homework9

November 3, 2024

```
[ ]: import os

     os.environ['http_proxy'] = "http://squid.cs.ait.ac.th:3128/"
     os.environ['https_proxy'] = "http://squid.cs.ait.ac.th:3128/"
```

```
[ ]: !pip install torch==1.13.1+cu116 torchvision==0.14.1+cu116 torchaudio==0.13.1␣
      ↪--extra-index-url https://download.pytorch.org/whl/cu116 --user
```

```
Looking in indexes: https://pypi.org/simple,
https://download.pytorch.org/whl/cu116
Collecting torch==1.13.1+cu116
  Downloading https://download.pytorch.org/whl/cu116/torch-1.13.1%2Bcu116-cp39-c
p39-linux_x86_64.whl (1977.9 MB)
     |                  | 1977.9 MB 672 bytes/s
| 1288.4 MB 77.0 MB/s eta 0:00:09
Collecting torchvision==0.14.1+cu116
  Downloading https://download.pytorch.org/whl/cu116/torchvision-0.14.1%2Bcu116-
cp39-cp39-linux_x86_64.whl (24.2 MB)
     |                  | 24.2 MB 15 kB/s
Collecting torchaudio==0.13.1
  Downloading https://download.pytorch.org/whl/cu116/torchaudio-0.13.1%2Bcu116-c
p39-cp39-linux_x86_64.whl (4.2 MB)
     |                  | 4.2 MB 62.2 MB/s
Requirement already satisfied: typing-extensions in
/opt/conda/lib/python3.9/site-packages (from torch==1.13.1+cu116) (4.12.2)
Requirement already satisfied: requests in /opt/conda/lib/python3.9/site-
packages (from torchvision==0.14.1+cu116) (2.27.1)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/opt/conda/lib/python3.9/site-packages (from torchvision==0.14.1+cu116) (8.4.0)
Requirement already satisfied: numpy in /opt/conda/lib/python3.9/site-packages
(from torchvision==0.14.1+cu116) (1.21.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.9/site-packages (from
requests->torchvision==0.14.1+cu116) (1.26.8)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.9/site-packages (from
requests->torchvision==0.14.1+cu116) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.9/site-
```

```
packages (from requests->torchvision==0.14.1+cu116) (3.3)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/opt/conda/lib/python3.9/site-packages (from
requests->torchvision==0.14.1+cu116) (2.0.10)
Installing collected packages: torch, torchvision, torchaudio
  WARNING: The scripts convert-caffe2-to-onnx, convert-onnx-to-caffe2 and

torchrun are installed in '/home/st125457/.local/bin' which is not on PATH.

  Consider adding this directory to PATH or, if you prefer to suppress this

warning, use --no-warn-script-location.
Successfully installed torch-1.13.1+cu116 torchaudio-0.13.1+cu116
torchvision-0.14.1+cu116
```

[ ]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F

from torchvision import datasets, transforms
from torch.utils.data import DataLoader
```

[ ]:
```
torch.cuda.is_available()
```

[ ]:
```
True
```

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

mnist_train = datasets.MNIST(root='./data', train=True, download=True,␣
 ↪transform=transform)
mnist_test = datasets.MNIST(root='./data', train=False, download=True,␣
 ↪transform=transform)

mnist_train_loader = DataLoader(mnist_train, batch_size=64, shuffle=True)
mnist_test_loader = DataLoader(mnist_test, batch_size=64, shuffle=False)

fashion_train = datasets.FashionMNIST(root='./data', train=True, download=True,␣
 ↪transform=transform)
fashion_test = datasets.FashionMNIST(root='./data', train=False, download=True,␣
 ↪transform=transform)

fashion_train_loader = DataLoader(fashion_train, batch_size=64, shuffle=True)
fashion_test_loader = DataLoader(fashion_test, batch_size=64, shuffle=False)
```

[ ]:
```
class CNNModel(nn.Module):
    def __init__(self, num_classes=10):
        super(CNNModel, self).__init__()

        self.conv1 = nn.Conv2d(1, 32, kernel_size=3) # 28x28x1 -> 26x26x32 / 2␣
↪= 13x13x32
        self.conv2 = nn.Conv2d(32, 64, kernel_size=3)# 11x11x64 -> 5x5x64
        self.pool = nn.MaxPool2d(2)
        self.fc1 = nn.Linear(64 * 5 * 5, 128)
        self.dropout = nn.Dropout(0.5)
        self.fc2 = nn.Linear(128, num_classes)

    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 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)

        return x
```

```
[ ]: def train(model, device, train_loader, optimizer, criterion, epoch):
    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_idx * len(data)}/
↪{len(train_loader.dataset)}] Loss: {loss.item():.6f}')

def test(model, device, test_loader, criterion):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += criterion(output, target).item()
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()
    test_loss /= len(test_loader.dataset)
    accuracy = 100. * correct / len(test_loader.dataset)
    print(f'\nTest set: Average loss: {test_loss:.4f}, Accuracy: {correct}/
 ↪{len(test_loader.dataset)} ({accuracy:.2f}%)\n')
    return accuracy
```

