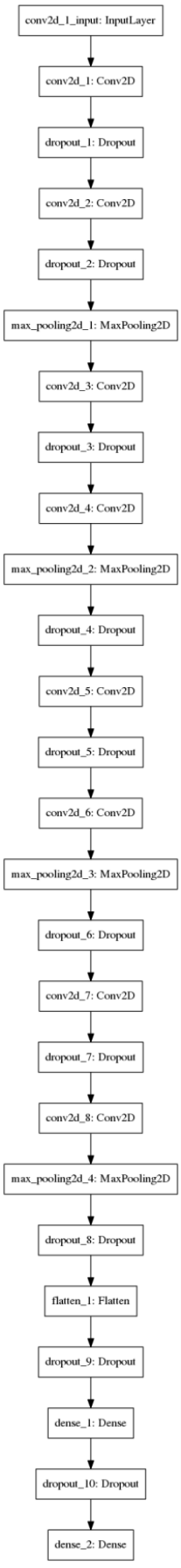
學號：B03705006 系級： 資管三 姓名：侯舜元

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：



模型架構(見左下二圖)：

每疊兩層 Convolutional layer 後就搭配一次 2x2 的 max_pooling，每層間都有不同比例的 dropout(多為 0.2~0.3)。

filter 的大小皆固定為 3x3（推測圖片要 3x3 才能比較好觀察他的 pattern，2x2、2x1 的準確率皆不佳）。而 Conv2D 的 filter 數由 32,64,128,256 倍增。Convolutional layer 的 activation 皆使用 relu。

CNN 完後接一層 Dense layer 後就接 output layer。有試過多層 DNN 並沒有顯著的提升準確率，故後來就只使用一層。最後的 optimizer 使用 adam，learning rate 設為 0.00025。

訓練過程：

在資料的部分，有先人工將 data 裡中網路死圖、無法辨識人臉等圖片去除(50張左右)。使用 95~97%的 training data 下去 train，嘗試 50~200 epoch(最佳結果為 150 epoch)。有很長一段時間卡在 0.63 上下，後來將 DNN 的 channel 數調大以及將 dropout rate 調高後就有顯著進步。(筆電硬體不佳，train 一次要耗時 10+小時)

準確率：在本機端自己切 valid data 的準確度為 65.0000000832%。Kaggle 上 public 的準確度為 0.65171。

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 46, 46, 32)	320
dropout_1 (Dropout)	(None, 46, 46, 32)	0
conv2d_2 (Conv2D)	(None, 44, 44, 32)	9248
dropout_2 (Dropout)	(None, 44, 44, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 22, 22, 32)	0
conv2d_3 (Conv2D)	(None, 22, 22, 64)	18496
dropout_3 (Dropout)	(None, 22, 22, 64)	0
conv2d_4 (Conv2D)	(None, 22, 22, 64)	36928
max_pooling2d_2 (MaxPooling2	(None, 11, 11, 64)	0
dropout_4 (Dropout)	(None, 11, 11, 64)	0
conv2d_5 (Conv2D)	(None, 11, 11, 128)	73856
dropout_5 (Dropout)	(None, 11, 11, 128)	0
conv2d_6 (Conv2D)	(None, 11, 11, 128)	147584
max_pooling2d_3 (MaxPooling2	(None, 5, 5, 128)	0
dropout_6 (Dropout)	(None, 5, 5, 128)	0
conv2d_7 (Conv2D)	(None, 5, 5, 256)	295168
dropout_7 (Dropout)	(None, 5, 5, 256)	0
conv2d_8 (Conv2D)	(None, 5, 5, 256)	590080
max_pooling2d_4 (MaxPooling2	(None, 2, 2, 256)	0
dropout_8 (Dropout)	(None, 2, 2, 256)	0
flatten_1 (Flatten)	(None, 1024)	0
dropout_9 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 1024)	1049600
dropout_10 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 7)	7175

Total params: 2,228,455
Trainable params: 2,228,455
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？

