# NEURAL NETWORKS

**Deepali Jain**
Department of Computer Science and Engineering
University of Buffalo
Buffalo, NY 14212
*deepalij@buffalo.edu*

## Abstract

This project implements neural network and convolutional neural network. The task is to carry out classification on Fashion-MNIST dataset. There are 10 classes of different types of clothing. Our task is to recognize an image and identify it as one of the ten classes. We will train classifiers using images from Zalando's clothing article. The project is divided into three tasks. The first task is to build single layer(hidden) neural network from scratch in python. The second task would be to build a multi-layer Neural Network with open-source neural-network library, Keras. The third task would be to build Convolutional Neural Network (CNN) with open-source neural-network library, Keras. Layers in all three types of networks will be trained and tested on Fashion-MNIST dataset. The inference will consist of comparison between accuracy and loss between all three ways to classify and predict data using neural networks.

## 1 Introduction

### 1.1 Neural Network

Neural networks are a set of algorithms, modeled loosely 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, contained 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. You can think of them as a clustering and 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 you can think of deep neural networks as components of larger machine-learning applications involving algorithms for reinforcement learning, classification and regression.) [1]

### 1.2 Single layer Neural Network

Single Layer Neural Network consists of one layer of input nodes, one hidden layer of nodes and one layer of output nodes. The input layer sends weighted input to subsequent hidden layer which further sends the weighted result to output layer. Some examples of single slayer neural network are perceptron, logic gates, single-layer binary linear classifier, etc.

### 1.3 Multi-Layer neural network

A multi-layer Neural Network is made up of many layers, including one input layer, one output layer and many hidden layers. The interior layers are called hidden layers because they are like a black box i.e. not directly observable from input & output side. Initially single layer neural networks were used in the field of AI and deep learning. The need to have multi-layer network was to understand data interaction in more depth on a non-linear level. The means that single layer networks were good for problems that were linearly separable like 2 class classification problem but multi-layer neural networks can solve high dimensional problems with more complexity.

### 1.4 Convolutional Neural Network (CNN)

Convolutional Neural Network is an algorithm that enables machines to perform tasks with human capabilities like pattern recognition, Image classification, etc. The flow of data in a CNN is as follows: The first step is Input layer whose main responsibility is to do image reduction in such a way to make prediction keeping the essential features intact. The second step is Convolutional layer performed at the kernel. The third step is Pooling layer which decreases the computational power required to process the image and extracts the dominant features from the image. There are two types of pooling – average pooling and max pooling. We will use max pooling in this project. The final step would be Classification – Fully connected Layer (FC Layer). We used ReLu Activation, Sigmoid function and SoftMax function in this layer.

## 2 Data Set

Fashion-MNIST dataset is a drop-in replacement for the original MNIST dataset. It is made by researchers at Zolando, an e-commerce fashion company. The dataset consists of clothing images from different generes like shoes, shirts, trousers, bags, etc. It consists of 60000 training data and 10000 testing data with 785 columns each. The images are grascale with 28X28 pixel size. Each pixel is associalted with a single value indicating the darkness of the image in a range from 0 to 255 with 0 being the lightest and 255 being the darkest. There are 10 different classes used as the classifier.

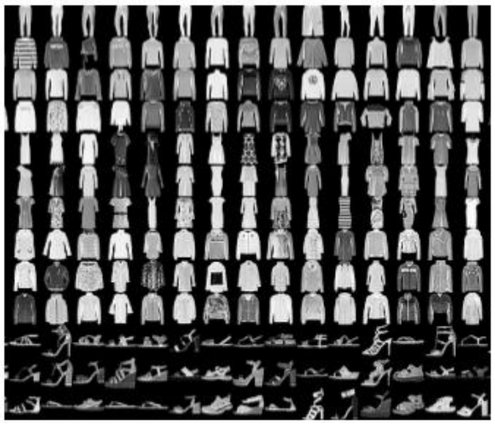


Fig: Snippet of the images in the Fashion-MNIST dataset

| | |
|---|---|
| **0** | T-shirt/Top |
| **1** | Trouser |
| **2** | Pullover |
| **3** | Dress |
| **4** | Coat |
| **5** | Sandal |
| **6** | Shirt |
| **7** | Sneaker |
| **8** | Bag |
| **9** | Ankle Boot |

Table: 10 classes representing class labels

## 3 Pre-processing

### 3.1 ***Load data from mnist reader***

Train and test data are loaded using the dataset library util_mnist_reader.

