【智慧化企業整合】Project #3

看臉時代

以 CNN 對人臉進行顏值評分

110030517 胡詠晴

指導教授：邱銘傳 教授

目錄

一、背景介紹

前言

美女的定義到底是什麼，從古至今說法不一，並沒有一個統一的標準，古代美女標準及現代美女標準透過網路搜索將其分為以下幾點：

中國古代的美女標準

1、膚若凝脂：皮膚光滑、細膩而潔白，如凝固的脂肪。
2、明眸善睞：明潔靈動、富有神韻的眼睛。
3、雲發豐艷：頭髮烏黑、亮麗、濃密、修長。
4、蛾眉青黛：眉毛細長、彎曲、淡青色。
5、杏臉桃腮：面容艷光照人、白裡透紅。
6、櫻唇貝齒：嘴唇色澤紅潤而小巧。
7、楊柳細腰：腰肢纖、柔。
8、纖纖素手：手指白嫩細膩、修長靈巧。
9、三寸金蓮：雙足纖小、弓彎、白淨。
10、軟玉溫香：身體嬌柔、白膩、芳香。

現代各國美女審美標準

中國標準美女臉：臉頰較爲狹窄，臉型纖瘦，下巴突出。
韓國標準美女臉：相對中國美女臉頰更圓，下巴稍微兜着，眼神溫和，很溫柔端莊的感覺。
日本標準美女臉：臉型相對較長，眼睛略微斜挑，下巴尖，臉頰豐滿。
白人標準美女臉：臉有幾分陽剛氣質，眼皮邊緣與眉毛之間的距離狹窄，下顎方方正正、有棱有角，臉頰突出，比起長相普通的白人女性有着更爲豐滿的雙脣。
非洲標準美女臉：非洲美女臉鼻樑較窄，眼睛更小更銳利，上嘴脣較小，與普通的非洲女性相比下巴更爲纖細。

文字敘述上雖能大概勾勒出各國及古代美人的樣貌，但是卻沒辦法用具體的分數去量化顏值這個部分，這也是亙古至今一直所缺乏的！

研究動機

在“外貌協會”盛行的風氣下，人們對外表的經營和保養越來越重視。這現象也反映在許多電視劇和漫畫情節之中，如“轉學來的女生”其中也有一集針對一女校如何使用即時評估系統，給女學生的顏值進行排名決定其使用校園資源的權益。在看臉時代漫畫中一開始男主也是因為顏值低等問題被霸凌，直到獲得新的高顏值身體人生才得到重生。然而，審美是非常主觀的，因人而異，並沒有一個確切的標準。

研究目的

藉由 CNN 對個人顏值進行客觀評分,讓顏值能夠被量化，使顏值有更明確的標準，亦能提供選美比賽一個參考標準！

5W1H

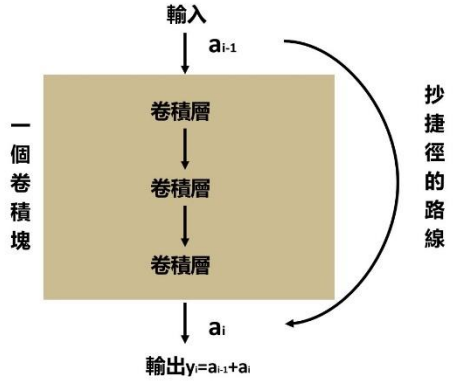
When	舉辦選美比賽時，或是想知道自己臉部顏值在哪一個等級時
Where	選美比賽現場等
Who	選美選手、選美評審、想知道自己顏值分數的人
What	選美選手到現場須經由評審評分，評審對美醜的優劣判斷不一
Why	評審評分太過主觀意見，藉由 CNN 能更客觀描述選美比賽選手的顏值
How	現場拍攝個人正面照片後，以 CNN 對個人顏值進行客觀評分

方法介紹

ResNets

殘差神經網路(簡稱 Resnets)是指使用殘差模組(residual module)，或稱殘差塊的架構，而殘差模組則是以「卷積塊」搭配跳接(skip connection)設計而成的架構。下圖為一個殘差模組的示意圖，中間方框為卷積塊裡面通常含多個卷積層設計(Ex.批次正規化層或丟棄層等)。

而殘差模組的特色就是加入了跳接(skip connection)設計，也就是下方抄捷徑的箭頭，跳接的作用是將準備傳入卷積塊的輸入(做卷積運算之前)「跨層」傳遞，再與經過卷積運算後的輸出(做卷積運算之後)，兩者直接「相加」後，作為最終的輸出，再進行後續的推理(通常是送入激活函數)。



當殘差模組接收到某輸入 a_{i-1}(神經層接收到的輸入都是前層的輸出，在此表示為 a_{i-1})，除了輸入卷積塊運算產生當層的輸出 a_i 之外，此 a_i 還要與殘差模組的輸出端的 a_{i-1} 輸入相加，成為殘差模組的輸出 y_i。

即：

$$y_i=a_{i-1}+a_i$$

抄捷徑從前一層傳來的　當層算出來的

根據上一段描述的殘差連接原理會發現如果 $a_i=0$，則輸出值 y_i 與原輸入無異，殘差連接在此時的作用如同「恆等函數」，所以當神經層以類似卷積塊做運算時即使無法學到有用的知識($a_i=0$)來減少神經網路損失至少能夠不礙事。而除了不礙事的優點，只要將數個殘差模組(卷積塊+跳接設計)堆疊在一起，就可以設計出多種變化。

ResNet50

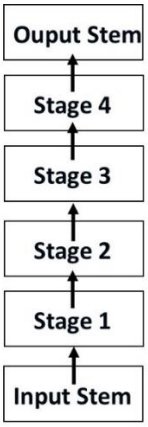
本次研究以 Resnet 50 作為建模之架構，ResNet 於 2015 年被提出，在 ImageNet 比賽 classification 任務上獲得第一名，因爲它“簡單與實用”並存，之後很多方法都建立在 ResNet50 或者 ResNet101 的基礎上完成的，檢測，分割，識別等領域都紛紛使用 ResNet。

