PRML PA-6

| Neural Networks |

Colab Link

Rhythm Patni (B22CS043)

Question 1 : Neural Networks

We are given the MNIST dataset, which contains a lot of images (28x28 images) of "Handwritten-Digits" with their labels. We have to explore the capability of Neural Networks in correctly classifying images.

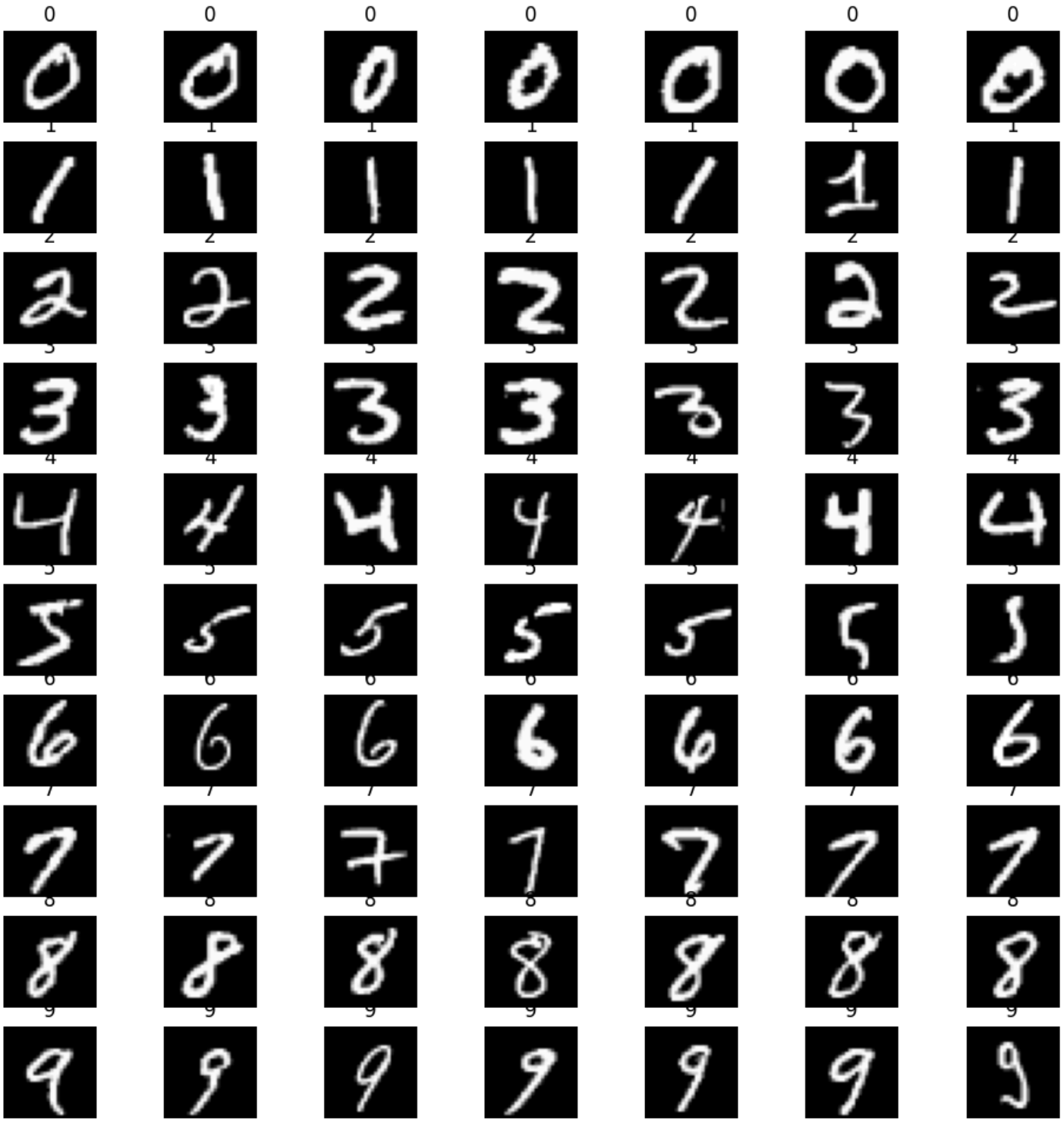
Task 0 : Loading the MNIST dataset.

The MNIST dataset was loaded using torch-vision, the data was split into training,testing and validation sets. The following pre-processing augmentations were performed on the dataset.

1. **RandomRotation** : All the images are randomly rotated by a certain degree, the amount of rotations is also randomly decided, The degree parameter is set to 10, so the images are randomly rotated between -10 to 10 degrees, this will help to train very robust models, which can handle rotated images.
2. **RandomCrop :** All the images are randomly cropped by a certain amount, the amount of crop is also randomly decided, this will help to train very robust models, which can handle rotated images. The size of the output images are 28x28 and the padding parameters is set to 2, that is, 2 extra blank pixels are added at the edges before cropping.
3. **ToTensor :** All the images are transformed to a PyTorch tensor, this is necessary the models accept only tensors as input.
4. **Normalize :** The image pixel values are normalized, so their values are subtracted from the mean value and divided by the variance. This helps in stabilizing and speeding up the training process, by keeping the values in range. The mean and standard deviation for each image is set to 0.5.

