學號：B03705027 系級：資管三 姓名：鄭從德

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？

下圖是我最佳的的 CNN model 結構：

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 48, 48)        320
_________________________________________________________________
dropout_1 (Dropout)          (None, 32, 48, 48)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 48, 48)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 24, 24)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64, 24, 24)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 24, 24)        36928
_________________________________________________________________
dropout_3 (Dropout)          (None, 64, 24, 24)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 128, 24, 24)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 128, 12, 12)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 128, 12, 12)       147584
_________________________________________________________________
dropout_4 (Dropout)          (None, 128, 12, 12)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 256, 12, 12)       295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 256, 6, 6)         0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 256, 6, 6)         590080
_________________________________________________________________
dropout_5 (Dropout)          (None, 256, 6, 6)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9216)              0
_________________________________________________________________
dropout_6 (Dropout)          (None, 9216)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              9438208
_________________________________________________________________
dropout_7 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 7175
=================================================================
Total params: 10,607,815
Trainable params: 10,607,815
```

我一共用了七層的 convolution 2D，kernel 大小皆為(2,2)、三次的 Max Pooling 也都是(2, 2)。Optimizer 是採用 Adam，learning rate 設為 0.00025。我將 99%的 data 作為 training data，1% 作為 validation set (因為 training data 越多會越準，所以我最生成最佳的 model 時都是採用幾乎全部的 data 來 train)

關於資料的處理部份，我有手做 histogram equalization，但是效果並不顯著。除此之外我還有用 keras 內建的 image generator，讓每次的 training data 稍有差異，對於準確度有很大的幫助。

最後進行了 400 個 epoch 的訓練，最後準確率約為 70%

```
Epoch 395/400
55/55 [==============================] - 19s - loss: 0.4869 - acc: 0.8185 - val_loss: 1.0311 - val_acc: 0.6910
Epoch 396/400
55/55 [==============================] - 19s - loss: 0.4835 - acc: 0.8180 - val_loss: 1.0556 - val_acc: 0.6771
Epoch 397/400
55/55 [==============================] - 19s - loss: 0.4836 - acc: 0.8199 - val_loss: 1.0574 - val_acc: 0.7083
Epoch 398/400
55/55 [==============================] - 19s - loss: 0.4828 - acc: 0.8178 - val_loss: 1.1015 - val_acc: 0.6979
Epoch 399/400
55/55 [==============================] - 19s - loss: 0.4836 - acc: 0.8194 - val_loss: 1.0826 - val_acc: 0.6979
Epoch 400/400
55/55 [==============================] - 19s - loss: 0.4829 - acc: 0.8210 - val_loss: 1.0592 - val_acc: 0.6910
Using Train data.
Train on 28421 samples, validate on 288 samples
Large CNN Error: 30.90%
```

2. 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？

我也做了一個七層的 DNN，參數接近一千萬，但準確率最高仍只有 42%而已，而且有嚴重的 overfitting 現象。這是因為 DNN 沒有辦法辨別兩個點之間合在一起的關係，更不用說圖像了。所以這樣硬 train 的結果只會讓 program 學到一些奇怪的原則，沒有真正的意義。

```
Epoch 195/200
22967/22967 [==============================] - 11s - loss: 0.1332 - acc: 0.9568 - val_loss: 5.0459 - val_acc: 0.4269
Epoch 196/200
22967/22967 [==============================] - 11s - loss: 0.0817 - acc: 0.9730 - val_loss: 5.0835 - val_acc: 0.4222
Epoch 197/200
22967/22967 [==============================] - 11s - loss: 0.0625 - acc: 0.9787 - val_loss: 5.3459 - val_acc: 0.4312
Epoch 198/200
22967/22967 [==============================] - 12s - loss: 0.0576 - acc: 0.9807 - val_loss: 5.2611 - val_acc: 0.4222
Epoch 199/200
22967/22967 [==============================] - 11s - loss: 0.1346 - acc: 0.9569 - val_loss: 4.7281 - val_acc: 0.4162
Epoch 200/200
22967/22967 [==============================] - 11s - loss: 0.1157 - acc: 0.9626 - val_loss: 5.0573 - val_acc: 0.4276
```

3. 答錯的圖片中，哪些 class 彼此間容易用混？[繪出 confusion matrix 分析]

七個 class 的 confusion matrix 如右。

