

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：我的 CNN 架構如下：

conv2d_1 (Conv2D)	(None, 46, 46, 32)	320
activation_1 (Activation)	(None, 46, 46, 32)	0
batch_normalization_1 (Batch Normalization)	(None, 46, 46, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 23, 23, 32)	0
conv2d_2 (Conv2D)	(None, 21, 21, 64)	18496
batch_normalization_2 (Batch Normalization)	(None, 21, 21, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 10, 10, 64)	0
conv2d_3 (Conv2D)	(None, 8, 8, 128)	73856
batch_normalization_3 (Batch Normalization)	(None, 8, 8, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dropout_1 (Dropout)	(None, 2048)	0
dense_1 (Dense)	(None, 512)	1049088
activation_2 (Activation)	(None, 512)	0
batch_normalization_4 (Batch Normalization)	(None, 512)	2048
dropout_2 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 1024)	525312
activation_3 (Activation)	(None, 1024)	0
batch_normalization_5 (Batch Normalization)	(None, 1024)	4096
dropout_3 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 7)	7175
activation_4 (Activation)	(None, 7)	0
Total params: 1,681,287		
Trainable params: 1,677,767		
Non-trainable params: 3,520		

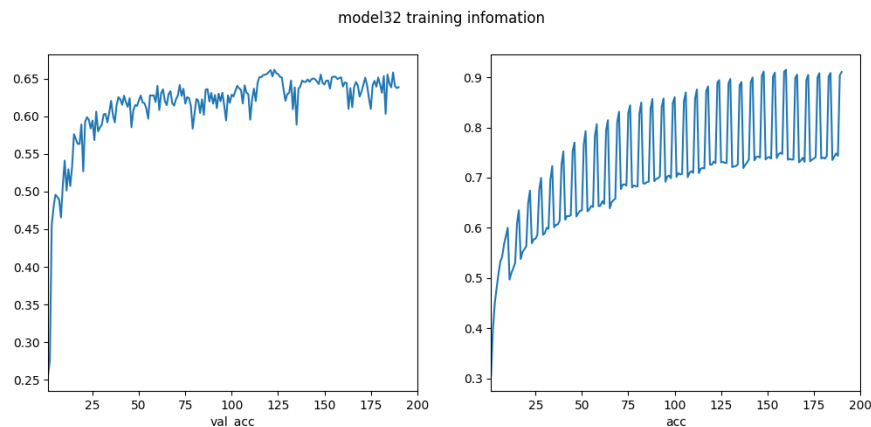
架構可以分成兩個部份來說明：第一個部分 convolution 每層是由 dropout(0.4) (Input 層沒有) → conv2D(3,3) → activation('relu') → batch normalization → max pooling 2D(2,2) 組成；接著經過 flatten 後，連接兩層 dense，第一層 dense 是由 dropout(0.4) → dense(512) → activation('relu') → batch normalization，第二層 dense 是由 dropout(0.4) → dense(1024) → activation('relu') → batch normalization，最後再 dropout(0.3) → dense(7) → activation('softmax') 分類。模型的 loss 是由 categorical crossentropy，optimizer 是 adamax。

我把資料拆成 valid set 是前 5000 項 data 跟 train set 則是剩餘的 data。訓練過程也可以分成兩個 part，我把它分成無雜訊(fit)跟有雜訊(fit generator)兩種，我使用 keras 的 preprocessing 的 image generator 把資料作旋轉(10 度內)，水平上下平移(10% 內)。並且採用 test.csv 當成 unlabel data。

首先我用 fit 訓練 10 個 epoch，接著跑 iteration 30 次，每次先用目前的 model 做 self-training(threshold = 0.9)，並把新的資料 concatenate 進 train_feature，接著做 fit

generator 4 個 epoch，然後用 fit 做 2 個 epoch，我發現交錯使用這兩種訓練過程會使得最後不論 kaggle 跟 valid set 的分數都會提高，我的猜測是因為如果沒有交錯，整個訓練最後會偏向只用 fit，沒有享受到加雜訊增加準確率的效果，或只用 fit generator 可能使得正式的图片反而預測不佳(每次 fit generator, train set accuracy 會下降)。

