mansi Wagh
Department of Computer Science
University at Buffalo
Buffalo, NY 14226
mansiwag@buffalo.edu

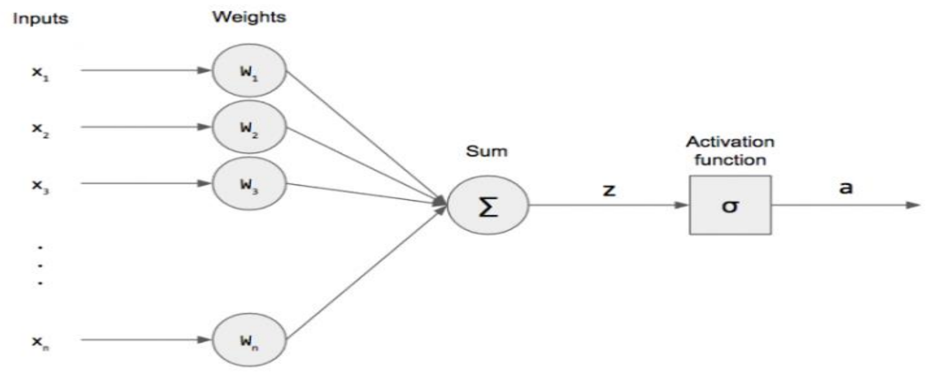
Abstract

The main task of this project is to implement multiclass classification of images in a dataset using neural network and convolutional neural netwok.The dataset used for the implementation of this project is the Fashion-MNIST clothing images dataset which is splitted into training and testing classifiers. The training set consists of 60,000 examples and a test set has 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. These training and testing classifiers are trained by neural network and convolutional neural network models to achieve the multiclass classification. This classification will recognize each image and map it to one of the ten classes.

1. Introduction

Neural networks are a set of algorithms, modeled after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, held in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

Neural networks help us cluster and classify. It acts like a classification layer on top of the data you store and manage. They help to group unlabeled data according to similarities among the example inputs, and they classify data when they have a labeled dataset to train on. Neural networks can also extract features that are fed to other algorithms for clustering and classification; so neural networks can be thought of as the components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression. Following image gives the overview of the working of neural network with one hidden layer.

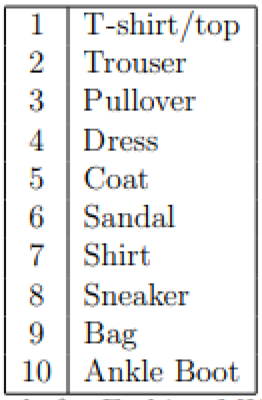


Our task 1 comprises of classifying the Fashion MNIST image-dataset using single hidden layer neural network as the classifier model which helps to categorize each fashion image into label associated with 10 classes. The neural network having one hidden layer in this project trains the dataset using techniques called forward propagation and backward propagation to give the desired output.

Task 2 and 3 of this project makes use of the concept of multi-layer neural network and convolutional neural network to achieve multiclass classification of our fashion image dataset respectively.A convolutional neural network is an algorithm which can take in an input image, assign importance to various objects in image and differentiate one from another. A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image . This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.Finally we plot graphs of the loss vs epoch and calculate accuracy, confusion matrix for each task respectively.

2. Dataset Description

For training and testing of our classifiers, we use the Fashion-MNIST dataset. 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 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. Each training and test example is 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

This Fashion MNIST dataset is loaded using fashion mnist reader notebook present inside the scripts folder.

3. Dataset Preprocessing

In this project, as we are given the fashion image dataset, we need not perform preprocessing for every task. The images given in the dataset are already flattened and available in the usable form. Although we directly use the input data without any preprocessing for task 1 and 2, we do need to implement techniques for enhancing its usability. Following are the techniques used after loading the dataset

3.1 One Hot encoding

Machine learning algorithms cannot work with categorical data directly. Categorical data must be converted to numbers. A one hot encoding is a representation of categorical variables as binary vectors. This first requires that the categorical values be mapped to integer values. Then, each integer value is represented as a binary vector that is all zero values except the index of the integer, which is marked with a 1.

