ASSIGNMENT-01
Amresh kumar
M23CSA004

Collab file link: M23CSA004.ipynb - Colaboratory (google.com)

1. Introduction:

Convolutional Neural Networks (CNNs) are widely used for image classification tasks due to their ability to automatically learn hierarchical features from input images. In this report, we present the implementation and evaluation of Convolutional Neural Networks (CNNs) for digit classification using PyTorch. Two experiments are conducted:

Experiment 1: A CNN model is implemented to solve a 10-class classification problem using the MNIST dataset.
Experiment 2: The MNIST dataset is modified to create four classes, and the CNN model from Experiment 1 is used to solve a 4-class classification problem.

2. Methodology:

2.1 Data Loading and Preprocessing:

The MNIST dataset consists of 60,000 training images and 10,000 testing images of handwritten digits.
Data is loaded using the idx2numpy library and normalized to a range of [0, 1].
There is no need for a Train-test split as separate training and testing data is provided.

2.2 Model Architecture:

The CNN architecture consists of three convolutional layers followed by a fully connected layer (output layer).
Each convolutional layer is followed by a max-pooling or average-pooling layer to reduce spatial dimensions.

Convolution Layer 1: Kernel Size: 7x7,Maxpool;Stride =1;Padding =3; output channels =16
Convolution Layer 2: Kernel Size: 5x5,Maxpool;Stride =1;Padding =2; output channels =8
Convolution Layer 3: Kernel Size: 3x3,Average Pool;Stride =2;Padding =1; output channels =4
Output size: Number of classes (**= 10 for Experiment 1**) and
Number of classes (**= 4 for Experiment 2**)

Input Dimension: The input dimension for Both Experiments are 784, which corresponds to images reshaped into a size of 28x28 pixels.

ReLU activation function is used for convolutional layers and softmax for the output layer.
Zero padding is applied to preserve input image dimensions.
The model is implemented using the PyTorch framework.

2.3 Training:

The model is trained for 10 epochs with a batch size of 32.
Adam optimizer and cross-entropy loss function are used for training.
Training loss and accuracy are printed and plotted per epoch to evaluate model performance.

2.4 Evaluation:

Test accuracy is calculated at epochs 1, 6, and 10 to monitor generalization.
Confusion matrix is generated for the test set to evaluate model performance across different classes.
Total trainable and non-trainable parameters are reported to understand model complexity.

3. Results:

3.1 Training Dynamics:

Training loss decreases steadily over epochs for both Experiments, indicating the model is learning from the training data.
Training accuracy increases over epochs, showing improvement in classification performance on the training set.

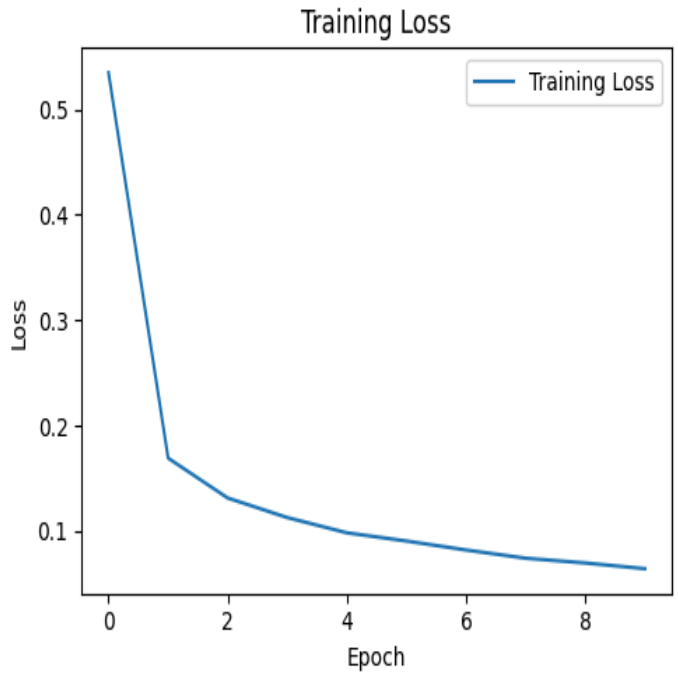
3.2 Test Accuracy:

Test accuracy improves with epochs, indicating the model's ability to generalize to unseen data.

