In-Depth Analysis of Convolutional Neural Networks with PyTorch Using MNIST.

Colab link : https://colab.research.google.com/drive/1iEdiWaXwYXIZm1TcYQCvdFfBMRnGdpo_?usp=sharing

Experiment 1:

Implement a Convolutional Neural Network to solve a 10-class classification problem.

A. Exploring Dataset:

The MNIST dataset, consisting of 60k images for training and 10k images for testing, is utilized for training the CNN.

B. Network Architecture:

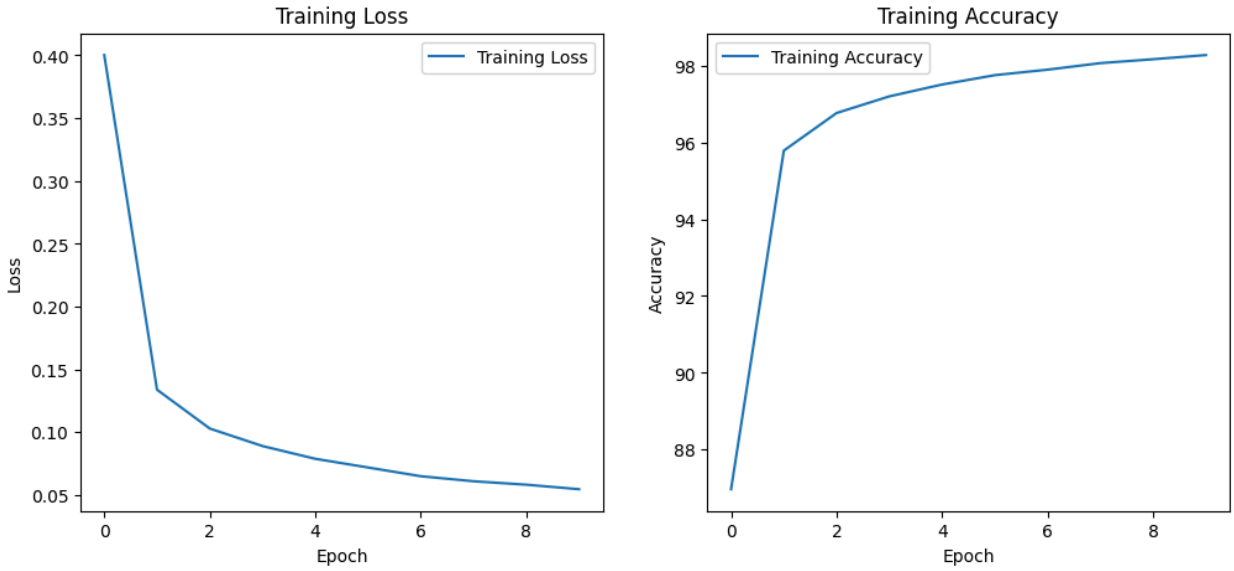
The implemented CNN comprises three convolutional layers and an output layer. The convolutional layers have different kernel sizes, pooling strategies, and output channels. ReLU is used as the activation function for convolution layers, and softmax is applied to the output layer.