Consider we have a sequence of labels with the values ‘T-shirt’ and ‘Bag’. We can assign ‘T-shirt’ an integer value of 0 and ‘Bag’ the integer value of 1. When we assign these numbers to these labels, this is called an integer encoding. Then we create a binary vector to represent each integer value. The vector will have a length of 2 for the 2 possible integer values. The ‘T-shirt’ label encoded as a 0 will be represented with a binary vector [1, 0] where the zeroth index is marked with a value of 1. In turn, the ‘Bag’ label encoded as a 1 will be represented with a binary vector [0, 1] where the first index is marked with a value of 1. We used numpy argmax() function to achieve one hot encoding in all of our tasks.

3.2 Unflatten the image

In task3, where we perform 2D convolution, we unflatten the image which was already flattened. The process of building a Convolutional Neural Network always involves four major steps: **Convolution, Pooling, Flattening and Full connection.** Convolution outputs a series of *filters*, which each are a grid shape. Flattening specifies a function mapping from these filters to a vector, so you can backpropagate errors back through the convolutional layers. So in order to do convolution, which is the essential step in the building of convolutional neural network, we unflatten the dataset.

4. Model Architecture

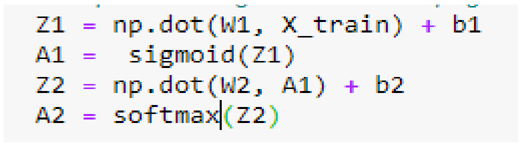
To obtain multiclass classification of our image dataset, we use 3 models in our project which are performed in task 1, 2 and 3 respectively..

4.1 Neural Network with one hidden layer

Here we build a neural network with 28X28 input layer, one hidden layer, and one output layer. The hidden layer has 90 nodes. We choose the nodes randomly after tunning the hyper-parameters to the node which gives the best value.The output layer has 10 nodes since we are solving a multiclass classification problem, where possible outputs can be number of labels in our dataset which is 10. The basic working principle of this model starts with the forward propagation phase where inputs from the previous layer are multiplied with the corresponding weights and are passed through the activation function to get the final value for the corresponding node in the next layer. This process is repeated for all the hidden layers until the output is calculated. In the back-propagation phase, the predicted output is compared with the actual output and the cost of error is calculated. The purpose is to minimize the cost function.

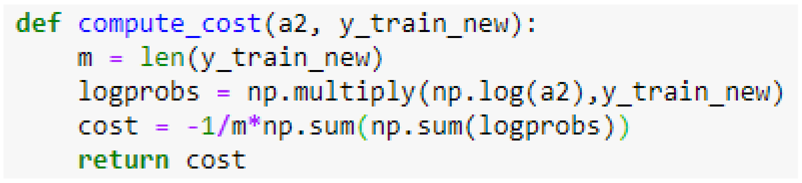
In forward propagation, we have 784 features from x1 to x784. To calculate the values for each node in the hidden layer, we have to multiply the input with the corresponding weights of the node for which we are calculating the value. We then pass the dot product through an activation function to get the final value. We use sigmoid and softmax activation to get the desired values.

Similarly, to calculate the value for the output layer, the values in the hidden layer nodes are treated as inputs. Therefore, to calculate the output, multiply the values of the hidden layer nodes with their corresponding weights and pass the result through an activation function.



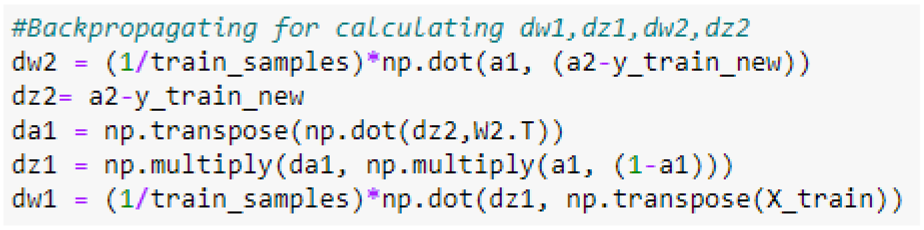
```
Z1 = np.dot(W1, X_train) + b1
A1 =  sigmoid(Z1)
Z2 = np.dot(W2, A1) + b2
A2 = softmax(Z2)
```

