Deep Learning Assignment 1

SAJAD AHMAD MIR (M21MA207)

January 24, 2024

https://colab.research.google.com/drive/1wUVS9X9GfDb-bpwgEz0eU$_N zXQblo0Ks?usp = sharing$

1 Introduction

In this project, we build a Convolutional Neural Network using the pytorch python library. The neural network takes an image of 28*28 pixels and predicts its label. Experiment 1 is 10 class classification problem and experiment 2 is 4 class classification problem, some classes are merged in the 2nd experiment.

Methodology

2 Experiment 1: Original MNIST Classification

2.1 Problem Statement

The goal is to classify handwritten digits from the MNIST dataset into the appropriate digit classes (0 to 9).

2.2 Network Architecture Design

- Design a CNN architecture with three convolutional layers and an output layer.
- Specify details for each convolutional layer (kernel size, pooling, stride, output channels).
- Implement an output layer with the size corresponding to the number of classes (10 for MNIST).
- Use ReLU activation for convolutional layers and softmax for the output layer.
- Apply zero-padding to preserve input image dimensions.

2.3 Training Setup

- Utilize the Adam optimizer and the cross-entropy loss function.
- Set batch size based on the last three digits of the roll number, following specific conditions.
- Train the model for 10 epochs.

2.4 Evaluation

- Monitor and record training loss, validation loss, and test accuracy for each epoch.
- Generate a confusion matrix to analyze the model's performance on the test set.
- Present total and trainable parameters of the model.

Experiment 2: Modified Classification with Combined Classes

2.5 Problem Statement

Combine digit classes from the MNIST dataset to create new classes and redefine the classification task:

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

2.6 Network Architecture Reuse

Reuse the CNN architecture from Experiment 1.

2.7 Data Modification

Combine digit images to create the specified new classes.

2.8 Training Setup

- Utilize the Adam optimizer and the cross-entropy loss function.
- Set batch size based on the last three digits of the roll number, following specific conditions.
- Train the model for 10 epochs.

2.9 Evaluation

- Monitor and record training loss, validation loss, and test accuracy for each epoch.
- Generate a confusion matrix to analyze the model's performance on the modified test set.
- Present total and trainable parameters of the model.

Common Steps for Both Experiments

2.10 Data Preprocessing

- Normalize the pixel values of the MNIST images.
- Split the dataset into training, validation, and test sets.

2.11 Implementation

- Implement the CNN models using a deep learning framework (e.g., PyTorch).
- Define data loaders for efficient data handling during training.

2.12 Training

- Train the models on the training set.
- Utilize the validation set to monitor and adjust hyperparameters.

2.13 Analysis and Interpretation

- Analyze the training curves to assess convergence and overfitting.
- Interpret the confusion matrices to understand the model's classification behavior.
- Assess the impact of combining classes on model performance in Experiment 2.

3 Network Architecture

The convolutional neural network (CNN) architecture used for this experiment consists of three fully convolutional layers and an output layer. The details of the network architecture are as follows:

1. Convolution Layer 1:
   - Kernel Size: 7×7
   - Maxpooling with Stride: 1
   - Output Channels: 16
2. Convolution Layer 2:
   - Kernel Size: 5×5
   - Maxpooling with Stride: 1
   - Output Channels: 8
3. Convolution Layer 3:
   - Kernel Size: 3×3
   - Average Pooling with Stride: 2
   - Output Channels: 4
4. Output Layer:
   - Size: Number of classes

Zero-padding is applied to preserve the input image dimensions. The activation function used for convolution layers is ReLU, and for the output layer, softmax is employed. The Adam optimizer is utilized with the cross-entropy loss function.

4 Experimental Setup

- Training Duration: 10 epochs
- Batch Size:
  - If the last three digits of the roll number are divisible by 2, then batch size is set to 32.
  - If divisible by 3, then batch size is set to 16.
  - Otherwise, batch size is set to 20.

