BHARAT INTERN TASK-2

NUMBER RECOGNITION

BY - Gurshaan Singh

Handwritten digit recognition using MNIST dataset to detect the scanned images of handwritten digits

```
import torch
import torchvision
from torch import nn
import torch.optim as optim
import torchvision.transforms as transforms
import matplotlib.pyplot as plt
from tqdm.notebook import tqdm
```

Load the MNIST dataset

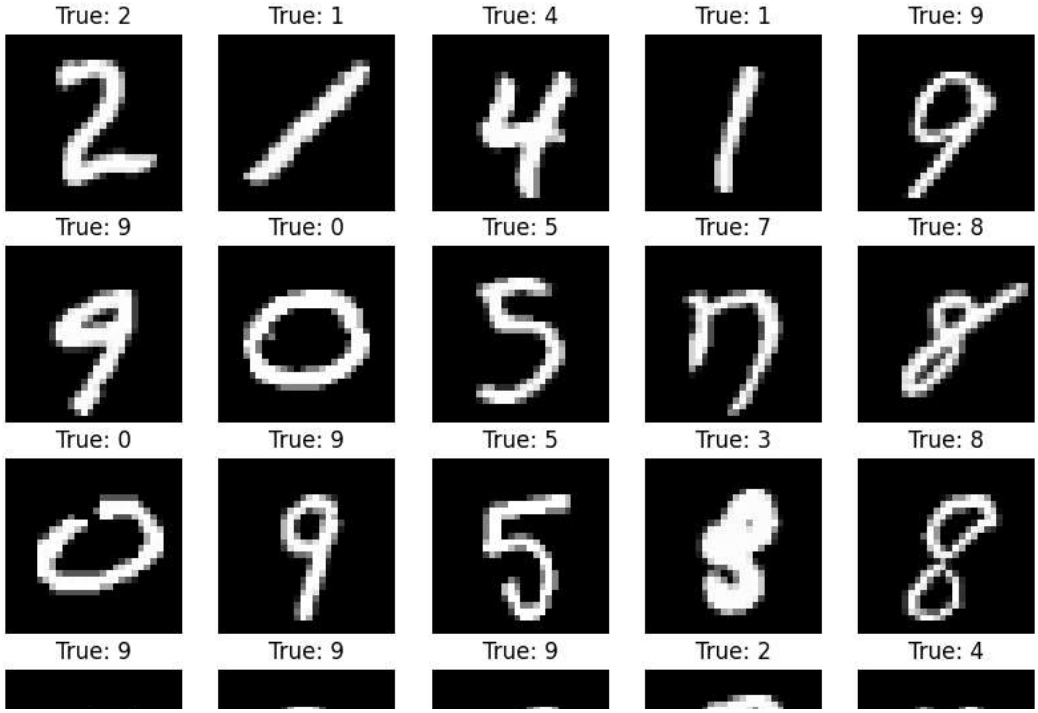
```
trainset = torchvision.datasets.MNIST(root='./data', train=True, download=True, transform=transforms.ToTensor())
trainloader = torch.utils.data.DataLoader(trainset, batch_size=64, shuffle=True)

testset = torchvision.datasets.MNIST(root='./data', train=False, download=True, transform=transforms.ToTensor())
testloader = torch.utils.data.DataLoader(testset, batch_size=64, shuffle=False)
```

```
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, 392105319.57it/s]Extracting ./data/MNIST/raw/train-images-idx3-ubyte.gz to ./data/MN


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, 29902664.48it/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, 193267700.55it/s]Extracting ./data/MNIST/raw/t10k-images-idx3-ubyte.gz to ./data/MNI

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, 16710990.15it/s]
Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw
```

Visualize Dataset

```
def show_images(images, labels, label_header="True"):
    figure = plt.figure(figsize=(10, 10))
    rows, cols = 5, 5
    for i in range(1, rows*cols+1):
        figure.add_subplot(rows, cols, i)
        plt.axis(False)
        plt.title(f"{label_header}: {labels[i-1].item()}")
        plt.imshow(images[i-1].permute(1,2,0), cmap='gray')

    plt.show()
```

```
images, labels = next(iter(trainloader))
show_images(images, labels)
```



Defining CNN model

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
        )

        self.fc = nn.Sequential(
            nn.Flatten(),
            nn.Linear(64 * 7 * 7, 128),
            nn.Linear(128, 10)
        )

    def forward(self, x):
        x = self.conv(x)
        x = self.fc(x)

        return x
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Initializing the model

```
net = CNN().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

Training the CNN Model

