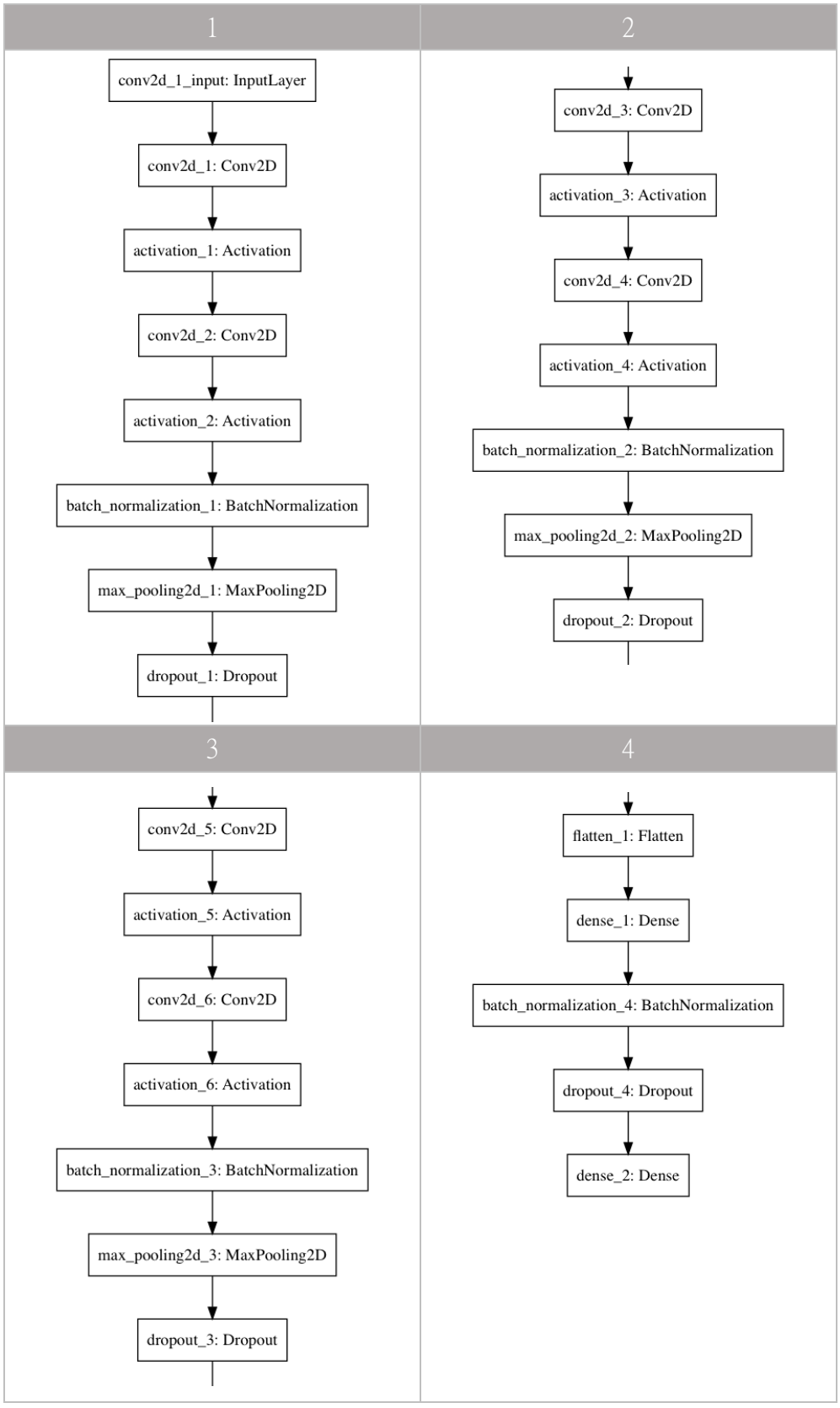
學號：B02902044　系級：資工四　姓名：陳映紅

1. **請說明你實作的 CNN model，其模型架構、訓練過程和準確率為何？**

A. 模型架構



如上表（因模型圖過長，所以已先切割），我使用「Con2D, Activation, Con2D, Activation, Batch Normalization, MaxPooling2D, Dropout」的架構，重複三次後接一層 Dense，最後輸出。

