Programming Assignment -6

Colab Link :

TASK-0:

- First, we set up the MNIST dataset for machine learning training.
- We used data augmentation techniques like random rotation and cropping toincrease the dataset's diversity, which helps the model generalize better.
- After transforming the data into PyTorch tensors and normalizing it, we split the dataset into training and validation sets.
- This division allowed monitoring of the model's performance on unseen data during training, aiding in preventing overfitting.
- Finally, we print the sizes of the training, validation, and test datasets.

TASK-1:

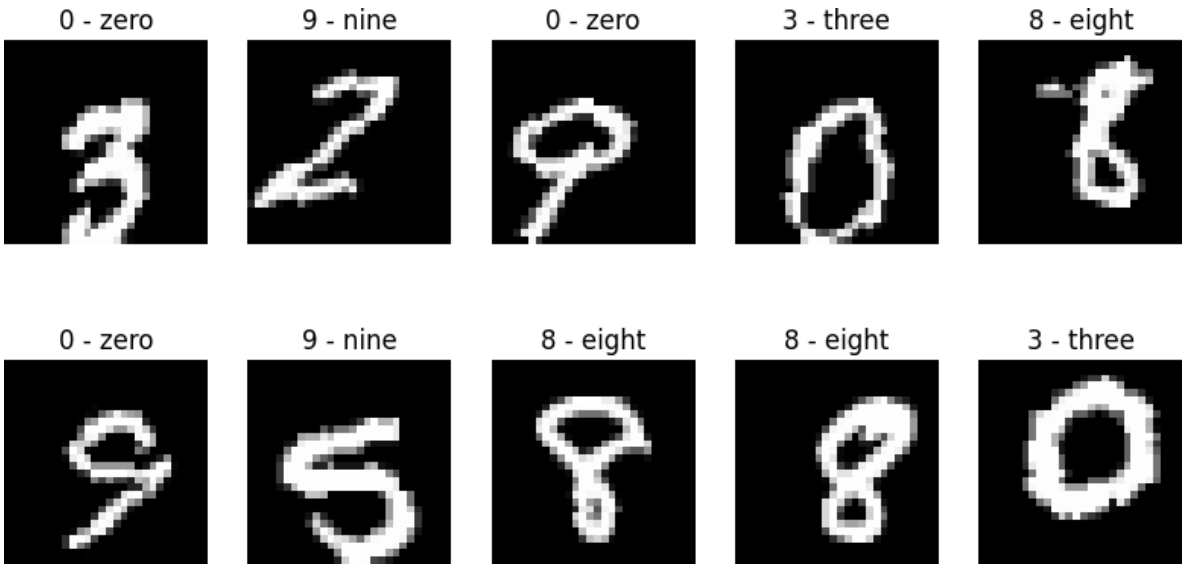
1. Image Visualization:

We plot a grid of 2 rows and 5 columns to display a total of 10 images.
For each image:

- It extracts the image and its corresponding label from the training dataset.
- The image is displayed in grayscale using `imshow()` from matplotlib.
- The class label associated with the image is used as the title for the subplot.
- The axis is turned off for better visualization.

2. Data Loader Creation:

Data loaders are created for the training, validation, and testing of datasets using `torch.utils.data.DataLoader`.
These data loaders help in efficiently loading the data in batches during model training and evaluation.



TASK-2:

1. Image Visualization:

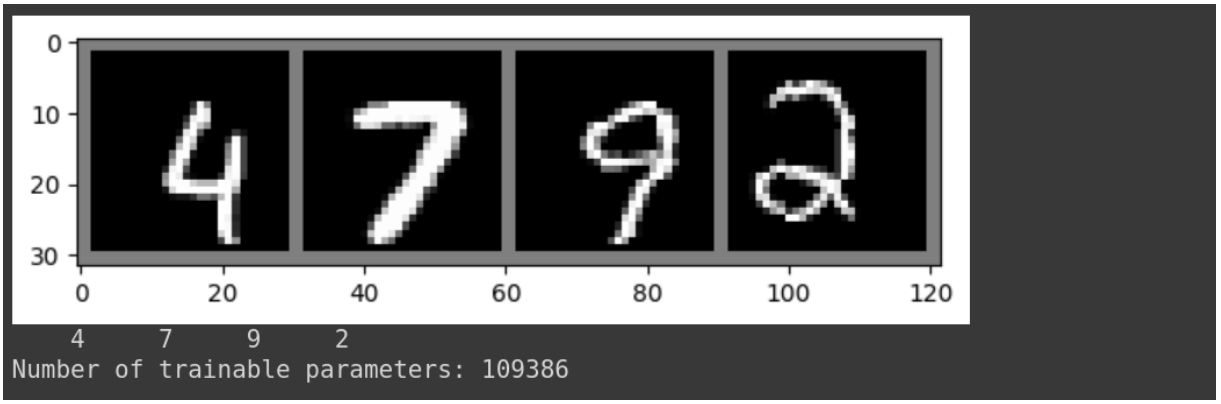
The MNIST dataset is downloaded and split into training, validation, and testing sets.
Images are displayed from the training set to visualize the data.

For each batch:

- Images are unnormalized and displayed using `imshow()`.
- The corresponding class labels are printed below each image.

2. MLP Definition and Parameter Count:

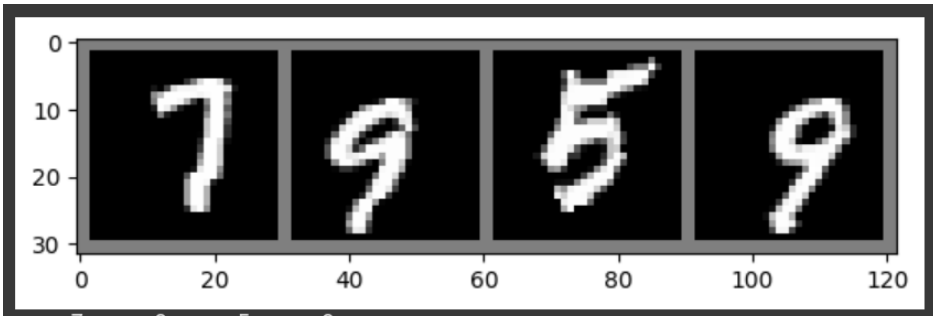
- A 3-layer Multi-Layer Perceptron (MLP) model is defined using PyTorch's `nn.Module`.
- The model consists of three fully connected layers (`nn.Linear`) with ReLU activation functions.
- The `forward` method defines the forward pass of the network.
- The total number of trainable parameters in the model is printed, providing insight into the model's complexity and memory requirements.



TASK-3:

Model Training and Evaluation:

- The model is trained for 5 epochs using the Adam optimizer and CrossEntropyLoss as the loss function.
- Training and validation loops are defined to compute loss and accuracy metrics.
- The model is trained on the GPU if available, or else on the CPU.
- After training, the best model based on validation loss is saved.
- The saved model is then loaded and evaluated on the test set, providing the final test loss and accuracy.




```
    7     9     5     9
Number of trainable parameters: 109386
Epoch 1/5, Train Loss: 0.8413, Train Acc: 0.7280, Val Loss: 0.4864, Val Acc: 0.8510
Epoch 2/5, Train Loss: 0.3916, Train Acc: 0.8773, Val Loss: 0.3789, Val Acc: 0.8778
Epoch 3/5, Train Loss: 0.2961, Train Acc: 0.9106, Val Loss: 0.2493, Val Acc: 0.9243
Epoch 4/5, Train Loss: 0.2523, Train Acc: 0.9236, Val Loss: 0.2462, Val Acc: 0.9253
Epoch 5/5, Train Loss: 0.2322, Train Acc: 0.9287, Val Loss: 0.2092, Val Acc: 0.9337
Test Loss: 0.1990, Test Accuracy: 0.9364
```

TASK-4:

- Correct and incorrect predictions are visualized along with their true and predicted labels.
- Loss and accuracy trends over epochs are plotted for both training and validation sets.