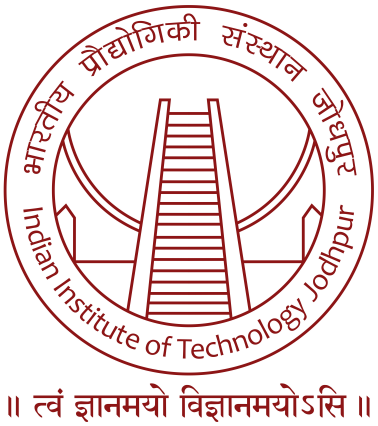
Indian Institute of Technology, Jodhpur



Building a CNN from Scratch and using it for Multi-class Classification

CSL7590: Deep Learning (Spring, 2024)

Assignment: 1

Due Date: January 25, 2024

Name:- Sahil
Roll No.:- M21MA210

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

Contents

List of Figures

List of Tables

1 Building a CNN using PyTorch Framework

1.1 Introduction to the 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. It is divided into 4 parts as follows:

- Training dataset images having $60,000$ images of size 28×28.
- Training dataset labels corresponding to each image in training set.
- Test dataset images having $10,000$ images of size 28×28.
- Test dataset labels corresponding to each image in test set.

Visualisation of Data point (image) in the given dataset (containing images):

Figure 1: Visualising the images of the given dataset

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:-

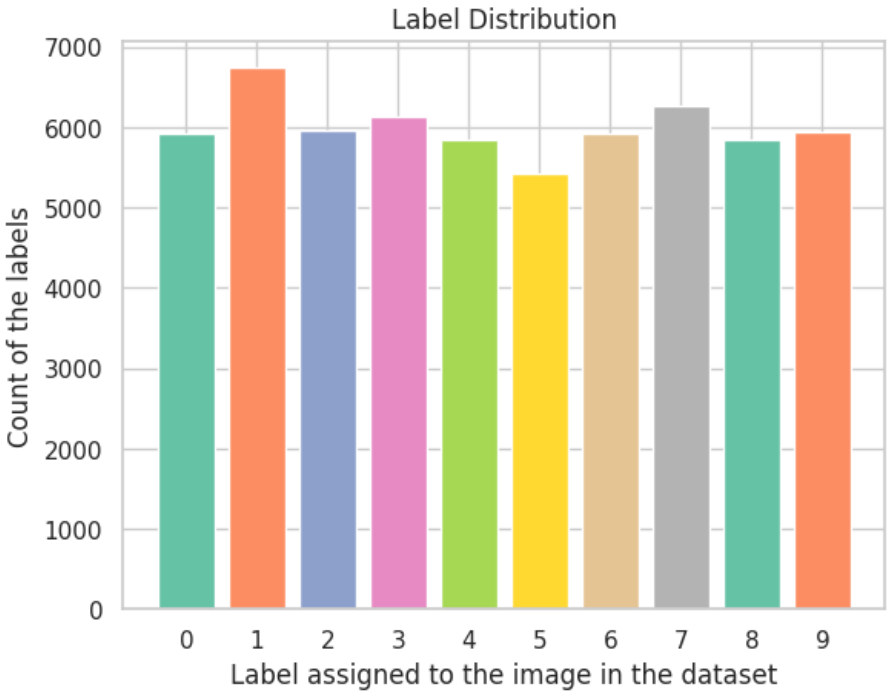


Figure 2: Label Count Histogram

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

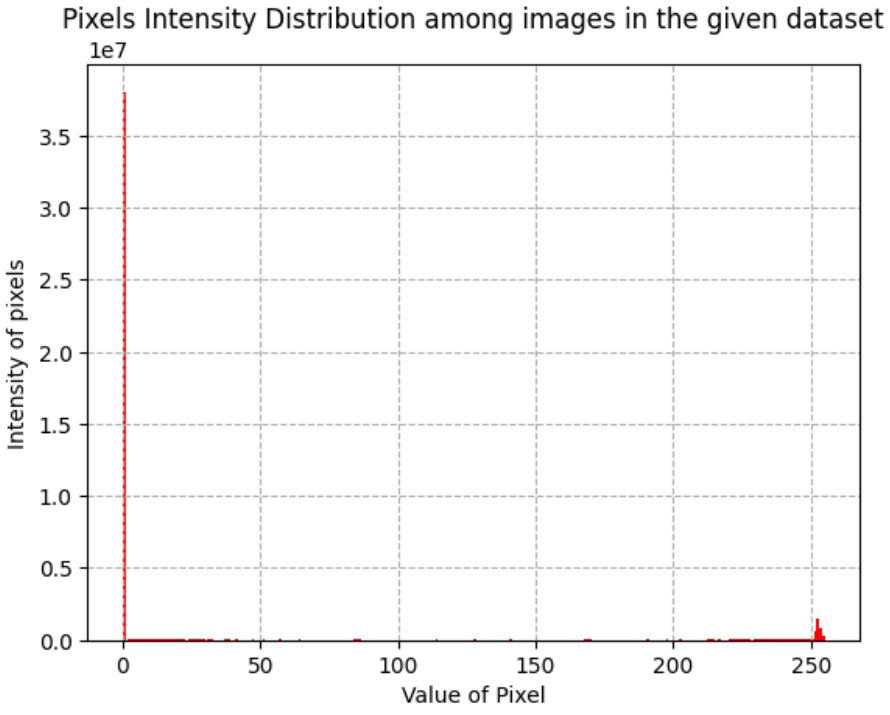


Figure 3: Pixel Intensity Distribution

1.3 Task Overview

- **For Experiment 1:** I have implemented a Convolutional Neural Network from scratch in Python using PyTorch framework. I built CNN architecture, loaded the data, trained the model, got information about its loss and accuracy trends using some plots and at last tested my model on training dataset, computed the loss and accuracy of the model on it and finally plotted the confusion matrix.
- **For Experiment 2:** We have to make the custom dataset by mapping data set into 4 different classes as :-

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

  Then following the same steps, we have to evaluate the new model and measure its' performance.

1.4 Network Architecture (Layer-wise description)

The CNN Architecture used in the assignment is as follows:- It has 3 fully convolutional layers and 1 fully connected layer.

1. **Convolutional Layer (I)**
   (a) Input channels: 1 (as these are black and white (gray-scale) images).
   (b) Output channels: 16
   (c) Kernel size: 7×7
   (d) Stride: 1
   (e) Zero-Padding: 3
   (f) Activation: ReLU
   (g) Pooling: Max pooling with a kernel size of 2×2 and stride of 2
2. **Convolutional Layer (II)**
