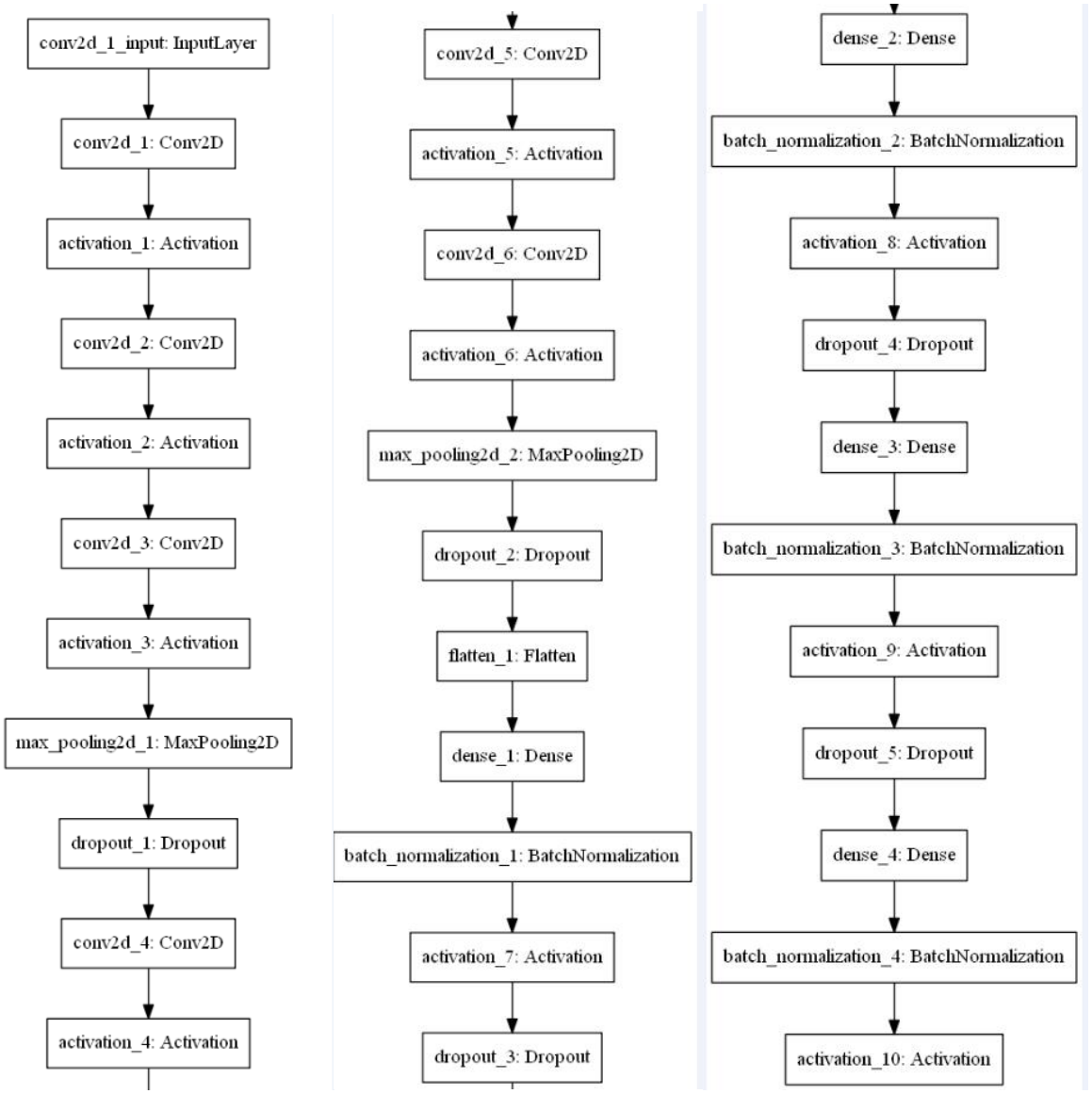
學號：B03505027 系級： 電機三 姓名：劉亦浚

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：

Model Structure (由左至右連接)



