機器學習作業1

-SCENE RECOGNITION

405410036 陳學韋

前提資訊

- 決定開始訓練的權重 - torch.manual_seed
- 進行一次iteration所訓練資料的數量 – batch size
- 將所有訓練資料訓練次數 – epoch
- 各種對於梯度下降演算法的優化方法 – optimizer
- 根據梯度下降的方向決定步長 – learning rate

預設參數

- Seed=123, epochs=10, batch_size=32, SGD, lr=0.01

```
16  torch.manual_seed(123)
17  torch.backends.cudnn.deterministic = True
18  torch.backends.cudnn.benchmark = False
19
```

```
40      #print(DATASET_ROOT)
41      train_set = IMAGE_Dataset(Path(DATASET_ROOT), data_transform)
42      data_loader = DataLoader(dataset=train_set, batch_size=32, shuffle=True, num_workers=1)
43      #print(train_set.num_classes)
44      model = VGG16(num_classes=train_set.num_classes)
45      model = model.cuda(CUDA_DEVICES)
46      model.train()
```

```
48      best_model_params = copy.deepcopy(model.state_dict())
49      best_acc = 0.0
50      count = 100.0
51      num_epochs = 10
52      criterion = nn.CrossEntropyLoss()
53      optimizer = torch.optim.SGD(params=model.parameters(), lr=0.01, momentum=0.9)
54
```

程式碼新增部分

- 每10個將model存成一個檔案，
  並在用python test.py執行

```
86          if (epoch+1)%10==0:
87              #torch.save(model, f'model.pth')
88              #tmp1,tmp2=test.test(f'model.pth')
89              #print(str(tmp1)+'\t'+str(tmp2))
90              torch.save(model, f'model-{count:.02f}-train.pth')
91              #result_test_loss.append(str(tmp1))
92              #result_test_acc.append(str(tmp2))
93              count=count+1
94
```

- 將訓練過程中得到的loss和accuracy分別
  存在檔案裡

```
107 if __name__ == '__main__':
108     train()
109     train_loss_file=open('train_loss_file.txt', 'w')
110     train_acc_file=open('train_acc_file.txt', 'w')
111     #test_loss_file=open('test_loss_file.txt', 'w')
112     #test_acc_file=open('test_acc_file.txt', 'w')
113
114     for i in result_train_loss:
115         train_loss_file.write(i)
116         train_loss_file.write('\n')
117     for i in result_train_acc:
118         train_acc_file.write(i)
119         train_acc_file.write('\n')
120     #for i in result_test_loss:
121     #    test_loss_file.write(i)
122     #    test_loss_file.write('\n')
123     #for i in result_test_acc:
124     #    test_acc_file.write(i)
125     #    test_acc_file.write('\n')
126     train_loss_file.close()
127     train_acc_file.close()
128     #test_loss_file.close()
129     #test_acc_file.close()
```

訓練過程與結果

- 第10個模型進行測試結果

```
(mylab) waynel116@gslave01:~/preset$ CU
Accuracy on the ALL test images: 85 %
Accuracy of street : 83 %
Accuracy of forest : 96 %
Accuracy of   sea : 86 %
Accuracy of mountain : 79 %
Accuracy of glacier : 84 %
Accuracy of buildings : 82 %
test_loss: 0.4336245342095693

test_acc: 0.8546666666666667
```

- 最佳訓練的模型測試結果

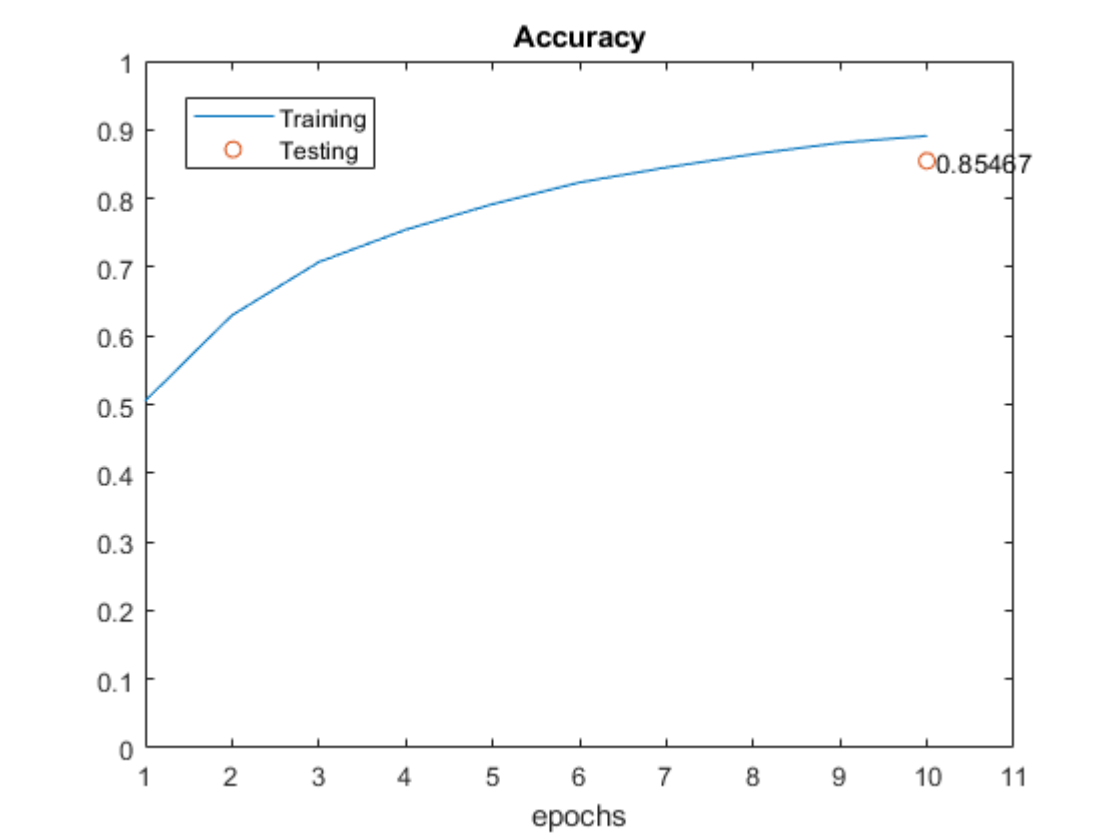
```
Accuracy on the ALL test images: 85 %
Accuracy of street : 83 %
Accuracy of forest : 96 %
Accuracy of   sea : 86 %
Accuracy of mountain : 79 %
Accuracy of glacier : 84 %
Accuracy of buildings : 82 %
test_loss: 0.43362453015645347

test_acc: 0.8546666666666667
```

