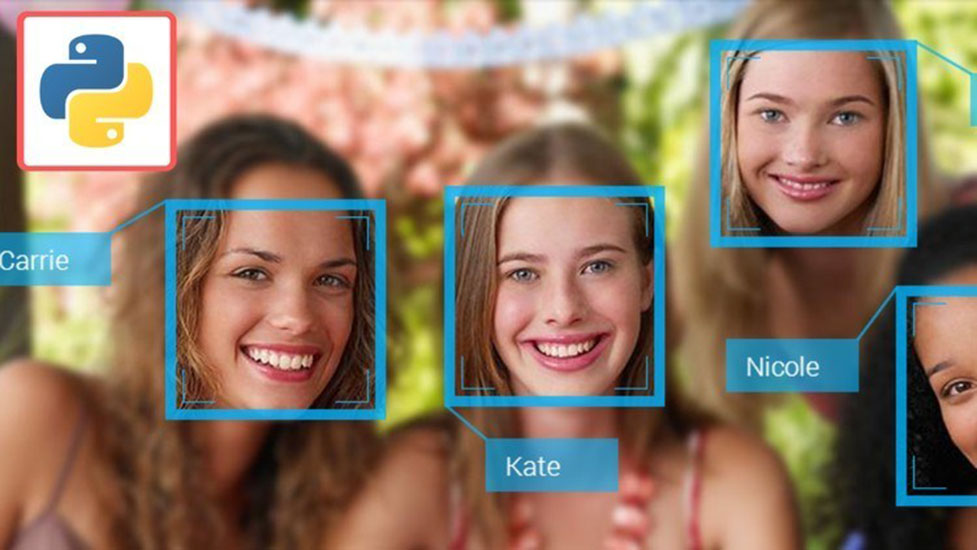
GIẢI THÍCH NHẬN DIỆN KHUÔN MẶT VÀ CẢM XÚC SỬ DỤNG RASPBERRY PI



**MỤC LỤC**

[**I. NHẬN DIỆN KHUÔN MẶT**](#_tttzf0ofak4o) **3**

[1. Mục đích của đoạn mã](#_735s9c5km5z8) 3

[2. Cấu trúc thư mục](#_j7mff9dana33) 3

[3. Cài đặt các gói cần thiết](#_fzytuq9hphev) 3

[4. Thư viện sử dụng](#_tom31icp5tff) 4

[5. Quy trình chính](#_jb46ipu0cb0q) 4

[5.1. Đọc nội dung của hình ảnh](#_sr0xaw9nyixs) 5

[5.2. Tạo đối tượng hình ảnh](#_qirjpqp5s53c) 5

[5.3. Gửi hình ảnh đến Google Cloud Vision API](#_gbv1q3csu7ip) 5

[5.4. Trích xuất thông tin khuôn mặt](#_c5i5xuxo07pk) 5

[5.5. Chuẩn bị tệp hình ảnh kết quả](#_hz8ye88guhk1) 5

[5.6. In thông tin kết quả](#_47whbadtt9bj) 5

[5.7. Đọc hình ảnh gốc](#_97qat7vww3je) 6

[5.8. Vẽ hình chữ nhật xung quanh khuôn mặt](#_vjzakjy3wb8f) 6

[5.9. Lưu hình ảnh kết quả](#_959werekrmr8) 6

[6. Sử dụng mã](#_zhcktt66iwpu) 6

[7. Chi tiết mã nguồn](#_tbehvoybyr40) 7

[**II. NHẬN DIỆN CẢM XÚC**](#_p3vnh071az85) **9**

[1. Mục đích của đoạn mã](#_tersy9qg0pus) 9

[2. Cấu trúc thư mục](#_ytsexeksh9po) 9

[3. Cài đặt các gói cần thiết](#_gmuza5ib50wr) 10

[4. Thư viện sử dụng](#_dgxtht36l4r4) 10

[5. Quy trình chính](#_k3diw317vody) 10

[5.1. Đọc nội dung của hình ảnh](#_57t6xn4252sw) 11

[5.2. Tạo đối tượng hình ảnh](#_c6fvqj8q06vm) 11

[5.3. Gửi hình ảnh đến Google Cloud Vision API](#_2axrp9qoj5fw) 11

[5.4. Trích xuất thông tin khuôn mặt](#_acgfj2jn5m77) 11