Thus we get the value of A2 which is the final output of our neural network. The next step we implement is back propogation in which we define our loss function. We defined compute_cost() function which takes our final output layer A2 and the training set to calculate our loss.



```
def compute_cost(a2, y_train_new):
    m = len(y_train_new)
    logprobs = np.multiply(np.log(a2),y_train_new)
    cost = -1/m*np.sum(np.sum(logprobs))
    return cost
```

In back propagation, we update the weights in such a way that the loss function is minimized. To find the minimum cost, we use gradient descent algorithm. Mathematically we implement backward propagation as follows.



```
#Backpropagating for calculating dw1,dz1,dw2,dz2
dw2 = (1/train_samples)*np.dot(a1, (a2-y_train_new))
dz2= a2-y_train_new
da1 = np.transpose(np.dot(dz2,W2.T))
dz1 = np.multiply(da1, np.multiply(a1, (1-a1)))
dw1 = (1/train_samples)*np.dot(dz1, np.transpose(X_train))
```

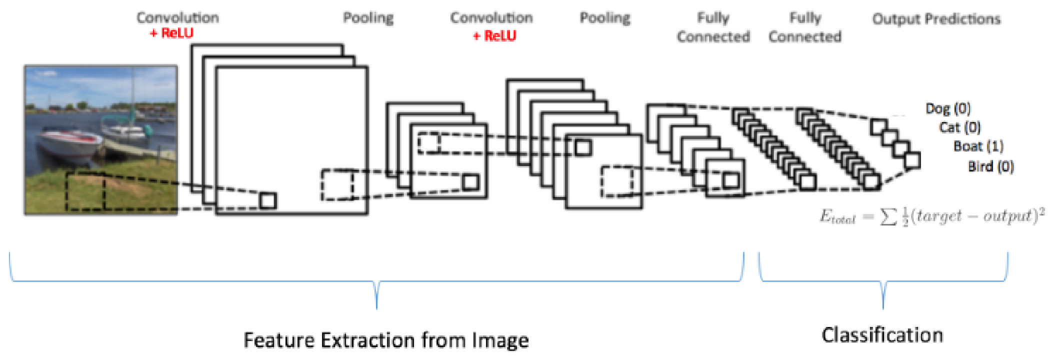
Our weights get updated after every iteration and our model keeps improving. Ultimately it assigns the feature to its class.

4.2 Neural Network with multiple hidden layer

We implement this model in our task 2. We use open-source neural-network library, Keras on Fashion MNIST dataset to do map the images to the classes. Here we use relu and softmax activation functions to forward propagate our data. Here we have multiple hidden layers.Our model is trained using our training set and it keeps learning when it is iterating over the number of epochs.

