方法描述-選擇參數的策略

這次作業要關注的參數有 torch.manual_seed()、batch size、epoch、learning rate。

torch.manual_seed()是為了讓當前的GPU設置隨機種子,因為訓練開始時，參數的初始化是隨機的，torch.manual_seed()可以讓每次的結果一致。

由於上網查都沒有找到關於設置 torch.manual_seed()的較佳建議，所以我測試作業要求的 123 還要小的 0、作業要求的 123 還要大的 2019。

batch size 的意義是將數據分成若干批，按批來更新參數。一個批中的一組數據共同決定了這次梯度的方向，使梯度下降較不容易跑偏。一開始在我的筆電測試時設定 batch size=32，但是筆電記憶體不夠導致無法執行程式。所以嘗試 batch size=16。後來等到 GPU 後，也嘗試了 batch size=64。上網查是說，batch size 沒有一定最佳的準則，要依據各個資料集做調整，所以就老實的慢慢嘗試。

epoch 的意思是訓練過程中數據將被輪幾次。epoch 跟 batch size 都是沒有一定最佳的準則，要依據各個資料集做調整，所以也老實的慢慢嘗試。不過上網查有一個網站說，通常 epoch 設定 40~50 就差不多了。

learning rate 控制模型的學習進度，即 stride(步長)

$$\omega^n \leftarrow \omega^n - \eta \frac{\partial L}{\partial \omega^n}$$ 就是反向傳播法中的 η

	learning rate 大	learning rate 小
學習速度	快	慢
使用時間點	剛開始訓練時	一定輪數過後
缺點	易 loss 值爆炸、易震盪	易過擬合、收斂速度慢

上網查到，0.01~0.001 通常較好，也可以設定≤10^{-4}後，再進行微調。一開始我設定 0.1，每次除以 10 去嘗試。

實驗結果

我想測試每個參數對準確率的影響，所以作了以下的實驗 ：

只調整 seed:

	seed=0	seed=123	seed=2019
batch_size=32, epoch=10, lr=0.01	82%	84%	83%

只調整 batch_size:

	batch_size=16	batch_size=32	batch_size=64
seed=123, epoch=10, lr=0.01	82%	84%	82%

只調整 epoch:

	epoch=10	epoch=20	epoch=30
seed=123, batch_size=32, lr=0.01	84%	82%	83%

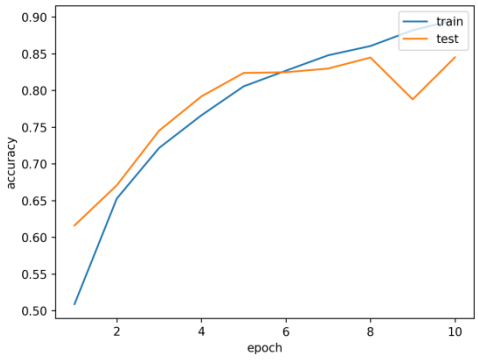
只調整 lr:

	lr=0.1	lr=0.01	lr=0.001
seed=123, batch_size=32, epoch=10	16%	84%	83%

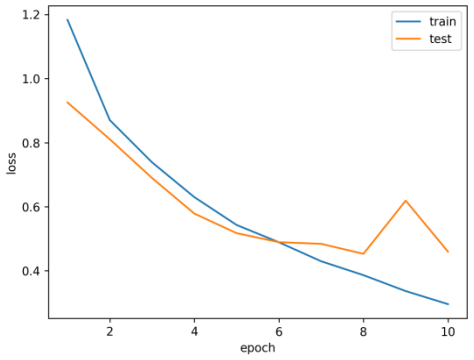
seed=123, batch_size=32, epoch=10, lr=0.01(作業要求):

```
color@mclab-WS660T: ~/89 [86x23]
連線(C)  編輯(E)  檢視(V)  視窗(W)  選項(O)  說明(H)
-----------
Training loss: 0.4297   accuracy: 0.8477

Epoch: 8/10
-----------
Training loss: 0.3864   accuracy: 0.8603

Epoch: 9/10
-----------
Training loss: 0.3367   accuracy: 0.8819

Epoch: 10/10
-----------
Training loss: 0.2960   accuracy: 0.8959

Accuracy on the ALL test images: 84 %
Accuracy of street : 85 %
Accuracy of mountain : 81 %
Accuracy of forest : 94 %
Accuracy of   sea : 89 %
Accuracy of buildings : 82 %
Accuracy of glacier : 75 %
                                                    47,1          Bot
```