Resnet 50 整體架構可以分為三大部分

1.Input stem: 使用一般的 convolution，並且用大的 stride 降低解析度。

2.Stage block: ResNet 共有 4 個 Stage block，每個 stage block 都是由數個 building block 堆疊而成。不論是用 stride 或是 pooling，每個 stage 一般都會先降低解析度並加大寬度 (channel)，再做一連串的 residual learning。

3.Output stem：依照任務，設計不同的輸出。



而不同深度的 ResNet 的差別有兩個：

ResNet18、ResNet34 使用一般的 residual block，而 ResNet50、ResNet101、ResNet152 使用了 expansion 為 4 的 bottleneck block。

剩下的差異就在於每個 stage 堆疊的 building block 層數不同，詳細差異可以參考下面的表格。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Flask

本次研究以 Flask 為基礎搭建網頁，Flask 是一個使用 Python 撰寫的輕量級 Web 應用程式框架，由於其輕量特性，也稱為 micro-framework（微框架）。Flask 核心十分簡單，主要是由 Werkzeug WSGI 工具箱和 Jinja2 模板引擎所組成，Flask 和 Django 不同的地方在於 Flask 給予開發者非常大的彈性，可以選用不同的擴充 extension 來增加其功能。相比之下，Django 雖然完善但技術選擇相對不彈性，不論是 ORM、表單驗證或是模版引擎都有自己的作法。世界上沒有最好的框架，只有合適的使用情境，Django 相比之下適合需要快速的開發大型的應用程式，在本例中較為不適用。

資料集描述

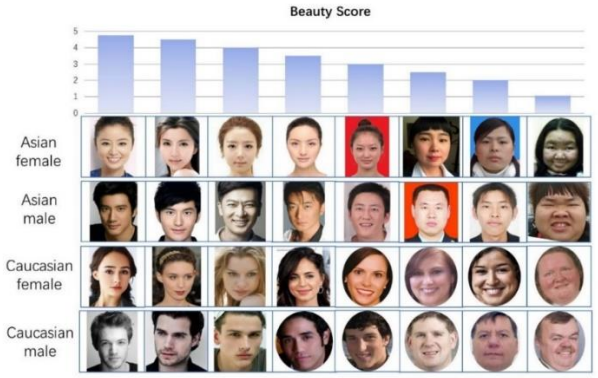
本研究採用華南理工大學所發布之 SCUT-FBP5500 資料集

1.說明

SCUT-FBP5500 資料集共有 5500 張具有不同屬性（男性/女性、亞洲/白種人、年齡）和不同標籤（Facial Landmark 人臉標記、5 個尺度的美容評分）的正面人臉，例如亞洲/白種人、男性/女性的基於外觀/形狀的面部美容分類/回歸/排名模型

2.資料集建設

SCUT-FBP5500 資料集可以分為四個不同種族和性別的子集，包括 2000 名亞洲女性（AF）、2000 名亞洲男性（AM）、750 名高加索女性（CF）和 750 名高加索男性（CM）。SCUT-FBP5500 的大部分圖片來自互聯網，其中部分亞洲人臉來自 DataTang、GuangzhouXiangSu 和華南理工大學(人機智能交互實驗室)，部分白人人臉來自 10k 美國成人人臉資料集。



所有圖像都由總共 60 名志願者用 [1, 5] 範圍內的美感評分標記，每張人臉標記有 86 個特徵點。

二、資料前處理

下載 SCUT-FBP5500 資料集解壓後可看到，圖片均存放於 Images 目錄中，評分結果在 All_Ratings.xlsx，All_Ratings.xlsx 資料集中為 60 個人對每一張照片的評分，因此每張圖片有 60 個評分，我們需要將每一張照片取平均後作為照片的最終得分：

載入資料集(Rater：評分者 Filename：圖片檔名)

```
[ ] ratings = pd.read_excel('C:/Users/user/1jupyter notebook example/facial_beauty_prediction-master/All_R
    ratings.head()
```

	Rater	Filename	Rating	original Rating
0	1	CF1.jpg	3	NaN
1	1	CF10.jpg	3	NaN
2	1	CF100.jpg	1	NaN
3	1	CF101.jpg	2	NaN
4	1	CF102.jpg	3	NaN

取平均(將每一張照片取平均獲得最終得分)

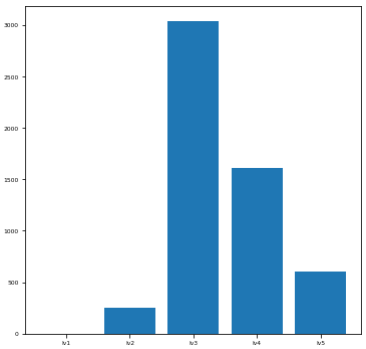
```
[ ] filenames = ratings.groupby('Filename').size().index.tolist()

    labels = []

    for filename in filenames:
        df = ratings[ratings['Filename'] == filename]
        count = Counter(df['Rating']).most_common(1)[0][0]
        score = round(df['Rating'].mean(), 2)
        labels.append({'Filename': filename, 'most_common': count, 'score': score})

    labels_df = pd.DataFrame(labels)
    labels_df.head()
```

	Filename	most_common	score
0	AF1.jpg	3	2.33
1	AF10.jpg	4	3.43
2	AF100.jpg	3	2.90
3	AF1000.jpg	4	3.97
4	AF1001.jpg	4	3.73

將所有數據分成五個 level，以長條圖繪出目前所有的照片之得分分布狀況

```
[ ] scores = sorted(labels_df.score.tolist())
```

```
lv1 = [x for x in scores if x<=1]
lv2 = [x for x in scores if x>1 and x<=2]
lv3 = [x for x in scores if x>2 and x<=3]
lv4 = [x for x in scores if x>3 and x<=4]
lv5 = [x for x in scores if x>4 and x<=5]

plt.bar(['lv1','lv2','lv3','lv4','lv5'],
            [len(x) for x in [lv1,lv2,lv3,lv4,lv5]])
```



