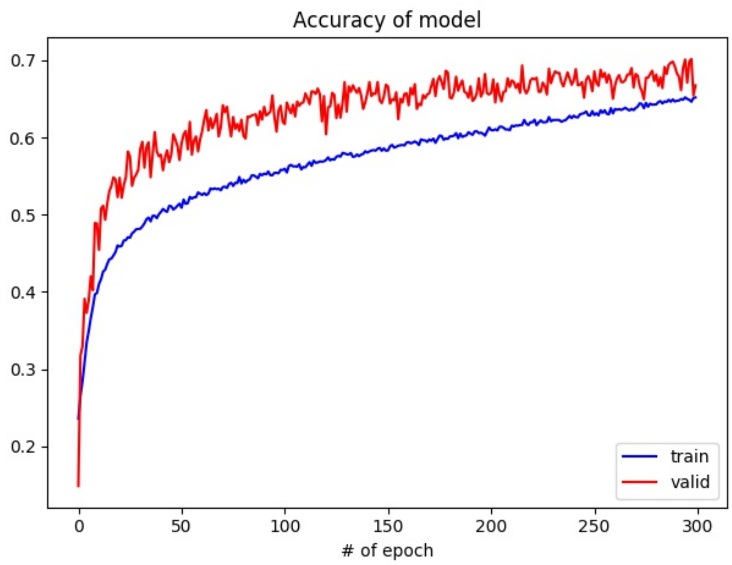
學號：R03922165 系級： 資工碩二 姓名：李冠穎

1.

大致 follow VGG16 的結構, 加入 batch normalization, 如下圖 Keras model.summary()

