機器學習-作業三

609410162 彭郁翔

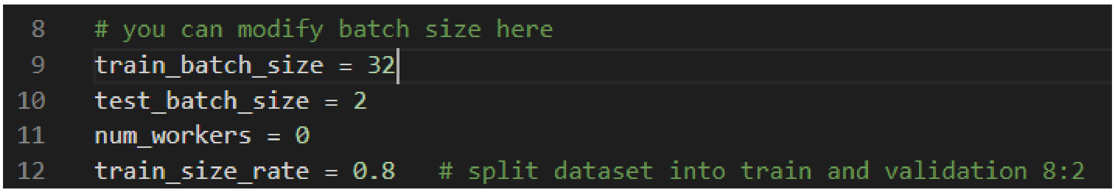
作業要求-1:

- 調整 epoch、batch size、learning rate

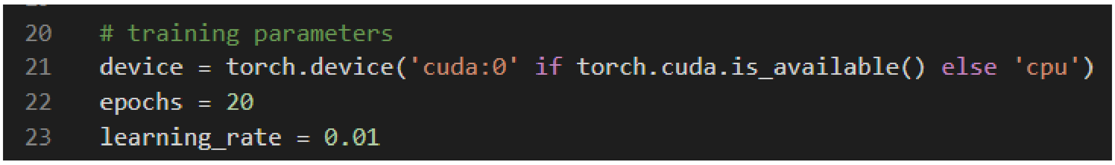
Epoch:20

Batch size:32

Learning rate:0.01

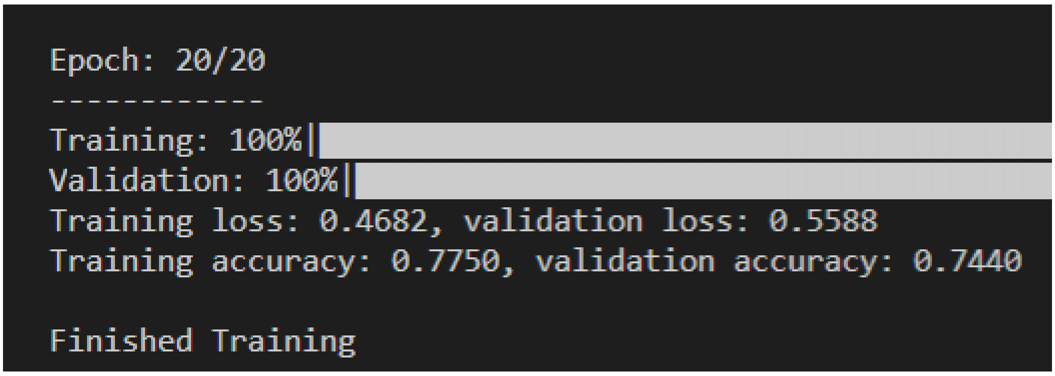


```
# you can modify batch size here
train_batch_size = 32
test_batch_size = 2
num_workers = 0
train_size_rate = 0.8   # split dataset into train and validation 8:2
```



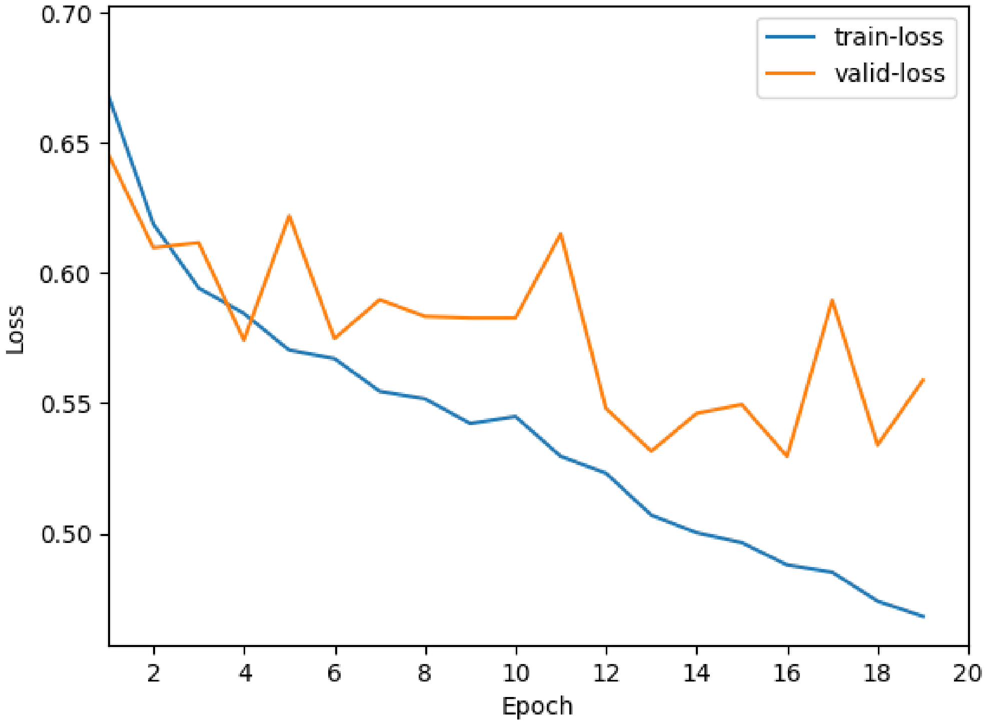
```
# training parameters
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
epochs = 20
learning_rate = 0.01
```

訓練後結果:

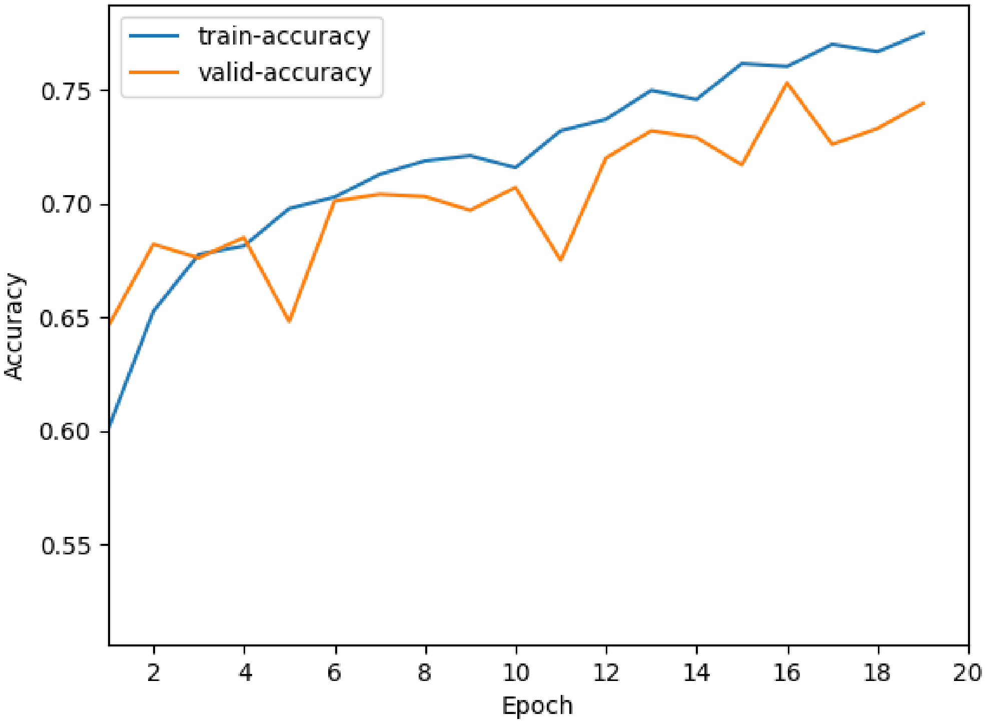


```
Epoch: 20/20
------------
Training: 100%|
Validation: 100%|
Training loss: 0.4682, validation loss: 0.5588
Training accuracy: 0.7750, validation accuracy: 0.7440

Finished Training
```

觀察並說明參數調整對損失函數及準確率的變化:

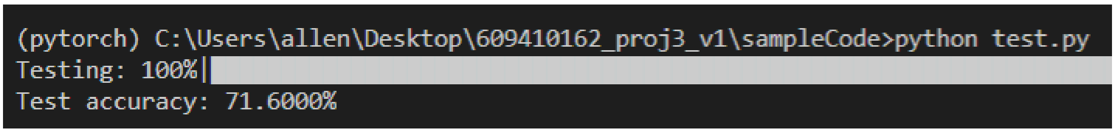


0.70
0.65
0.60
0.55
0.50
Loss
2
4
6
8
10
12
14
16
18
20
Epoch
train-loss
valid-loss



0.75
0.70
0.65
0.60
0.55
Accuracy
2
4
6
8
10
12
14
16
18
20
Epoch
train-accuracy
valid-accuracy

觀察在測試集上的結果:

