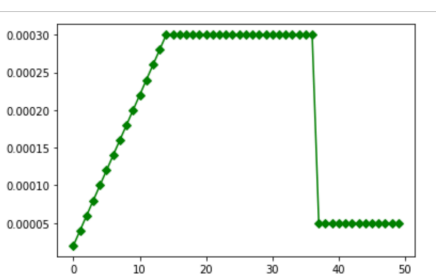
Lab2 - report

Implement part

在實作ResNet的部分，只需要按照投影片中的表格，依序的的模型建構起來就行了，主要是由basic block建構而成，因為這次可以使用pytorch所以順暢很多。

在實作learning rate schdule的部分，只需要將我的optimzer的learning rate讀出來，並且修改，就可以完成了。為了簡潔程式碼，所以我另外開了function來設定不同learning rate的規劃。

```python
def warn_up(optimizer, epoch):
    if epoch < 15:
        optimizer.param_groups[0]['lr'] = 0.00002*epoch
    elif epoch < 38:
        optimizer.param_groups[0]['lr'] = 0.0003
    else:
        optimizer.param_groups[0]['lr'] = 0.00005
```



因為resnet本身的模型架構簡單，所以多train幾個epoch就提升validation，下圖是超過門檻的證明。

```
Epoch: 41 |Train Loss:  0.1368 |Train Acc:93.6033 |Val Loss:  0.2153 |Val Acc:90.2128
Epoch: 42 |Train Loss:  0.1249 |Train Acc:93.8711 |Val Loss:  0.1942 |Val Acc:91.0638
Epoch: 43 |Train Loss:  0.1299 |Train Acc:94.2020 |Val Loss:  0.1995 |Val Acc:89.6454
Epoch: 44 |Train Loss:  0.1245 |Train Acc:94.2020 |Val Loss:  0.2104 |Val Acc:90.2128
Epoch: 45 |Train Loss:  0.1290 |Train Acc:93.8869 |Val Loss:  0.2176 |Val Acc:89.9291
Epoch: 46 |Train Loss:  0.1259 |Train Acc:94.2493 |Val Loss:  0.2119 |Val Acc:90.2128
Epoch: 47 |Train Loss:  0.1309 |Train Acc:93.6821 |Val Loss:  0.2042 |Val Acc:89.5035
Epoch: 48 |Train Loss:  0.1234 |Train Acc:94.1862 |Val Loss:  0.2346 |Val Acc:88.5106
Epoch: 49 |Train Loss:  0.1228 |Train Acc:93.9814 |Val Loss:  0.2386 |Val Acc:89.5035
Epoch: 50 |Train Loss:  0.1203 |Train Acc:94.2335 |Val Loss:  0.2191 |Val Acc:89.2199
```

Compare pretrain-nonpretrain model

比較兩個是否預設weight的模型可以發現，如果模型經過pretrain，可以更快的收斂，在每次更新之後，Acc都有提高比較多、loss的遞減也快很多。主要原因是因為convolution中有大量的參數，如果沒有經過好好的初始化，會讓參數非常雜亂，convolution的效果也會變差，因而導致back propagaiton效率很低。

```
-------------saving model--------------
Epoch: 46 |Train Loss:  0.4539 |Train Acc:82.5902 |Val Loss:  0.7605 |Val Acc:73.9007
Epoch: 47 |Train Loss:  0.4412 |Train Acc:82.8265 |Val Loss:  0.7333 |Val Acc:72.4823
Epoch: 48 |Train Loss:  0.4471 |Train Acc:82.8896 |Val Loss:  0.6784 |Val Acc:75.0355
Epoch: 49 |Train Loss:  0.4456 |Train Acc:83.6143 |Val Loss:  0.7212 |Val Acc:74.7518
Epoch: 50 |Train Loss:  0.4467 |Train Acc:82.8108 |Val Loss:  0.7943 |Val Acc:72.6241
```

上圖為non-pretrain、下圖為pretrained model

```
Epoch: 46 |Train Loss:  0.1438 |Train Acc:93.5087 |Val Loss:  0.2868 |Val Acc:88.7943
Epoch: 47 |Train Loss:  0.1508 |Train Acc:93.0518 |Val Loss:  0.3055 |Val Acc:87.5177
Epoch: 48 |Train Loss:  0.1366 |Train Acc:93.4142 |Val Loss:  0.3350 |Val Acc:87.5177
Epoch: 49 |Train Loss:  0.1403 |Train Acc:93.6348 |Val Loss:  0.3070 |Val Acc:88.3688
Epoch: 50 |Train Loss:  0.1356 |Train Acc:93.7608 |Val Loss:  0.3181 |Val Acc:88.5106
```

