**BÁO CÁO NHẬN DẠNG KHUÔN MẶT VÀ TÊN THẬT BẰNG CNN**

**2a. Giới thiệu**

- ***Mục tiêu:*** Bài báo cáo này tập trung vào việc nhận dạng khuôn mặt và dự đoán tên sử dụng thư viện face\_recognition, một thư viện mã nguồn mở phổ biến trong lĩnh vực xử lý ảnh và thị giác máy tính. Mục tiêu của đề tài là xây dựng một ứng dụng đơn giản để phát hiện khuôn mặt và đoán tên trong thời gian thực từ camera.

- ***Phương pháp:*** Thư viện face\_recognition sử dụng một loạt các phương pháp tiên tiến trong học sâu (deep learning) để thực hiện các tác vụ nhận diện khuôn mặt. Dưới đây là các phương pháp chính được sử dụng bởi thư viện này:

* Phát hiện khuôn mặt (Face Detection)

Phát hiện khuôn mặt là bước đầu tiên trong quá trình nhận diện khuôn mặt. face\_recognition sử dụng phương pháp Histogram of Oriented Gradients (HOG) và một mạng nơ-ron tích chập (CNN) để phát hiện các khuôn mặt trong hình ảnh.

* Histogram of Oriented Gradients (HOG):

HOG là một phương pháp truyền thống để phát hiện các đối tượng. Nó sử dụng sự phân phối hướng của gradient trong một khung hình để phát hiện khuôn mặt. Phương pháp này nhanh chóng và nhẹ, nhưng có thể kém chính xác hơn so với các phương pháp hiện đại.

* Convolutional Neural Network (CNN):

CNN là một mạng nơ-ron tích chập, một loại mạng nơ-ron sâu được thiết kế để xử lý dữ liệu dạng lưới như hình ảnh. CNN được huấn luyện để phát hiện khuôn mặt với độ chính xác cao hơn HOG, nhưng đòi hỏi tài nguyên tính toán nhiều hơn.

* Mã hóa khuôn mặt (Face Encoding)

Sau khi phát hiện các khuôn mặt trong hình ảnh, bước tiếp theo là chuyển đổi mỗi khuôn mặt thành một vector đặc trưng, gọi là "face encoding". Vector này là đại diện toán học của các đặc điểm trên khuôn mặt và được sử dụng để so sánh các khuôn mặt với nhau.

* Deep Metric Learning:

face\_recognition sử dụng một mạng nơ-ron sâu, cụ thể là một mô hình được gọi là Deep Metric Learning. Mô hình này học cách ánh xạ các khuôn mặt thành các điểm trong không gian đa chiều sao cho các khuôn mặt giống nhau sẽ gần nhau và các khuôn mặt khác nhau sẽ cách xa nhau.

Mô hình này thường dựa trên các kiến trúc mạng nơ-ron như ResNet hoặc các biến thể của nó, đã được huấn luyện trên các tập dữ liệu lớn chứa hàng triệu khuôn mặt.

* So sánh khuôn mặt (Face Comparison)

Sau khi có được mã hóa khuôn mặt, face\_recognition có thể so sánh các mã hóa này để xác định xem hai khuôn mặt có phải là cùng một người hay không.

* Euclidean Distance:

Thư viện sử dụng khoảng cách Euclidean giữa các vector mã hóa khuôn mặt để xác định sự giống nhau. Nếu khoảng cách này nhỏ hơn một ngưỡng định trước (thường là 0.6), thì hai khuôn mặt được coi là giống nhau.

face\_recognition.compare\_faces so sánh mã hóa khuôn mặt mới với các mã hóa đã biết và trả về kết quả True/False dựa trên ngưỡng này.

* Xác định vị trí các đặc điểm trên khuôn mặt (Face Landmarks)

face\_recognition cũng có khả năng xác định các đặc điểm cụ thể trên khuôn mặt như mắt, mũi, miệng, và cằm.

* Facial Landmark Detection:

Quá trình này sử dụng các mô hình dự đoán vị trí của các điểm đặc trưng trên khuôn mặt. Thông thường, các mô hình này cũng được huấn luyện bằng deep learning và có thể xác định chính xác vị trí của các đặc điểm ngay cả khi khuôn mặt bị nghiêng hoặc có các biểu cảm khác nhau.

- ***Quá trình nhận diện khuôn mặt và đoán tên:***

* Khởi tạo và tải mã hóa khuôn mặt:

Tạo một đối tượng SimpleFacerec.

Gọi phương thức load\_encoding\_images với đường dẫn tới thư mục chứa ảnh của các khuôn mặt đã biết. Các ảnh sẽ được tải, mã hóa khuôn mặt sẽ được trích xuất và lưu trữ cùng với tên (dựa trên tên file ảnh).