訓練過程可由下圖表示：



可看出 valid accuracy 在後面 iterations 有緩慢上升，但到後面就已經有飽和的趨勢。準確率在 kaggle public set 的分數為 0.67004。

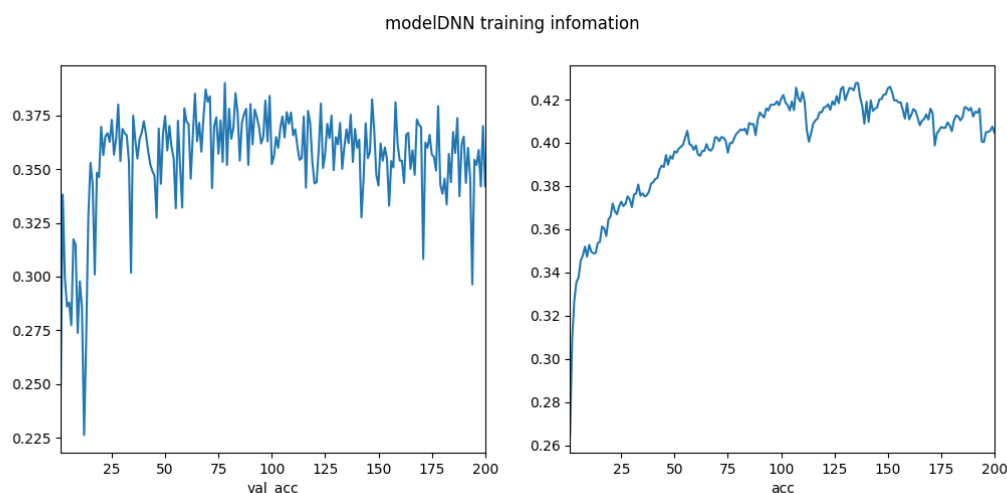
2. (1%) 承上題，請用與上述 CNN 接近的參數量，實做簡單的 DNN model。其模型架構、訓練過程和準確率為何？試與上題結果做比較，並說明你觀察到了什麼？

答：我的 DNN 架構如下：

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 512)	1180160
activation_1 (Activation)	(None, 512)	0
batch_normalization_1 (Batch Normalization)	(None, 512)	2048
dropout_1 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 1024)	525312
activation_2 (Activation)	(None, 1024)	0
batch_normalization_2 (Batch Normalization)	(None, 1024)	4096
dropout_2 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 7)	7175
activation_3 (Activation)	(None, 7)	0
Total params: 1,718,791		
Trainable params: 1,715,719		
Non-trainable params: 3,072		

架構是由數層 dense 組成，第一層 dense 是由 $\text{dense}(512) \rightarrow \text{activation}(\text{'relu'}) \rightarrow \text{batch normalization}$ ，第二層 dense 是由 $\text{dropout}(0.4) \rightarrow \text{dense}(1024) \rightarrow \text{activation}(\text{'relu'}) \rightarrow \text{batch normalization}$ ，最後再 $\text{dropout}(0.3) \rightarrow \text{dense}(7) \rightarrow \text{activation}(\text{'softmax'})$ 分類。模型的 loss 是由 categorical_crossentropy，optimizer 是 adamax。

