使用 OpenCV 搭配 dlib 實作人臉偵測與辨識

教授：陳巧旻 博士　學生: 教碩一 華仁瑋

一、 研究動機

在幾年前 iphone 剛推出 face id 時，就有很多討論的聲音是關於雙胞胎或是臉部特徵相似的人，使用該項系統時，系統是否會辨識錯誤，導致手機使用上有安全的疑慮。實際上為了訓練 Face ID 的人臉辨識能力，蘋果團隊挑選來自全球上百萬張的臉部照片，使資料庫中擁有夠多跨種族、跨越膚色的人類圖像，用來提高臉部辨識準確度。Face ID 的運作方式如下：

1. 使用者每次面對鏡頭，「泛光感應元件」就會開始偵測臉部，而且在黑暗中也能進行
2. 紅外光攝影機捕捉影像，點陣投射器在使用者臉上投射 3 萬多個紅外光點
3. 紅外光影像和點陣圖形會在神經網路建構出使用者臉部的數學模型，這款神經網路採用 A11 仿生晶片，專門用於處理 Face ID
4. 運算過程中，手機會將使用者臉部的資訊傳送至處理器進行檢測，比對是否和裝置內儲存的臉部特徵一致
5. 最後若兩者結果相符，使用者的身分就得到驗證，手機就能解鎖

以上這些動作，都在不到一秒的時間之內運作完成，即使有許多網友進行了刁難的測驗，Face ID 通常還是能在三秒內辨識並且驗證。(Dr.A，2017)

這次剛好可以藉由這門課的機會，可以試試簡易版的人臉辨識系統，並希望能透過這次的嘗試，能更瞭解人臉辨識背後的模型與原理。

二、 研究工具

因為人練辨識系統模型的訓練較麻煩，因此這次是使用以訓練好的模型進行實作，主要使用的函式與模型分別為:

1. Open Source Computer Vision Library(OpenCV):

該庫擁有超過 2500 個優化算法，其中包括一整套經典和最先進的計算機視覺和機器學習算法。這些算法可用於檢測和識別人臉、識別物體、對視頻中的人類動作進行分類、跟踪攝像機運動、跟踪移動物體、提取物體的 3D 模型、從立體攝像機生成 3D 點雲、將圖像拼接在一起以生成高分辨率圖像。整個場景的圖像，從圖像數據庫中查找相似圖像，從使用閃光燈拍攝的圖像中消除紅眼，跟踪眼球運動，識別風景並建立標記以將其與增強現實疊加等(OpenCV)。

2. Dlib

dlib 是一套包含了機器學習、計算機視覺、圖像處理等的函式庫，使用 C++開發而成，目前廣泛使用於工業及學術界，也應用在機器人、嵌入式系統、手機、甚至於大型的運算架構中，而且最重要的是，它不但開源且完全免費，而且可跨平台使用（Linux、Mac OS、Windows），並且除了 C++之外還提供了 Python API ，因此如果我們想要建立一套物件偵測系統，dlib 是相當適合的平台(Tseng.CH，2016)。

3. shape_predictor_68_face_landmarks.dat

根據 Dlib 提供的 shape_predictor ()方法載入 68 個特徵點模型，此方法為人臉表情識別的偵測器。

4. dlib_face_recognition_resnet_model_v1.dat.bz2

該模型是一個具有 29 個卷積層的 ResNet 網絡。它本質上是 He、Zhang、Ren 和 Sun 所著的《圖像識別深度殘差學習》論文中的 ResNet-34 網絡的一個版本，其中刪除了幾層，並且每層的濾波器數量減少了一半。

該網絡是在大約 300 萬張面孔的數據集上從頭開始訓練的。該數據集源自多個數據集。The face scrub dataset、VGG 數據集，然後是網上抓取的大量圖片。通過刪除標籤錯誤來清理數據集，並反複訓練人臉識別 CNN，然後使用圖聚類方法和大量手動審查來清理數據集來做到這一點。最後大約一半的圖像來自 VGG 和 The face scrub dataset。此外，數據集中的個人身份總數為 7485。確保避免與 LFW 中的身份重疊。

網絡訓練從隨機初始化的權重開始，並使用結構化度量損失，嘗試將所有身份投影到半徑為 0.6 的非重疊球中。該損失基本上是一種成對鉸鏈損失，它運行在小批量中的所有對上，並包括小批量級別的硬負挖掘(Davisking)。

三、 實作結果

1. 人臉偵測

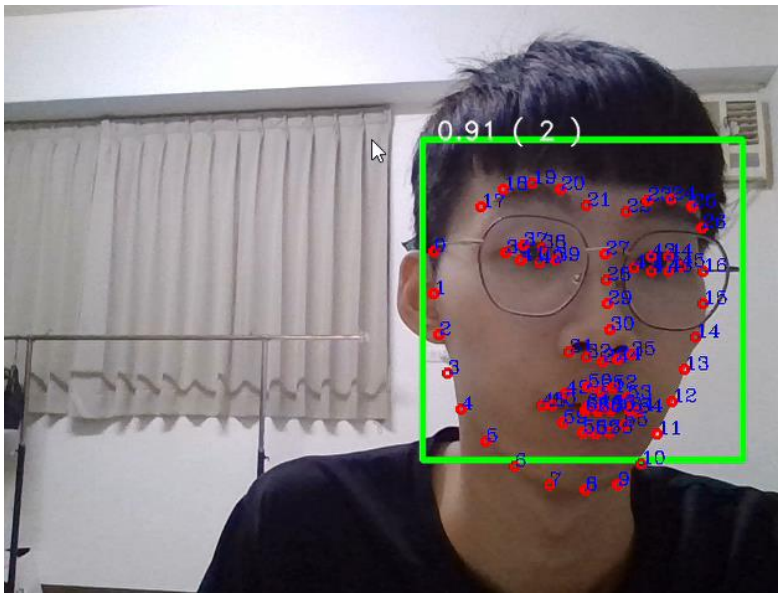
用 OpenCV 搭配 Dlib 這套 Machine learning 函式庫來實作人臉辨識，我們可以使用 Dlib 所提供的人臉辨識演算法來實作，數字為偵測到的分數，分數越高判斷為人臉的機率越大，而右邊括號內的數字為子偵測器的編號，也可以解釋為人臉的方向，0 就為正面，其他數字則為不同方向的編號。

- 程式碼

In [2]:
```
import dlib
import cv2
import imutils
```

In [3]:
```
#選擇第一隻攝影機
cap = cv2.VideoCapture(0)
#調整預設影像大小
cap.set(cv2. CAP_PROP_FRAME_WIDTH, 650)
cap.set(cv2. CAP_PROP_FRAME_HEIGHT, 500)

#取得預設的臉部偵測器
detector = dlib.get_frontal_face_detector()
#根據shape_predictor方法載入68個特徵點模型，此方法為人臉表情識別的偵測器
predictor = dlib.shape_predictor("C:\shape_predictor_68_face_landmarks.dat")
  #當攝影機打開時，對每個frame進行偵測
while(cap.isOpened()):
    #讀出frame資訊
    ret, frame = cap.read()

    #偵測人臉
    face_rects, scores, idx = detector.run(frame, 0)

    #取出偵測的結果
    for i, d in enumerate(face_rects):
      x1 = d.left()
      y1 = d.top()
      x2 = d.right()
      y2 = d.bottom()
      text = " %2.2f ( %d )" % (scores[i], idx[i])

      #繪製出偵測人臉的矩形範圍
      cv2.rectangle(frame, (x1, y1), (x2, y2), ( 0, 255, 0), 4, cv2. LINE_AA)

      #標上人臉偵測分數與人臉方向子偵測器編號
      cv2.putText(frame, text, (x1, y1), cv2. FONT_HERSHEY_DUPLEX,
      0.7, ( 255, 255, 255), 1, cv2. LINE_AA)

      #給68特徵點辨識取得一個轉換顏色的frame
      landmarks_frame = cv2.cvtColor(frame, cv2. COLOR_BGR2RGB)

      #找出特徵點位置
      shape = predictor(landmarks_frame, d)

      #繪製68個特徵點
      for i in range( 68):
        cv2.circle(frame,(shape.part(i).x,shape.part(i).y), 3,( 0, 0, 255), 2)
        cv2.putText(frame, str(i),(shape.part(i).x,shape.part(i).y),cv2. FONT_HERSHEY_COMPLEX, 0.5,( 255, 0, 0), 1)
    #輸出到畫面
    cv2.imshow( "Face Detection", frame)

    #如果按下ESC鍵，就退出
    if cv2.waitKey( 10) == 27:
        break
#釋放記憶體
cap.release()
#關閉所有視窗
cv2.destroyAllWindows()
```

- 成品



2. 人臉辨識

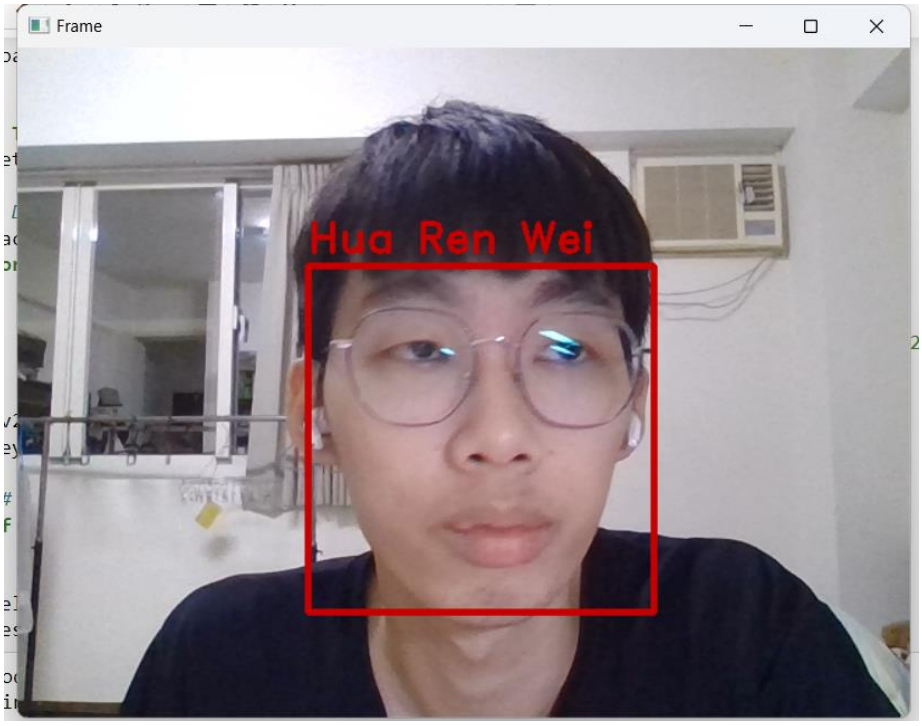
實時人臉識別，其過程與單個圖像的過程類似，但有更多功能。首先，我們通過一個簡單的 Opencv 函數，使用電腦攝影機並重複循環。接著，透過電腦攝影機識別面部，並將影像傳遞給函式 detector_known_faces(frame)。它會給我們這個人的名字和一個數組，其中包含每個動作時刻的位置。最後，剪裁輸出的影像大小，並將照片或人像置於鏡頭前，使系統辨識。

- 程式碼

In [1]:
```
import cv2
from simple_facerec import SimpleFacerec

# Load Camera
cap = cv2.VideoCapture(0)

#Encode faces from a folder
sfr = SimpleFacerec()
sfr.load_encoding_images("C:\images")


while True:
    ret, frame = cap.read()

    # Detect Faces
    face_locations, face_names = sfr.detect_known_faces(frame)
    for face_loc, name in zip(face_locations, face_names):
        y1, x2, y2, x1 = face_loc[0], face_loc[1], face_loc[2], face_loc[3]

        cv2.putText(frame, name,(x1, y1 - 10), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 200), 2)
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 200), 4)

    cv2.imshow('Frame', frame)
    key = cv2.waitKey(1)

     # 若按下 q 鍵則離開迴圈
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

```
6 encoding images found.
Encoding images loaded
```

- 成品



四、 結論與建議

透過這次的實作，讓我發現到在人臉偵測中，模型辨識臉部特徵點的重要性，如同 iphone 的 300 萬個特徵點都還有可能出錯，更何況是區區 68 個特徵點，在辨識的過程中，我發現只要臉部特徵差異不大，系統就容易辨識錯誤，因此後來有將其餘的辨識圖片調整為外國人，才成功辨識出正確的人臉與姓名。

因此如果下次有機會再做這類的實作，我會嘗試使用多一點的臉部特徵點，或是嘗試自己訓練模型，做出不一樣的挑戰。

五、 參考文獻

Face ID 原理大解密！iPhone X 就靠它再創高峰。取自 https://www.dra-3c.com/article/iphone-knowledge/face-id-decryption

Dlib 好用的的 Machine learning 工具（一）。取自 https://reurl.cc/v7ab4j

OpenCV。取自 https://opencv.org/about/

dlib-模型。取自 https://github.com/davisking/dlib-models

學習 Python3 Dlib19.7 進行人臉面部識別 https://www.itread01.com/article/1516763865.html

基於 python 語言使用 OpenCV 搭配 dlib 實作人臉偵測與辨識 https://www.tpisoftware.com/tpu/articleDetails/950

使用 OpenCV 和 Python 進行實時人臉識別 https://pysource.com/2021/08/16/face-recognition-in-real-time-with-opencv-and-python/