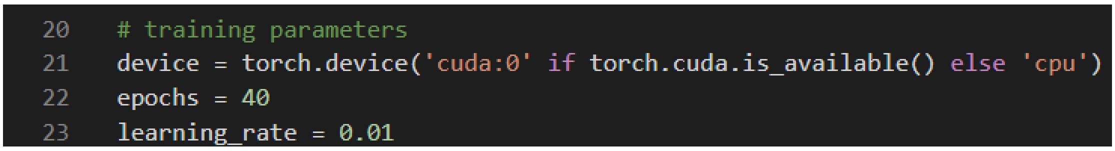


```
(pytorch) C:\Users\allen\Desktop\609410162_proj3_v1\sampleCode>python test.py
Testing: 100%|
Test accuracy: 71.6000%
```

Epoch:40

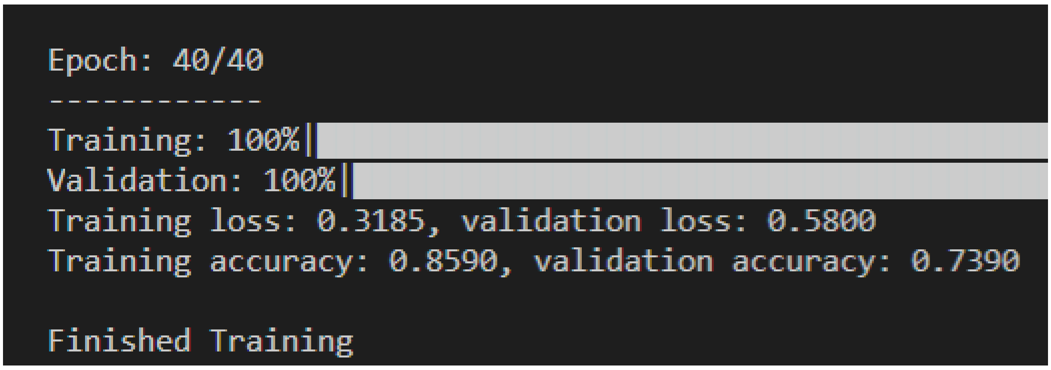
Batch size:32

Learning rate:0.01



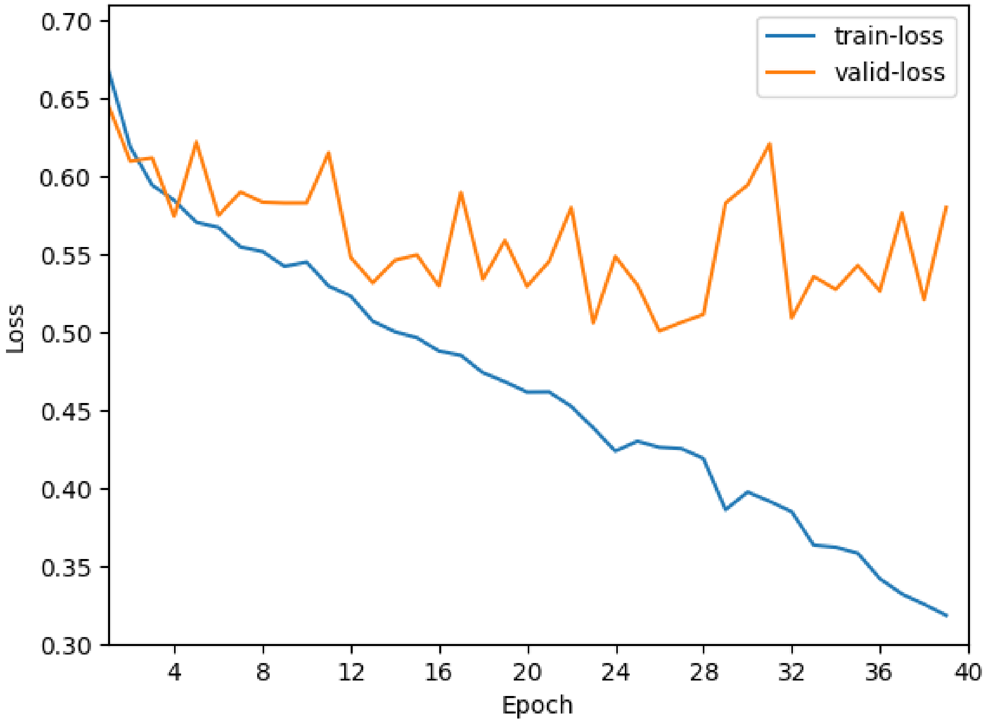
```python
# training parameters
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
epochs = 40
learning_rate = 0.01
```

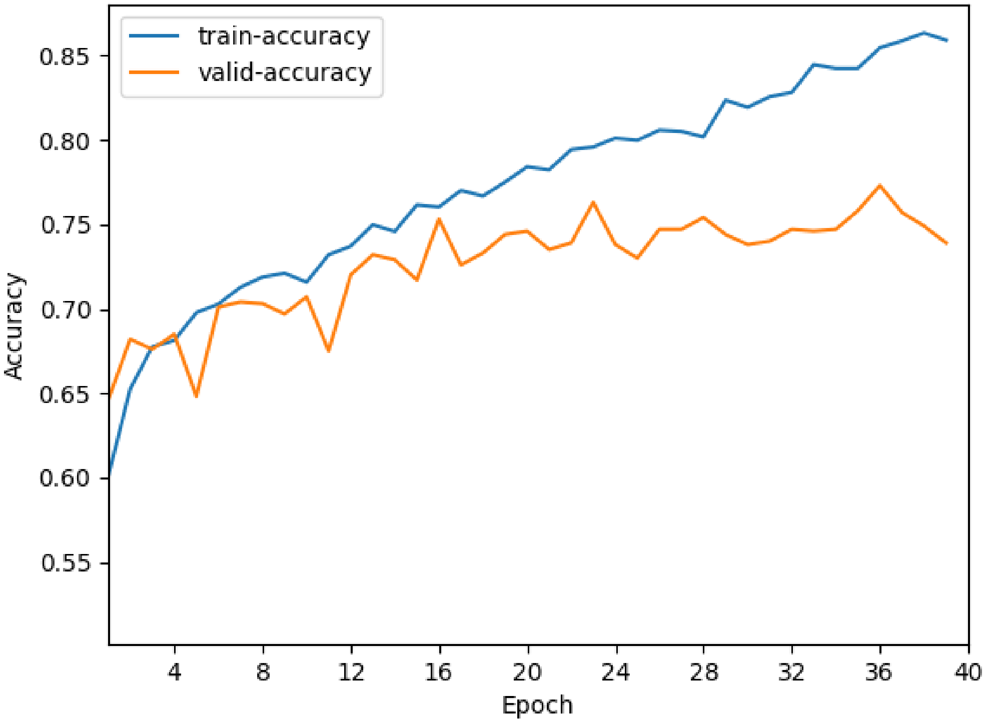
訓練後的結果:



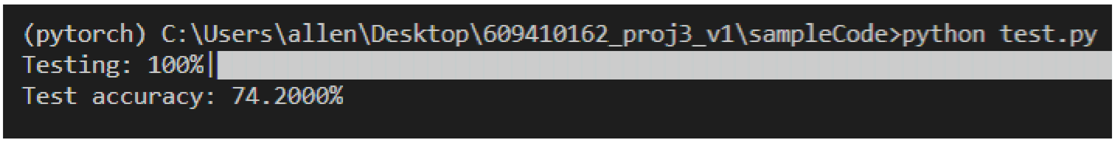
```
Epoch: 40/40
------------
Training: 100%|
Validation: 100%|
Training loss: 0.3185, validation loss: 0.5800
Training accuracy: 0.8590, validation accuracy: 0.7390

Finished Training
```

觀察並說明參數調整對損失函數及準確率的變化:


0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
Loss
4
8
12
16
20
24
28
32
36
40
Epoch
train-loss
valid-loss


0.85
0.80
0.75
0.70
0.65
0.60
0.55
Accuracy
4
8
12
16
20
24
28
32
36
40
Epoch
train-accuracy
valid-accuracy

觀察在測試集上的結果:

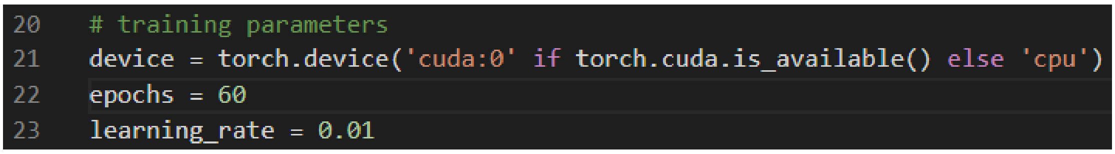


```
(pytorch) C:\Users\allen\Desktop\609410162_proj3_v1\sampleCode>python test.py
Testing: 100%|
Test accuracy: 74.2000%
```

