Psychoinformatics - Week 13 (Exercises)

by Forky (b11209016@ntu.edu.tw)

1 進一步研究CNN (4 points)

1.1 為何ResNet50會判斷小女孩照片為ping-pong_bal, bubble, or Band_Aid? (4 points)

In [ ]:

```
import numpy as np
import urllib.request
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions

model = ResNet50(weights='imagenet')

urllib.request.urlretrieve('http://mil.psy.ntu.edu.tw/~tren/girl.jpg','girl.jpg')
img = image.load_img('girl.jpg', target_size=(224, 224)) # Or use cv2.resize to resize images
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
x = preprocess_input(x)

preds = model.predict(x)
# decode the results into a list of tuples (class, description, probability)
# (one such list for each sample in the batch)
print('Predicted:', decode_predictions(preds, top=3)[0])
```

```
WARNING:tensorflow:From c:\Users\88698\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\utils\version_utils.py:76: The na
me tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From c:\Users\88698\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\normalization\batch_normal
ization.py:964: The name tf.nn.fused_batch_norm is deprecated. Please use tf.compat.v1.nn.fused_batch_norm instead.

WARNING:tensorflow:From c:\Users\88698\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\layers\normalization\batch_normal
ization.py:883: _colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From c:\Users\88698\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training_v1.py:635: The na
me tf.data.Iterator is deprecated. Please use tf.compat.v1.data.Iterator instead.

WARNING:tensorflow:From c:\Users\88698\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training_utils_v1.py:50: T
he name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

Predicted: [('n03942813', 'ping-pong_ball', 0.17008628), ('n09229709', 'bubble', 0.10647234), ('n02786058', 'Band_Aid', 0.10364515)]
```

Please write your discussion here

ResNet50 是在 ImageNet 數據集上訓練的，而 ImageNet 數據集包含了大量的物體類別，其中也包括了一些與人類相關的類別，例如小女孩、人類等。然而，ImageNet 的類別不僅僅涵蓋人類生活的各個方面，還包括了許多日常生活中不太可能出現的物體。

當 ResNet50 在這樣的模型上進行預測時，它會嘗試將輸入的圖片映射到它在訓練過程中見過的類別。由於 ImageNet 有超過一千個類別，其中一些可能與人類或人類活動相關，模型可能會產生與人類有關的類別作為預測結果。而且如果假設給的圖片不是正規(可能正規表示有兩個眼睛，一個鼻子，一個嘴巴，還有有臉的輪廓等等)的圖片，我的意思是照片可能受人臉的特定表情、姿勢、光線變化等，而這些可能不是 ImageNet 數據集所能充分涵蓋的，比如照片上就有一個葉子擋在小女孩的鼻子前，這就是忽略局部與整體的關聯。

模型預測的結果包括了 'ping-pong_ball'、'bubble' 和 'Band_Aid'，這些可能是 ImageNet 數據集中的某些類別，而模型認為這些類別與輸入圖片相關。這不一定表示模型真正理解了圖片，而只是它將圖片映射到了它訓練過的類別中。模型的預測結果可能受到數據集中類別的影響，因此結果可能有時候會出現直覺上看似不合理的情況。

1.2 請展示有別人pre-trained好的Keras model可以成功辨認girl.jpg為人臉 (4 points)