Advance

Machine

在這次的比賽中，如果使用比resnet18還大的模型，很容易會造成實驗室的server GPU RAM不足，因此我借用到了多個TWCC的tesla-V100，讓我在訓練上速度跟RAM都有更大的彈性，尤其是之後會提到的ResNet101模型，不過在處理SSH上花了很多時間，而且錢包QQ

另外我還有用我自己的桌機RTX 3060ti，但是因為todevice指令不知為何會消耗很久的時間，所以並沒有太多的嘗試就轉回TWCC了

(上圖為我的乾爹 TWCC)

Tried model

這次lab中我先嘗試了一些比較複雜的模型，像是ResNet系列的ResNet18, ResNet34, ResNet50, ResNet101, VGG16，還有在cifar-100中的state-of-the-art EfficientNetV2（之所以賄選在cifar100表現最好的是因為圖片尺寸類似）。但在這些模型上的表現都沒有到太好，尤其是在ResNet101中，可以看到model在第一個epoch就來到95%的精準度，但是valid去只有73%，這很明顯是overfitting的現象；如果從理論想，其實也很明顯，因為大模型的參數很多，很容易就fit input data's feature，而且同時會因為參數過多，導致更新起來效率很低。另外比對ResNet34, 50, 101，可以發現隨著模型的化減，確實performace有變好，在VGG16, EffiecientNetv2上也是相同的結果，證明了不應該選過大的推測，因此我並沒有過多的嘗試，就選擇改良ResNet34。

ResNet101

```
Epoch:  3 |Train Loss:  0.2823 |Train Acc:94.6090 |Val Loss:  3.0172 |Val Acc:68.0851
Epoch:  4 |Train Loss:  0.0914 |Train Acc:94.9127 |Val Loss:  2.3085 |Val Acc:70.2128
Epoch:  5 |Train Loss:  0.0748 |Train Acc:95.9757 |Val Loss:  1.8607 |Val Acc:69.9088
Epoch:  6 |Train Loss:  0.0716 |Train Acc:96.3554 |Val Loss:  3.0820 |Val Acc:68.3891
Epoch:  7 |Train Loss:  0.0669 |Train Acc:96.4313 |Val Loss:  1.5577 |Val Acc:69.9088
Epoch:  8 |Train Loss:  0.0721 |Train Acc:95.8238 |Val Loss:  2.1124 |Val Acc:69.6049
Epoch:  9 |Train Loss:  0.0730 |Train Acc:95.9757 |Val Loss:  1.5076 |Val Acc:70.5167
Epoch: 10 |Train Loss:  0.0632 |Train Acc:96.7350 |Val Loss:  2.6944 |Val Acc:66.8693
Epoch: 11 |Train Loss:  0.1348 |Train Acc:96.5072 |Val Loss:  2.7958 |Val Acc:68.6930
Epoch: 12 |Train Loss:  0.0688 |Train Acc:96.6591 |Val Loss:  1.9288 |Val Acc:70.2128
Epoch: 13 |Train Loss:  0.0604 |Train Acc:96.3554 |Val Loss:  3.2485 |Val Acc:69.6049
Epoch: 14 |Train Loss:  0.0667 |Train Acc:96.5072 |Val Loss:  2.3411 |Val Acc:68.3891
Epoch: 15 |Train Loss:  0.1180 |Train Acc:96.2794 |Val Loss:  5.5952 |Val Acc:66.2614
Epoch: 16 |Train Loss:  0.0622 |Train Acc:96.7350 |Val Loss:  2.1241 |Val Acc:68.0851
Epoch: 17 |Train Loss:  0.1100 |Train Acc:96.4313 |Val Loss:  2.3262 |Val Acc:70.2128
Epoch: 18 |Train Loss:  0.0633 |Train Acc:96.5831 |Val Loss:  1.5448 |Val Acc:69.6049
Epoch: 19 |Train Loss:  0.0837 |Train Acc:96.6591 |Val Loss:  1.6912 |Val Acc:69.9088
Epoch: 20 |Train Loss:  0.0645 |Train Acc:97.3424 |Val Loss:  3.0748 |Val Acc:68.0851
Epoch: 21 |Train Loss:  0.0613 |Train Acc:96.6591 |Val Loss:  3.9571 |Val Acc:68.3891
Epoch: 22 |Train Loss:  0.0826 |Train Acc:96.3554 |Val Loss:  1.9495 |Val Acc:69.3009
Epoch: 23 |Train Loss:  0.0813 |Train Acc:96.5072 |Val Loss:  2.0527 |Val Acc:69.3009
Epoch: 24 |Train Loss:  0.0731 |Train Acc:96.1276 |Val Loss:  2.9187 |Val Acc:69.9088
Epoch: 25 |Train Loss:  0.0621 |Train Acc:96.2794 |Val Loss:  2.5492 |Val Acc:67.4772
```

