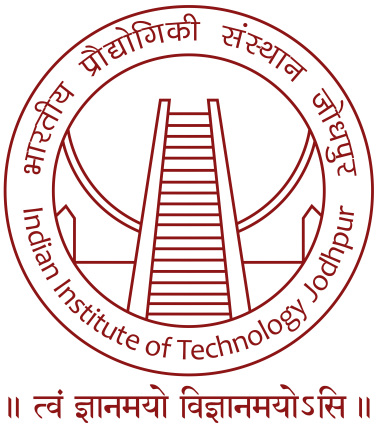
Indian Institute of Technology, Jodhpur



CSL7620 Machine Learning

Assignment 3

Name:- Sahil **Roll No.:-** M21MA210

M.Sc - M.Tech (Data and Computational Sciences)
Department of Mathematics

Submission Date: November 15, 2023

Contents

List of Figures

1 Building a Neural Network from Scratch

Google Collab Notebook link:-
https://colab.research.google.com/drive/1qVt1Q5ppGDyFZBtb0kwuEls7uu2VpiAg?usp=sharing

1.1 About the given dataset

The given dataset consists of a total of $70,000$ images. Each image in the dataset is of 28×28 pixels and contains multi-class images labelled from 0 to 9.

1.2 Exploratory Data Analysis

Using the given dataset, we can analyze the following characteristics of it:-

- As our dataset has 10 different types of images labeled from 0 to 9, so we have counted the number of occurrences of each label, and the result is as follows:-

Label No.	Occurences
0	7000
1	7000
2	7000
3	7000
4	7000
5	7000
6	7000
7	7000
8	7000
9	7000

Figure 1: Occurence of different labels

- Digit(Label) Count in the dataset (Histogram)

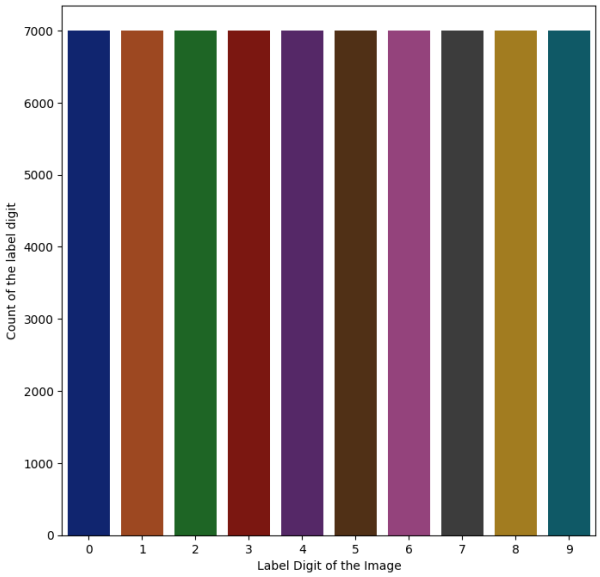


Figure 2: Label Count Histogram

- Distribution of the pixel intensity among all the images given in the dataset.