### 3.2 ***Split the training data into training and validation data***

The train data contains 60000 samples, split it into: -

i. Train data (55000)
ii. Validation data (5000)

### 3.3 ***Normalization of the data***

Normalization of data means reorganizing the data values to a common scale. Here normalization was done to bring the values in the range between 0 and 1 that was earlier in the range [0,255]. Basically, we divided the data by 255.0

### 3.4 ***One Line Hot Encoding***

It converts the data into categorical data. We used **get_dummies()** from pandas to do this.
For example, (60000,) -> (60000,10)

### 3.5 ***Set the hyperparameters***

Hyperparameters are those parameters which are defined beforehand the beginning of the learning process. We used three hyperparameters in this project.

i. Epochs
ii. Learning Rate
iii. Number of nodes

### 3.6 ***Sigmoid function***

It maps the output value between 0 to 1 depending how the function works.

$$s(x) = \frac{1}{1 \div e^{-x}}$$

### 3.7 ***Derivative of Sigmoid function***

$$s'(x) = s(x)\,[1 - s(x)]$$

### 3.8 ***SoftMax function***

Softmax function outputs a vector that represents the probability distributions of a list of potential outcomes.

### 3.9 ***Cost Function***

In order to minimize loss, we use the concept of gradient descent.

Gradient descent calculates the slope of the loss function, then shifts the weights and biases according to that slope to a lower loss. Computing loss helps us to figure out where our progress is going with each epoch.

```
z = xw + b                    -> z = function(w)
a = sig(z) or softmax(z)      -> a = function(z)
c = -(y*log(a3))              -> c = function(a)
```

**3.10** ***Accuracy Function***

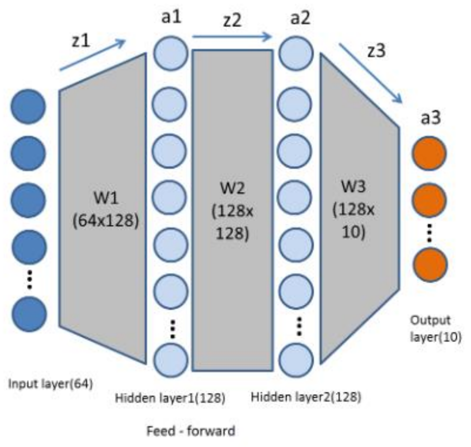
After finding the optimized parameters, we use this metrics from sklearns to evaluate accuracy of out model's prediction in comparison to true data.

**3.11** ***Initialization of weights and biases***

Initialize random values to weights and biases.

**3.12** ***Forward Feed***

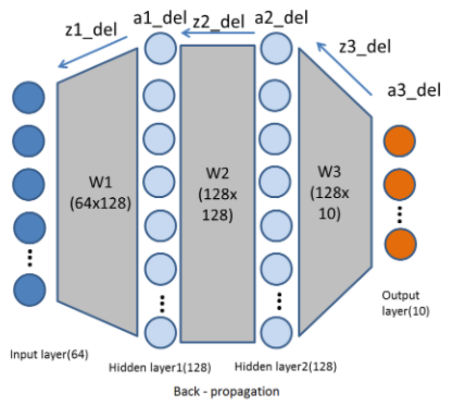
In Forward Propagation, values are sent to successive layers and learns values of hyper parameters for best prediction.



[2]Fig: Forward Propagation

**3.13** ***Backward Feed***

After finding the error in our prediction through forward propagation, we need to update the value of weights and biases. To do this, we need derivatives of loss function with respect to weights and biases. This is known as Backward propagation.