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

base_model = CNNModel().to(device)
optimizer = torch.optim.Adam(base_model.parameters(), lr=0.001)
criterion = nn.CrossEntropyLoss()
```

[ ]:
```
for epoch in range(1, 6):
    train(base_model, device, mnist_train_loader, optimizer, criterion, epoch)
    test(base_model, device, mnist_test_loader, criterion)
```

```
Train Epoch: 1 [0/60000] Loss: 2.298948
Train Epoch: 1 [6400/60000] Loss: 0.356841
Train Epoch: 1 [12800/60000] Loss: 0.360970
Train Epoch: 1 [19200/60000] Loss: 0.138276
Train Epoch: 1 [25600/60000] Loss: 0.225226
Train Epoch: 1 [32000/60000] Loss: 0.147682
Train Epoch: 1 [38400/60000] Loss: 0.069759
Train Epoch: 1 [44800/60000] Loss: 0.169684
Train Epoch: 1 [51200/60000] Loss: 0.045199
Train Epoch: 1 [57600/60000] Loss: 0.116820

Test set: Average loss: 0.0007, Accuracy: 9851/10000 (98.51%)

Train Epoch: 2 [0/60000] Loss: 0.111176
Train Epoch: 2 [6400/60000] Loss: 0.076573
Train Epoch: 2 [12800/60000] Loss: 0.093073
Train Epoch: 2 [19200/60000] Loss: 0.027703
Train Epoch: 2 [25600/60000] Loss: 0.104041
Train Epoch: 2 [32000/60000] Loss: 0.360858
Train Epoch: 2 [38400/60000] Loss: 0.153771
Train Epoch: 2 [44800/60000] Loss: 0.057919
Train Epoch: 2 [51200/60000] Loss: 0.017944
Train Epoch: 2 [57600/60000] Loss: 0.099698

Test set: Average loss: 0.0006, Accuracy: 9881/10000 (98.81%)

Train Epoch: 3 [0/60000] Loss: 0.028135
Train Epoch: 3 [6400/60000] Loss: 0.016654
Train Epoch: 3 [12800/60000] Loss: 0.035037
Train Epoch: 3 [19200/60000] Loss: 0.069002
Train Epoch: 3 [25600/60000] Loss: 0.038148
Train Epoch: 3 [32000/60000] Loss: 0.059592
Train Epoch: 3 [38400/60000] Loss: 0.055264
Train Epoch: 3 [44800/60000] Loss: 0.173720
Train Epoch: 3 [51200/60000] Loss: 0.018039
Train Epoch: 3 [57600/60000] Loss: 0.034520

Test set: Average loss: 0.0005, Accuracy: 9897/10000 (98.97%)

Train Epoch: 4 [0/60000] Loss: 0.045303
Train Epoch: 4 [6400/60000] Loss: 0.034586
Train Epoch: 4 [12800/60000] Loss: 0.076844
Train Epoch: 4 [19200/60000] Loss: 0.080042
Train Epoch: 4 [25600/60000] Loss: 0.009379
Train Epoch: 4 [32000/60000] Loss: 0.292314
Train Epoch: 4 [38400/60000] Loss: 0.080148
Train Epoch: 4 [44800/60000] Loss: 0.029393
Train Epoch: 4 [51200/60000] Loss: 0.119722
Train Epoch: 4 [57600/60000] Loss: 0.002480

Test set: Average loss: 0.0004, Accuracy: 9905/10000 (99.05%)

Train Epoch: 5 [0/60000] Loss: 0.046421
Train Epoch: 5 [6400/60000] Loss: 0.017639
Train Epoch: 5 [12800/60000] Loss: 0.014149
Train Epoch: 5 [19200/60000] Loss: 0.070688
Train Epoch: 5 [25600/60000] Loss: 0.037091
Train Epoch: 5 [32000/60000] Loss: 0.104506
Train Epoch: 5 [38400/60000] Loss: 0.099262
Train Epoch: 5 [44800/60000] Loss: 0.061198
Train Epoch: 5 [51200/60000] Loss: 0.008817
Train Epoch: 5 [57600/60000] Loss: 0.010695