Task 1 : Plotting Images and Creating DataLoaders

Shown Below are some sample images of the dataset. All the classes of digits with their variants are shown, DataLoaders for the training , testing and validation sets are created



Task 2 :Implementing a 3-Layer Perceptron using PyTorch

A 3-layer MLP was implemented using PyTorch, The number of trainable parameters is coming out to be 109386.

Task 3 : Training the Model.

The model was trained using Adam as the optimizer and CrossEntropyLoss as the loss function.

Adam stands for Adaptive Moment Estimation, Adam performs optimization with superior performance and faster convergence than conventional gradient descent algorithms by varying the learning rate for every parameter during training. It works well with different kinds of neural network topologies and datasets because it automatically modifies the learning rate based on the gradients of each parameter.

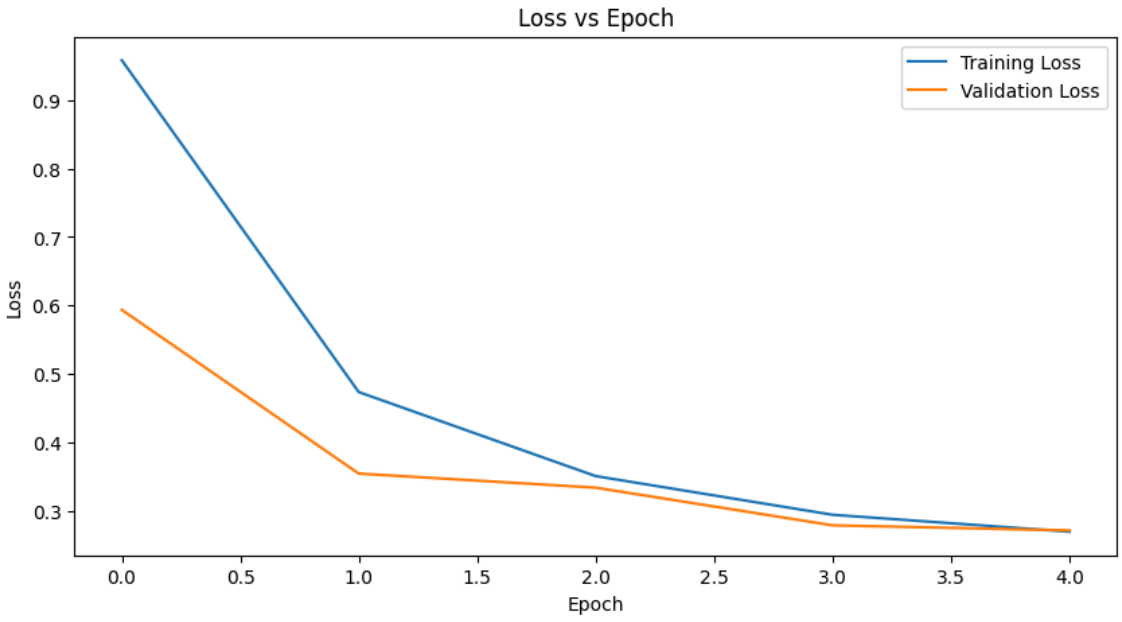
CrossEntropyLoss calculates the loss between the predicted probabilities (output of the model) and the true labels (ground truth). It penalizes incorrect predictions more heavily, leading to better training of the model. Cross-entropy loss is widely used because it's efficient, easy to compute, and produces meaningful gradients for backpropagation.

The model was evaluated at each epoch, here are the results for the same.

```
Epoch 1/5, Training Loss: 0.9585, Training Accuracy: 0.6922, Validation Loss: 0.5936, Validation Accuracy: 0.8107
Epoch 2/5, Training Loss: 0.4734, Training Accuracy: 0.8533, Validation Loss: 0.3542, Validation Accuracy: 0.8886
Epoch 3/5, Training Loss: 0.3505, Training Accuracy: 0.8917, Validation Loss: 0.3336, Validation Accuracy: 0.8944
Epoch 4/5, Training Loss: 0.2940, Training Accuracy: 0.9103, Validation Loss: 0.2786, Validation Accuracy: 0.9123
Epoch 5/5, Training Loss: 0.2693, Training Accuracy: 0.9181, Validation Loss: 0.2711, Validation Accuracy: 0.9192
```

Task 4 : Visualizing Correct/Incorrect predictions

Loss-Epoch Graphs was plotted as shown :



Accuracy-Epoch graph was plotted as shown.