   (a) Input channels: 16 (as these are black and white (gray-scale) images).
   (b) Output channels: 8
   (c) Kernel size: 5×5
   (d) Stride: 1
   (e) Zero-Padding: 2
   (f) Activation: ReLU
   (g) Pooling: Max pooling with a kernel size of 2×2 and stride of 2

3. **Convolutional Layer (III)**
   (a) Input channels: 8 (as these are black and white (gray-scale) images).
   (b) Output channels: 4
   (c) Kernel size: 3×3
   (d) Stride: 1
   (e) Zero-Padding: 1
   (f) Activation: ReLU
   (g) Pooling: Average pooling with a kernel size of 2×2 and stride of 2
4. **Fully Connected Layer (Output Layer)**
   (a) Flattened the output received from the previous layer
   (b) Fully connected (Linear) layer with $4 \times 3 \times 3$ input features
   (c) Activation: Softmax
   (d) Since number of classes = 10 for Experiment 1, so we set the output size = 10
   (e) Since number of classes = 4 for Experiment 2, so we set the output size = 4
5. Zero Padding is used in it so that the dimension of the input images can be preserved.
6. The standard train-test was done on the dataset (as per the given dataset).
7. As my Roll No is M21MA210, whose last three digits are 210, divisible by 2 and hence **batch size = 32**.
8. Used Adam optimiser with learning rate as 0.001 and loss function as **Cross Entropy Loss**.

1.5 Implementation of CNN Model (Methodology) and Training setup

The CNN model is trained using the training dataset in following methodology and the observations were recorded as below:

1. The data was fetched from drive and some data analysis was performed on it.
2. Using the CNN architecture, CNN Model was created.
3. The dataset is converted into PyTorch Tensors and then loaded as training dataset using the function defined: data_loading
4. The optimiser and loss function employed, were called for training setup and then used for our model.
5. Then for 10 epochs, the model is trained by calling function defined: model_training.

6. The model accuracy and loss occurred during training were recorded and loss, accuracy curves were plotted.
7. Then the model is tested on test dataset and accuracy on test data is noted.
8. Confusion matrix is plotted and total parameters were noted.
9. Classification report for the model was obtained for both the models of Experiment 1 and 2.
10. At final step, the values of all the model parameters for both the experiments are written in a text file.