可以看到有超過一半的評分都在 3~4 分的等級，兩側極端值人數都極為稀少。

將所有照片轉成 numpy.ndarray 格式，目標是這張照片的評分。

```
[ ] img_width, img_height, channels = 350, 350, 3
    sample_dir = 'D:/SCUT-FBP5500_v2/Images'
    nb_samples = len(os.listdir(sample_dir))
    input_shape = (img_width, img_height, channels)

    x_total = np.empty((nb_samples, img_width, img_height, channels), dtype=np.float32)
    y_total = np.empty((nb_samples, 1), dtype=np.float32)

    for i, fn in enumerate(os.listdir(sample_dir)):
        img = load_img('%s/%s' % (sample_dir, fn))
        x = img_to_array(img).reshape(img_height, img_width, channels)
        x = x.astype('float32') / 255.
        y = labels_df[labels_df.Filename == fn].score.values
        y = y.astype('float32')
        x_total[i] = x
        y_total[i] = y
```

將數據拆分成訓練集 2560 個，驗證集 640 個，測試集 800 個。

```
In [9]: seed = 42
        x_train_all, x_test, y_train_all, y_test = train_test_split(x_total, y_total, test_size=0.2, random_state=seed)
        x_train, x_val, y_train, y_val = train_test_split(x_train_all, y_train_all, test_size=0.2, random_state=seed)

In [10]: np.save('x_train.npy', x_train)
         np.save('y_train.npy', y_train)
         np.save('x_val.npy', x_val)
         np.save('y_val.npy', y_val)
         np.save('x_test.npy', x_test)
         np.save('y_test.npy', y_test)

In [11]: for item in [x_train, y_train, x_val, y_val, x_test, y_test]:
             print(item.shape)

         (2560, 350, 350, 3)
         (2560, 1)
         (640, 350, 350, 3)
         (640, 1)
         (800, 350, 350, 3)
         (800, 1)
```

三、模型建構

使用 Keras 內建的 ResNet50 架構進行訓練，去掉最後的 softmax 層，然後再加上一層 dense 全連接層，並將 ResNet50 模型的其餘參數設為不可訓練，使其先直接訓練最後的 dense 層參數。

```
[ ] img_width, img_height, channels = 350, 350, 3
    input_shape = (img_width, img_height, channels)
```

```
[ ] resnet = ResNet50(include_top=False, pooling='avg', input_shape=input_shape)
    model = Sequential()
    model.add(resnet)
    model.add(Dense(1))
    model.layers[0].trainable = False
    model.summary()
```

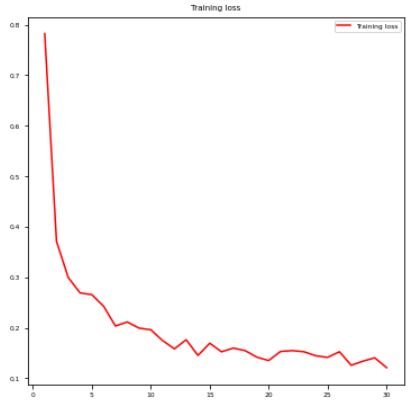
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 2048)              23587712
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 2049
=================================================================
Total params: 23,589,761
Trainable params: 2,049
Non-trainable params: 23,587,712
_________________________________________________________________
```

以 MSE(均方誤差)作為我們的損失函數為後期回歸分析做預備，並訓練 30 個 epoch

```
[ ] model.compile(loss='mse', optimizer='adam')
    history = model.fit(batch_size=32, x=x_train, y=y_train, epochs=30)
```

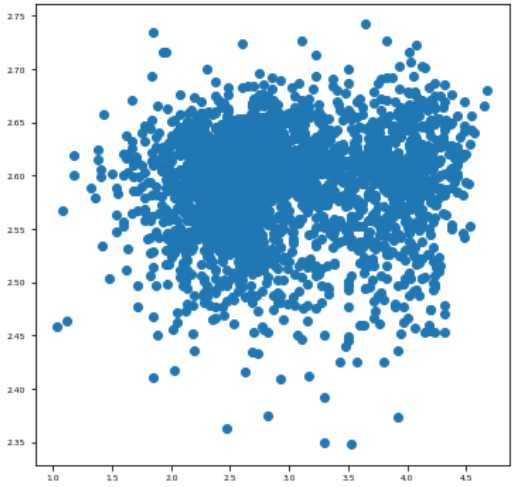
將損失函數圖畫出後可以看到其 loss 下降到大概 0.12

```
[ ] plt.rcParams['figure.figsize'] = (6,6)

    loss = history.history['loss']
    epochs = range(1, len(loss) + 1)

    plt.figure()
    plt.title('Training loss')
    plt.plot(epochs, loss, 'red', label='Training loss')
    plt.legend()

    plt.show()
```



以散布圖查看目前數據分布狀況，發現其數據是糾結在一起的，並沒有趨勢跟結果

```
[ ] plt.scatter(y_train, model.predict(x_train))
```



四、參數調整

設置了 checkpoint callback，將驗證集的 loss 設為監控項，如果 val loss 下降則保存其該值最低的模型並將其儲存 h5 檔，反之 val loss 沒有下降則判定為沒有改善，則不儲存結果。

將原先 keras 內建 learning rate 以 ReduceLROnPlateau 功能設至 min 0.00001，其目的是當訓練已無改善時，可以降低學習率，追求更細微的改善，找到更精準的最佳解。而 batch size 由原先的 32 調低至 8，並將 ResNet50 的參數設置為可訓練，再訓練 10 個 epochs，

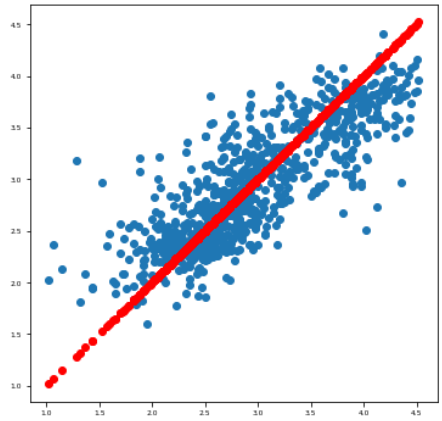
本例最終模型結果為 07-0.15.h5 檔

```
[ ] filepath="{epoch:02d}-{val_loss:.2f}.h5"
    checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='min')
    reduce_learning_rate = ReduceLROnPlateau(monitor='loss',
                                             factor=0.1,
                                             patience=2,
                                             cooldown=2,
                                             min_lr=0.00001,
                                             verbose=1)
    callback_list = [checkpoint, reduce_learning_rate]
```