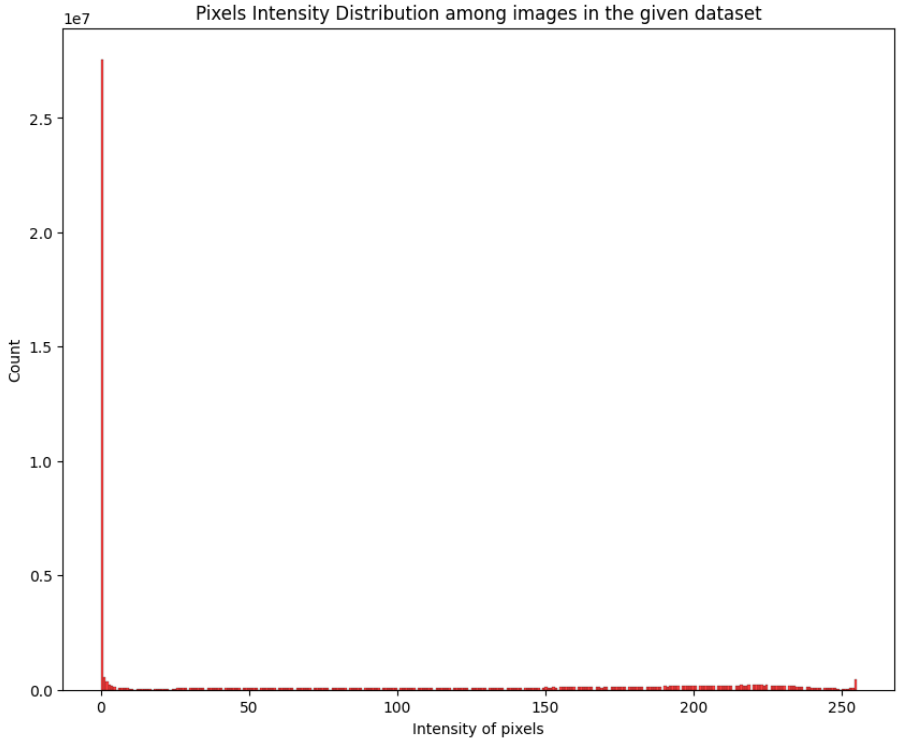


Figure 3: Pixel Intensity Distribution

1.3 Task to be performed

I have implemented a neural network from scratch in Python. I built a feed-forward neural network and implemented back propagation for training. For updating the weights and bias during the training, I used Gradient descent algorithm and at last computed the loss and accuracy of the model and finally plotted the confusion matrix.

1.4 Network Architecture

1. The network contains 3 hidden layers, excluding input and output layers as :-
   - Input Layer = 784 neurons
   - Output Layer = 10 neurons (10 classes in the datatset)
   - Hidden Layer 1 = 128 neurons
   - Hidden Layer 2 = 64 neurons
   - Hidden Layer 3 = 32 neurons
2. Since my Roll No. is M21MA210, the last three digits are 210. Therefore, I used **seed value** = 210. and as per given in the question, set the **bias**= 1.
3. For training purposes, I have split the data in three configurations.
   - 70:30
   - 80:20
   - 90:10
4. My year of admission is 2021, so I used batch size = 21.
5. For activation of the hidden layers, I used *sigmoid* function and for output layer, I used $softmax$ function.
   - $\sigma(x) = \frac{1}{1+e^{-x}}$
   - $\text{Softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$
6. I used Loss function as *Cross Entropy* $= -\sum_{i=1}^{n} y_i \cdot \log(p_i)$

2 Training of the Neural Network

I have shuffled the data points in the given dataset for training purposes. Here is a sample of some shuffled data points.

```
Here are the indices of some of the shuffled datapoints:-

 [28266 60114 44874 35705 61936 38076 69824  1871 12318 37472 57606 32809
64348  3514 21357 11538 59620 17046 39151  9784 47666 18843 38487 62242
24301 42870 38849 43800 38517 45859 61498 38815 40752  2501 55494  1231
40402 19359  8558 42114 31852 32472 17453 17395 12824 20462 60399 58656
49786 13176 25699 48625 32428 43833 40783 65409 57076 33574 59706 59723
 6461 60144 57410 15723 47121 17804  8729 22523 57495 33895 42084  1094
 9629 43830 46300 25137 24728 69616 33001 67248  6082 23316 19598 20741
51608 63410 25983 38657 36623 19490 10885 56703 27652 26379 22473 44473
58436 64384 68303 30633 33245 66084 25277 26519 68546 37379 14599 41829
58833 50351 33688 28971 66343 14481  6270 65796  6982 49524 38056 21820
16062 15914 63003 38605  9418  5461 13223 69205 27067 17175 14356 44916
12797 40940 39321  9497 49414 27632  3890 12413 65089 49207 51757 67851
46850 22852 48345 65065 23235 21026 36886 41557 40293 16831 67434 13016
45098 54721 55722 14961 17616 16664  7815 54395 68995 32619 59926 38776
 4654 61620 44603 56978 17326 56167 42678 19349 48840 19444 32698 27236
61355 64729  8571  5686  9626 53847 62218  9498 33501 53099 43194 42736
24022 69998 59457  5341 28445 11914  8409 55441  2286 53574 23732 47297
 7983 55759 36738 12003 15977  4729 15077 13126 38573 50057 28684 28372
 8894 36970 28980 63713   872 59328 48887 60495 10163 50658 22663 28904
67837 29458  2535 55543 24508 29489 18127 22047 25048 50048 24591 58530
58698 24316 62059 33379 41849 56061 14128 61510  9872 63785 31110   121
21837 58424 38117 69423 43950 32181  1199 66134  9179 64775 33140  3429
23650 65843 35176 40859 66770  1545  9736 35695 31569 55348 58398  6068
44072  8206 13327 32764 23423 49089 10395 62448 17760 69278 13109 42727
 9352 16878 62826 16232  7426  7520  5687 21614 14569  7310  7102 19712]
```

