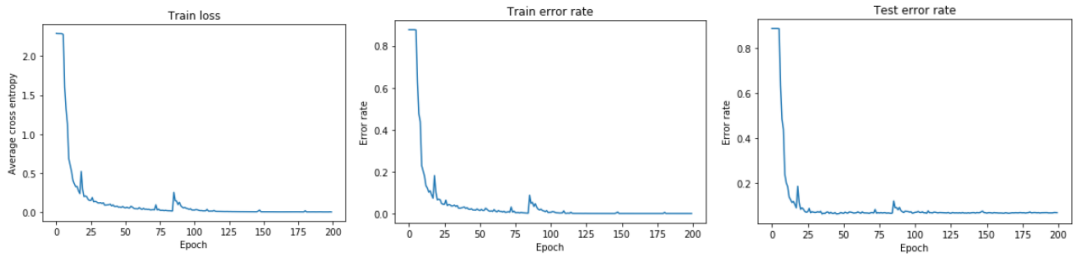
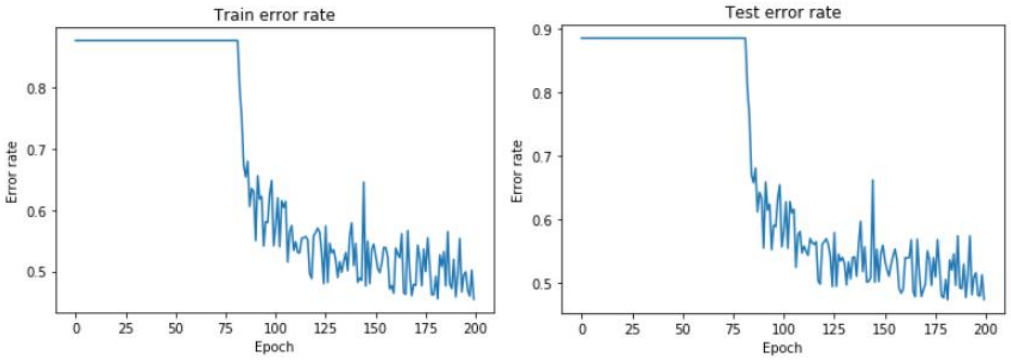
Deep Learning HW1　0752617 統計碩張家綸

1.1

在這邊設定以下hyper parameters
Number of hidden layers : 2
Number of hidden units : 15 & 10
Learning rate : 0.1
Number of iterations : 200
Mini-batch size : 120