Training Procedure

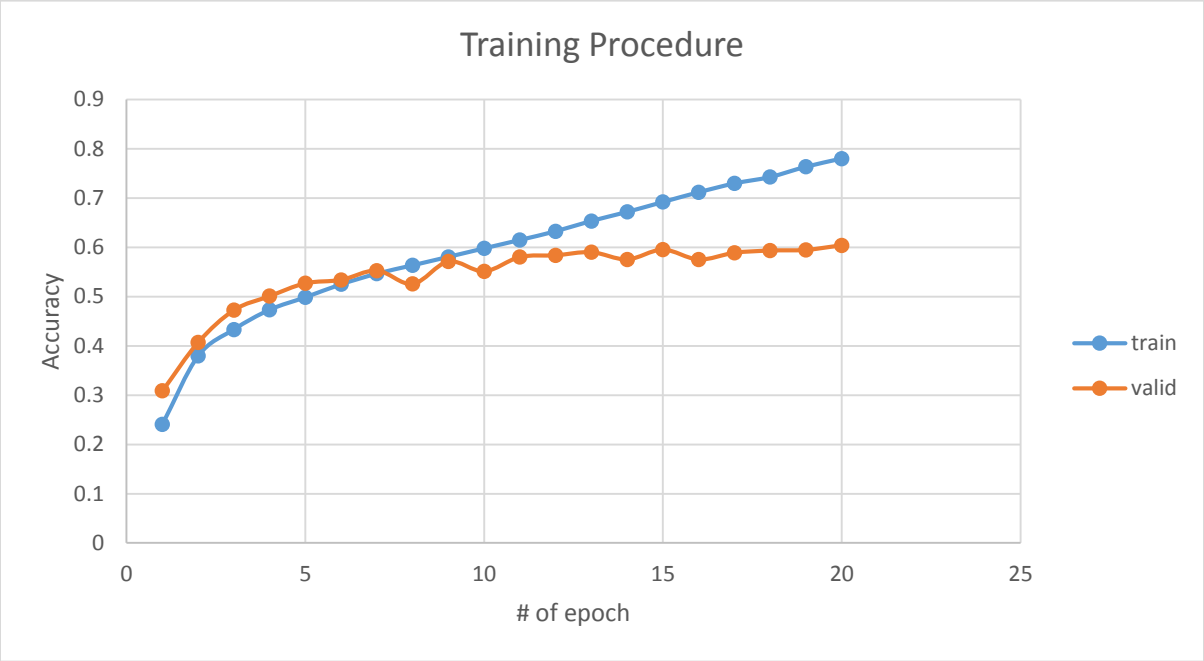


Image Preprocessing

最後有採用照片前處理的方法，以下為其中所設定的參數：

```
#augmentation for training data
if data_augmentation == True:
    print("data augmentation")
    datagen = ImageDataGenerator(
            featurewise_center=False,  # set input mean to 0 over the dataset
            samplewise_center=False,  # set each sample mean to 0
            featurewise_std_normalization=False,  # divide inputs by std of the dataset
            samplewise_std_normalization=False,  # devide each input by its std
            zca_whitening=False,  # apply ZCA whitening
            rotation_range=7,  # randomly rotate images in range
            width_shift_range=0.1,  # randomly shift images horizontally
            height_shift_range=0.1,  # randomly shift images vertically
            horizontal_flip=True,  # randomly flip images
            vertical_flip=False)  # randomly flip images
    datagen.fit(x_train)
```

