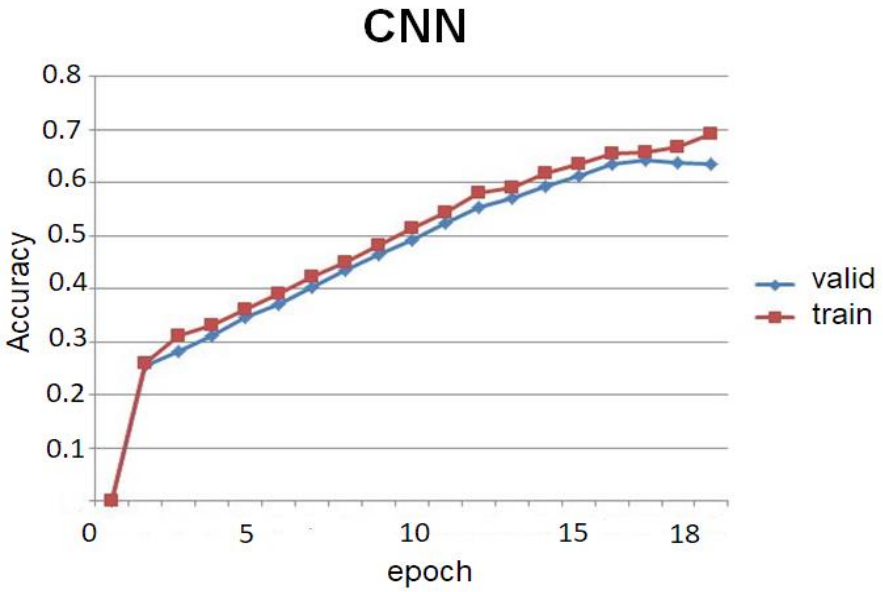
學號：R05943110 系級： 電子碩二 姓名：蕭 堯

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？ (Collaborators: None)

我把 train_data 做了左右翻轉和部分放大特寫的處理，使我的 data 變為原有的 4 倍，也有效增加了 CNN 泛化的能力，準確率在訓練 18 epochs 後再 kaggle 分數達 6.24 左右，batch 取 400。