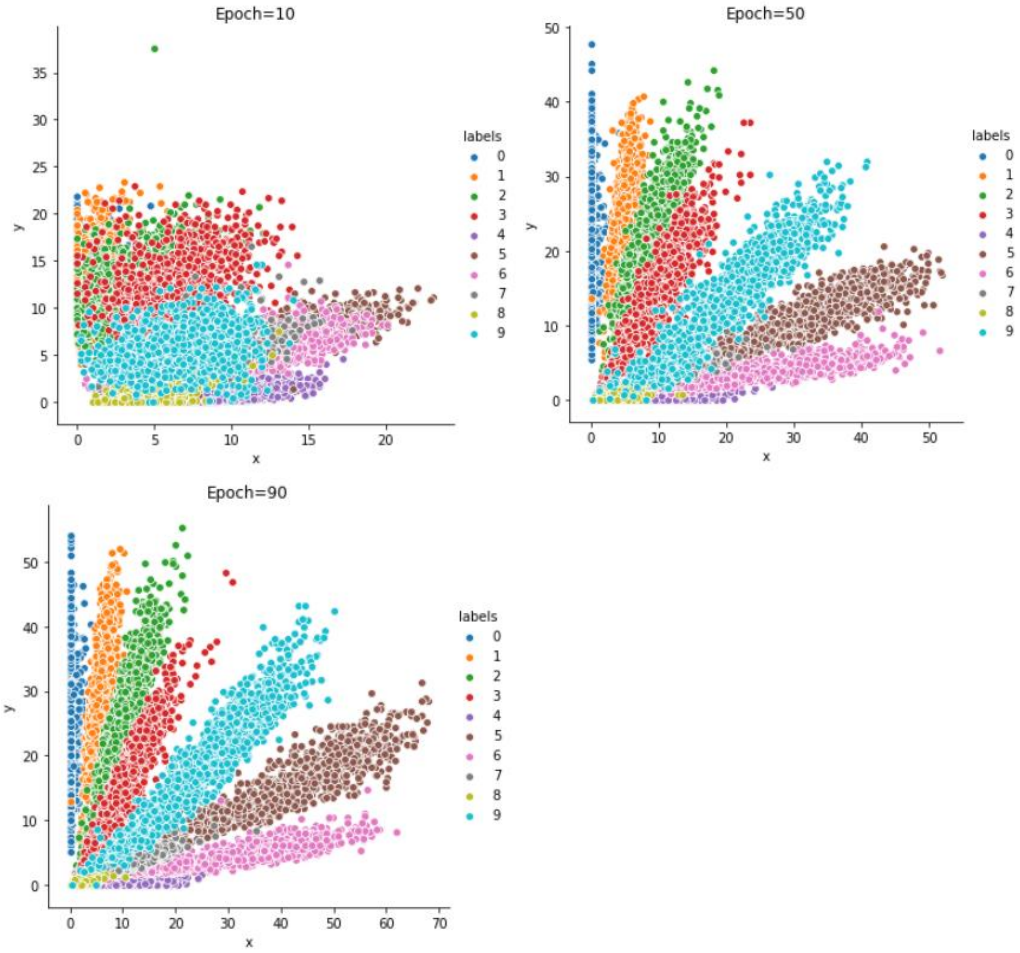
1.2



比較上下兩種圖可以發現 zero initializations 的 error rate 最好都只降到 0.55 左右，且幅度變動比較大，代表不穩定無法收斂，且下降速度慢，原因是因為 zero initializations 在 W 的更新時會有許多項為 0，導致在過程中遺失許多資訊

1.3

在這邊，將第二層 hidden layer 的 units 改為 2



從上面兩張圖可以發現，在第 10 個 epoch 其實分的有些輪廓已經出來了，除了灰色 7 的與咖啡色 5 和粉紅色 6 還有綠色 2 跟紅色 3 的比較混在一起，，到了第 50 個 epoch 各個類別分地又更明顯，且 x 軸與 y 軸的上界都在不斷擴大，在後來第 90 個 epoch 的分類狀況看起來與 50 差不多但各個類別的界線有更清楚地感覺。

1.4

	0	1	2	3	4	5	6	7	8	9
0	651	7	3	0	0	0	1	2	0	0
1	7	649	4	0	0	0	0	0	0	1
2	4	5	525	49	0	0	0	0	0	1
3	0	2	59	516	1	3	0	15	3	1
4	1	0	1	2	585	3	13	0	11	35
5	0	0	0	3	2	389	4	5	3	1
6	3	0	0	1	3	8	477	7	2	1
7	1	0	2	10	5	6	19	399	5	2
8	0	0	0	7	16	3	1	3	542	0
9	0	2	5	3	28	1	3	0	0	636

在 confusion matrix 可以發現 4、8、9 和 2、3 的相對錯誤率是最高的，表示他們圖像相似處可能非常多，導致被分類錯誤

2.1

首先進行資料讀取，由於讀進來的都是一大張圖片，但在進行分類時是對每張圖片的每個人是否有帶口罩進行，因此需將每張大圖片內的人用 csv 檔中的位置進行切割並除上 255 再將 channel 的維度拉到最前面，由於每張小照片大小都不同，對於處理不同大小的 CNN 有許多方法，如 SPP-net、resize、corp，在這邊使用 resize 的方法將每張圖片長寬都轉成 64，最後將每張小圖片用 list 存起來並轉成 4 維的 array。

但在最後要轉換 list 為 tensor 時發現有小圖片的照片維度不同，發現有些照片的 channel 是 4 的為 RGBA，因此在讀取時將所有圖片都轉成轉成 RGB 防止出現 channel 不同的情況。