NOTE: Since we are using torch.nn 's cross entropy loss which automatically applies softmax we have not applied softmax in our architecture, as applying it two times would balance out the probabilities.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 16, 28, 28]             800
         MaxPool2d-2          [-1, 16, 14, 14]               0
            Conv2d-3           [-1, 8, 14, 14]           3,208
         MaxPool2d-4             [-1, 8, 7, 7]               0
            Conv2d-5             [-1, 4, 4, 4]             292
         AvgPool2d-6             [-1, 4, 2, 2]               0
            Linear-7                  [-1, 10]             170
================================================================
Total params: 4,470
Trainable params: 4,470
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.14
Params size (MB): 0.02
Estimated Total Size (MB): 0.16
----------------------------------------------------------------
```

C. Training:

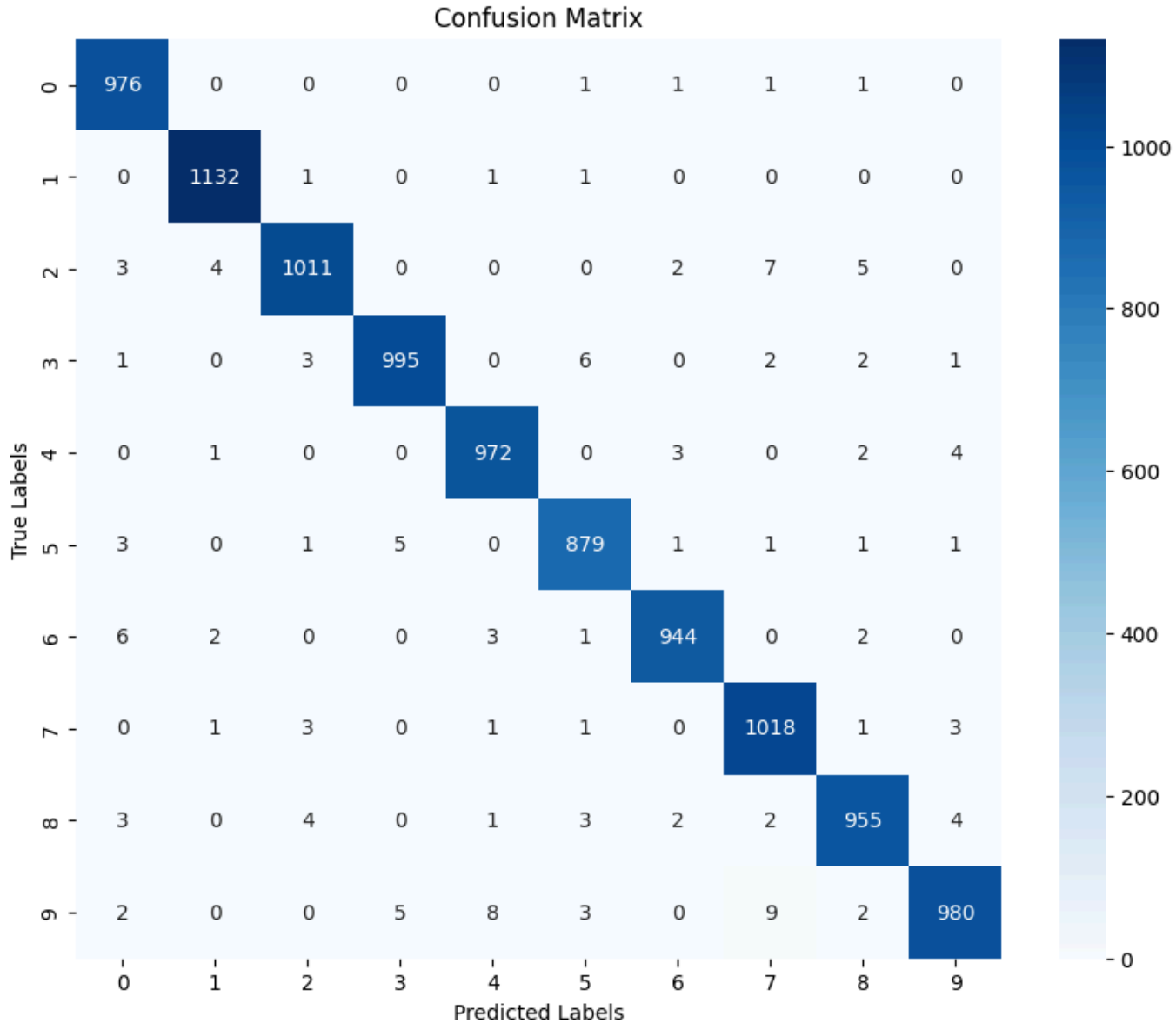
We have normalized our data with mean 0 and standard deviation 1.
The CNN is trained for 10 epochs using the Adam optimizer with a cross-entropy loss function. The batch size is 20 (as roll no. is M23CSE**013**)



- The Training Accuracy of the model was found to be **98.27%.**
- The Testing Accuracy of the model was found to be **98.27%.**

D. Confusion Matrix:

A confusion matrix was generated to understand the classification performance across different classes.

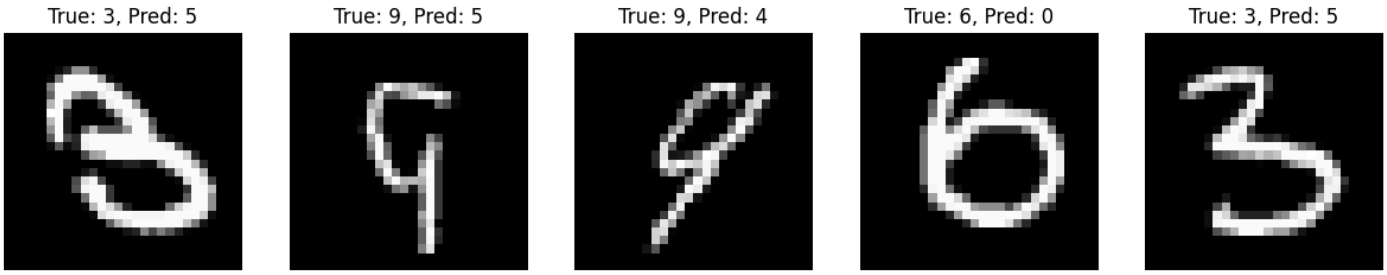


E. Total Trainable and Non-Trainable parameters:

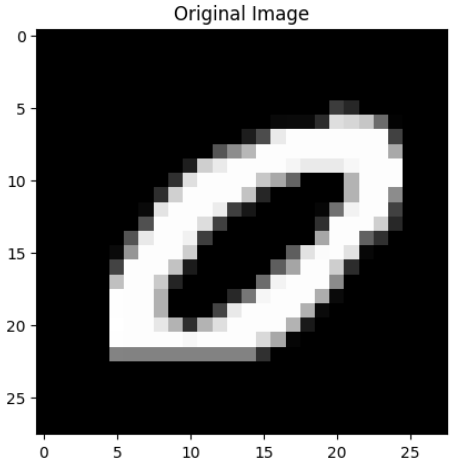
- Total parameters for the architecture were found to be: 4,470
- Trainable parameters for the architecture were found to be: 4,470