人臉辨識 - 轉換、對齊、裁剪、特徵擷取與比對

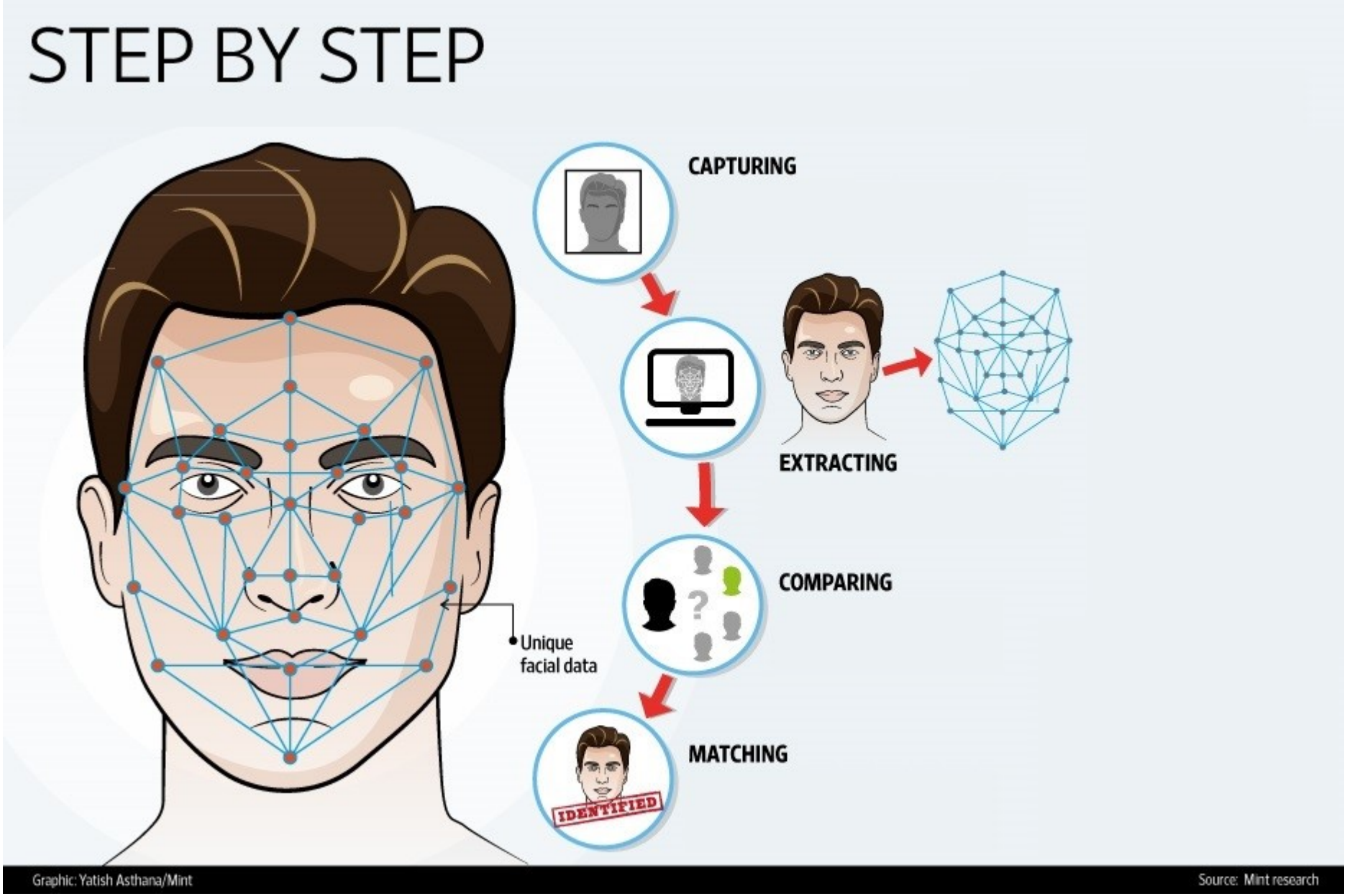
人臉辨識大致可分成四個主要的步驟:

1. 人臉偵測
2. 人臉轉換、對齊與裁剪

3. 人臉特徵擷取
4. 人臉特徵比對

這個Jupyter Notebook展示了一個四個步驟整合起來的結果。但要進行這個Jupyter notebook之前, 要先完成以下兩個程序：

- [01-face-detect-align-and-crop] - 介紹如何進行"人臉偵測"、"對齊" & "裁剪"。
- [02-face-embedding-and-recognition-classifier] - 介紹如何進行人臉特徵擷取(FaceNet) & 訓練人臉分類器。



face-recognition 專案說明

face-recognition包含了使用MTCNN、FaceNet以及SVM(Suport Vector Machine)三種演算法來進行人臉辨識的整個循環。

安裝

```
git clone https://github.com/erhwenkuo/face-recognition.git
cd face-recognition
...
```

資料集說明

LFW資料集是一個常見的人臉資料集，歷史非常悠久。LFW資料集中收錄了5749位公眾人物的人臉影像，總共有超過一萬三千多張影像檔案。但大部份公眾人物的影像都只有一張，只有1680位有超過一張照片，而極少數有超過10張照片。