Epoch:60

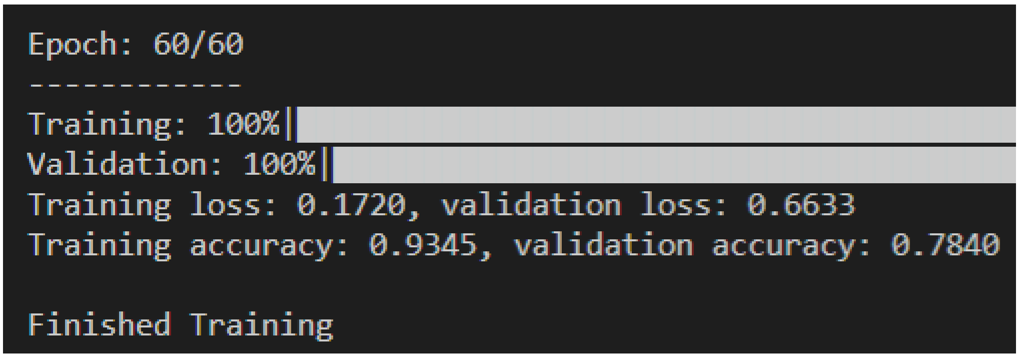
Batch size:32

Learning rate:0.01



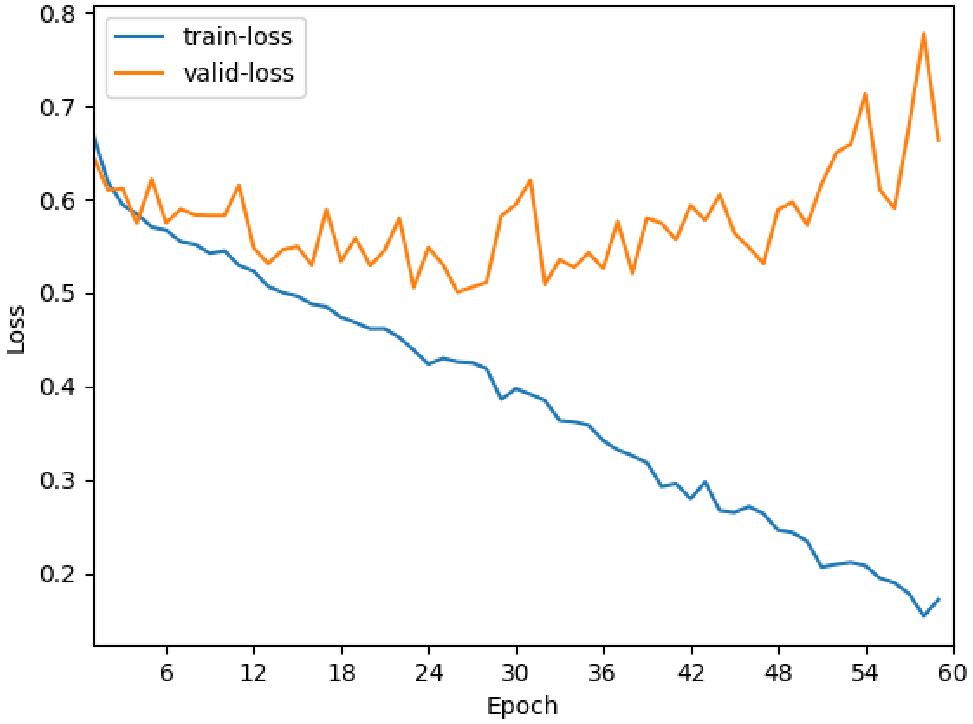
```
# training parameters
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
epochs = 60
learning_rate = 0.01
```

訓練後的結果:

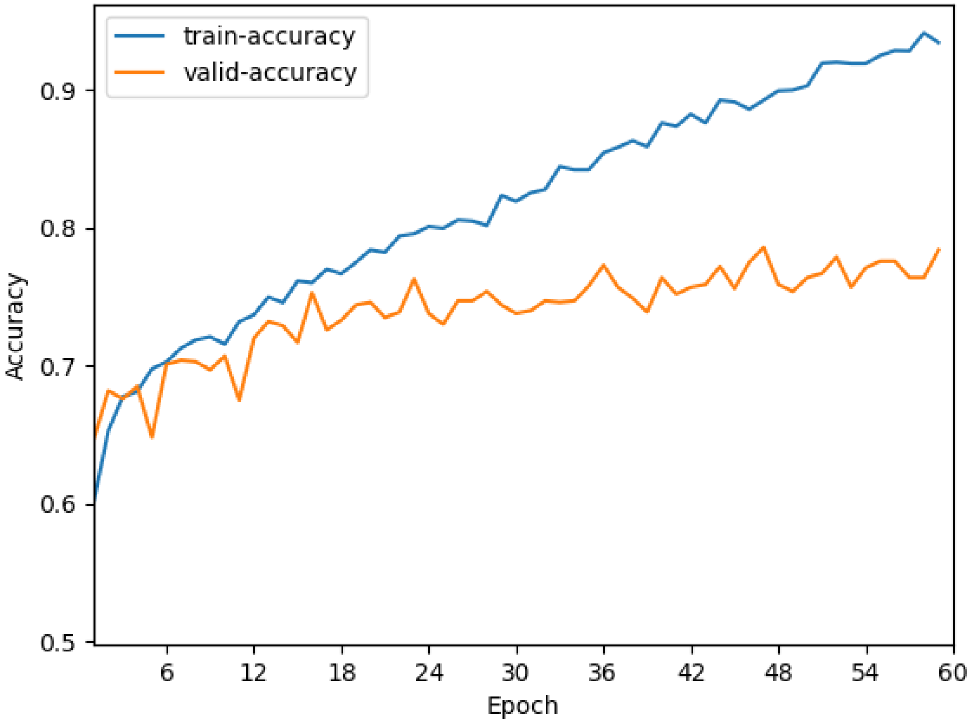


```
Epoch: 60/60
------------
Training: 100%|
Validation: 100%|
Training loss: 0.1720, validation loss: 0.6633
Training accuracy: 0.9345, validation accuracy: 0.7840

Finished Training
```

觀察並說明參數調整對損失函數及準確率的變化:

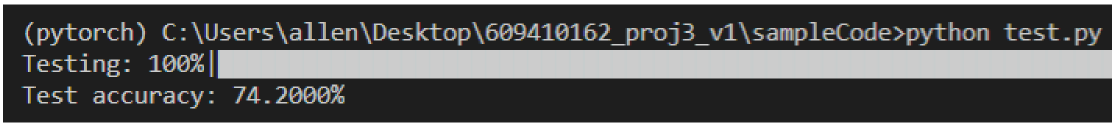


train-loss
valid-loss
0.8
0.7
0.6
0.5
0.4
0.3
0.2
Loss
6
12
18
24
30
36
42
48
54
60
Epoch



train-accuracy
valid-accuracy
0.9
0.8
0.7
0.6
0.5
Accuracy
6
12
18
24
30
36
42
48
54
60
Epoch

觀察在測試集上的結果:

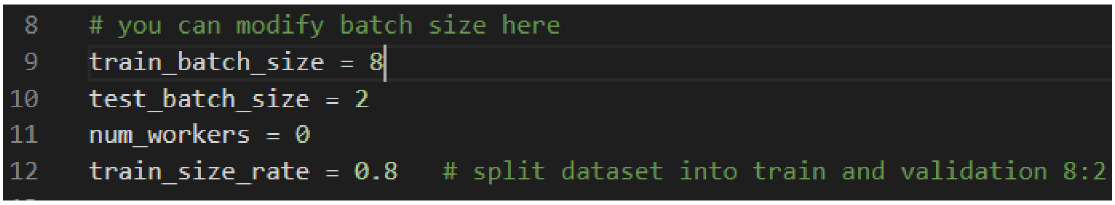


```
(pytorch) C:\Users\allen\Desktop\609410162_proj3_v1\sampleCode>python test.py
Testing: 100%|
Test accuracy: 74.2000%
```

