```
!pip install torch

Requirement already satisfied: torch in
/usr/local/lib/python3.10/dist-packages (2.1.0+cu121)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from torch) (3.13.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from torch) (4.9.0)
Requirement already satisfied: sympy in
/usr/local/lib/python3.10/dist-packages (from torch) (1.12)
Requirement already satisfied: networkx in
/usr/local/lib/python3.10/dist-packages (from torch) (3.2.1)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from torch) (3.1.3)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.10/dist-packages (from torch) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in
/usr/local/lib/python3.10/dist-packages (from torch) (2.1.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath>=0.19 in
/usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)

!pip install torchvision

Requirement already satisfied: torchvision in
/usr/local/lib/python3.10/dist-packages (0.16.0+cu121)
Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (from torchvision) (1.23.5)
Requirement already satisfied: requests in
/usr/local/lib/python3.10/dist-packages (from torchvision) (2.31.0)
Requirement already satisfied: torch==2.1.0 in
/usr/local/lib/python3.10/dist-packages (from torchvision)
(2.1.0+cu121)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/usr/local/lib/python3.10/dist-packages (from torchvision) (9.4.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0-
>torchvision) (3.13.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0-
>torchvision) (4.9.0)
Requirement already satisfied: sympy in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0-
>torchvision) (1.12)
Requirement already satisfied: networkx in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0-
>torchvision) (3.2.1)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0-
```

```
>torchvision) (3.1.3)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0-
>torchvision) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in
/usr/local/lib/python3.10/dist-packages (from torch==2.1.0-
>torchvision) (2.1.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->torchvision)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests->torchvision)
(3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->torchvision)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->torchvision)
(2024.2.2)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch==2.1.0-
>torchvision) (2.1.5)
Requirement already satisfied: mpmath>=0.19 in
/usr/local/lib/python3.10/dist-packages (from sympy->torch==2.1.0-
>torchvision) (1.3.0)
```

```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.48.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: numpy>=1.20 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.23.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in
```

```
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

```
'''from torchvision import datasets
from torchvision.transforms import ToTensor'''
```

```
{"type":"string"}
```

```
'''train_data = datasets.MNIST(
    root = 'data',
    train = True,
    transform = ToTensor(),
    download = True
)
test_data = datasets.MNIST(
    root = 'data',
    train = False,
    transform = ToTensor(),
    download = True
)'''
```

```
{"type":"string"}
```

```
'''train_data'''
```

```
{"type":"string"}
```

```
'''test_data'''
```

```
{"type":"string"}
```

```
'''train_data.data'''
```

```
{"type":"string"}
```

```
'''train_data.data.shape'''
```

```
{"type":"string"}
```

```
'''train_data.targets.size()'''
```

```
{"type":"string"}
```

```
'''train_data.targets'''
```

```
{"type":"string"}
```

```
'''from torch.utils.data import DataLoader

loaders = {
    'train' : DataLoader(train_data,
                         batch_size = 100,
                         shuffle = True,
```

```
                          num_workers =1),
    'test' : DataLoader(test_data,
                        batch_size = 100,
                        shuffle = True,
                        num_workers =1)
}'''
```

```
{"type":"string"}
```

```
'''loaders'''
```

```
{"type":"string"}
```

```
'''import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim



class CNN(nn.Module):

  def __init__(self):
    super(CNN, self).__init__()

    self.conv1 = nn.Conv2d(1,10,kernel_size=5)
    self.conv2 = nn.Conv2d(10,20,kernel_size=5)
    self.conv2_drop = nn.Dropout2d()
    self.fc1 = nn.Linear(320,50 )
    self.fc2 = nn.Linear(50,10)

  def forward(self,x):
    x = F.relu(F.max_pool2d(self.conv1(x),2))
    x = F.relu(F.max_pool2d(self.conv2_drop(self.conv2(x)),2))
    x = x.view(-1,320)
    x = F.relu(self.fc1(x))
    x = F.dropout(x,training = self.training)
    x = self.fc2(x)