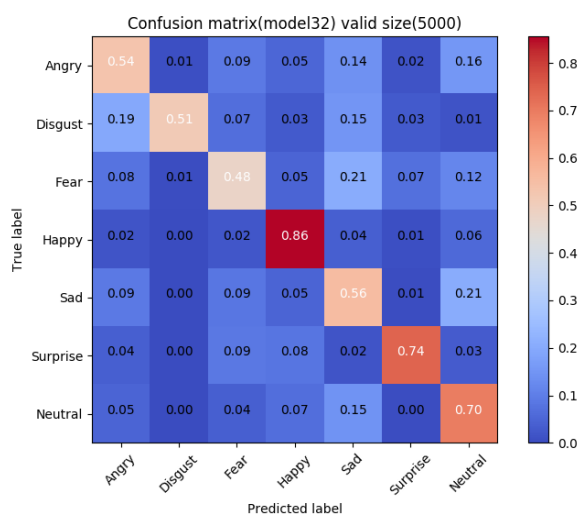
我把資料拆成 valid set 是前 5000 項 data 跟 train set 則是剩餘的 data。訓練過程則是全部使用 fit，直接用 fit 訓練 200 個 epoch。在差不多參數量的情況下，DNN 的預測能力非常差，準確率在 kaggle public set 的分數為 0.35135，而且隨著 epoch 數增加，並無明顯改善，幾乎都是大幅度的跳動，可參考下圖：。



我覺得是因為 DNN 相對 CNN 而言並沒有將圖片分區塊辨識的能力，所以當圖片歧異度很高時，純粹用數值的訓練不容易成功。

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：confusion matrix 如下圖




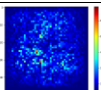
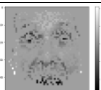

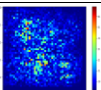
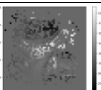

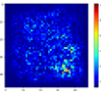


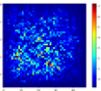
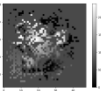

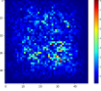
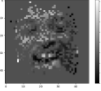

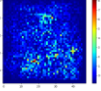
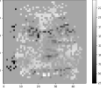

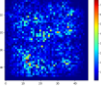
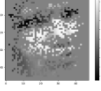

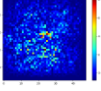
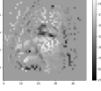

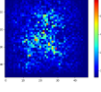
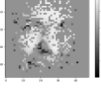

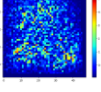

如同訓練 CNN 時的假設(我把資料拆成 valid set 是前 5000 項 data 跟 train set 則是剩餘的 data)，所以這邊是用 valid set 的 data 去做 confusion matrix。

根據圖片可以看出 happy、surprise 的準確率最高，而且 surprise 中錯誤判斷的最多是判斷成 happy，可以把這兩個歸類成「正面情緒」。而其餘 angry, disgust, fear 的誤差就較大，但是判斷錯也大多是歸屬於其他兩項或是 neutral，把這三個歸類成「負面情緒」。

我推測是因為可能正面情緒的表情較為誇張，例如微笑的嘴角上揚，加上同屬正面情緒的選擇較少，所以可能誤差較小。但是因為負面情緒除了選擇多，加上表情小還可能跟 neutral 搞混。

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：saliency maps 可見下表

編號	原圖	saliency map	masked	編號	原圖	saliency map	masked
9				23			
30				74			
187				1111			
1933				2751			
4796				15668			

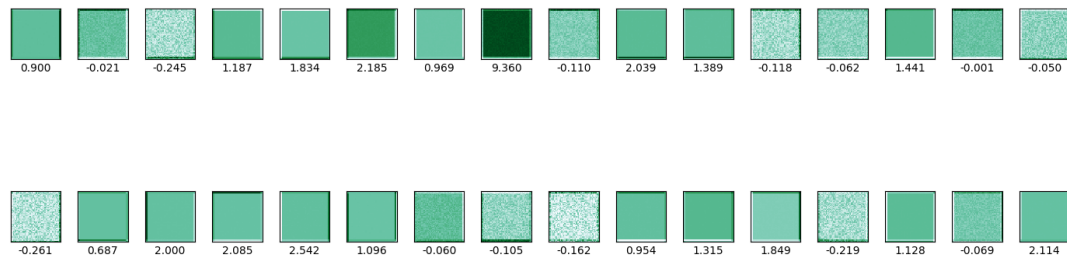
根據上表可以很明顯看到，在眼睛、鼻子、嘴巴、眉毛周圍區塊特別明顯，而我們分別一個人的表情，通常也是靠著這些因素來判斷，所以實作 CNN 確實是有幫助到擷取關鍵 data 再去做 training。