Test set: Average loss: 0.0004, Accuracy: 9911/10000 (99.11%)
```

[ ]:
```python
transfer_model = CNNModel().to(device)
transfer_model.load_state_dict(base_model.state_dict())
```

[ ]:
```
<All keys matched successfully>
```

[ ]:
```python
for param in transfer_model.parameters():
    # print(param)
    param.requires_grad = False
```

[ ]:
```python
transfer_model.fc2 = nn.Linear(128, 10, device=device)
transfer_model.fc2.requires_grad = True
```

[ ]:
```python
optimizer = torch.optim.Adam(transfer_model.fc2.parameters(), lr=0.001)

for epoch in range(1, 6):
    train(transfer_model, device, fashion_train_loader, optimizer, criterion,␣
 ↪epoch)
    test(transfer_model, device, fashion_test_loader, criterion)

test_accuracy = test(transfer_model, device, fashion_test_loader, criterion)
print(f'Final Test Accuracy on Fashion MNIST: {test_accuracy:.2f}%')
```

```
Train Epoch: 1 [0/60000] Loss: 2.631480
Train Epoch: 1 [6400/60000] Loss: 1.386769
Train Epoch: 1 [12800/60000] Loss: 1.131358
Train Epoch: 1 [19200/60000] Loss: 1.324272
Train Epoch: 1 [25600/60000] Loss: 0.970456
Train Epoch: 1 [32000/60000] Loss: 1.211585
Train Epoch: 1 [38400/60000] Loss: 1.112245
Train Epoch: 1 [44800/60000] Loss: 1.074822
Train Epoch: 1 [51200/60000] Loss: 1.256624
Train Epoch: 1 [57600/60000] Loss: 1.088293

Test set: Average loss: 0.0153, Accuracy: 6852/10000 (68.52%)

Train Epoch: 2 [0/60000] Loss: 0.893758
Train Epoch: 2 [6400/60000] Loss: 1.107784
Train Epoch: 2 [12800/60000] Loss: 1.112038
Train Epoch: 2 [19200/60000] Loss: 1.207096
Train Epoch: 2 [25600/60000] Loss: 0.996716
Train Epoch: 2 [32000/60000] Loss: 0.967878
Train Epoch: 2 [38400/60000] Loss: 1.060265
Train Epoch: 2 [44800/60000] Loss: 1.056669
Train Epoch: 2 [51200/60000] Loss: 1.119566
Train Epoch: 2 [57600/60000] Loss: 1.296211

Test set: Average loss: 0.0145, Accuracy: 6984/10000 (69.84%)

Train Epoch: 3 [0/60000] Loss: 0.989445
Train Epoch: 3 [6400/60000] Loss: 0.921946
Train Epoch: 3 [12800/60000] Loss: 0.917456
Train Epoch: 3 [19200/60000] Loss: 1.124455
Train Epoch: 3 [25600/60000] Loss: 1.412250
Train Epoch: 3 [32000/60000] Loss: 1.195006
Train Epoch: 3 [38400/60000] Loss: 0.932100
Train Epoch: 3 [44800/60000] Loss: 1.069095
Train Epoch: 3 [51200/60000] Loss: 0.989817
Train Epoch: 3 [57600/60000] Loss: 1.105925

Test set: Average loss: 0.0143, Accuracy: 6985/10000 (69.85%)
```

```
Train Epoch: 4 [0/60000] Loss: 1.176192
Train Epoch: 4 [6400/60000] Loss: 1.124177
Train Epoch: 4 [12800/60000] Loss: 1.030968
Train Epoch: 4 [19200/60000] Loss: 1.331551
Train Epoch: 4 [25600/60000] Loss: 1.069659
Train Epoch: 4 [32000/60000] Loss: 0.930640
Train Epoch: 4 [38400/60000] Loss: 1.038912
Train Epoch: 4 [44800/60000] Loss: 1.022467
Train Epoch: 4 [51200/60000] Loss: 0.969537
Train Epoch: 4 [57600/60000] Loss: 1.094155

Test set: Average loss: 0.0142, Accuracy: 7021/10000 (70.21%)

Train Epoch: 5 [0/60000] Loss: 1.046914
Train Epoch: 5 [6400/60000] Loss: 1.237893
Train Epoch: 5 [12800/60000] Loss: 1.291262
Train Epoch: 5 [19200/60000] Loss: 1.210085
Train Epoch: 5 [25600/60000] Loss: 1.310768
Train Epoch: 5 [32000/60000] Loss: 1.247080
Train Epoch: 5 [38400/60000] Loss: 0.831143
Train Epoch: 5 [44800/60000] Loss: 0.880109
Train Epoch: 5 [51200/60000] Loss: 1.120324
Train Epoch: 5 [57600/60000] Loss: 1.163934
```

[ ]:

[ ]:

[ ]: