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


class MNISTWithBoundingBoxes:
    def __init__(self, train=True):
        self.dataset = datasets.MNIST(
            root='./data',
            train=train,
            download=True,
            transform=transforms.ToTensor()
        )

    def __getitem__(self, idx):
        img, label = self.dataset[idx]
        img_np = img.squeeze(0).numpy()  # Convert to numpy for bbox calculation

        # Calculate the bounding box for the digit
        rows, cols = np.where(img_np > 0)
        y_min, x_min = rows.min(), cols.min()
        y_max, x_max = rows.max(), cols.max()

        # Normalize the bounding box coordinates
        bbox = torch.tensor([x_min / 28, y_min / 28, x_max / 28, y_max / 28], dtype=torch.float32)

        return img, label, bbox

    def __len__(self):
        return len(self.dataset)
 # Initialize DataLoaders
train_dataset = MNISTWithBoundingBoxes(train=True)
test_dataset = MNISTWithBoundingBoxes(train=False)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-idx3-ubyte.gz to ./data/MNIST/raw/train-images-idx3-ubyte.gz
100%|██████████| 9.91M/9.91M [00:00<00:00, 88.3MB/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
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-idx1-ubyte.gz to ./data/MNIST/raw/train-labels-idx1-ubyte.gz
100%|██████████| 28.9k/28.9k [00:00<00:00, 14.2MB/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

Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw/t10k-images-idx3-ubyte.gz
100%|██████████| 1.65M/1.65M [00:00<00:00, 55.0MB/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
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz
100%|██████████| 4.54k/4.54k [00:00<00:00, 1.49MB/s]Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw
```

Start coding or generate with AI.

3. *Define* the Localization Model

```
class LocalizationModel(nn.Module):
    def __init__(self):
        super(LocalizationModel, self).__init__()
        self.backbone = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),
            nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2, 2)
        )
        self.fc = nn.Sequential(
            nn.Flatten(),
            nn.Linear(32 * 7 * 7, 128),
            nn.ReLU(),
            nn.Linear(128, 4)  # 4 outputs: [x_min, y_min, x_max, y_max]
        )

    def forward(self, x):
        features = self.backbone(x)
        bbox = self.fc(features)
        return bbox
 # Initialize model
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = LocalizationModel().to(device)


criterion = nn.MSELoss()  # Mean Squared Error for bounding box regression
optimizer = optim.Adam(model.parameters(), lr=0.001)


epochs = 5
for epoch in range(epochs):
  model.train()
  total_loss = 0
  for imgs, _, bboxes in train_loader:
      imgs, bboxes = imgs.to(device), bboxes.to(device)

      optimizer.zero_grad()
      pred_bboxes = model(imgs)
      loss = criterion(pred_bboxes, bboxes)
      loss.backward()
      optimizer.step()

      total_loss += loss.item()

      print(f"Epoch [{epoch+1}/{epochs}], Loss: {total_loss / len(train_loader):.4f}")
```

```
Epoch [5/5], Loss: 0.0001
```

```
model.eval()
with torch.no_grad():
  for imgs, _, bboxes in test_loader:
      imgs, bboxes = imgs.to(device), bboxes.to(device)
      pred_bboxes = model(imgs)
      print("Predicted BBox:", pred_bboxes[0].cpu().numpy())
      print("Ground Truth BBox:", bboxes[0].cpu().numpy())
      break
```

```
Predicted BBox: [0.20574346 0.25867143 0.7338158  0.93095803]
Ground Truth BBox: [0.21428572 0.25       0.75       0.9285714 ]
```