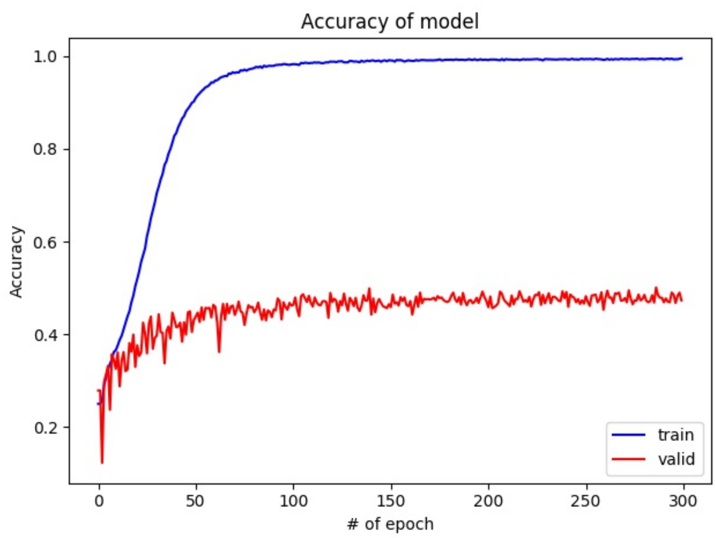
(type)	Output Shape	Param #
k1_conv1 (Conv2D)	(None, 48, 48, 64)	640
y_re_lu_1 (LeakyReLU)	(None, 48, 48, 64)	0
l_c1 (BatchNormalization	(None, 48, 48, 64)	256
k1_conv2 (Conv2D)	(None, 48, 48, 64)	36928
y_re_lu_2 (LeakyReLU)	(None, 48, 48, 64)	0
l_c2 (BatchNormalization	(None, 48, 48, 64)	256
k1_pool (MaxPooling2D)	(None, 24, 24, 64)	0
k2_conv1 (Conv2D)	(None, 24, 24, 128)	73856
y_re_lu_3 (LeakyReLU)	(None, 24, 24, 128)	0
2_c1 (BatchNormalization	(None, 24, 24, 128)	512
k2_conv2 (Conv2D)	(None, 24, 24, 128)	147584
y_re_lu_4 (LeakyReLU)	(None, 24, 24, 128)	0
2_c2 (BatchNormalization	(None, 24, 24, 128)	512
k2_pool (MaxPooling2D)	(None, 12, 12, 128)	0
k3_conv1 (Conv2D)	(None, 12, 12, 256)	295168
y_re_lu_5 (LeakyReLU)	(None, 12, 12, 256)	0
3_c1 (BatchNormalization	(None, 12, 12, 256)	1024
k3_conv2 (Conv2D)	(None, 12, 12, 256)	590080
y_re_lu_6 (LeakyReLU)	(None, 12, 12, 256)	0
3_c2 (BatchNormalization	(None, 12, 12, 256)	1024
block3_conv3 (Conv2D)	(None, 12, 12, 256)	590080
leaky_re_lu_7 (LeakyReLU)	(None, 12, 12, 256)	0
bn_b3_c3 (BatchNormalization	(None, 12, 12, 256)	1024
block3_pool (MaxPooling2D)	(None, 6, 6, 256)	0
block4_conv1 (Conv2D)	(None, 6, 6, 512)	1180160
leaky_re_lu_8 (LeakyReLU)	(None, 6, 6, 512)	0
bn_b4_c1 (BatchNormalization	(None, 6, 6, 512)	2048
block4_conv2 (Conv2D)	(None, 6, 6, 512)	2359808
leaky_re_lu_9 (LeakyReLU)	(None, 6, 6, 512)	0
bn_b4_c2 (BatchNormalization	(None, 6, 6, 512)	2048
block4_conv3 (Conv2D)	(None, 6, 6, 512)	2359808
leaky_re_lu_10 (LeakyReLU)	(None, 6, 6, 512)	0
bn_b4_c3 (BatchNormalization	(None, 6, 6, 512)	2048
block4_pool (MaxPooling2D)	(None, 3, 3, 512)	0
block5_conv1 (Conv2D)	(None, 3, 3, 512)	2359808
leaky_re_lu_11 (LeakyReLU)	(None, 3, 3, 512)	0
bn_b5_c1 (BatchNormalization	(None, 3, 3, 512)	2048
block5_conv2 (Conv2D)	(None, 3, 3, 512)	2359808
leaky_re_lu_12 (LeakyReLU)	(None, 3, 3, 512)	0
bn_b5_c2 (BatchNormalization	(None, 3, 3, 512)	2048
block5_conv3 (Conv2D)	(None, 3, 3, 512)	2359808
leaky_re_lu_13 (LeakyReLU)	(None, 3, 3, 512)	0
bn_b5_c3 (BatchNormalization	(None, 3, 3, 512)	2048
block5_pool (MaxPooling2D)	(None, 1, 1, 512)	0
flatten (Flatten)	(None, 512)	0
fc1 (Dense)	(None, 4096)	2101248
leaky_re_lu_14 (LeakyReLU)	(None, 4096)	0
dropout_1 (Dropout)	(None, 4096)	0
fc2 (Dense)	(None, 4096)	1678131
leaky_re_lu_15 (LeakyReLU)	(None, 4096)	0
dropout_2 (Dropout)	(None, 4096)	0
fc3 (Dense)	(None, 1000)	4097000
leaky_re_lu_16 (LeakyReLU)	(None, 1000)	0
dropout_3 (Dropout)	(None, 1000)	0
predictions (Dense)	(None, 7)	7007

Total params: 37,716,999
Trainable params: 37,708,551
Non-trainable params: 8,448



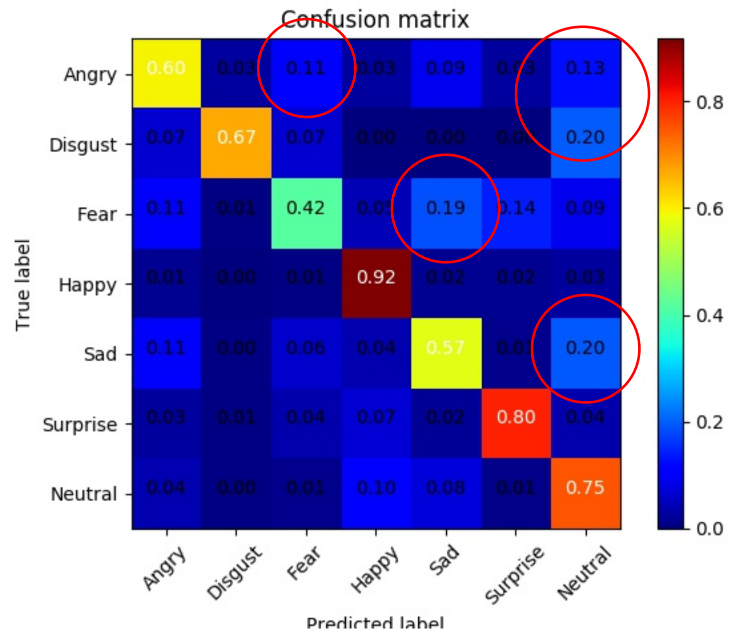
performance based on 20-fold cross-validation 可以到 0.7 左右(如左圖), 有個比較特別值得注意的點是加入 batch normalization 之後, 模型的 validation accuracy 總是比 training 還高,不過礙於時間因素,還沒能深入探討,猜測或許跟 training 的時候利用 Keras 所提供的 image data generator 來 augment data 也有關係。