資料集的網站: http://vis-www.cs.umass.edu/lfw

專案的檔案路徑佈局

1. 使用Git從erhwenkuo/face-recognition下載整個專案源碼
2. 在 `face-recognition` 的目錄裡產生二個子目錄 `data` 與 `model`
3. 從[Labeled Faces in the Wild資料集官網]點擊All images as gzipped tar file來下載 `lfw.tgz` 。
4. 解壓縮 `lfw.tgz` 到 `face-recognition/data/` 的目錄下
5. 執行 `01-face-detect-align-and-crop.ipynb` 來進行臉部偵測、對齊 & 裁剪
6. 下載Facenet模型檔20170511-185253.zip(168M)並解壓縮到"model/facenet"的目錄下。
7. 在"model"的目錄下產生一個子目錄"svm"來存放"人臉分類器"的模型。
8. 執行 `02-face-embedding-and-recognition-classifier.ipynb` 來進行人臉特徵擷取(FaceNet) & 訓練人臉分類器
9. 在"data"的目錄下產生一個子目錄"test"來存放"人臉辨識"用的測試圖像

最後你的目錄結構看起來像這樣: (這裡只列出來在這個範例會用到的相關檔案與目錄)

```
face-recognition/
├── 01-face-detect-align-and-crop.ipynb
├── 02-face-embedding-and-recognition-classifier.ipynb
├── 03-face-classification.ipynb
├── detect_face.py
├── facenet.py
├── visualization_utils.py
├── model/
│   ├── svm/                                <--- 人臉分類器(svm)的模型
│   │   └── lfw_svm_classifier.pkl
│   ├── mtcnn/
│   │   ├── det1.npy
│   │   ├── det2.npy
│   │   └── det3.npy
```

```
│   └── facenet/                              <--- Facenet的模型
│       └── 20170512-110547/
│           ├── 20170512-110547.pb
│           ├── model-20170512-110547.ckpt-250000.data-00000-of-00001
│           ├── model-20170512-110547.ckpt-250000.index
│           └── model-20170512-110547.meta
└── data/
    ├── test/
    ├── lfw/
    │   ├── Aaron_Eckhart/
    │   │   └── Aaron_Eckhart_0001.jpg
    │   ├── ...
    │   └── Zydrunas_Ilgauskas/
    │       └── Zydrunas_Ilgauskas_0001.jpg
    └── lfw_crops/                            <--- 經過偵測、對齊 & 裁剪後的人臉圖像
        ├── Aaron_Eckhart/
        │   └── Aaron_Eckhart_0001.png
        ├── ...
        └── Zydrunas_Ilgauskas/
            └── Zydrunas_Ilgauskas_0001.png
```

STEP 1. 載入相關函式庫

In [ ]:

```python
# 屏蔽Jupyter的warning訊息
import warnings
warnings.filterwarnings('ignore')

# Utilities相關函式庫
import os
from os.path import join as pjoin
import sys
import time
import copy
import random
import math
from tqdm import tqdm
from scipy import misc
from scipy.spatial import distance # 用來計算歐幾里德距離 (euclidean)

# 圖像處理相關函式庫
import cv2
import matplotlib.pyplot as plt
```

```
# 多維向量處理相關函式庫
import numpy as np

# 深度學習相關函式庫
import tensorflow as tf

# 機械學習
from sklearn.svm import LinearSVC
import joblib

# 模型序列化函式庫
import pickle

# 專案相關函式庫
import facenet
import detect_face
import visualization_utils as vis_utils
```

```
WARNING:tensorflow:From c:\Users\88698\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

STEP 2. 設定相關設定與參數

In [ ]:

```
# 專案的根目錄路徑
ROOT_DIR = os.getcwd()

# 訓練/驗證用的資料目錄
DATA_PATH = os.path.join(ROOT_DIR, "data")

# 模型的資料目錄
MODEL_PATH = os.path.join(ROOT_DIR, "model")

# MTCNN的模型
MTCNN_MODEL_PATH = os.path.join(MODEL_PATH, "mtcnn")

# FaceNet的模型
FACENET_MODEL_PATH = os.path.join(MODEL_PATH, "facenet","20170512-110547","20170512-110547.pb")

# Classifier的模型
SVM_MODEL_PATH = os.path.join(MODEL_PATH, "svm", "lfw_svm_classifier.pkl")

# 訓練/驗證用的圖像資料目錄
IMG_IN_PATH = os.path.join(DATA_PATH, "lfw")
```

```
# 訓練/驗證用的圖像資料目錄
IMG_OUT_PATH = os.path.join(DATA_PATH, "lfw_crops")
```

STEP 3. 載入人臉Facenet處理過的相關的人臉embedding資料

轉換每張人臉的圖像成為Facenet的人臉特徵向量(128 bytes)表示。

函式: `facenet.get_dataset`

參數:
    paths (string): 圖像資料集的檔案路徑
    has_class_directories (bool): 是否使用子目錄名作為人臉的identity (預設為True)
    path_expanduser (bool): 是否把path中包含的"~"和"~user"轉換成在作業系統下的用戶根目錄 (預設為False)
回傳:
    dataset (list[ImageClass]) : 人臉類別(ImageClass)的列表與圖像路徑

In [ ]:
```
# 反序列化相關可重覆使用的資料
# "人臉embedding"的資料
with open(os.path.join(DATA_PATH,'lfw_emb_features.pkl'), 'rb') as emb_features_file:
    emb_features =pickle.load(emb_features_file)