```
for epoch in range(5):
    running_loss = 0.0
    for idx, (inputs, labels) in tqdm(enumerate(trainloader), total=len(trainloader)):

        inputs = inputs.to(device)
        labels = labels.to(device)

        optimizer.zero_grad()

        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
```

```
        optimizer.step()

        running_loss += loss.item()

    print(f'Epoch: {epoch + 1}, Loss: {running_loss / 100:.3f}')

print('Training is done!')
```

```
100%                                                 938/938 [01:11<00:00, 10.23it/s]
Epoch: 1, Loss: 8.940
100%                                                 938/938 [01:12<00:00, 14.74it/s]
Epoch: 2, Loss: 2.734
100%                                                 938/938 [01:11<00:00, 14.48it/s]
Epoch: 3, Loss: 1.921
100%                                                 938/938 [01:11<00:00, 15.58it/s]
Epoch: 4, Loss: 1.426
100%                                                 938/938 [01:11<00:00, 14.80it/s]
Epoch: 5, Loss: 1.133
Training is done!
```

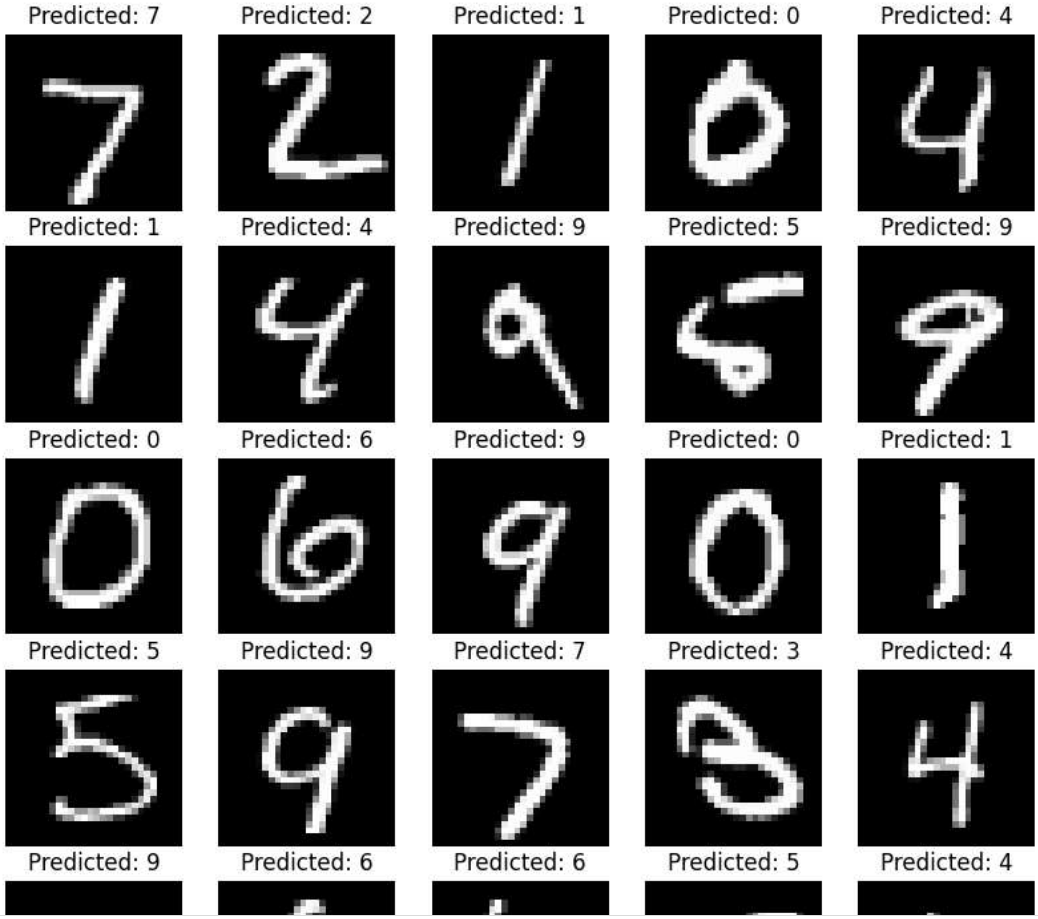
Evaluating the CNN Model on test data

```
correct = 0
total = 0
with torch.no_grad():
    for (images, labels) in testloader:

        images = images.to(device)
        labels = labels.to(device)

        outputs = net(images)
        predicted = outputs.argmax(1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

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

```
Accuracy on test set: 97.12%
```

```
images, labels = next(iter(testloader))
with torch.no_grad():
    pred = net(images.to(device))
    pred = pred.argmax(1)
```

```
show_images(images, pred, "Predicted")
```



✓ 2s completed at 3:17 PM