```
[ ] model.layers[0].trainable = True
    model.compile(loss='mse', optimizer='adam')
    history = model.fit(x=x_train,
                        y=y_train,
                        batch_size=8,
                        epochs=10,
                        validation_data=(x_val, y_val),
                        callbacks=callback_list)
```

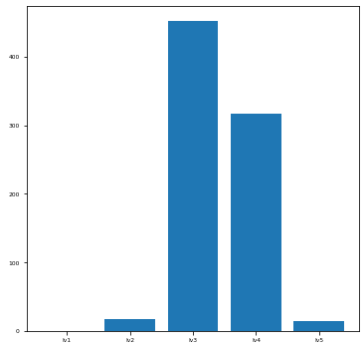
讀入上面所存之 07-0.15.h5 檔，畫出測試集之回歸分析，發現其結果與目標值較接近。

```
[ ] plt.scatter(y_test,  best_model.predict(x_test))
    plt.plot(y_test,  y_test,  'ro')
```



將測試集之長條圖畫出，其結果跟原始數據大致相同都是中間高極端值較少

```
[ ] lv1 = [x for x in scores if x<=1]
    lv2 = [x for x in scores if x>1 and x<=2]
    lv3 = [x for x in scores if x>2 and x<=3]
    lv4 = [x for x in scores if x>3 and x<=4]
    lv5 = [x for x in scores if x>4 and x<=5]

    plt.bar(['lv1','lv2','lv3','lv4','lv5'],
                [len(x) for x in [lv1, lv2, lv3, lv4, lv5]])
```



模型結果

測試集結果如下圖，誤差值在 0.7 左右。

```
[ ]  plt.rcParams['font.size'] = 9
     plt.rcParams['figure.figsize'] = (9,9)

     from random import randint
     nb_test_samples = x_test.shape[0]
     nb_rows, nb_cols = 5, 5

     def check_test_result():
            for k in range(nb_rows * nb_cols):
                   i = randint(0, nb_test_samples - 1)
                   predicted = best_model.predict(x_test[i].reshape((1,) + x_test[i].shape))
                   plt.subplot(nb_rows, nb_cols, k+1)
                   plt.imshow(x_test[i])
                   plt.title("p:%.2f a:%.2f" % (predicted[0][0], y_test[i]))
                   plt.axis('off')

     check_test_result()
```

在根據華南理工大學所發布之文獻可以看到其誤差值高加索男性、高加索女性、亞洲男性集亞洲女性數據標準差平均都在 0.7 左右。

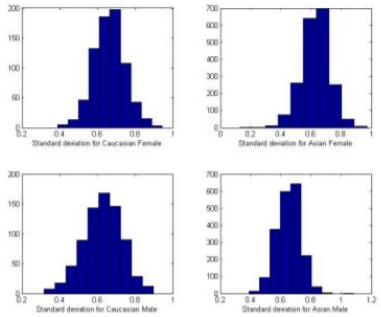


Fig. 3. Distribution of standard deviations of Caucasian female, Asian female, Caucasian male and Asian male, respectively.

將分數分為五個 lv 區間，並隨機印出測試之結果

```
plt.rcParams['font.size'] = 9
plt.rcParams['figure.figsize'] = (9,9)

from random import randint
nb_test_samples = x_test.shape[0]
nb_rows, nb_cols = 5, 5

def check_test_result():
    for k in range(nb_rows * nb_cols):
        i = randint(0, nb_test_samples - 1)
        predicted = best_model.predict(x_test[i].reshape((1,) + x_test[i].shape))
        if (predicted[0][0]<=1):
            lv=1
        if (predicted[0][0]<=2 and predicted[0][0]>1):
            lv=2
        if (predicted[0][0]<=3 and predicted[0][0]>2):
            lv=3
        if (predicted[0][0]<=4 and predicted[0][0]>3):
            lv=4
        if (predicted[0][0]<=5 and predicted[0][0]>4):
            lv=5
        if (y_test[i]<=1):
            testlv=1
        if (y_test[i]<=2 and y_test[i]>1):
            testlv=2
        if (y_test[i]<=3 and y_test[i]>2):
            testlv=3
        if (y_test[i]<=4 and y_test[i]>3):
            testlv=4
        if (y_test[i]<=5 and y_test[i]>4):
            testlv=5
        plt.subplot(nb_rows, nb_cols, k+1)
        plt.imshow(x_test[i])
        plt.title('p:%.2f a:%.2f' % (lv, testlv))
        plt.axis('off')

check_test_result()
```

實測網路隨機下載照片並進行預測

以范冰冰為例其結果為 4 分

```
[ ] plt.rcParams['font.size'] = 6
    plt.rcParams['figure.figsize'] = (6,6)

    img = load_img('C:/Users/user/1jupyter notebook example/Beauty-predict-master/images/fbb.jpg')
    plt.imshow(img)
    img = imresize(img, size=(img_height, img_width))
    test_x = img_to_array(img).reshape(img_height, img_width, channels)
    test_x = test_x / 255.
    test_x = test_x.reshape((1,) + test_x.shape)
    predicted = best_model.predict(test_x)
    if (predicted[0][0]<=1):
            lv=1
            p=20
    if (predicted[0][0]<=2 and predicted[0][0]>1):
            lv=2
            p=40
    if (predicted[0][0]<=3 and predicted[0][0]>2):
            lv=3
            p=60
    if (predicted[0][0]<=4 and predicted[0][0]>3):
            lv=4
            p=80
    if (predicted[0][0]<=5 and predicted[0][0]>4):
            lv=5
            p=100
    print(("predicted: {}, 你贏了{}%的人").format( lv,p))
```

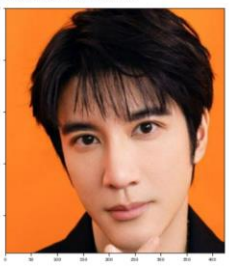
```
C:\Users\user\.conda\envs\face_predict\lib\site-packages\ipykernel_launcher.py:6: DeprecationWarning: `imresize` is deprecated!
    `imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
    Use ``skimage.transform.resize`` instead.

predicted: 4，你贏了80%的人
```

下圖再以不同張范冰冰照片測其結果還是 4 分

```
C:\Users\user\.conda\envs\face_predict\lib\site-packages\ipykernel_launcher.py:6: DeprecationWarning: `imresize` is deprecated!
    `imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
    Use ``skimage.transform.resize`` instead.