[5.5. Khởi tạo danh sách các cảm xúc](#_3kpasarl39iw) 11

[5.6. In thông tin kết quả](#_pdtunhb74gau) 12

[6. Sử dụng mã](#_ph4mcrd76bep) 12

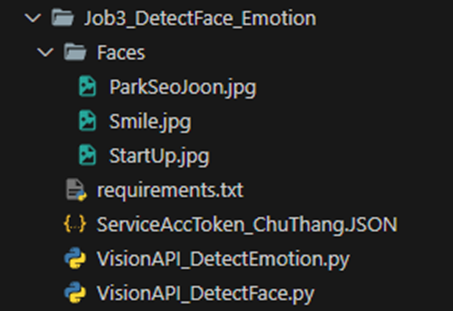
[7. Chi tiết mã nguồn](#_8a7axw3c2r) 14

# NHẬN DIỆN KHUÔN MẶT

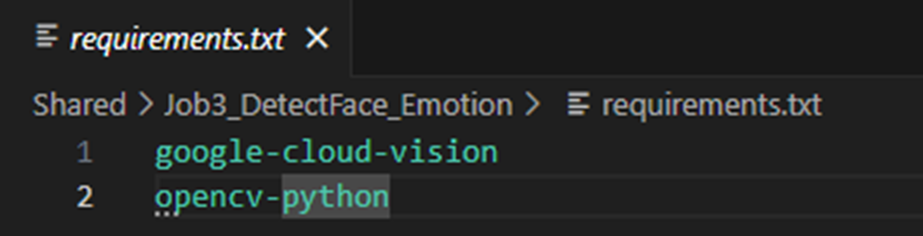
## Mục đích của đoạn mã

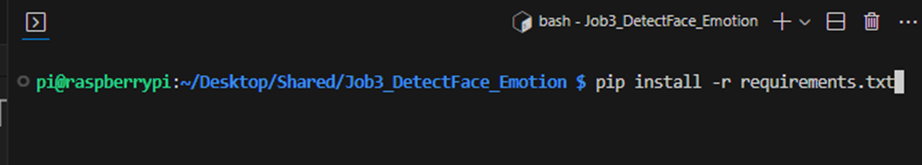
Đoạn mã Python này sử dụng Google Cloud Vision API để phát hiện các khuôn mặt trong một hình ảnh và sau đó vẽ hình vuông xung quanh từng khuôn mặt được phát hiện. Kết quả cuối cùng được lưu trong một hình ảnh mới.

## Cấu trúc thư mục

****

## Cài đặt các gói cần thiết

Chúng ta cần cài đặt các gói cần thiết như sau. Trước hết ta sẽ để những gói cần thiết vào 1 file mang tên **requirements.txt**. Trong tệp cần có tên các gói như hình dưới  


Sau đó sẽ gõ lệnh sau đây trên terminal: **pip install -r requirements.txt**

## Thư viện sử dụng

* **cv2 (OpenCV):** Thư viện này được sử dụng để xử lý hình ảnh, vẽ hình vuông và lưu kết quả.
* **os** : Thư viện này được sử dụng để làm việc với hệ thống tệp và thư mục.
* **io** : Thư viện này được sử dụng để đọc nội dung của tệp hình ảnh.
* **numpy (as np)**: Thư viện này được sử dụng để xử lý dữ liệu số học, trong trường hợp này là để xử lý các điểm cụ thể của khuôn mặt.
* **google.cloud.vision\_v1**: Thư viện này chứa các công cụ cho việc gửi hình ảnh đến Google Cloud Vision API và nhận kết quả.

