Report

110511118 陳孟頡

Part I.

- **Target:**

  Use the training techniques learned today to further improve your accuracy.

- **Best accuracy**: 90.12%

```
Test Loss: 0.295320

Test Accuracy of Class     0: 85.50% (855/1000)
Test Accuracy of Class     1: 97.60% (976/1000)
Test Accuracy of Class     2: 83.90% (839/1000)
Test Accuracy of Class     3: 92.70% (927/1000)
Test Accuracy of Class     4: 84.00% (840/1000)
Test Accuracy of Class     5: 96.50% (965/1000)
Test Accuracy of Class     6: 72.10% (721/1000)
Test Accuracy of Class     7: 96.40% (964/1000)
Test Accuracy of Class     8: 97.50% (975/1000)
Test Accuracy of Class     9: 95.00% (950/1000)

Test Accuracy (Overall): 90.12% (9012/10000)
```

- **Training techniques:**

1. Feature normalization

為了使各個特徵在進行 gradient descent 時收斂速度相當，加速訓練過程，因此在進行訓練前，先對資料進行 normalization，程式如下:

```
train_tfms = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])
valid_tfms = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])
```

在架構中進入每層的 activation function 前，也會進行 batch normalization，讓 error surface 平坦化，在有限的訓練 epoch 數下達到較高的準確度。架構中 batch normalization 程式如下:

```
    class Net(nn.Module):
        def __init__(self):
            super(Net, self).__init__()
            self.fc1 = nn.Linear(28 * 28, 512)
            self.bn1 = nn.BatchNorm1d(512)
            self.fc2 = nn.Linear(512, 256)
            self.bn2 = nn.BatchNorm1d(256)
            self.fc3 = nn.Linear(256, 10)

        def forward(self, x):
            x = x.view(-1, 28 * 28)
            x = self.fc1(x)
            x = F.relu(self.bn1(x))
            x = self.fc2(x)
            x = F.relu(self.bn2(x))
            x = self.fc3(x)
            return x
```

2. Optimizer

將原先的 SGD 改為 Adam optimizer，選用 Adam 原因是他同時具備 momentum、RMSprop 以及 correction 的功能，能將訓練效果最佳化。作業中程式如下:

```
    optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

3. Learning rate scheduler

使用 learning rate scheduler 可在訓練一開始有較高的更新速度，同時在 loss 減少時降低 learning rate，讓模型的 loss 能持續減少而不震盪。使用程式如下:

```
    scheduler =
torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max =
10, eta_min = 0.0001)
```

- Discussion:
  1. 為何不使用 data augmentation?

     不使用 data augmentation 的原因有二，一是經過測試，使用 data augmentation 並不會使準確度有明顯提升，二是在使用後，會使訓練時間變長。因此經考量後，決定不使用。

  2. 為何不使用 dropout?

     Dropout 通常用於神經網路架構深層參數較多的模型中，用以防止發生 overfitting 現象，此次的神經網路架構，為前面層數 neurons 數較多、後面層數較少的類型，因此推測使用 dropout 效益並不大。

     經過實測後也發現，不使用 dropout 訓練出的模型普遍具有較佳的準確性，因此選擇不使用 dropout。