- 訓練過程

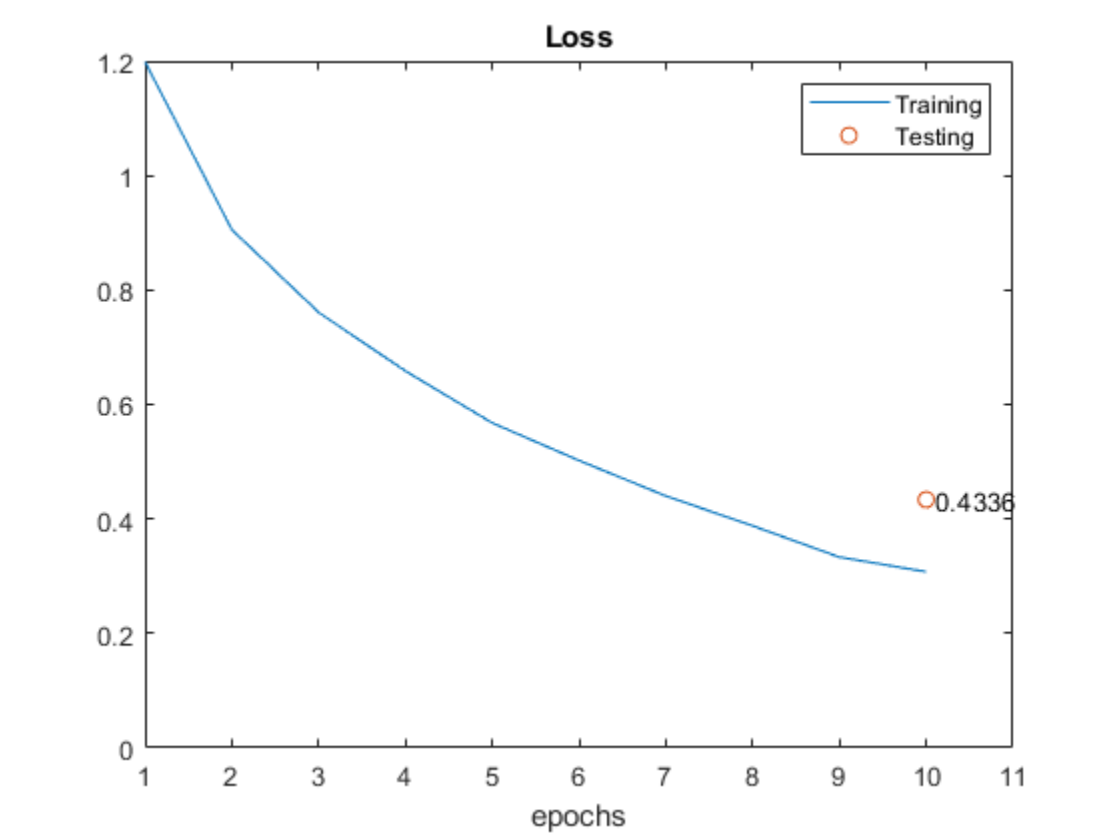
```
Epoch: 1/10
-----------
Training loss: 1.1987   accuracy: 0.5053

Epoch: 2/10
-----------
Training loss: 0.9055   accuracy: 0.6299

Epoch: 3/10
-----------
Training loss: 0.7609   accuracy: 0.7074

Epoch: 4/10
-----------
Training loss: 0.6584   accuracy: 0.7545

Epoch: 5/10
-----------
Training loss: 0.5678   accuracy: 0.7917

Epoch: 6/10
-----------
Training loss: 0.5021   accuracy: 0.8233

Epoch: 7/10
-----------
Training loss: 0.4401   accuracy: 0.8451

Epoch: 8/10
-----------
Training loss: 0.3880   accuracy: 0.8645

Epoch: 9/10
-----------
Training loss: 0.3332   accuracy: 0.8810

Epoch: 10/10
------------
Training loss: 0.3077   accuracy: 0.8911
```

Training+Testing Accuracy/Loss matlab 繪圖

- Training+Testing Accuracy



- Training+Testing Loss



調整參數–考量因素seed

- 調整seed，其他因素不變(batch_size=32, epochs=10, SGD, lr=0.01)

✓ 當seed值從123->0時，seed值=0的訓練最佳結果為90%，而測試整體準確率為83%

✓ 當seed值從123->1024時，seed值=1024的訓練最佳結果為89%，而測試整體準確率為85%

➢ 結論: 儘管seed值=0訓練準確率最好，但測試準確率卻不如預期

- 用最佳訓練模型測試seg_test結果

```
Epoch: 1/10
----------
Training loss: 1.2017   accuracy: 0.5077
Epoch: 2/10
----------
Training loss: 0.8726   accuracy: 0.6584
Epoch: 3/10
----------
Training loss: 0.7347   accuracy: 0.7239
Epoch: 4/10
----------
Training loss: 0.6311   accuracy: 0.7725
Epoch: 5/10
----------
Training loss: 0.5516   accuracy: 0.8038
Epoch: 6/10
----------
Training loss: 0.4786   accuracy: 0.8279
Epoch: 7/10
----------
Training loss: 0.4158   accuracy: 0.8533
Epoch: 8/10
----------
Training loss: 0.3797   accuracy: 0.8655
Epoch: 9/10
----------
Training loss: 0.3534   accuracy: 0.8735
Epoch: 10/10
----------
Training loss: 0.2845   accuracy: 0.9008
```

```
^[[AAccuracy on the ALL test images: 83 %
Accuracy of street : 83 %
Accuracy of forest : 96 %
Accuracy of   sea : 76 %
Accuracy of mountain : 82 %
Accuracy of glacier : 79 %
Accuracy of buildings : 84 %
test_loss: 0.5311218338807424

test_acc: 0.838
```