而事實上這次做的這 6 個情緒辨識，這些負面情緒在我們現實世界本來就比較難以透過眼睛鼻子眉毛來區分，更多可能是用眼神或是肌肉變化看出，這可能是我們可以改進的地方。

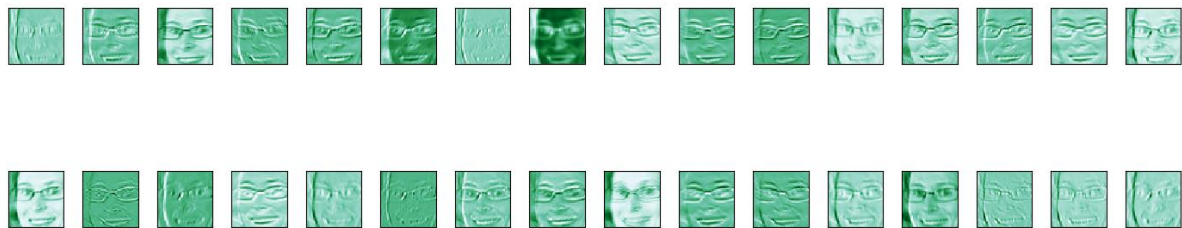
5. (1%) 承(1)(2)，利用上課所提到的 gradient ascent 方法，觀察特定層的 filter 最容易被哪種圖片 activate。

答：gradient ascent 採用的 learning rate 為 1，並訓練 20 次。我觀察了 conv2d_1、conv2d_2、conv2d_3 的 filter，並把最後的 loss 記錄在 filter 下方，filter output 則是用編號 11 的圖說明，結果如下：

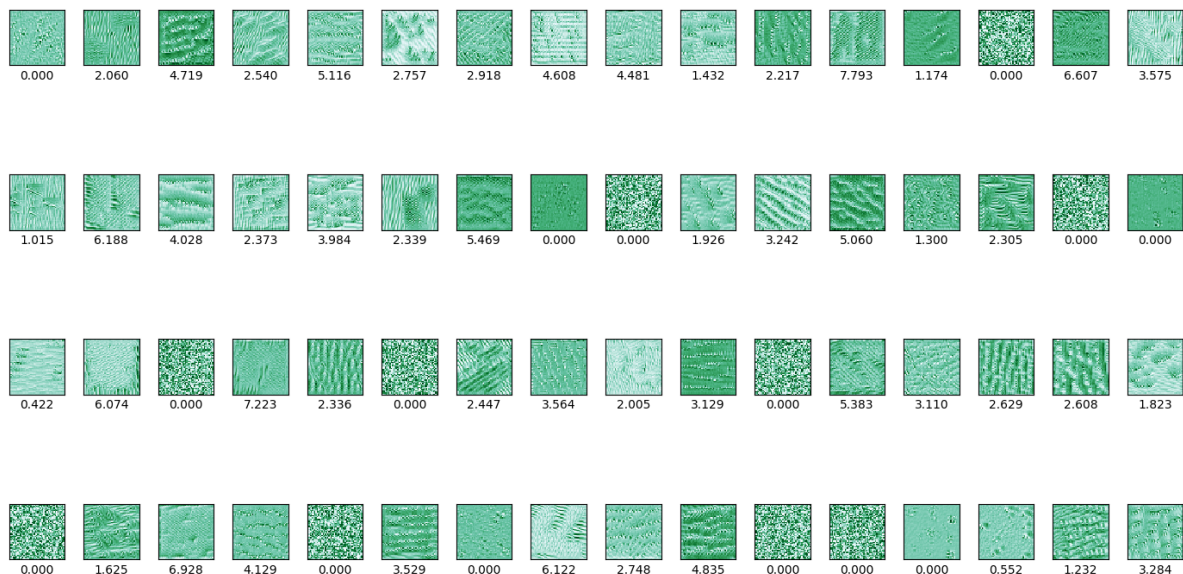
Filters of layer (conv2d_1) (# Ascent Epoch 20)



