```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
!pip install kaggle
!mkdir ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d hojjatk/mnist-dataset

!unzip mnist-dataset.zip
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (1.6.17)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2024.8.30)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from kaggle) (4.66.6)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle) (2.2.3)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.4.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.10)
cp: cannot stat 'kaggle.json': No such file or directory
chmod: cannot access '/root/.kaggle/kaggle.json': No such file or directory
Dataset URL: https://www.kaggle.com/datasets/hojjatk/mnist-dataset
License(s): copyright-authors
Downloading mnist-dataset.zip to /content
 50% 11.0M/22.0M [00:00<00:00, 115MB/s]
100% 22.0M/22.0M [00:00<00:00, 169MB/s]
Archive:  mnist-dataset.zip
  inflating: t10k-images-idx3-ubyte/t10k-images-idx3-ubyte
  inflating: t10k-images.idx3-ubyte
  inflating: t10k-labels-idx1-ubyte/t10k-labels-idx1-ubyte
  inflating: t10k-labels.idx1-ubyte
  inflating: train-images-idx3-ubyte/train-images-idx3-ubyte
  inflating: train-images.idx3-ubyte
  inflating: train-labels-idx1-ubyte/train-labels-idx1-ubyte
  inflating: train-labels.idx1-ubyte
```

```
def unpickle(file):
    import pickle
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    return dict
```

```
!pip install numpy
import numpy as np
import torch
from torch.utils.data import Dataset, DataLoader, sampler
import torch.nn as nn

train_images_file = np.fromfile('/content/train-images.idx3-ubyte',dtype=np.uint8)
train_labels_file = np.fromfile('/content/train-labels.idx1-ubyte',dtype=np.uint8)
test_images_file = np.fromfile('/content/t10k-images.idx3-ubyte',dtype=np.uint8)
test_labels_file = np.fromfile('/content/t10k-labels.idx1-ubyte',dtype=np.uint8)
```

```
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
```

```
import pickle

with open('mnist_data.pickle', 'wb') as f:
    pickle.dump((train_images_file, train_labels_file, test_images_file, test_labels_file), f)
```

```
print(len(train_images_file), len(train_labels_file))
print(type(train_images_file[0]))
print(train_labels_file[0])
print(max(train_labels_file), min(train_labels_file))

print(len(test_images_file), len(test_labels_file))
print(type(test_images_file[0]))
```

```
print(test_labels_file[0])
print(max(test_labels_file), min(test_labels_file))
```

```
47040016 60008
<class 'numpy.uint8'>
0
234 0
7840016 10008
<class 'numpy.uint8'>
0
39 0
```

```
!pip install torchvision
import torchvision
from torchvision import datasets, transforms
class MNIST(Dataset):
  def __init__(self, images_file, labels_file):
    self.data = torchvision.datasets.MNIST(root='./data', train=True, download=True)
    self.images = torch.tensor(self.data.data.numpy(), dtype=torch.float32)
    self.labels = torch.tensor(self.data.targets.numpy(), dtype=torch.long)

  def __len__(self):
    return len(self.images)

  def __getitem__(self, idx):
    return self.images[idx], self.labels[idx]
```

```
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (0.20.1+cu121)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision) (1.26.4)
Requirement already satisfied: torch==2.5.1 in /usr/local/lib/python3.10/dist-packages (from torchvision) (2.5.1+cu121)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-packages (from torchvision) (11.0.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch==2.5.1->torchvision) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch==2.5.1->torchvision) (4.1
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch==2.5.1->torchvision) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch==2.5.1->torchvision) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch==2.5.1->torchvision) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch==2.5.1->torchvision) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy==1.13.1->torch==2.5.1->torchvis
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch==2.5.1->torchvision) (3.0.
```

```
train_images_file = '/content/train-images.idx3-ubyte'
train_labels_file = '/content/train-labels.idx1-ubyte'
test_images_file = '/content/t10k-images.idx3-ubyte'
test_labels_file = '/content/t10k-labels.idx1-ubyte'


train_data = MNIST(train_images_file, train_labels_file)
test_data = MNIST(test_images_file, test_labels_file)

train_dataloader = DataLoader(train_data, batch_size=64, shuffle=True)
test_dataloader = DataLoader(test_data, batch_size=10, shuffle=True)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz to ./data/MNIST/raw/train-images-idx3-ubyte.gz
100%|██████████| 9.91M/9.91M [00:00<00:00, 50.2MB/s]
Extracting ./data/MNIST/raw/train-images-idx3-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz to ./data/MNIST/raw/train-labels-idx1-ubyte.gz
100%|██████████| 28.9k/28.9k [00:00<00:00, 1.97MB/s]Extracting ./data/MNIST/raw/train-labels-idx1-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz

Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw/t10k-images-idx3-ubyte.gz
100%|██████████| 1.65M/1.65M [00:00<00:00, 14.4MB/s]
Extracting ./data/MNIST/raw/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw
```

```
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz
100%|██████████| 4.54k/4.54k [00:00<00:00, 3.38MB/s]Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw
```

```
from torch.utils.data import DataLoader

train_dataloader = DataLoader(train_data, batch_size=64, shuffle=True)
test_dataloader = DataLoader(test_data, batch_size=10, shuffle=True)

for X, y in test_dataloader:
    print(f"Shape of X [N, C, H, W]: {X.shape}")
    print(f"Shape of y: {y.shape} {y.dtype}")
    break
```

```
Shape of X [N, C, H, W]: torch.Size([10, 28, 28])
Shape of y: torch.Size([10]) torch.int64
```

```
class MLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.layers = nn.Sequential(
            nn.Flatten(),
            nn.Linear(28*28, 1024),
            nn.ReLU(),
            nn.Linear(1024, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        return self.layers(x)


device = 'cuda' if torch.cuda.is_available() else 'cpu'
model = MLP().to(device)
print(model)
```

```
MLP(
  (layers): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=784, out_features=1024, bias=True)
    (2): ReLU()
    (3): Linear(in_features=1024, out_features=512, bias=True)
    (4): ReLU()
    (5): Linear(in_features=512, out_features=10, bias=True)
  )
)
```

```
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters())


def train(data_loader, model, criterion, optimizer):
    model.train()
    total_loss = 0
    total_correct = 0
    num_items = len(data_loader.dataset)

    for data, target in data_loader:
        data, target = data.to(device), target.to(device)


        output = model(data)
        loss = criterion(output, target)


        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

