學號：R05943139 系級：電子所碩一 姓名：張育瑄

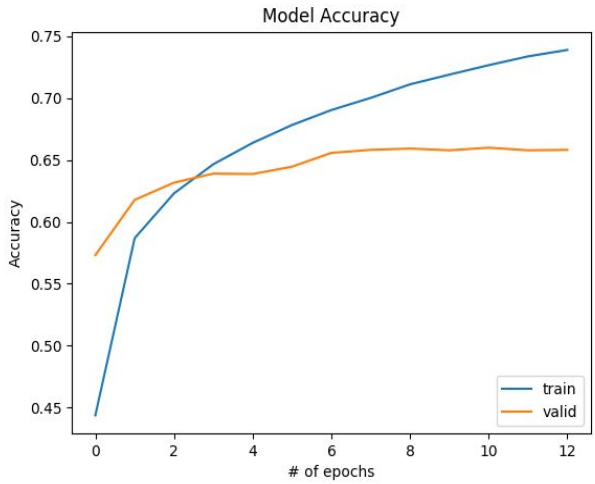
1. (1%) 請說明你實作的 CNN model，其模型架構、訓練過程和準確率為何？
   答：

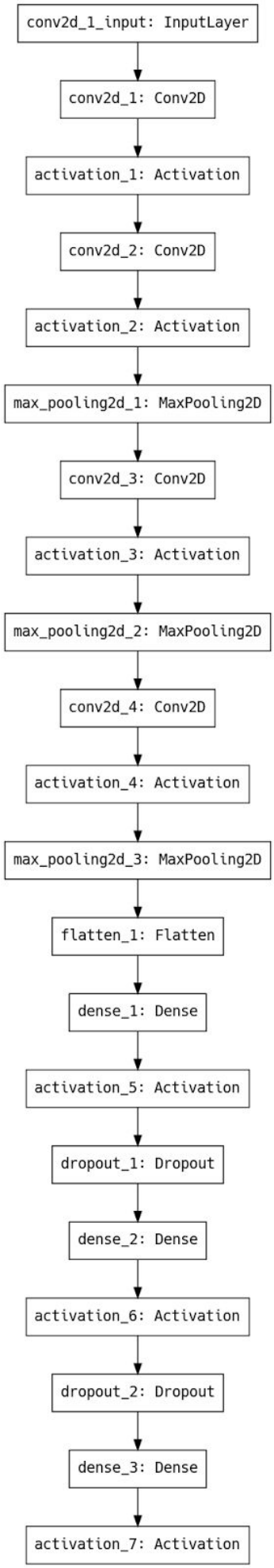
Conv2D(32, (3, 3))
Conv2D(64, (3, 3))
MaxPooling2D((2, 2))
Conv2D(128, (3, 3))
MaxPooling2D((2, 2))
Conv2D(256, (3, 3))
MaxPooling2D((2, 2))
Flatten()
Dense(1024)
Dropout(0.5)
Dense(1024)
Dropout(0.5)

基本上是 trial and error 得出這樣的堆法。
對所有 input images 做 normalization，但不做任何 preprocessing，最後加入 dropout 避免 overfitting，可以過 simple baseline。
但是要過 strong baseline 的話必須使用 Keras 的 ImageDataGenerator 做 preprocessing，包含 rotation, shift, horizontal flip, 以及 shear。
做這些 preprocessing 的原因很直覺，以下列出幾點：

1. 每張人臉的角度不同 → rotation
2. 每張人臉的拍攝角度不同 → shear
3. 每張人臉的位置不同 → shift
4. 側臉 → horizontal flip

切出 10% training data 作為 validation set，batch size 設 256，epoch 設 50，並設定 early stopping，最後在 Kaggle 上的成績為 66.4%。




conv2d_1_input: InputLayer
conv2d_1: Conv2D
activation_1: Activation
conv2d_2: Conv2D
activation_2: Activation
max_pooling2d_1: MaxPooling2D
conv2d_3: Conv2D
activation_3: Activation
max_pooling2d_2: MaxPooling2D
conv2d_4: Conv2D
activation_4: Activation
max_pooling2d_3: MaxPooling2D
flatten_1: Flatten
dense_1: Dense
activation_5: Activation
dropout_1: Dropout
dense_2: Dense
activation_6: Activation
dropout_2: Dropout
dense_3: Dense
activation_7: Activation