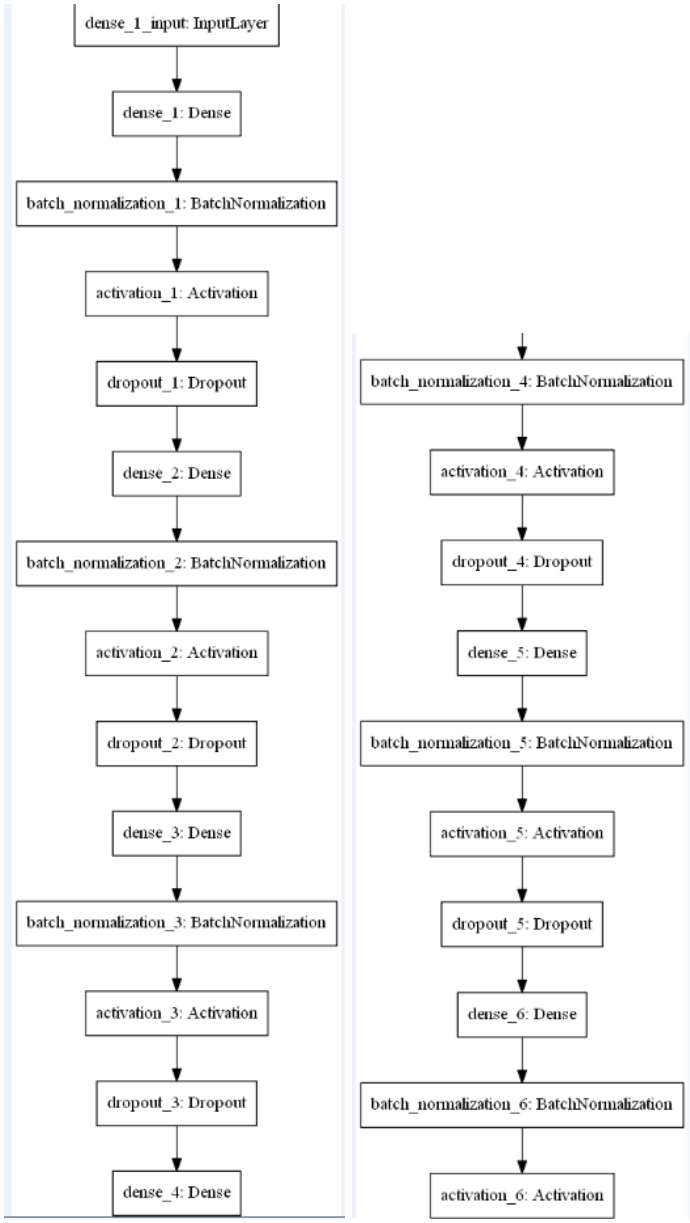
使用照片前處理，可以避免 model overfitting 的狀況，且提高 testing 的準確率，最後在 kaggle 上的分數為 0.69713，其結果比沒有使用照片前處理可高出約 5%的準確率。

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？

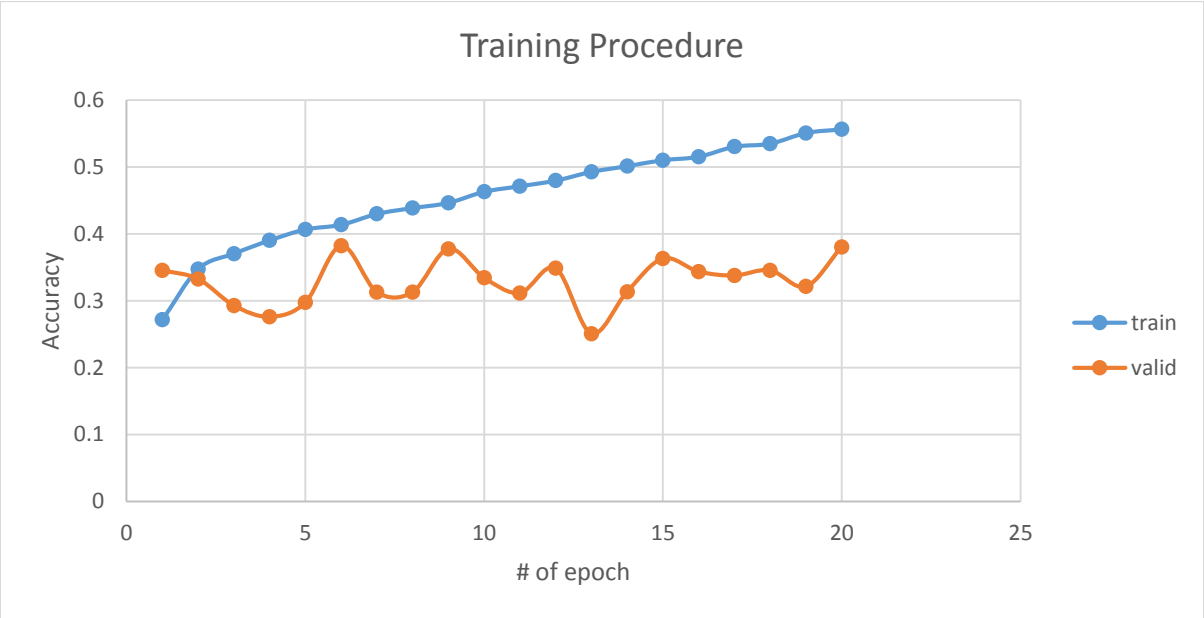
答：

```
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 7)                 3591
_________________________________________________________________
batch_normalization_6 (Batch (None, 7)                 28
_________________________________________________________________
activation_6 (Activation)    (None, 7)                 0
=================================================================
Total params: 2,244,643.0
Trainable params: 2,239,509.0
Non-trainable params: 5,134.0
_________________________________________________________________
```