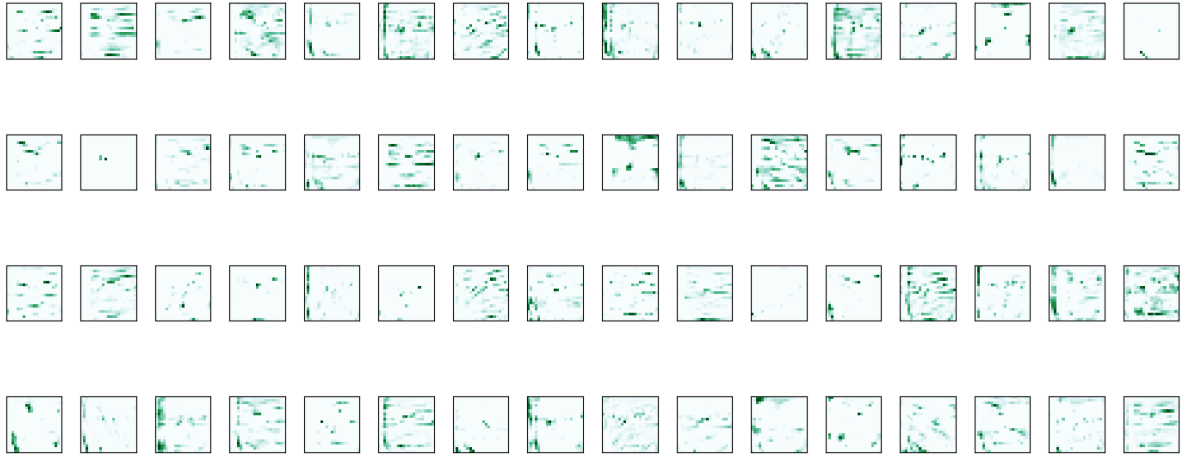
Output of layer(conv2d_1) (Given image 1111)