- Non-trainable parameters for the architecture were found to be: 0

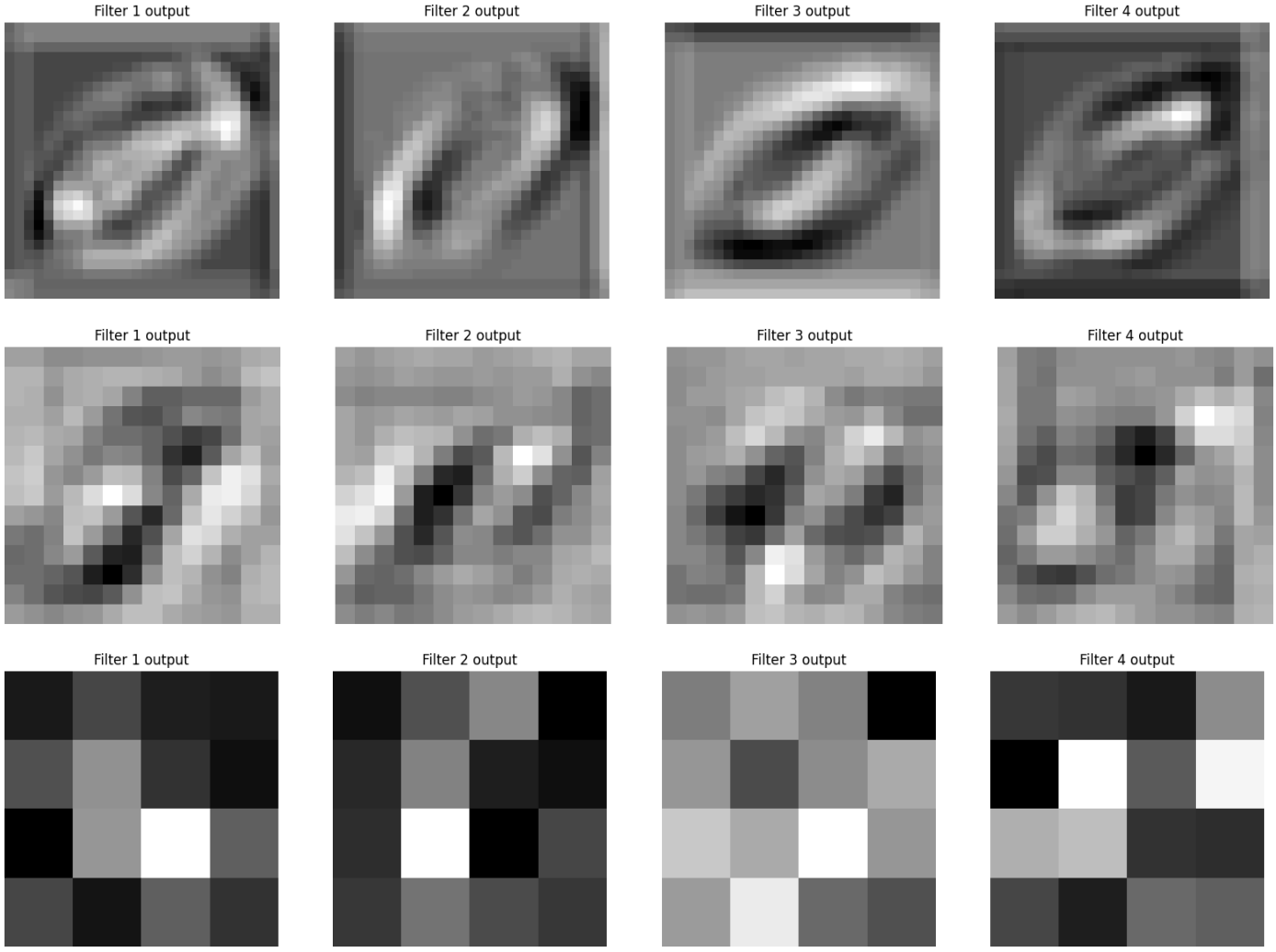
F. Visualizations:

1. Error Analysis:

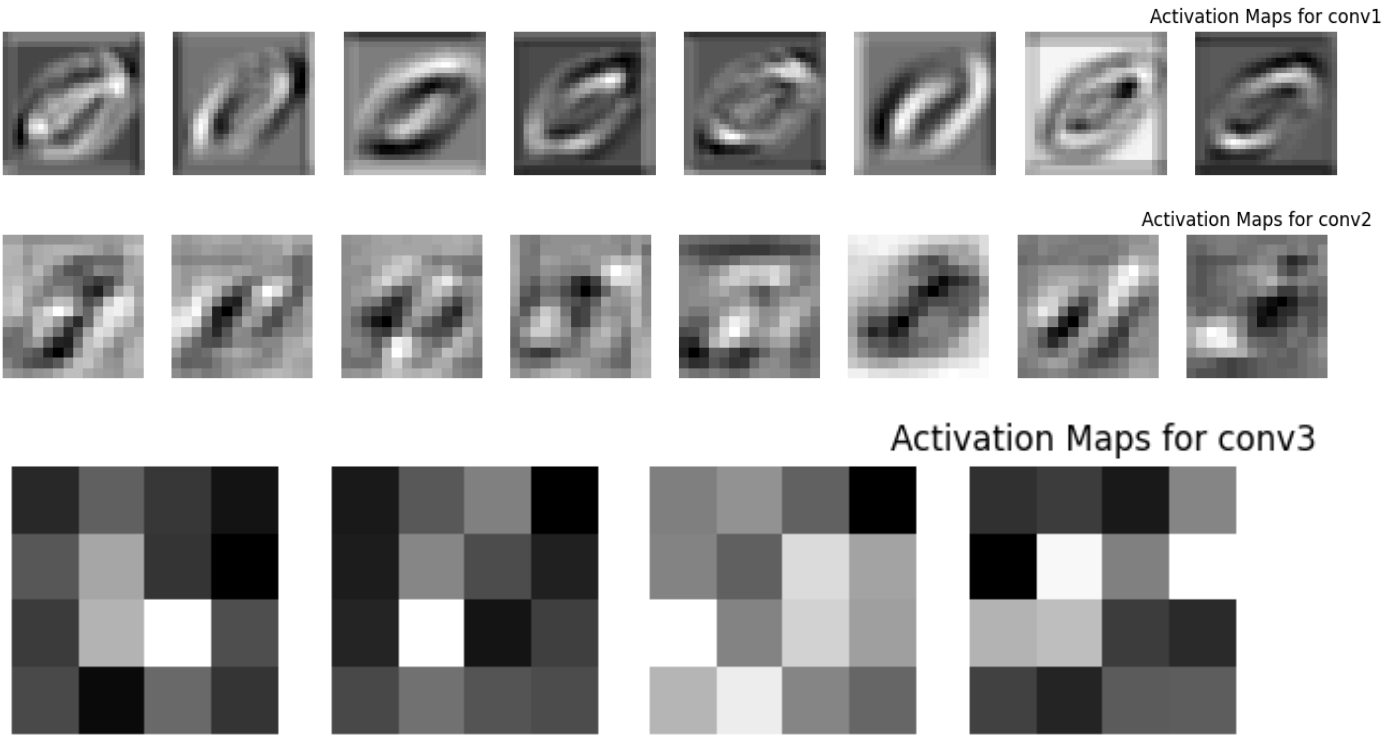


2. Visualize Each Layers Output:

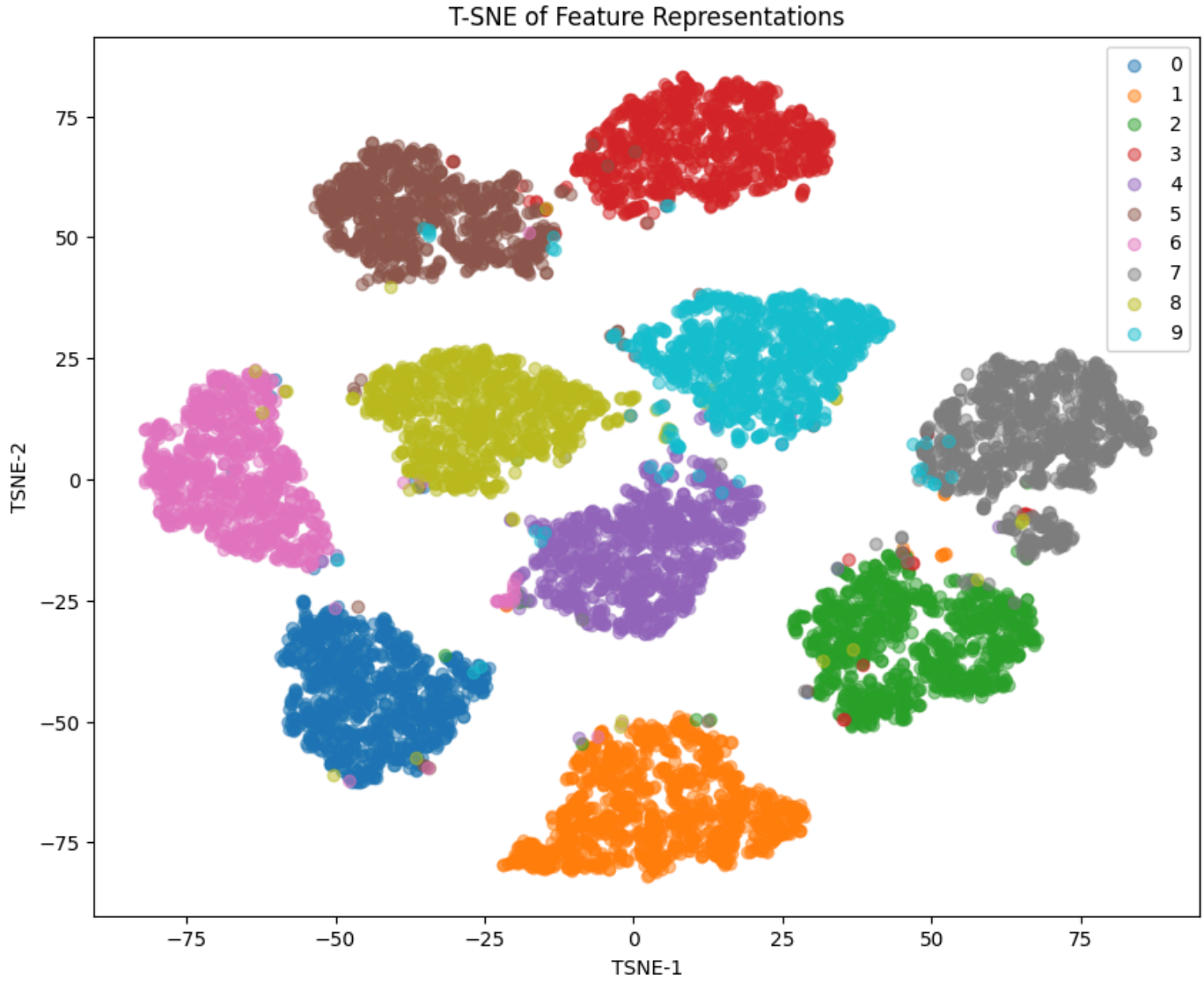



Filter 1 output
Filter 2 output
Filter 3 output
Filter 4 output
Filter 1 output
Filter 2 output
Filter 3 output
Filter 4 output
Filter 1 output
Filter 2 output
Filter 3 output
Filter 4 output

3. Visualization for Activation Maps:



4. T-SNE Visualization of Feature Representations



Experiment 2:

Implement the same Convolutional Neural Network to solve a 4-class(by combining digit images) classification problem.

A. Network Architecture:

The main Architecture remains same this architecture inherits the previous architecture with the modification of output mapping to 4 classes:

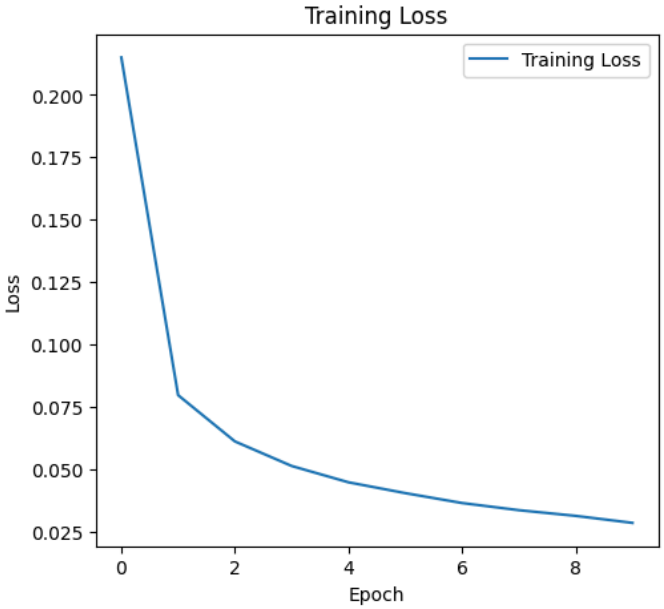