# "人臉embedding"所對應的標籤(label)的資料
with open(os.path.join(DATA_PATH,'lfw_emb_labels.pkl'), 'rb') as emb_lables_file:
    emb_labels =pickle.load(emb_lables_file)

# "標籤(label)對應到人臉名稱的字典的資料
with open(os.path.join(DATA_PATH,'lfw_emb_labels_dict.pkl'), 'rb') as emb_lables_dict_file:
    emb_labels_dict =pickle.load(emb_lables_dict_file)
```

由於lfw的人臉資料庫的人臉圖像太少, 因此經過過濾之後我們從lfw的人臉資料庫中選出423個人臉的類別(每個類別都至少有5張的圖像以上)來做為人臉辨識的目標範例資料集。

In [ ]:
```
emb_dict = {} # key 是label, value是embedding list
for feature,label in zip(emb_features, emb_labels):
    # 檢查key有沒有存在
    if label in emb_dict:
        emb_dict[label].append(feature)
    else:
        emb_dict[label] = [feature]
```

In [ ]:
```
# 計算兩個人臉特徵(Facenet Embedding 128 bytes vector)的歐式距離
def calc_dist(face1_emb, face2_emb):
    return distance.euclidean(face1_emb, face2_emb)
```

```
face_distance_threshold = 1.1

# 計算一個人臉的embedding是不是歸屬某一個人
# 根據Google Facenet的論文, 透過計算兩個人臉embedding的歐氏距離
# 0: 代表為同一個人臉 , threshold <1.1 代表兩個人臉非常相似
def is_same_person(face_emb, face_label, threshold=1.1):
    emb_distances = []
    emb_features = emb_dict[face_label]
    for i in range(len(emb_features)):
        emb_distances.append(calc_dist(face_emb, emb_features[i]))

    # 取得平均值
    if np.mean(emb_distances) > threshold: # threshold <1.1 代表兩個人臉非常相似
        return False
    else:
        return True
```

STEP 4. 載入預訓練MTCNN的模型來偵測人臉位置

設定人臉偵測模型所需的相關參數

In [ ]:
```
minsize = 40  # 最小的臉部的大小
threshold = [0.6, 0.7, 0.7]  # 三個網絡(P-Net, R-Net, O-Net)的閥值
factor = 0.709  # scale factor

margin = 44 # 在裁剪人臉時的邊框margin
image_size = 182 # 160 + 22

batch_size = 1000
input_image_size = 160
```

In [ ]:
```
# 創建Tensorflow Graph物件
tf_g = tf.compat.v1.Graph().as_default()
gpu_options = tf.compat.v1.GPUOptions(per_process_gpu_memory_fraction=0.6)

# 創建Tensorflow Session物件
tf_sess = tf.compat.v1.Session(config=tf.compat.v1.ConfigProto(gpu_options=gpu_options, log_device_placement=False))

# 把這個Session設成預設的session
tf_sess.as_default()
```

```
WARNING:tensorflow:From C:\Users\88698\AppData\Local\Temp\ipykernel_26996\3418658045.py:3: The name tf.GPUOptions is deprecated. Please use
tf.compat.v1.GPUOptions instead.
```

Out[ ]:
```
<contextlib._GeneratorContextManager at 0x15a0fa9dad0>
```

In [ ]:
```
# 載入MTCNN模型 (偵測人臉位置)
# 載入MTCNN模型 (偵測人臉位置)
tf.compat.v1.disable_eager_execution()
sess = tf.compat.v1.Session()
pnet, rnet, onet = detect_face.create_mtcnn(tf_sess, MTCNN_MODEL_PATH)
```

STEP 5. 載入預訓練FaceNet的模型來擷取人臉特徵

In [ ]:
```
# 載入Facenet模型
with tf.compat.v1.Session() as sess:
    # Load the MTCNN model
    print('Loading feature extraction model')
    modeldir =  FACENET_MODEL_PATH
    facenet.load_model(modeldir)

