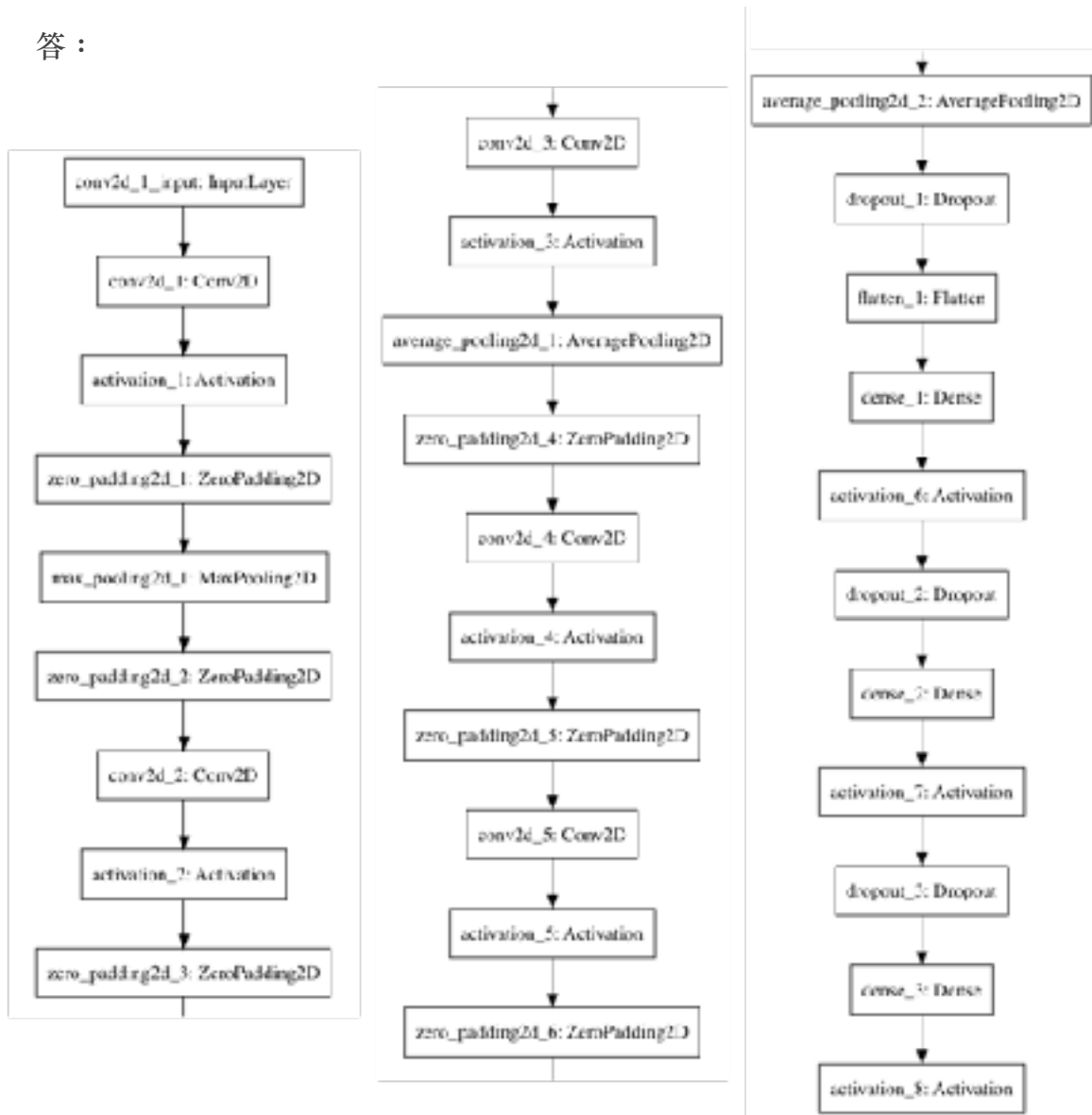


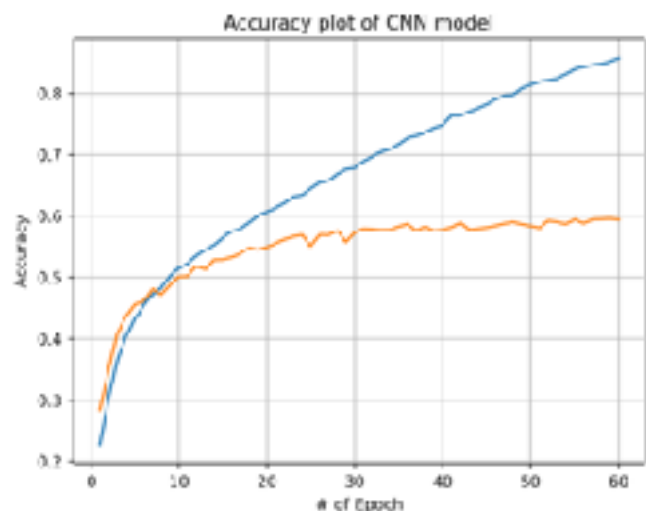
1. (1%) 請說明你實作的 CNN model，其模型架構、訓練過程和準確率為何？

答：



照助教的sample code，疊一層64個5*5的filters的CNN，第二層和第三層是64個3*3的filters，第四層和第五層是128個3*3的filters，最後再疊兩層1024個neural的fully connected network。中間也是照助教的sample code做ZeroPadding、Maxpooling、AveragePooling、Dropout。

在training的過程中accuracy不斷上升，validation set則漸趨0.6。在kaggle public testing set上的accuracy為**0.61186**。



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？

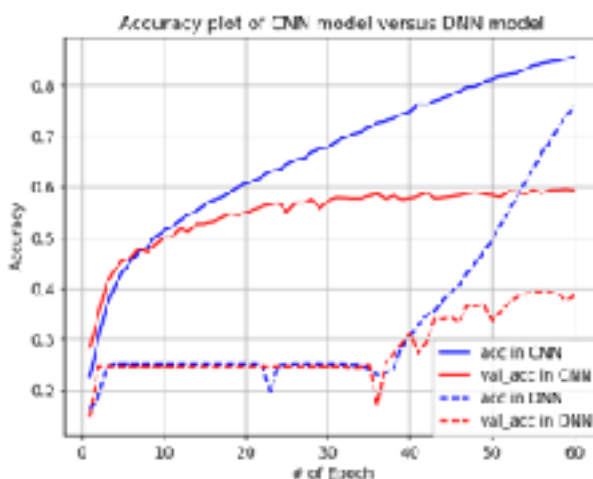
答：DNN中的input為 $48 \times 48 = 2304$ 維的陣列，每層的output設為filter邊長的平方*filter數，如64個 5×5 的filters，則output為 $64 \times 5 \times 5 = 1600$ ，則計算出來的param # = $(2304+1) \times 1600 = 3688000$ ，以此類推。

DNN model summary

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1000)	1000000
activation_1 (Activation)	(None, 1000)	0
dense_2 (Dense)	(None, 500)	501100
activation_2 (Activation)	(None, 500)	0
dense_3 (Dense)	(None, 100)	101100
activation_3 (Activation)	(None, 100)	0
dense_4 (Dense)	(None, 10)	1010
activation_4 (Activation)	(None, 10)	0
dense_5 (Dense)	(None, 10)	1010
activation_5 (Activation)	(None, 10)	0
dense_6 (Dense)	(None, 10)	1010
activation_6 (Activation)	(None, 10)	0
dense_7 (Dense)	(None, 10)	1010
activation_7 (Activation)	(None, 10)	0
dense_8 (Dense)	(None, 1)	10
activation_8 (Activation)	(None, 1)	0
Total params: 1,612,100		
Trainable params: 1,612,100		
Non-trainable params: 0		

CNN model summary (上題)