Model Structure (由左至右連接)



Training Procedure

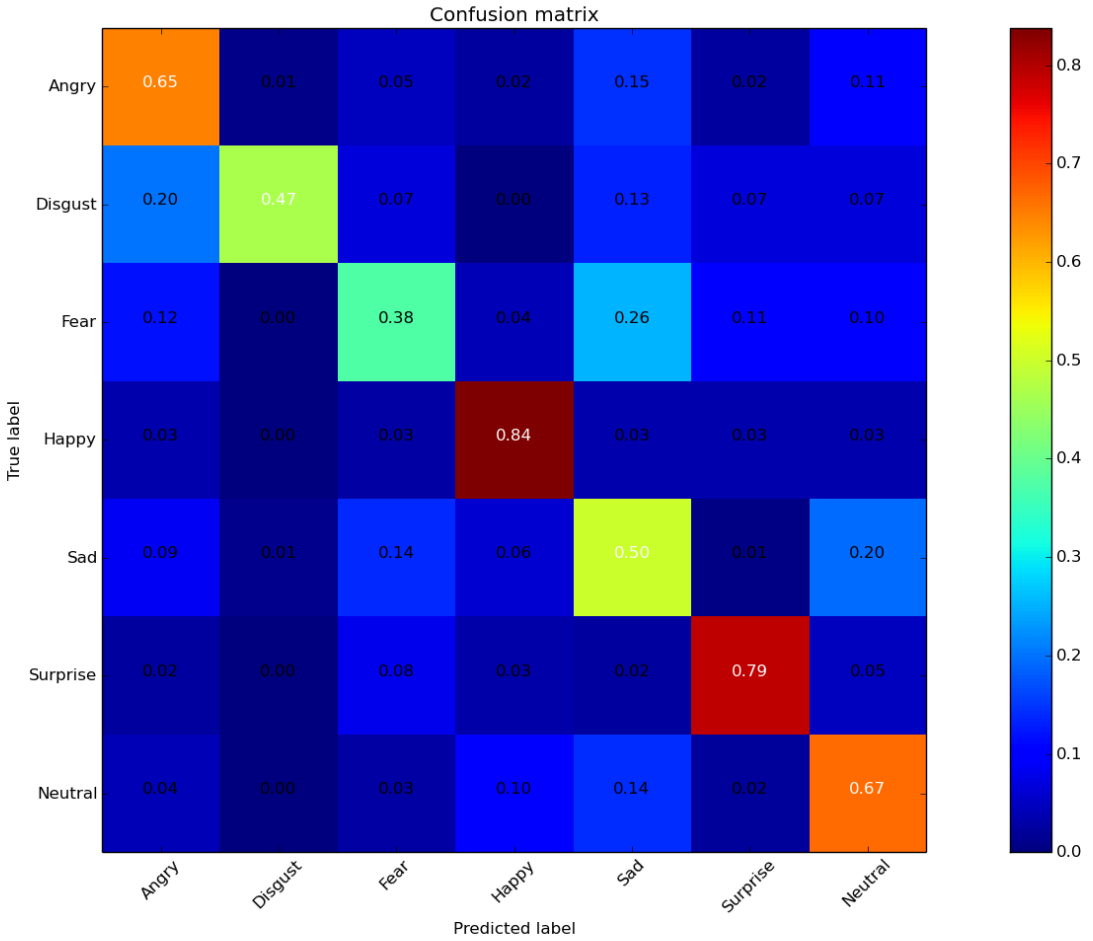


Compare DNN with CNN

從 Training procedure 可清楚看出，DNN 在 valid 上的準確率十分不穩定，且整體看起來沒有上升的趨勢；而在 CNN 中 valid 的準確率較為穩定，且有逐漸上升的趨勢。其原因應為 DNN 是把整張照片丟進去 train，沒有照片局部特點的概念，不像 CNN 透過 filter 取出照片中的特點在丟進去 train，可使 model 往後看到相同特點的照片時，會給予較符合的 label，因此 DNN 的預測能力相當不好。

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]