    # Get the input and output placeholders from the model
    images_placeholder = tf.compat.v1.get_default_graph().get_tensor_by_name("input:0")
    embeddings = tf.compat.v1.get_default_graph().get_tensor_by_name("embeddings:0")
    phase_train_placeholder = tf.compat.v1.get_default_graph().get_tensor_by_name("phase_train:0")
    embedding_size = embeddings.get_shape()[1]

    # Print the size of the face embedding vector
    print("Face embedding size: ", embedding_size)
```

```
Loading feature extraction model
Model filename: c:\Users\88698\OneDrive\桌面\大二上\心理與神經資訊學\hw\hw13\face-recognition\model\facenet\20170512-110547\20170512-110547.pb
WARNING:tensorflow:From c:\Users\88698\OneDrive\桌面\大二上\心理與神經資訊學\hw\hw13\face-recognition\facenet.py:443: FastGFile.__init__ (from
tensorflow.python.platform.gfile) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.gfile.GFile.
Face embedding size:  128
```

STEP 6. 載入預訓練SVM分類器模型來進行人臉識別

In [ ]:
```
# 載入SVM分類器模型
classifier_filename = SVM_MODEL_PATH

with open(classifier_filename, 'rb') as svm_model_file:
    (face_svc_classifier, face_identity_names) = pickle.load(svm_model_file)
```

```
    HumanNames = face_identity_names     #訓練時的人臉的身份

    print('load classifier file-> %s' % classifier_filename)
    print(face_svc_classifier)
```

```
load classifier file-> c:\Users\88698\OneDrive\桌面\大二上\心理與神經資訊學\hw\hw13\face-recognition\model\svm\lfw_svm_classifier.pkl
LinearSVC(C=1)
```

STEP 7. 進行人臉識別

從網路上找一個人臉圖像來進行測試:

圖像URL: https://xk.usembassy.gov/wp-content/uploads/sites/133/2016/09/Four_Presidents-1.jpg

把這個圖像下載下來到專案的測試目錄下: "**data/test/Four_Presidents-1.jpg**"

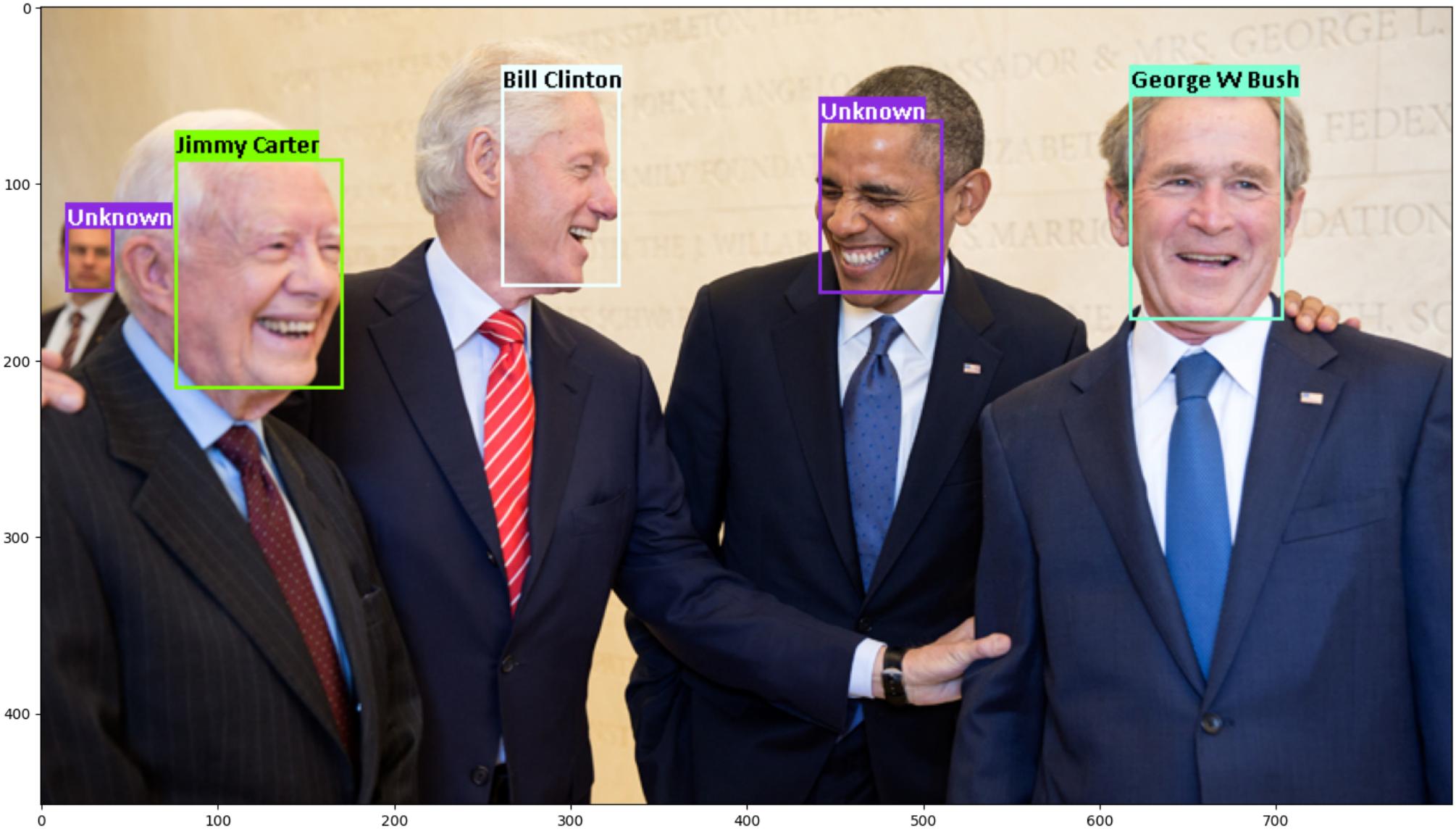
In [ ]:

```
from skimage.transform import resize
print('Start Recognition!')
from PIL import Image, ImageDraw
face_input = "data/test/Four_Presidents-1.jpg"

find_results = []
image = Image.open(face_input).convert('RGB')
frame = np.array(image)
bounding_boxes, _ = detect_face.detect_face(frame, minsize, pnet, rnet, onet, threshold, factor)

draw = frame.copy() # 複製原圖像
# Detect faces in the frame

# 步驟 #1.偵測人臉位置
# 偵測人臉的邊界框

nrof_faces = bounding_boxes.shape[0] # 被偵測到的臉部總數
if nrof_faces > 0: # 如果有偵測到人臉
    # 每一個 bounding_box包括了(x1,y1,x2,y2,confidence score):
    #     左上角座標 (x1,y1)
    #     右下角座標 (x2,y2)
    #     信心分數 confidence score
    det = bounding_boxes[:, 0:4].astype(int) # 取出邊界框座標
    img_size = np.asarray(frame.shape)[0:2] # 原圖像大小 (height, width)

    print("Image: ", img_size)

    # 人臉圖像前處理的暫存
    cropped = []
    scaled = []
    scaled_reshape = []
    bb = np.zeros((nrof_faces,4), dtype=np.int32)

    # 步驟 #2.擷取人臉特徵
    for i in range(nrof_faces):
        print("faces#{}".format(i))
        emb_array = np.zeros((1, embedding_size))

        x1 = bb[i][0] = det[i][0]
        y1 = bb[i][1] = det[i][1]
        x2 = bb[i][2] = det[i][2]
        y2 = bb[i][3] = det[i][3]

        print('({}, {}) : ({}, {})'.format(x1,y1,x2,y2))
        # inner exception
        if bb[i][0] <= 0 or bb[i][1] <= 0 or bb[i][2] >= len(frame[0]) or bb[i][3] >= len(frame):
```

```
            print('face is out of range!')
            continue

        # **人臉圖像的前處理 **

        # 根據邊界框的座標來進行人臉的裁剪
        cropped.append(frame[bb[i][1]:bb[i][3], bb[i][0]:bb[i][2], :])
        cropped[i] = facenet.flip(cropped[i], False)
        scaled.append(resize(cropped[i], (image_size, image_size)))
        scaled[i] = cv2.resize(scaled[i], (input_image_size,input_image_size),
                               interpolation=cv2.INTER_CUBIC)
        scaled[i] = facenet.prewhiten(scaled[i])
        scaled_reshape.append(scaled[i].reshape(-1,input_image_size,input_image_size,3))
        feed_dict = {images_placeholder: scaled_reshape[i], phase_train_placeholder: False}

        # 進行臉部特徵擷取
        # 進行臉部特徵擷取
        emb_array[0, :] = tf_sess.run(embeddings, feed_dict=feed_dict)
        emb_array = emb_array.reshape(-1)