Epoch:20

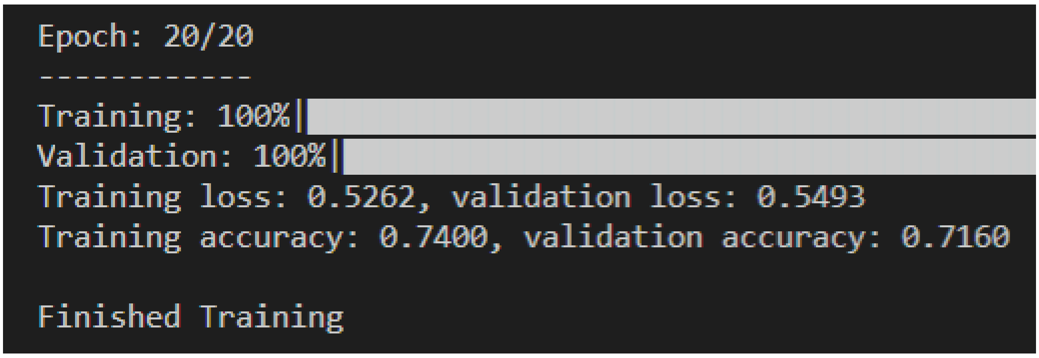
Batch size:8

Learning rate:0.01



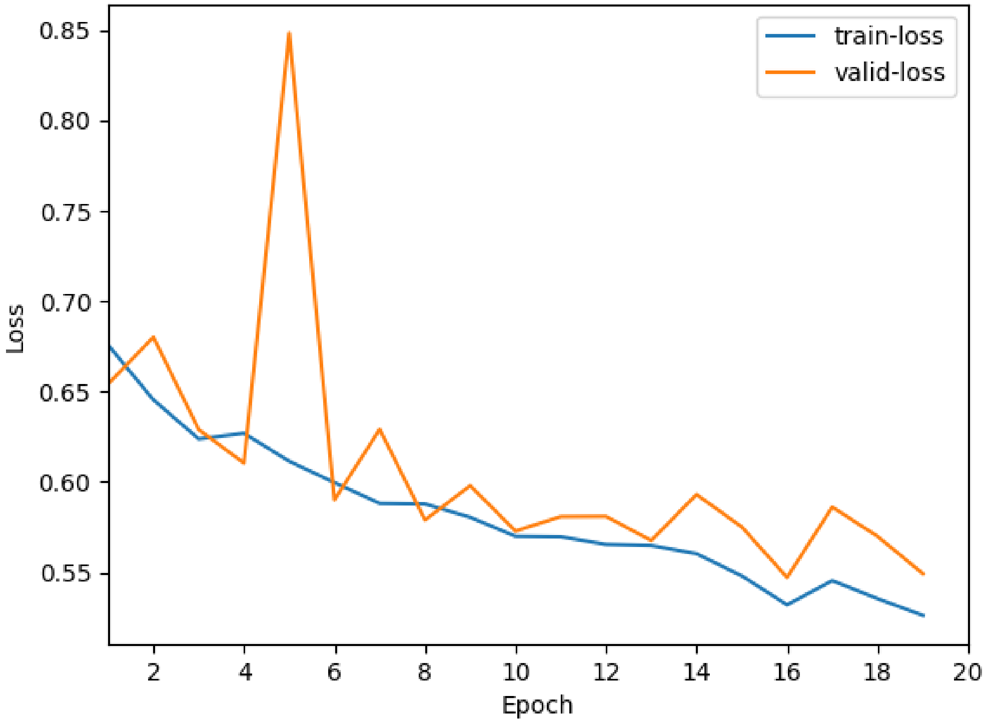
```
# you can modify batch size here
train_batch_size = 8
test_batch_size = 2
num_workers = 0
train_size_rate = 0.8   # split dataset into train and validation 8:2
```

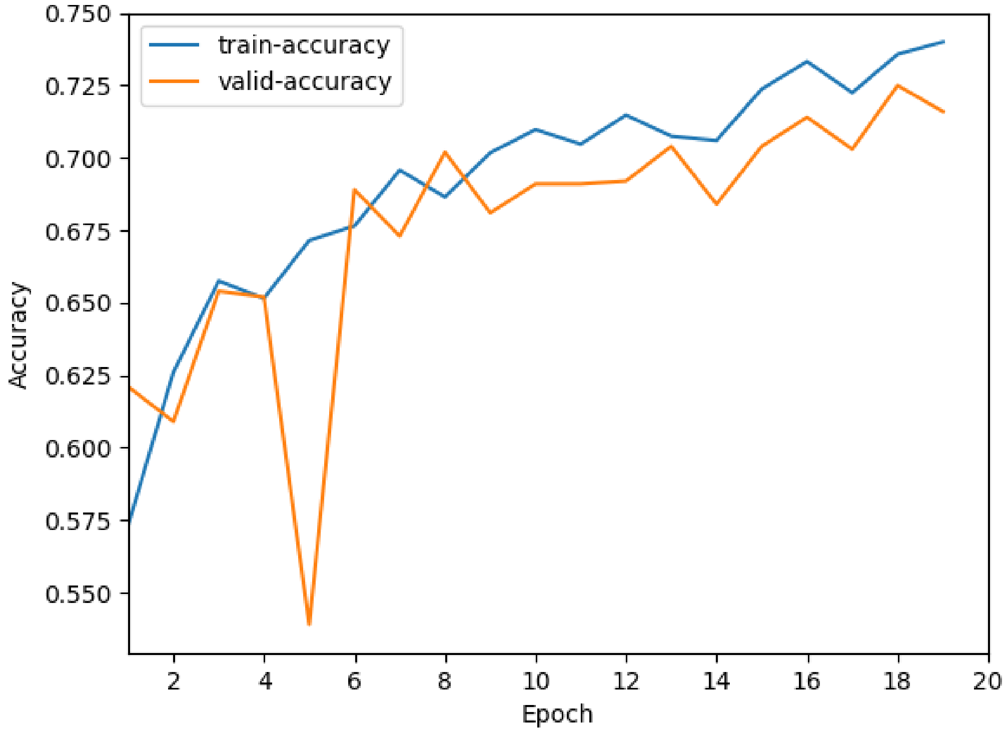
訓練後的結果:



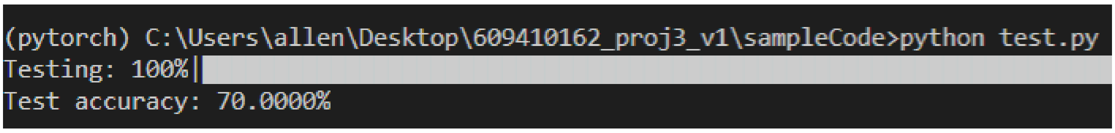
```
Epoch: 20/20
------------
Training: 100%|
Validation: 100%|
Training loss: 0.5262, validation loss: 0.5493
Training accuracy: 0.7400, validation accuracy: 0.7160

Finished Training
```

觀察並說明參數調整對損失函數及準確率的變化:


train-loss
valid-loss
Loss
Epoch
0.85
0.80
0.75
0.70
0.65
0.60
0.55
2
4
6
8
10
12
14
16
18
20


train-accuracy
valid-accuracy
Accuracy
Epoch
0.750
0.725
0.700
0.675
0.650
0.625
0.600
0.575
0.550
2
4
6
8
10
12
14
16
18
20

觀察在測試集上的結果:

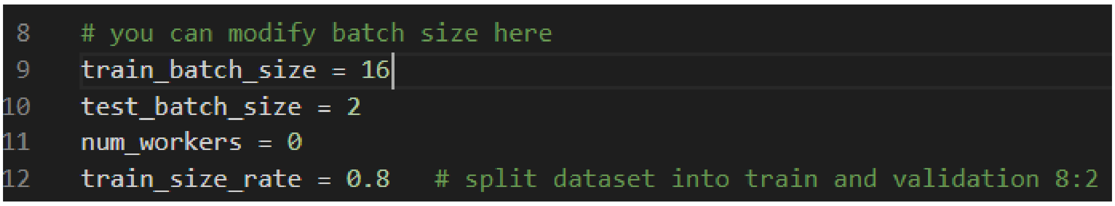


```
(pytorch) C:\Users\allen\Desktop\609410162_proj3_v1\sampleCode>python test.py
Testing: 100%|
Test accuracy: 70.0000%
```

Epoch:20

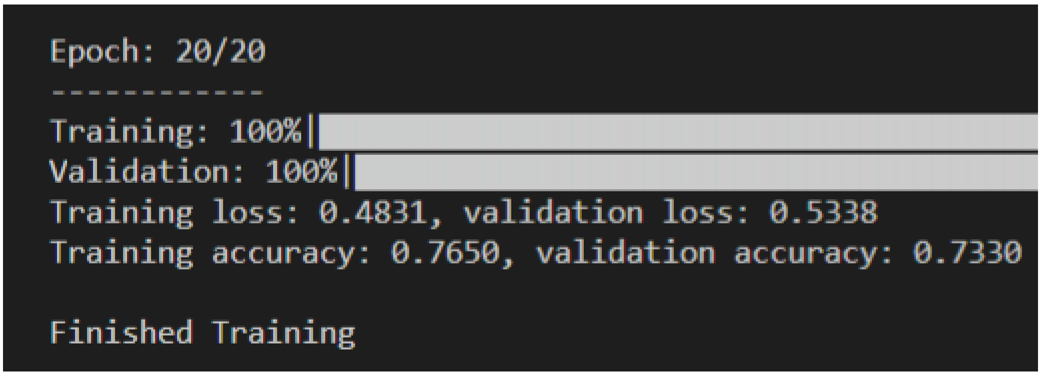
Batch size:16

Learning rate:0.01



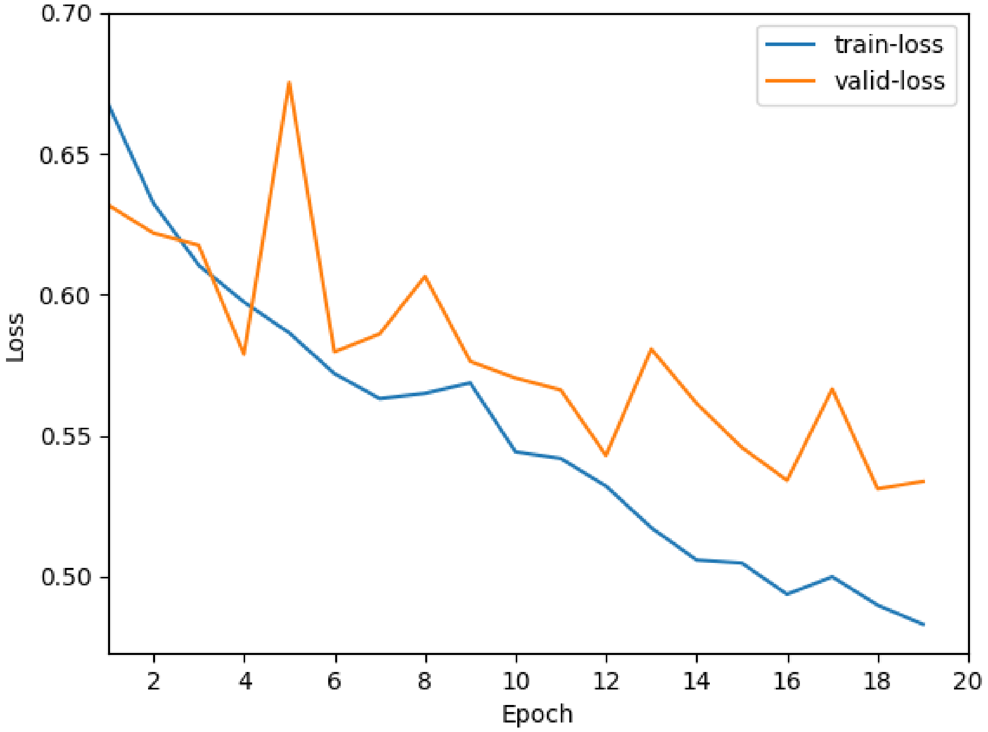
```
# you can modify batch size here
train_batch_size = 16
test_batch_size = 2
num_workers = 0
train_size_rate = 0.8   # split dataset into train and validation 8:2
```

