```
import numpy as np
from torch import nn
import torch.nn.functional as F
import torch
import helper
import matplotlib.pyplot as ply
from torchvision import datasets,transforms
transform=transforms.Compose([transforms.ToTensor(),transforms.Normalize((0.5,),(0.5,)),])
trainset=datasets.MNIST('MNIST_data/',download=True,train=True,transform=transform)
trainloader=torch.utils.data.DataLoader(trainset,batch_size=64,shuffle=True)
testset=datasets.MNIST('MNIST_data/',download=True,train=False,transform=transform)
testloader=torch.utils.data.DataLoader(testset,batch_size=64,shuffle=True)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to MNIST_data/MN
99%                                       9805824/9912422 [00:02<00:00, 1290997.86it/s]
Extracting MNIST_data/MNIST/raw/train-images-idx3-ubyte.gz to MNIST_data/MNIST/raw
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to MNIST_data/MN
57%                                       16384/28881 [00:00<00:00, 75552.77it/s]
Extracting MNIST_data/MNIST/raw/train-labels-idx1-ubyte.gz to MNIST_data/MNIST/raw
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to MNIST_data/MNI
53%                                       876544/1648877 [00:01<00:02, 263409.04it/s]
Extracting MNIST_data/MNIST/raw/t10k-images-idx3-ubyte.gz to MNIST_data/MNIST/raw
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to MNIST_data/MNI
0%                                        0/4542 [00:00<?, ?it/s]
Extracting MNIST_data/MNIST/raw/t10k-labels-idx1-ubyte.gz to MNIST_data/MNIST/raw
Processing...
Done!
```

```
from torch import optim
model =nn.Sequential(nn.Linear(784,128),
                     nn.ReLU(),nn.Linear(128,64),
                     nn.ReLU(),nn.Linear(64,32),
                     nn.ReLU(),nn.Linear(32,10),nn.LogSoftmax(dim=1))
crite=nn.NLLLoss()    #Loss Function
epoch=6
test_losses,train_losses=[],[]
optimizer=optim.SGD(model.parameters(),lr=0.02)
for i in range(epoch):
  runlos=0
  for images,labels in trainloader:
    images=images.view(images.shape[0],-1)   #flatten the images
    optimizer.zero_grad()
    output=model.forward(images)
    loss=crite(output,labels)
    loss.backward()
    optimizer.step()
    runlos+=loss.item()
  else:
    testloss=0
    acc=0
    with torch.no_grad():
      model.eval()
      for images,labels in testloader:
        images=images.view(images.shape[0],-1)
        logps=model(images)
        testloss+=crite(logps,labels)
        ps=torch.exp(logps)
        topp,topclass=ps.topk(1,dim=1)
        equals= topclass==labels.view(*topclass.shape)
        acc+=torch.mean(equals.type(torch.FloatTensor))
    model.train()
    train_losses.append(runlos/len(trainloader))
    test_losses.append(testloss/len(testloader))
    print(f"Training Loss:{runlos/len(trainloader)}")
    print(f"Test Loss:{testloss/len(testloader)}")
    print(f"Accuracy:{acc/len(testloader)}")
```

```
ERROR! Session/line number was not unique in database. History logging moved to new sess
Training Loss:1.2209810409655195
Test Loss:0.44680145382881165
Accuracy:0.8693272471427917
Training Loss:0.36890021783075355
Test Loss:0.3240319788455963
Accuracy:0.9010748267173767
Training Loss:0.2840409790521174
Test Loss:0.23320476710796356
Accuracy:0.928244411945343
Training Loss:0.22735261429013856
Test Loss:0.20856468379497528
Accuracy:0.9368033409118652
Training Loss:0.1850831166946335
Test Loss:0.17273502051830292
Accuracy:0.944167971611023
Training Loss:0.15657212165222048
Test Loss:0.14685150980949402
Accuracy:0.9566082954406738
```