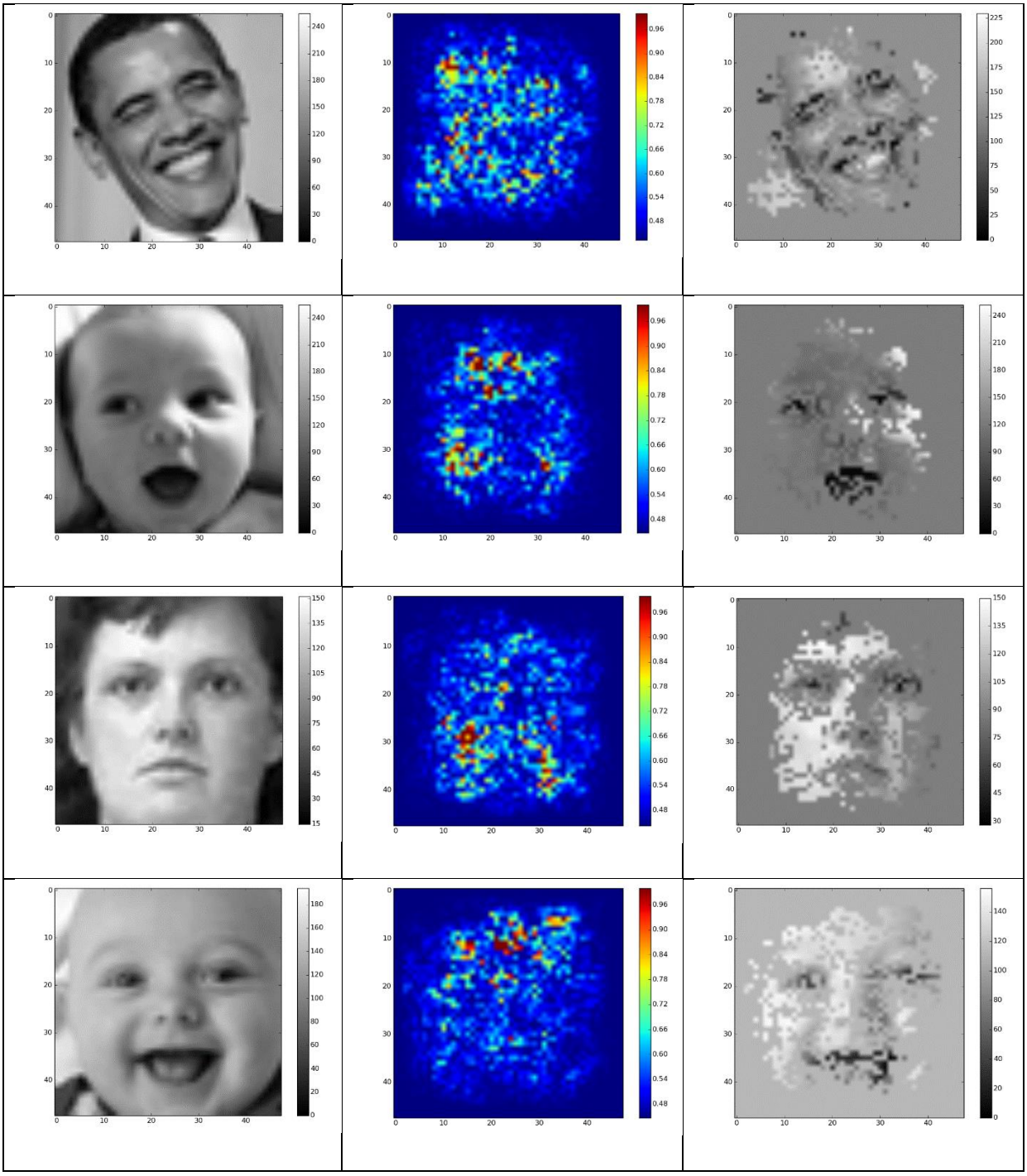
答：



在 angry 中，準確率還可；在 disgust 中，容易與 angry 或 sad 搞混；在 fear 中，容易與 sad 搞混；在 happy 中，準確率頗高；在 sad 中，容易與 fear 或 neutral 搞混；在 surprise 