- Class 0: {0, 6}
- Class 1: {1, 7}
- Class 2: {2, 3, 8, 5}
- Class 3: {4, 9}

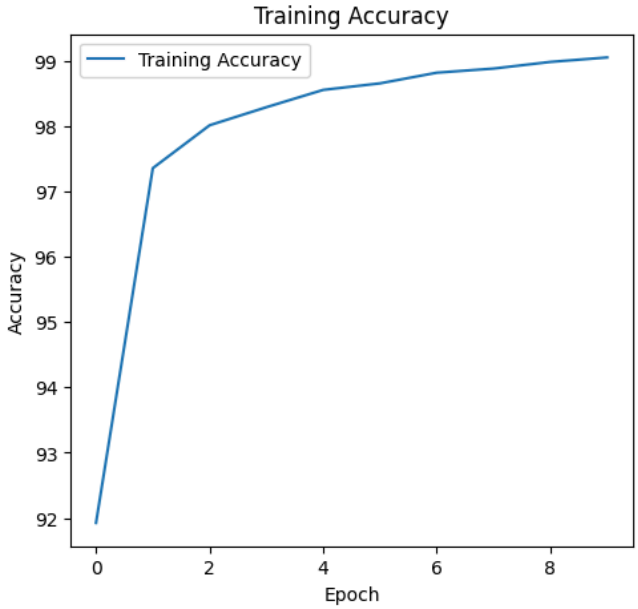
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 16, 28, 28]             800
         MaxPool2d-2          [-1, 16, 14, 14]               0
            Conv2d-3           [-1, 8, 14, 14]           3,208
         MaxPool2d-4             [-1, 8, 7, 7]               0
            Conv2d-5             [-1, 4, 4, 4]             292
         AvgPool2d-6             [-1, 4, 2, 2]               0
            Linear-7                   [-1, 4]              68
================================================================
Total params: 4,368
Trainable params: 4,368
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.14
Params size (MB): 0.02
Estimated Total Size (MB): 0.15
----------------------------------------------------------------
```