accuracy

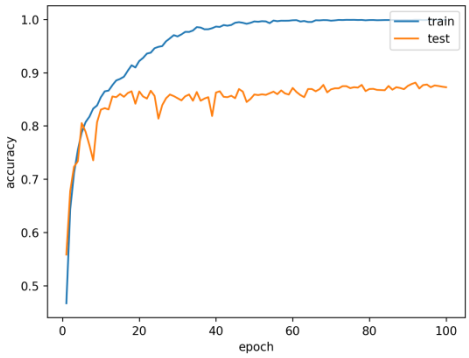
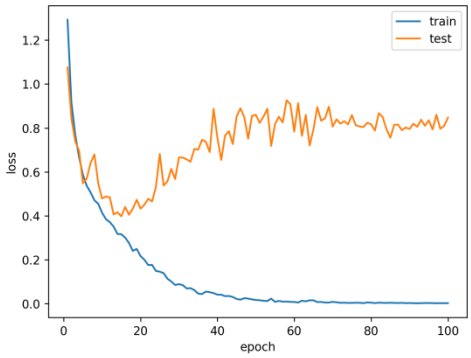


loss



seed=123, batch_size=32, epoch=100, lr=0.001:

```
color@mclab-WS660T: ~/89 [86x23]
連線(C)  編輯(E)  檢視(V)  視窗(W)  選項(O)  說明(H)
-------------
Training loss: 0.0023   accuracy: 0.9994

Epoch: 98/100
-------------
Training loss: 0.0026   accuracy: 0.9992

Epoch: 99/100
-------------
Training loss: 0.0024   accuracy: 0.9991

Epoch: 100/100
--------------
Training loss: 0.0029   accuracy: 0.9989

Accuracy on the ALL test images: 87 %
Accuracy of street : 88 %
Accuracy of mountain : 84 %
Accuracy of forest : 96 %
Accuracy of   sea : 89 %
Accuracy of buildings : 87 %
Accuracy of glacier : 79 %
                                                    407,1         Bot
```

accuracy	loss
	

討論

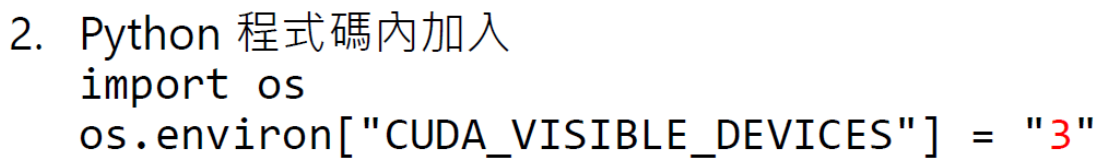
在我針對只調整 seed、只調整 batch_size、只調整 epoch、只調整 lr 的實驗中，發現並非參數越大、準確率越好。而是要針對各種資料集進行測試，找出適合的參數。

在作業要求的 seed=123, batch_size=32, epoch=10, lr=0.01 中，發現到最後幾次 epoch 時，test 的 loss 會上升，發生過擬合現象。

而在我自己測試的 seed=123, batch_size=32, epoch=100, lr=0.001 中過擬合的現象在整個訓練中更快的發生，test 的曲線很快就開始震盪。我想是因為 learning rate 設定較小(lr=0.001)的緣故。

問題與困難

一開始我在程式碼中按照助教的投影片加入指定使用哪一顆 GPU 的指令(如下圖)



但是會一直出現找不到 CUDA 的錯誤回報，後來不使用這段程式碼後就解決了。我想是因為 torch 對參數處存的位置有記憶，而我有指定換過 CUDA，才會發生這樣的問題。

要把 train 和 test 的數據畫在圖一張圖上，代表需要在一支程式中同時存在 train 和 test 的數據。

我多放了一個 hw1.py 的檔案，是把 train 和 test 結合成一支程式，所以只要執行 hw1.py 程式就可以產生 accuracy 和 loss 的曲線圖、testing accuracy，不產生 pth 檔。

train.py 和 test.py 是跟作業要求的規定一樣，先執行 train 產生 pth 檔、accuracy 和 loss 的曲線圖，再執行 test 得到 testing accuracy。

在儲存 train 和 test 的 accuracy、loss 的數據時，我使用 extend 的方式把數據存到 list 裡面，但是卻出現
train_loss_list.extend(training_loss)

TypeError: 'float' object is not iterable

的錯誤回報。後來改成用 append 才可以執行。