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 48, 48, 64)	1664
activation_1 (Activation)	(None, 48, 48, 64)	0
zero_padding2d_1 (ZeroPadding2D)	(None, 50, 50, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 24, 24, 64)	0
zero_padding2d_2 (ZeroPadding2D)	(None, 26, 26, 64)	0
conv2d_2 (Conv2D)	(None, 26, 26, 64)	16912
activation_2 (Activation)	(None, 26, 26, 64)	0
zero_padding2d_3 (ZeroPadding2D)	(None, 28, 28, 64)	0
conv2d_3 (Conv2D)	(None, 28, 28, 64)	16912
activation_3 (Activation)	(None, 28, 28, 64)	0
average_pooling2d_1 (AveragePooling2D)	(None, 14, 14, 64)	0
zero_padding2d_4 (ZeroPadding2D)	(None, 16, 16, 64)	0
conv2d_4 (Conv2D)	(None, 16, 16, 128)	73856
activation_4 (Activation)	(None, 16, 16, 128)	0
zero_padding2d_5 (ZeroPadding2D)	(None, 18, 18, 128)	0
conv2d_5 (Conv2D)	(None, 18, 18, 128)	24704
activation_5 (Activation)	(None, 18, 18, 128)	0
zero_padding2d_6 (ZeroPadding2D)	(None, 20, 20, 128)	0
average_pooling2d_2 (AveragePooling2D)	(None, 10, 10, 128)	0
dropout_1 (Dropout)	(None, 10, 10, 128)	0
flatten_1 (Flatten)	(None, 10240)	0
dense_1 (Dense)	(None, 1024)	10617856
activation_6 (Activation)	(None, 1024)	0
dropout_2 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 1024)	10617856
activation_7 (Activation)	(None, 1024)	0
dropout_3 (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 1)	10
activation_8 (Activation)	(None, 1)	0
Total params: 11,971,504		
Trainable params: 11,971,504		
Non-trainable params: 0		

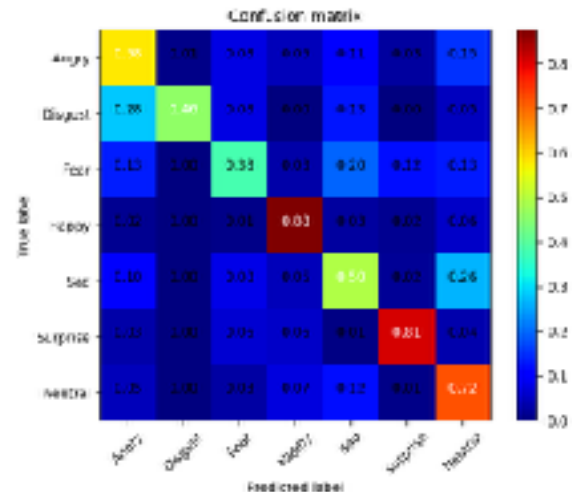


使用相似參數量及相同的層數，此 DNN模型在kaggle public testing set上的 accuracy是**0.38506**，且在訓練時，testing set和validation set的accuracy都沒有CNN的 accuracy高。可推測CNN在偵測圖片的特徵上，效果比DNN沒有考慮圖片特徵的相對位置的情況好，所以CNN比DNN適合用在圖片特徵辨識上。

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：

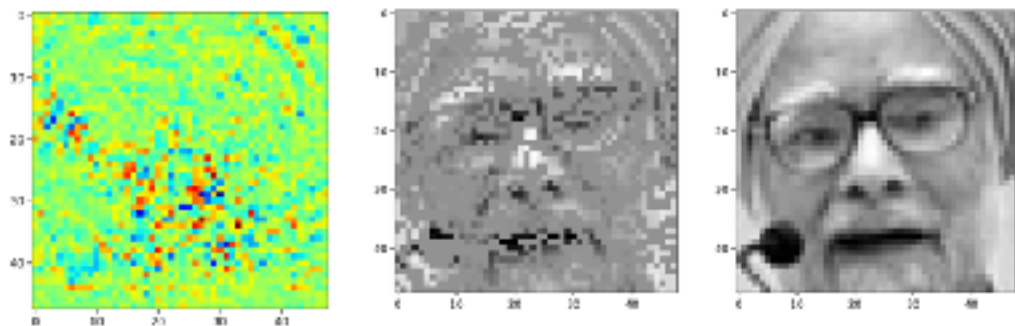
觀察對角線以外的格子，Disgust 會容易被誤認成Angry，Sad容易被誤認為Neural，其次是Fear容易被誤認為Sad。在現實情況中這些表情確實較難被分辨出來。



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：



根據此張圖(training data #1)，約focus在眼睛、嘴巴(深色)的部分，較容易被凸顯出來，另外其他的作圖如果臉的部分不夠多的話，經過saliencymap會看不出來filter選擇的區域跟表情的關係。

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: <https://gist.github.com/howard1337/f416e5aebd98fbbbdd43a137009757e>)

答：

此圖為training set #1，label為0(Angry)，第一排第一個、第二排第一個和第六個和第十個、第三排第七個和第十三個、第四排第二和第四個和八個filter過後臉型較明顯，因此這些filter可能就是分辨Angry的重要filter。

