HW4DLP1

May 3, 2025

[1]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,)),
    transforms.Lambda(lambda x: x.view(-1))])

train_data = torchvision.datasets.MNIST(
    root='data', train=True, download=True, transform=transform)

test_data = torchvision.datasets.MNIST(
    root='data', train=False, download=True, transform=transform)

train_loader = torch.utils.data.DataLoader(train_data,␣
 ↪batch_size=128,shuffle=True,num_workers=4, pin_memory=True)

test_loader = torch.utils.data.DataLoader(
    test_data, batch_size=1000, shuffle=False)
```

```
100%|      | 9.91M/9.91M [00:00<00:00, 17.7MB/s]
100%|      | 28.9k/28.9k [00:00<00:00, 486kB/s]
100%|      | 1.65M/1.65M [00:00<00:00, 4.50MB/s]
100%|      | 4.54k/4.54k [00:00<00:00, 8.78MB/s]
/usr/local/lib/python3.11/dist-packages/torch/utils/data/dataloader.py:624:
UserWarning: This DataLoader will create 4 worker processes in total. Our
suggested max number of worker in current system is 2, which is smaller than
what this DataLoader is going to create. Please be aware that excessive worker
creation might get DataLoader running slow or even freeze, lower the worker
number to avoid potential slowness/freeze if necessary.
  warnings.warn(
```

[2]:
```
class MLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(784, 200)
        self.fc2 = nn.Linear(200, 50)
        self.fc3 = nn.Linear(50, 10)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        return self.fc3(x)

def get_model():
    return MLP().to(device)
```

[3]:
```
def train_one_epoch(model, optimizer, scheduler=None):
    model.train()
    for data, target in train_loader:
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.cross_entropy(output, target)
        loss.backward()
        optimizer.step()
    if scheduler:
        scheduler.step()

def evaluate(model):
    model.eval()
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            preds = output.argmax(dim=1)
            correct += (preds == target).sum().item()
    acc = correct / len(test_loader.dataset)
    return acc
```

[4]:
```
num_epochs = 10
results = {}

for opt_name in ['SGD', 'SGD_Momentum', 'AdaGrad', 'RMSprop', 'Adam']:
    print(f'\nOptimizer: {opt_name}')
    model = get_model()

    if opt_name == 'SGD':
```

```
        optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
        scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=5,␣
↪gamma=0.5)
    elif opt_name == 'SGD_Momentum':
        optimizer = torch.optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
        scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=5,␣
↪gamma=0.5)
    elif opt_name == 'AdaGrad':
        optimizer = torch.optim.Adagrad(model.parameters(), lr=0.01)
        scheduler = None
    elif opt_name == 'RMSprop':
        optimizer = torch.optim.RMSprop(model.parameters(), lr=0.001, alpha=0.9)
        scheduler = None
    elif opt_name == 'Adam':
        optimizer = torch.optim.Adam(model.parameters(), lr=0.001, betas=(0.9,␣
↪0.999))
        scheduler = None

    for epoch in range(1, num_epochs + 1):
        train_one_epoch(model, optimizer, scheduler)
        if epoch % 5 == 0:
            acc = evaluate(model)
            print(f'epoch {epoch:2d} Test Accuracy: {acc:.4f}')

    final_acc = evaluate(model)
    results[opt_name] = final_acc
    print(f'Final test Acuracy ({opt_name}): {final_acc:.4f}')

print('\nsummary of test accuracies:')
for name, acc in results.items():
    print(f'  {name:15s}: {acc:.4f}')
```

```
Optimizer: SGD
epoch  5 Test Accuracy: 0.9333
epoch 10 Test Accuracy: 0.9422
Final test Acuracy (SGD): 0.9422

Optimizer: SGD_Momentum
epoch  5 Test Accuracy: 0.9755
epoch 10 Test Accuracy: 0.9787
Final test Acuracy (SGD_Momentum): 0.9787

Optimizer: AdaGrad
epoch  5 Test Accuracy: 0.9740
epoch 10 Test Accuracy: 0.9778
Final test Acuracy (AdaGrad): 0.9778
```

```
Optimizer: RMSprop
epoch  5 Test Accuracy: 0.9744
epoch 10 Test Accuracy: 0.9788
Final test Acuracy (RMSprop): 0.9788

Optimizer: Adam
epoch  5 Test Accuracy: 0.9762
epoch 10 Test Accuracy: 0.9786
Final test Acuracy (Adam): 0.9786

summary of test accuracies:
  SGD             : 0.9422
  SGD_Momentum    : 0.9787
  AdaGrad         : 0.9778
  RMSprop         : 0.9788
  Adam            : 0.9786
```