## Quy trình chính

* Đoạn mã bắt đầu bằng việc thiết lập đường dẫn đến tệp JSON chứa thông tin xác thực dịch vụ của Google Cloud Vision API. Đường dẫn này được lưu trong biến `key\_path`, sau đó được đặt trong biến môi trường `GOOGLE\_APPLICATION\_CREDENTIALS` để xác thực với API.

|  |
| --- |
| key\_path = '/home/pi/Desktop/Shared/GoogleVisionAPI/ServiceAccToken\_ChuThang.JSON'  os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = key\_path |

* Sau đó, đoạn mã tạo một phiên bản của `**ImageAnnotatorClient**` để gửi hình ảnh đến Google Cloud Vision API và nhận kết quả.

|  |
| --- |
| client = vision\_v1.ImageAnnotatorClient() |

* Hàm **detectEmotion(FILE\_PATH)** được định nghĩa để thực hiện công việc phát hiện khuôn mặt và vẽ hình vuông xung quanh chúng. Hàm này nhận đối số `FILE\_PATH`, đó là đường dẫn đến tệp hình ảnh cần xử lý.

|  |
| --- |
| def detectFace(FILE\_PATH): |

### 5.1. Đọc nội dung của hình ảnh

Hàm bắt đầu bằng việc mở tệp hình ảnh từ đường dẫn **FILE\_PATH** và đọc nội dung của nó bằng cách sử dụng io.open trong chế độ đọc nhị phân ('rb'). Nội dung hình ảnh được lưu trong biến content.

|  |
| --- |
| with io.open(FILE\_PATH, 'rb') as image\_file:  content = image\_file.read() |

### 5.2. Tạo đối tượng hình ảnh

image = vision\_v1.Image(content=content): Đối tượng hình ảnh image được tạo ra từ dữ liệu hình ảnh content

|  |
| --- |
| image = vision\_v1.Image(content=content) |

### 5.3. Gửi hình ảnh đến Google Cloud Vision API

Hàm sử dụng **client.face\_detection(image=image)** để gửi đối tượng hình ảnh đã chuẩn bị đến Google Cloud Vision API và yêu cầu phân tích khuôn mặt trong hình ảnh. Kết quả phân tích được lưu trong biến response.

|  |
| --- |
| response = client.face\_detection(image=image) |

### 5.4. Trích xuất thông tin khuôn mặt

Từ kết quả trả về (**response**), hàm trích xuất danh sách các khuôn mặt được phát hiện và lưu chúng trong biến faces. Mỗi khuôn mặt được biểu diễn bằng một đối tượng **face\_annotations**.

|  |
| --- |
| faces = response.face\_annotations |

### 5.5. Chuẩn bị tệp hình ảnh kết quả

Hàm đặt tên cho tệp hình ảnh kết quả là "**out.jpg**" bằng cách gán giá trị cho biến **output\_file**.

|  |
| --- |
| output\_file = "out.jpg" |

### 5.6. In thông tin kết quả

Hàm in ra số lượng khuôn mặt được phát hiện trong hình ảnh và thông báo rằng nó sẽ được lưu vào tệp "**out.jpg**".

|  |
| --- |
| print(f"Found {len(faces)} face.")  print(f"Writing to file {output\_file}") |

### 5.7. Đọc hình ảnh gốc

Hàm đọc tệp hình ảnh gốc bằng **cv2.imread(FILE\_PATH)** để chuẩn bị cho việc vẽ hình chữ nhật xung quanh các khuôn mặt trên hình ảnh gốc. Hình ảnh này được lưu trong biến **image\_cv2**.

|  |
| --- |
| image\_cv2 = cv2.imread(FILE\_PATH) |

### 5.8. Vẽ hình chữ nhật xung quanh khuôn mặt

Hàm duyệt qua danh sách **faces** (các khuôn mặt được phát hiện) và trích xuất tọa độ của các đỉnh của hình chữ nhật bao quanh từng khuôn mặt. Sau đó, nó sử dụng **cv2.rectangle** để vẽ hình chữ nhật xung quanh từng khuôn mặt trên hình ảnh gốc (**image\_cv2**).

|  |
| --- |
| for face in faces:  vertices = [(vertex.x, vertex.y) for vertex in face.bounding\_poly.vertices]  x, y, width, height = cv2.boundingRect(np.array(vertices))  out\_vertice = ', '.join(map(str, vertices))  print(f"Face bounds: {out\_vertice}")  # Vẽ hình vuông  cv2.rectangle(image\_cv2, (x, y), (x + width, y + height), (0, 255, 0), 2) |

### 5.9. Lưu hình ảnh kết quả

Cuối cùng, hàm sử dụng **cv2.imwrite(output\_file, image\_cv2)** để lưu hình ảnh gốc với các hình chữ nhật đã vẽ xung quanh các khuôn mặt vào tệp "**out.jpg**".

|  |
| --- |
| cv2.imwrite(output\_file, image\_cv2) |

## Sử dụng mã

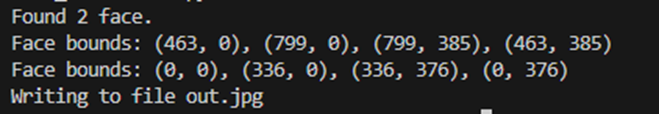
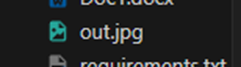
* Để sử dụng đoạn mã này, bạn cần cung cấp đường dẫn đến một tệp hình ảnh muốn xử lý bằng cách đặt đường dẫn này vào biến `image\_path`.

|  |
| --- |
| image\_path = './Faces/ParkSeoJoon.jpg' |

* Sau đó, gọi hàm `detectEmotion(image\_path)` để thực hiện phát hiện khuôn mặt và vẽ hình vuông.

|  |
| --- |
| detectFace(image\_path) |

* Kết quả sau cùng sẽ là một tệp `out.jpg` chứa hình ảnh ban đầu với hình vuông vẽ xung quanh các khuôn mặt được phát hiện.



## Chi tiết mã nguồn

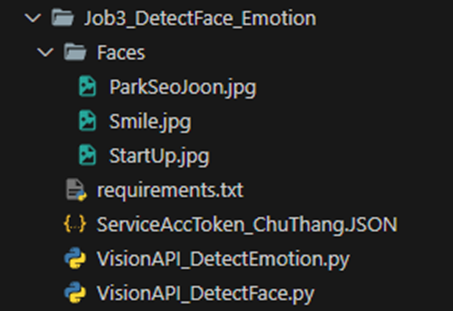
|  |
| --- |
| import cv2  import os, io  import numpy as np  from google.cloud import vision\_v1  # Thiết lập đường dẫn đến tệp JSON chứa thông tin xác thực dịch vụ của Google Cloud Vision API  key\_path = '/home/pi/Desktop/Shared/GoogleVisionAPI/ServiceAccToken\_ChuThang.JSON'  # Đặt biến môi trường để xác thực với Google Cloud Vision API  os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = key\_path  # Tạo một phiên bản của ImageAnnotatorClient để gửi hình ảnh đến Google Cloud Vision API  client = vision\_v1.ImageAnnotatorClient()  # Định nghĩa hàm để phát hiện khuôn mặt và vẽ hình chữ nhật xung quanh chúng  def detectFace(FILE\_PATH):  # Đọc nội dung của tệp hình ảnh  with io.open(FILE\_PATH, 'rb') as image\_file:  content = image\_file.read()    # Tạo đối tượng hình ảnh từ nội dung  image = vision\_v1.Image(content=content)    # Gửi hình ảnh đến Google Cloud Vision API để phát hiện khuôn mặt  response = client.face\_detection(image=image)    # Lấy danh sách các khuôn mặt được phát hiện  faces = response.face\_annotations    # Đặt tên cho tệp hình ảnh kết quả  output\_file = "out.jpg"    # In số lượng khuôn mặt được phát hiện  print(f"Found {len(faces)} face.")    # In tên của tệp hình ảnh kết quả  print(f"Writing to file {output\_file}")    # Đọc hình ảnh ban đầu bằng OpenCV  image\_cv2 = cv2.imread(FILE\_PATH)    # Vòng lặp qua từng khuôn mặt và vẽ hình chữ nhật xung quanh chúng  for face in faces:  vertices = [(vertex.x, vertex.y) for vertex in face.bounding\_poly.vertices]  x, y, width, height = cv2.boundingRect(np.array(vertices))  out\_vertice = ', '.join(map(str, vertices))  print(f"Face bounds: {out\_vertice}")  # Vẽ hình chữ nhật  cv2.rectangle(image\_cv2, (x, y), (x + width, y + height), (0, 255, 0), 2)    # Lưu hình ảnh kết quả có hình chữ nhật vào tệp "out.jpg"  cv2.imwrite(output\_file, image\_cv2)  # Đường dẫn đến tệp hình ảnh cần xử lý  image\_path = './Faces/ParkSeoJoon.jpg'  # Gọi hàm để phát hiện khuôn mặt và vẽ hình chữ nhật  detectFace(image\_path) |

# NHẬN DIỆN CẢM XÚC

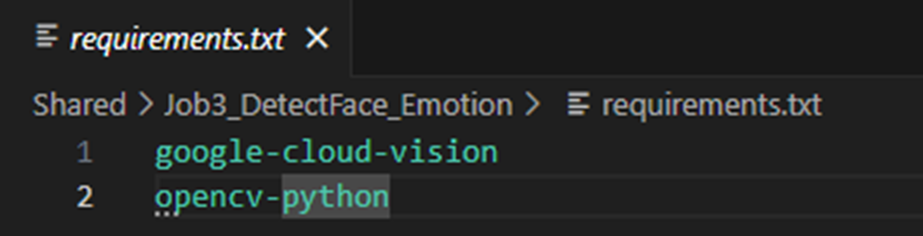
## Mục đích của đoạn mã

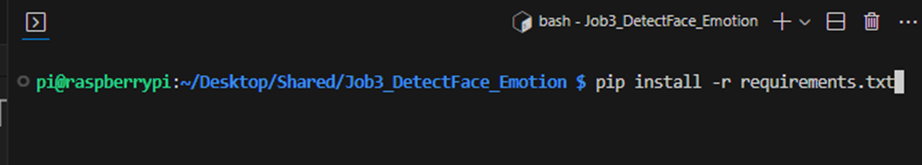
Đoạn mã Python này sử dụng Google Cloud Vision API để phân tích một hình ảnh và phát hiện các khuôn mặt trong hình ảnh đó. Sau đó, nó đánh giá cảm xúc trên mỗi khuôn mặt, như cảm giác tức giận (**angry**), vui vẻ (**joy**), buồn (**sorrow**), ngạc nhiên (**surprise**), và việc đội mũ (**headwear**). Kết quả cuối cùng là việc hiển thị thông tin về cảm xúc trên mỗi khuôn mặt trong hình ảnh.

## Cấu trúc thư mục



## Cài đặt các gói cần thiết

Chúng ta cần cài đặt các gói cần thiết như sau. Trước hết ta sẽ để những gói cần thiết vào 1 file mang tên **requirements.txt**. Trong tệp cần có tên các gói như hình dưới  


Sau đó sẽ gõ lệnh sau đây trên terminal: **pip install -r requirements.txt**

## Thư viện sử dụng

* **os** : Thư viện này được sử dụng để làm việc với hệ thống tệp và thư mục.
* **io** : Thư viện này được sử dụng để đọc nội dung của tệp hình ảnh.
* **google.cloud.vision\_v1**: Thư viện này chứa các công cụ cho việc gửi hình ảnh đến Google Cloud Vision API và nhận kết quả.

## Quy trình chính

* Đoạn mã bắt đầu bằng việc thiết lập đường dẫn đến tệp JSON chứa thông tin xác thực dịch vụ của Google Cloud Vision API. Đường dẫn này được lưu trong biến `key\_path`, sau đó được đặt trong biến môi trường `GOOGLE\_APPLICATION\_CREDENTIALS` để xác thực với API.

|  |
| --- |
| key\_path = '/home/pi/Desktop/Shared/GoogleVisionAPI/ServiceAccToken\_ChuThang.JSON'  os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = key\_path |

* Sau đó, đoạn mã tạo một phiên bản của `**ImageAnnotatorClient**` để gửi hình ảnh đến Google Cloud Vision API và nhận kết quả.

|  |
| --- |
| client = vision\_v1.ImageAnnotatorClient() |

* Hàm **detectEmotion(FILE\_PATH)** được định nghĩa để thực hiện công việc phát hiện khuôn mặt và vẽ hình vuông xung quanh chúng. Hàm này nhận đối số `FILE\_PATH`, đó là đường dẫn đến tệp hình ảnh cần xử lý.

|  |
| --- |
| def detectFace(FILE\_PATH): |

### 5.1. Đọc nội dung của hình ảnh

Hàm bắt đầu bằng việc mở tệp hình ảnh từ đường dẫn **FILE\_PATH** và đọc nội dung của nó bằng cách sử dụng io.open trong chế độ đọc nhị phân (**'rb'**). Nội dung hình ảnh được lưu trong biến content.

|  |
| --- |
| with io.open(FILE\_PATH, 'rb') as image\_file:  content = image\_file.read() |

### 5.2. Tạo đối tượng hình ảnh

Sử dụng **vision\_v1.Image** để tạo một đối tượng hình ảnh từ nội dung của tấm hình. Điều này chuẩn bị dữ liệu hình ảnh để được gửi đến Google Cloud Vision API để phân tích

|  |
| --- |
| image = vision\_v1.Image(content=content) |

### 5.3. Gửi hình ảnh đến Google Cloud Vision API

Hàm sử dụng **client.face\_detection(image=image)** để gửi đối tượng hình ảnh đã chuẩn bị đến Google Cloud Vision API và yêu cầu phân tích khuôn mặt trong hình ảnh. Kết quả phân tích được lưu trong biến response.

|  |
| --- |
| response = client.face\_detection(image=image) |

### 5.4. Trích xuất thông tin khuôn mặt

Từ kết quả trả về (**response**), hàm trích xuất danh sách các khuôn mặt được phát hiện và lưu chúng trong biến faces. Mỗi khuôn mặt được biểu diễn bằng một đối tượng **face\_annotations**.

|  |
| --- |
| faces = response.face\_annotations |

### 5.5. Khởi tạo danh sách các cảm xúc

Hàm khởi tạo một danh sách chứa các nhãn cảm xúc từ 'UNKNOWN' đến 'VERY LIKELY'. Biến likehood chứa danh sách này.

|  |
| --- |
| likehood = ('UNKNOWN', 'VERY UNLIKELY', 'UNLIKELY', 'POSSIBLY', 'LIKELY', 'VERY LIKELY') |

### 5.6. In thông tin kết quả

Hàm bắt đầu bằng việc in ra số lượng khuôn mặt được phát hiện trong hình ảnh bằng cách sử dụng len(**faceAnnotation**). Sau đó, nó duyệt qua từng khuôn mặt trong biến **faceAnnotation** và in ra thông tin về cảm xúc trên từng khuôn mặt.

* **Detection confidence**: Đây là tỷ lệ xác định khuôn mặt, được làm tròn và hiển thị dưới dạng phần trăm (ví dụ: 95.0%).
* **Angry**, **Joy**, **Sorrow**, **Surprise**, **Headwear**: Đây là các cảm xúc hoặc sự xuất hiện của đội mũ trên khuôn mặt, được hiển thị dựa trên nhãn từ danh sách **likehood**.

|  |
| --- |
| print(f"Found {len(faceAnnotation)} face.")  for face in faceAnnotation:  print(f"Faces:")  print(f' Detection confidence : {round(face.detection\_confidence \* 100, 2)}')  print(f' Angry : {likehood[face.anger\_likelihood]}')  print(f' Joy : {likehood[face.joy\_likelihood]}')  print(f' Sorrow : {likehood[face.sorrow\_likelihood]}')  print(f' Sup : {likehood[face.surprise\_likelihood]}')  print(f' Headwear : {likehood[face.headwear\_likelihood]}\n') |

## Sử dụng mã

* Để sử dụng đoạn mã này, bạn cần cung cấp đường dẫn đến một tệp hình ảnh muốn xử lý bằng cách đặt đường dẫn này vào biến `file\_path`.