    return F.softmax(x)'''
```

```
{"type":"string"}
```

```
'''import torch

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = CNN().to(device)

optimizer = optim.Adam(model.parameters(),lr =0.01)
loss_fn = nn.CrossEntropyLoss()

def train(epoch):
  model.train()
```

```
  for batch_idx , (data,target) in enumerate(loaders['train']):
    data,target = data.to(device),target.to(device)
    optimizer.zero_grad()
    output = model(data)
    loss = loss_fn(output,target)
    loss.backward()
    optimizer.step()
    if batch_idx % 20 == 0:
      print(f"Train Epoch: {epoch} [{batch_idx *
len(data)}/{len(loaders['train'].dataset)} ({100. * batch_idx /
len(loaders['train']):.0f}%)]\t{loss.item():.6f}")
def test():
  model.eval()
  test_loss = 0
  correct = 0
  with torch.no_grad():
    for data , target in loaders['test']:
      data , target = data.to(device) , target.to(device)
      output = model(data)
      test_loss += loss_fn(output,target).item()
      pred = output.argmax(dim = 1,keepdim = True)
      correct += pred.eq(target.view_as(pred)).sum().item()

  test_loss /= len(loaders['test'].dataset)
  print(f"\nTest set: Average loss: {test_loss:.4f}, Accuracy
{correct}/{len(loaders['test'].dataset)} ({100. *
correct/len(loaders['test'].dataset):.0f}%)\n")
'''
```

