學號：B049091067 系級： 電機三 姓名：陳博彥

1. (1%) 請說明你實作的 CNN model，其模型架構、訓練參數和準確率為何？

Model 1

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 48, 48, 64)	1664
max_pooling2d_1 (MaxPooling2	(None, 24, 24, 64)	0
dropout_1 (Dropout)	(None, 24, 24, 64)	0
conv2d_2 (Conv2D)	(None, 24, 24, 64)	102464
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 64)	0
dropout_2 (Dropout)	(None, 12, 12, 64)	0
conv2d_3 (Conv2D)	(None, 12, 12, 64)	102464
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 64)	0
dropout_3 (Dropout)	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 1024)	2360320
dropout_4 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1024)	1049600
dropout_5 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 1024)	1049600
dropout_6 (Dropout)	(None, 1024)	0
dense_4 (Dense)	(None, 7)	7175

Model 2

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 48, 48, 64)	1664
leaky_re_lu_1 (LeakyReLU)	(None, 48, 48, 64)	0
batch_normalization_1 (Batch	(None, 48, 48, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 24, 24, 64)	0
dropout_1 (Dropout)	(None, 24, 24, 64)	0
conv2d_2 (Conv2D)	(None, 24, 24, 128)	204928
leaky_re_lu_2 (LeakyReLU)	(None, 24, 24, 128)	0
batch_normalization_2 (Batch	(None, 24, 24, 128)	512
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 128)	0
dropout_2 (Dropout)	(None, 12, 12, 128)	0
conv2d_3 (Conv2D)	(None, 12, 12, 256)	819456
leaky_re_lu_3 (LeakyReLU)	(None, 12, 12, 256)	0
batch_normalization_3 (Batch	(None, 12, 12, 256)	1024
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 256)	0
dropout_3 (Dropout)	(None, 6, 6, 256)	0
flatten_1 (Flatten)	(None, 9216)	0
dense_1 (Dense)	(None, 256)	2359552
batch_normalization_4 (Batch	(None, 256)	1024
dropout_4 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 512)	131584
batch_normalization_5 (Batch	(None, 512)	2048
dropout_5 (Dropout)	(None, 512)	0

Model 3

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 48, 48, 70)	1820
p_re_lu_1 (PReLU)	(None, 48, 48, 70)	161280
batch_normalization_1 (Batch	(None, 48, 48, 70)	280
max_pooling2d_1 (MaxPooling2	(None, 24, 24, 70)	0
dropout_1 (Dropout)	(None, 24, 24, 70)	0
conv2d_2 (Conv2D)	(None, 24, 24, 130)	82030
p_re_lu_2 (PReLU)	(None, 24, 24, 130)	74880
batch_normalization_2 (Batch	(None, 24, 24, 130)	520
max_pooling2d_2 (MaxPooling2	(None, 12, 12, 130)	0
dropout_2 (Dropout)	(None, 12, 12, 130)	0
conv2d_3 (Conv2D)	(None, 12, 12, 500)	585500
p_re_lu_3 (PReLU)	(None, 12, 12, 500)	72000
batch_normalization_3 (Batch	(None, 12, 12, 500)	2000
max_pooling2d_3 (MaxPooling2	(None, 6, 6, 500)	0
dropout_3 (Dropout)	(None, 6, 6, 500)	0
conv2d_4 (Conv2D)	(None, 6, 6, 500)	2250500
p_re_lu_4 (PReLU)	(None, 6, 6, 500)	18000
batch_normalization_4 (Batch	(None, 6, 6, 500)	2000
max_pooling2d_4 (MaxPooling2	(None, 3, 3, 500)	0
dropout_4 (Dropout)	(None, 3, 3, 500)	0
flatten_1 (Flatten)	(None, 4500)	0
dense_1 (Dense)	(None, 500)	2250500
batch_normalization_5 (Batch	(None, 500)	2000
dropout_5 (Dropout)	(None, 500)	0
dense_2 (Dense)	(None, 500)	250500
batch_normalization_6 (Batch	(None, 500)	2000
dropout_6 (Dropout)	(None, 500)	0
dense_3 (Dense)	(None, 7)	3507