* Phát hiện và nhận diện khuôn mặt trong video:

Mở một thiết bị capture video (webcam hoặc file video).

Liên tục đọc các khung hình từ luồng video.

Phát hiện các khuôn mặt trong từng khung hình bằng phương thức detect\_known\_faces.

Vẽ các hình chữ nhật xung quanh khuôn mặt được phát hiện và gán tên tương ứng.

* Hiển thị và thoát:

Hiển thị các khung hình với khuôn mặt và tên được phát hiện.

Thoát khỏi vòng lặp và giải phóng tài nguyên khi nhấn 'q'.

**2b. Cài đặt và chạy chương trình**

- ***Cài thư viện tải các model cần thiết:***

Các thư viện cần tải: OpenCv, numpy, face\_recognition, os, glob.

A black rectangle with white text

Description automatically generated

+ Trước khi import các thư viện trong máy phải tiến hành tải thư viện về trước bằng dòng lệnh “pip install face\_recognition” trong terminal của python để cài thư viện face\_recognition và tương tự với các thư viện còn lại.

**- Lớp SimpleFacerec**

Lớp này cung cấp các chức năng cho việc tải hình ảnh các khuôn mặt đã biết, mã hóa chúng, và nhận diện các khuôn mặt này trong các khung hình video mới.

* Phương thức khởi tạo

A black screen with green and red squares

Description automatically generated

* self.known\_face\_encodings: Một danh sách để lưu trữ các mã hóa khuôn mặt của các cá nhân đã biết. Mỗi mã hóa là một biểu diễn số của các đặc điểm khuôn mặt.
* self.known\_face\_names: Một danh sách để lưu trữ tên tương ứng với các mã hóa khuôn mặt. Chỉ số của mỗi tên tương ứng với chỉ số của mã hóa trong self.known\_face\_encodings.
* self.frame\_resizing: Một hệ số được sử dụng để thay đổi kích thước các khung hình video để xử lý nhanh hơn.
* Phương thức: load\_encoding\_images

A screen shot of a computer program

Description automatically generatedPhương thức này tải hình ảnh từ thư mục được chỉ định, mã hóa các khuôn mặt, và lưu trữ các mã hóa này cùng với tên tương ứng.

* Câu lệnh đầu tiên thu thập tất cả các tệp trong thư mục images\_path được chỉ định. Hàm glob.glob được sử dụng để tìm tất cả các đường dẫn tệp phù hợp với mẫu \*.\* (bất kỳ tệp nào).
* Câu lệnh print in số lượng hình ảnh được tìm thấy trong thư mục.
* Vòng lặp for duyệt qua mỗi đường dẫn tệp hình ảnh trong danh sách images\_path.
* Dùng cv2.imread để đọc hình ảnh từ đường dẫn và cv2.cvtColor để chuyển đổi không gian màu của hình ảnh từ BGR (mà OpenCV sử dụng) sang RGB (mà face\_recognition sử dụng).
* Dùng os.path.basename để lấy tên tệp từ đường dẫn và os.path.splitext để tách phần mở rộng của tên tệp.
* Sau đó sử dụng face\_recognition.face\_encodings để mã hóa các khuôn mặt trong hình ảnh đã chuyển đổi sang RGB. Chú ý rằng chúng ta sử dụng chỉ số [0] để lấy mã hóa của khuôn mặt đầu tiên. Điều này giả định rằng mỗi hình ảnh chỉ chứa một khuôn mặt, vì vậy chúng ta chỉ lấy mã hóa của khuôn mặt đầu tiên.
* Thêm mã hóa của khuôn mặt và tên tệp vào danh sách self.known\_face\_encodings và self.known\_face\_names tương ứng.
* In ra thông báo để xác nhận rằng tất cả các hình ảnh đã được mã hóa và lưu trữ.
* Phương thức detect\_known\_faces:

Phương thức này nhận một khung hình (frame) đầu vào và thực hiện nhận diện các khuôn mặt trong đó.

A screen shot of a computer program

Description automatically generated

* Khung hình đầu vào được thay đổi kích thước để tăng tốc độ phát hiện khuôn mặt. Hệ số thay đổi kích thước được xác định bởi self.frame\_resizing.
* Khung hình đã thay đổi kích thước được chuyển đổi từ không gian màu BGR (sử dụng bởi OpenCV) sang không gian màu RGB (sử dụng bởi face\_recognition).
* Sử dụng face\_recognition.face\_locations để xác định vị trí của tất cả các khuôn mặt trong khung hình.
* Sử dụng face\_recognition.face\_encodings để mã hóa các khuôn mặt được xác định trong khung hình.
* Duyệt qua từng mã hóa khuôn mặt được xác định. Sử dụng `face\_recognition.compare\_faces` để so sánh mã hóa khuôn mặt này với các mã hóa khuôn mặt đã biết.
* Nếu một khuôn mặt khớp với các khuôn mặt đã biết, lấy tên tương ứng từ danh sách các tên đã biết.
* Chuyển đổi vị trí của các khuôn mặt từ kích thước thay đổi của khung hình trở lại kích thước ban đầu. Điều này giúp điều chỉnh tọa độ của các khuôn mặt để phù hợp với khung hình gốc.
* Cuối cùng, trả về vị trí và tên của các khuôn mặt đã được nhận diện trong khung hình dưới dạng mảng NumPy và danh sách Python tương ứng.