Figure 4: Some of the shuffled data points

2.1 For the train-test split of 70 : 30

- The size of the dataset after splitting is as follows:-

```
Enter the split ratio, (e.g if split is 70:30, give input as ~~ 0.7 ~~) :- 0.7

The size of the training dataset is:- (49000, 784)
and that of testing dataset is (21000, 784)
```

Figure 5: Split Ratio 70 : 30

- Loss and Accuracy of the model during the 25 epochs for this train-test split

```
<ipython-input-86-0256e54624b8>:3: RuntimeWarning: overflow encountered in exp
  return 1/(1+np.exp(-x))
Epoch:1, Loss:- 29.1602 and Accuracy:- 55.32%
Epoch:2, Loss:- 22.9342 and Accuracy:- 66.41%
Epoch:3, Loss:- 20.1010 and Accuracy:- 69.60%
Epoch:4, Loss:- 18.6660 and Accuracy:- 72.19%
Epoch:5, Loss:- 17.9168 and Accuracy:- 73.51%
Epoch:6, Loss:- 17.5897 and Accuracy:- 74.28%
Epoch:7, Loss:- 17.4485 and Accuracy:- 74.37%
Epoch:8, Loss:- 17.2877 and Accuracy:- 74.53%
Epoch:9, Loss:- 17.1202 and Accuracy:- 74.73%
Epoch:10, Loss:- 16.9009 and Accuracy:- 74.90%
Epoch:11, Loss:- 16.5834 and Accuracy:- 75.39%
Epoch:12, Loss:- 16.2549 and Accuracy:- 75.70%
Epoch:13, Loss:- 15.9065 and Accuracy:- 76.01%
Epoch:14, Loss:- 15.6100 and Accuracy:- 76.33%
Epoch:15, Loss:- 15.3324 and Accuracy:- 76.76%
Epoch:16, Loss:- 15.0756 and Accuracy:- 77.21%
Epoch:17, Loss:- 14.8293 and Accuracy:- 77.56%
Epoch:18, Loss:- 14.6063 and Accuracy:- 77.88%
Epoch:19, Loss:- 14.4068 and Accuracy:- 78.02%
Epoch:20, Loss:- 14.2255 and Accuracy:- 78.28%
Epoch:21, Loss:- 14.0414 and Accuracy:- 78.61%
Epoch:22, Loss:- 13.8887 and Accuracy:- 78.82%
Epoch:23, Loss:- 13.7499 and Accuracy:- 79.09%
Epoch:24, Loss:- 13.6662 and Accuracy:- 79.12%
Epoch:25, Loss:- 13.5851 and Accuracy:- 79.33%
```

