```
!pip install torch torchvision
import torch.nn as nn
import torch


class SimpleNN(nn.Module):
  def __init__(self):
    super(SimpleNN, self).__init__()
    self.fc1 = nn.Linear(784,128)
    self.fc2 = nn.Linear(128,64)
    self.fc3 = nn.Linear(64,10)

  def forward(self,x):
    x = torch.relu(self.fc1(x))
    x = torch.relu(self.fc2(x))
    x = torch.softmax(self.fc3(x), dim = 1)

    return x

model = SimpleNN()
print(model)

#TRAIN THE MODEL

import torch.optim as optim
import torch.nn.functional as F
from torchvision import datasets, transforms

transform = transforms.Compose([transforms.ToTensor()])
trainset = datasets.MNIST(root = './data', train = True, download = True, transform = transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size = 32, shuffle = True)

#define the loss function and optimizer

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr = 0.01)

#training loop

for epoch in range(1,6):
  running_loss = 0.0
  for images, labels in trainloader:

    #flatten images into vectors
    images = images.view(images.shape[0], -1)
    #zero the parameter gradients
    optimizer.zero_grad()
    #forward pass
    output = model(images)
    #compute loss
    loss = criterion(output, labels)

    #backward pass and optimisation
    loss.backward()
    optimizer.step()

    #update running loss
    running_loss += loss.item()

  print(f'Epoch {epoch}, Loss:{running_loss/len(trainloader)}')


##EVALUATE THE MODEL

import torch # Ensure torch is imported

testset = datasets.MNIST(root = './data', train = False, download = True, transform = transform)
testloader = torch.utils.data.DataLoader(testset, batch_size = 32, shuffle = True)

correct = 0
total = 0

# Use torch.no_grad() as a function call instead of a context manager
for images, labels in testloader:
  with torch.no_grad(): # Apply torch.no_grad() to each iteration
    images = images.view(images.shape[0], -1)
```

```
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print(f'Accuracy: {100* correct/total}%')


#FUNCTION TO DISPLAY BATCH OF IMAGES ALONG WITH PREDICTIONS
import matplotlib.pyplot as plt
import numpy as np

# Function to display a batch of test images along with predictions
def visualize_predictions(model, testloader):
    # Get a batch of test data
    dataiter = iter(testloader)
    images, labels = next(dataiter)

    # Flatten images for model input
    images_flattened = images.view(images.shape[0], -1)

    # Predict using the model
    with torch.no_grad():
        outputs = model(images_flattened)
        _, predicted = torch.max(outputs, 1)

    # Convert images to numpy for display
    images = images.numpy()

    # Display the first 10 images with predictions and actual labels
    fig, axes = plt.subplots(1, 10, figsize=(15, 4))
    for idx in range(10):
        ax = axes[idx]
        ax.imshow(images[idx].squeeze(), cmap='gray')
        ax.set_title(f"P: {predicted[idx].item()}\nT: {labels[idx].item()}")
        ax.axis('off')
    plt.tight_layout()
    plt.show()

# Call the visualization function
visualize_predictions(model, testloader)
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.5.1+cu121)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (0.20.1+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.16.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.12.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2024.10.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.10/dist-packages (from torch) (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) (1.
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision) (1.26.4)
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: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (3.0.2)
```

Start coding or generate with AI.