CSC 561: Neural Networks and Deep Learning

PyTorch example

Marco Alvarez

Department of Computer Science and Statistics
University of Rhode Island

Spring 2024


```
        # define constants (hyperparameters)
        device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
        b_size = 64
        l_rate = 0.0005
        n_epochs = 5
```

```
# define the transformations
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])

# load the MNIST data
train_data = datasets.MNIST('~/.cache/', train=True, download=True, transform=transform)
test_data = datasets.MNIST('~/.cache/', train=False, download=True, transform=transform)

# create data loaders
train_loader = DataLoader(train_data, batch_size=b_size, shuffle=True)
test_loader = DataLoader(test_data, batch_size=len(test_data), shuffle=False)
```



```
class MLP(torch.nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.fc1 = Linear(28*28, 512)
        self.d1 = Dropout(0.4)
        self.fc2 = Linear(512, 512)
        self.d2 = Dropout(0.4)
        self.fc3 = Linear(512, 10)

    def forward(self, x):
        x = x.view(-1, 28*28)
        x = relu(self.fc1(x))
        x = self.d1(x)
        x = relu(self.fc2(x))
        x = self.d2(x)
        # note softmax is not used explicitly here
        x = self.fc3(x)
        return x
```



```
def train(model, device, loader, opt, epoch, log_every=1):
    model.train()

    for batch_idx, (data, target) in enumerate(loader):
        data, target = data.to(device), target.to(device)
        opt.zero_grad()
        output = model(data)
        # cross entropy loss between input logits and target
        # note predictions are not a probability distribution
        loss = cross_entropy(output, target)

        loss.backward()
        opt.step()

        if batch_idx % log_every == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(loader.dataset),
                100. * batch_idx / len(loader), loss.item()))
```

```python
def test(model, device, loader):
    model.eval()
    test_loss = 0
    correct = 0

    with torch.no_grad():
        for data, target in loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += cross_entropy(output, target, reduction='sum').item()

            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()
    test_loss /= len(loader.dataset)

    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
        test_loss, correct, len(loader.dataset),
        100. * correct / len(loader.dataset)))
```



```python
model = MLP().to(device)
optimizer = AdamW(model.parameters(), lr=l_rate)

for epoch in range(n_epochs):
    train(model, device, train_loader, optimizer, epoch, log_every=100)
    test(model, device, test_loader)
```

```
Train Epoch: 0 [0/60000 (0%)]    Loss: 2.330892
Train Epoch: 0 [6400/60000 (11%)]     Loss: 0.301402
Train Epoch: 0 [12800/60000 (21%)]    Loss: 0.398661
Train Epoch: 0 [19200/60000 (32%)]    Loss: 0.162983
Train Epoch: 0 [25600/60000 (43%)]    Loss: 0.218468
Train Epoch: 0 [32000/60000 (53%)]    Loss: 0.168886
Train Epoch: 0 [38400/60000 (64%)]    Loss: 0.204717
Train Epoch: 0 [44800/60000 (75%)]    Loss: 0.111907
Train Epoch: 0 [51200/60000 (85%)]    Loss: 0.347752
Train Epoch: 0 [57600/60000 (96%)]    Loss: 0.232287

Test set: Average loss: 0.1187, Accuracy: 9632/10000 (96%)

Train Epoch: 1 [0/60000 (0%)]    Loss: 0.157370
Train Epoch: 1 [6400/60000 (11%)]     Loss: 0.125208
Train Epoch: 1 [12800/60000 (21%)]    Loss: 0.093008
Train Epoch: 1 [19200/60000 (32%)]    Loss: 0.061850
Train Epoch: 1 [25600/60000 (43%)]    Loss: 0.051277
Train Epoch: 1 [32000/60000 (53%)]    Loss: 0.132460
Train Epoch: 1 [38400/60000 (64%)]    Loss: 0.053607
Train Epoch: 1 [44800/60000 (75%)]    Loss: 0.041962
Train Epoch: 1 [51200/60000 (85%)]    Loss: 0.129111
Train Epoch: 1 [57600/60000 (96%)]    Loss: 0.136489

Test set: Average loss: 0.0928, Accuracy: 9697/10000 (97%)

Train Epoch: 2 [0/60000 (0%)]    Loss: 0.214178
Train Epoch: 2 [6400/60000 (11%)]     Loss: 0.094175
Train Epoch: 2 [12800/60000 (21%)]    Loss: 0.170777
Train Epoch: 2 [19200/60000 (32%)]    Loss: 0.083292
Train Epoch: 2 [25600/60000 (43%)]    Loss: 0.047731
Train Epoch: 2 [32000/60000 (53%)]    Loss: 0.073823
Train Epoch: 2 [38400/60000 (64%)]    Loss: 0.123365
Train Epoch: 2 [44800/60000 (75%)]    Loss: 0.078627
Train Epoch: 2 [51200/60000 (85%)]    Loss: 0.111702
Train Epoch: 2 [57600/60000 (96%)]    Loss: 0.093963

Test set: Average loss: 0.0772, Accuracy: 9761/10000 (98%)

Train Epoch: 3 [0/60000 (0%)]    Loss: 0.027528
Train Epoch: 3 [6400/60000 (11%)]     Loss: 0.069929
Train Epoch: 3 [12800/60000 (21%)]    Loss: 0.118869
Train Epoch: 3 [19200/60000 (32%)]    Loss: 0.097563
Train Epoch: 3 [25600/60000 (43%)]    Loss: 0.052691
Train Epoch: 3 [32000/60000 (53%)]    Loss: 0.027609
Train Epoch: 3 [38400/60000 (64%)]    Loss: 0.183863
Train Epoch: 3 [44800/60000 (75%)]    Loss: 0.154557
Train Epoch: 3 [51200/60000 (85%)]    Loss: 0.050854
Train Epoch: 3 [57600/60000 (96%)]    Loss: 0.052494

Test set: Average loss: 0.0694, Accuracy: 9797/10000 (98%)

Train Epoch: 4 [0/60000 (0%)]    Loss: 0.081912
Train Epoch: 4 [6400/60000 (11%)]     Loss: 0.045535
Train Epoch: 4 [12800/60000 (21%)]    Loss: 0.030686
Train Epoch: 4 [19200/60000 (32%)]    Loss: 0.054139
Train Epoch: 4 [25600/60000 (43%)]    Loss: 0.077830
Train Epoch: 4 [32000/60000 (53%)]    Loss: 0.019547
Train Epoch: 4 [38400/60000 (64%)]    Loss: 0.029897
Train Epoch: 4 [44800/60000 (75%)]    Loss: 0.104272
Train Epoch: 4 [51200/60000 (85%)]    Loss: 0.244031
Train Epoch: 4 [57600/60000 (96%)]    Loss: 0.063373

Test set: Average loss: 0.0683, Accuracy: 9795/10000 (98%)
```