2. (1%) 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？
答：

Layer (type)	Output Shape	Param #
max_pooling2d_1 (MaxPooling2	(None, 22, 22, 64)	0
conv2d_3 (Conv2D)	(None, 20, 20, 128)	73856
activation_3 (Activation)	(None, 20, 20, 128)	0
max_pooling2d_2 (MaxPooling2	(None, 10, 10, 128)	0
conv2d_4 (Conv2D)	(None, 8, 8, 256)	295168
activation_4 (Activation)	(None, 8, 8, 256)	0
max_pooling2d_3 (MaxPooling2	(None, 4, 4, 256)	0
flatten_1 (Flatten)	(None, 4096)	0
dense_1 (Dense)	(None, 1024)	4195328
activation_5 (Activation)	(None, 1024)	0
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1024)	1049600
activation_6 (Activation)	(None, 1024)	0
dropout_2 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 7)	7175
activation_7 (Activation)	(None, 7)	0

Total params: 5,639,943.0
Trainable params: 5,639,943.0
Non-trainable params: 0.0

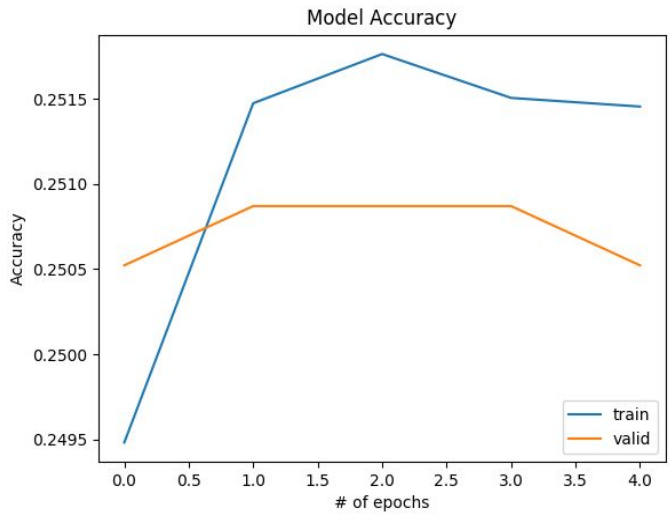
Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 1024)	2360320
activation_1 (Activation)	(None, 1024)	0
dropout_1 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1024)	1049600
activation_2 (Activation)	(None, 1024)	0
dropout_2 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 1024)	1049600
activation_3 (Activation)	(None, 1024)	0
dropout_3 (Dropout)	(None, 1024)	0
dense_4 (Dense)	(None, 1024)	1049600
activation_4 (Activation)	(None, 1024)	0
dropout_4 (Dropout)	(None, 1024)	0
dense_5 (Dense)	(None, 7)	7175
activation_5 (Activation)	(None, 7)	0

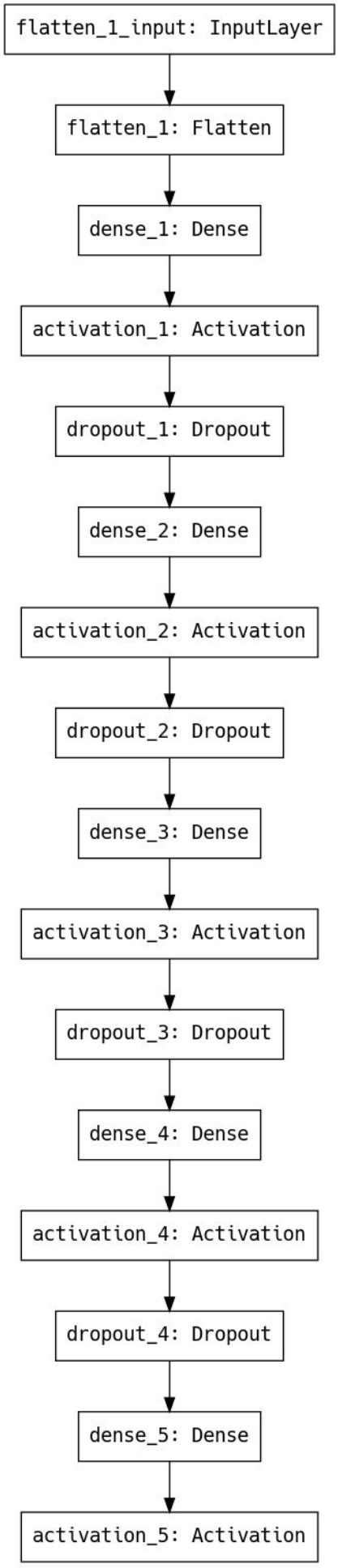
Total params: 5,516,295.0
Trainable params: 5,516,295.0
Non-trainable params: 0.0