我使用 keras.callbacks 的 EarlyStopping，如下圖。

```
earlystop = EarlyStopping(monitor = 'val_loss', min_delta = 0.00001
                          , patience = 5, verbose = 1, mode = 'auto')
```

我也使用 keras.preprocessing.image 的 ImageDataGenerator，如下圖，試著將訓練資料隨機轉動、隨機水平或垂直平移、隨機翻轉。

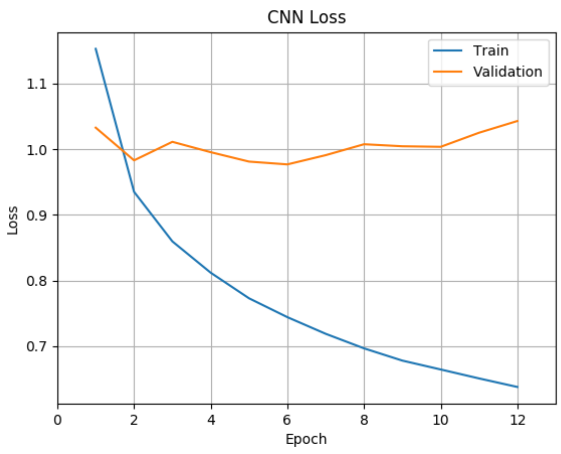
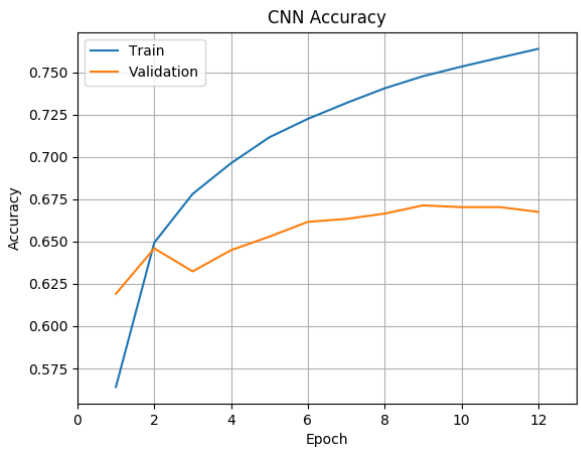
```
datagen = ImageDataGenerator(rotation_range = 10, width_shift_range = 0.1
                          , height_shift_range = 0.1, horizontal_flip = True)
```

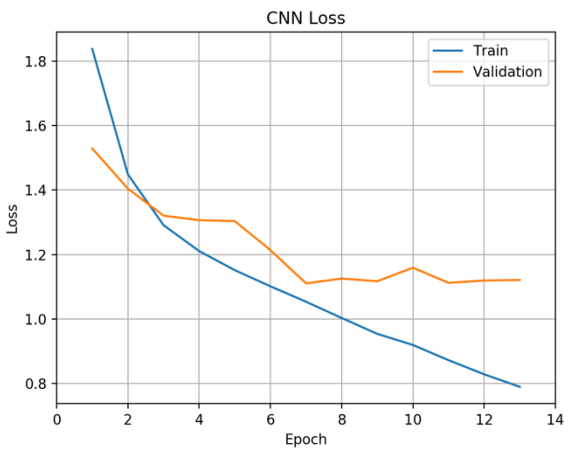
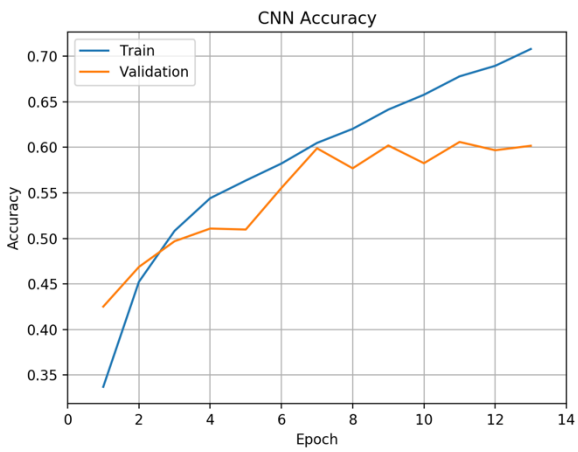
另外，我以前 10%的訓練資料作為 Validation Set。

