Deep Learning

Assignment 1

Sireejaa Uppal (M23CSE023)

25th January, 2024

Experiment 1: 10-Class Classification

Dataset

The MNIST dataset containing handwritten digits was utilized, with 60k images for training and 10k images for testing.

Network Architecture

- Convolution Layer 1: Kernel Size=7x7, Maxpool, Stride=1, Output Channels=16
- Convolution Layer 2: Kernel Size=5x5, Maxpool, Stride=1, Output Channels=8
- Convolution Layer 3: Kernel Size=3x3, Avg Pooling, Stride=2, Output Channels=4
- Output Layer: Softmax activation, Output size=10 (number of classes)

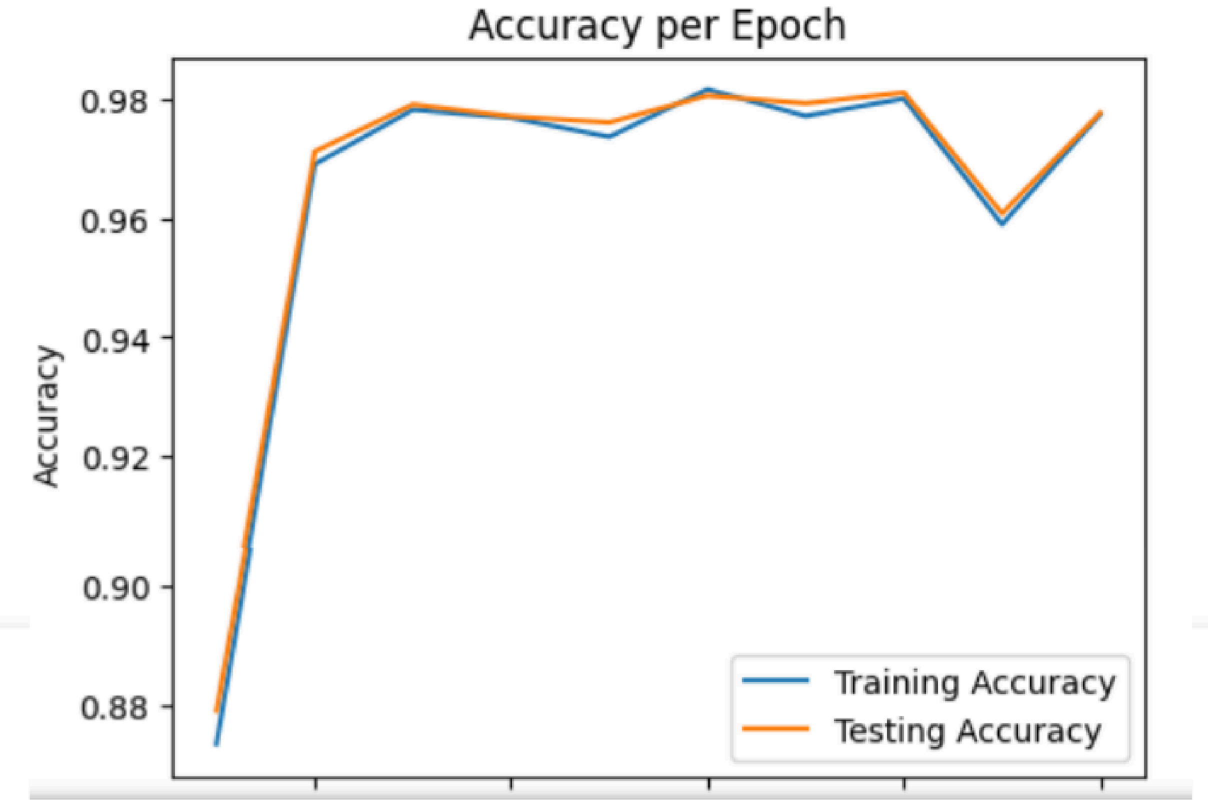
Training Configuration

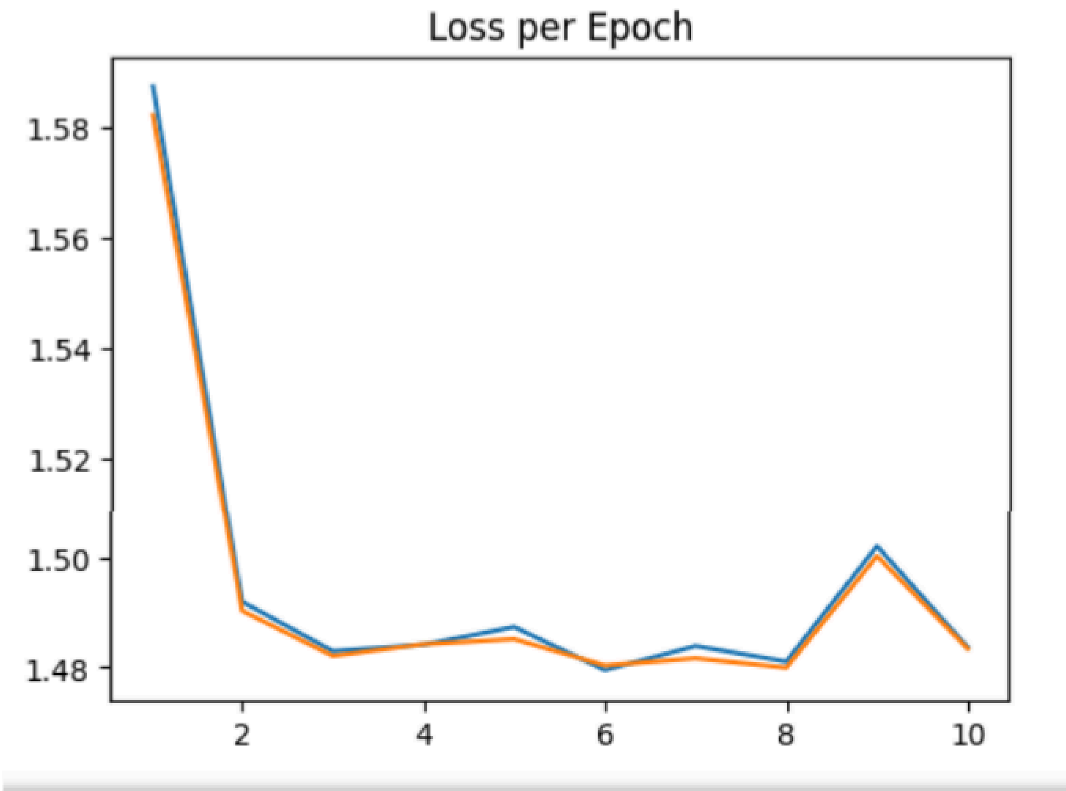
- Batch Size: Determined by roll number (Batch Size = 20, as 23%3=2)
- Activation Functions: ReLU for convolution layers, Softmax for the output layer
- Optimizer: Adam
- Loss Function: Cross-Entropy
- Training Epochs: 10

Results

Training and testing were conducted for 10 epochs, and the following metrics were observed:

Accuracy and Loss per Epoch


Accuracy per Epoch
Accuracy
0.98
0.96
0.94
0.92
0.90
0.88
Training Accuracy
Testing Accuracy



Confusion Matrix

Model Parameters

- Total Trainable Parameters: Calculated based on the model architecture
- Total Non-trainable Parameters: Calculated based on the model architecture

Manual Calculation of Trainable Parameters:

Convolution Layer 1:

- Input channels: 1
- Kernel size: 7x7
- Output channels: 16

- Trainable parameters = (input_channels * kernel_size * output_channels) + output_channels (bias terms)
- Trainable parameters = (1 * 7 * 7 * 16) + 16 = 800

Convolution Layer 2:

- Input channels: 16
- Kernel size: 5x5
- Output channels: 8
- Trainable parameters = (input_channels * kernel_size * output_channels) + output_channels (bias terms)
- Trainable parameters = (16 * 5 * 5 * 8) + 8 = 3,208

Convolution Layer 3:

- Input channels: 8
- Kernel size: 3x3
- Output channels: 4
- Trainable parameters = (input_channels * kernel_size * output_channels) + output_channels (bias terms)
- Trainable parameters = (8 * 3 * 3 * 4) + 4 = 292