        # 步驟 #3.進行人臉識別分類
        face_id_idx = face_svc_classifier.predict(emb_array.reshape(1, -1))

        if is_same_person(emb_array, int(face_id_idx), 1.1):
            face_id_name = HumanNames[int(face_id_idx)] # 取出人臉的名字
            bb_color = vis_utils.STANDARD_COLORS[i] # 給予不同的顏色
            bb_fontcolor = 'black'
        else:
            face_id_name = 'Unknown'
            bb_color = 'BlueViolet' # 給予紅色
            bb_fontcolor = 'white'

        # 進行視覺化的展現
        vis_utils.draw_bounding_box_on_image_array(draw,y1,x1,y2,x2,
                                                   color=bb_color,
                                                   thickness=2,
                                                   display_str_list=[face_id_name],
                                                   fontname='calibrib.ttf',         # <-- 替換不同的字型
                                                   fontsize=15,                     # <-- 根據圖像大小設定字型大小
                                                   fontcolor=bb_fontcolor,
                                                   use_normalized_coordinates=False)
else:
    print('Unable to align')

print('Detected_FaceNum: %d' % nrof_faces)
```

```
# 設定展示的大小
plt.figure(figsize=(20,10))

# 展示偵測出來的結果
plt.imshow(draw) # 轉換成RGB來給matplotlib展示
plt.show()
```

```
Start Recognition!
Image:  [452 800]
faces#0
(441, 65) : (511, 161)
faces#1
(76, 87) : (171, 215)
faces#2
(14, 125) : (41, 160)
faces#3
(617, 50) : (704, 176)
faces#4
(261, 47) : (328, 157)
Detected_FaceNum: 5
```



從以上的結果來看, 我們辨識出5張人臉。其中"Barack Obama(歐巴馬)"並不在我們從lfw的人臉資料庫中選出423個人臉的類別裡頭。所以從驗證結果來看, 這個人臉的識別效果還不錯。

參考:

- Facenet-Github (davidsandberg/facenet) - https://github.com/davidsandberg/facenet
- OpenFace-Github (cmusatyalab/openface) - https://github.com/cmusatyalab/openface
- Multi-task Cascaded Convolutional Networks論文 - https://arxiv.org/abs/1604.02878

- FaceNet: A Unified Embedding for Face Recognition and Clustering論文 - https://arxiv.org/abs/1503.03832

STEP 8. 進行人臉識別(2)

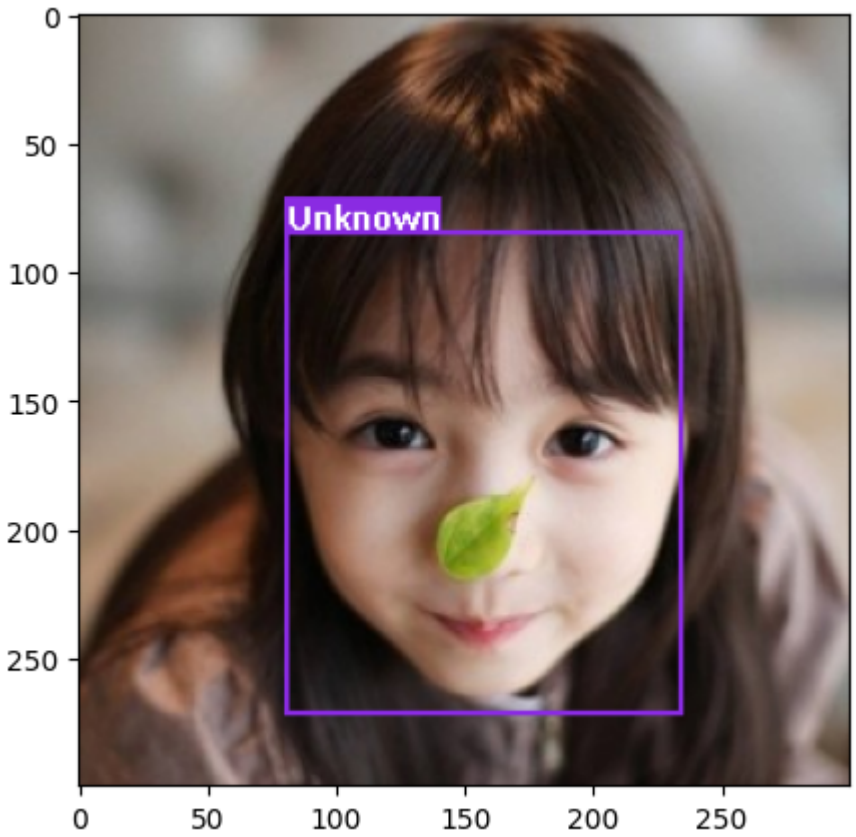
1.2 請展示有別人pre-trained好的Keras model可以成功辨認girl.jpg為人臉 (4 points)

In [ ]:
```
from skimage.transform import resize
print('Start Recognition!')
from PIL import Image, ImageDraw
face_input = "data/test/girl.jpg"