- seed值=0

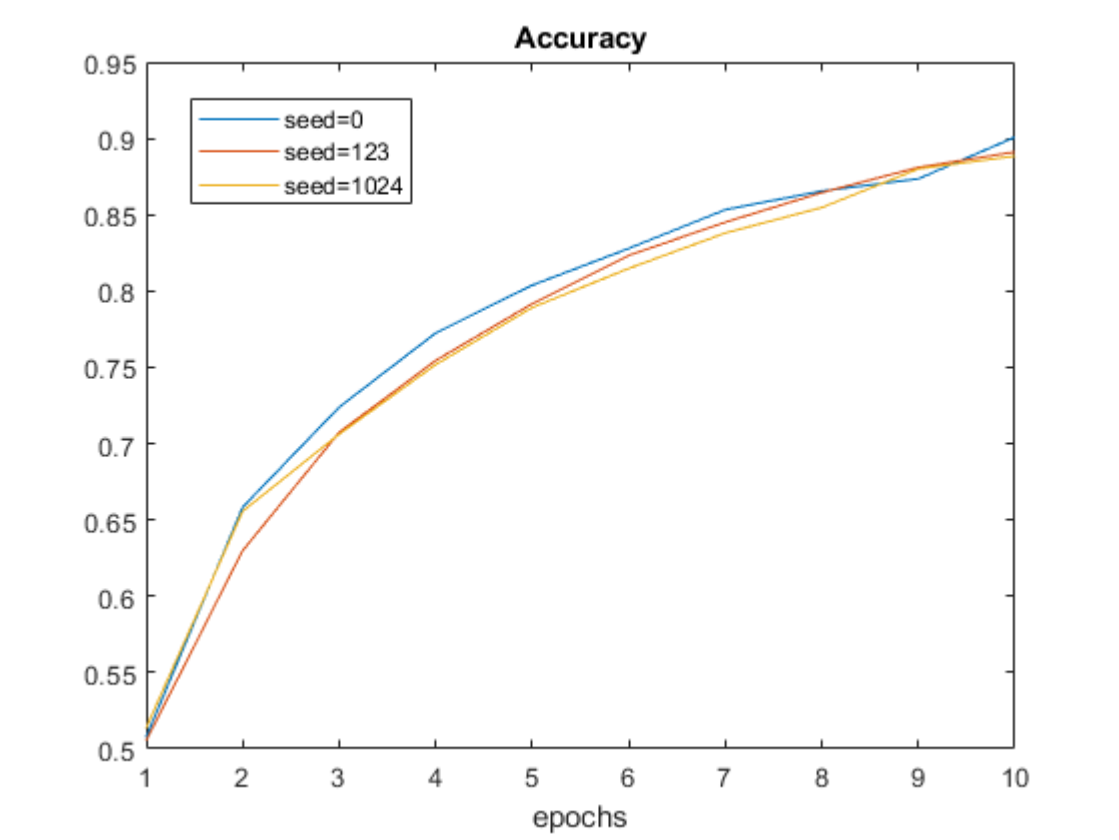
```
Accuracy on the ALL test images: 85 %
Accuracy of street : 91 %
Accuracy of forest : 96 %
Accuracy of   sea : 92 %
Accuracy of mountain : 83 %
Accuracy of glacier : 69 %
Accuracy of buildings : 79 %
test_loss: 0.42865071817239125

test_acc: 0.8513333333333334
```

- seed值=1024

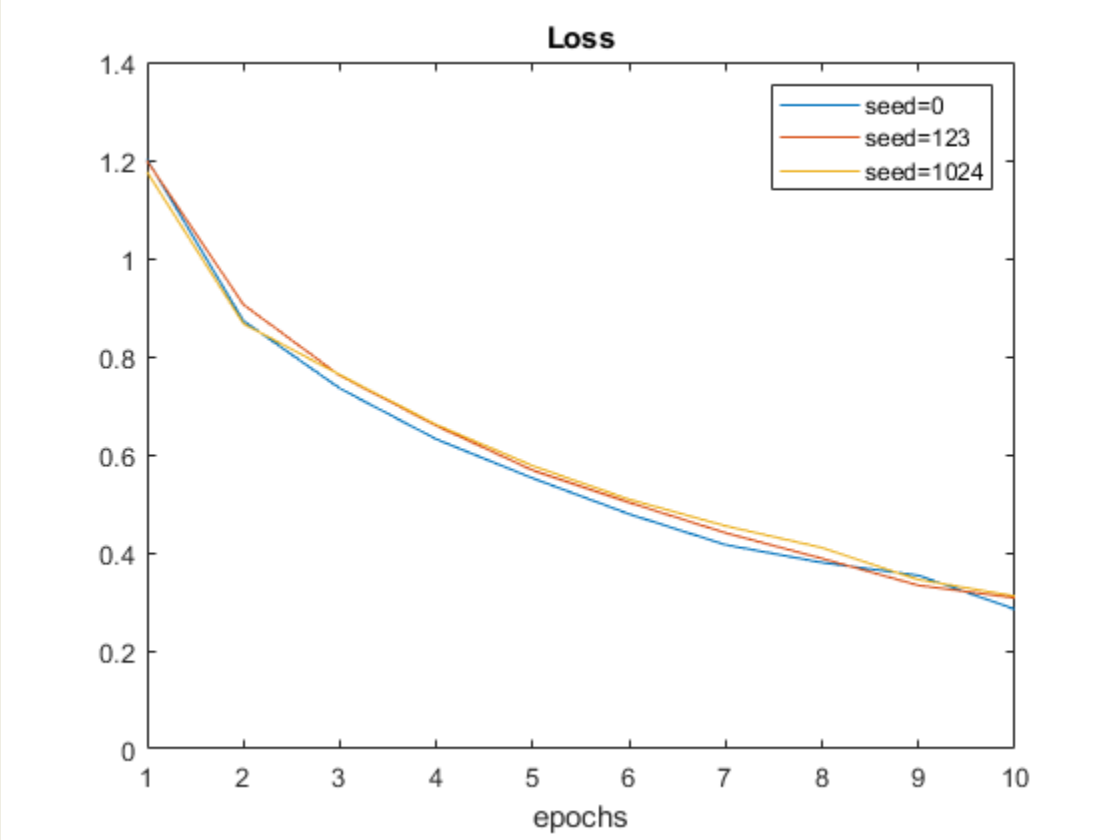
```
Epoch: 1/10
----------
Training loss: 1.1761   accuracy: 0.5137
Epoch: 2/10
----------
Training loss: 0.8666   accuracy: 0.6557
Epoch: 3/10
----------
Training loss: 0.7624   accuracy: 0.7062
Epoch: 4/10
----------
Training loss: 0.6609   accuracy: 0.7518
Epoch: 5/10
----------
Training loss: 0.5768   accuracy: 0.7892
Epoch: 6/10
----------
Training loss: 0.5085   accuracy: 0.8148
Epoch: 7/10
----------
Training loss: 0.4543   accuracy: 0.8380
Epoch: 8/10
----------
Training loss: 0.4100   accuracy: 0.8547
Epoch: 9/10
----------
Training loss: 0.3448   accuracy: 0.8801
Epoch: 10/10
----------
Training loss: 0.3114   accuracy: 0.8884
```

Training Accuracy & loss

- Training Accuracy



- Training loss



調整參數–考量因素batch_size

- 調整batch_size，其他因素不變(Seed=123, epochs=10, SGD, lr=0.01)

✓ 當batch_size從32->64時，batch_size=64的訓練最佳結果為88%，而測試整體準確率為82%

✓ 當batch_size從32->16時，batch_size=16的訓練最佳結果為87%，而測試整體準確率為83%

➢ 結論: 使用batch_size=32測試結果會比較好

◦ 用最佳訓練模型測試seg_test結果

```
(mylab) wayne1116@gslave01:~/hw1_batch
Accuracy on the ALL test images: 83 %
Accuracy of street : 87 %
Accuracy of forest : 91 %
Accuracy of   sea : 80 %
Accuracy of mountain : 76 %
Accuracy of glacier : 81 %
Accuracy of buildings : 87 %
test_loss: 0.4472797746658325

test_acc: 0.838
```