Fully Connected Output Layer:

- Input size: Calculated dynamically in the `calculate_fc_size` method
- Output size: 10 (for 10 classes)
- Trainable parameters = (input_size * output_size) + output_size (bias terms)
- Trainable parameters = (calculated_input_size * 10) + 10

Manual Calculation of Non-trainable Parameters:

MaxPool and AvgPool Layers:

- MaxPool1: (2x2) with stride=1
- MaxPool2: (2x2) with stride=1
- AvgPool: (2x2) with stride=2
- Non-trainable parameters = 0 (as these layers don't have parameters)

Summary:

- Total Trainable Parameters = 800 + 3,208 + 292 + fc_trainable
- Total Non-trainable Parameters = 0 (as there are no other layers with parameters)

Experiment 2: 4-Class Classification

Dataset Modification

The classes in the dataset were combined as follows:

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

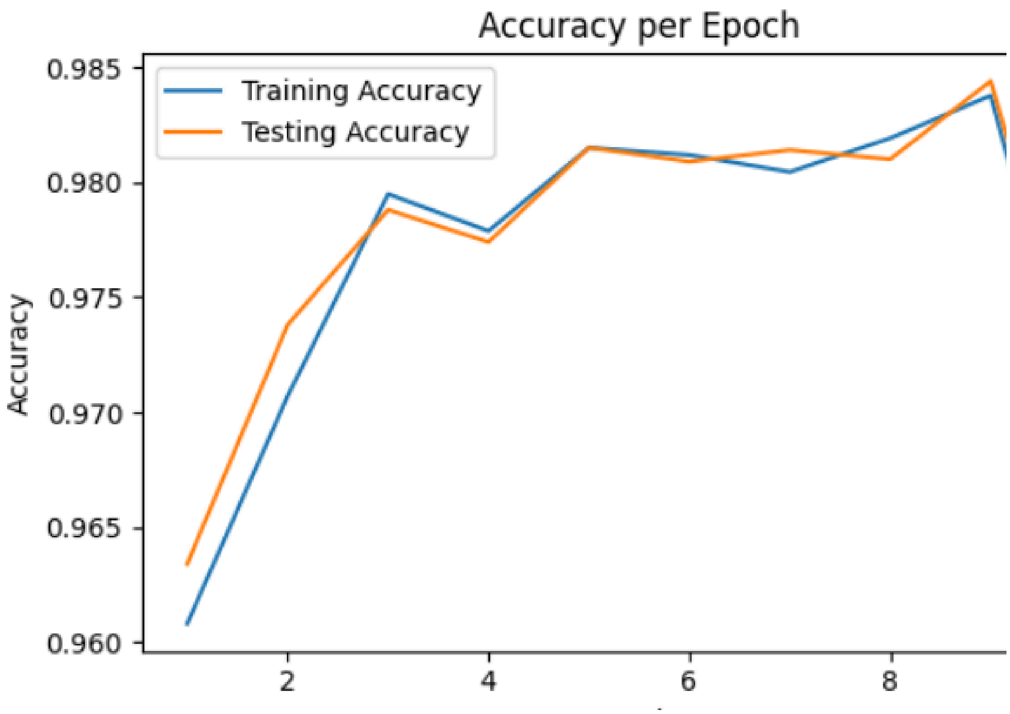
Training Configuration

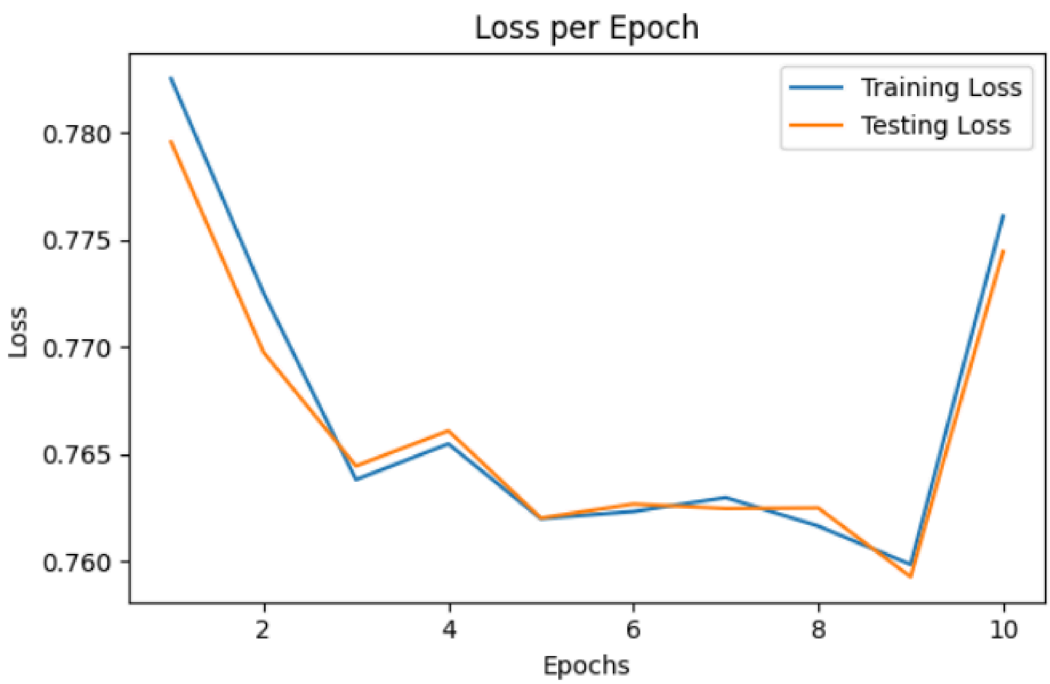
The same CNN model from Experiment 1 was used with modifications to the number of classes (num_classes=4).

Results

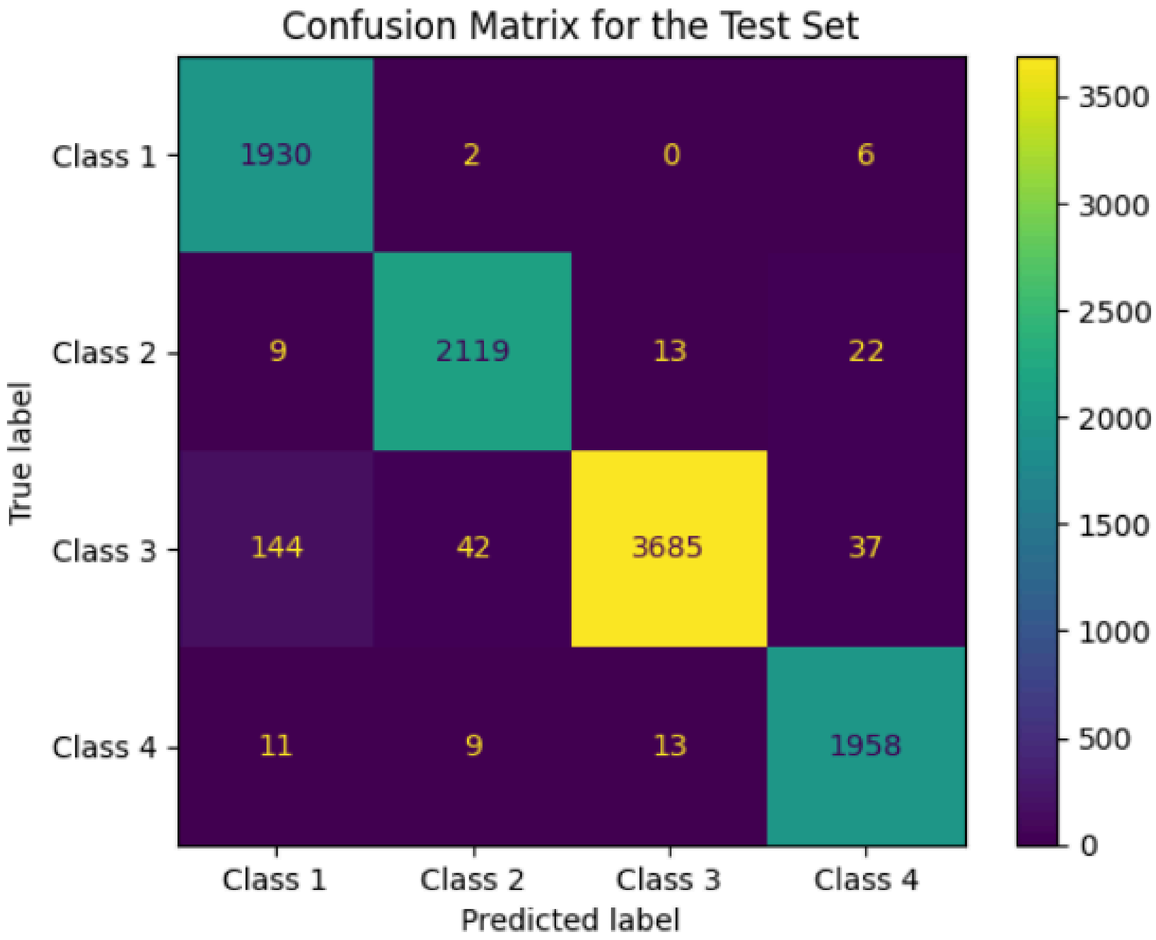
Training and testing were conducted for 10 epochs, and the following metrics were observed:

Accuracy and Loss per Epoch





Confusion Matrix



Model Parameters

- Total Trainable Parameters: Calculated based on the model architecture
- Total Non-trainable Parameters: Calculated based on the model architecture

Manual Calculation of Trainable Parameters:

Convolution Layer 1:

- Input channels: 1
- Kernel size: 7x7
- Output channels: 16

- Trainable parameters = (input_channels * kernel_size * output_channels) + output_channels (bias terms)
- Trainable parameters = (1 * 7 * 7 * 16) + 16 = 800

Convolution Layer 2:

- Input channels: 16
- Kernel size: 5x5
- Output channels: 8
- Trainable parameters = (input_channels * kernel_size * output_channels) + output_channels (bias terms)
- Trainable parameters = (16 * 5 * 5 * 8) + 8 = 3,208

Convolution Layer 3:

- Input channels: 8
- Kernel size: 3x3
- Output channels: 4
- Trainable parameters = (input_channels * kernel_size * output_channels) + output_channels (bias terms)
- Trainable parameters = (8 * 3 * 3 * 4) + 4 = 292

Fully Connected Output Layer:

- Input size: Calculated dynamically in the `calculate_fc_size` method
- Output size: 4 (for 4 classes)
- Trainable parameters = (input_size * output_size) + output_size (bias terms)
- Trainable parameters = (calculated_input_size * 4) + 4

Manual Calculation of Non-trainable Parameters:

MaxPool and AvgPool Layers:

- MaxPool1: (2x2) with stride=1
- MaxPool2: (2x2) with stride=1
- AvgPool: (2x2) with stride=2
- Non-trainable parameters = 0 (as these layers don't have parameters)

Summary:

- Total Trainable Parameters = 800 + 3,208 + 292 + fc_trainable
- Total Non-trainable Parameters = 0 (as there are no other layers with parameters)

Bonus: Performance Improvement

In the context of deep learning models, improving performance and avoiding overfitting can be achieved through various techniques:

1. Dropout:

- Implement dropout layers within the model architecture during training.
- Dropout helps prevent overfitting by randomly setting a fraction of input units to zero at each update.

2. Data Augmentation:

- Augment the training dataset by applying random transformations to the images.
- Common transformations include rotation, flipping, and scaling.
- This helps the model generalize better to variations in the input data.

3. Learning Rate Scheduling:

- Adjust the learning rate during training to allow the model to converge faster or avoid overshooting.
- Learning rate schedules like step decay or adaptive learning rate methods (e.g., Adam optimizer) can be beneficial.

4. Batch Normalization:

- Introduce batch normalization layers to normalize the input of each layer, improving stability and convergence.

- Batch normalization can act as a regularizer and reduce the dependence on initialization.

5. Early Stopping:

- Monitor the model's performance on a validation set and stop training once performance starts degrading.
- Early stopping helps prevent overfitting and saves computational resources.

Colab Link

https://colab.research.google.com/drive/1M21lrupOPfxqlyGmplHrkGfnnml-aEzL?usp=sharing