NOTE: Since we are using torch.nn 's cross entropy loss which automatically applies softmax we have not applied softmax in our architecture, as applying it two times would balance out the probabilities.

B. Training:

The CNN is trained for 10 epochs using the Adam optimizer with a cross-entropy loss function. The batch size is 20 (as roll no. is M23CSE**013**)

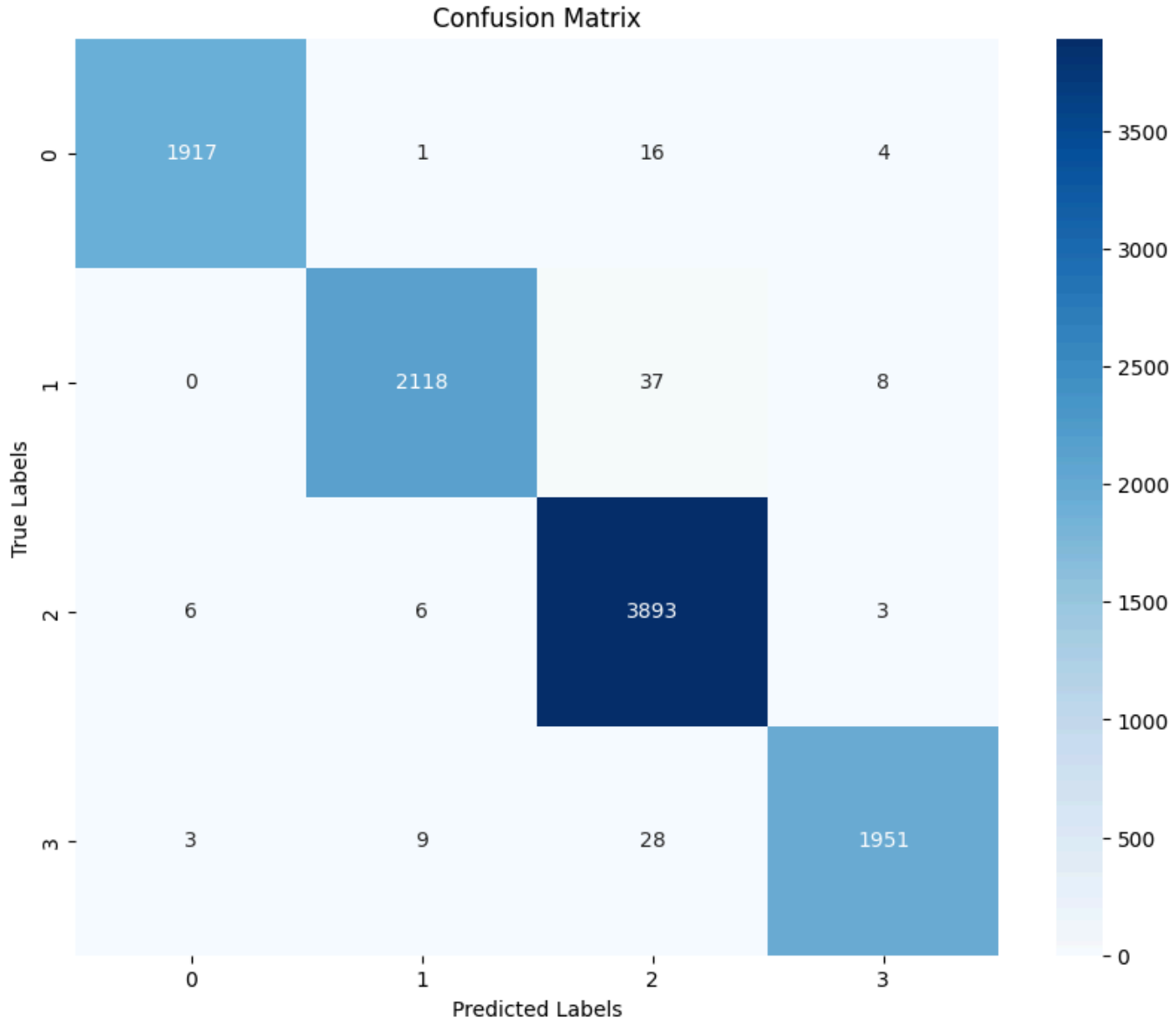




- The Training Accuracy of the model was found to be **99.05%.**
- The Testing Accuracy of the model was found to be **98.79%**
- **Slightly better than 10 class classification, likely because grouping of visually similar looking digits .**

C. Confusion Matrix:

A confusion matrix was generated to understand the classification performance across different classes.