Figure 6: Training Loss and Accuracy of the model

- Curve representing the loss during the training for this split ratio

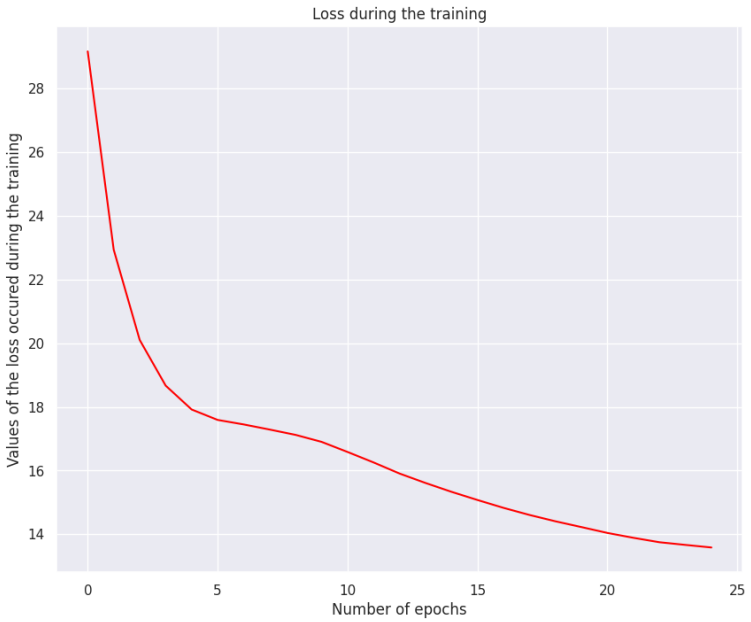


Figure 7: Loss Curve during the training

- Curve representing the accuracy of the model during the training for this split ratio

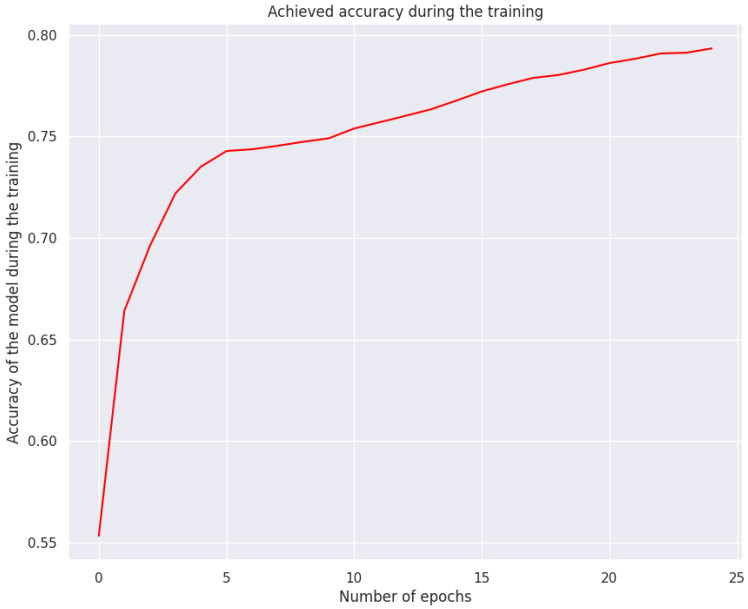


Figure 8: Accuracy Curve during the training

- Accuracy obtained by the model on the test data set is as follows:-

```
<ipython-input-86-0256e54624b8>:3: RuntimeWarning: overflow encountered in exp
  return 1/(1+np.exp(-x))
Accuracy on the test dataset : 78.65238%
```