2 Results

2.1 Experiment 1

Following are the results obtained when we classified the data for 10 classes:

- The table below shows the Loss and Accuracy per epoch during training phase:

Epoch	**Loss**	**Accuracy (in %)**
1	1.6840	77.7250
2	1.5990	86.1600
3	1.5945	86.6300
4	1.5614	89.9400
5	1.4973	96.3800
6	1.4977	96.3417
7	1.4951	96.5800
8	1.4949	96.6233
9	1.4941	96.6900
10	1.4921	96.9117

Table 1: Training of model in Experiment 1

  Therefore, we get accuracy of the model on training set: 96.9117%

- **Curve representing the loss** during the training for Experiment 1

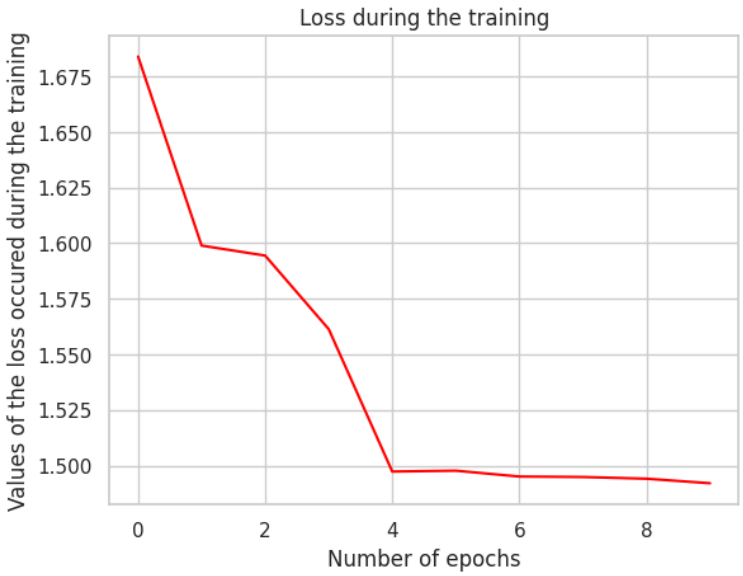


Figure 4: Loss Curve during the training

- **Curve representing the Accuracy** during the training for Experiment 1

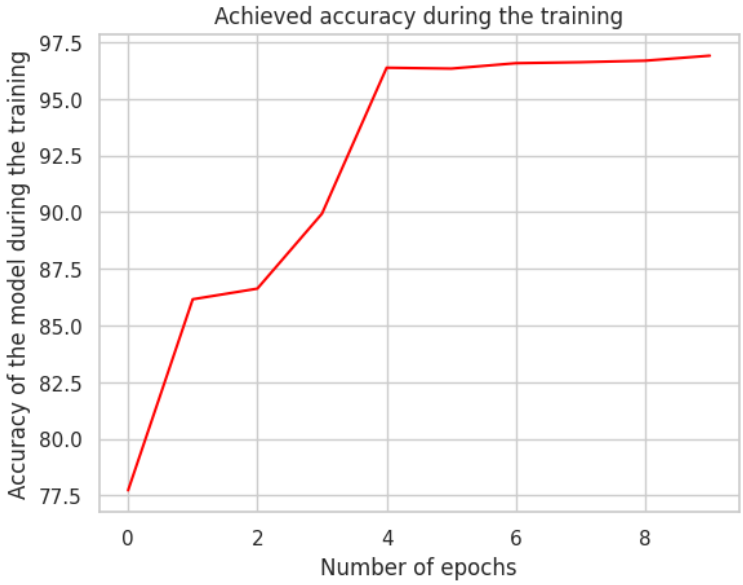


Figure 5: Accuracy Curve during the training

- **Accuracy** obtained by the model on the test data set is as: 97.11%
- **Confusion Matrix** obtained as follows:-