2. 若使用相同參數量的 fc model (模型結構如下圖), validation accuracy 就只能到 0.5 左右(右圖), 但 training accuracy 可以接近 1.0, 直觀來看因為 convolution 可以算是某種形式上的 regularization, 不會如 fully connected 這般 over fitting, 不過可以探討的點或許是加入 dropout 能讓兩者的 performance 接近多少。



Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 2000)	4610000
dense_2 (Dense)	(None, 2000)	4002000
dense_3 (Dense)	(None, 2000)	4002000
dense_4 (Dense)	(None, 2000)	4002000
dense_5 (Dense)	(None, 2000)	4002000
dense_6 (Dense)	(None, 2000)	4002000
dense_7 (Dense)	(None, 1000)	2001000
dense_8 (Dense)	(None, 1000)	1001000
dense_9 (Dense)	(None, 1000)	1001000
dense_10 (Dense)	(None, 1000)	1001000
dense_11 (Dense)	(None, 1000)	1001000
dense_12 (Dense)	(None, 1000)	1001000
dense_13 (Dense)	(None, 1000)	1001000
dense_14 (Dense)	(None, 1000)	1001000
dense_15 (Dense)	(None, 1000)	1001000
dense_16 (Dense)	(None, 1000)	1001000
dense_17 (Dense)	(None, 1000)	1001000
dense_18 (Dense)	(None, 1000)	1001000
dense_19 (Dense)	(None, 512)	512512
dense_20 (Dense)	(None, 128)	65664
dense_21 (Dense)	(None, 7)	903
activation_1 (Activation)	(None, 7)	0

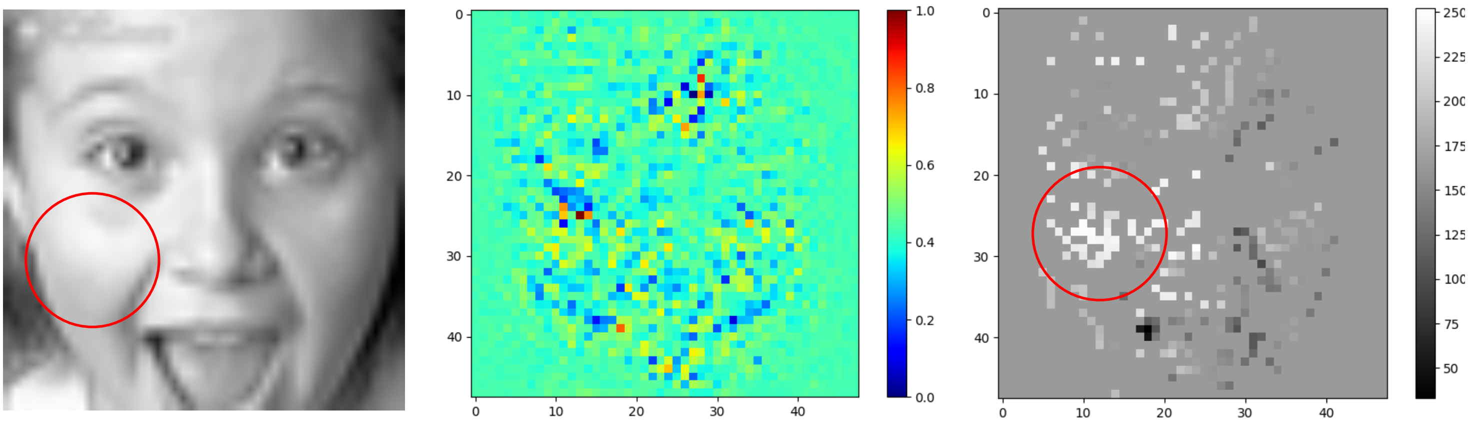
Total params: 38,211,079
Trainable params: 38,211,079
Non-trainable params: 0