中，準確率頗高；在 neutral 中，準確率還可。

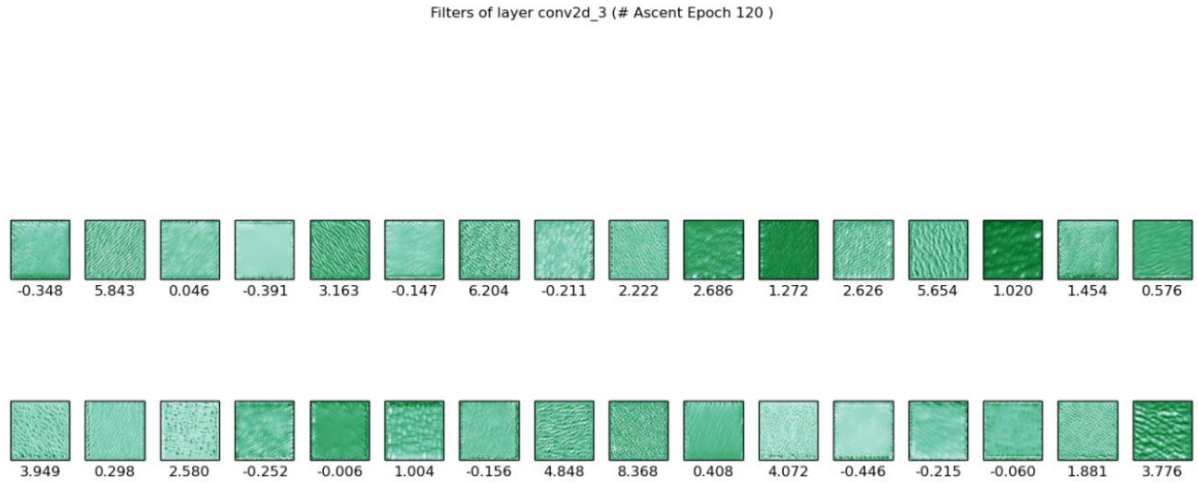
4. (1%) 從(1)(2)可以發現，使用 CNN 的確有些好處，試繪出其 saliency maps，觀察模型在做 classification 時，是 focus 在圖片的哪些部份？