B. 訓練過程

如下表，比較有無使用 ImageDataGenerator 的差別（CNN 架構不變）。雖然使用前的 validation loss 和 accuracy 看起來都有明顯進步，但是仍然較使用 ImageDataGenerator 後差。

另外，不管有無使用 ImageDataGenerator，執行 10～15 個 epoch 即自行停止，但使用 ImageDataGenerator 時每個 epoch 大約執行 470 秒，不使用時大約執行 40 秒。

With ImageDataGenerator	
	

Without ImageDataGenerator	
	

C. 準確率

下表為兩個版本的 CNN 於 Kaggle Public Set 上的成績，使用 ImageDataGenerator 可顯著提升模型表現。

With ImageDataGenerator	Without ImageDataGenerator
0.69351	0.61577

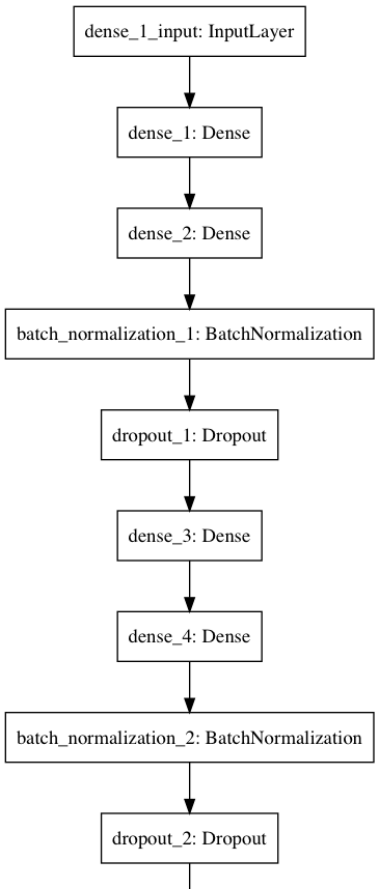
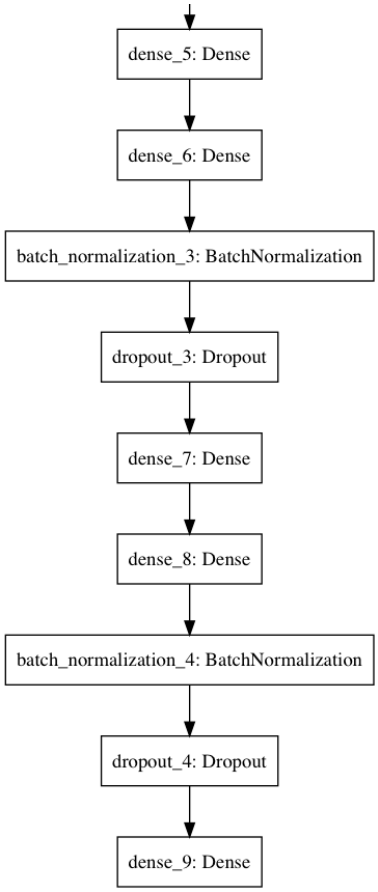
2. 承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model。其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？

A. 參數數量

CNN	DNN
Total params: 1,268,135 Trainable params: 1,266,727 Non-trainable params: 1,408	Total params: 1,268,450 Trainable params: 1,267,876 Non-trainable params: 574