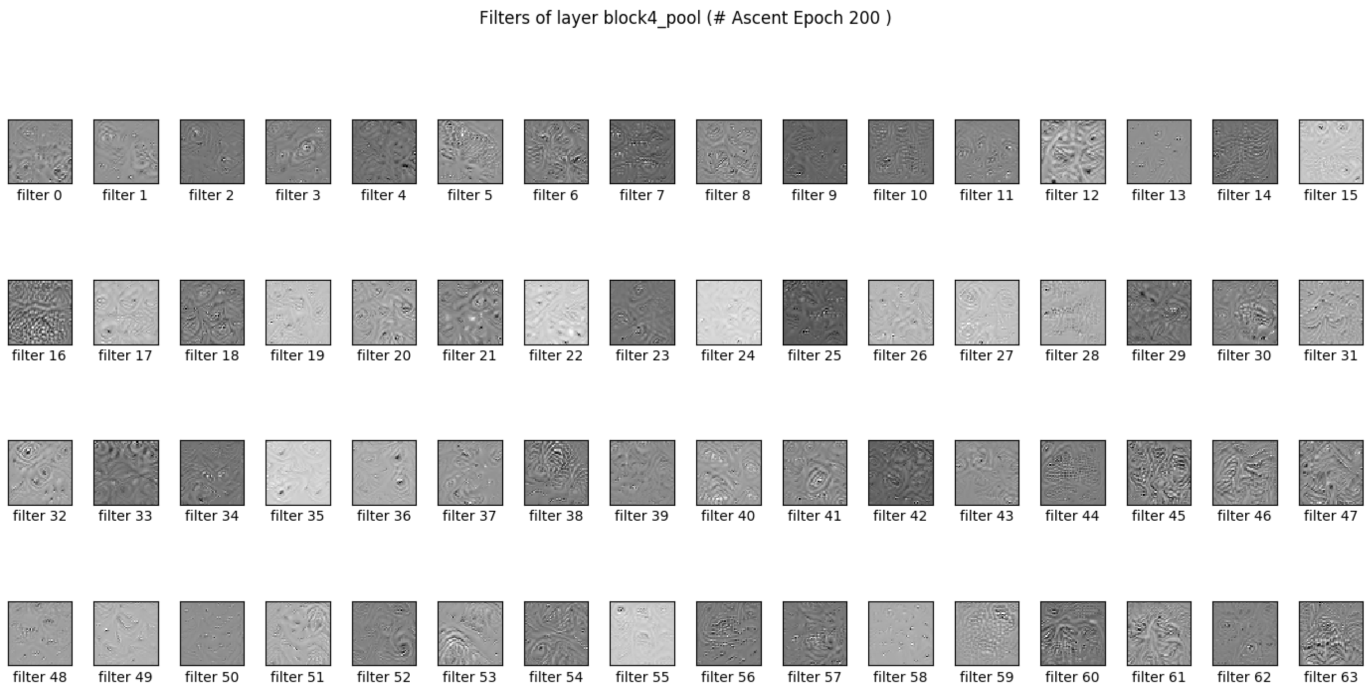
3.
20 fold 的 validation data 所畫出來的 confusion matrix 如上圖, Angry, Disgust 以及 Sad 容易跟 Neutral 混淆, 或許是因為這幾個表情都較為嚴肅, Fear 容易被判為 Sad 或 Surprise 則可能是因為害怕通常伴隨著驚訝或是難過。

其實在 train 這 dataset 的過程有做了一些 survey, 發現許多結論都是人情緒大多時候是綜合的, 也就是為什麼要 classify 到單一情緒有時候是較為困難的, 而這也跟自己實際下去做的結果蠻吻合的。

4.
依據 neural network 的 gradient 把所有沒超過 0.5 的 pixel 改成 image mean 所畫出來的結果, 不過跟我想像中可能會 focus 在眼睛嘴巴的情形似乎相差甚遠, 不過可能的解釋是因為 Happy 常常引發笑, 所以會 focus 在臉頰的部分。



5.
對第四次 pooing layer 做 gradient ascent 200 次得到的結果, 但卻還沒有明顯的人臉樣。而第二層 pooling layer 的結果(下圖)卻有明顯的人臉輪廓, 猜想可能跟從 random noise 進行 gradient ascent 有關。



過完第二次 pooling layer 得到的 feature map 畫出來的結果, 可以看到很多像是人的臉, 且有兩塊像眼睛以及一塊像嘴巴的白色區域。