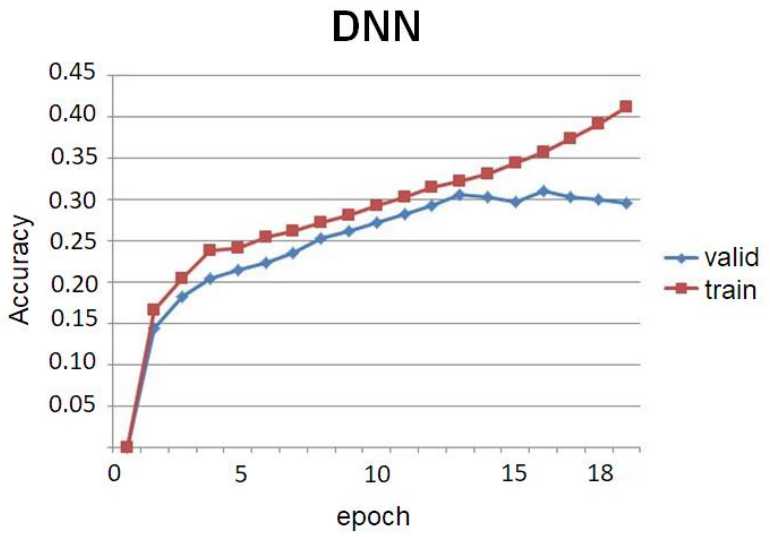


Layer (type)	Output Shape	Param #
block1_conv1 (Conv2D)	(None, 48, 48, 64)	1792
leaky_re_lu_1 (LeakyReLU)	(None, 48, 48, 64)	0
block1_conv2 (Conv2D)	(None, 48, 48, 64)	36928
leaky_re_lu_2 (LeakyReLU)	(None, 48, 48, 64)	0
block1_pool (MaxPooling2D)	(None, 24, 24, 64)	0
dropout_1 (Dropout)	(None, 24, 24, 64)	0
block2_conv1 (Conv2D)	(None, 24, 24, 128)	73856
leaky_re_lu_3 (LeakyReLU)	(None, 24, 24, 128)	0
block2_conv2 (Conv2D)	(None, 24, 24, 128)	147584
leaky_re_lu_4 (LeakyReLU)	(None, 24, 24, 128)	0
block2_pool (MaxPooling2D)	(None, 12, 12, 128)	0
dropout_2 (Dropout)	(None, 12, 12, 128)	0
block3_conv1 (Conv2D)	(None, 12, 12, 256)	295168
leaky_re_lu_5 (LeakyReLU)	(None, 12, 12, 256)	0
block3_conv2 (Conv2D)	(None, 12, 12, 256)	590080
leaky_re_lu_6 (LeakyReLU)	(None, 12, 12, 256)	0
block3_pool (MaxPooling2D)	(None, 6, 6, 256)	0
dropout_3 (Dropout)	(None, 6, 6, 256)	0
flatten_1 (Flatten)	(None, 9216)	0
fc (Dense)	(None, 2048)	18876416
leaky_re_lu_7 (LeakyReLU)	(None, 2048)	0
dropout_4 (Dropout)	(None, 2048)	0
predictions (Dense)	(None, 7)	14343

Total params: 20,036,167
Trainable params: 20,036,167
Non-trainable params: 0

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？
(Collaborators:None)

DNN 的表現明顯低了很多，雖然參數數量和 CNN 接近，訓練方式也類似的情況下，準確率到了 0.315 後就不太能 train 下去了。



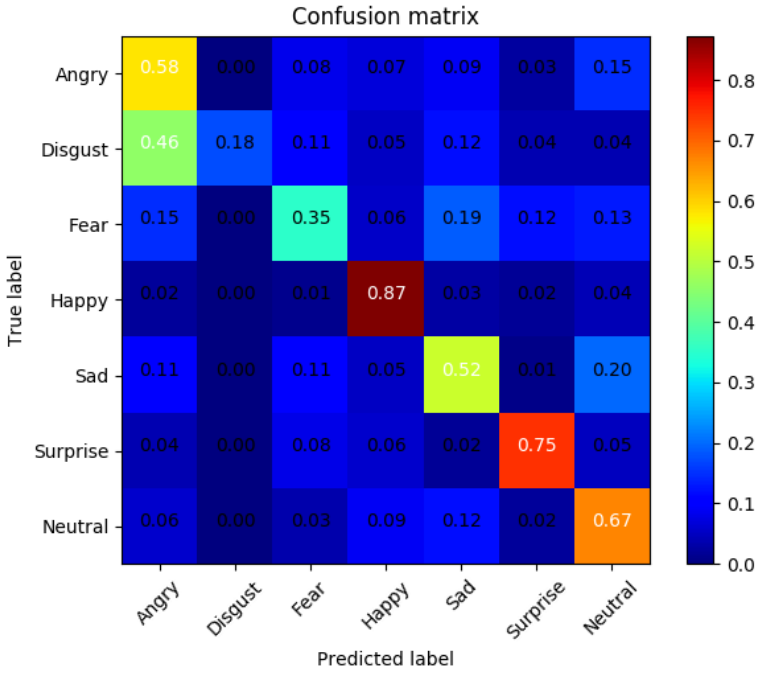
Layer (type)	Output Shape	Param #
L1 (Dense)	(None, 48, 48, 1024)	4096
L2 (Dense)	(None, 48, 48, 1024)	1049600
L3 (Dense)	(None, 48, 48, 1024)	1049600
L4 (Dense)	(None, 48, 48, 1024)	1049600
L5 (Dense)	(None, 48, 48, 1024)	1049600
L6 (Dense)	(None, 48, 48, 1024)	1049600
L7 (Dense)	(None, 48, 48, 1024)	1049600
L8 (Dense)	(None, 48, 48, 1024)	1049600
L9 (Dense)	(None, 48, 48, 1024)	1049600
L10 (Dense)	(None, 48, 48, 1024)	1049600
L11 (Dense)	(None, 48, 48, 1024)	1049600
L12 (Dense)	(None, 48, 48, 1024)	1049600
L13 (Dense)	(None, 48, 48, 1024)	1049600
L14 (Dense)	(None, 48, 48, 1024)	1049600
L15 (Dense)	(None, 48, 48, 1024)	1049600
L16 (Dense)	(None, 48, 48, 1024)	1049600
dropout_1 (Dropout)	(None, 48, 48, 1024)	0
L17 (Dense)	(None, 48, 48, 1024)	1049600
L18 (Dense)	(None, 48, 48, 1024)	1049600
dropout_2 (Dropout)	(None, 48, 48, 1024)	0
L19 (Dense)	(None, 48, 48, 1024)	1049600
dropout_3 (Dropout)	(None, 48, 48, 1024)	0
L20 (Dense)	(None, 48, 48, 1024)	1049600
dropout_4 (Dropout)	(None, 48, 48, 1024)	0
predictions (Dense)	(None, 48, 48, 7)	7175