Figure 9: Test Accuracy of the model

- Confusion Matrix obtained as follows:-

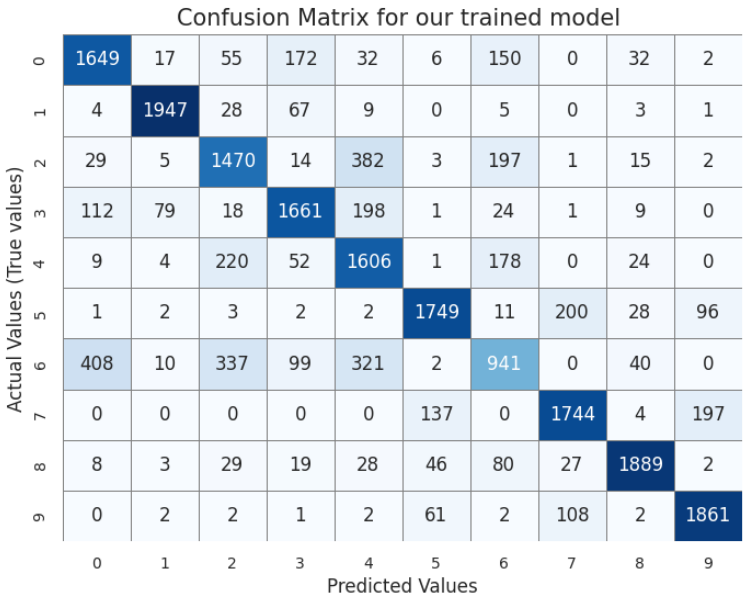


Figure 10: Confusion Matrix

2.2 For the train-test split of 80 : 20

- The size of the dataset after splitting is as follows:-

```
Enter the split ratio, (e.g if split is 70:30, give input as ~~ 0.7 ~~) :- 0.8

The size of the training dataset is:- (56000, 784)
and that of testing dataset is (14000, 784)
```

Figure 11: Split Ratio 80 : 20

- Loss and Accuracy of the model during the 25 epochs for this train-test split

```
<ipython-input-114-0256e54624b8>:3: RuntimeWarning: overflow encountered in exp
  return 1/(1+np.exp(-x))
Epoch:1, Loss:- 28.6135 and Accuracy:- 56.49%
Epoch:2, Loss:- 22.1995 and Accuracy:- 67.06%
Epoch:3, Loss:- 19.5263 and Accuracy:- 70.48%
Epoch:4, Loss:- 18.2906 and Accuracy:- 72.91%
Epoch:5, Loss:- 17.7666 and Accuracy:- 73.91%
Epoch:6, Loss:- 17.5340 and Accuracy:- 74.34%
Epoch:7, Loss:- 17.3362 and Accuracy:- 74.60%
Epoch:8, Loss:- 17.0884 and Accuracy:- 74.88%
Epoch:9, Loss:- 16.8110 and Accuracy:- 75.04%
Epoch:10, Loss:- 16.4713 and Accuracy:- 75.51%
Epoch:11, Loss:- 16.1107 and Accuracy:- 75.78%
Epoch:12, Loss:- 15.7182 and Accuracy:- 76.37%
Epoch:13, Loss:- 15.3561 and Accuracy:- 76.81%
Epoch:14, Loss:- 15.0475 and Accuracy:- 77.32%
Epoch:15, Loss:- 14.7728 and Accuracy:- 77.68%
Epoch:16, Loss:- 14.5272 and Accuracy:- 77.96%
Epoch:17, Loss:- 14.3370 and Accuracy:- 78.18%
Epoch:18, Loss:- 14.1599 and Accuracy:- 78.35%
Epoch:19, Loss:- 14.0111 and Accuracy:- 78.54%
Epoch:20, Loss:- 13.8865 and Accuracy:- 78.67%
Epoch:21, Loss:- 13.7943 and Accuracy:- 78.72%
Epoch:22, Loss:- 13.7028 and Accuracy:- 78.93%
Epoch:23, Loss:- 13.6253 and Accuracy:- 79.13%
Epoch:24, Loss:- 13.5781 and Accuracy:- 79.17%
Epoch:25, Loss:- 13.5418 and Accuracy:- 79.12%
```