Data argument

我認為這是最大的挑戰，經過統計，可以發現這次除了data量很少，還有樣本數偏差的問題，actinic keratosis資料量只有其他皮膚病的1/3不到；如果點進dataset中檢查資料，可以發現有些圖片的背景是藍色的、而大多數的是皮膚色的，這在電腦視覺上會有很大的問題。

```
data name: nevus                                Num: 336
data name: basal cell carcinoma                 Num: 353
data name: melanoma                             Num: 409
data name: actinic keratosis                    Num: 117
data name: pigmented benign keratosis           Num: 431
```

根據上面幾個問題，我分別對資料做幾個操作，最必要的事normalization跟resize，在沒有經過normalization的dataset色差很大，幾乎無法訓練，在normalization的hyper parameter上，我選擇image net使用的數據。再來是Random Rotate，考量到皮膚病的特性「沒有方向」，所以可以利用任意旋轉圖片，之後再利用CenterCrop的技巧，增加dataset；跟上一點相同的理由，我們可以利用沒有方向性的性質，垂直/水平翻轉照片。再來是幾個比較特殊的技巧，我們可以利用模糊化解決圖片中毛，所以我採用gaussian等作法，增加dataset，另外還有想到laplcian，考量他會完全改變圖片的相貌，所以最後沒有實裝這個功能。最後是對比度，因為皮膚病在傷口/皮膚上有明顯色差，因此可以利用這個性質加強圖片的特徵，在performace上增加一點表現。

在data argument上有個隱患，即在資料量比較少時，會因為大量翻轉/旋轉產出相似照片，相似的照片可能會被分到validation set，當模型overfitting on training set的同時，也會overfitting在相似的照片（validation set），造成無法有效評估模型的好壞，這應該也是為什麼我總是可以在validation上拿到優異成績，上傳kaggle卻有明顯差距。

Fully connected layers

原先ResNet18的output為512個class，如果直接修改成5，會讓參數急遽變化，影響performance，因此我用了兩個方案替代，一個是(512, 128, 32, 5)另一個則是(512, 64, 5)，都對結果有不錯的表現；另外為了降低參數的差距，我選擇在每一個fully connected層後加上Batch Normalization，讓參數比較不容易兩極化，減緩Gradient Appear和Internal Covariate Shift。

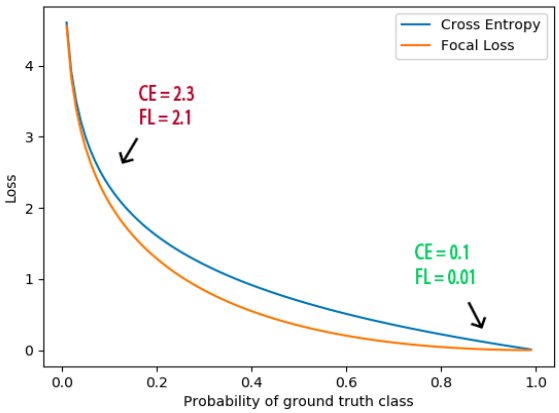
Optimizers/ Loss

經過交叉使用Adam, SGD作為Optimizers，我認為在Adam上收斂速度比較快，因此之後的訓練主要都是用Adam作為Optimizers。

最主流的classification loss是CrossEntropy，一開始我也是選用loss function，但在遇到performance瓶頸時，我和學長討論使用focal loss-也就是特化的CrossEntropy(When alpha = 1, r = 0)

會選用focal loss是因為他有這樣的性質：

對於比較容易分類且正確分類的樣本，會因為pt->1，是他對loss的貢獻比較小；在比較難分類樣本中，會有比較多的loss貢獻，讓模型更注重比較困難的樣本。



Others trick

另外我還有對模型做這些以下幾個優化：

Freeze model，在advance中我們可以使用pretrain model，而這些參數都已經更新的差不多，而fully connected network則是完全沒有被更新的狀況，因此我選擇先將ResNet的參數凍結不更新，先train數個epoch的fully connected Network，等FC比較fit模型的時候，我再一起更新整體模型。