答：

模型架構：

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 800)               1844000
_________________________________________________________________
dropout_1 (Dropout)          (None, 800)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1600)              1281600
_________________________________________________________________
dropout_2 (Dropout)          (None, 1600)              0
_________________________________________________________________
dense_3 (Dense)              (None, 7)                 11207
=================================================================
Total params: 3,136,807
Trainable params: 3,136,807
Non-trainable params: 0
```

簡單的 DNN Model 如上圖所示，兩層 Dense Layer 後就接 output layer。Dropout rate 設為 0.45。

訓練過程：

先將網路死圖、無法辨識人臉等圖片去除。使用 95~97%的 training data 下去 train，一樣嘗試 50~150 epoch。DNN 很迅速就能夠訓練完。

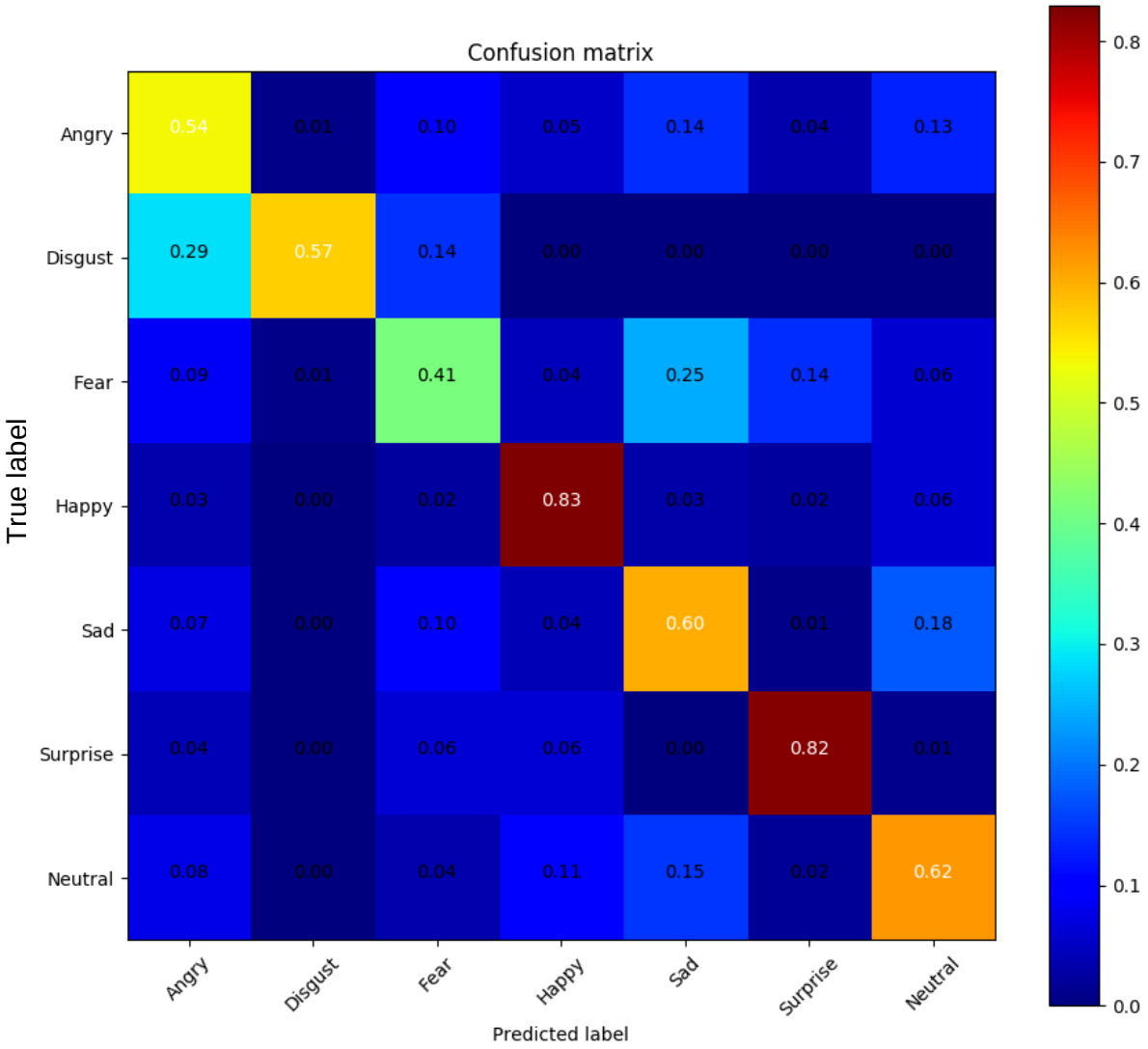
準確率： 34.94%

跟上題所比較，明顯看出雖然參數在同一個數量級(甚至 DNN 較多)，但是 DNN Model 的準確度低相當的多。將 CNN 疊深的威力就可以在此顯現。光靠一個很肥的 DNN 是比不上較深的 CNN Model。