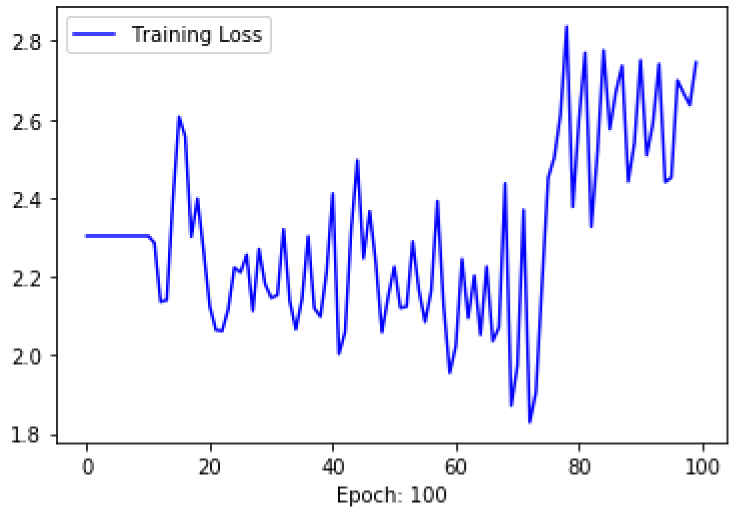
4.3 Convolutional Network Network

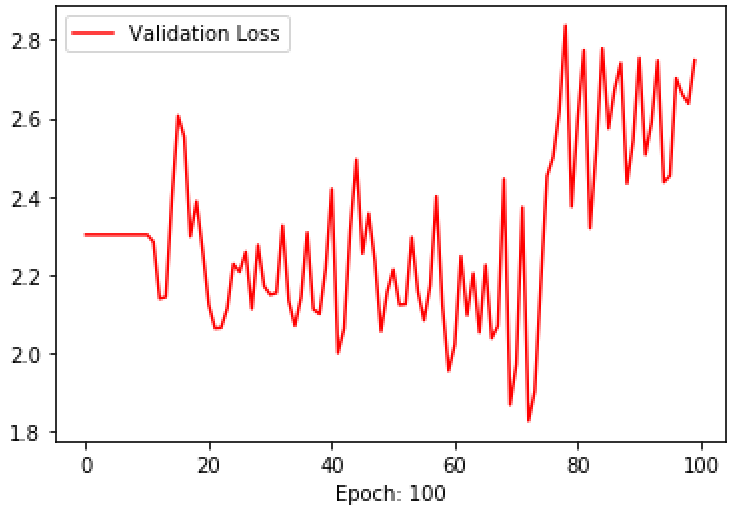
Task 3 of our project includes implementation of Convolutinal Neural Networks using open-source neural-network library, Keras on Fashion MNIST dataset . The convolution layer is the main building block of a convolutional neural network. The convolution layer comprises of a set of independent filters . Each filter is independently convolved with the image and we end up with 6 feature maps of shape 28*28*1. All these filters are initialized randomly and become our parameters which will be learned by the network subsequently. Through back propagation, filters tune themselves to become blobs of coloured pieces and edges. As we go deeper to other convolution layers, the filters are doing dot products to the input of the previous convolution layers. So, they are taking the smaller coloured pieces or edges and making larger pieces out of them. In our project we use Conv2D() which gives us the convolutional layer. Pictorically convolutional neural networks are represented as follows



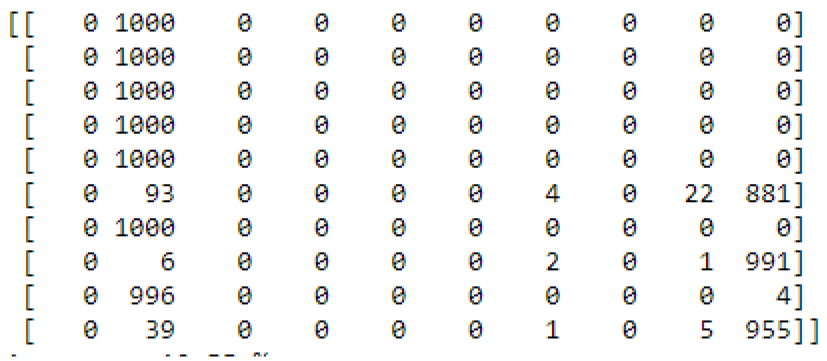
5. Results

Task 1: Output obtained from the implementation of **Neural Network having single hidden layer** is as follows





Confusion Matrix:

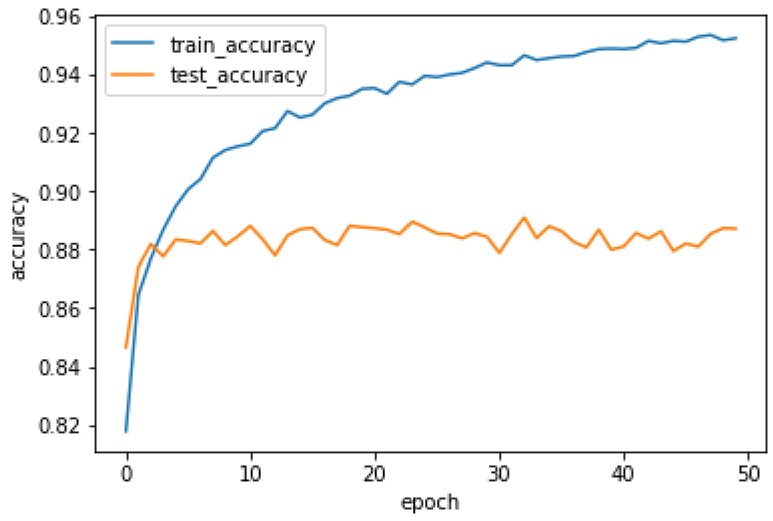


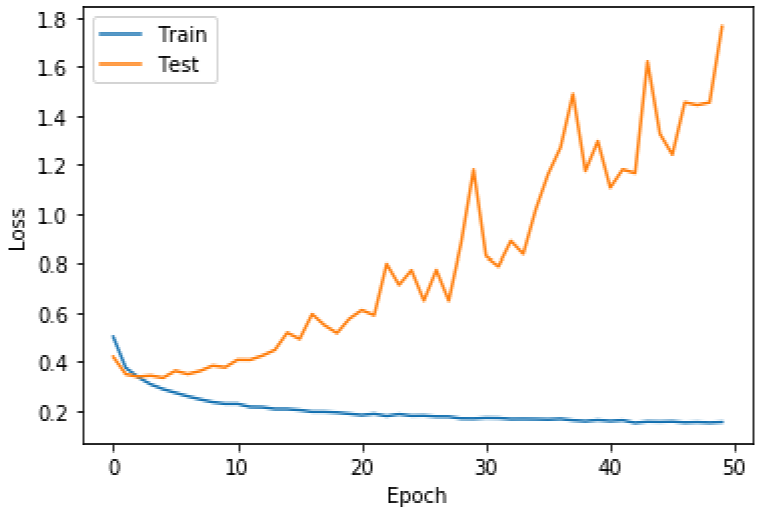
```
[[   0 1000    0    0    0    0    0    0    0    0]
 [   0 1000    0    0    0    0    0    0    0    0]
 [   0 1000    0    0    0    0    0    0    0    0]
 [   0 1000    0    0    0    0    0    0    0    0]
 [   0 1000    0    0    0    0    0    0    0    0]
 [   0   93    0    0    0    0    4    0   22  881]
 [   0 1000    0    0    0    0    0    0    0    0]
 [   0    6    0    0    0    0    2    0    1  991]
 [   0  996    0    0    0    0    0    0    0    4]
 [   0   39    0    0    0    0    1    0    5  955]]
```

Accuracy: 19.55%

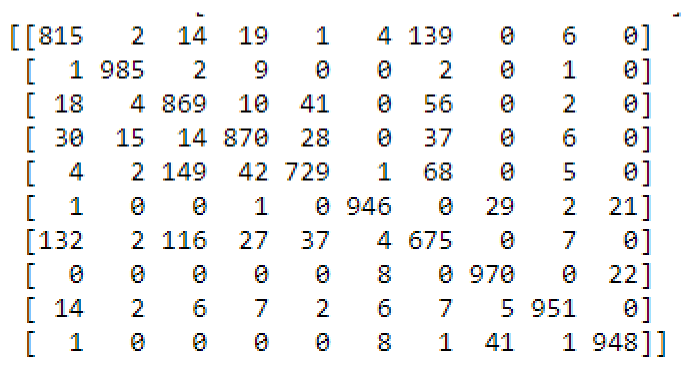
Task 2: Output obtained from the implementation of **Multi Layer Neural Network** using Keras is as follows:

Epochs taken: 50





Confusion matrix:

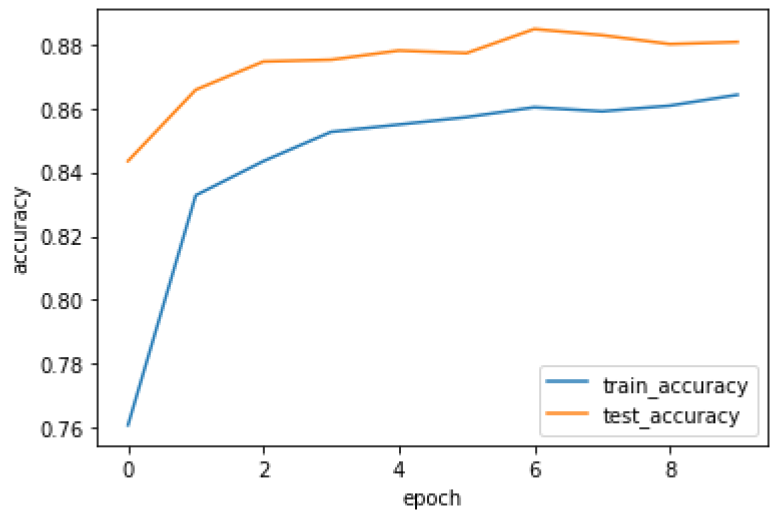


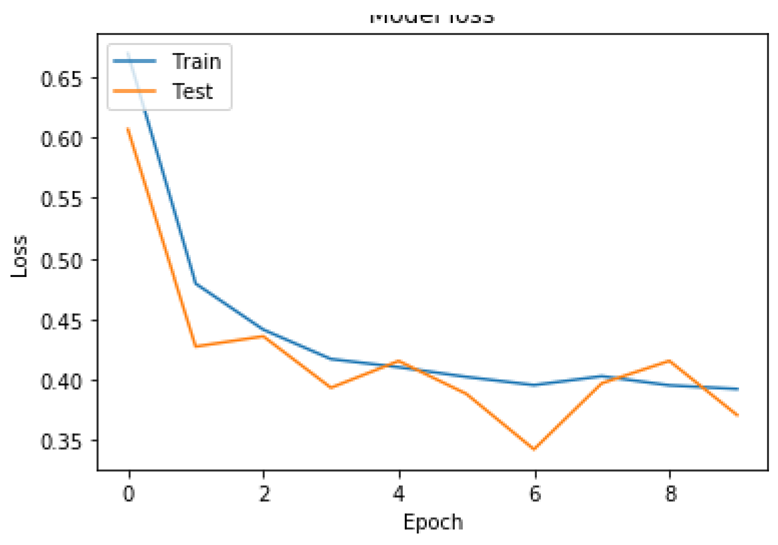
```
[[815   2  14  19   1   4 139   0   6   0]
 [  1 985   2   9   0   0   2   0   1   0]
 [ 18   4 869  10  41   0  56   0   2   0]
 [ 30  15  14 870  28   0  37   0   6   0]
 [  4   2 149  42 729   1  68   0   5   0]
 [  1   0   0   1   0 946   0  29   2  21]
 [132   2 116  27  37   4 675   0   7   0]
 [  0   0   0   0   0   8   0 970   0  22]
 [ 14   2   6   7   2   6   7   5 951   0]
 [  1   0   0   0   0   8   1  41   1 948]]
```

Accuracy: 87.3%

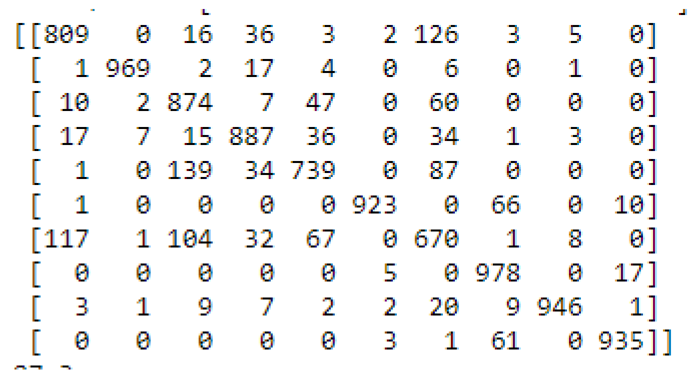
Task 3: Output obtained from the implementation of **Convolutional Neural Network** using Keras is as follows:

Epochs taken: 30





Confusion Matrix:



```
[[809   0  16  36   3   2 126   3   5   0]
 [  1 969   2  17   4   0   6   0   1   0]
 [ 10   2 874   7  47   0  60   0   0   0]
 [ 17   7  15 887  36   0  34   1   3   0]
 [  1   0 139  34 739   0  87   0   0   0]
 [  1   0   0   0   0 923   0  66   0  10]
 [117   1 104  32  67   0 670   1   8   0]
 [  0   0   0   0   0   5   0 978   0  17]
 [  3   1   9   7   2   2  20   9 946   1]
 [  0   0   0   0   0   3   1  61   0 935]]
```

Accuracy: 87.58%

6. Conclusion:

From the above results obtained from all the 3 tasks, we can conclude that our convolutional neural network having an accuracy of 87.58% performs the best among single hidden layer neural network and multi-layer neural network. It classifies our images from the Fashion MNIST dataset and maps it to its best match label amongst the 10 classes.

7. References:

1. https://skymind.ai/wiki/neural-network
2. http://deeplearning.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/
3. https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/
   for token in tokenize(text):
4. https://stackabuse.com/creating-a-neural-network-from-scratch-in-python-adding-hidden-layers/
5. https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8