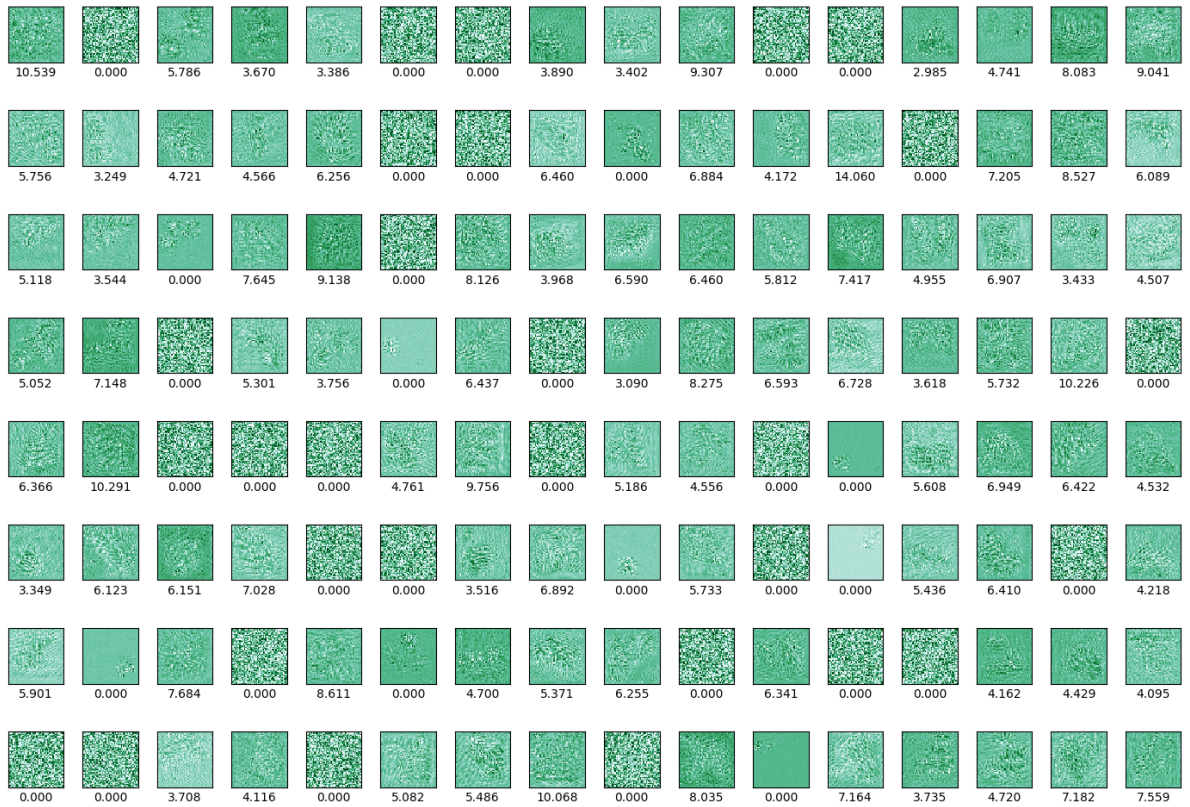
Filters of layer (conv2d_2) (# Ascent Epoch 20)

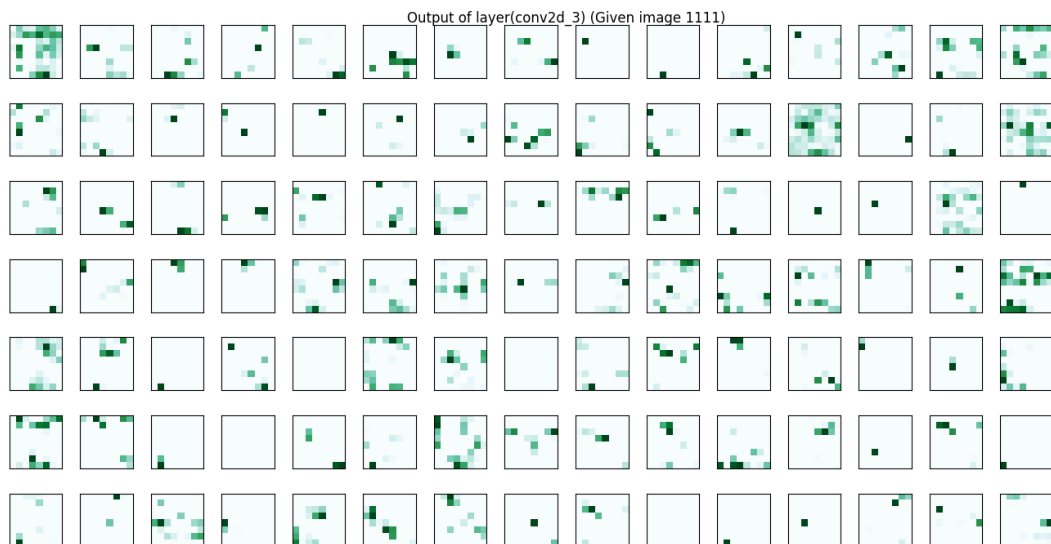


Output of layer(conv2d_2) (Given image 1111)



Filters of layer (conv2d_3) (# Ascent Epoch 20)