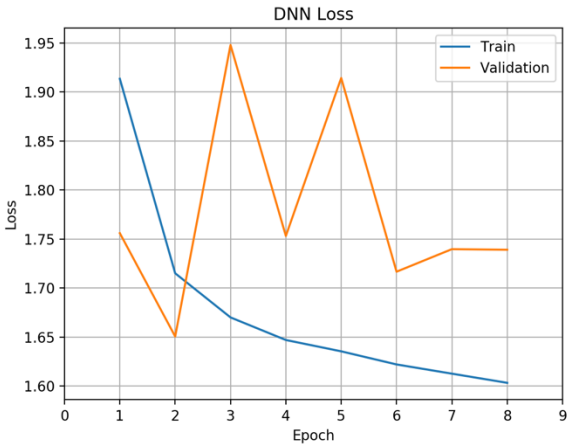
B. 模型架構

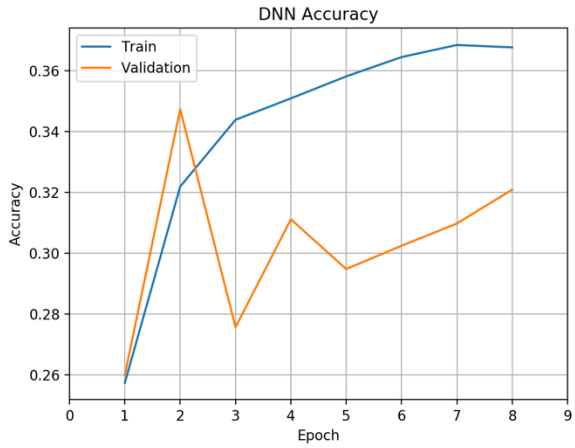
如下表（因模型圖過長，所以已先切割），我使用「Dense, Dense, Batch Normalization, Dropout」的架構，重複四次後輸出。和上一題的 CNN 架構比起來，我的 DNN 比較寬，深度較淺。

1	2
	

C. 訓練過程

EarlyStopping 和取 Validation Set 的方式不變，每個 epoch 大約執行 10 秒，執行 25 個 epoch 後自行停止。與 CNN 相比，DNN 在第一個 epoch 的表現就比較差，且訓練時的 loss 和 accuracy 的進步速度較慢，即使訓練較多個 epoch，最後的表現也較差。