左圖為 CNN 參數量，右圖為 DNN 參數量。

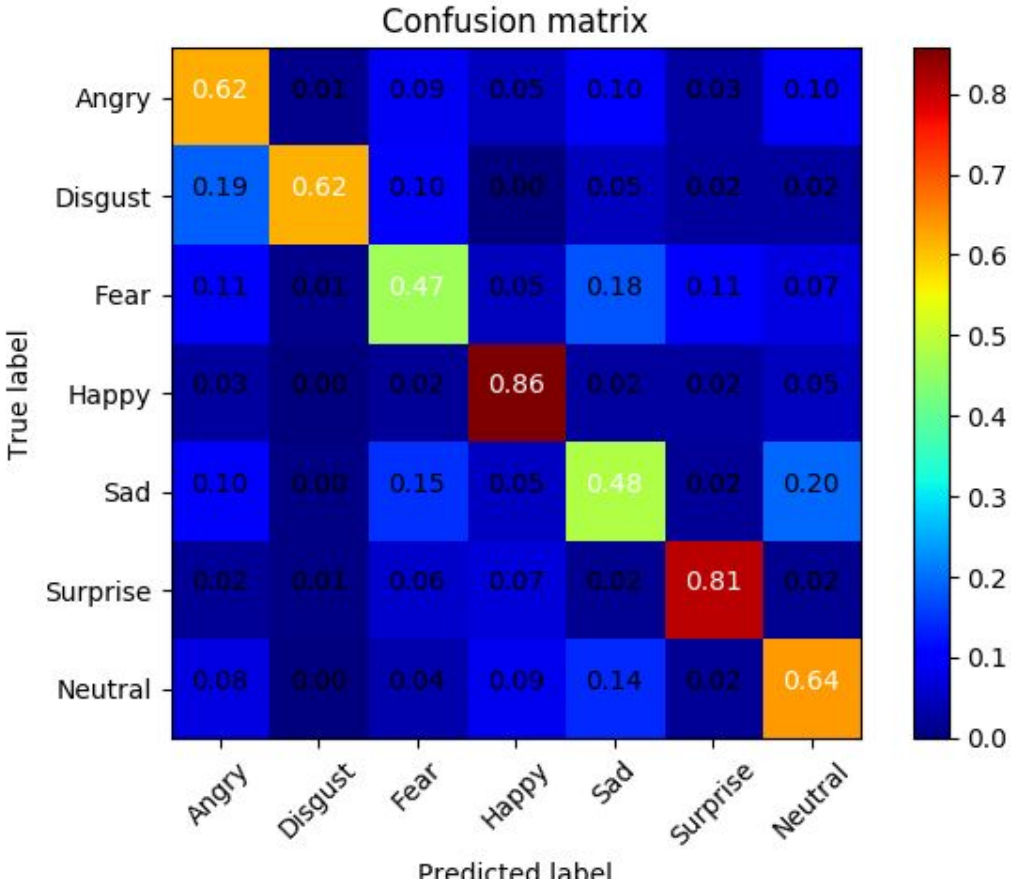
用跟 CNN 一模一樣的 training configuration， 結果為 25%。

觀察 confusion matrix 後發現它把幾乎所有照片都以為是 happy。




flatten_1_input: InputLayer
flatten_1: Flatten
dense_1: Dense
activation_1: Activation
dropout_1: Dropout
dense_2: Dense
activation_2: Activation
dropout_2: Dropout
dense_3: Dense
activation_3: Activation
dropout_3: Dropout
dense_4: Dense
activation_4: Activation
dropout_4: Dropout
dense_5: Dense
activation_5: Activation