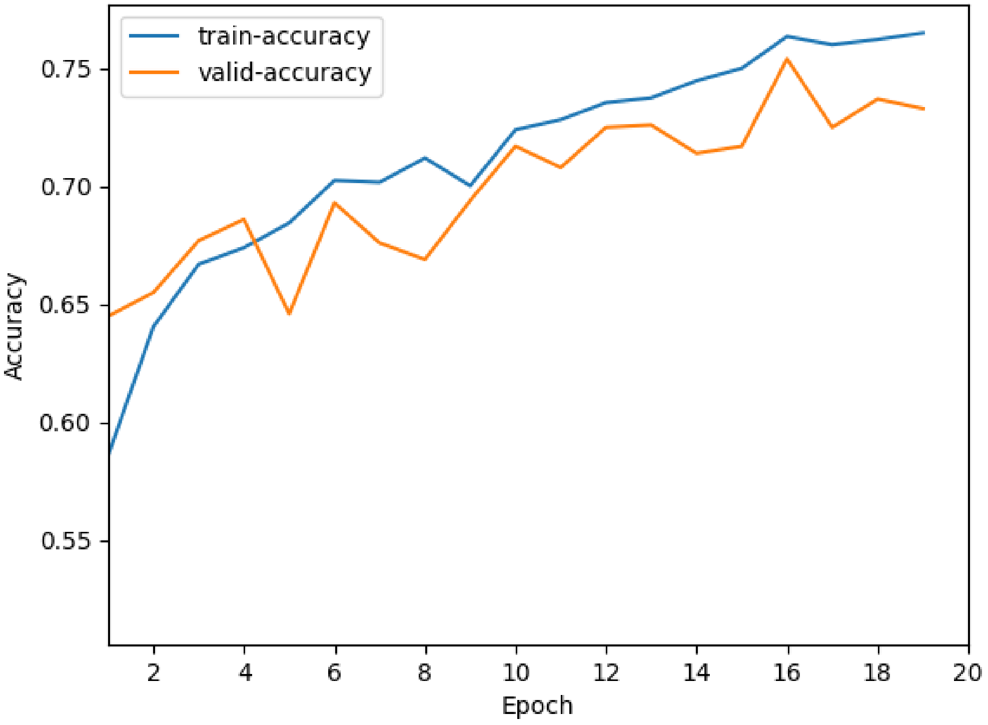
訓練後的結果:



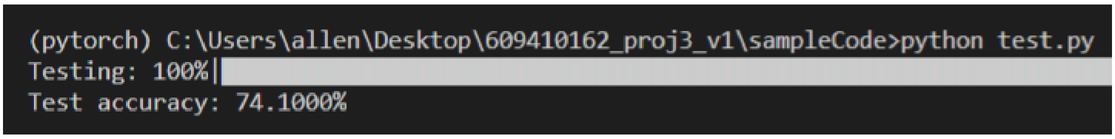
```
Epoch: 20/20
------------
Training: 100%|
Validation: 100%|
Training loss: 0.4831, validation loss: 0.5338
Training accuracy: 0.7650, validation accuracy: 0.7330

Finished Training
```

觀察並說明參數調整對損失函數及準確率的變化:


0.70
0.65
0.60
0.55
0.50
Loss
2
4
6
8
10
12
14
16
18
20
Epoch
train-loss
valid-loss


0.75
0.70
0.65
0.60
0.55
Accuracy
2
4
6
8
10
12
14
16
18
20
Epoch
train-accuracy
valid-accuracy

觀察在測試集上的結果:

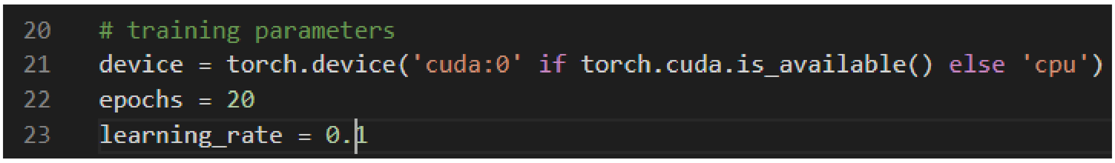


```
(pytorch) C:\Users\allen\Desktop\609410162_proj3_v1\sampleCode>python test.py
Testing: 100%|
Test accuracy: 74.1000%
```

Epoch:20

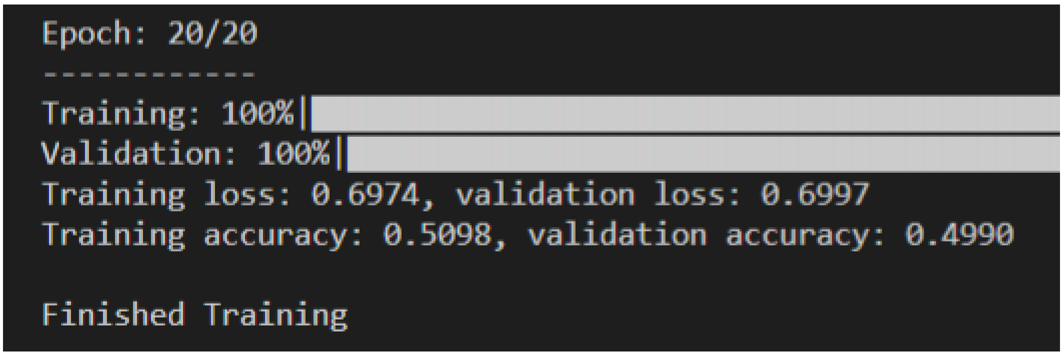
Batch size:32

Learning rate:0.1



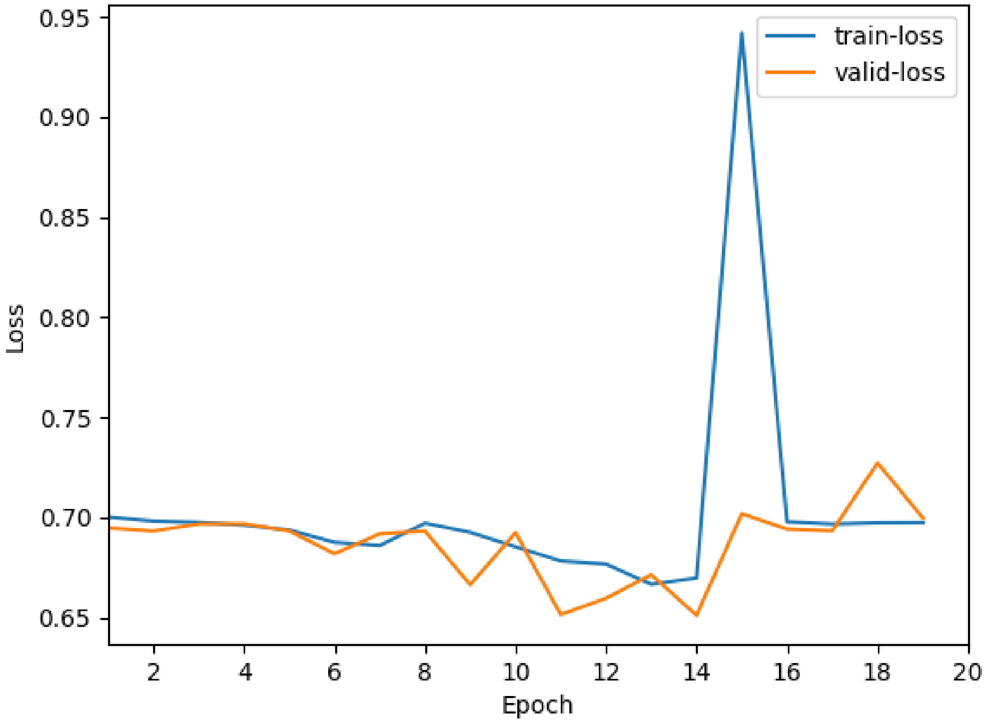
```python
# training parameters
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
epochs = 20
learning_rate = 0.1
```