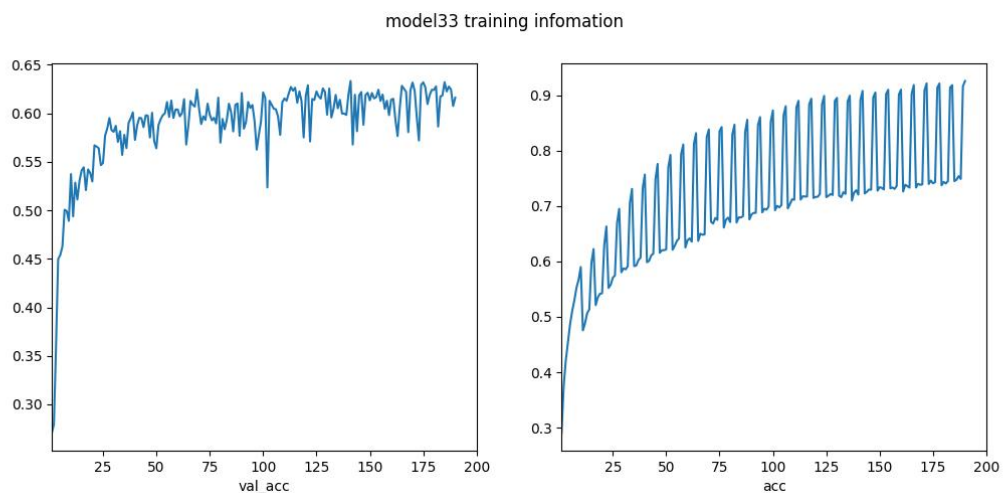




從上面數張圖可以看出，最終的聚集點是眼睛、鼻子、嘴巴，跟 saliency map 分析的情況差不多，此外可以看出 conv2d_1 大致圖片都還算完整，到了 conv2d_2 圖片開始有點扭曲，最後到 conv2d_3，就是機器用他自己的方法找出聚焦點了。

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

我採用的是 self-training 的方式，並且將預測結果視為真的 label 的閾值定在 0.9。首先把資料拆成三個部分：valid set 是前 5000 項、unlabel set 是前 5001 到 10000 項(r 約等於 0.2)、train set 則是剩餘的 data，model 的架構如同第一題所示。model33 實作了 semi-supervised，model35(沒有 self training)一樣是用相同的 valid set 和 train set，兩者的準確率(kaggle 分數)分別為 0.66620 和 0.64113。兩者訓練過程如下圖：





可以看出做 self-training 的 model33，因為加入的是不能完全確定的 label，所以 valid_acc 會有劇烈變化，比較有可能持續上升。

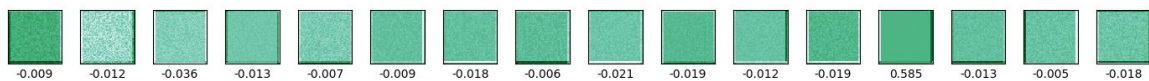
此外我實作了用 test.csv 當作 self-training 的 unlabel data，發現這個 model(32)是所有裡面最好的，準確率(kaggle 分數)是 0.67004。訓練過程如下圖：



[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%]

(A) 使用較差 model (kaggle 分數 0.57008)，三層 conv 分別有 25、50、100 個 filter，這邊選出前 16、48、96 個 filter，並一樣在 filter 下方標示 loss，選用編號 1111 圖片說明：

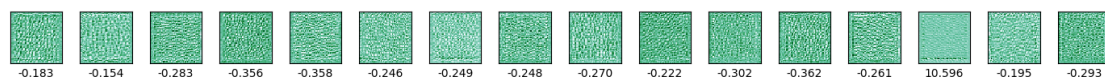
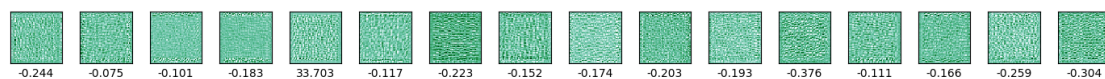
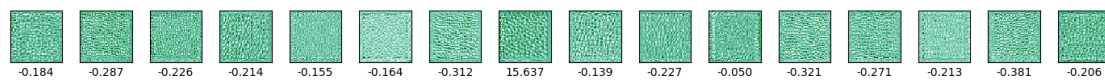
Filters of layer (conv2d_1) (# Ascent Epoch 20)