3. (1%) 觀察答錯的圖片中，哪些 class 彼此間容易用混？[繪出 confusion matrix 分析]
   答：

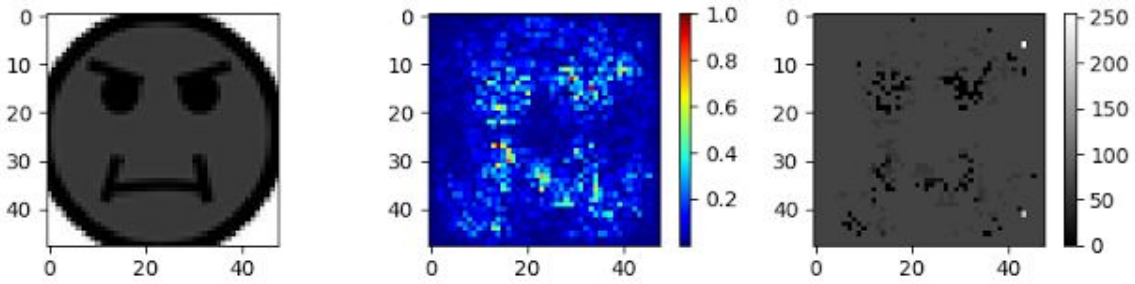


從 confusion matrix 可觀察到以下幾點：

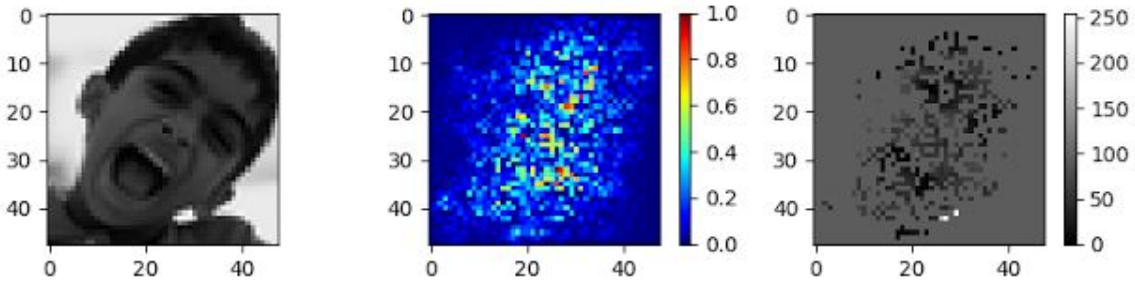
1. happy 非常好認：可能是因為 training data 中 happy 的圖片較多。
2. fear 跟 sad 較容易互相混淆：很直覺，因為連我自己肉眼看都不是很分的出來。
3. disgust 容易被認為是 angry，但幾乎沒有其他 sentiment 會被認為是 disgust：發現可能是因為 training data 中 disgust 的圖片本身就比較少，因此比較沒辦法認出 disgust。
4. 所有 sentiment 都容易與 neutral 混淆，除了 disgust 與 surprise：這個也蠻直覺的，因為除了 surprise 以外的 sentiment 都可能不是很誇張的表情，因此較容易被誤認為 neutral。