predicted: 4，你贏了80%的人
```

以王力宏為例，其結果為 4 分

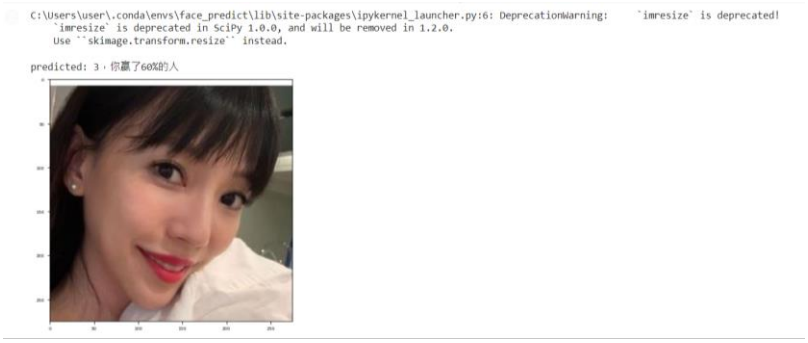
```
C:\Users\user\.conda\envs\face_predict\lib\site-packages\ipykernel_launcher.py:6: DeprecationWarning: `imresize` is deprecated!
    `imresize` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
    Use ``skimage.transform.resize`` instead.

predicted: 4，你贏了80%的人
```



以 Yumi 為例，其結果為 4 分

以徐若瑄為例，其結果為 4 分

以李靜蕾為例，其結果為 3 分

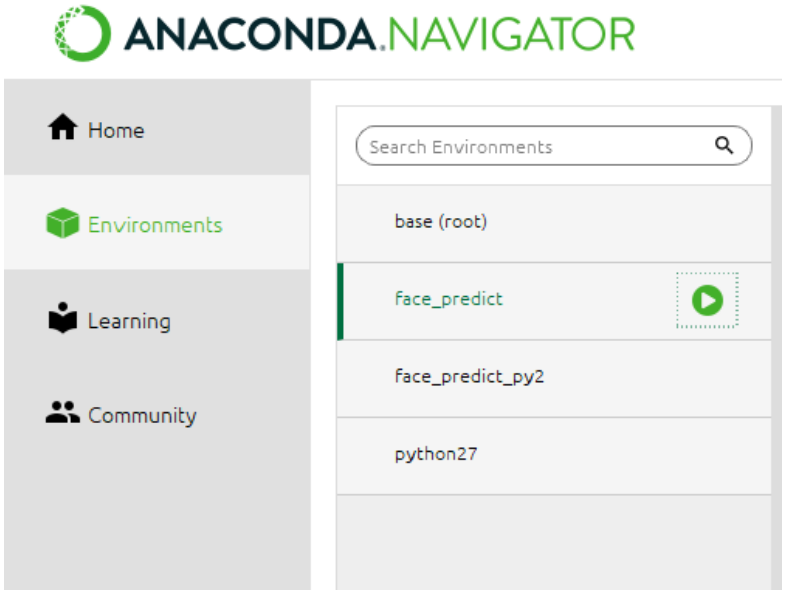


以邱銘傳老師為例，其結果為 3 分

五、網頁架設

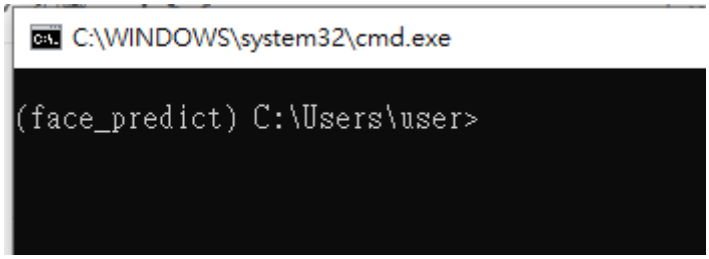
第一部份 套件安裝

Step1 :打開 Anaconda 如圖一



(圖一)

Step2： 選擇你的環境，此例是用 face_predict，並點選  ，點選“Open Terminnal”叫出終端機，如圖二



(圖二)

Step2： 再命令行打上表一的所有需要的套件

Pip install keras==2.1.5
Pip install flask==2.0.2
Pip install tensorflow==1.5.0
Pip install scipy==1.1.0
Pip install numpy==1.19.5
Pip install opencv-python==4.5.4.60

(表一)

Step3 準備模型檔案，此例檔名為” 07-0.15.h5”