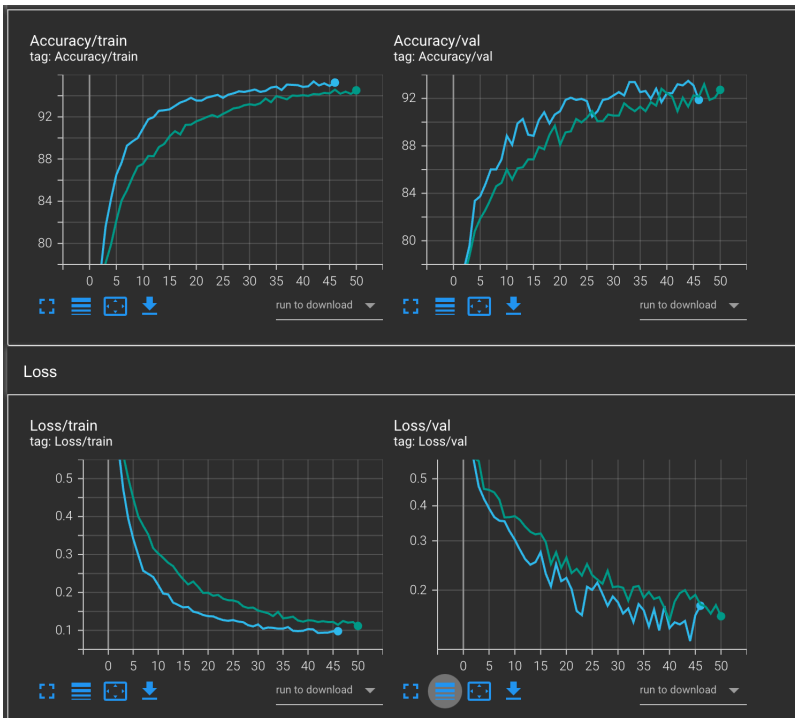
different Learning rate，如同上面所描述，可以發現前面的Convolution比較不需要更新，後面的fully connected比較需要更新，因此我嘗試對於不同區段的模型，設定不一樣的learning rate，就可以減緩前面模型的更新，而這樣的在實作上也比較容易，只要指定參數區域跟learning rate即可。

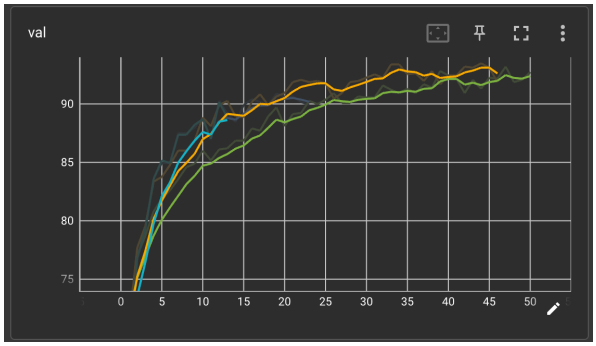
```
optimizer = optim.Adam([
                {'params': model[0].parameters(), 'lr': 1e-5},
                {'params': model[1:].parameters(), 'lr': 1e-3}
            ])
```

Vote，在這次lab中我訓練了很多不同架構的模型，雖然都可以在test上拿到還不錯的分數，但如果去比對不同模型的predict就會發現他們的結果相差甚大，況且考慮兩個相似的class，可能以51%:49%的機率選擇前者class，但是其他模型可能預測出1%:99%的結果，所以我將最後prediect testing data改成取得多個模型的predict，將他們結果內積之後找max，這樣的做法在最後的preformance有很大的進步。如果時間充裕，我會想要嘗試將prediect label當作input放入LSTM, FC, transformer中，prediect更好的label。

TensorBoard

TB是機器學習的小幫手，我有將後期trian的幾個模型的Acc/Loss記錄起來，可以發現在train上面loss接近嚴格遞減，然後ACC則是逐漸提高，但在validation則不斷上下震盪，總體來說也是逐漸提高，我認為會有這樣的現象，是因為dataset不大，因此validation會因為一兩筆資料影響整體看起的表現，基本上並不是模型本身的問題。





心得

這次的雖然實作的部分比上次簡單很多，但是在消耗時間上比前者大很多，因為training data給很少，而且也很奇異，所以我花了大量的時間在處理data的品質；之後又不斷的尋找更好的參數來提高模型的精準度，確實算是個「調參俠」。也因為模型的訓練會花時間，但又不能放著不管，所以基本上每個週末都泡在這次的lab中，甚至通宵想辦法強化，真的是可以被當成崔醫師了

如果有更多的時間，會想要試著利用DCGAN等生成模型，嘗試製造多一點的data；再透過resnet34將影像辨識做的精確一些，這次因為時間比較短，並且還不太了解大型模型的訓練方式，所以沒有花特別多的時間在嘗試這方面。