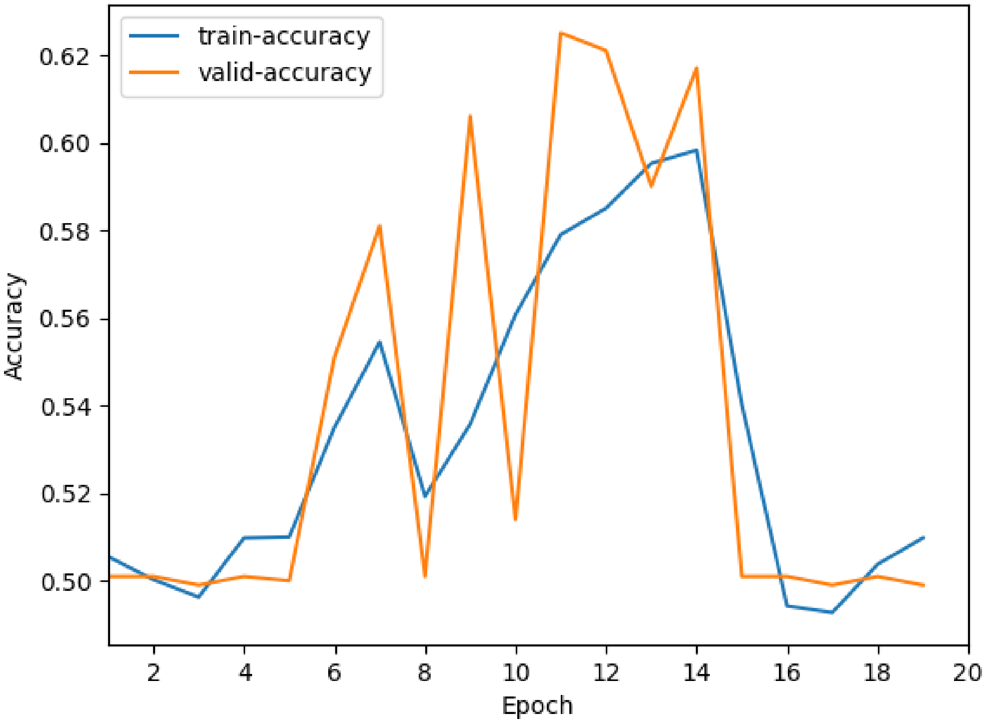
訓練後的結果:



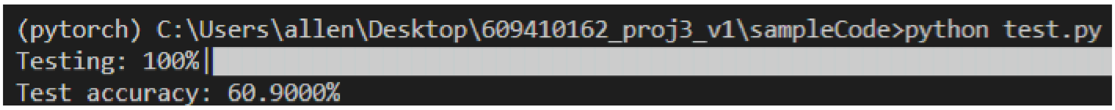
```
Epoch: 20/20
------------
Training: 100%|
Validation: 100%|
Training loss: 0.6974, validation loss: 0.6997
Training accuracy: 0.5098, validation accuracy: 0.4990

Finished Training
```

觀察並說明參數調整對損失函數及準確率的變化:


0.95
0.90
0.85
0.80
0.75
0.70
0.65
Loss
2
4
6
8
10
12
14
16
18
20
Epoch
train-loss
valid-loss


0.62
0.60
0.58
0.56
0.54
0.52
0.50
Accuracy
2
4
6
8
10
12
14
16
18
20
Epoch
train-accuracy
valid-accuracy

觀察在測試集上的結果:

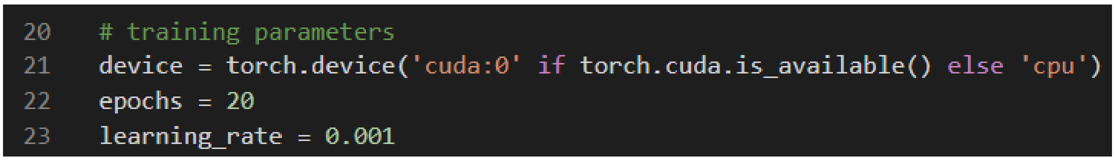


```
(pytorch) C:\Users\allen\Desktop\609410162_proj3_v1\sampleCode>python test.py
Testing: 100%|
Test accuracy: 60.9000%
```

Epoch:20

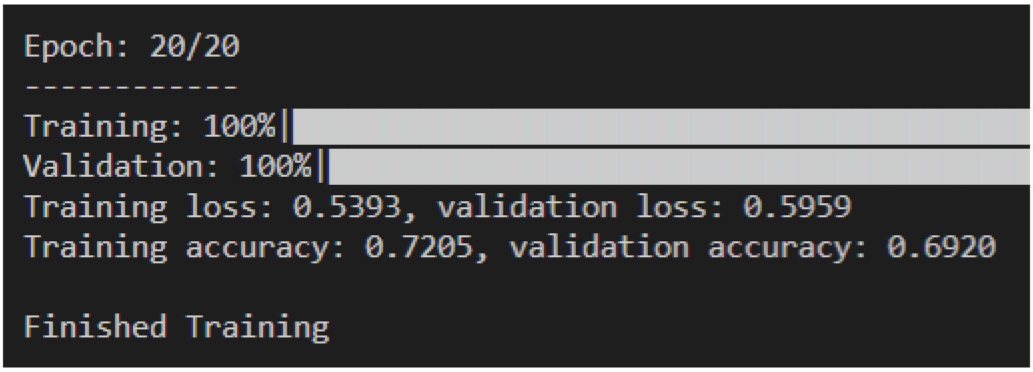
Batch size:32

Learning rate:0.001



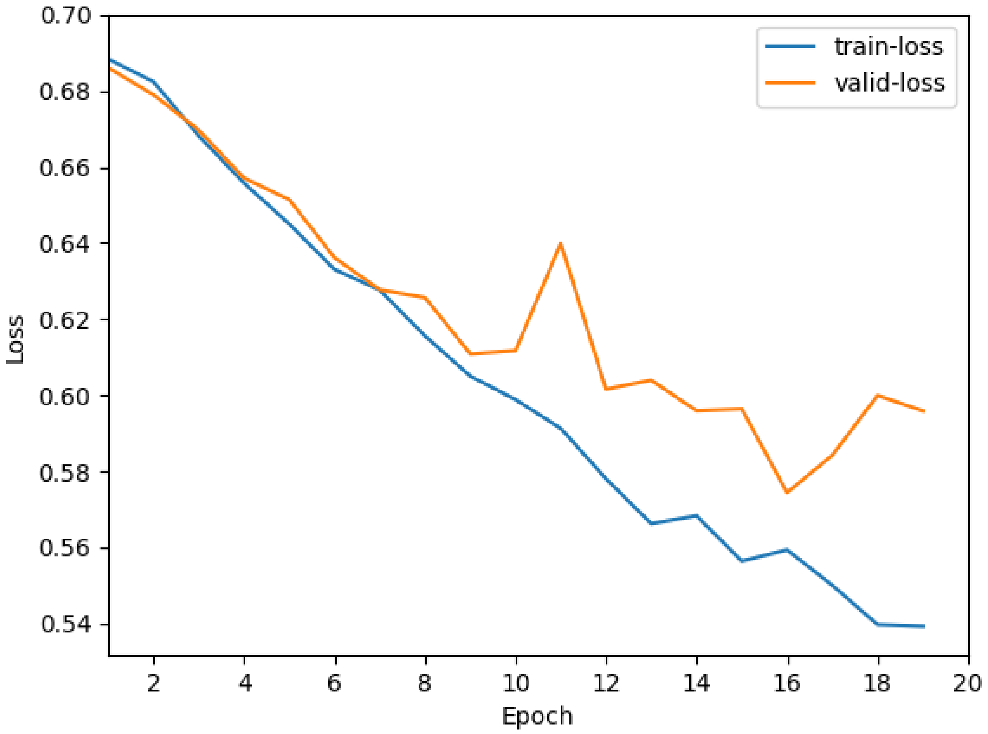
```
# training parameters
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
epochs = 20
learning_rate = 0.001
```