[3]Fig: Backward Propagation

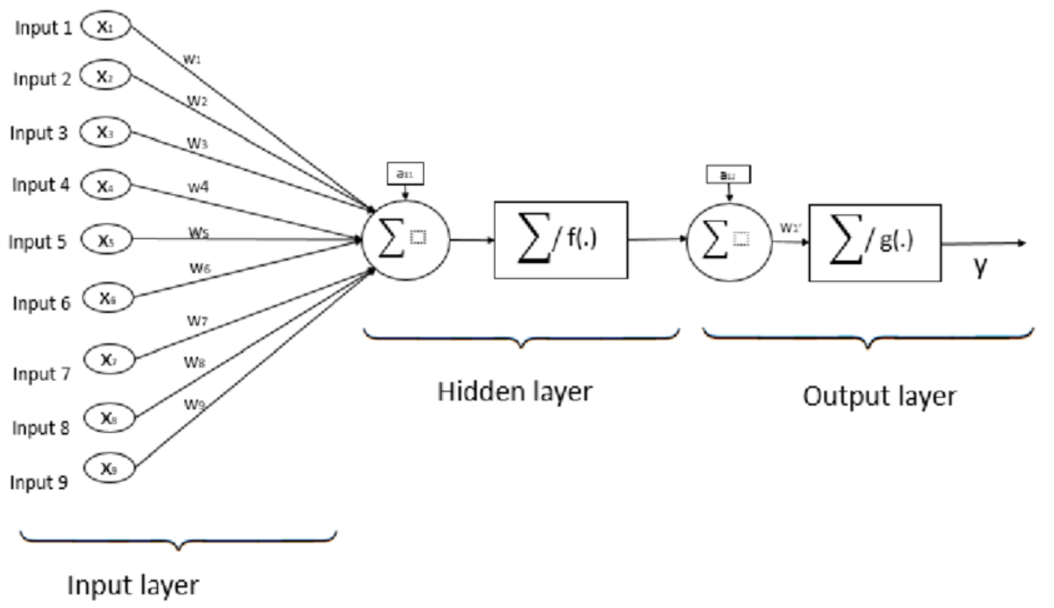
**3.14** ***Calculate Accuracy, Loss, Confusion matrix and Classification report***

Result is inferred on the basis these 4 parameters. Ideally, accuracy should increase and loss should decrease with increasing number of epochs. Confusion matrix is the summary of prediction results obtained by classification problem. Both Confusion matrix and Classification report are functions of sklearn.metrics library.

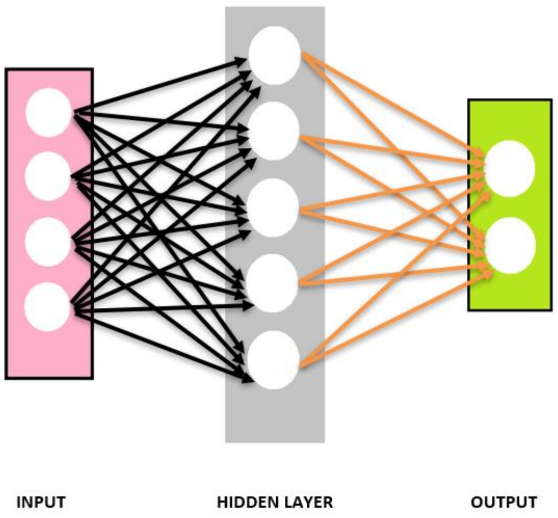
**3.15** ***Plot the graphs***

Using Matplotlib, we plotted two graphs – Accuracy and Loss.