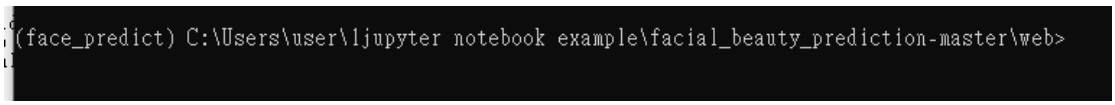
Step4 準備 OpenCV 下的特徵訓練的分類器，此例檔名為”
haarcascade_frontalface_alt.xml”

Step5 撰寫 html (成功與失敗各一) 程式說明於附件一

Step6 撰寫與 html 與 python 連接的 python 檔，此例名稱為 “NEWapp.py”，並用 flask 做撰寫，程式說明於附件二

Step7 用 Step2 的方式叫出終端機

Step8 移動到步驟 6 選寫完的 python 檔案目錄，此例目錄位置是
“C:\Users\user\1jupyter notebook example\facial_beauty_prediction-master\web”
所以在終端機的命令行打上
“cd C:\Users\user\1jupyter notebook example\facial_beauty_prediction-master\web”
即可移動至該目錄，如圖三

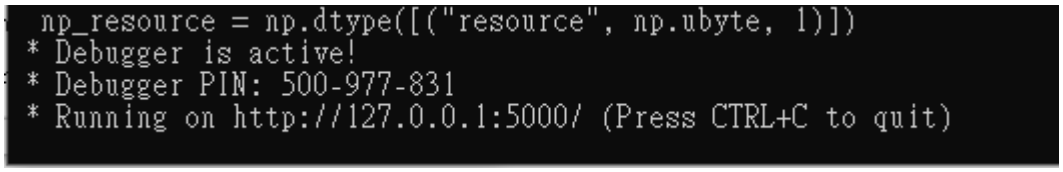


(圖三)

Step9 在命令行輸入 “ python NEW app.py”，執行 py 檔，如圖四

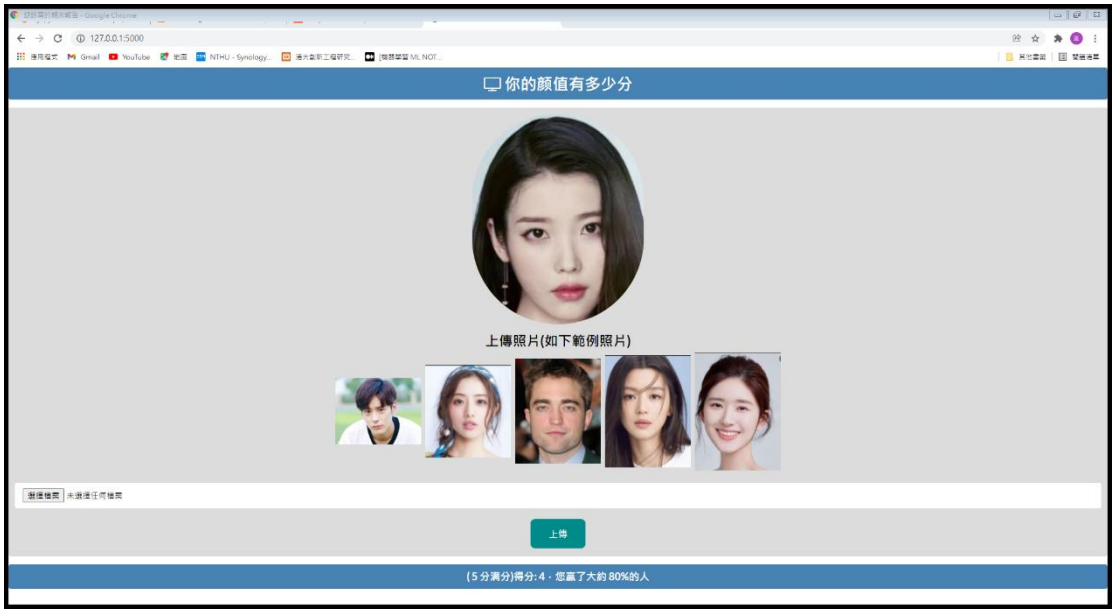
(圖四)

Step10 執行成功後，如圖五



(圖五)

Step11 照著圖五說明在 google 搜尋 127.0.0.1:5000，成功如圖六

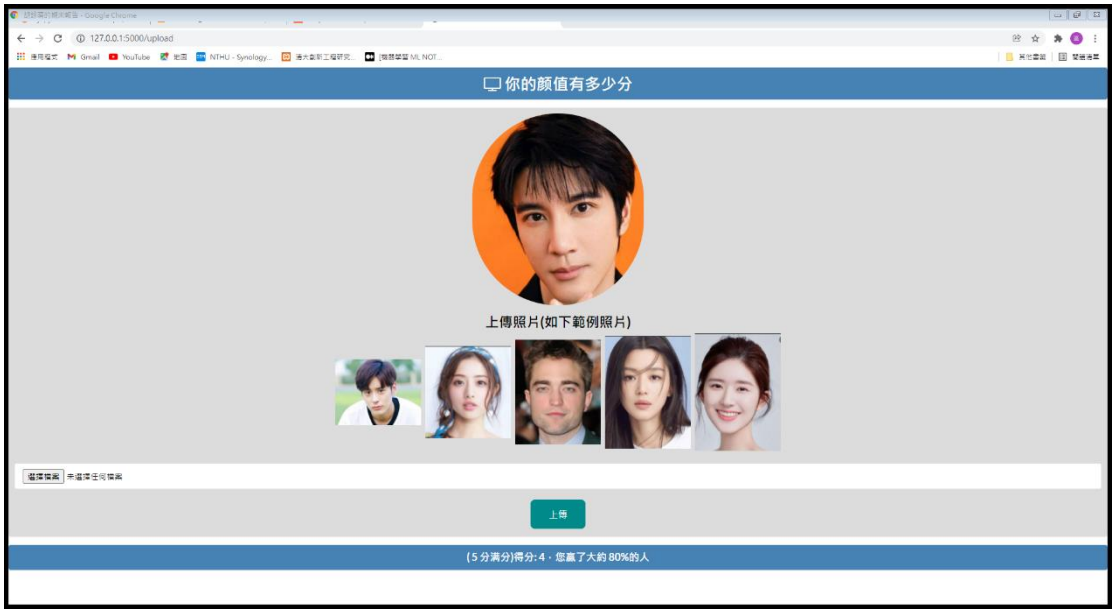


(圖六)

Step12 點選上傳檔案按鈕，並從電腦找尋一張人的圖片

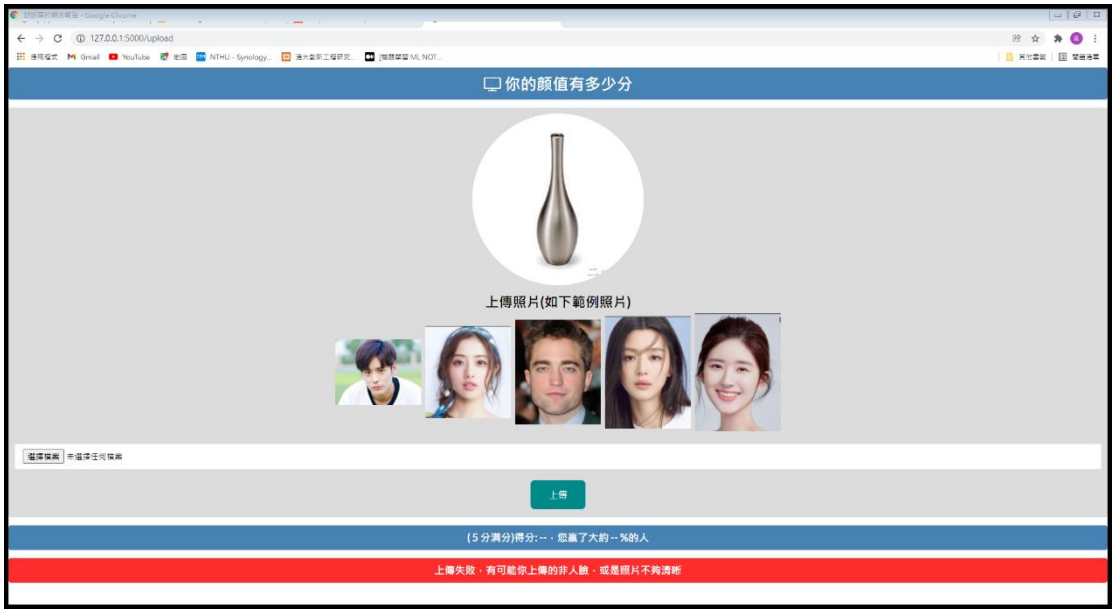
Step13 點選上傳，讓系統跑

Step14-1 若系統偵測出人臉就得出解答，如圖七



(圖七)

Step14-2 若系統無法偵測出人臉就得會警告，如圖八



(圖八)

附錄一

成功版的 html