- Batch_size=16

```
(mylab) wayne1116@gslave01:~/hw1_v4$ C
Accuracy on the ALL test images: 85 %
Accuracy of street : 83 %
Accuracy of forest : 96 %
Accuracy of   sea : 86 %
Accuracy of mountain : 79 %
Accuracy of glacier : 84 %
Accuracy of buildings : 82 %
test_loss: 0.4336245404879252

test_acc: 0.8546666666666667
```

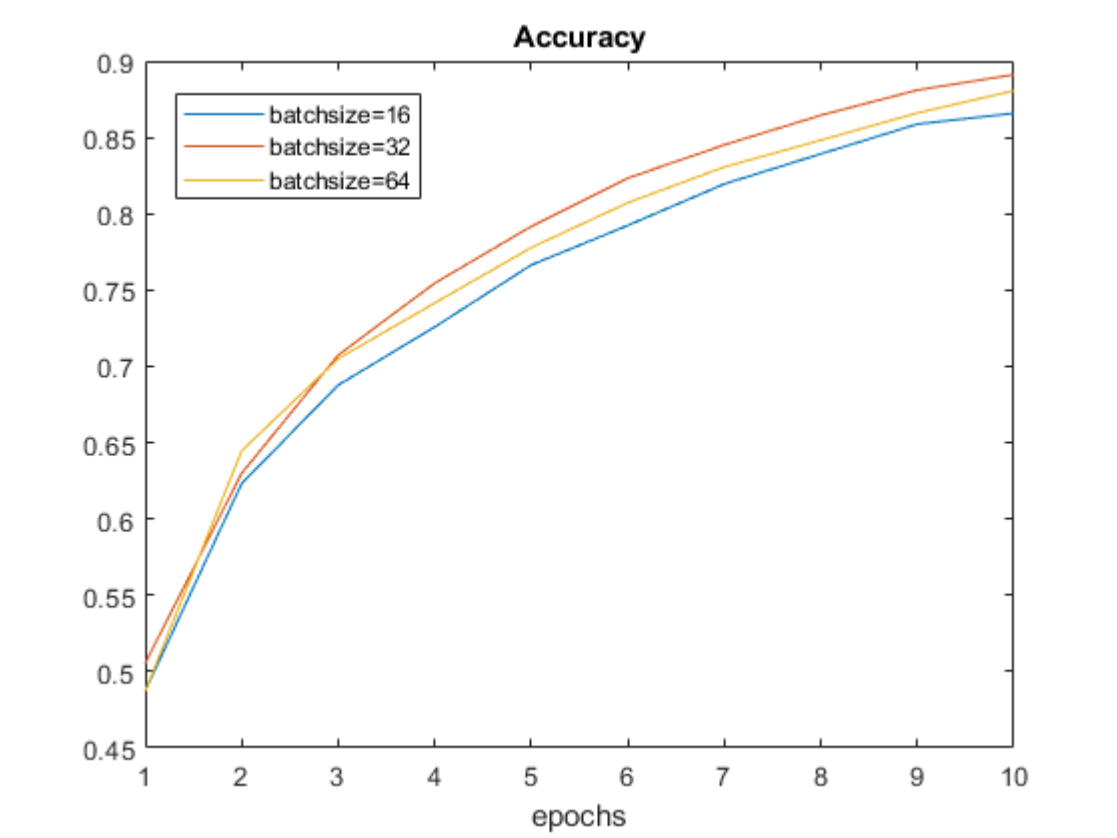
- Batch_size=32

```
(mylab) wayne1116@gslave01:~/hw1_v2$ py
Accuracy on the ALL test images: 82 %
Accuracy of street : 85 %
Accuracy of forest : 90 %
Accuracy of   sea : 96 %
Accuracy of mountain : 62 %
Accuracy of glacier : 81 %
Accuracy of buildings : 76 %
test_loss: 0.5457783784866334

test_acc: 0.8203333333333334
```