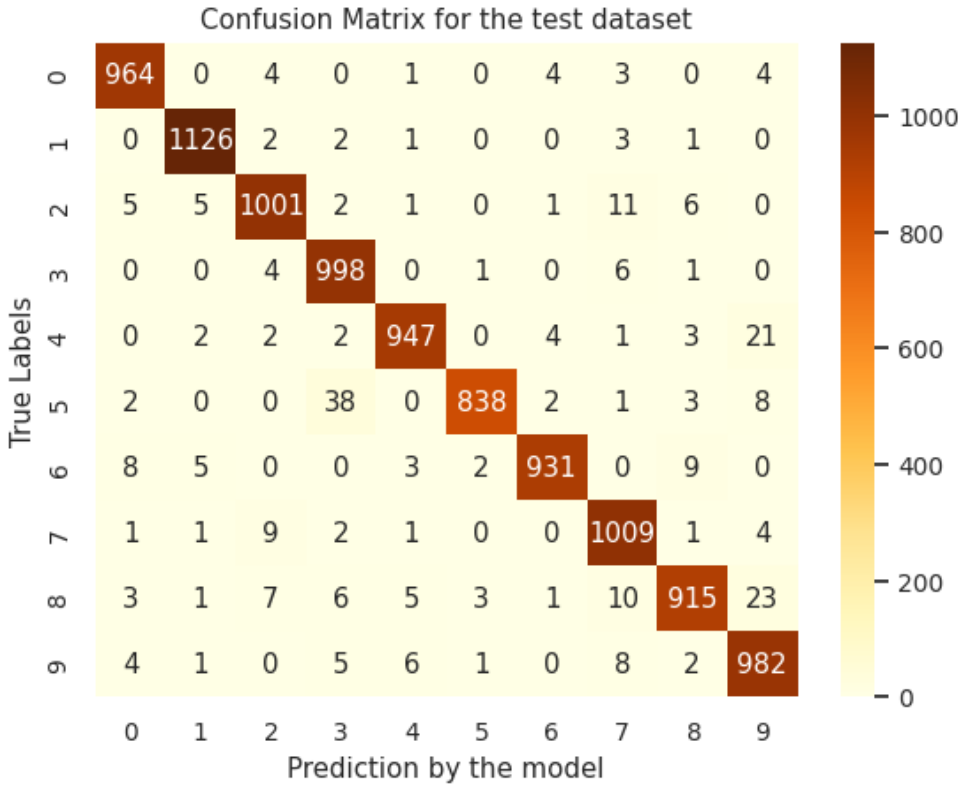


Figure 6: Confusion Matrix

- **Classification Report for Experiment 1**

Class	Precision	Recall	F1-Score	Support
0	0.98	0.98	0.98	980
1	0.99	0.99	0.99	1135
2	0.97	0.97	0.97	1032
3	0.95	0.99	0.97	1010
4	0.98	0.96	0.97	982
5	0.99	0.94	0.96	892
6	0.99	0.97	0.98	958
7	0.96	0.98	0.97	1028
8	0.97	0.94	0.96	974
9	0.94	0.97	0.96	1009
Accuracy			0.97	10000
Macro Avg	0.97	0.97	0.97	10000
Weighted Avg	0.97	0.97	0.97	10000

Table 2: Classification Report

- Total number of Parameters are 4670, out of trainable parameters are 4670 and 0 non-trainable parameters.

- All the values of these parameters are depicted in the file attached to the link below: Click here for file containing the values of all the parameters.

2.2 Experiment 2

Following are the results obtained when data is classified for 4 classes:

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

- The table below shows the Loss and Accuracy per epoch during training phase: Therefore, we get accuracy of the model on training set: 97.8033%.

Epoch	Loss	Accuracy (in %)
1	0.7962	94.8450
2	0.7684	97.5050
3	0.7664	97.7033
4	0.7655	97.7950
5	0.7664	97.7017
6	0.7669	97.6500
7	0.7663	97.7267
8	0.7674	97.6200
9	0.7680	97.5483
10	0.7656	97.8033