```
<html>

  <html>

  <head>
    <title>胡詠晴的期末報告</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script src="/static/js/jquery.min.js"></script>
    <script src="/static/js/semantic.min.js"></script>
    <link rel="stylesheet" href="/static/css/semantic.min.css" />
    <style>


    h1 {color: #FFFFFF;}
    h2 {color: #000000;}
    h3 {color: #FFFFFF;}
    .button {
    background-color: #008B8B;
    border: none;
    color: white;
    padding: 15px 32px;
    border-radius: 8px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    }


    </style>
  </head>
```

```
  <body>
  <div class="ui inverted    center aligned big segment" style="background-color:#4682B4;padding:10px;">
        <h1>
            <i class="tv icon"> </i>你的顏值有多少分
        </h1>
      </div>
      <div class="ui inverted center aligned big segment" style="background-color:#DCDCDC;padding:10px;">
          <form class="ui form" method=POST enctype=multipart/form-data action="{{ url_for('upload') }}">
            <div class="field">
              <img class="ui centered medium circular image" src="
{{url_for('send_image', filename=image_name)}}">
            </div>
            <div class="field">
              <label><h2>上傳照片(如下範例照片)</h2></label>
              <div class="ui small images">
                <img class="ui image" src="/uploads/demo6.jpg">
                <img class="ui image" src="/uploads/demo10.jpg">
                <img class="ui image" src="/uploads/demo7.jpg">
                <img class="ui image" src="/uploads/demo8.jpg">
                <img class="ui image" src="/uploads/demo9.jpg">

              </div>
            </div>
            <div class="field">

              <input type="file" name="photo" >



            </div style="background-color:#D2691E;">

            <input type="submit" value="上傳" class="button" style="vertical-align:middle;">

          </form>
```

```
        </div>
        <div class="ui inverted    center aligned big segment" style="background-
color:#4682B4;padding:10px;">
            <div class="field">
                <label><h3>( 5 分滿分)得分: {{ score }}，您贏了大約 {{word}}%的人
</h3></label>
            </div>
        </div>
    </body>
</html><html>
```

失敗版

```
<head>
    <title>胡詠晴的期末報告</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script src="/static/js/jquery.min.js"></script>
    <script src="/static/js/semantic.min.js"></script>
    <link rel="stylesheet" href="/static/css/semantic.min.css" />
    <style>


    h1 {color: #FFFFFF;}
    h2 {color: #000000;}
    h3 {color: #FFFFFF;}
    .button {
    background-color: #008B8B;
    border: none;
    color: white;
    padding: 15px 32px;
    border-radius: 8px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
    }


    </style>
  </head>

  <body>
  <div class="ui inverted    center aligned big segment" style="background-
color:#4682B4;padding:10px;">
        <h1>
            <i class="tv icon"> </i>你的顏值有多少分
        </h1>
    </div>
```

```
    <div class="ui inverted center aligned big segment" style="background-color:#DCDCDC;padding:10px;">
        <form class="ui form" method=POST enctype=multipart/form-data action="{{ url_for('upload') }}">
          <div class="field">
            <img class="ui centered medium circular image" src="{{url_for('send_image', filename=image_name)}}">
          </div>
          <div class="field">
            <label><h2>上傳照片(如下範例照片)</h2></label>
            <div class="ui small images">
              <img class="ui image" src="/uploads/demo6.jpg">
              <img class="ui image" src="/uploads/demo10.jpg">
              <img class="ui image" src="/uploads/demo7.jpg">
              <img class="ui image" src="/uploads/demo8.jpg">
              <img class="ui image" src="/uploads/demo9.jpg">

            </div>
          </div>
          <div class="field">

            <input type="file" name="photo" >



          </div style="background-color:#D2691E;">

          <input type="submit" value="上傳" class="button" style="vertical-align:middle;">

        </form>
    </div>
    <div class="ui inverted    center aligned big segment" style="background-color:#4682B4;padding:10px;">
      <div class="field">
        <label><h3>( 5 分滿分)得分: --，您贏了大約 -- %的人</h3></label>
      </div>
    </div>
```

```
      <div class="ui inverted    center aligned big segment" style="background-
color:#FF2D2D;padding:10px;">
          <div class="field">
             <label><h3>上傳失敗，有可能你上傳的非人臉，或是照片不夠清晰
</h3></label>
          </div>
       </div>
   </body>
</html>
```

App.py 程式碼

```
#設定程式編碼為 utf-8
#coding: utf-8
#宣告所使用套件
from uuid import uuid4
from flask import Flask, render_template, request, send_from_directory
from flask_uploads import UploadSet, configure_uploads, IMAGES
import keras
from keras.models import load_model
from keras.preprocessing.image import array_to_img, img_to_array, load_img
from scipy.misc import imresize
import numpy as np
import cv2
from gevent.pywsgi import WSGIServer
#初始化 Flask 物件，並貼上 app 這個標籤。接著我們就可以很方便的使用這個
#物件裡面的各種功能。
app = Flask(__name__)
# flask_uploads 用以檔案傳輸，以下 UploadSet(name,限制上傳格式)
#命名為"photo"，上傳的格式為照片
photos = UploadSet('photos', IMAGES)
#設定檔案上傳後的儲存地址，名為 uploads 的資料夾
app.config['UPLOADED_PHOTOS_DEST'] = 'uploads'
#初始化 flask_uploads，將剛設定的檔案上傳的形式，套用到 Flask 上
configure_uploads(app, photos)
#設定照片高、寬、RGB
img_height, img_width, channels = 350, 350, 3
#打開剛剛步驟四準備的分類器
haar_face_cascade = cv2.CascadeClassifier('C:/Users/user/1jupyter notebook
example/facial_beauty_prediction-master/web/data/haarcascade_frontalface_alt.xml')
#偵測臉部的自訂函數有一個參數，需要輸入，一是檔案路徑
def detect_face(filepath):
    #利用 KERAS 內函數 load_img，讀取圖片
    img = load_img(filepath)
    #利用 KERAS 內函數 imresize，將圖片改至指定大小
    img = imresize(img, size=(img_height, img_width))
    #照片轉陣列，以方便電腦判讀
```