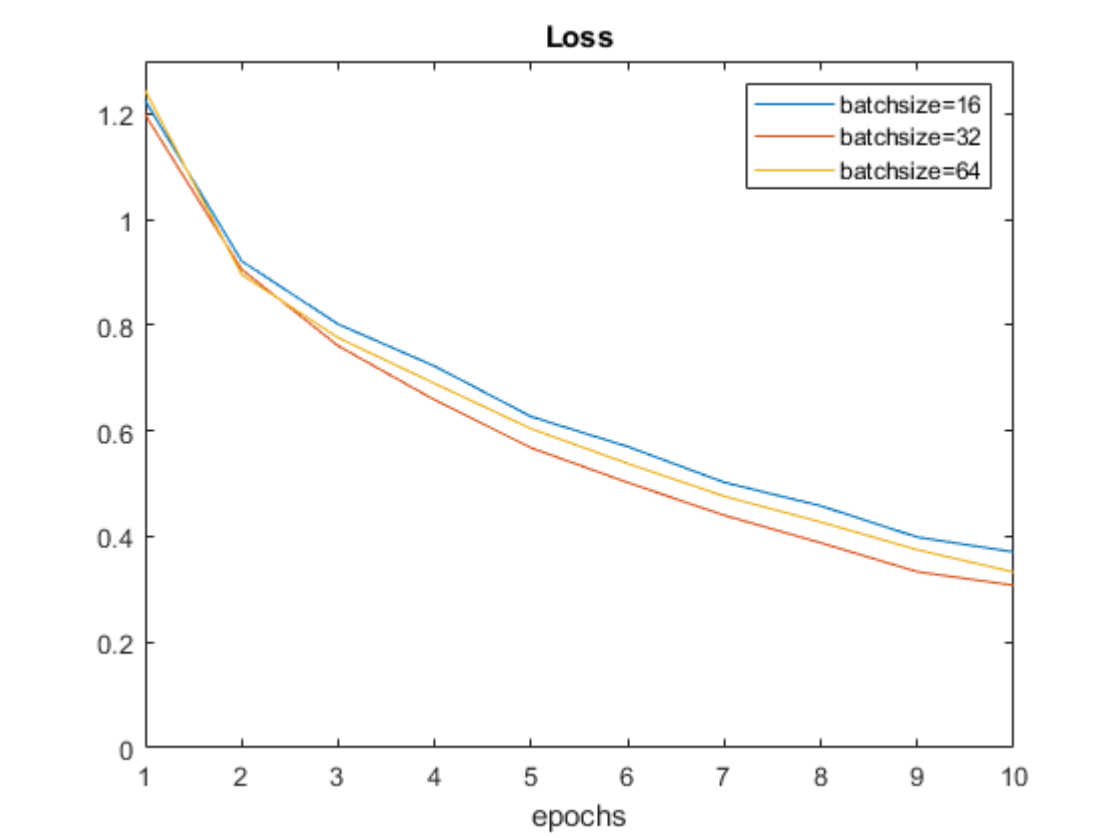
- Batch_size=64

Training Accuracy & loss

- Training Accuracy



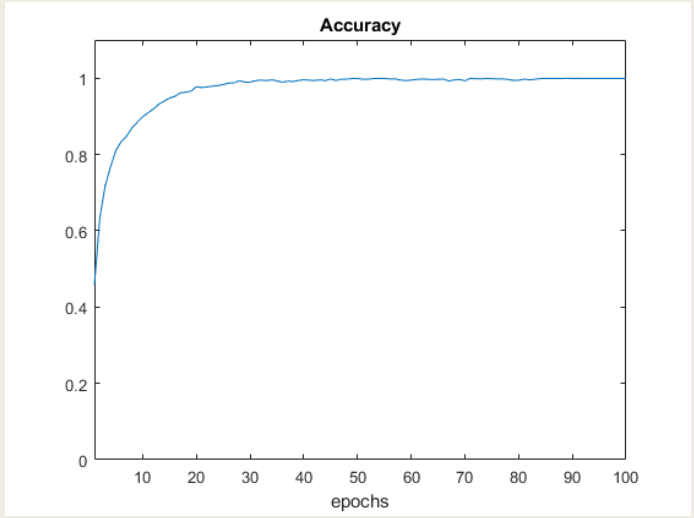
- Training loss

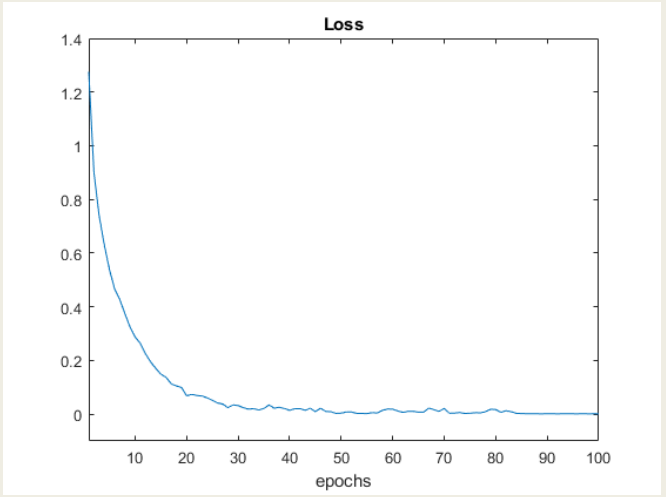


調整參數–考量因素epochs

- 調整epoch，其他因素不變(Seed=123, batch_size=32, SGD, lr=0.01)
- ✓ 將epoch=10改為epoch=100，觀察訓練次數增加時 Accuracy曲線、Loss曲線狀況
- ✓ 結論: 當epochs到達25的時候，訓練準確率已到99%，且Loss也愈接近0

```
Accuracy on the ALL test images: 85 %
Accuracy of   sea : 89 %
Accuracy of buildings : 83 %
Accuracy of forest : 96 %
Accuracy of street : 88 %
Accuracy of glacier : 77 %
Accuracy of mountain : 81 %
```





- Best testing result
- Training Accuracy
- Training loss

調整參數–考量因素learning rate

- 調整lr，其他因素不變(Seed=123, batch_size=32, epoch=10, SGD)

✓ 當lr從0.01->0.1，發現train_loss一直處於nan，lr=0.1的訓練最佳結果為17%，而測試整體準確率為16%

✓ 當lr從0.01->0.001，lr=0.001的訓練最佳結果為85%，而測試整體準確率為85%

✓ 結論: lr=0.01在訓練模型時雖然準確率及loss比lr=0.001好，但測試時相差無幾，而lr=0.01 loss的部分相對較好

◦ 用最佳訓練模型測試seg_test結果

```
Epoch: 1/10
----------
Training loss: nan      accuracy: 0.1693
Epoch: 2/10
----------
Training loss: nan      accuracy: 0.1697
Epoch: 3/10
----------
Training loss: nan      accuracy: 0.1697
Epoch: 4/10
----------
Training loss: nan      accuracy: 0.1697
Epoch: 5/10
----------
Training loss: nan      accuracy: 0.1697
Epoch: 6/10
----------
Training loss: nan      accuracy: 0.1697
Epoch: 7/10
----------
Training loss: nan      accuracy: 0.1697
Epoch: 8/10
----------
Training loss: nan      accuracy: 0.1697
Epoch: 9/10
----------
Training loss: nan      accuracy: 0.1697
Epoch: 10/10
----------
Training loss: nan      accuracy: 0.1697
```

```
(mylab) wayne1116@gslave01:~/hw1_lr$ CU
Accuracy on the ALL test images: 16 %
Accuracy of street : 100 %
Accuracy of forest :  0 %
Accuracy of   sea :  0 %
Accuracy of mountain :  0 %
Accuracy of glacier :  0 %
Accuracy of buildings :  0 %
test_loss: nan

test_acc: 0.167
```

- Learning_rate=0.1

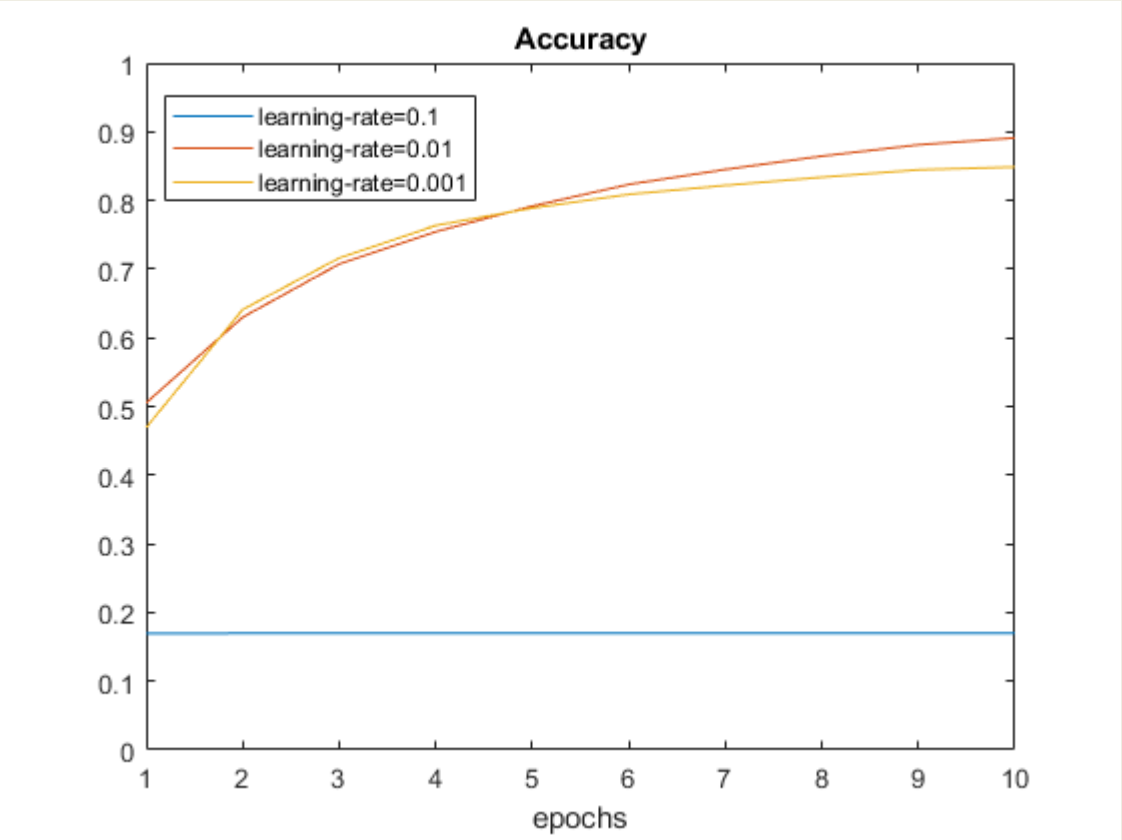
```
Accuracy on the ALL test images: 85 %
Accuracy of street : 91 %
Accuracy of forest : 95 %
Accuracy of   sea : 91 %
Accuracy of mountain : 79 %
Accuracy of glacier : 77 %
Accuracy of buildings : 74 %
test_loss: 0.4236939247449239

test_acc: 0.8523333333333334
```