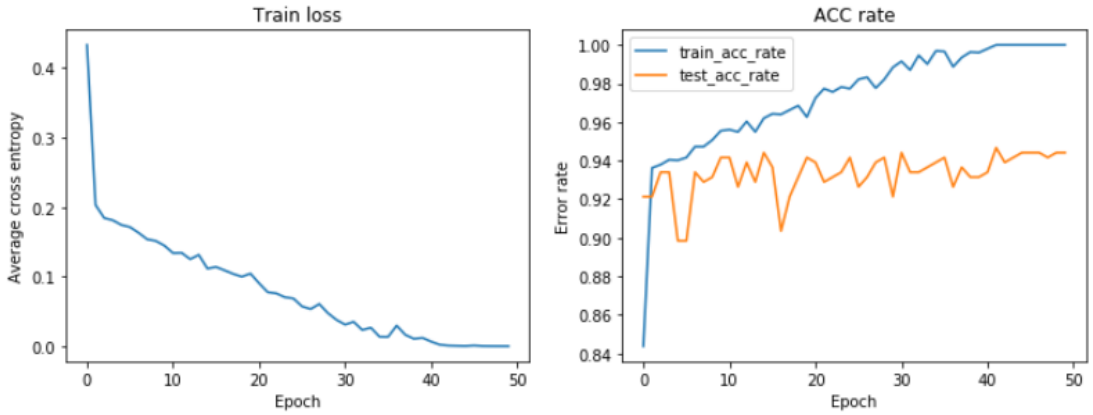
最終 train 的 shape 為 (3528, 3, 64, 64)、test 的 shape 為(394, 3, 64, 64)

2.2

再來開始建構 CNN 的 model，這邊一開始共設定 4 層卷積層、2 層 MaxPool 分別再 2 層卷積層後，最終輸出一層全連接層並且輸出設定為 3 個 labels

首先我們設定一些參數，stride 跟 padding 都為 1，epoch 跟 learning rate 和 batch size 設為 50、0.001、32，下面為 CNN 結構圖(會隨參數設定不同而有不同的係數，但整體結構不變)與 loss curve 和 train、test 的 ACC 圖表