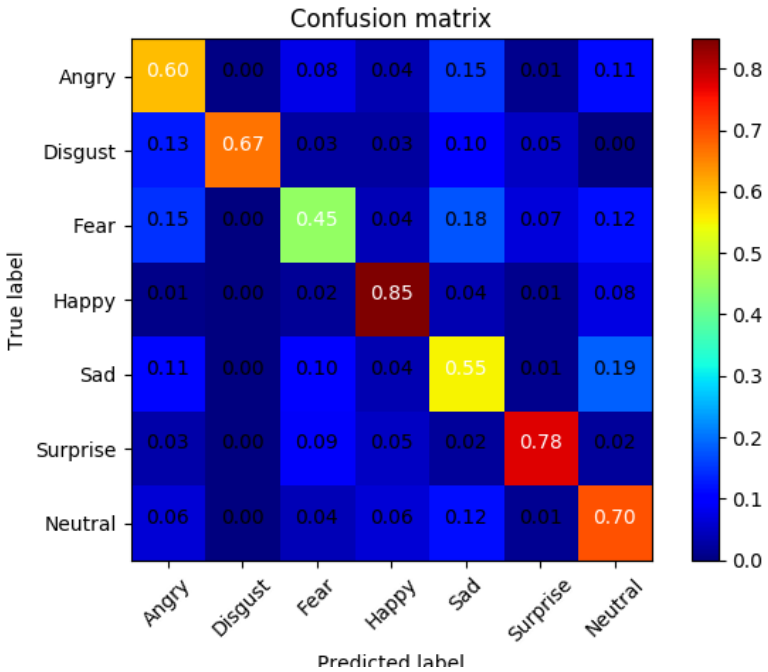


D. 準確率

Kaggle Public Score：0.32488

3. **觀察答錯的圖片中，哪些 class 彼此間容易用混？**

A. Confusion Matrix



B. 容易混淆的 Class（≥ 0.15）

i. Sad 誤認為 Neutral

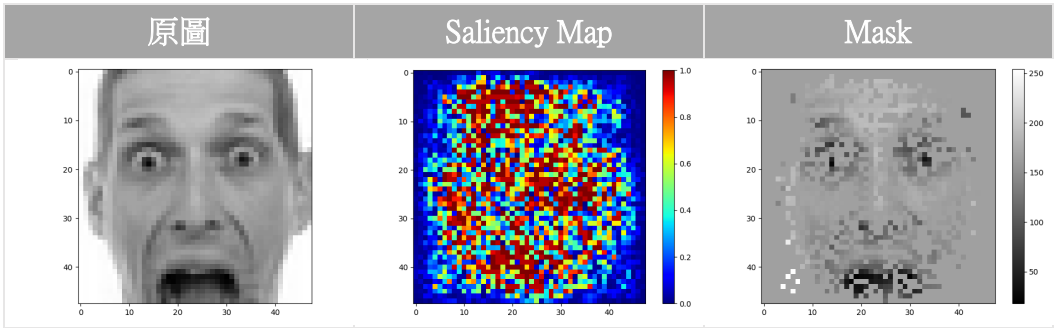
ii. Fear 誤認為 Sad

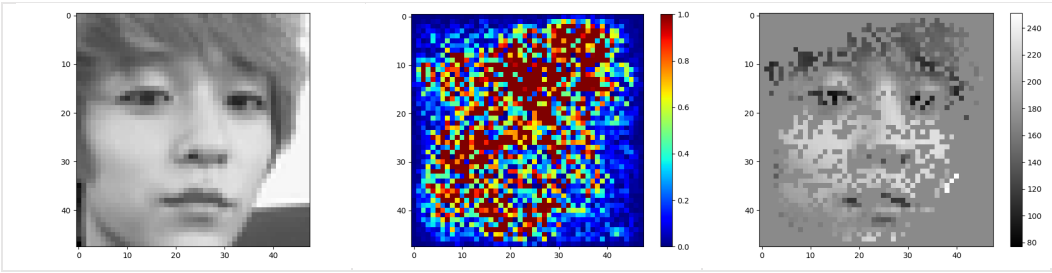
iii. Angry 誤認為 Sad

iv. Fear 誤認為 Angry

觀察可發現，容易混淆的 Class 大致偏向負面情緒，我推測可能是嘴型張大或下垂會讓模型混淆。

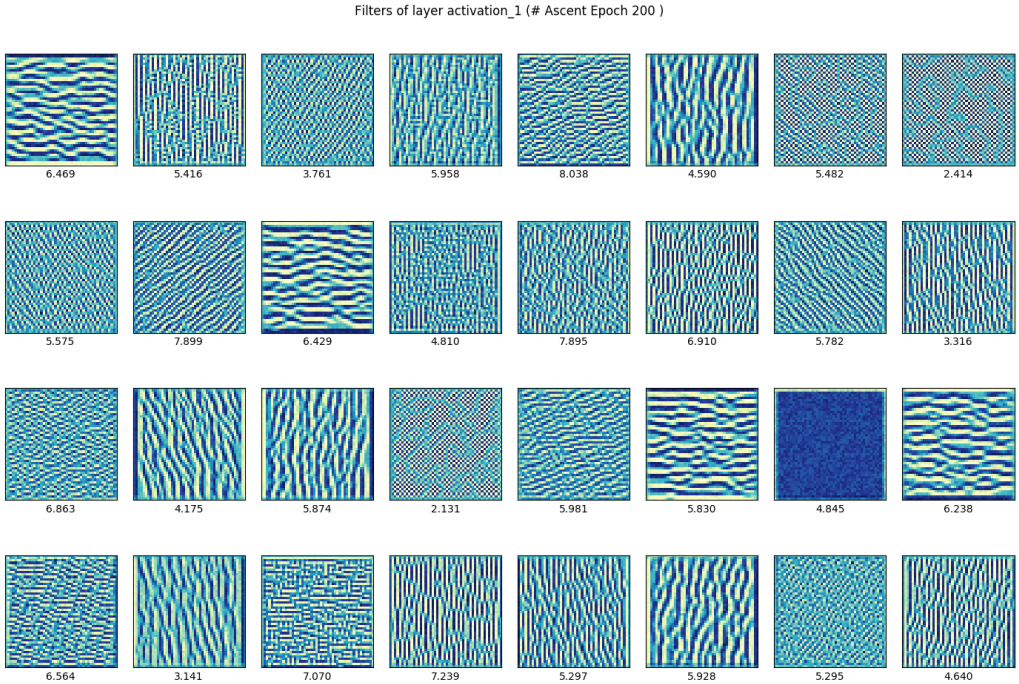
4. **從(1)(2)可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份？**