## 4 Architecture

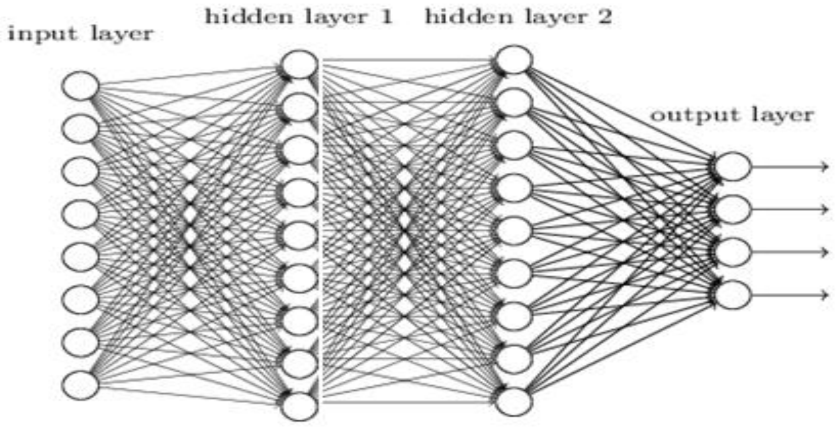


**[4]Fig: GENERAL EXAMPLE OF NEURAL NETWORKS**



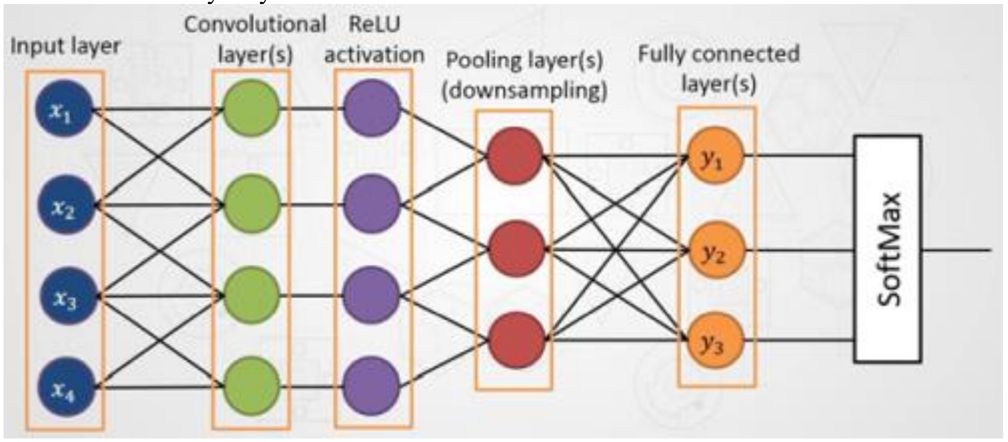
**SINGLE LAYER NEURAL NETWORK**

Single Layer Neural Network has the simplest architecture as compared to the other two networks. The training data is fed to the input layer. Using the equation $Z = X^TW + B$, the data is modified by weights and biases and passed to the only hidden layer between input and output layer. In the hidden layer, activation function is applied which uses sigmoid/softmax functions. Finally, we receive the updated value of parameters. We back propagate through the network, update values and obtain the correct prediction i.e. output.



**[5] Fig: MULTI LAYER NEURAL NETWORK**

Multi-Layer Neural Network is quite similar to Single Layer Neural Network but instead of having one hidden layer, it has more than one hidden layer. The output of genesis equation is modified by weights and biases for every layer.



**[6] Fig: CONVOLUTION NEURAL NETWORK**

Convolution is a weighted sum of the pixel values of the image, as window slides across the whole image. A typical CCN has multiple convolution layers and each CNN layer has many convolutions. The number of parameters is independent of the size of image. For example, for a 256 X 256 image, if we apply convolution with no of convolutions = 64 then,

Original: 256 * 256 = 65,536 parameters
Applying Convolution: 5 * 5 * 64 = 1600 parameters

## 5 Result

In this project we implemented Neural Network in three different ways and got different results. We can see in the following table that all three tasks have different accuracy and loss with task 3 being the most effiecient and task 1 being the least efficient.