Total params: 19,953,671
Trainable params: 19,953,671
Non-trainable params: 0

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]
(Collaborators: None)

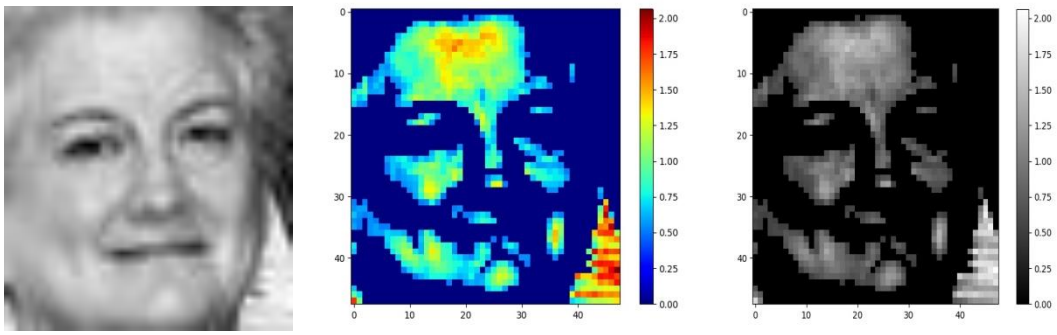
我大 confusion matrix 在對角線明顯特別亮，代表大多數的 prediction 都很正確，值得注意的是，Disgust 的 label 在我的 model 反而有很高的機率誤判為 Angry，

算是我 model 在判斷上的弱項。

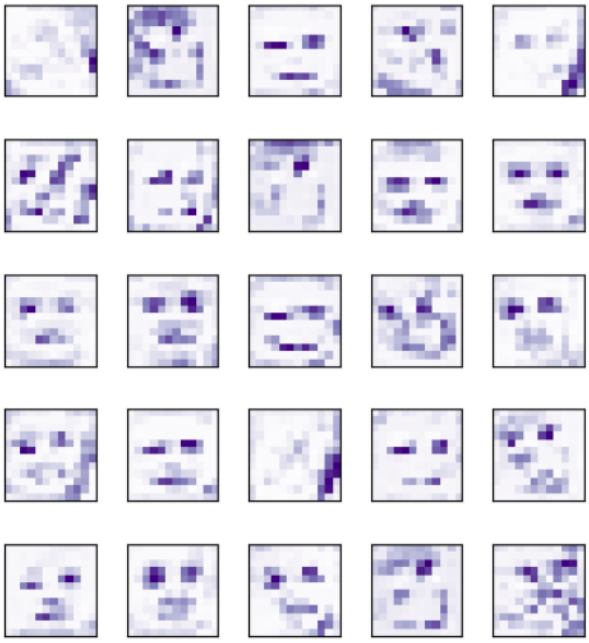


4. (1%) 從(1)(2)可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份？
(Collaborators: None)

左圖為原圖，label 為 Happy。比較讓我意外的是人的五官(如:眼睛、鼻、嘴巴)反而是 heat 比較少的地方，反而是五官的交界面和皮膚本身的 heat 都偏高，也就是說對我的 model 而言，五官交界面的變化也許才是區別表情的重要因素。



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。
(Collaborators: None)
<Mode 1>
我用 block2_pool 這一層的 filters 來做這一題，使用的照片與第四題是同一張，label 為 Happy。看的出來這一層的每個 feature map 都還留有人臉的圖案。



<Mode 2>

我用 block3_pool 這一層的 filters 來做這一題，在 256 個 filters 中，我取最 activate 的前 36 張作圖，看的出來這層的 filters 學到的不只是簡單的直線，已經有一些曲線和複雜的圖形了，只人好像沒有看到人臉的圖形。