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:



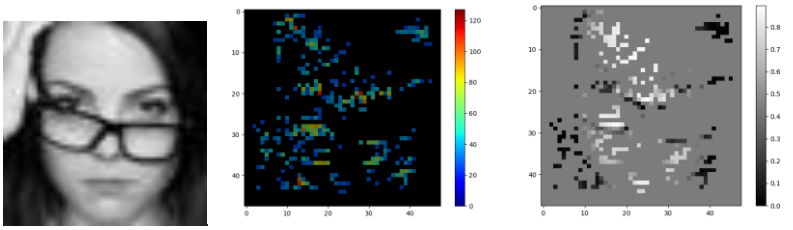
從 Confusion Matrix 中可以看出 Happy 跟 Surprise 兩者皆屬於較不容易搞混的 class，推測圖中的人表情都較大，故比較容易辨別。而 Disgust 和 Angry 兩者屬於最容易搞混的 class。厭惡的表情容易被推測成生氣。另外 Fear 和 Sad 也較容易搞混，兩者皆偏向比較負面的情緒。

`file: confusion_matrix.py`

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？

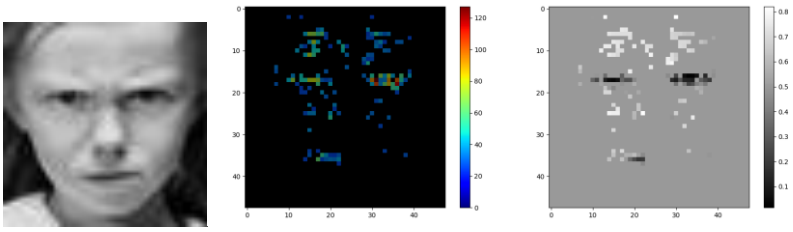
   答： 以下的圖片從左到右邊分別是原圖、Saliency Map、Mask 掉 heat 的圖 (threshold 設為 0.6)

   `file: saliency_map.py`



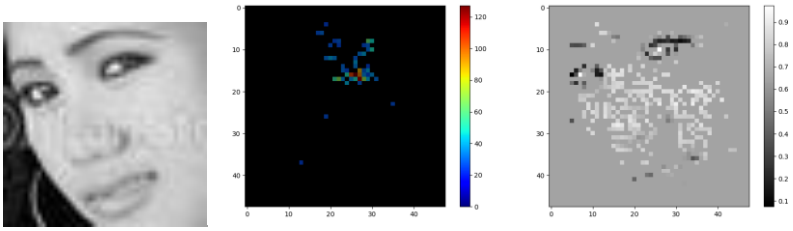
(1019, training data)

模型在作 classification 時，可以看到有 focus 在眼睛、眼鏡和臉型四周的地方。已經可以依稀看出臉的輪廓。

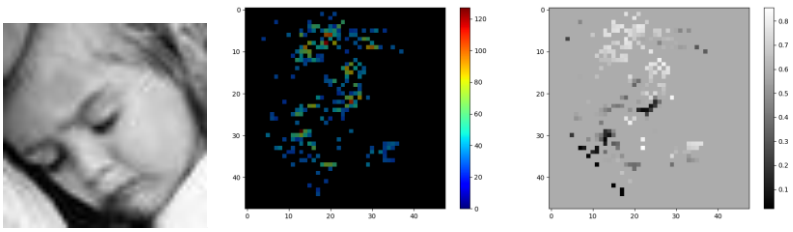


(1018, training data)

這張也是主要 focus 在眼睛，以及一點唇部。

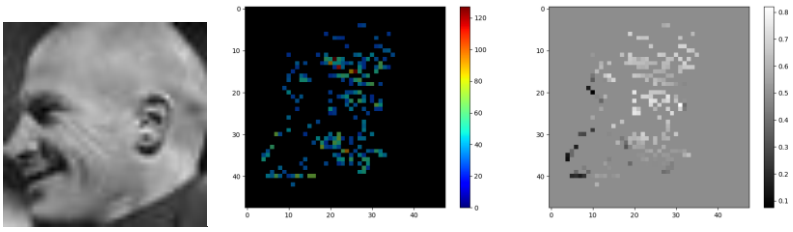


(1, testing data)



(23, testing data)

這兩張是沒被 train 過的 testing data。無論是正臉或是偏側臉，只要雙眼都可見，可以明顯看出模型是 focus 在眼睛的地方。

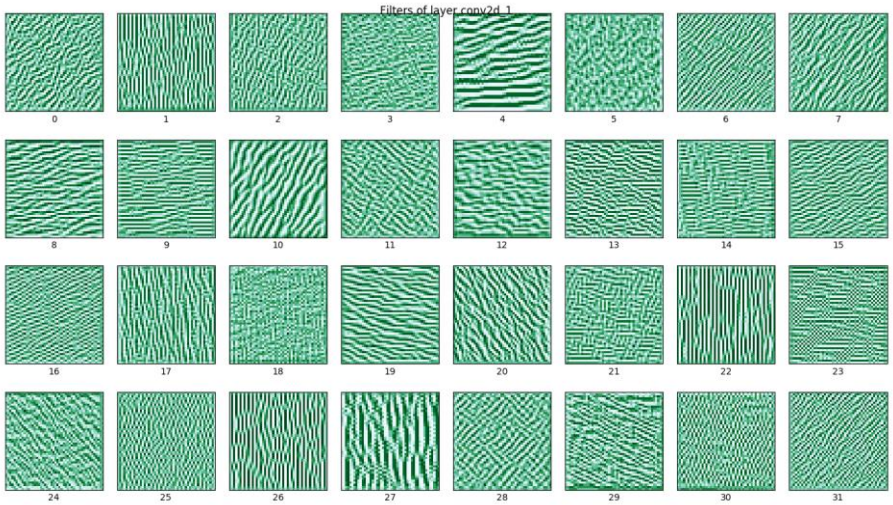


(45, testing data)

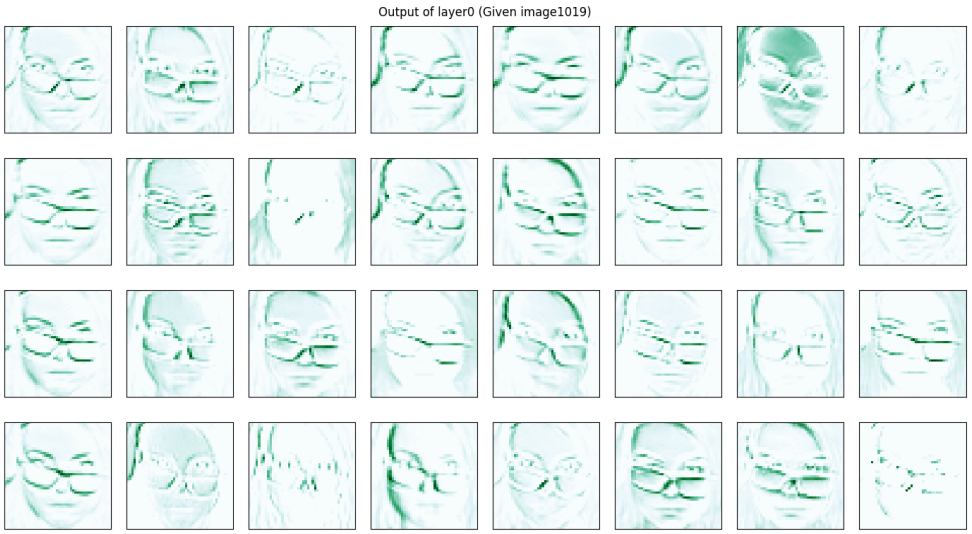
這張是沒被 train 過的 testing data，由於是側臉，雙眼並沒有在圖片中顯示，故也較難看出模型聚焦於臉的何處。