5 Experiment 1: Results

5.1 Training Results

Epoch	Train Loss	Validation Loss	Test Accuracy
1	0.2559	0.1006	0.9783
2	0.0791	0.0703	0.9838
3	0.0603	0.0589	0.9860
4	0.0521	0.0506	0.9862
5	0.0452	0.0468	0.9865
6	0.0415	0.0397	0.9860
7	0.0377	0.0380	0.9875
8	0.0352	0.0357	0.9888
9	0.0334	0.0332	0.9892
10	0.0308	0.0310	0.9896

Table 1: Training Results

As the number of epochs increases the model accuracy increases

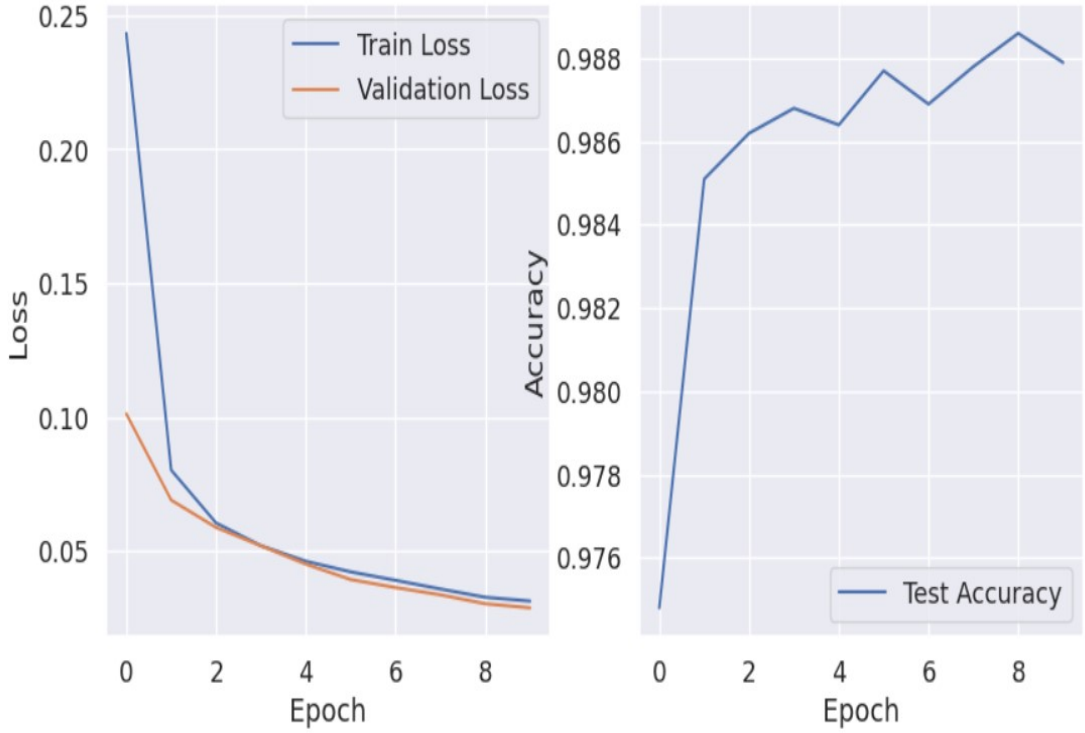
5.2 Confusion Matrix

The confusion matrix provides insights into the model's classification performance on the test set. Each row represents the actual digit, and each column represents the predicted digit. For example, the entry in the 3rd row and 4th column represents the number of times the digit 2 was predicted as 3.

Actual	Predicted								
	0	1	2	3	4	5	6	7	8
0	972	0	1	0	0	2	1	0	3
1	0	1131	0	2	0	1	1	0	0
2	0	6	1019	1	0	0	2	2	2
3	0	0	0	1009	0	1	0	0	0
4	1	1	0	0	977	0	1	0	1
5	0	0	0	5	0	886	1	0	0
6	2	3	0	0	3	5	941	0	4
7	0	6	8	0	0	1	0	1009	1
8	1	1	1	1	0	1	0	1	965
9	0	3	1	1	4	4	0	6	3

Table 2: Confusion Matrix

5.3 Graphical Analysis



5.4 Model Parameters

- Total Parameters: 5750
- Trainable Parameters: 5750

5.5 Observations

Training Loss and Validation Loss:

Both training and validation losses decrease steadily over the epochs, indicating effective learning and generalization.

Test Accuracy:

The model achieves high test accuracy, reaching 98.96% after 10 epochs. This suggests that the model is performing well on unseen data.

Confusion Matrix:

The confusion matrix provides a detailed breakdown of the model's predictions for each digit. Off-diagonal elements may indicate misclassifications.

Model Parameters:

The model has a total of 5750 parameters, all of which are trainable. This indicates the complexity of the model.