答：

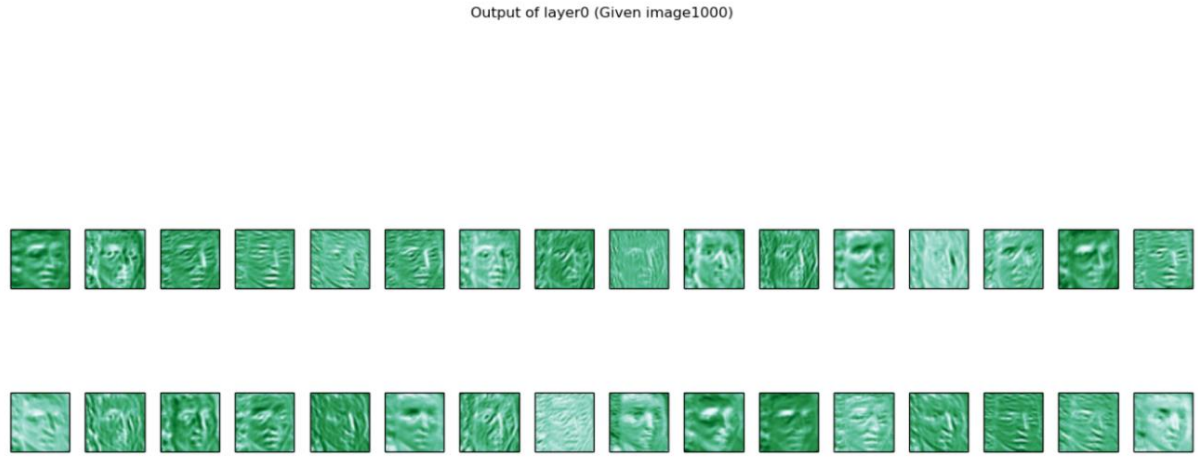


挑選了四張照片做為結果，經過 saliency map 處理後，發現 model 通常都 focus 在人臉的五官，且以眼睛、嘴巴最為明顯。所以機器在預估的時候，因以此特徵做為分類依據。

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：



觀察第三層 convolution 的 filter (數量為 32)，發現容易被斜線條、霧狀、格子狀以及點狀圖形等 activate。



此為第 1000 張照片在第三層 convolution 的 output，發現許多張照片有上面 filter 的痕跡，如斜線、霧狀、格子狀及點狀等。有相同痕跡的圖片，應為其 filter 的確容易受到此圖形而 activate，我們也可從這些 filter 了解 model 在這層是以哪些特點來觀察圖片的。

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

我是採取把 testing data 當成 unlabeled data，來實作 semi-supervised learning。

```
#for self learning
for i in range(iteration):
    if data_augmentation == True:
        datagen.fit(x_train)
        model.fit_generator(datagen.flow(x_train, y_train, batch_size=batch_size),
                            steps_per_epoch=(x_train.shape[0]//batch_size),
                            epochs=epochs)
    else:
        model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs)

    pre_label = model.predict(x_test)
    tmp_data = []
    tmp_label = []
    delete_index = []
    for j in range(len(pre_label)):
        for k in range(7):
            if pre_label[j][k] > bound:
                tmp_data.append(x_test[j])
                tmp_label.append(k)
                delete_index.append(j)
    tmp_data = np.array(tmp_data)
    tmp_label = np.array(tmp_label)
    delete_index = np.array(delete_index)
    tmp_label = np_utils.to_categorical(tmp_label, num_classes)
    x_train = np.concatenate((x_train, tmp_data), axis=0)
    y_train = np.concatenate((y_train, tmp_label), axis=0)
    x_test = np.delete(x_test, delete_index, axis=0)
    print(x_train.shape)
    print(y_train.shape)
    print(x_test.shape)
```

把 testing data 丟進訓練好的 model，並觀察照片每個分類的機率，只要其中有大於 0.9 的話，即將此照片 label 標為此類，並丟到 training data 中，全部檢查完後，將新的 training data 丟進 model 繼續 train，重複此動作 10 次。最後也得到約 69%的準確率。