Model 1 : batch size = 300, epoch = 300, Public score = 0.59793, Private score = 0.59849

Model 2 : batch size = 300, epoch = 300, with data augmentation, Public score = 0.63945, Private score = 0.64558

Model 3 : batch size = 120 epoch = 1000, with data augmentation, Public score = 0.70632, Private score = 0.70270

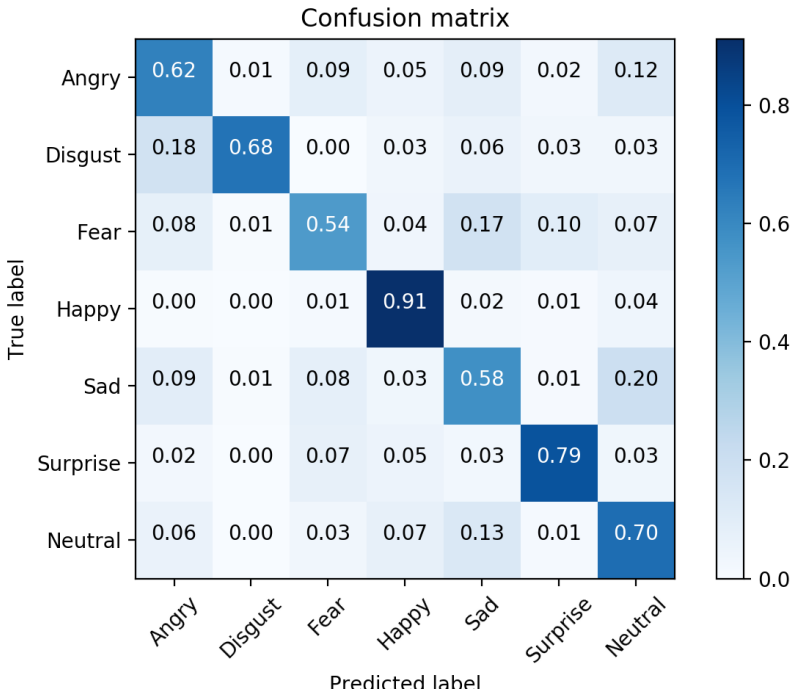
2. (1%) 請嘗試 data normalization, data augmentation,說明實行方法並且說明對準確率有什麼樣的影響？

我使用第一題中model 2的架構做測試（batch size=300, iteration = 30），data normalization的方式是在model中加入batch normalization，data augmentation的方式是用ImageDataGenerator()進行生成。結果如下表（分數為public跟private的平均）。

	normalization	without normalization
augmentation	0.63948	0.62983
without augmentation	0.59581	0.58912

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]