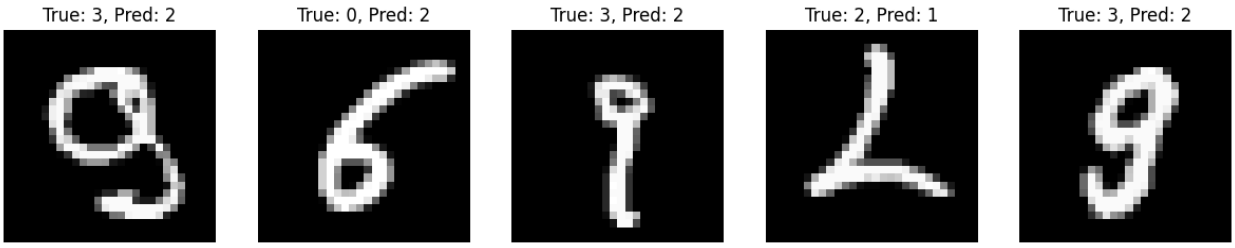


D. Total Trainable and Non-Trainable parameters:

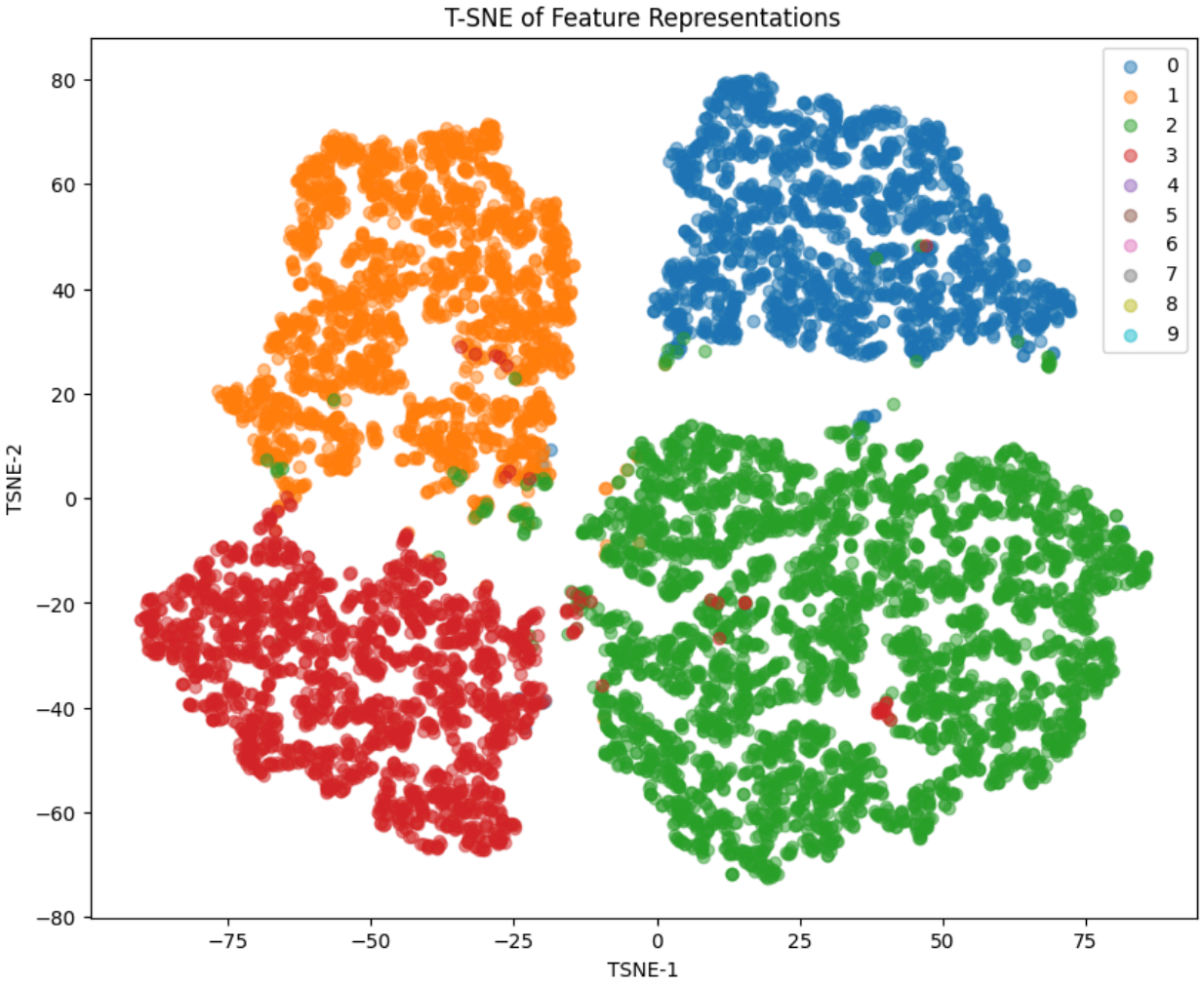
- Total parameters for the architecture were found to be: 4,470
- Trainable parameters for the architecture were found to be: 4,470
- Non-trainable parameters for the architecture were found to be: 0

E. Visualizations:

5. Error Analysis:



6. T-SNE Visualization of Feature Representations



Bonus Section:

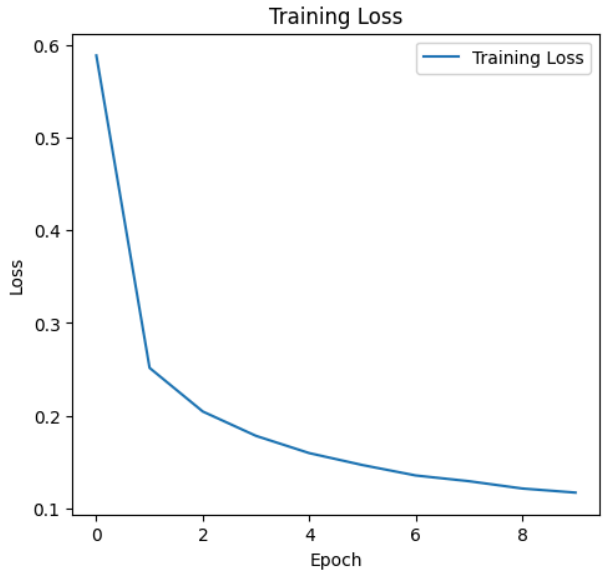
1. Data Augmentation:

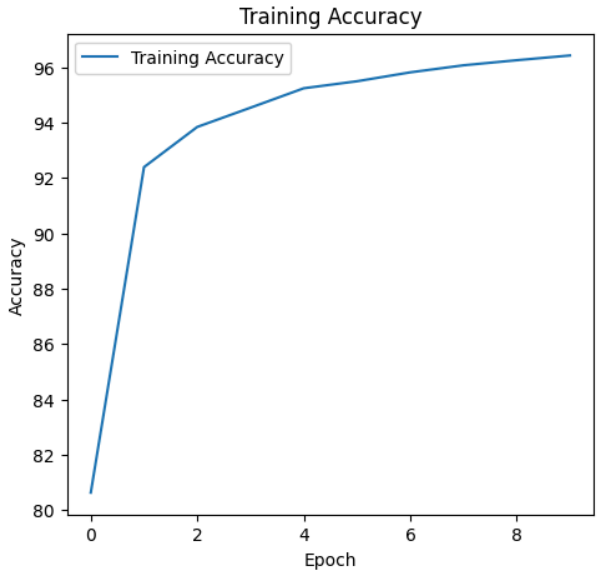
Data augmentation involves applying various transformations to the training images, creating new training samples. This helps improve the model's generalization by exposing it to a more diverse set of inputs.
We have augmented the image data(from the test dataset) in the following manner:

- Rotation by 10 degrees.
- Translating the image 0.1 and 0.1(maximum by 10%)
- Reducing brightness by 0.5.

The Following observations were made while doing 10 epochs:

- The Training Accuracy of the model was found to be **96.42%.**
- The Testing Accuracy of the model was found to be **98.37%**
- The model is generalizing since we are making transformations in the training dataset.