```
{"type":"string"}
```

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

# Define the CNN model
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(1, 32, kernel_size=3, stride=1,
padding=1)
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3, stride=1,
padding=1)
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)
        self.fc1 = nn.Linear(64 * 7 * 7, 128)
```

```
        self.fc2 = nn.Linear(128, 10)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 64 * 7 * 7)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Set up data transformations and loaders
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_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False)

# Initialize model, loss function, and optimizer
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = CNN().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-
ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-
ubyte.gz to ./data/MNIST/raw/train-images-idx3-ubyte.gz

100%|██████████| 9912422/9912422 [00:00<00:00, 159708788.52it/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
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-
ubyte.gz to ./data/MNIST/raw/train-labels-idx1-ubyte.gz

100%|██████████| 28881/28881 [00:00<00:00, 115257558.35it/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
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
to ./data/MNIST/raw/t10k-images-idx3-ubyte.gz

100%|██████████| 1648877/1648877 [00:00<00:00, 55428232.27it/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
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
to ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz

100%|██████████| 4542/4542 [00:00<00:00, 13824766.89it/s]

Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to
./data/MNIST/raw


# Training loop
num_epochs = 5
for epoch in range(num_epochs):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, target)
        loss.backward()
        optimizer.step()

        if batch_idx % 100 == 0:
            print(f'Train Epoch: {epoch}, Batch: {batch_idx}, Loss:
{loss.item():.4f}')

# Testing loop
model.eval()
correct = 0
total = 0

with torch.no_grad():
    for data, target in test_loader:
        data, target = data.to(device), target.to(device)
        outputs = model(data)
        _, predicted = torch.max(outputs.data, 1)
```

```
        total += target.size(0)
        correct += (predicted == target).sum().item()

accuracy = correct / total
print(f'Test Accuracy: {100 * accuracy:.2f}%')
```

```
Train Epoch: 0, Batch: 0, Loss: 2.2945
Train Epoch: 0, Batch: 100, Loss: 0.2725
Train Epoch: 0, Batch: 200, Loss: 0.1665
Train Epoch: 0, Batch: 300, Loss: 0.0686
Train Epoch: 0, Batch: 400, Loss: 0.1478
Train Epoch: 0, Batch: 500, Loss: 0.0138
Train Epoch: 0, Batch: 600, Loss: 0.0186
Train Epoch: 0, Batch: 700, Loss: 0.1274
Train Epoch: 0, Batch: 800, Loss: 0.0281
Train Epoch: 0, Batch: 900, Loss: 0.0270
Train Epoch: 1, Batch: 0, Loss: 0.0313
Train Epoch: 1, Batch: 100, Loss: 0.0503
Train Epoch: 1, Batch: 200, Loss: 0.0271
Train Epoch: 1, Batch: 300, Loss: 0.0282
Train Epoch: 1, Batch: 400, Loss: 0.0374
Train Epoch: 1, Batch: 500, Loss: 0.0341
Train Epoch: 1, Batch: 600, Loss: 0.0474
Train Epoch: 1, Batch: 700, Loss: 0.0132
Train Epoch: 1, Batch: 800, Loss: 0.1363
Train Epoch: 1, Batch: 900, Loss: 0.0275
Train Epoch: 2, Batch: 0, Loss: 0.0027
Train Epoch: 2, Batch: 100, Loss: 0.0029
Train Epoch: 2, Batch: 200, Loss: 0.0145
Train Epoch: 2, Batch: 300, Loss: 0.0537
Train Epoch: 2, Batch: 400, Loss: 0.0480
Train Epoch: 2, Batch: 500, Loss: 0.0051
Train Epoch: 2, Batch: 600, Loss: 0.0165
Train Epoch: 2, Batch: 700, Loss: 0.0145
Train Epoch: 2, Batch: 800, Loss: 0.0540
Train Epoch: 2, Batch: 900, Loss: 0.0388
Train Epoch: 3, Batch: 0, Loss: 0.0019
Train Epoch: 3, Batch: 100, Loss: 0.0471
Train Epoch: 3, Batch: 200, Loss: 0.0334
Train Epoch: 3, Batch: 300, Loss: 0.0114
Train Epoch: 3, Batch: 400, Loss: 0.0119
Train Epoch: 3, Batch: 500, Loss: 0.0153
Train Epoch: 3, Batch: 600, Loss: 0.0149
Train Epoch: 3, Batch: 700, Loss: 0.0049
Train Epoch: 3, Batch: 800, Loss: 0.0587
Train Epoch: 3, Batch: 900, Loss: 0.0426
Train Epoch: 4, Batch: 0, Loss: 0.0052
Train Epoch: 4, Batch: 100, Loss: 0.0181
Train Epoch: 4, Batch: 200, Loss: 0.0045
Train Epoch: 4, Batch: 300, Loss: 0.0072
```

```
Train Epoch: 4, Batch: 400, Loss: 0.0028
Train Epoch: 4, Batch: 500, Loss: 0.0049
Train Epoch: 4, Batch: 600, Loss: 0.0026
Train Epoch: 4, Batch: 700, Loss: 0.0181
Train Epoch: 4, Batch: 800, Loss: 0.0053
Train Epoch: 4, Batch: 900, Loss: 0.0453
Test Accuracy: 99.08%
```

```
import matplotlib.pyplot as plt
model.eval()
data , target = test_dataset[6]
data = data.unsqueeze(0).to(device)
output = model(data)
prediction = output.argmax(dim = 1 , keepdim = True).item()
print(f'Prediction: {prediction}')
```

```
Prediction: 4
```

```
'''for epoch in range(1,11):
  train(epoch)
  test()'''
```

```
{"type":"string"}
```

```
'''device'''
```

```
{"type":"string"}
```

```
'''import matplotlib.pyplot as plt
model.eval()
data , target = test_data[1]
data = data.unsqueeze(0).to(device)
output = model(data)
prediction = output.argmax(dim =1,keepdim = True).item()
print(f'Prediction: {prediction}')
image = data.squeeze(0).squeeze(0).cpu().numpy()
plt.imshow(image,cmap='gray')
plt.show()'''
```

```
{"type":"string"}
```