Table 3: Training of model in Experiment 2

- **Curve representing the loss** during the training for Experiment 2

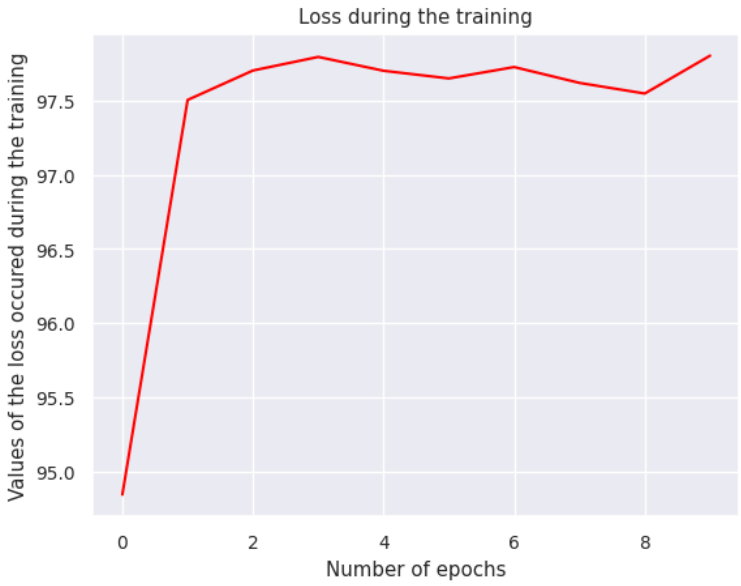


Figure 7: Loss Curve during the training

- **Curve representing the Accuracy** during the training for Experiment 2

Figure 8: Accuracy Curve during the training

- **Accuracy** obtained by the model on the test data set is as: 96.10%
- **Confusion Matrix** obtained as follows:-

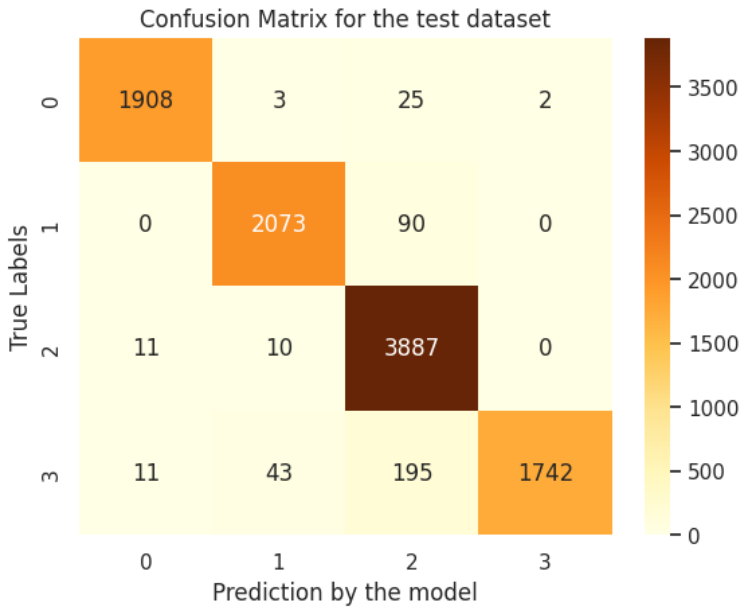


Figure 9: Confusion Matrix

- **Classification Report for Experiment 2**

Class	Precision	Recall	F1-Score	Support
0	0.99	0.98	0.99	1938
1	0.97	0.96	0.97	2163
2	0.93	0.99	0.96	3908
3	1.00	0.87	0.93	1991
Accuracy			0.97	10000
Macro Avg	0.97	0.97	0.97	10000
Weighted Avg	0.97	0.97	0.97	10000

Table 4: Classification Report

- Total number of Parameters are 4448, out of trainable parameters are 4448 and 0 non-trainable parameters.
- All the values of these parameters are depicted in the file attached to the link below: Click here for file containing the values of all the parameters.

3 Bonus Question (Attempted)

Use techniques to improve performance and avoid over fitting, if it occurs. To avoid over fitting and improve the performance of the model, the following techniques were employed:

1. Applying L2 Regularisation in the loss function, in the experiment this happened by adding parameter weight_decay= $1e-4$ in the Adam optimiser, this hyper-parameter was tuned after training the model several times. This don't allow the model to give importance to only specific features.

2. We also applied dropout regularisation to the fully connected layer. By doing so, we are making fraction of input units to zero during training and thus help prevents the over fitting.

3. Learning Rate Scheduling: In order to improve the performance of the model, we can change the learning rate during training after some steps. In the model, we applied lr_scheduler with step size = 5 by decay factor =0.9, so that after every 5 epochs, the learning rate is decayed by the decay factor

4. It can also improved by doing batch normalisation.

5. By optimising the hyper parameters, over fitting can be reduced and performance can be enhanced.

4 Link to Notebook

Google Collab Notebook link:- Click here to navigate to Google Colab Notebook

5 Resources Used

- https://pytorch.org/tutorials/beginner/nlp/pytorch_tutorial.html
- https://pytorch.org/docs/stable/nn.html
- https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/
- Lecture Slides provided by the instructor

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