4. (1%) 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？
   答：

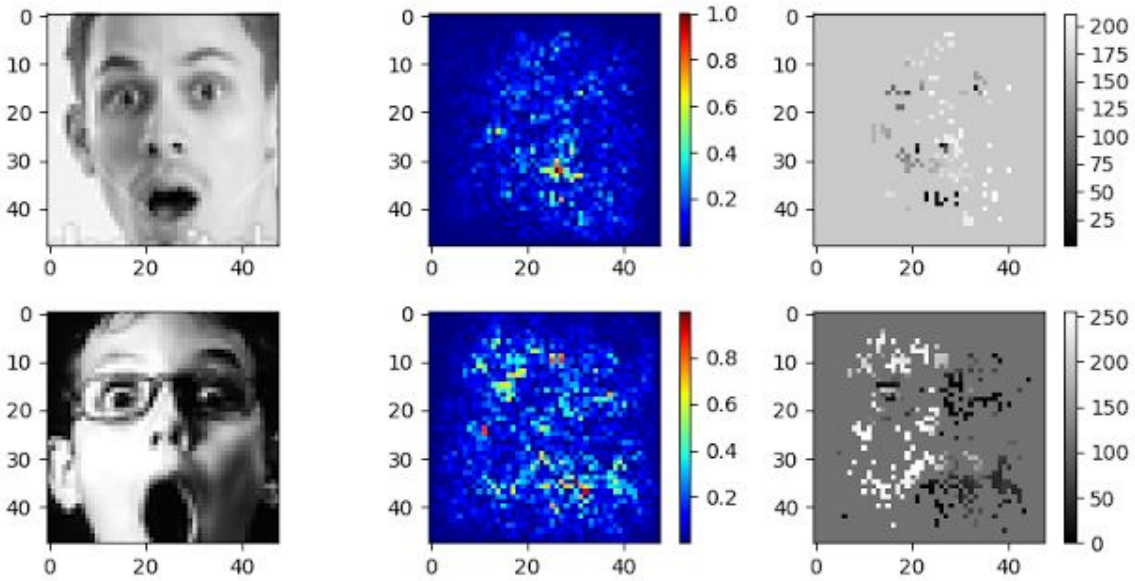
以下面這張 angry 的圖很明顯就可以看出他是 focus 在眼睛、眉毛以及嘴巴。