```
        test_x = img_to_array(img).reshape(img_height, img_width, channels)
        #如果認定非人臉
        if len(test_x) == 0:
            #回傳一個全為 0 的矩陣
            return np.zeros((1,1))
        #抓取檔名，從最後數來，只要遇到/就停止，這樣可以抓取檔名
        filename = filepath.split('/')[-1]
        #存照片
        cv2.imwrite("uploads/cropped_{}".format(filename), test_x)
        #回傳剛剛裁切的圖片
        return test_x
#獲取分數的自訂函數，參數為檔案路徑
def get_score(filepath):
        #呼叫剛剛寫好的偵測臉部的自訂函數
        test_x = detect_face(filepath)
        #如果沒東西
        if not test_x.any():
            #回傳沒有
            return None
        #如果有，除以 255 將圖像標準化是將數據通過去均值實現中心化
        test_x = test_x / 255.
        test_x = test_x.reshape((1,) + test_x.shape)
        #將之前讀的模型清空，若不做這一步，照片只能傳一次
        keras.backend.clear_session()
        #讀取模型
        model = load_model("C:/Users/user/1jupyter notebook
example/facial_beauty_prediction-master/07-0.15.h5")

        #把剛剛去中心化的的照片，餵給模型
        predicted = model.predict(test_x)
        #回傳模型得出的結果
        return round(predicted[0][0],2)
#當輸入 127.0.0.1:5000 預設的網頁
@app.route("/")
def index():
        return render_template("upload.html", image_name="demo.jpg",
score=4,word=80)
#辨別是否有資料進出，取得資訊的時候 GET，送出資訊的時候 POST
```

```
@app.route('/upload', methods=['GET', 'POST'])
def upload():
    #如果偵測到送出資料並圖片有上傳
    if request.method == 'POST' and 'photo' in request.files:
        #照片儲存
        filename = photos.save(request.files['photo'],
name="{}.".format(str(uuid4())))
        #呼叫 get_score，得到結果
        score = get_score("C:/Users/user/1jupyter notebook
example/facial_beauty_prediction-master/web/uploads/{}".format(filename))
        #如果沒有結果
        if not score:
            #從路徑中名為 template 資料夾中，回傳 ERROR 的 HTML
            return render_template("error.html",
image_name="{}".format(filename))
        else:
            如果有值，並界於區間，則得到我們想要的績效指標
            if (score<=1):
                lv=1
                p=20
            if (score<=2 and score>1):
                lv=2
                p=40
            if (score<=3 and score>2):
                lv=3
                p=60
            if (score<=4 and score>3):
                lv=4
                p=80
            if (score<=5 and score>4):
                lv=5
                p=100
            #從路徑中名為 template 資料夾中，回傳 upload 的 HTML
            return render_template("upload.html",
image_name="{}".format(filename), score=lv,word=p)
#上傳圖片，用內建功能即可
@app.route('/uploads/<filename>')
def send_image(filename):
```

```
    return send_from_directory("uploads", filename)


if __name__ == '__main__':
    #將 debug 功能打開，幫助自己 DEBUG
    app.debug=True
    #啟動 FLASK
    app.run()
```

六、研究結論與未來展望

結論

該資料集相較以前的數據集有更清晰的分類，更豐富的人臉庫，但該數據集沒有有色人種，可能存在一點倫理上的問題，期盼有更多的資料讓整個模型結果更為精準，本次研究結果針對每一張人臉照都有做出分數的判斷，但因為數據分數本身的量尺分數只有 1-5 分造成區間落在 3-4 的比例居多，但還是能做為一個簡單的顏值評分工具。

未來展望

在做模型評估的時候因為時間限制所以參數調整的部分較不完全，未來如果有機會自己製作數據集的時候可以把量尺分數拉寬，這樣的分數可以更加精準，在選美比賽上分數也更為直觀，同時此技術也可以用在基礎整形外科臉部評分中，讓患者知道自己的整形前後顏值各是落在哪個區間！

七、參考文獻

中國古代的美女標準竟如此嚴苛
https://kknews.cc/zh-tw/culture/p55zjgp.html

衡量美女標準是什麼？1-10 分，5 分就算美女!
https://www.xuehua.us/a/5ec1fb8d6fcd4773f3aa3983?lang=zh-hk

SCUT-FBP5500: A Diverse Benchmark Dataset for Multi-Paradigm Facial Beauty Prediction

【深度學習】ResNet 解讀及程式碼實現
https://www.itread01.com/content/1542368643.html

Residual Leaning: 認識 ResNet 與他的冠名後繼者 ResNeXt、ResNeSt
https://medium.com/%E8%BB%9F%E9%AB%94%E4%B9%8B%E5%BF%83/deep-learning-residual-leaning-%E8%AA%8D%E8%AD%98resnet%E8%88%87%E4%BB%96%E7%9A%84%E5%86%A0%E5%90%8D%E5%BE%8C%E7%B9%BC%E8%80%85resnext-resnest-6bedf9389ce

How Attractive Are You in the Eyes of Deep Neural Network?
https://towardsdatascience.com/how-attractive-are-you-in-the-eyes-of-deep-neural-network-3d71c0755ccc

SCUT-FBP5500 人臉美學預測 論文解讀
https://zhuanlan.zhihu.com/p/346096100

Keras Callback 的使用
https://ithelp.ithome.com.tw/m/articles/10234641

基於深度學習的顏值打分器
https://www.zhihu.com/column/p/36138077