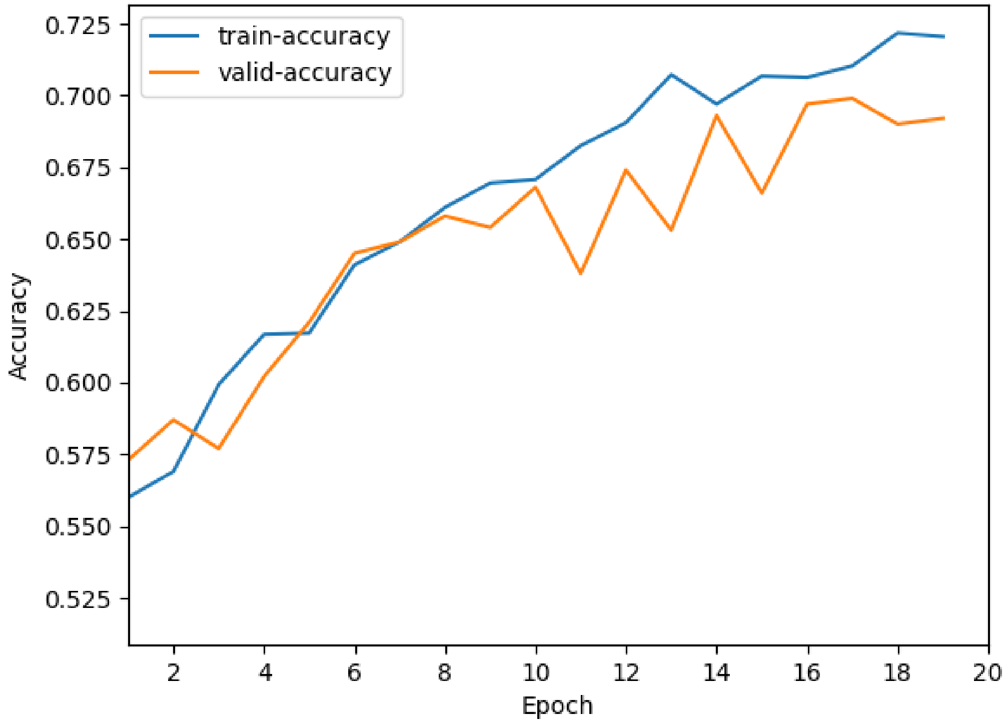
訓練後的結果:



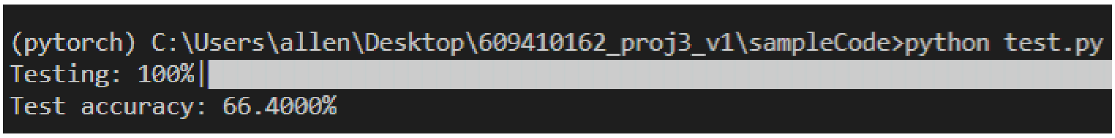
```
Epoch: 20/20
------------
Training: 100%|
Validation: 100%|
Training loss: 0.5393, validation loss: 0.5959
Training accuracy: 0.7205, validation accuracy: 0.6920

Finished Training
```

觀察並說明參數調整對損失函數及準確率的變化:


0.70
0.68
0.66
0.64
0.62
0.60
0.58
0.56
0.54
Loss
2
4
6
8
10
12
14
16
18
20
Epoch
train-loss
valid-loss


0.725
0.700
0.675
0.650
0.625
0.600
0.575
0.550
0.525
Accuracy
2
4
6
8
10
12
14
16
18
20
Epoch
train-accuracy
valid-accuracy

觀察在測試集上的結果:



```
(pytorch) C:\Users\allen\Desktop\609410162_proj3_v1\sampleCode>python test.py
Testing: 100%|
Test accuracy: 66.4000%
```

作業要求-2

- 修改提供的 CNN model(修改 MyCNN model)

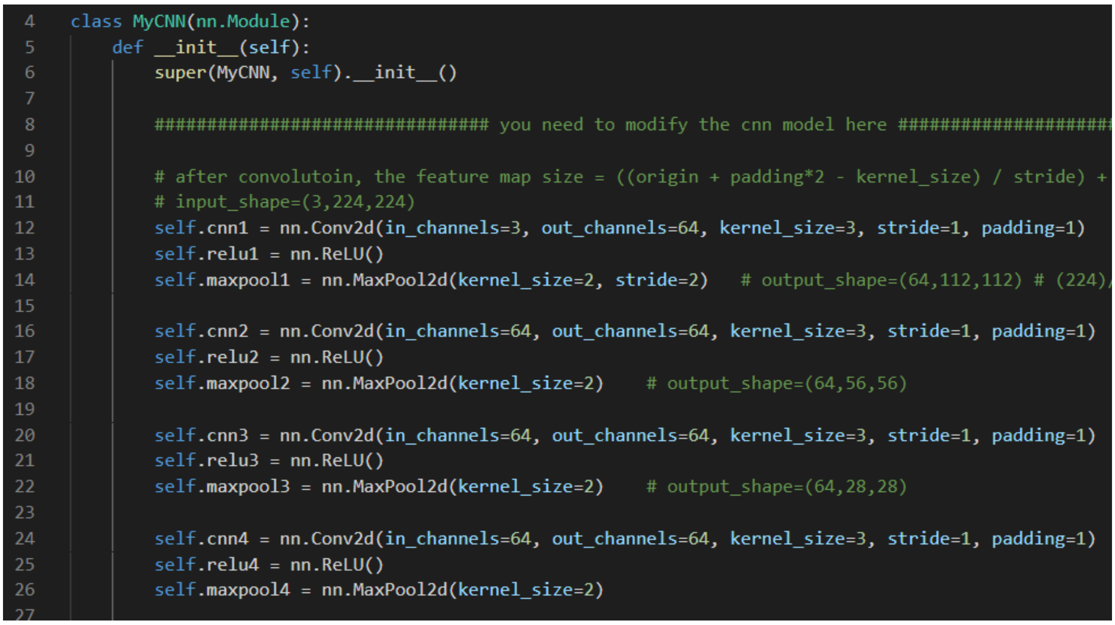
參數調整:

Epoch:60

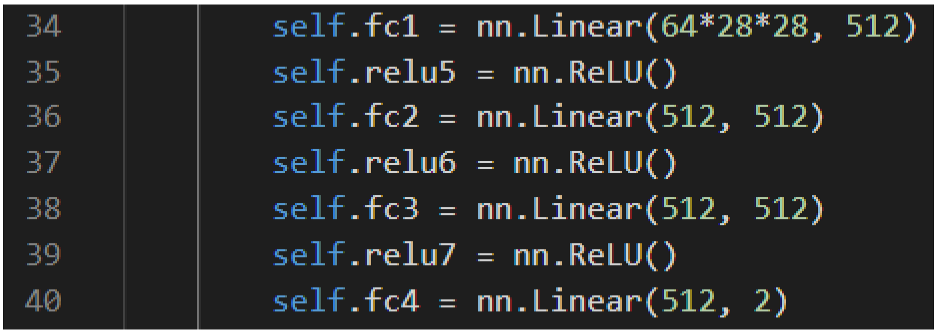
Batch size:32

Learning rate:0.01

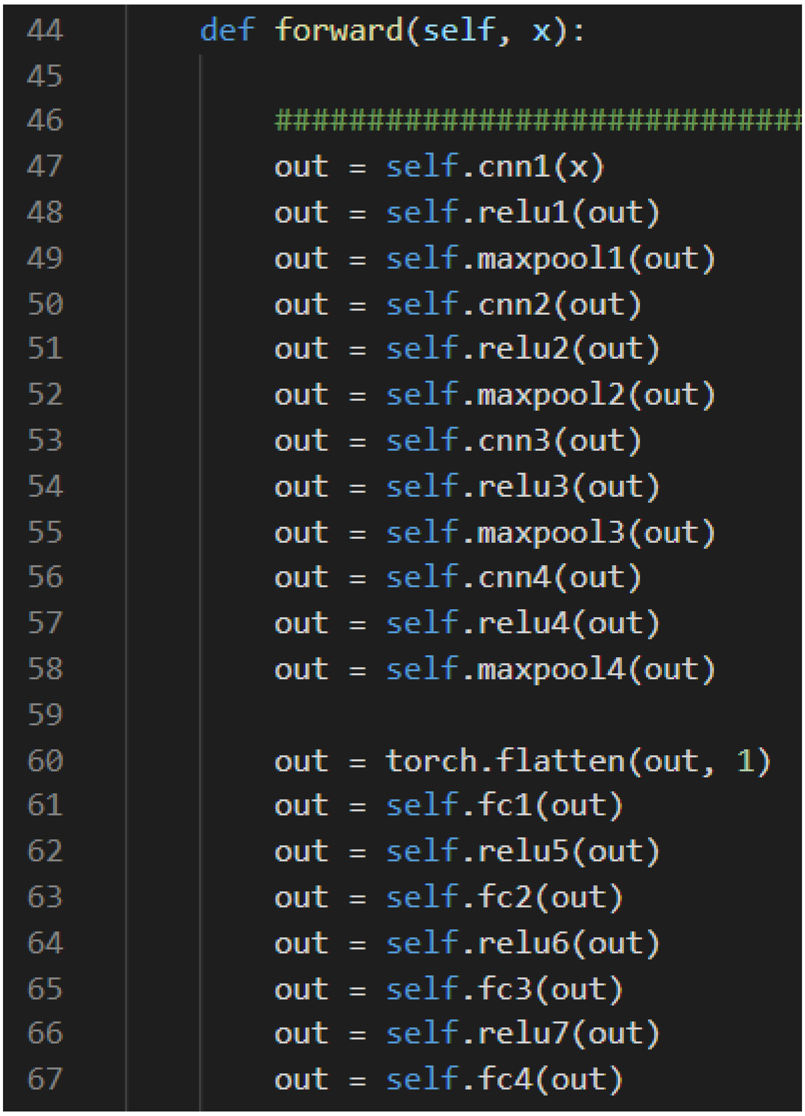
增加一層 convolution layer。



```
class MyCNN(nn.Module):
    def __init__(self):
        super(MyCNN, self).__init__()

        ################################ you need to modify the cnn model here #####################

        # after convolutoin, the feature map size = ((origin + padding*2 - kernel_size) / stride) +
        # input_shape=(3,224,224)
        self.cnn1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3, stride=1, padding=1)
        self.relu1 = nn.ReLU()
        self.maxpool1 = nn.MaxPool2d(kernel_size=2, stride=2)   # output_shape=(64,112,112) # (224)/

        self.cnn2 = nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, stride=1, padding=1)
        self.relu2 = nn.ReLU()
        self.maxpool2 = nn.MaxPool2d(kernel_size=2)    # output_shape=(64,56,56)

        self.cnn3 = nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, stride=1, padding=1)
        self.relu3 = nn.ReLU()
        self.maxpool3 = nn.MaxPool2d(kernel_size=2)    # output_shape=(64,28,28)

        self.cnn4 = nn.Conv2d(in_channels=64, out_channels=64, kernel_size=3, stride=1, padding=1)
        self.relu4 = nn.ReLU()
        self.maxpool4 = nn.MaxPool2d(kernel_size=2)
```