以下面這張 happy 的圖也可以看出他主要是 focus 在眼睛以及嘴巴。

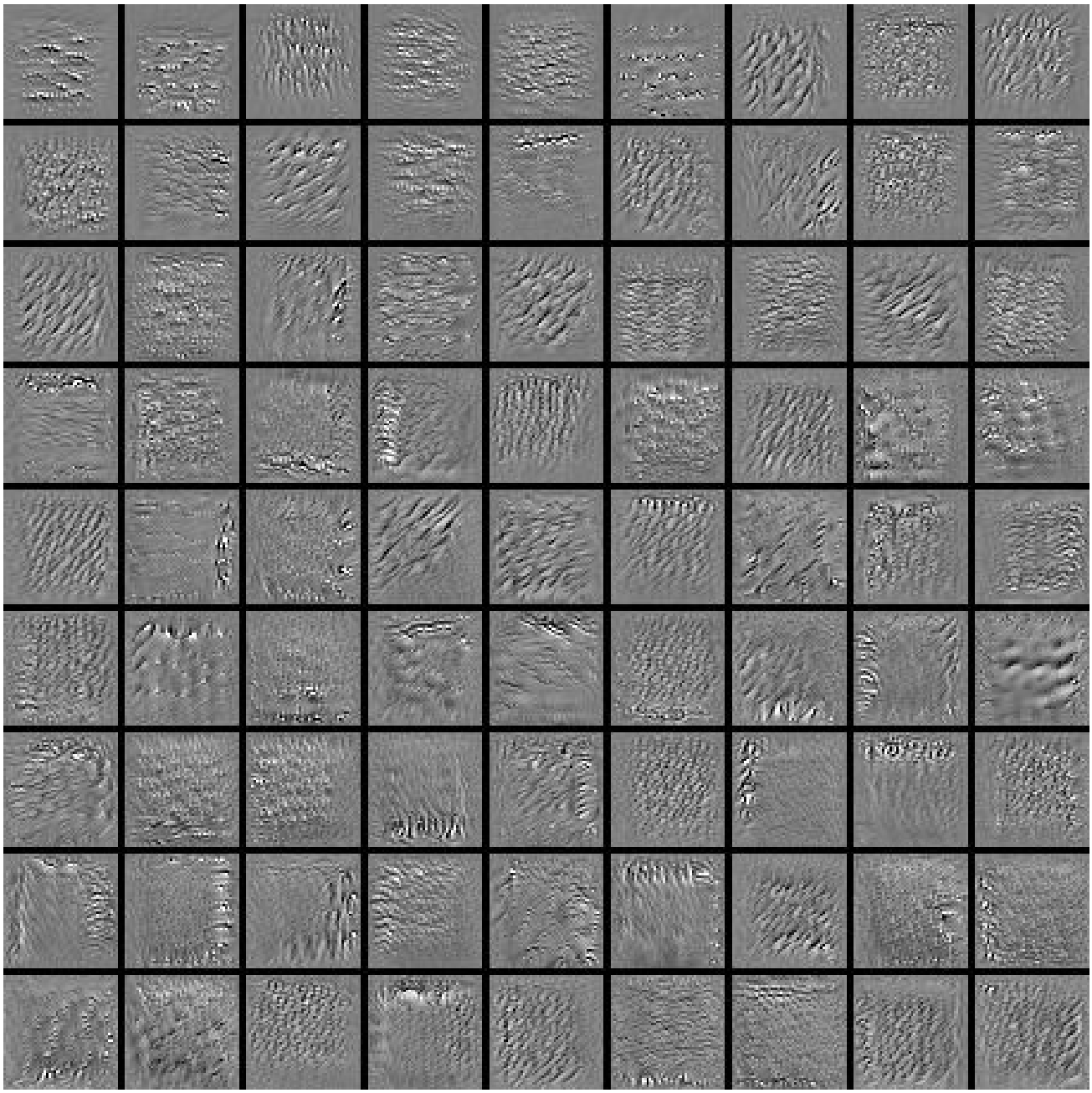


以下面這兩張 surprise 的圖也可以看出他主要是 focus 在眼睛以及嘴巴。



5. (1%) 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。
   答：

以下為最後一層 convolution layer 的 filters 反應最興奮的圖。

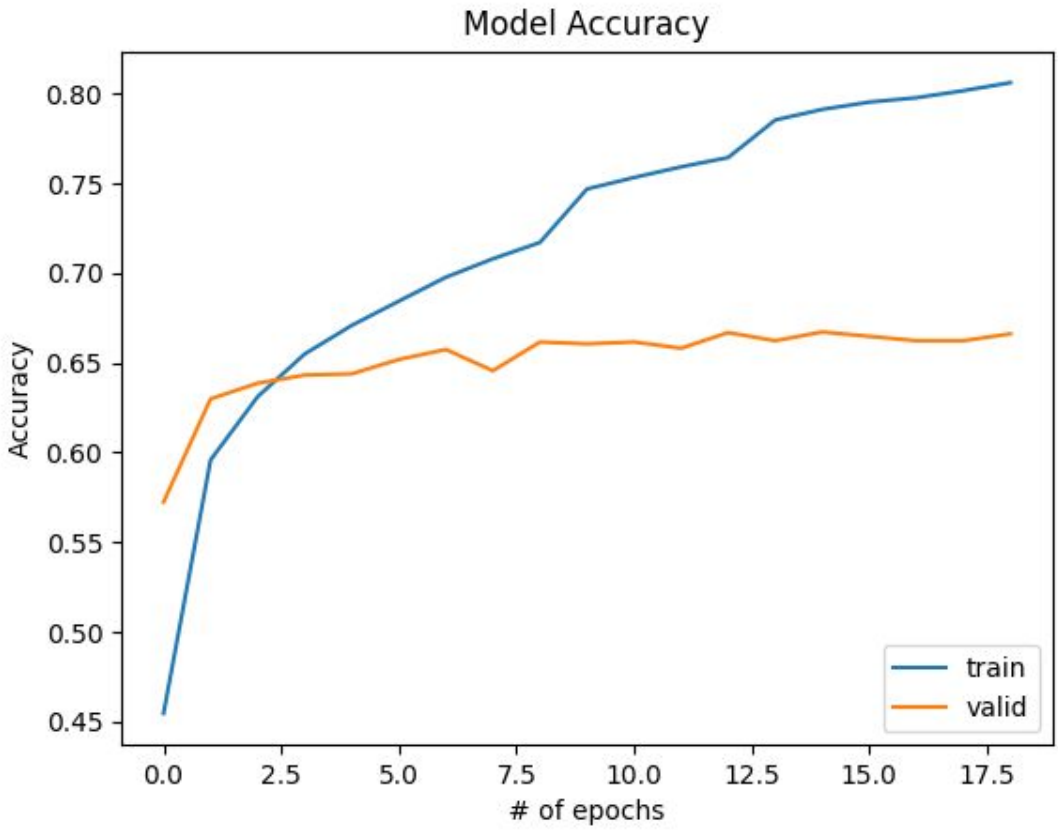


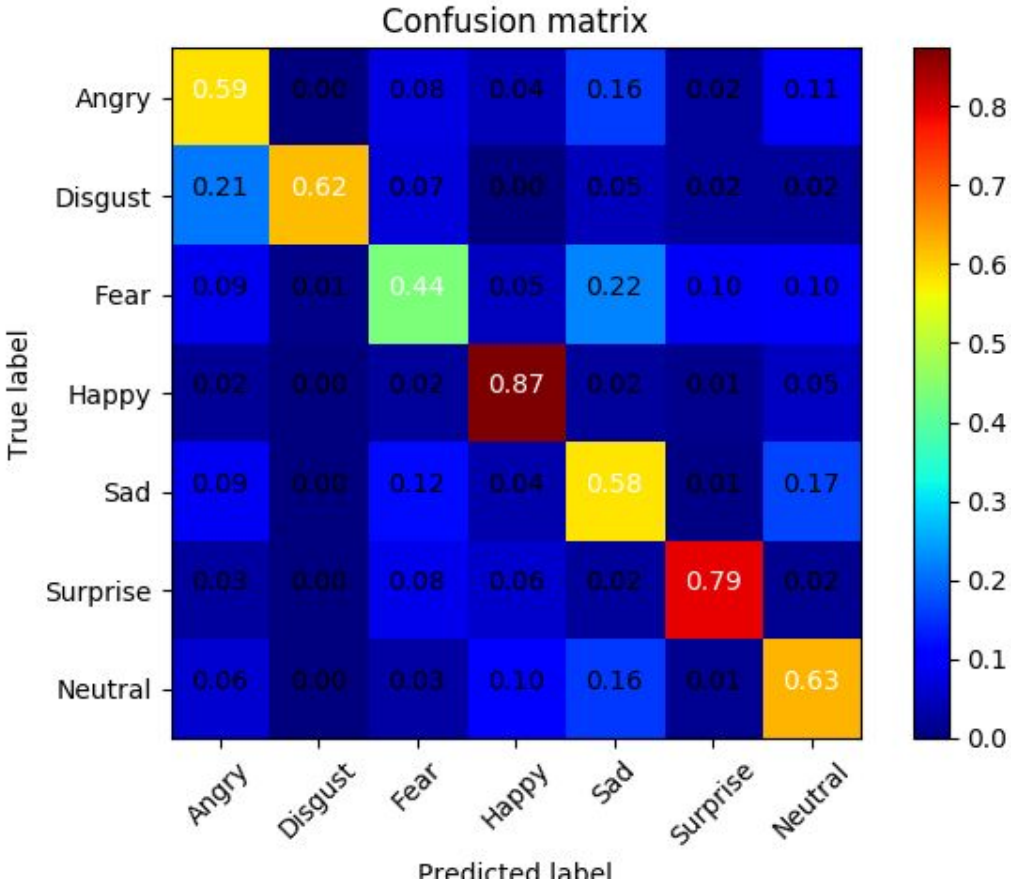
[Bonus] (1%) 從 training data 中移除部份 label，實做 semi-supervised learning

使用前面的 CNN 架構先 train 一次所有的 training data，然後 predict testing data，並將 predict 出來較有信心的 data 加入 labeled data，再繼續 train，直到超過七成的 testing data 被 label 為止，最後 Kaggle 成績為 67.7%。

而所謂有信心指的是 predict 出來某一個 class 的值大於 0.8 我才將它加入 labeled data。

下圖為 training 的過程，可以發現 validation set 的 accuracy 一直在震盪，我推測是由於新加入的 label 不一定是對的，有可能造成原本就辨識正確的會越 train 越好，但錯的就會越 train 越錯，這一來一往，accuracy 就無法穩定上升了。





[Bonus] (1%) 在Problem 5 中，提供了3個 hint，可以嘗試實作及觀察 (但也可以不限於 hint 所提到的方向，也可以自己去研究更多關於 CNN 細節的資料)，並說明你做了些什麼？ [完成1個: +0.4%, 完成2個: +0.7%, 完成3個: +1%]