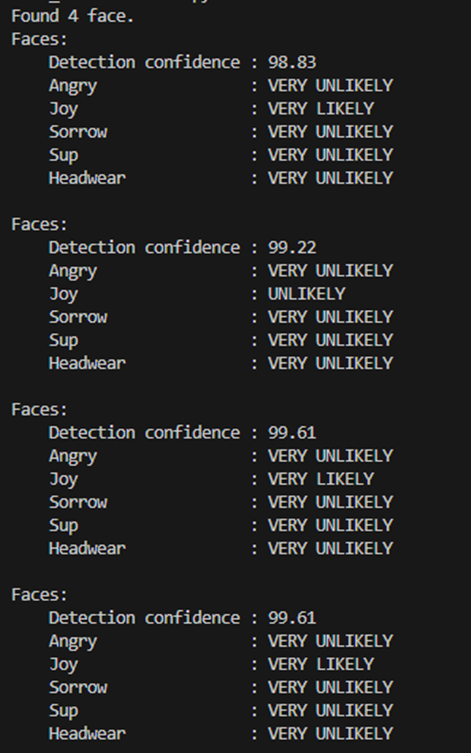
|  |
| --- |
| img\_name = 'StartUp.jpg'  file\_path = f'./Faces/{img\_name}' |

* Sau đó, gọi hàm ` detectEmotion(file\_path) để thực hiện phát hiện khuôn mặt và vẽ hình vuông.

|  |
| --- |
| detectEmotion(file\_path) |

* Kết quả sau cùng sẽ như sau:





## Chi tiết mã nguồn

|  |
| --- |
| # Import các thư viện cần thiết  import os, io  from google.cloud import vision\_v1  # Thiết lập đường dẫn đến tệp JSON chứa thông tin xác thực dịch vụ của Google Cloud Vision API  key\_path = '/home/pi/Desktop/Shared/GoogleVisionAPI/ServiceAccToken\_ChuThang.JSON'  # Đặt biến môi trường để xác thực với Google Cloud Vision API  os.environ['GOOGLE\_APPLICATION\_CREDENTIALS'] = key\_path  # Tạo một phiên bản của ImageAnnotatorClient để gửi hình ảnh đến Google Cloud Vision API  client = vision\_v1.ImageAnnotatorClient()  # Định nghĩa hàm để phân tích cảm xúc trong khuôn mặt  def detectEmotion(FILE\_PATH):  with io.open(FILE\_PATH, 'rb') as image\_file:  content = image\_file.read()  image = vision\_v1.Image(content=content)  response = client.face\_detection(image=image)  faceAnnotation = response.face\_annotations  likehood = ('UNKNOWN', 'VERY UNLIKELY', 'UNLIKELY', 'POSSIBLY', 'LIKELY', 'VERY LIKELY')  print(f"Found {len(faceAnnotation)} face.")  for face in faceAnnotation:  print(f"Faces:")  print(f' Detection confidence : {round(face.detection\_confidence \* 100, 2)}')  print(f' Angry : {likehood[face.anger\_likelihood]}')  print(f' Joy : {likehood[face.joy\_likelihood]}')  print(f' Sorrow : {likehood[face.sorrow\_likelihood]}')  print(f' Sup : {likehood[face.surprise\_likelihood]}')  print(f' Headwear : {likehood[face.headwear\_likelihood]}')  print()  # Tên hình ảnh cần xử lý  img\_name = 'StartUp.jpg'  # Đường dẫn đến tệp hình ảnh  file\_path = f'./Faces/{img\_name}'  # Gọi hàm để phân tích cảm xúc trong khuôn mặt  detectEmotion(file\_path) |