```
[[22  0  2  3  7  0  5]
 [ 1  2  0  0  1  0  0]
 [ 4  0 26  1 10  5  0]
 [ 1  0  0 60  3  1  1]
 [ 4  0  6  1 23  2  9]
 [ 0  0  3  1  0 22  3]
 [ 2  0  1  5  7  0 44]]
```

生氣(0)最容易被誤判為難過(4)、

恐懼(2)也容易被誤判為難過(4)、

難過(4)最容易被判斷為中立(6)、

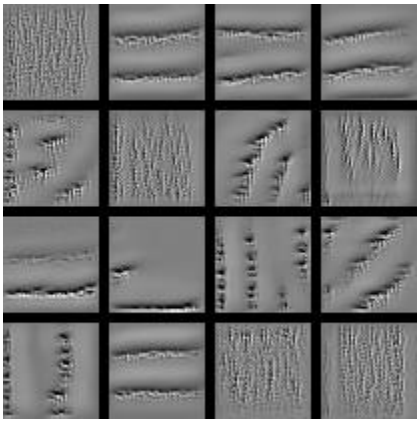
中立(6)最容易辦判斷為難過(4)。

最容易被彼此混用的是難過與中立

4. 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？

繪出 saliency map 可以發現，我的 CNN model 在處理人臉影像時會 focus 在人臉的下半部，也就是嘴巴、臉頰、以及臉頰上的紋路等等。我本來以為眉毛和眼睛也是一個很重要的判斷依據，但實作的結果卻不是如此。反倒是眉毛之間的位置也有被當作一個重要的判斷線索。

5. 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。



右圖是我 visualize 我第六層的 convolutional 2D (conv2d_6)得出來，前 100 個 filter 中做 200 個 epoch 的 gradient ascent，得到最佳的 16 個 filter。其中儘管有一些 pattern，仍然很難看出這跟人臉有什麼關聯。我覺得是因為這次要分別的七種情緒，是屬於比較細部的分類，我們只能從這些 filter 中看出，他們 focus 在大概多大的單位上。把這張圖跟前一題一起看可以猜測，我的 model 可能會 focus 在比較大的臉部區域，像是臉頰、額頭、鼻子等等。嘴巴應該要是一個極重要的分類依據，但從這一層的 filter 中真的看不出來誰跟嘴巴比較有關係。

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

Data 分配： Training Data: 20%、Unlabeled Data: 60%、Testing Data: 20%
訓練方式：先將 training data 丟入與之前一樣架構的 model 中做 100 次 epoch 的訓練，接著在用結果預測 60%的 unlabeled data 。預測完再一起加入 Training data，再進行 100 次訓練。

在我加入 unlabeled data 之前，準確率僅僅 53.2%，加入之後整體的準確率上升到 58.5%。雖然總體的表現還是沒有我之前拿超過 90%的 data 拿來 train 好，但是比起沒有 unlabeled 的時候已經大幅上升，而且考慮到這時候的 labeled data 只有 20%，已經進步相當多。

```
Epoch 95/100
89/89 [==============================] - 37s - loss: 1.0473 - acc: 0.7669 - val_loss: 1.1231 - val_acc: 0.5853
Epoch 96/100
89/89 [==============================] - 37s - loss: 1.0488 - acc: 0.7650 - val_loss: 1.1278 - val_acc: 0.5824
Epoch 97/100
89/89 [==============================] - 37s - loss: 1.0493 - acc: 0.7652 - val_loss: 1.1379 - val_acc: 0.5799
Epoch 98/100
89/89 [==============================] - 37s - loss: 1.0466 - acc: 0.7665 - val_loss: 1.1181 - val_acc: 0.5867
Epoch 99/100
89/89 [==============================] - 37s - loss: 1.0491 - acc: 0.7622 - val_loss: 1.1324 - val_acc: 0.5817
Epoch 100/100
89/89 [==============================] - 37s - loss: 1.0422 - acc: 0.7676 - val_loss: 1.1296 - val_acc: 0.5848
Large CNN Error: 41.52%
```

接著我做的是把兩步驟的 epoch 都提升到 200 次。加入 unlabeled data 之前結果進步到 57.1%，而加入 unlabeled data 之後進步到 60.2%，這大概就是我這個 model

對於少量 training data 的極限了。由此可知加入 unlabeled data 對於準確率的提升很有幫助！(在 training data 很少的情況下)