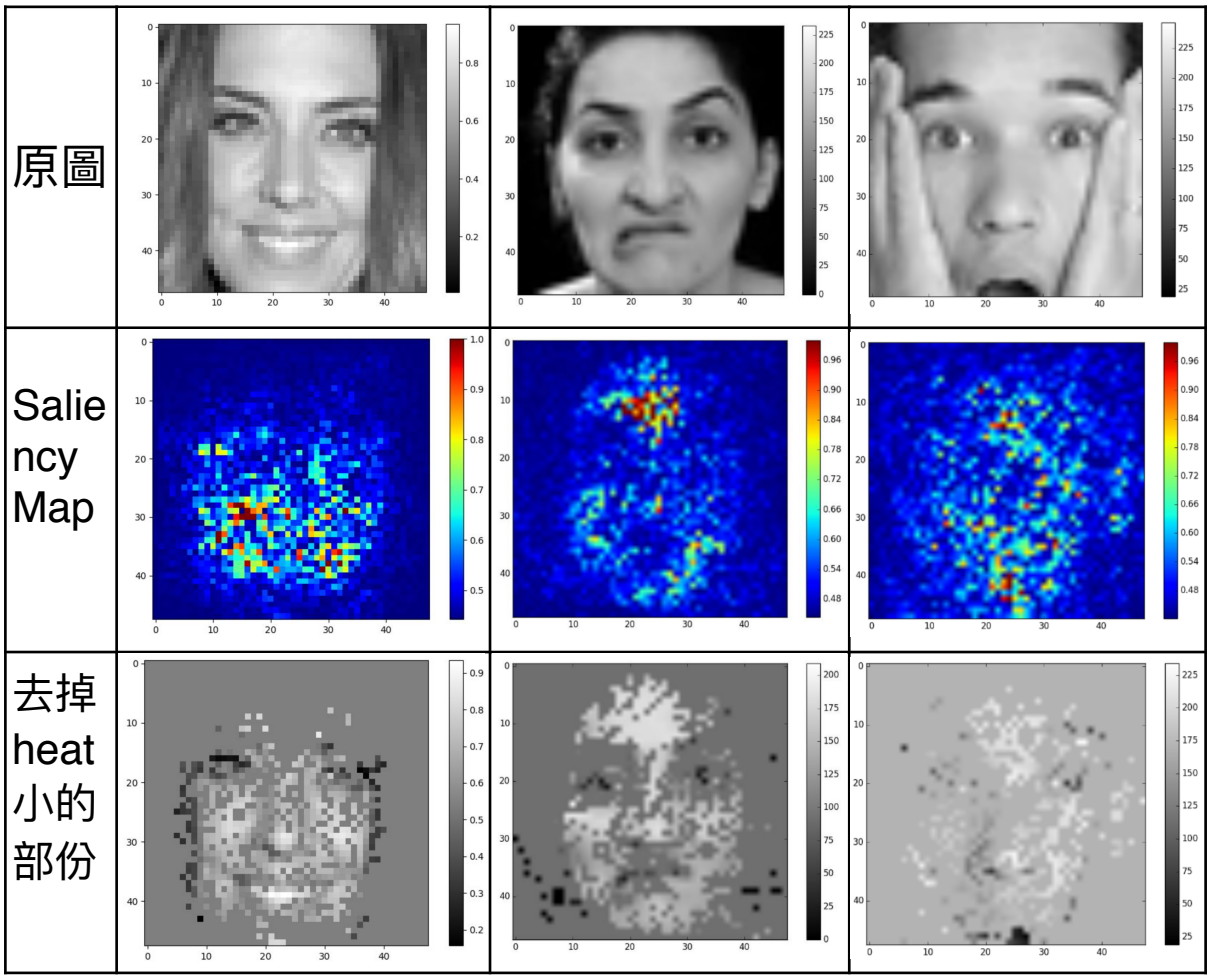
基本上，Happy是最容易被分辨的。較不容易分辨的，有Sad跟Neutral、Disgust跟Fear、Sad跟Fear。