Figure 12: Training Loss and Accuracy of the model

- Curve representing the loss during the training for this split ratio

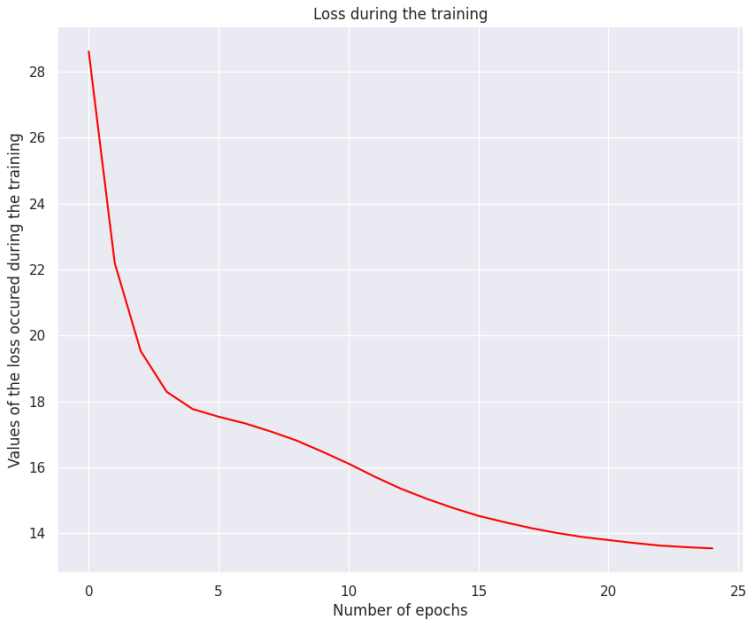


Figure 13: Loss Curve during the training

- Curve representing the accuracy of the model during the training for this split ratio

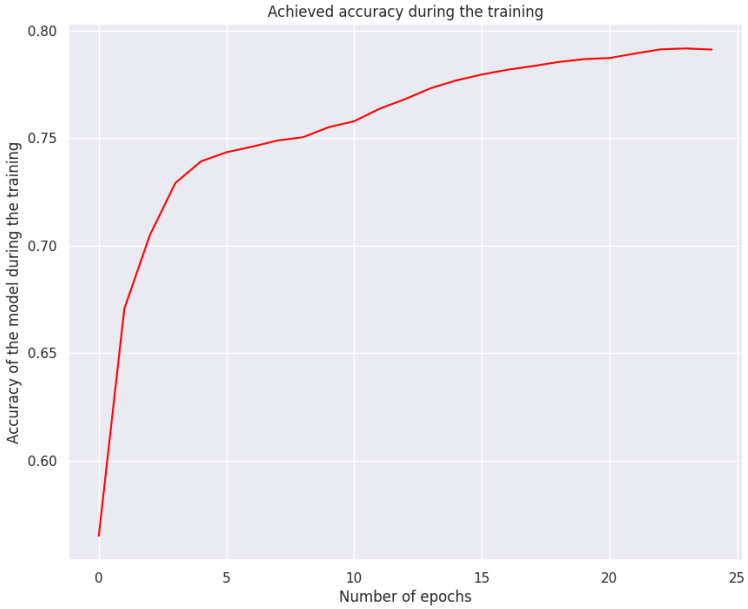


Figure 14: Accuracy Curve during the training

- Accuracy obtained by the model on the test data set is as follows:-

```
<ipython-input-114-0256e54624b8>:3: RuntimeWarning: overflow encountered in exp
  return 1/(1+np.exp(-x))
Accuracy on the test dataset : 78.82857%
```

Figure 15: Test Accuracy of the model

- Confusion Matrix obtained as follows:-

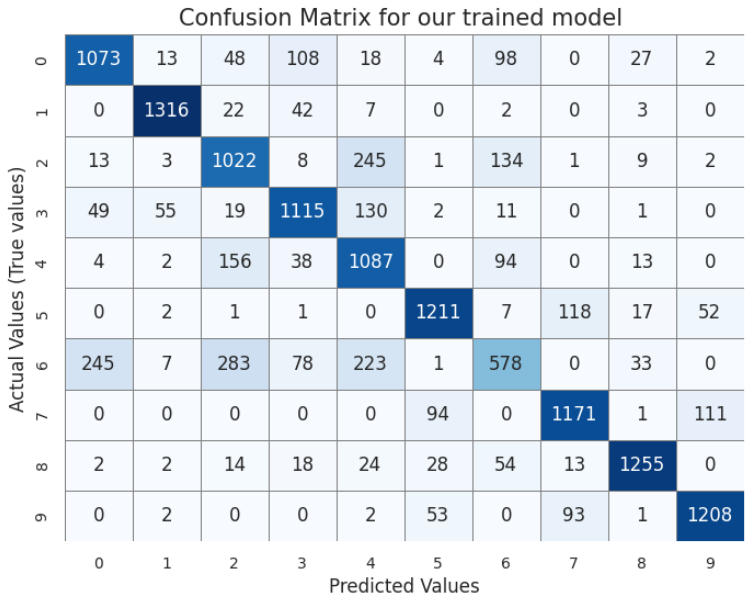


Figure 16: Confusion Matrix

2.3 For the train-test split of 90 : 10

- The size of the dataset after splitting is as follows:-

```
Enter the split ratio, (e.g if split is 70:30, give input as ~~ 0.7 ~~) :- 0.9

The size of the training dataset is:- (63000, 784)
and that of testing dataset is (7000, 784)
```