- Learning_rate=0.001

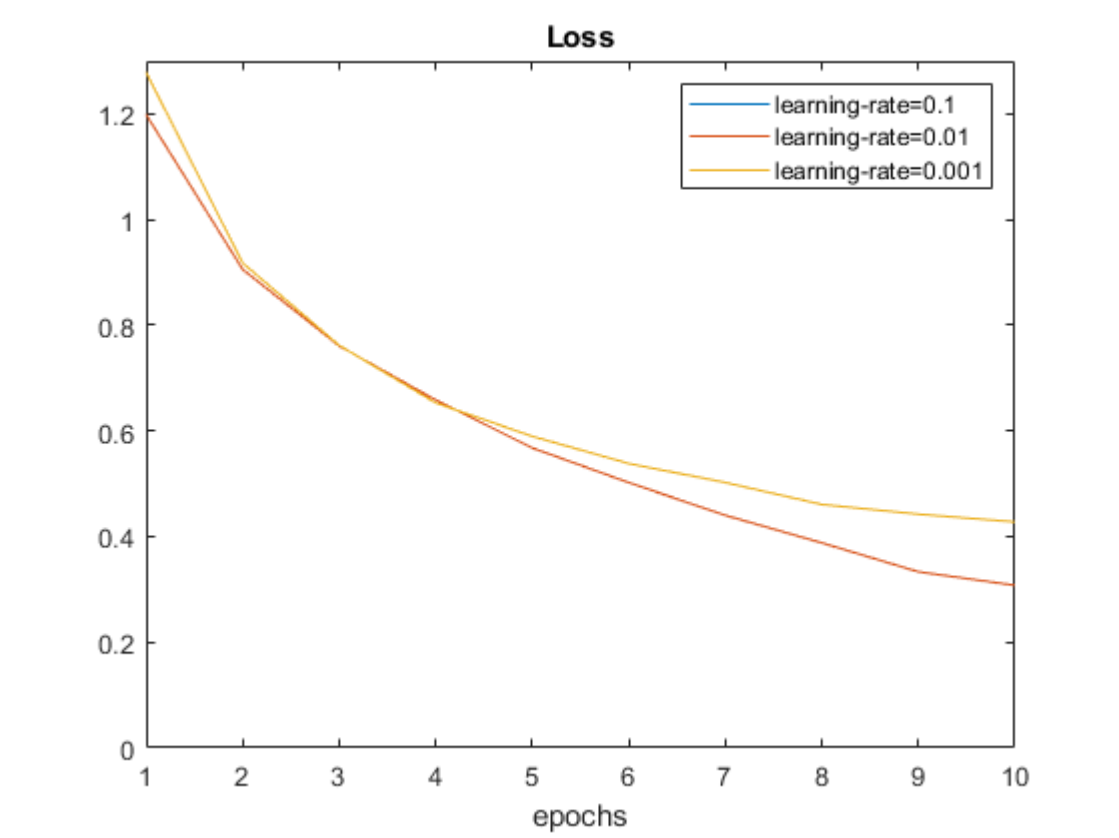
```
Epoch: 1/10
----------
Training loss: 1.2785   accuracy: 0.4692
Epoch: 2/10
----------
Training loss: 0.9177   accuracy: 0.6412
Epoch: 3/10
----------
Training loss: 0.7619   accuracy: 0.7162
Epoch: 4/10
----------
Training loss: 0.6530   accuracy: 0.7635
Epoch: 5/10
----------
Training loss: 0.5897   accuracy: 0.7889
Epoch: 6/10
----------
Training loss: 0.5382   accuracy: 0.8088
Epoch: 7/10
----------
Training loss: 0.5022   accuracy: 0.8219
Epoch: 8/10
----------
Training loss: 0.4607   accuracy: 0.8338
Epoch: 9/10
----------
Training loss: 0.4423   accuracy: 0.8446
Epoch: 10/10
----------
Training loss: 0.4277   accuracy: 0.8487
```

Training Accuracy & loss

- Training Accuracy



- Training loss(learning rate=0.1的loss都是NAN)



調整參數–考量因素optimizer

- 調整optimizer，其他因素不變(Seed=123, batch_size=32, epoch=10)

```
optimizer = torch.optim.Adam(params=model.parameters(), lr=0.001, betas=(0.9,0.999), eps=1e-08, weight_decay=0);
```

✓ 當optimizer改為Adam，Adam的訓練最佳結果為18%，而測試準確率為17%

✓ 結論:雖然訓練時間較SGD短，但準確率卻不如SGD

- 用最佳訓練模型測試seg_test結果

```
Accuracy on the ALL test images: 17 %
Accuracy of street :  0 %
Accuracy of forest :  0 %
Accuracy of   sea :  0 %
Accuracy of mountain : 100 %
Accuracy of glacier :  0 %
Accuracy of buildings :  0 %
test_loss: 1.7899835192362468

test_acc: 0.175
```

```
Epoch: 1/10
----------
Training loss: 2.6425   accuracy: 0.1712
Epoch: 2/10
----------
Training loss: 1.7912   accuracy: 0.1791
Epoch: 3/10
----------
Training loss: 1.7912   accuracy: 0.1751
Epoch: 4/10
----------
Training loss: 1.7911   accuracy: 0.1786
Epoch: 5/10
----------
Training loss: 1.7910   accuracy: 0.1790
Epoch: 6/10
----------
Training loss: 1.7911   accuracy: 0.1786
Epoch: 7/10
----------
Training loss: 1.7910   accuracy: 0.1770
Epoch: 8/10
----------
Training loss: 1.7909   accuracy: 0.1789
Epoch: 9/10
----------
Training loss: 1.7910   accuracy: 0.1790
Epoch: 10/10
----------
Training loss: 1.7910   accuracy: 0.1790
```