6 Experiment2: Results

Modified Classification Setup

The digit images were combined into the following classes:

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

6.1 Training Results

The training results over 10 epochs are as follows:

Epoch	Train Loss	Validation Loss	Test Accuracy
1	0.1527	0.0716	0.9824
2	0.0521	0.0461	0.9840
3	0.0400	0.0366	0.9876
4	0.0343	0.0296	0.9908
5	0.0300	0.0269	0.9916
6	0.0270	0.0245	0.9912
7	0.0248	0.0240	0.9914
8	0.0237	0.0224	0.9919
9	0.0220	0.0203	0.9916
10	0.0214	0.0195	0.9918

Table 3: Training Results

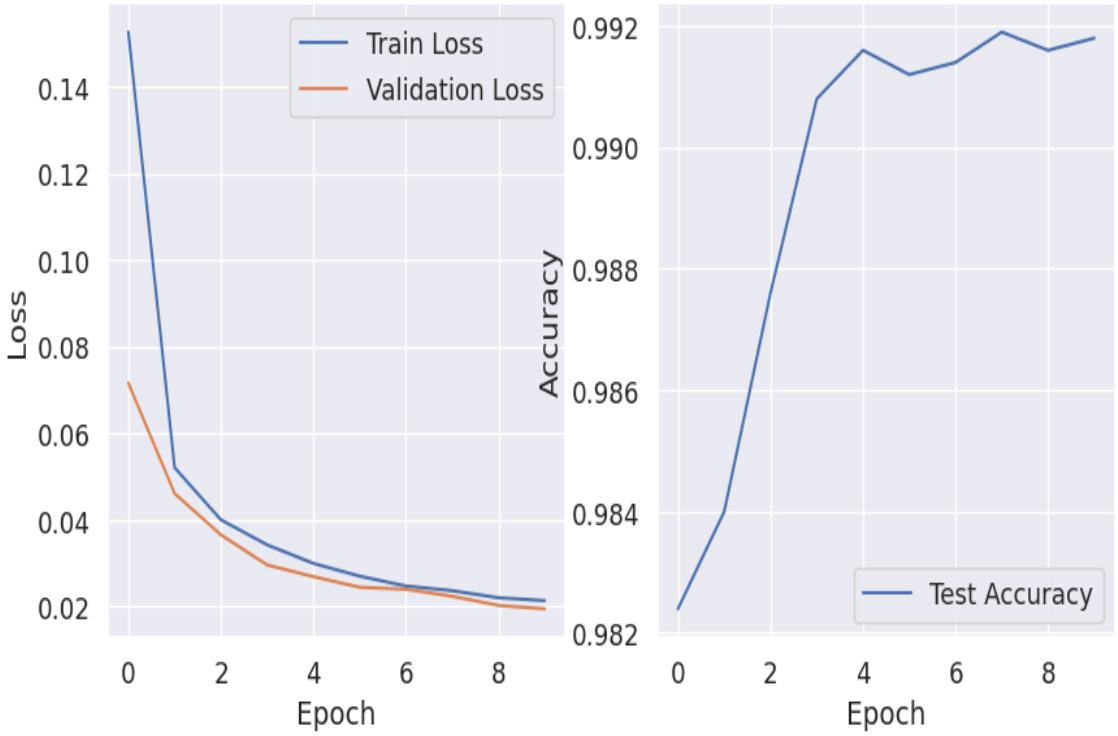
6.2 Confusion Matrix

The confusion matrix provides insights into the model's classification performance on the test set:

Actual	Predicted			
	Class 1	Class 2	Class 3	Class 4
0	1909	5	22	2
1	1	2145	15	2
2	4	9	3890	5
3	0	3	14	1974

Table 4: Confusion Matrix

6.3 Graphical Analysis



6.4 Model Parameters

The model has the following parameters:

- Total Parameters: 4880
- Trainable Parameters: 4880

6.5 Observations

Training Loss and Validation Loss: - Both training and validation losses decrease over epochs, indicating effective learning.
Test Accuracy: - The model achieves high test accuracy, reaching 99.18% after 10 epochs. This suggests that the model is performing well on the modified classification task.
Confusion Matrix: - The confusion matrix illustrates the model's ability to classify digits into the new combined classes. High values along the diagonal indicate accurate predictions.
Model Parameters: - The model has a total of 4880 parameters, all of which are trainable. This indicates the complexity of the model.

6.6 Conclusion

In summary, the modified classification task yields a well-performing model, demonstrating high accuracy and effective learning on the combined classes. Further analysis, such as visualizing misclassifications, can provide additional insights into the model's behavior.

7 Colab Link

https://colab.research.google.com/drive/1wUVS9X9GfDb-bpwgEz0eU$_N$zXQblo0Ks?usp = sharing