Figure 17: Split Ratio 90 : 10

- Loss and Accuracy of the model during the 25 epochs for this train-test split

```
<ipython-input-148-0256e54624b8>:3: RuntimeWarning: overflow encountered in exp
  return 1/(1+np.exp(-x))
Epoch:1, Loss:- 28.1362 and Accuracy:- 57.42%
Epoch:2, Loss:- 21.5492 and Accuracy:- 67.61%
Epoch:3, Loss:- 19.0563 and Accuracy:- 71.26%
Epoch:4, Loss:- 17.9738 and Accuracy:- 73.31%
Epoch:5, Loss:- 17.5856 and Accuracy:- 74.19%
Epoch:6, Loss:- 17.3476 and Accuracy:- 74.67%
Epoch:7, Loss:- 17.1307 and Accuracy:- 74.85%
Epoch:8, Loss:- 16.8306 and Accuracy:- 75.10%
Epoch:9, Loss:- 16.4584 and Accuracy:- 75.62%
Epoch:10, Loss:- 16.0669 and Accuracy:- 75.91%
Epoch:11, Loss:- 15.6768 and Accuracy:- 76.29%
Epoch:12, Loss:- 15.2909 and Accuracy:- 76.80%
Epoch:13, Loss:- 14.9457 and Accuracy:- 77.40%
Epoch:14, Loss:- 14.6669 and Accuracy:- 77.93%
Epoch:15, Loss:- 14.4241 and Accuracy:- 78.23%
Epoch:16, Loss:- 14.2053 and Accuracy:- 78.55%
Epoch:17, Loss:- 14.0412 and Accuracy:- 78.66%
Epoch:18, Loss:- 13.9233 and Accuracy:- 78.75%
Epoch:19, Loss:- 13.7873 and Accuracy:- 79.05%
Epoch:20, Loss:- 13.7000 and Accuracy:- 79.09%
Epoch:21, Loss:- 13.6387 and Accuracy:- 79.20%
Epoch:22, Loss:- 13.5840 and Accuracy:- 79.27%
Epoch:23, Loss:- 13.5371 and Accuracy:- 79.36%
Epoch:24, Loss:- 13.5049 and Accuracy:- 79.42%
Epoch:25, Loss:- 13.4767 and Accuracy:- 79.42%
```

Figure 18: Training Loss and Accuracy of the model

- Curve representing the loss during the training for this split ratio

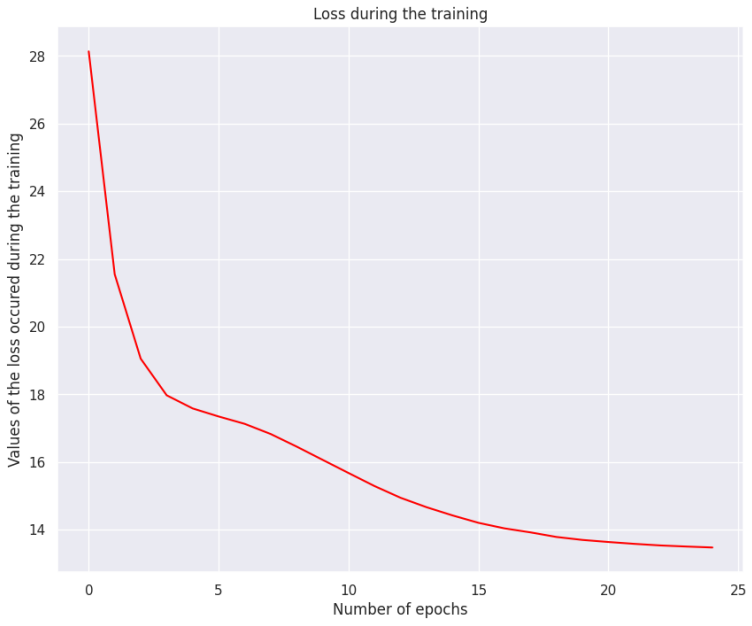


Figure 19: Loss Curve during the training

- Curve representing the accuracy of the model during the training for this split ratio