統整調整參數後結果

- 最佳結果參數設定

  - seed=1024, batch_size=32, epoch=35, SGD, learning_rate=0.01

  - learning rate每30個epoch會乘上0.1做為調整

```
16  torch.manual_seed(1024)
17  torch.backends.cudnn.deterministic = True
18  torch.backends.cudnn.benchmark = False
```

```
45  #print(DATASET_ROOT)
46  train_set = IMAGE_Dataset(Path(DATASET_ROOT), data_transform)
47  data_loader = DataLoader(dataset=train_set, batch_size=32, shuffle=True, num_workers=1)
48  #print(train_set.num_classes)
```

```
54  best_acc = 0.0
55  count = 100.0
56  num_epochs = 35
57  criterion = nn.CrossEntropyLoss()
58  optimizer = torch.optim.SGD(params=model.parameters(), lr=0.01, momentum=0.9)
```

```
34  def adjust_learning_rate(optimizer, epoch):
35      lr=0.01*(0.1**(epoch//30))
36      for param_group in optimizer.param_groups:
37          param_group['lr']=lr
38
```

```
60      for epoch in range(num_epochs):
61          print(f'Epoch: {epoch + 1}/{num_epochs}')
62          print('-' * len(f'Epoch: {epoch + 1}/{num_epochs}'))
63          adjust_learning_rate(optimizer,num_epochs);
```

最佳結果顯示&過程

- 最佳結果顯示:86%>預設參數85%

```
Accuracy on the ALL test images: 86 %
Accuracy of street : 91 %
Accuracy of forest : 95 %
Accuracy of   sea : 89 %
Accuracy of mountain : 83 %
Accuracy of glacier : 84 %
Accuracy of buildings : 72 %
test_loss: 0.7284157524108886

test_acc: 0.8606666666666667
```

- 執行過程

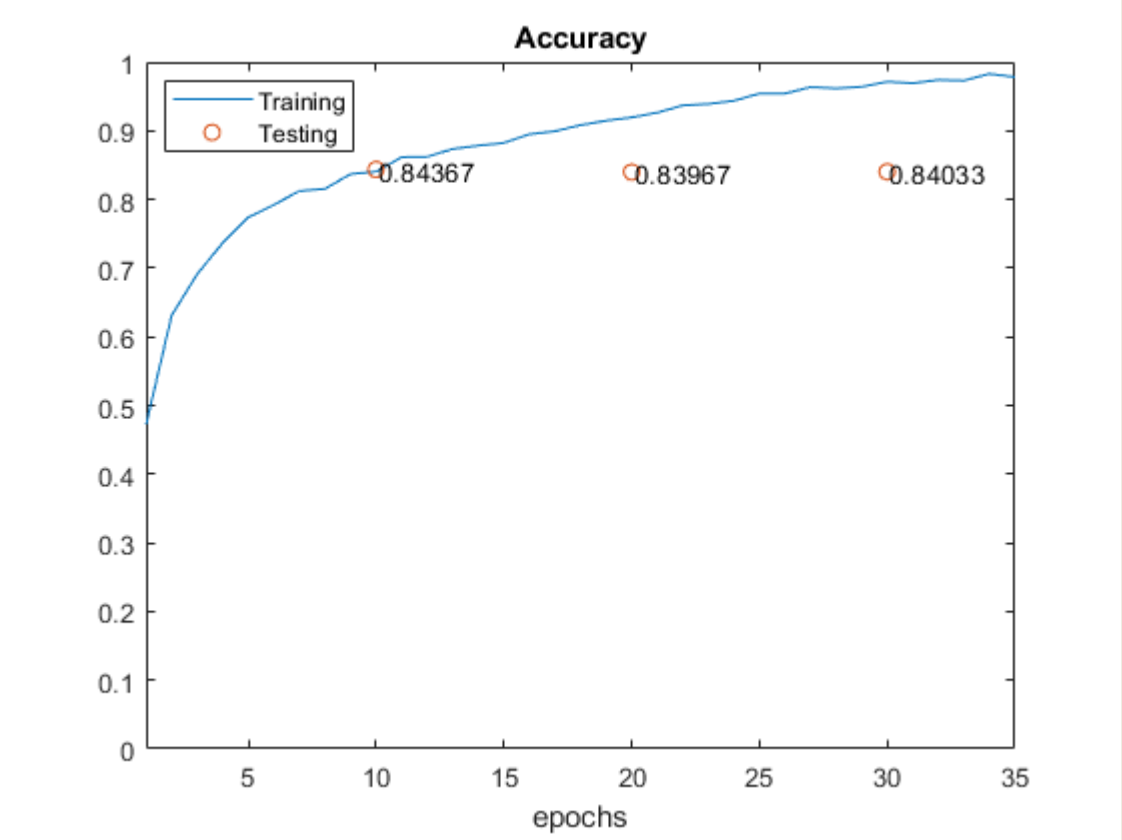
```
Epoch: 1/35
-----------
Training loss: 1.2857   accuracy: 0.4722

Epoch: 2/35
-----------
Training loss: 0.9374   accuracy: 0.6312

Epoch: 3/35
-----------
Training loss: 0.8162   accuracy: 0.6913

Epoch: 4/35
-----------
Training loss: 0.7131   accuracy: 0.7370

Epoch: 5/35
-----------
Training loss: 0.6247   accuracy: 0.7741

Epoch: 6/35
-----------
Training loss: 0.5644   accuracy: 0.7922

Epoch: 7/35
-----------
Training loss: 0.5246   accuracy: 0.8123

Epoch: 8/35
-----------
Training loss: 0.4986   accuracy: 0.8154

Epoch: 9/35
-----------
Training loss: 0.4610   accuracy: 0.8370

Epoch: 10/35
-----------
Training loss: 0.4401   accuracy: 0.8405

Epoch: 11/35
-----------
Training loss: 0.3985   accuracy: 0.8616

Epoch: 12/35
-----------
Training loss: 0.3836   accuracy: 0.8621
```

```
Epoch: 13/35
-----------
Training loss: 0.3558   accuracy: 0.8735

Epoch: 14/35
-----------
Training loss: 0.3338   accuracy: 0.8784

Epoch: 15/35
-----------
Training loss: 0.3296   accuracy: 0.8822

Epoch: 16/35
-----------
Training loss: 0.2998   accuracy: 0.8949

Epoch: 17/35
-----------
Training loss: 0.2771   accuracy: 0.8993

Epoch: 18/35
-----------
Training loss: 0.2591   accuracy: 0.9082

Epoch: 19/35
-----------
Training loss: 0.2389   accuracy: 0.9147

Epoch: 20/35
-----------
Training loss: 0.2268   accuracy: 0.9194

Epoch: 21/35
-----------
Training loss: 0.2008   accuracy: 0.9263

Epoch: 22/35
-----------
Training loss: 0.1766   accuracy: 0.9369

Epoch: 23/35
-----------
Training loss: 0.1659   accuracy: 0.9390

Epoch: 24/35
-----------
Training loss: 0.1603   accuracy: 0.9436

Epoch: 25/35
-----------
Training loss: 0.1294   accuracy: 0.9542
```