For Experiment - 01

Epoch 1/10 => Loss: 0.5346, Accuracy: 0.8304
Epoch 2/10 => Loss: 0.1692, Accuracy: 0.9492
Epoch 3/10 => Loss: 0.1313, Accuracy: 0.9599
Epoch 4/10 => Loss: 0.1128, Accuracy: 0.9656
Epoch 5/10 => Loss: 0.0983, Accuracy: 0.9702
Epoch 6/10 => Loss: 0.0906, Accuracy: 0.9718
Epoch 7/10 => Loss: 0.0820, Accuracy: 0.9747
Epoch 8/10 => Loss: 0.0743, Accuracy: 0.9776
Epoch 9/10 => Loss: 0.0698, Accuracy: 0.9783
Epoch 10/10 => Loss: 0.0643, Accuracy: 0.9801(This is final train accuracy.)
Final Training loss is : 0.0643

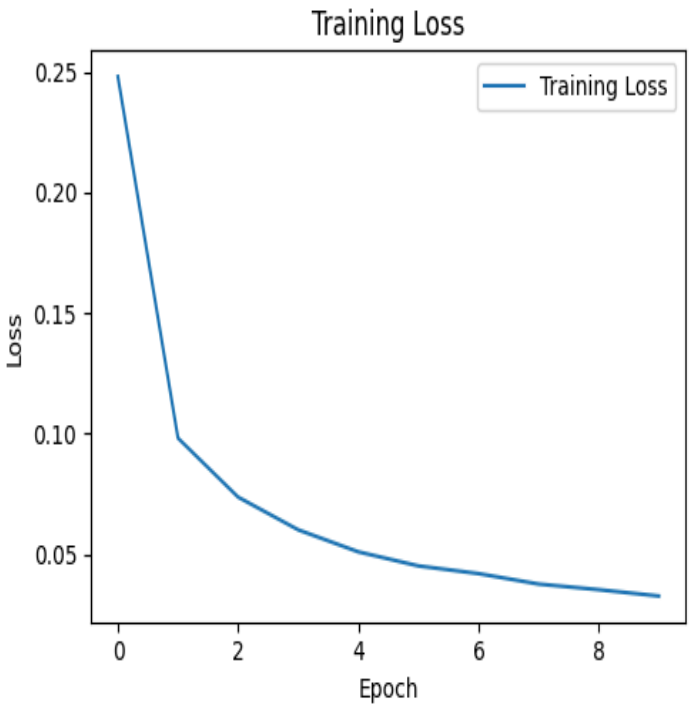
Test Accuracy at Epoch 1: 0.9410
Test Accuracy at Epoch 6: 0.9738
Test Accuracy at Epoch 10: 0.9801(This is final test accuracy)

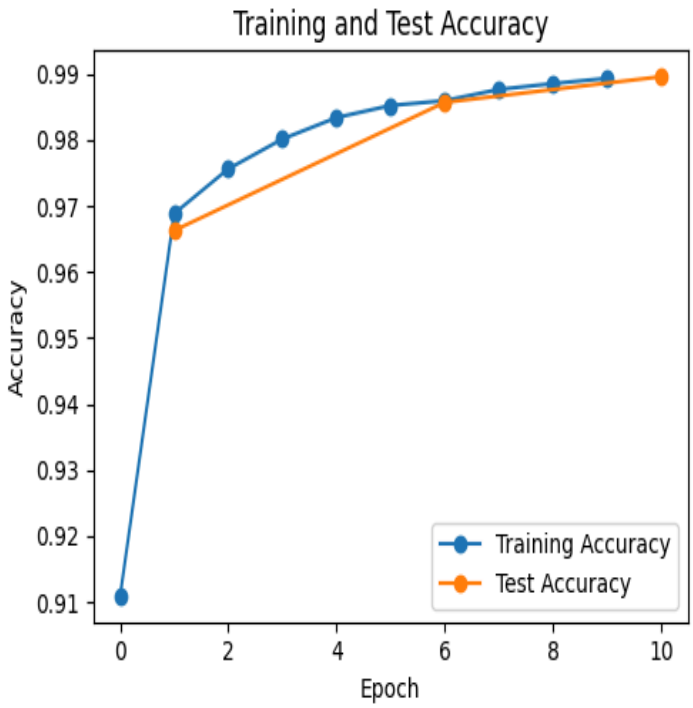


For Experiments - 02

Epoch 1/10 => Loss: 0.2480, Accuracy: 0.9109
Epoch 2/10 => Loss: 0.0981, Accuracy: 0.9689
Epoch 3/10 => Loss: 0.0736, Accuracy: 0.9756
Epoch 4/10 => Loss: 0.0601, Accuracy: 0.9801
Epoch 5/10 => Loss: 0.0509, Accuracy: 0.9834
Epoch 6/10 => Loss: 0.0451, Accuracy: 0.9852
Epoch 7/10 => Loss: 0.0419, Accuracy: 0.9860
Epoch 8/10 => Loss: 0.0376, Accuracy: 0.9877
Epoch 9/10 => Loss: 0.0352, Accuracy: 0.9886
Epoch 10/10 => Loss: 0.0326, Accuracy: 0.9893(This is final train accuracy.)
Final Training loss is :0.0326