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？

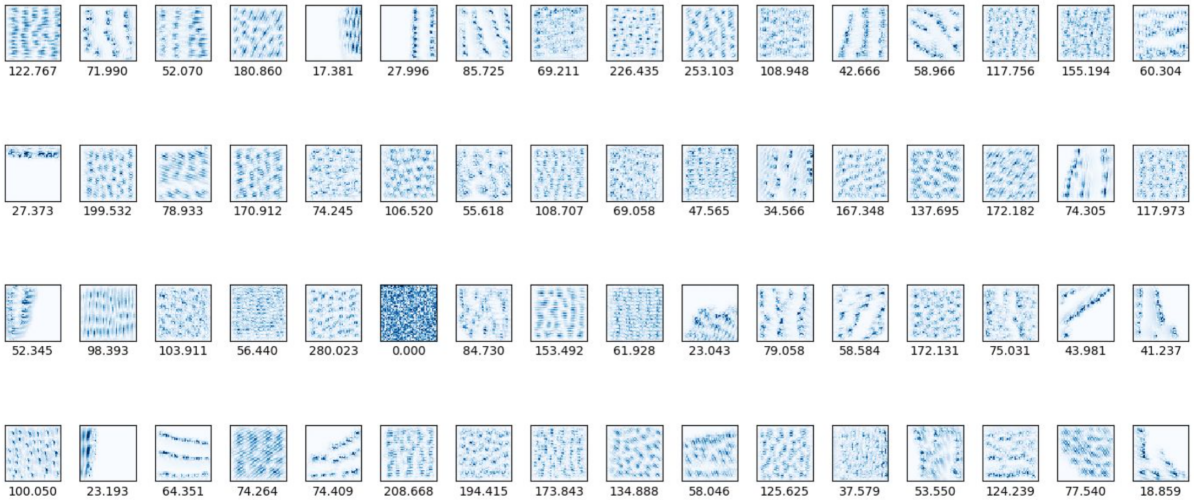
由以下結果可以看見，第一張圖（labeled happy），focus在嘴巴部分。Happy這個類別，很可能是所有類別中最容易露出牙齒的，model聚焦在這邊很合理。第

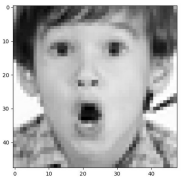
二張（angry），聚焦在眼睛，第三張（fear）比較均勻分布。觀測去掉heat小的結果，可以看見，第一張圖（happy）依然可以輕易用肉眼辨別為happy，這也說明為何第三題中，happy的辨識率那麼高。

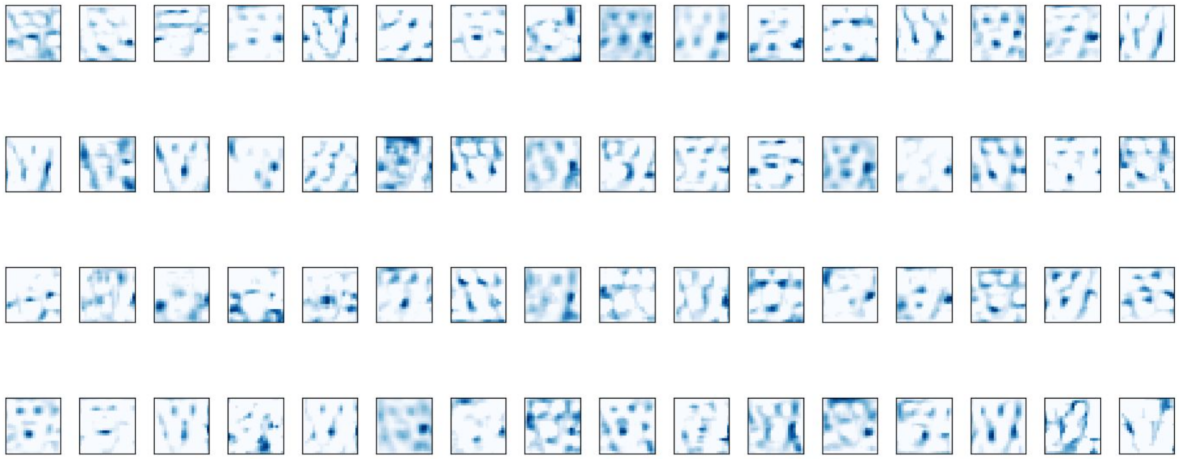


5. (1%) 承(4) 利用上課所提到的 gradient ascent 方法，觀察特定層的filter最容易被哪種圖片 activate與觀察filter的output。(Collaborators:)

第二層Convolution，epoch = 300。



input 



在gradient ascent，可看出，由於人臉具有對稱性，常常可看見兩個filter非常相似，只是左右翻轉，例如第四row的第三個第五個。

當圖片經過這個layer後，可以看見，特徵被加深，像是眼睛跟嘴巴部分，多餘的部分則變為空白。