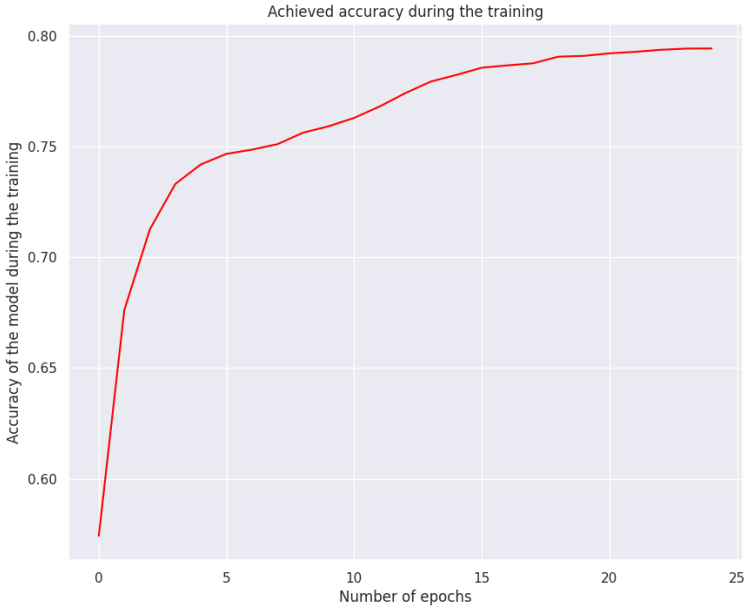


Figure 20: Accuracy Curve during the training

- Accuracy obtained by the model on the test data set is as follows:-

```
<ipython-input-148-0256e54624b8>:3: RuntimeWarning: overflow encountered in exp
  return 1/(1+np.exp(-x))
 Accuracy on the test dataset : 79.37143%
```

Figure 21: Test Accuracy of the model

- Confusion Matrix obtained as follows:-

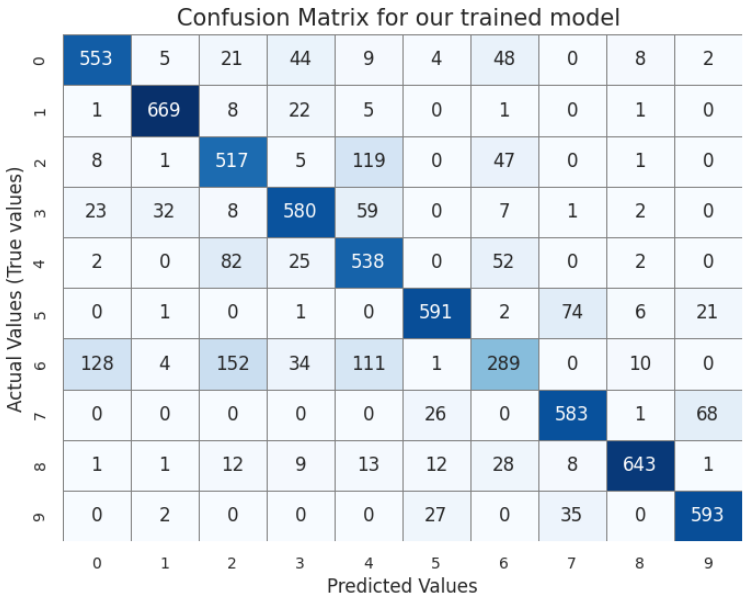


Figure 22: Confusion Matrix

2.4 Report total trainable and non-trainable parameters

Please refer to the attached text file, which includes the value of all the trainable and non-trainable parameters found after the model's training.

3 Resources Used

- https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/
- https://www.ibm.com/topics/neural-networks
- https://en.wikipedia.org/wiki/Neural_network
- https://scikit-learn.org/stable/index.html

********** THE END **********