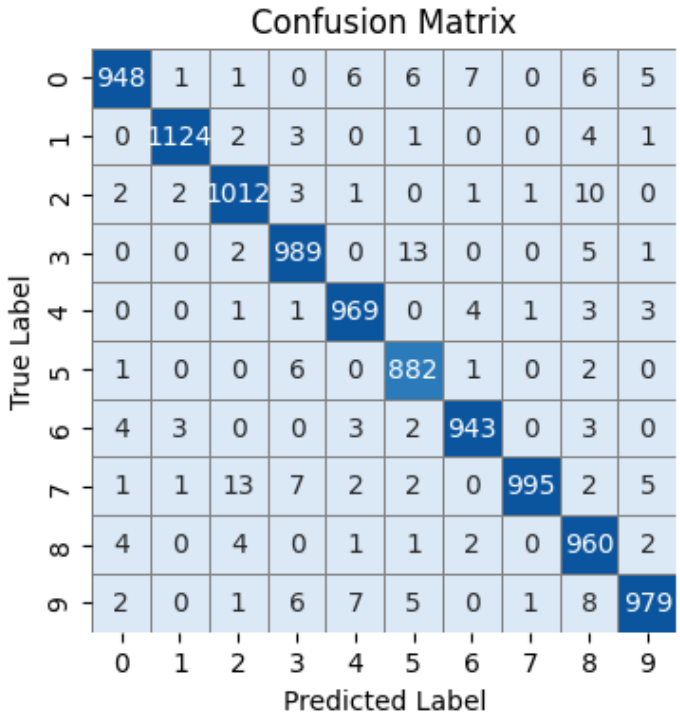
Test Accuracy at Epoch 1: 0.9663
Test Accuracy at Epoch 6: 0.9857
Test Accuracy at Epoch 10: 0.9896(This is final test accuracy)





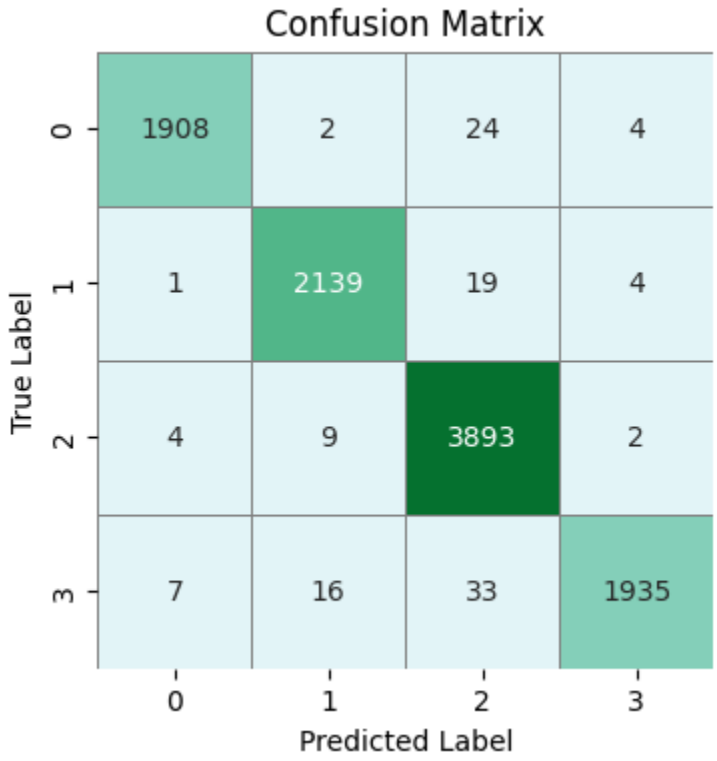
3.3 Confusion Matrix:

For Experiment -1



Confusion matrix provides insights into the model's performance for each class.
Diagonal elements represent correctly classified instances, while off-diagonal elements represent misclassifications.
It provides mapping between True class and predicted class.

For Experiment-2



3.4 Total Trainable and Non-Trainable Parameters:

Total parameters: 4470
Trainable parameters: 4470
Non-trainable parameters: 0

4. Observations:

The model effectively learns to classify digits over epochs as evidenced by decreasing loss and increasing accuracy.
Test accuracy improves with epochs, indicating good generalization of the model.
Some classes may have higher misclassification rates than others, which can be further analyzed to improve model performance.

Understanding the number of trainable and non-trainable parameters helps in model optimization and resource management.

5. Conclusion:

The implemented CNN model demonstrates effective learning and generalization on the MNIST dataset for digit classification. Further analysis of the confusion matrix and parameter optimization can enhance the model's performance. The provided code and results serve as a foundation for building and evaluating CNN models for similar classification tasks.

6. References:

a. Training a Classifier — PyTorch Tutorials 2.2.0+cu121 documentation
b. Convolution Visualizer (ezyang.github.io)
c. https://stackoverflow.com/questions/56515411/how-to-fix-valueerror-expected-input-batch-size-1-to-match-target-batch-size
d. Organize your life and...work with monday.com - the customizable work management platform (youtube.com)