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：

`<1> 65% acc.的 Model，convolutional layer 2d_1 with 32 filters`

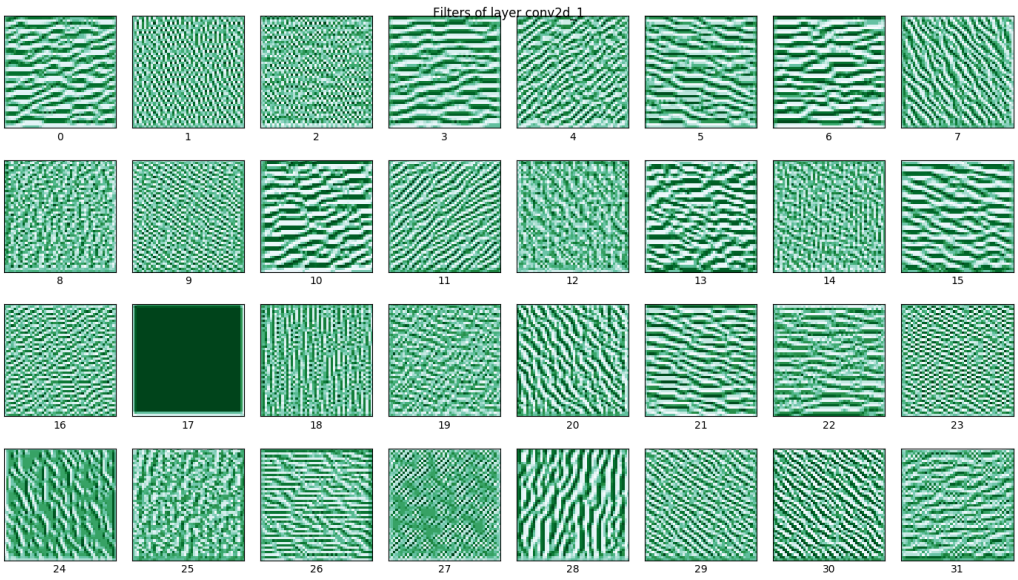


觀察第一層 convolutional layer 的 32 個 filter。由於這是第一層，所以推測大概都只會看到基本的紋理組合，不會有太複雜的圖形。推測跟觀察到的差不多，容易激活不同 filter 的圖片有著不同的基本紋路。但是如果仔細觀察，可以看到其實有些 filter 的圖片是很類似的，只是隨機的轉了不同的角度。例如編號 1、22、26filter 的圖其實紋路和 pattern 是非常接近的只是角度不太一樣。

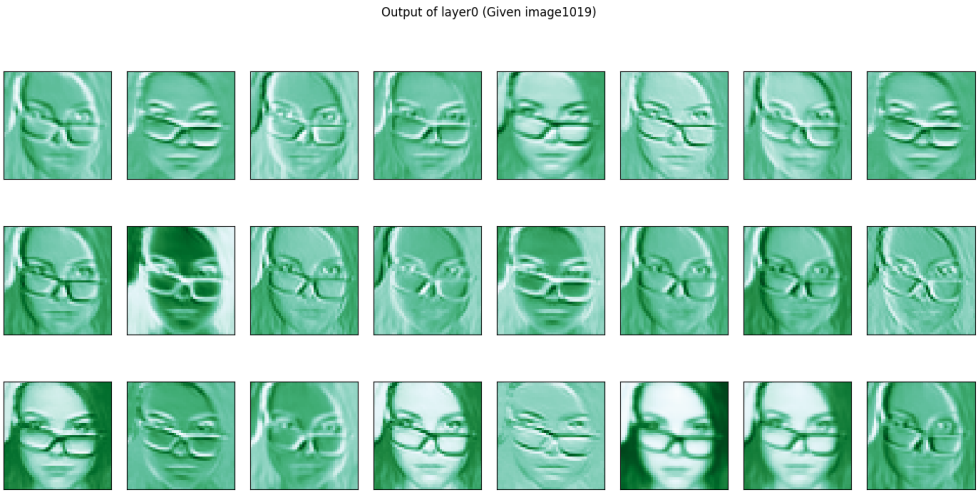


將編號 1019 的照片 input 到第一層 convolutional layer。可以看到許多 filter 加強了五官的輪廓，尤其是眼睛(眼鏡)的部分。且幾乎只留下輪廓。

<2> 55% acc.的 Model，convolutional layer 2d_1 with 32 filters



這是準確度比較差的模型的結果，一樣是由一些基本紋路組成。跟前面模型比較不同的是可以看編號 17 的 filter 最能激活的圖形是全黑的。



不像前面的模型只留下輪廓，這個模型的結果人臉仍然清楚可見。亦有稍稍加強臉的輪廓。

`file：visual_filter.py image2layer.py`

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

只保留約 70%的 training data 當作 label data，剩下的移除 label 去做 semi-supervised learning（使用投影片上 self-learning 的方法）。Model 結構如下附圖。而每一個 iteration 後將 1000 筆 unlabeled data 改成 pseudo label 去 train 下一個模型。

Iteration	1(Init.)	2	3	4	5	6	7
Accuracy	acc: 49.19%	acc: 47.38%	acc: 50.41%	acc: 48.60%	acc: 49.07%	acc: 48.49%	acc: 49.94%

可以看到每次 train 出來新的 model 其實沒有太大的進步，跟一開始只使用 label data train 出來的 model 相差無幾。有可能是我取的 label data 太多導致。

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 46, 46, 16)        160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 16)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 23, 23, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 21, 21, 32)        4640
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 32)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 10, 10, 32)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 3200)              0
_________________________________________________________________
dense_1 (Dense)              (None, 400)               1280400
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 2807
=================================================================
Total params: 1,288,007
Trainable params: 1,288,007
Non-trainable params: 0
```

file：semi-supervised.py

[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%]