**-Import SimpleFacerec từ module simple\_facerec**

Module simple\_facerec là một tệp Python (simple\_facerec.py) chứa định nghĩa của lớp SimpleFacerec. Import lớp này để sử dụng các phương thức và thuộc tính của lớp SimpleFacerec.



Phần code sau đây thực hiện nhận diện khuôn mặt trên video từ camera và hiển thị tên của người trong khuôn mặt nếu họ được nhận diện.

A computer screen with many colorful text

Description automatically generated

* Tạo một đối tượng SimpleFacerec từ lớp đã được import trước đó.
* Sử dụng phương thức load\_encoding\_images để tải các hình ảnh và mã hóa khuôn mặt từ thư mục "images/".
* Sử dụng cv2.VideoCapture(0) để truy cập camera.
* Trong vòng lặp While, sử dụng cap.read() để đọc từng khung hình từ camera, ret là một biến boolean cho biết liệu việc đọc frame thành công hay không, frame là frame hình ảnh được đọc từ camera.
* Sử dụng phương thức detect\_known\_faces từ đối tượng SimpleFacerec để phát hiện và nhận diện các khuôn mặt trong frame.
* Kết quả được lưu trong face\_locations và face\_names.
* Vòng lặp for duyệt qua mỗi khuôn mặt được nhận diện và vẽ hộp bao quanh khuôn mặt và hiển thị tên của người trong khuôn mặt đó lên trên hình ảnh.
* Hộp và tên được vẽ bằng hàm cv2.rectangle và cv2.putText.
* Hiển thị khung hình với các khuôn mặt được nhận diện và tên của họ.
* Thoát khỏi vòng lặp nếu phím ESC được nhấn.
* Hàm cv2.waitKey được sử dụng để chờ phím được nhấn, với tham số là thời gian chờ tính bằng mili giây.
* Giải phóng tài nguyên camera sau khi hoàn thành công việc.
* Đóng tất cả các cửa sổ hiển thị bằng cách gọi cv2.destroyAllWindows().

**2c. Kết quả đạt được**

Nhận diện và label được 3 khuôn mặt đã đặt tên, đồng thời đối với khuôn mặt không có trong tập được gán nhãn thì được label mang tên “Unknown”.

A screenshot of a computer

Description automatically generated

**2d. Ưu điểm và hạn chế**

***- Ưu điểm***

* Tính phức tạp thấp: Mã có cấu trúc đơn giản và dễ hiểu, dễ dàng mở rộng và tùy chỉnh cho các ứng dụng khác nhau.
* Tích hợp dễ dàng: Việc sử dụng lớp SimpleFacerec giúp tích hợp mã hóa khuôn mặt và nhận diện khuôn mặt một cách dễ dàng vào bất kỳ dự án nào mà không cần viết lại mã.
* Hiệu năng cao: Mã được tối ưu hóa để hoạt động nhanh chóng, đặc biệt là trong việc nhận diện khuôn mặt trực tiếp từ video từ camera.
* Hiển thị thông tin: Tên của người được nhận diện được hiển thị trực tiếp trên video, giúp người dùng dễ dàng nhận ra và tương tác với hệ thống.

***- Hạn chế***

* Độ chính xác có thể bị giảm: Việc sử dụng một hệ số thay đổi kích thước khung hình có thể làm giảm độ chính xác của quá trình nhận diện khuôn mặt.
* Phụ thuộc vào dữ liệu: Độ chính xác của mã phụ thuộc vào dữ liệu huấn luyện và dữ liệu khuôn mặt được mã hóa. Nếu dữ liệu huấn luyện không đủ đa dạng hoặc không đại diện cho tất cả các trường hợp có thể xảy ra, hiệu suất nhận diện có thể bị ảnh hưởng.
* Yêu cầu phần cứng: Hiệu suất của mã có thể phụ thuộc vào cấu hình phần cứng của máy tính, đặc biệt là trong việc xử lý video và các thuật toán nhận diện khuôn mặt.
* Khả năng nhận diện hạn chế: Các thuật toán nhận diện khuôn mặt có thể gặp khó khăn trong việc nhận diện khuôn mặt trong điều kiện ánh sáng yếu, góc nhìn khó khăn hoặc các biến thể khác nhau của khuôn mặt.