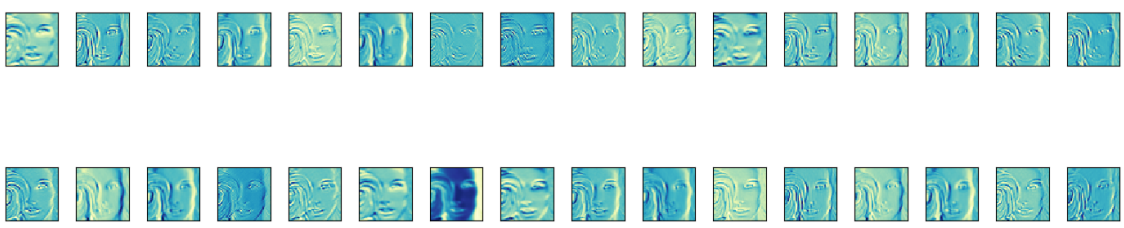




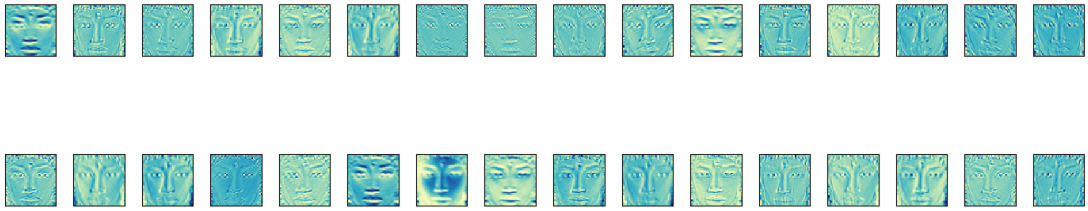
分類時幾乎都針對五官（尤其眼睛和嘴巴），臉頰也會影響分類結果。

5. 承(1)(2)，利用上課所提到的 gradient ascent 方法，觀察特定層的 filter 最容易被哪種圖片 activate。





Output of layer0 (Given image2044)



由第一張圖可發現，可讓激活各 filter 的圖大不相同，有的是明顯的長條紋，有些則是點狀、網狀紋路。

對照第一張與第二張圖可發現，經過某些 filter 後幾乎認不出原圖，而經橫/直條紋後的圖形較明顯，我認為是因為原本的 input 所包含的橫/直條紋較多，因此經過 filter 後的結果也較明顯。

再比對第二及第三張圖，image2044 原圖的橫線部分較 image320 多，而 image320 的直條紋路又較 image2044 多，因此兩張圖對 filter 的反應也不同：image2044 對橫條紋 filter 的反應明顯，image320 則對直條紋 filter 反應明顯。

註：其他 python 檔案的執行方式

- dnn.py、plot_filter.py、plot_layer_output.py、plot_saliency.py：
  直接執行 python3
- plot_struct_confusion.py：
  python3 plot_struct_confusion.py [model_structure.png] [confusion_matrix.png]
- plot_chart.py：
  python3 plot_chart.py [training_history.csv] [chart_title]
  training_history.csv 是我在訓練時自行產生的檔案