2. Dropout with Data Augmentation:

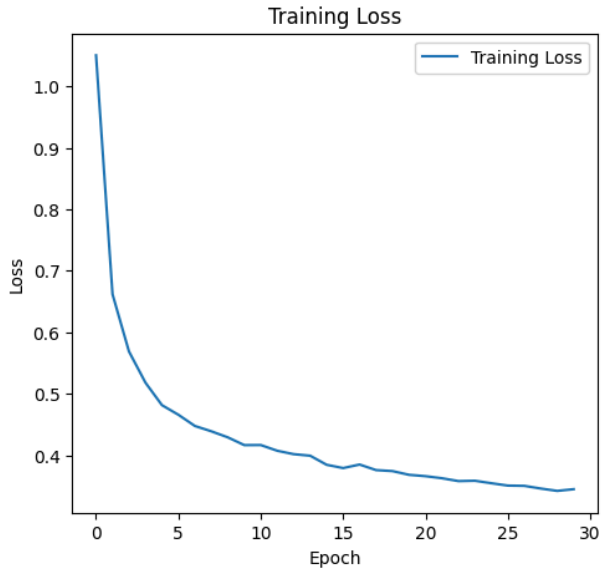
Dropout is a regularization technique where random neurons are dropped during training. This prevents overfitting by making the network more robust and reducing reliance on specific neurons.

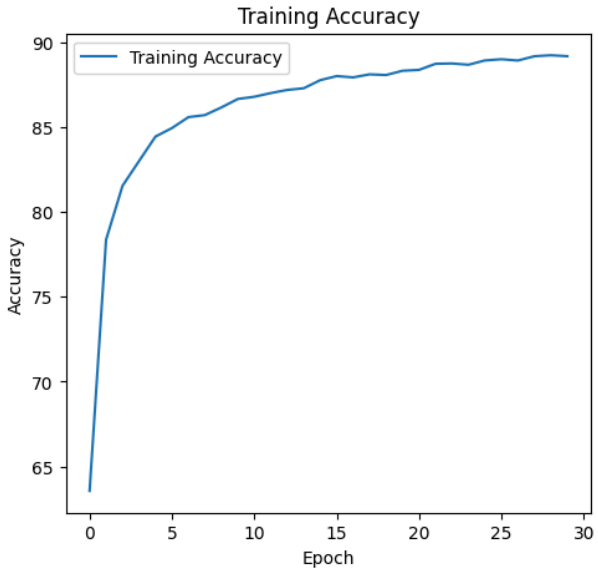
We have used the technique of dropout with the hope it will generalize our dataset along with data augmentation in the following manner:

- Dropout rate = 0.3
- Rotation by 10 degrees.
- Translating the image 0.1 and 0.1(maximum by 10%).
- Reducing brightness by 0.5.

The Following observations were made while doing 30 epochs (because of using dropout technique):

- The Training Accuracy of the model was found to be **89.18%.**
- The Testing Accuracy of the model was found to be **97.82%**
- Our model is very much generalized and we can feed a wide variety of data and it is likely to give a very better accuracy with comparison to our original architecture.



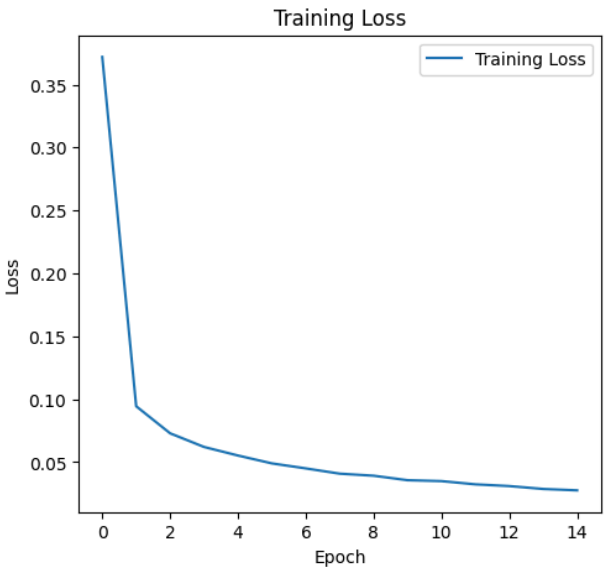


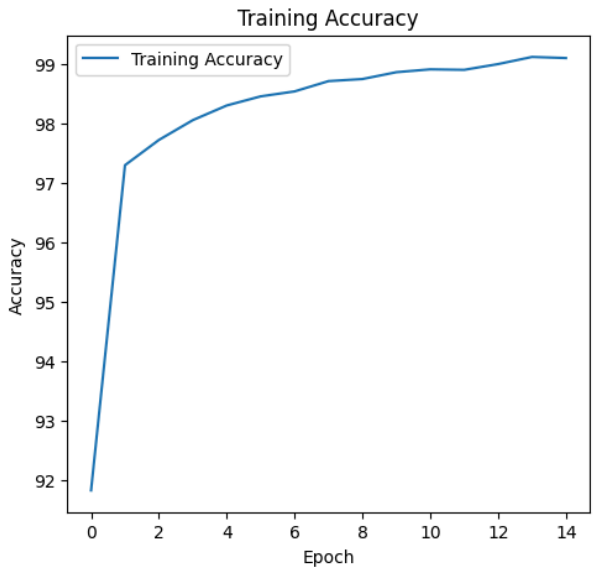
3. Batch Normalization:

- Batch normalization normalizes the input of each layer, reducing internal covariate shift. This stabilizes and accelerates training, leading to better generalization.

The Following observations were made while doing 15 epochs (because of using dropout technique):

- The Training Accuracy of the model was found to be **99.10%.**
- The Testing Accuracy of the model was found to be **98.66%**


Training Loss
Training Loss
Loss
Epoch
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
2
4
6
8
10
12
14


Training Accuracy
Training Accuracy
Accuracy
Epoch
99
98
97
96
95
94
93
92
0
2
4
6
8
10
12
14