```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
import matplotlib.pyplot as plt


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)


transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

train_dataset = datasets.MNIST(root="./data", train=True, download=True, transform=transform)
test_dataset = datasets.MNIST(root="./data", train=False, download=True, transform=transform)

train_dataloader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_dataloader = DataLoader(test_dataset, batch_size=64, shuffle=False)


device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = MLP().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters())


def train(model, dataloader, criterion, optimizer, device):
    model.train()
    total_loss = 0
    correct = 0
    for data, target in dataloader:
        data, target = data.to(device), target.to(device)

        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, target)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()
        correct += (output.argmax(1) == target).type(torch.float).sum().item()

    avg_loss = total_loss / len(dataloader.dataset)
    accuracy = correct / len(dataloader.dataset) * 100
    return avg_loss, accuracy


def test(model, dataloader, criterion, device):
    model.eval()
    total_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in dataloader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            loss = criterion(output, target)

            total_loss += loss.item()
            correct += (output.argmax(1) == target).type(torch.float).sum().item()

    avg_loss = total_loss / len(dataloader.dataset)
```

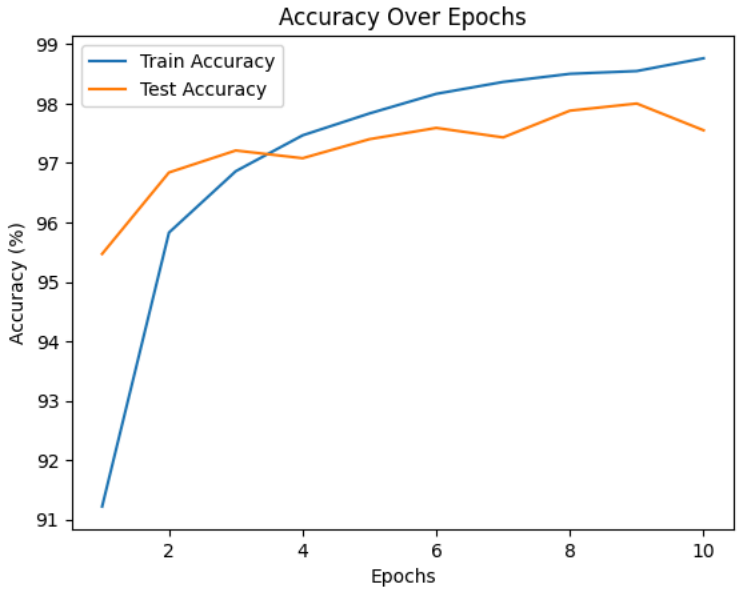
```
        accuracy = correct / len(dataloader.dataset) * 100
        return avg_loss, accuracy


    epochs = 10
    train_losses, test_losses = [], []
    train_accuracies, test_accuracies = [], []

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

        train_loss, train_accuracy = train(model, train_dataloader, criterion, optimizer, device)
        test_loss, test_accuracy = test(model, test_dataloader, criterion, device)

        train_losses.append(train_loss)
        test_losses.append(test_loss)
        train_accuracies.append(train_accuracy)
        test_accuracies.append(test_accuracy)

        print(f"Train Loss: {train_loss:.4f}, Train Accuracy: {train_accuracy:.2f}%")
        print(f"Test Loss: {test_loss:.4f}, Test Accuracy: {test_accuracy:.2f}%")
```

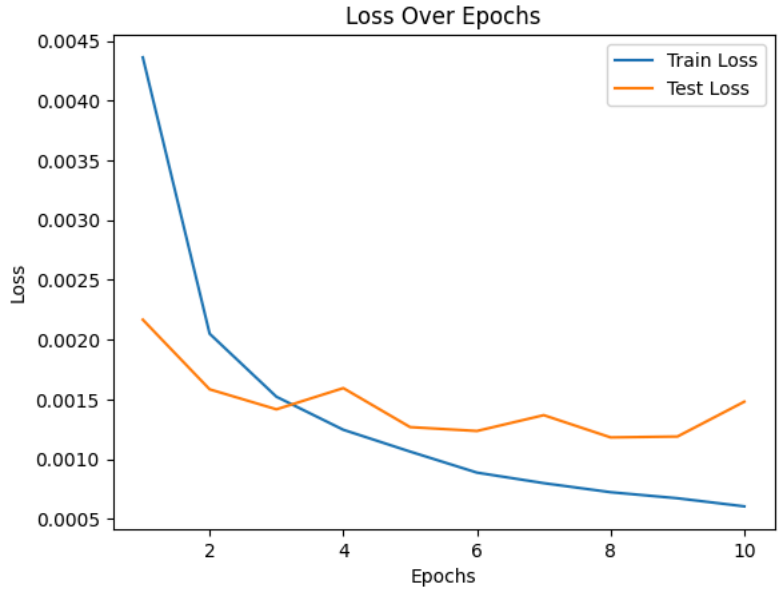
```
Epoch 1/10
Train Loss: 0.0044, Train Accuracy: 91.22%
Test Loss: 0.0022, Test Accuracy: 95.47%
Epoch 2/10
Train Loss: 0.0020, Train Accuracy: 95.83%
Test Loss: 0.0016, Test Accuracy: 96.84%
Epoch 3/10
Train Loss: 0.0015, Train Accuracy: 96.86%
Test Loss: 0.0014, Test Accuracy: 97.21%
Epoch 4/10
Train Loss: 0.0012, Train Accuracy: 97.47%
Test Loss: 0.0016, Test Accuracy: 97.08%
Epoch 5/10
Train Loss: 0.0011, Train Accuracy: 97.83%
Test Loss: 0.0013, Test Accuracy: 97.40%
Epoch 6/10
Train Loss: 0.0009, Train Accuracy: 98.16%
Test Loss: 0.0012, Test Accuracy: 97.59%
Epoch 7/10
Train Loss: 0.0008, Train Accuracy: 98.36%
Test Loss: 0.0014, Test Accuracy: 97.43%
Epoch 8/10
Train Loss: 0.0007, Train Accuracy: 98.50%
Test Loss: 0.0012, Test Accuracy: 97.88%
Epoch 9/10
Train Loss: 0.0007, Train Accuracy: 98.55%
Test Loss: 0.0012, Test Accuracy: 98.00%
Epoch 10/10
Train Loss: 0.0006, Train Accuracy: 98.76%
Test Loss: 0.0015, Test Accuracy: 97.55%
```

```
plt.plot(range(1, epochs + 1), train_accuracies, label='Train Accuracy')
plt.plot(range(1, epochs + 1), test_accuracies, label='Test Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy (%)')
plt.title('Accuracy Over Epochs')
plt.legend()
plt.show()
```

```
plt.plot(range(1, epochs + 1), train_losses, label='Train Loss')
plt.plot(range(1, epochs + 1), test_losses, label='Test Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Loss Over Epochs')
plt.legend()
plt.show()
```

[ ]


Accuracy Over Epochs
Train Accuracy
Test Accuracy
Accuracy (%)
Epochs
99
98
97
96
95
94
93
92
91
2
4
6
8
10


Loss Over Epochs
Train Loss
Test Loss
Loss
Epochs
0.0045
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
2
4
6
8
10