Output of layer(conv2d_1) (Given image 1111)

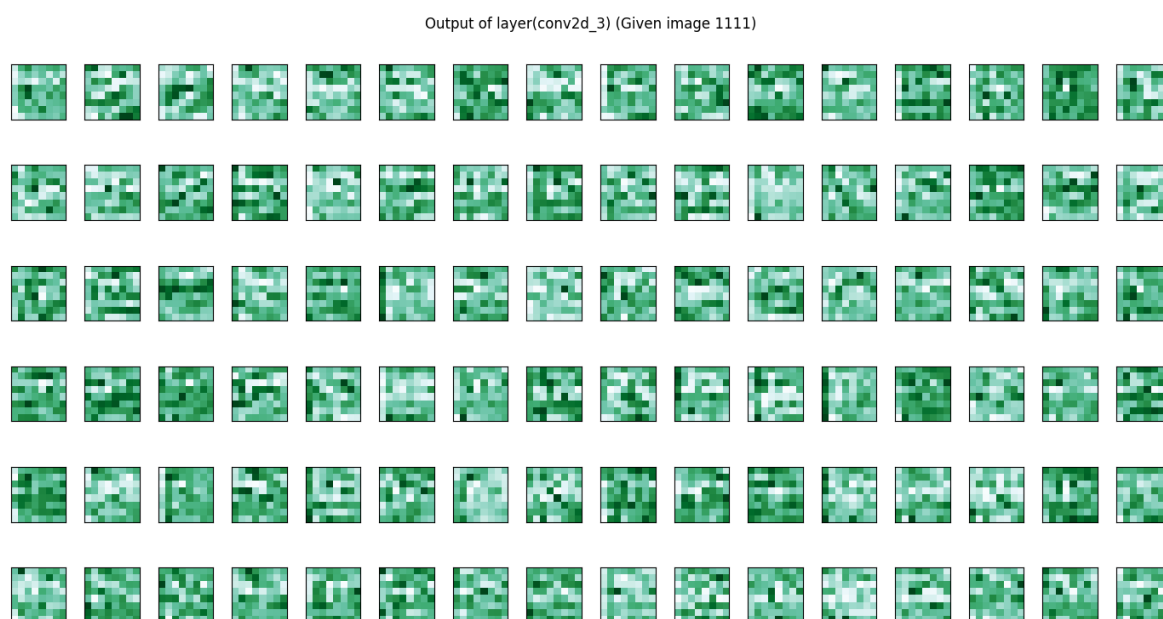


Filters of layer (conv2d_2) (# Ascent Epoch 20)



Output of layer(conv2d_2) (Given image 1111)





從 conv2d_1 的 filter 和結果看不出明顯差異。但到了 conv2d_2 便可看出，較差的 model filter 長的都差不多，而且幾乎沒有變化，但是較好 model 的變化明顯，每個 filter 的歧異度很高，看起來有斜線、點點等分，這也展示在圖片結果上，較差的 model 仍是一片臉，但是較好 model 已經開始只剩下輪廓，跟全部的五官。

最明顯的就是 conv2d_3，一樣較差的 model filter 幾乎全部類型相同，filter 本身變化也不大，但較好 model filter 的本身變化就較大，反映到結果，較差的 model 是一塊塊好像打了馬賽克，雖然有開始分類的感覺，但相比較好的 model 已經聚焦到某個很小區域，可以看出從前面 convolution 層就是造成預測準確率的關鍵。