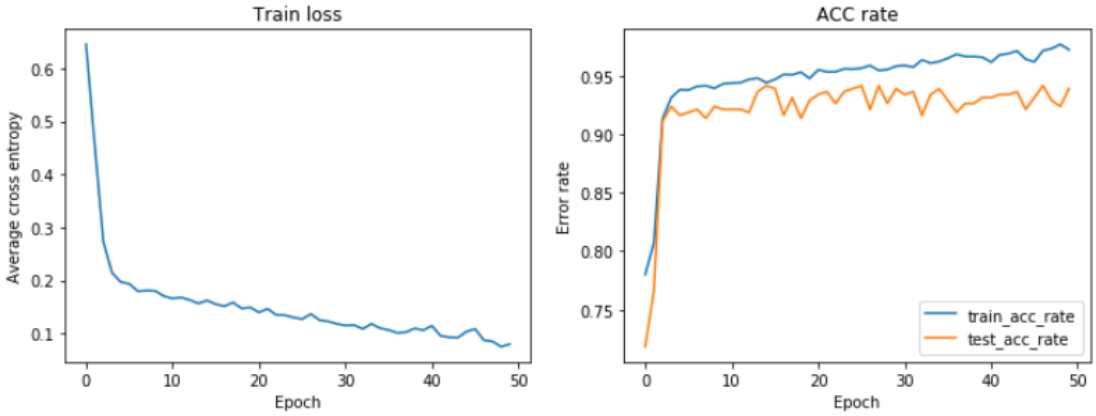
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 12, 64, 64]             336
              ReLU-2          [-1, 12, 64, 64]               0
            Conv2d-3          [-1, 12, 64, 64]           1,308
              ReLU-4          [-1, 12, 64, 64]               0
         MaxPool2d-5          [-1, 12, 32, 32]               0
            Conv2d-6          [-1, 24, 32, 32]           2,616
              ReLU-7          [-1, 24, 32, 32]               0
            Conv2d-8          [-1, 24, 32, 32]           5,208
              ReLU-9          [-1, 24, 32, 32]               0
        MaxPool2d-10          [-1, 24, 16, 16]               0
           Linear-11                   [-1, 3]          18,435
================================================================
Total params: 27,903
Trainable params: 27,903
Non-trainable params: 0
```



從上面可以發現起始 ACC 還不錯，有 0.84 以上，且在後面也有近乎收斂的狀態，整體能達到 0.94 以上效果還算不錯。

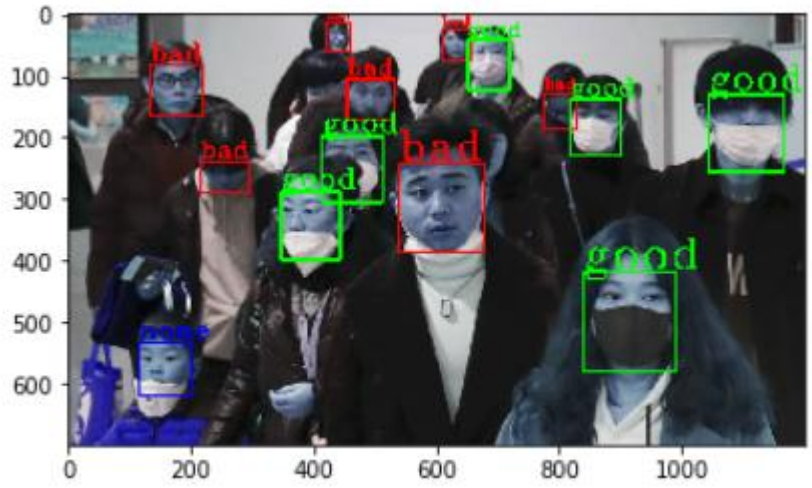
接著試著調整 batch size 和 pooling 的 kernel size 為 64 與 4，下面為跑完結果，結果比上面差蠻多的，起始只有 0.7 以上，train 和 test 在 ACC 上最好只能達到 0.93 多以上，且下降速度則與上面差不多，可能原因是因為 kernel size 拉大後，導致過程中資料遺失過多，因為最後只剩 4*4*24 相較上面 1

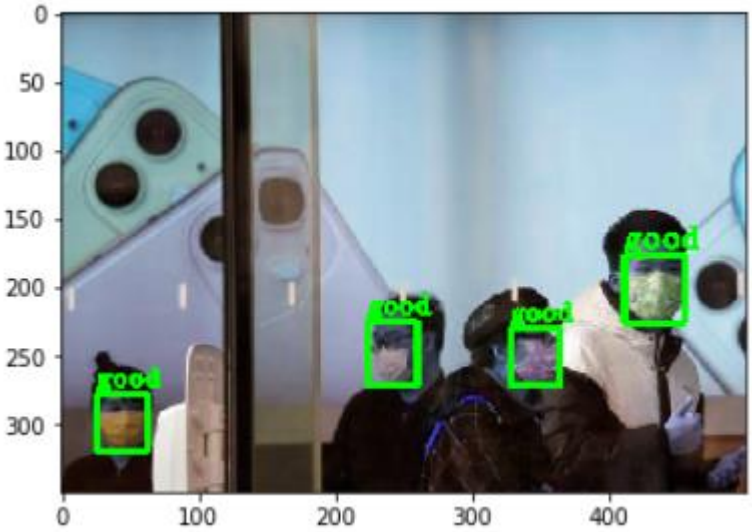
6*16*24 少了蠻多參數。



2.3

在下面我們隨便舉幾張 test 的預測照片為例，並標記人是否有帶口罩





	class	Train	Test
0	bad	1.0	0.921348
1	good	1.0	0.978799
2	None	1.0	0.454545

(i) None 表現是最不好的，在第一張照片也可以發現，真實的 None 有 3 個，但照片只預測一個，原因是從一開始的 training labels 可以看到 None 的比例是最少的，因此在訓練過程中 None 的資訊量最少導致後續的 Test None 結果不好。

(ii) 可以對於資料進行 over or under sampling 或在 loss function 中進行加權。在這邊我們使用對 loss function 進行加權的方式，從下面看起來效果好像不太好

	class	Train	Test
0	bad	0.979239	0.853933
1	good	1.000000	0.992933
2	None	0.567308	0.227273

(iii) 整體來說 model 表現還算不錯，但在 None 上面表現不算好，可是因為 None 的比例不算多，因此在整體 ACC 影響不大；但若是目標著重於 None 的話，就需要對 None 做些處理以提升它的 ACC，也可以發現 train 的過程中有點 overfitting 了，所以 epoch 可能需要減少或是增加 regularization 項。