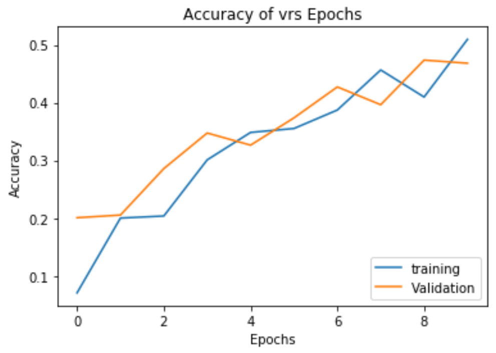
| **Task** | **Type of Neural Network used** | **Accuracy** | **Loss** |
|---|---|---|---|
| Task 1 | Single Layer Neural Network | 72.09 | 0.8544 |
| Task 2 | Multi Layer Neural Network | 88.64 | 0.4261 |
| Task 3 | Convolution Neural Network | 90.88 | 0.2553 |

We calculated four parameters to compare our results for the three types of neurel networks:

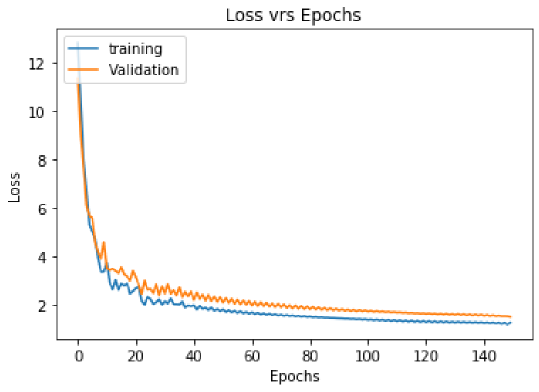
1. Accuracy
2. Loss
3. Confusion Matrix
4. Classification Report

<u>TASK 1:</u>

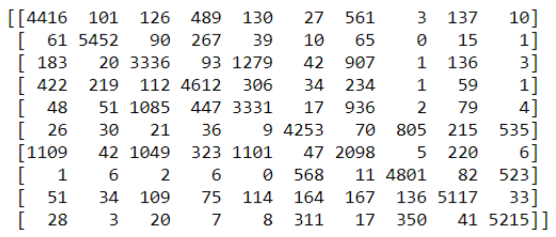
## 1. Accuracy



## 2. Loss

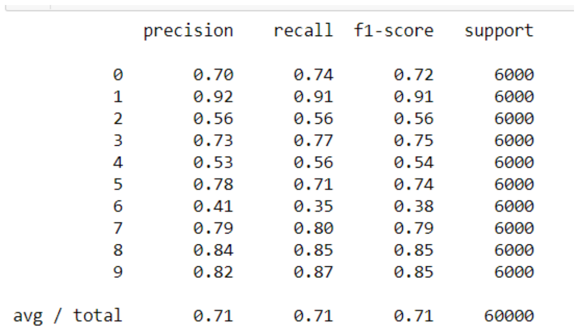


## 3. Confusion Matrix



```
[[4416  101  126  489  130   27  561    3  137   10]
 [  61 5452   90  267   39   10   65    0   15    1]
 [ 183   20 3336   93 1279   42  907    1  136    3]
 [ 422  219  112 4612  306   34  234    1   59    1]
 [  48   51 1085  447 3331   17  936    2   79    4]
 [  26   30   21   36    9 4253   70  805  215  535]
 [1109   42 1049  323 1101   47 2098    5  220    6]
 [   1    6    2    6    0  568   11 4801   82  523]
 [  51   34  109   75  114  164  167  136 5117   33]
 [  28    3   20    7    8  311   17  350   41 5215]]
```