find_results = []
image = Image.open(face_input).convert('RGB')
frame = np.array(image)
bounding_boxes, _ = detect_face.detect_face(frame, minsize, pnet, rnet, onet, threshold, factor)

draw = frame.copy() # 複製原圖像
# Detect faces in the frame

# 步驟 #1.偵測人臉位置
# 偵測人臉的邊界框

nrof_faces = bounding_boxes.shape[0] # 被偵測到的臉部總數
if nrof_faces > 0: # 如果有偵測到人臉
    # 每一個 bounding_box包括了(x1,y1,x2,y2,confidence score):
    #     左上角座標 (x1,y1)
    #     右下角座標 (x2,y2)
    #     信心分數 confidence score
    det = bounding_boxes[:, 0:4].astype(int) # 取出邊界框座標
    img_size = np.asarray(frame.shape)[0:2] # 原圖像大小 (height, width)

    print("Image: ", img_size)

    # 人臉圖像前處理的暫存
    cropped = []
    scaled = []
    scaled_reshape = []
    bb = np.zeros((nrof_faces,4), dtype=np.int32)

    # 步驟 #2.擷取人臉特徵
    for i in range(nrof_faces):
        print("faces#{}".format(i))
        emb_array = np.zeros((1, embedding_size))
```

```
x1 = bb[i][0] = det[i][0]
y1 = bb[i][1] = det[i][1]
x2 = bb[i][2] = det[i][2]
y2 = bb[i][3] = det[i][3]

print('({}, {}) : ({}, {})'.format(x1,y1,x2,y2))
# inner exception
if bb[i][0] <= 0 or bb[i][1] <= 0 or bb[i][2] >= len(frame[0]) or bb[i][3] >= len(frame):
    print('face is out of range!')
    continue

# **人臉圖像的前處理 **

# 根據邊界框的座標來進行人臉的裁剪
cropped.append(frame[bb[i][1]:bb[i][3], bb[i][0]:bb[i][2], :])
cropped[i] = facenet.flip(cropped[i], False)
scaled.append(resize(cropped[i], (image_size, image_size)))
scaled[i] = cv2.resize(scaled[i], (input_image_size,input_image_size),
                       interpolation=cv2.INTER_CUBIC)
scaled[i] = facenet.prewhiten(scaled[i])
scaled_reshape.append(scaled[i].reshape(-1,input_image_size,input_image_size,3))
feed_dict = {images_placeholder: scaled_reshape[i], phase_train_placeholder: False}

# 進行臉部特徵擷取
# 進行臉部特徵擷取
emb_array[0, :] = tf_sess.run(embeddings, feed_dict=feed_dict)
emb_array = emb_array.reshape(-1)

# 步驟 #3.進行人臉識別分類
face_id_idx = face_svc_classifier.predict(emb_array.reshape(1, -1))

if is_same_person(emb_array, int(face_id_idx), 1.1):
    face_id_name = HumanNames[int(face_id_idx)] # 取出人臉的名字
    bb_color = vis_utils.STANDARD_COLORS[i] # 給予不同的顏色
    bb_fontcolor = 'black'
else:
    face_id_name = 'Unknown'
    bb_color = 'BlueViolet' # 給予紅色
    bb_fontcolor = 'white'

# 進行視覺化的展現
vis_utils.draw_bounding_box_on_image_array(draw,y1,x1,y2,x2,
                                           color=bb_color,
                                           thickness=2,
                                           display_str_list=[face_id_name],
                                           fontname='calibrib.ttf',         # <-- 替換不同的字型
```

```
                                                    fontsize=15,                    # <-- 根據圖像大小設定字型大小
                                                    fontcolor=bb_fontcolor,
                                                    use_normalized_coordinates=False)
else:
    print('Unable to align')

print('Detected_FaceNum: %d' % nrof_faces)

# 設定展示的大小
plt.figure(figsize=(10,5))

# 展示偵測出來的結果
plt.imshow(draw) # 轉換成RGB來給matplotlib展示
plt.show()
```

```
Start Recognition!
Image:  [300 300]
faces#0
(80, 85) : (234, 271)
Detected_FaceNum: 1
```



參考網站: https://github.com/erhwenkuo/face-recognition/blob/master/03-face-recognition.ipynb