```
            total_loss += loss.item() * data.size(0)
            pred = output.argmax(dim=1)
            total_correct += pred.eq(target).sum().item()

        avg_loss = total_loss / num_items
        accuracy = 100. * total_correct / num_items
        return avg_loss, accuracy


    def test(data_loader, model, criterion):
        model.eval()
        total_loss = 0
        total_correct = 0
        num_items = len(data_loader.dataset)

        with torch.no_grad():
            for data, target in data_loader:
                data, target = data.to(device), target.to(device)

                output = model(data)
                loss = criterion(output, target)

                total_loss += loss.item() * data.size(0)
                pred = output.argmax(dim=1)
                total_correct += pred.eq(target).sum().item()

        avg_loss = total_loss / num_items
        accuracy = 100. * total_correct / num_items
        return avg_loss, accuracy


    import matplotlib.pyplot as plt

    train_losses = []
    test_losses = []
    train_accuracies = []
    test_accuracies = []

    epochs = 10
    for epoch in range(epochs):
        print(f"Epoch {epoch+1}/{epochs}")

        train_loss, train_accuracy = train(train_dataloader, model, criterion, optimizer)
        train_losses.append(train_loss)
        train_accuracies.append(train_accuracy)
        print(f"Train Loss: {train_loss:.4f}, Train Accuracy: {train_accuracy:.2f}%")

        test_loss, test_accuracy = test(test_dataloader, model, criterion)
        test_losses.append(test_loss)
        test_accuracies.append(test_accuracy)
        print(f"Test Loss: {test_loss:.4f}, Test Accuracy: {test_accuracy:.2f}%")

    plt.figure(figsize=(10, 5))
    plt.plot(range(1, epochs + 1), train_losses, label='Training Loss')
    plt.plot(range(1, epochs + 1), test_losses, label='Testing Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.title('Training and Testing Loss Over Epochs')
    plt.legend()
    plt.show()
```

```
Epoch 1/10
Train Loss: 0.4185, Train Accuracy: 92.95%
Test Loss: 0.1149, Test Accuracy: 96.57%
Epoch 2/10
Train Loss: 0.1296, Train Accuracy: 96.13%
Test Loss: 0.0879, Test Accuracy: 97.27%
Epoch 3/10
Train Loss: 0.1135, Train Accuracy: 96.82%
Test Loss: 0.1028, Test Accuracy: 97.04%
Epoch 4/10
Train Loss: 0.1096, Train Accuracy: 97.02%
Test Loss: 0.0808, Test Accuracy: 97.47%
Epoch 5/10
Train Loss: 0.0966, Train Accuracy: 97.35%
Test Loss: 0.0796, Test Accuracy: 98.01%
Epoch 6/10
Train Loss: 0.0994, Train Accuracy: 97.41%
Test Loss: 0.0656, Test Accuracy: 98.10%
Epoch 7/10
Train Loss: 0.0889, Train Accuracy: 97.74%
Test Loss: 0.0951, Test Accuracy: 97.66%
Epoch 8/10
Train Loss: 0.0839, Train Accuracy: 97.92%
Test Loss: 0.0637, Test Accuracy: 98.14%
Epoch 9/10
Train Loss: 0.0728, Train Accuracy: 98.17%
Test Loss: 0.0788, Test Accuracy: 98.09%
Epoch 10/10
Train Loss: 0.0755, Train Accuracy: 98.22%
Test Loss: 0.0679, Test Accuracy: 98.21%
```