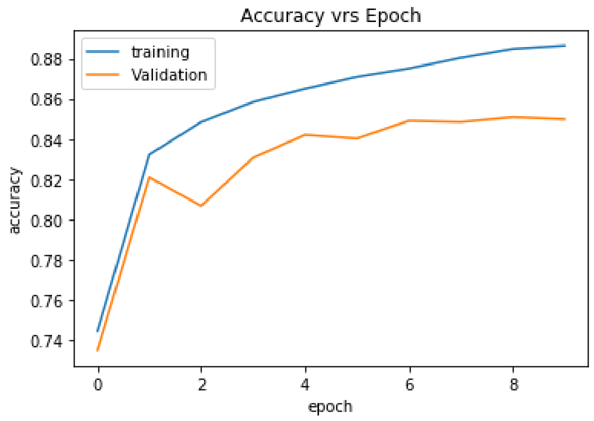
## 4. Classification Matrix



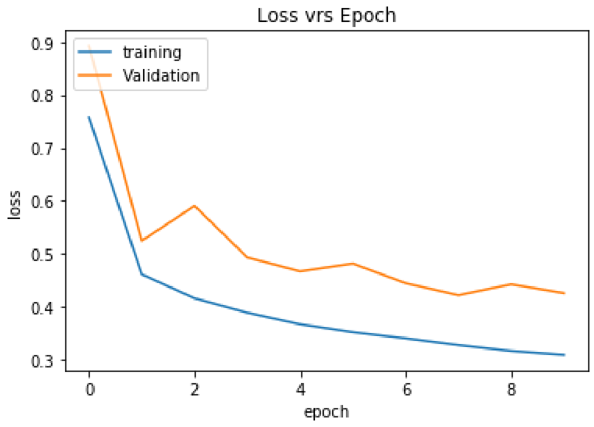
```
             precision    recall  f1-score   support

          0       0.70      0.74      0.72      6000
          1       0.92      0.91      0.91      6000
          2       0.56      0.56      0.56      6000
          3       0.73      0.77      0.75      6000
          4       0.53      0.56      0.54      6000
          5       0.78      0.71      0.74      6000
          6       0.41      0.35      0.38      6000
          7       0.79      0.80      0.79      6000
          8       0.84      0.85      0.85      6000
          9       0.82      0.87      0.85      6000

avg / total       0.71      0.71      0.71     60000
```

TASK 2:

## 1. Accuracy



## 2. Loss



## 3. Confusion Matrix

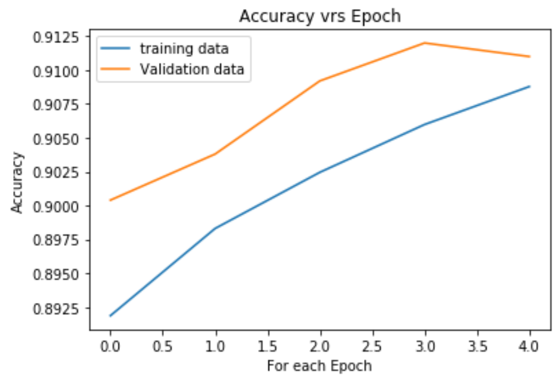
```
[[864   8  23  36   4   1  51   0  13   0]
 [  0 980   1  15   2   0   1   0   1   0]
 [ 17   6 855  10  63   0  46   0   3   0]
 [ 21  43  23 858  34   0  17   0   4   0]
 [  0   2 163  27 755   0  50   0   3   0]
 [  0   0   0   1   0 938   0  36   2  23]
 [178  11 127  44  65   0 557   0  18   0]
 [  0   0   0   0   0  33   0 934   0  33]
 [  3   1   3   6   3   2   2   3 977   0]
 [  0   0   0   0   0   4   1  37   0 958]]
```