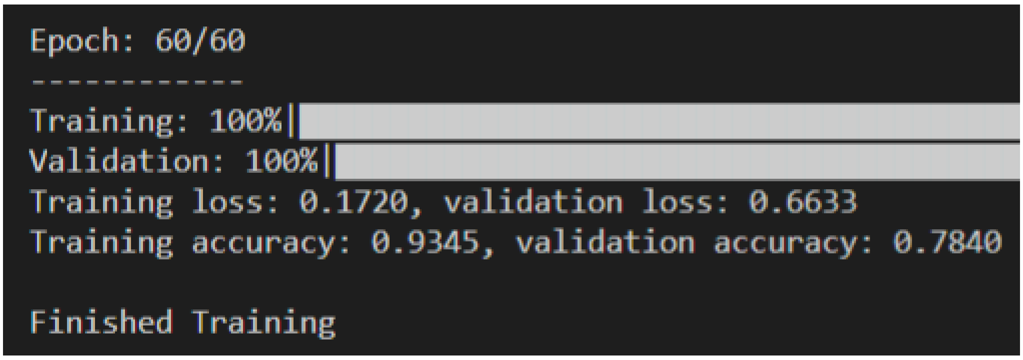


```
        self.fc1 = nn.Linear(64*28*28, 512)
        self.relu5 = nn.ReLU()
        self.fc2 = nn.Linear(512, 512)
        self.relu6 = nn.ReLU()
        self.fc3 = nn.Linear(512, 512)
        self.relu7 = nn.ReLU()
        self.fc4 = nn.Linear(512, 2)
```



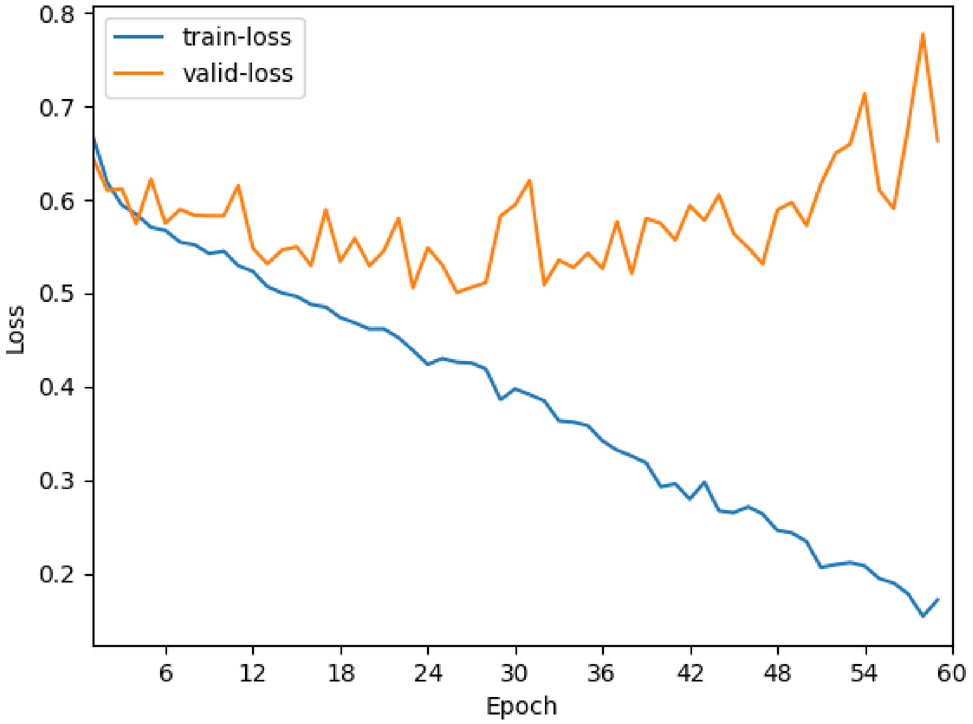
```
    def forward(self, x):

        ############################
        out = self.cnn1(x)
        out = self.relu1(out)
        out = self.maxpool1(out)
        out = self.cnn2(out)
        out = self.relu2(out)
        out = self.maxpool2(out)
        out = self.cnn3(out)
        out = self.relu3(out)
        out = self.maxpool3(out)
        out = self.cnn4(out)
        out = self.relu4(out)
        out = self.maxpool4(out)

        out = torch.flatten(out, 1)
        out = self.fc1(out)
        out = self.relu5(out)
        out = self.fc2(out)
        out = self.relu6(out)
        out = self.fc3(out)
        out = self.relu7(out)
        out = self.fc4(out)
```

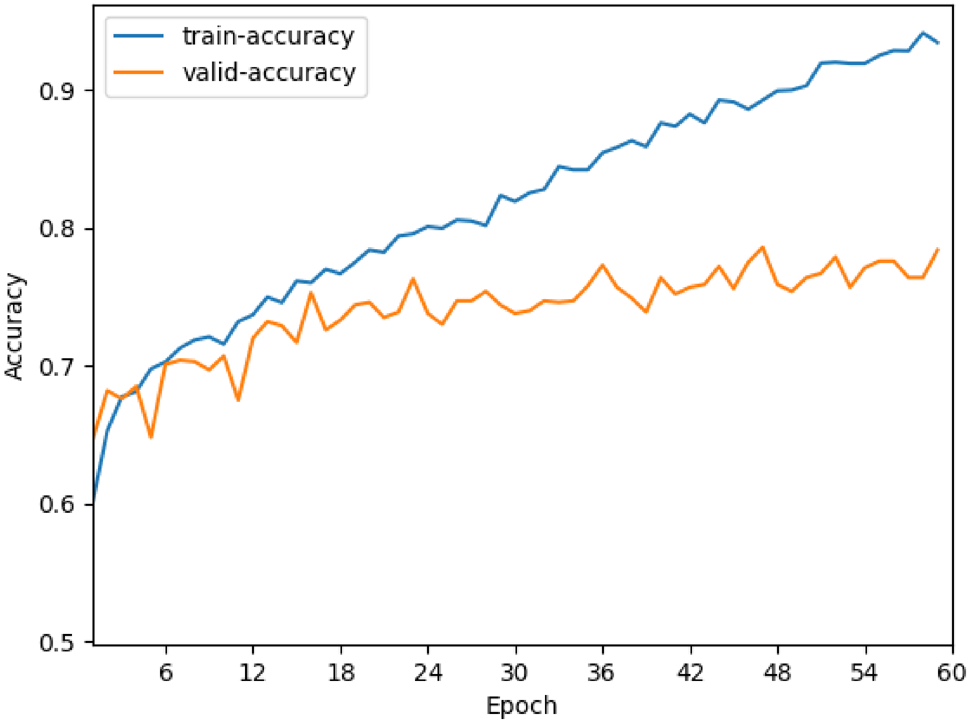
訓練後的結果:



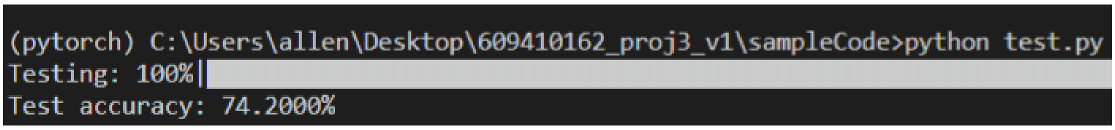
```
Epoch: 60/60
------------
Training: 100%|
Validation: 100%|
Training loss: 0.1720, validation loss: 0.6633
Training accuracy: 0.9345, validation accuracy: 0.7840

Finished Training
```

觀察並說明參數調整對損失函數及準確率的變化:


train-loss
valid-loss
Loss
0.8
0.7
0.6
0.5
0.4
0.3
0.2
6
12
18
24
30
36
42
48
54
60
Epoch


train-accuracy
valid-accuracy
Accuracy
0.9
0.8
0.7
0.6
0.5
6
12
18
24
30
36
42
48
54
60
Epoch

觀察在測試集上的結果:



```
(pytorch) C:\Users\allen\Desktop\609410162_proj3_v1\sampleCode>python test.py
Testing: 100%|
Test accuracy: 74.2000%
```