T0_MNIST_Classifer

October 28, 2025

[ ]:
```python
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
from torch.utils.data import DataLoader

print(f"PyTorch  : {torch.__version__}")
print(f"torchvision  : {torchvision.__version__}")

#
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

#

BATCH_SIZE = 64

#   MNIST

train_dataset = torchvision.datasets.MNIST(
    root='../foundation/data', train=True, transform=transform, download=True
)
test_dataset = torchvision.datasets.MNIST(
    root='../foundation/data', train=False, transform=transform, download=True
)

#
train_loader = DataLoader(
    dataset=train_dataset, batch_size=BATCH_SIZE, shuffle=True, num_workers=2
)
test_loader = DataLoader(
    dataset=test_dataset, batch_size=BATCH_SIZE, shuffle=False, num_workers=2
)
```

```
print(f"    ")
print(f"    : {len(train_loader)},     : {len(test_loader)}")

#
inputs, labels = next(iter(train_loader))
print(f"     : {inputs.shape}")
print(f"     : {labels.shape}")
```

```
PyTorch  : 2.5.1
torchvision  : 0.20.1
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 404: Not Found

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 ../foundation/data/MNIST/raw/train-images-idx3-ubyte.gz

100.0%

Extracting ../foundation/data/MNIST/raw/train-images-idx3-ubyte.gz to
../foundation/data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Failed to download (trying next):
HTTP Error 404: Not Found

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 ../foundation/data/MNIST/raw/train-labels-idx1-ubyte.gz

100.0%

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

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 404: Not Found

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 ../foundation/data/MNIST/raw/t10k-images-idx3-ubyte.gz

100.0%

Extracting ../foundation/data/MNIST/raw/t10k-images-idx3-ubyte.gz to
../foundation/data/MNIST/raw
```

```
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Failed to download (trying next):
HTTP Error 404: Not Found

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 ../foundation/data/MNIST/raw/t10k-labels-idx1-ubyte.gz

100.0%

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


   : 938,     : 157


     : torch.Size([64, 1, 28, 28])
     : torch.Size([64])
```

1

[ ]:
```python
#
if torch.backends.mps.is_available() and torch.backends.mps.is_built():
    device = torch.device("mps")
    print("  Apple Silicon GPU (MPS)")
else:
    device = torch.device("cpu")
    print("  CPU")

#    f(x; theta) --
class SimpleCNN(nn.Module):
    def __init__(self):
        super(SimpleCNN, self).__init__()
        self.conv_stack = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2), # 28x28 -> 14x14
            nn.Conv2d(16, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2) # 14x14 -> 7x7
        )
        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(32*7*7, 128), # 32*7*7 = 1568
            nn.ReLU(),
```

```
            nn.Linear(128, 10) # 10
        )

    def forward(self, x):
        x = self.conv_stack(x)
        x = self.classifier(x)
        return x #   logits

#
model = SimpleCNN().to(device)
criterion = nn.CrossEntropyLoss() #
optimizer = optim.Adam(model.parameters(), lr=0.001) # Adam

print(f" ,          {device}  ")
```

```
  Apple Silicon GPU (MPS)
"  "     mps
```

2

[ ]:
```
#      (Epochs)
NUM_EPOCHS = 5

print(f"---     {NUM_EPOCHS}    ---")

for epoch in range(NUM_EPOCHS):

    # ---    ---
    model.train()
    running_loss = 0.0

    for i, (inputs, labels) in enumerate(train_loader):
        inputs, labels = inputs.to(device), labels.to(device)

        #
        optimizer.zero_grad()      # 1.
        outputs = model(inputs)    # 2.
        loss = criterion(outputs, labels) # 3.
        loss.backward()            # 4.
        optimizer.step()           # 5.
        # ==========================

        running_loss += loss.item()
        if (i + 1) % 200 == 0:
            print(f'Epoch [{epoch+1}/{NUM_EPOCHS}],   [{i+1}/
 ↪{len(train_loader)}],  : {running_loss / 200:.4f}')
            running_loss = 0.0
```

```
    # ---    ---
    model.eval()
    correct = 0
    total = 0

    with torch.no_grad(): #
        for images, labels in test_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    print(f'--- Epoch {epoch+1}    ---')
    print(f'   : {accuracy:.2f} %')
    print(f'--------------------------------')

print('---    ---')
```

```
---     5    ---
Epoch [1/5],    [200/938],   : 0.5381
Epoch [1/5],    [400/938],   : 0.1353
Epoch [1/5],    [600/938],   : 0.0953
Epoch [1/5],    [800/938],   : 0.0751
--- Epoch 1    ---
   : 98.19 %
--------------------------------
Epoch [2/5],    [200/938],   : 0.0614
Epoch [2/5],    [400/938],   : 0.0542
Epoch [2/5],    [600/938],   : 0.0515
Epoch [2/5],    [800/938],   : 0.0483
--- Epoch 2    ---
   : 98.73 %
--------------------------------
Epoch [3/5],    [200/938],   : 0.0459
Epoch [3/5],    [400/938],   : 0.0339
Epoch [3/5],    [600/938],   : 0.0412
Epoch [3/5],    [800/938],   : 0.0350
--- Epoch 3    ---
   : 98.59 %
--------------------------------
Epoch [4/5],    [200/938],   : 0.0263
Epoch [4/5],    [400/938],   : 0.0313
Epoch [4/5],    [600/938],   : 0.0263
Epoch [4/5],    [800/938],   : 0.0312
--- Epoch 4    ---
```

```
   : 98.80 %
--------------------------------
Epoch [5/5],   [200/938],  : 0.0234
Epoch [5/5],   [400/938],  : 0.0238
Epoch [5/5],   [600/938],  : 0.0227
Epoch [5/5],   [800/938],  : 0.0182
--- Epoch 5    ---
   : 99.04 %
--------------------------------
---    ---
```