## 4. Classification Matrix

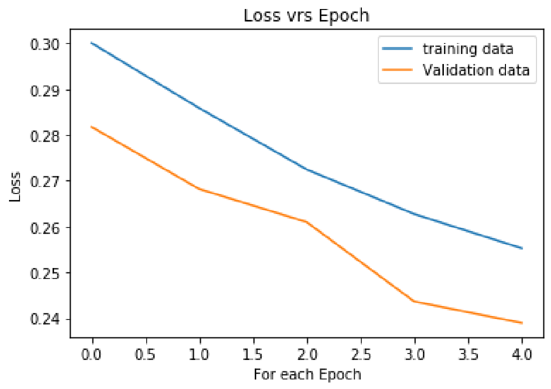
```
             precision    recall  f1-score   support

          0       0.80      0.86      0.83      1000
          1       0.93      0.98      0.96      1000
          2       0.72      0.85      0.78      1000
          3       0.86      0.86      0.86      1000
          4       0.82      0.76      0.78      1000
          5       0.96      0.94      0.95      1000
          6       0.77      0.56      0.65      1000
          7       0.92      0.93      0.93      1000
          8       0.96      0.98      0.97      1000
          9       0.94      0.96      0.95      1000

avg / total       0.87      0.87      0.86     10000
```

TASK 3:

1. **Accuracy**



2. **Loss**



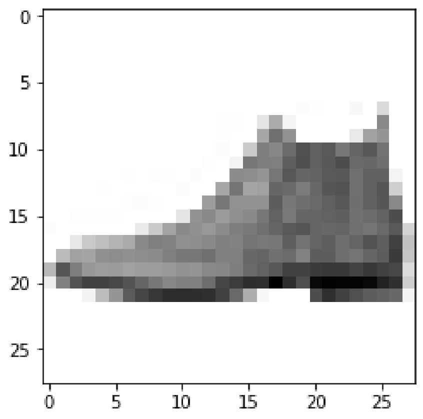
3. **Confusion Matrix**

```
[[871   0  26  15   3   1  80   0   4   0]
 [  1 978   2  13   3   0   1   0   2   0]
 [ 15   0 889   8  34   0  53   0   1   0]
 [ 24  11  20 884  21   1  37   0   2   0]
 [  2   1  93  29 794   0  81   0   0   0]
 [  0   0   0   0   0 976   0  17   0   7]
 [156   2  97  18  57   0 659   0  11   0]
 [  0   0   0   0   0  12   0 967   1  20]
 [  4   1   9   3   1   3   8   4 967   0]
 [  1   0   0   0   0   4   0  33   0 962]]
```

4. **Classification Matrix**

```
             precision    recall  f1-score   support

          0       0.81      0.87      0.84      1000
          1       0.98      0.98      0.98      1000
          2       0.78      0.89      0.83      1000
          3       0.91      0.88      0.90      1000
          4       0.87      0.79      0.83      1000
          5       0.98      0.98      0.98      1000
          6       0.72      0.66      0.69      1000
          7       0.95      0.97      0.96      1000
          8       0.98      0.97      0.97      1000
          9       0.97      0.96      0.97      1000

avg / total       0.90      0.89      0.89     10000
```

5. **Image for the given test data**



In this graph, both validation and training data follow almost the same curve showing that the training data was trained effectively and it worked well on validation data as well. Observing the loss and accracy graphs for all the three tasks, it is infered that Convolutional Neural Network produces the most accurate predictions as compared to Single Layer Neural Networks and Multi Layer Neural Networks.

## 6 Conclusion

Neural Networks implemented by three different ways. The conclusion is Convolutional Neural gave best accuracy(90.88) and least loss(0.2553).

### References

[1] https://www.quora.com/In-multilayer-neural-networks-are-weights-in-hidden-layers-closer-to-the-input-updated-more-strongly-than-weights-in-layers-closer-to-the-output

[2] https://towardsdatascience.com/neural-networks-from-scratch-easy-vs-hard-b26ddc2e89c7

[3] https://towardsdatascience.com/neural-networks-from-scratch-easy-vs-hard-b26ddc2e89c7

[4] https://www.simplilearn.com/multilayer-artificial-neural-network-tutorial

[5] https://dimensionless.in/what-is-neural-network/

[6] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Other References:

[7] https://towardsdatascience.com/mnist-cnn-python-c61a5bce7a19

[8] https://towardsdatascience.com/neural-networks-from-scratch-easy-vs-hard-b26ddc2e89c7