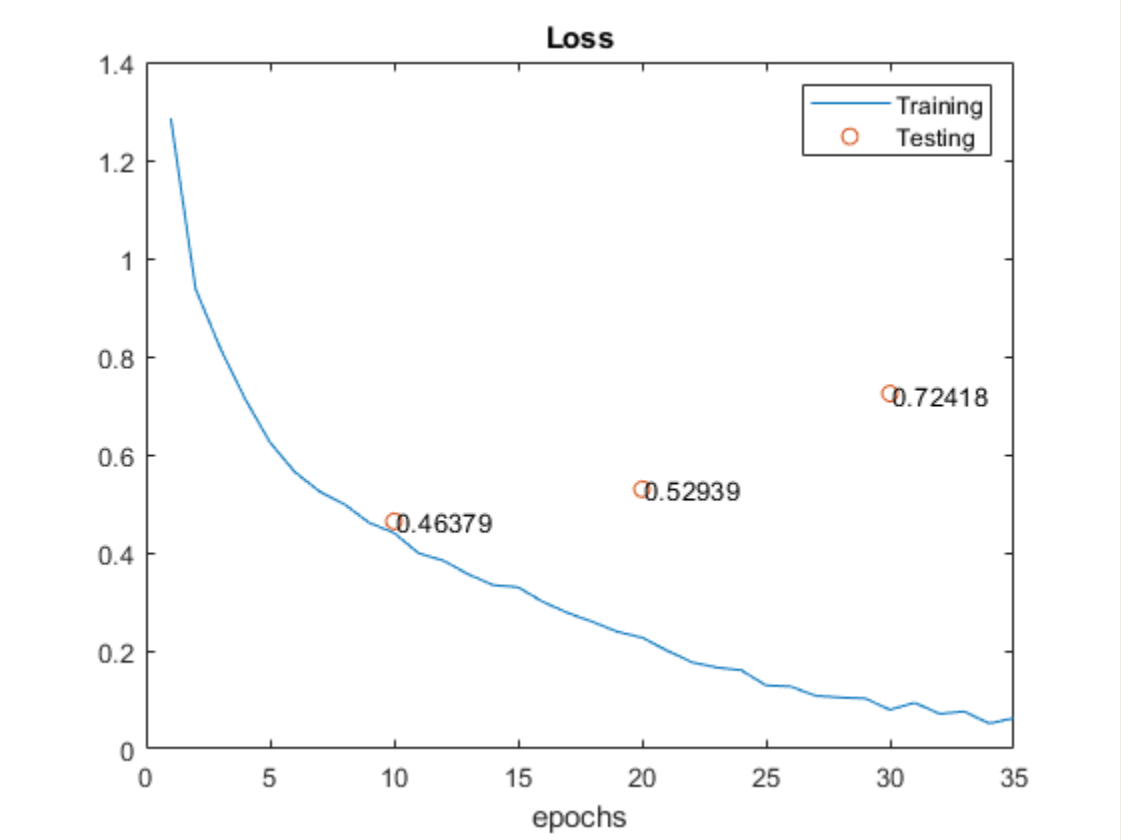
```
Epoch: 26/35
-----------
Training loss: 0.1271   accuracy: 0.9541

Epoch: 27/35
-----------
Training loss: 0.1081   accuracy: 0.9635

Epoch: 28/35
-----------
Training loss: 0.1047   accuracy: 0.9617

Epoch: 29/35
-----------
Training loss: 0.1027   accuracy: 0.9639

Epoch: 30/35
-----------
Training loss: 0.0798   accuracy: 0.9714

Epoch: 31/35
-----------
Training loss: 0.0941   accuracy: 0.9692

Epoch: 32/35
-----------
Training loss: 0.0714   accuracy: 0.9740

Epoch: 33/35
-----------
Training loss: 0.0762   accuracy: 0.9731

Epoch: 34/35
-----------
Training loss: 0.0518   accuracy: 0.9828

Epoch: 35/35
-----------
Training loss: 0.0626   accuracy: 0.9782
```

Training+Testing Accuracy/Loss matlab 繪圖

- Training+Testing Accuracy



- Training+Testing Loss



結論Discussion

1. 根據預設參數以及調整參數的數據顯示，對於森林的分類有較高的準確率

2. 根據自己調整參數數據顯示，每10次epoch對現在model做一次testing的loss值明顯上升

3.根據數據顯示，儘管自己調整參數training accuracy達到97%，而預設參數的training accuracy只有89%，但是兩者的testing accuracy卻沒有明顯的差異(86%,85%)

4.Batch size越大，訓練時間越短

5.當訓練次數越多，training accuracy就越高

問題與困難Problem & difficulties

1. 使用工學院的深度學習伺服器訓練模型，原本電腦都要一直開著，而且有時候還會中斷連線，而且那時候epoch是50，所以每一次都需要重跑一次，後來SA提供了nohup的方法

2.我一開始以為training accuracy越高，testing accuracy應該也要越高才對，但跑完模型並testing後發現不是這樣

3.對於optimizer的選擇，對於Adam optimizer結合了AdaGrad依照梯度去調整learning rate的特性，以及SGD中momentum對梯度方向做梯度速度調整，理論上應該會比這兩個optimizer結果還要來的好，但是經由實驗數據顯示卻不如預期

4. 因為當時我想要測試最佳model的準確率，但是深度學習伺服器都有人使用，所以我將model從伺服器下載下來並使用專題實驗室附有GPU的電腦測試最佳model準確率，但是結果很奇怪都是30%，最後我重新回到伺服器測試才比較正常，準確率為85%

參考文獻Reference

- ［機器學習ML NOTE]SGD,Momentum,AdaGrad,Adam optimizer

https://medium.com/%E9%9B%9E%E9%9B%9E%E8%88%87%E5%85%94%E5%85%94%E7%9A%84%E5%B7%A5%E7%A8%8B%E4%B8%96%E7%95%8C/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92ml-note-sgd-momentum-adagrad-adam-optimizer-f20568c968db

https://blog.csdn.net/aliceyangxi1987/article/details/73210204

- Batch_size

https://www.leiphone.com/news/201710/RIIlL7LdIlT1Mvm8.html

https://www.zhihu.com/question/32673260

- Deep learning

https://ithelp.ithome.com.tw/articles/10189086