FACE_RECOGNITION_SYSTEM

Created	@May 18, 2025 10:36 AM
Class	AI

Abstract:

Hệ thống điểm danh bằng Nhận diện khuôn mặt. Hệ thống tích hợp các thư viện như Flask, OpenCV và module face_recognition để xử lý video từ webcam, nhận diện khuôn mặt và lưu trữ dữ liệu điểm danh vào cơ sở dữ liệu SQLite. Báo cáo trình bày mục tiêu, lý thuyết cơ sở và quá trình thực hiện

I. Giới thiệu

1. Mục tiêu

- Nhận diện khuôn mặt thông qua webcam
- Ghi lại thông tin điểm danh (Tên người - Thời gian điểm danh) và lưu trữ dữ liệu đấy vào cơ sở dữ liệu SQLite
- Giao diện web cơ bản để theo dõi cơ sở dữ liệu đấy

2. Công cụ

- Flask: Dùng để xây dựng ứng dụng web và xử lý request từ HTTP
- OpenCV: Thư viện xử lý ảnh, truy cập camera → Hiển thị video
- face_recognition: Mô hình pretrained dựa trên mạng CNN (ResNet), dùng để trích xuất vector đặc trưng (embedding) 128 chiều, nhận diện khuôn mặt, và so sánh đặc trưng hiện tại với đặc trưng gốc đã lưu.
- SQLite: Cơ sở dữ liệu → Lưu trữ thông tin điểm danh (Keep tracking database)
- HTML/CSS: Web

II. Lý thuyết cơ sở

1. Nhận diện khuôn mặt

Là công nghệ nhận diện khuôn mặt trong đó hệ thống này có thể tự động nhận diện, xác minh được danh tính của người thông qua video, hình ảnh bằng cách phân tích các đặc điểm, đặc trưng của mặt người

2. Tìm hiểu về thuật toán bên trong module face_recognition

Thư viện face_recognition sử dụng mô hình DL dựa trên mạng CNN là ResNet được huấn luyện sẵn

Bước 1: Phát hiện khuôn mặt:

- Sử dụng thuật toán HOG algorithms (mặc định là HOG, ngoài ra có thể dùng cả CNN) để phát hiện vùng khuôn mặt → return bounding box

Bước 2: Trích xuất đặc trưng:

- Áp dụng CNN pretrained → trích xuất 1 vector 128 chiều đại diện cho các features của mặt

Bước 3: Chuẩn hóa và lưu trữ

- Vector 128 chiều đc chuẩn hóa normalize để đảm bảo độ dài là 1 → So sánh hiệu quả hơn bằng k/c Euclidean hoặc cosine similarity

3. Tìm hiểu về HOG algorithms

Sử dụng thuật toán HOG dựa trên đặc trưng hình học để phát hiện đối tượng

Cụ thể quy trình HOG

1. Chuyển từ ảnh màu sang ảnh xám → Loại bỏ thông tin về màu sắc → Tập trung thông tin về cấu trúc hình học
2. Tính gradient: Sử dụng Sobel filter (Gaussian) để tính gradient theo x, y → Xác định hướng và cường độ của ảnh << So sánh các pixel xung quanh ⟶ Tìm hướng >>
3. Tạo histogram: Chia ảnh thành các cells nhỏ → Tính histogram của hướng gradient trong các ô
4. Chuẩn hóa (block normalization): Nhóm các ô thành khối lớn hơn và chuẩn hóa → Giảm ảnh hưởng của độ sáng kh đều

Tạo đặc trưng: Gộp tất cả các histogram lại thành 1 vector đặc trưng đại diện cho ảnh

III. Các bước thực hiện

1. Chuẩn bị môi trường (setup environment)

Các thư viện cần trong file requirement.txt

flask → Web
face_recognition → module có sẵn dùng để nhận diện khuôn mặt → pretrained lại (Cần cài cmake và dlib tích hợp để sử dụng module này)
opencv-python → Dùng để tích hợp webcam
numpy
pillow

Các thực hiện:

→ Tạo môi trường ảo: Mở terminal: python -m venv venv

Kích hoạt môi trường ảo: .\venv\Scripts\activate

Cài thư viện ở trên: pip install -r requirement.txt

2. Cấu trúc dự án

- data
  - images: Chứa các thư mục con cho từng người (ví dụ: person1, person2), lưu trữ ảnh định
    - person1: (lưu trữ các ảnh định dạng .jpg)
    - person2: .....
    - person3: .....
  - embedding.pkl: Chứa vector đặc trưng khuôn mặt (embedding)
- database: attendance.db: Chứa database
- src: chứa các file python
  - database_utils.py: Tạo và chứa logic quản lý database

    - embedding_face.py: trích xuất đặc trưng khuôn mặt
    - recognize_faces.py: Nhận diện khuôn mặt
- static: Giao diện CSS
- templates: Quản lý giao diện web html

3. Chuẩn bị dữ liệu (data)

Thu thập khoảng 5-10 ảnh của mỗi người. Số người tùy theo dự án

Ảnh đc chụp góc chính diện, gocs nghiêng....

4. Trích xuất đặc trưng (embedding)

face_recognition sử dụng mô hình dlib HOG/CNN để tự động phát hiện khuôn mặt và trích xuất đặc trưng (embedding) (128 vector)

→ Tự điều chỉnh kích thước đầu vào nội bộ → không nhất thiết phải resize (note: Tuy nhiên trong các module khác có thể cần resize)

⇒ Trích xuất dữ liệu này đc lưu vào thư mục: data/embedding.pkl

5. Xây dựng nhận diện khuôn mặt (recognize face)

⟶ Kết nối webcam camera

⟶ Duyệt từng khuôn mặt trong frame

Sử dụng hàm face_locations: để phát hiện vùng khuôn mặt → return (top, right, bottom, left) của vùng mặt

Sử dụng hàm face_encodings: để trích xuất encodings của ảnh hiện tại

⟶ Sau đó đến bước so sánh

So sánh: `compare_faces` vs `face_distance`

`compare_faces(known, test, tolerance)`

- Trả về danh sách `True/False`, dựa vào **ngưỡng** `tolerance` .
- Ví dụ: nếu khoảng cách giữa 2 embedding < 0.6 → trả về `True`.

`face_distance(known, test)`

- Trả về **khoảng cách Euclidean** giữa các vector → giá trị nhỏ tức là giống nhau.
- Bạn có thể dùng nó để **tìm người giống nhất (min distance)**, rồi mới kiểm tra threshold.

⇒ Do đó, **dùng kết hợp cả 2** sẽ **chính xác + linh hoạt** hơn.

6. Quản lý database

- **Mục đích**: Lưu trữ và truy xuất thông tin điểm danh để theo dõi dễ dàng.
- **Thực hiện**: Sử dụng database_utils.py để tạo bảng, ghi dữ liệu (tên, thời gian), và truy vấn dữ liệu từ attendance.db.

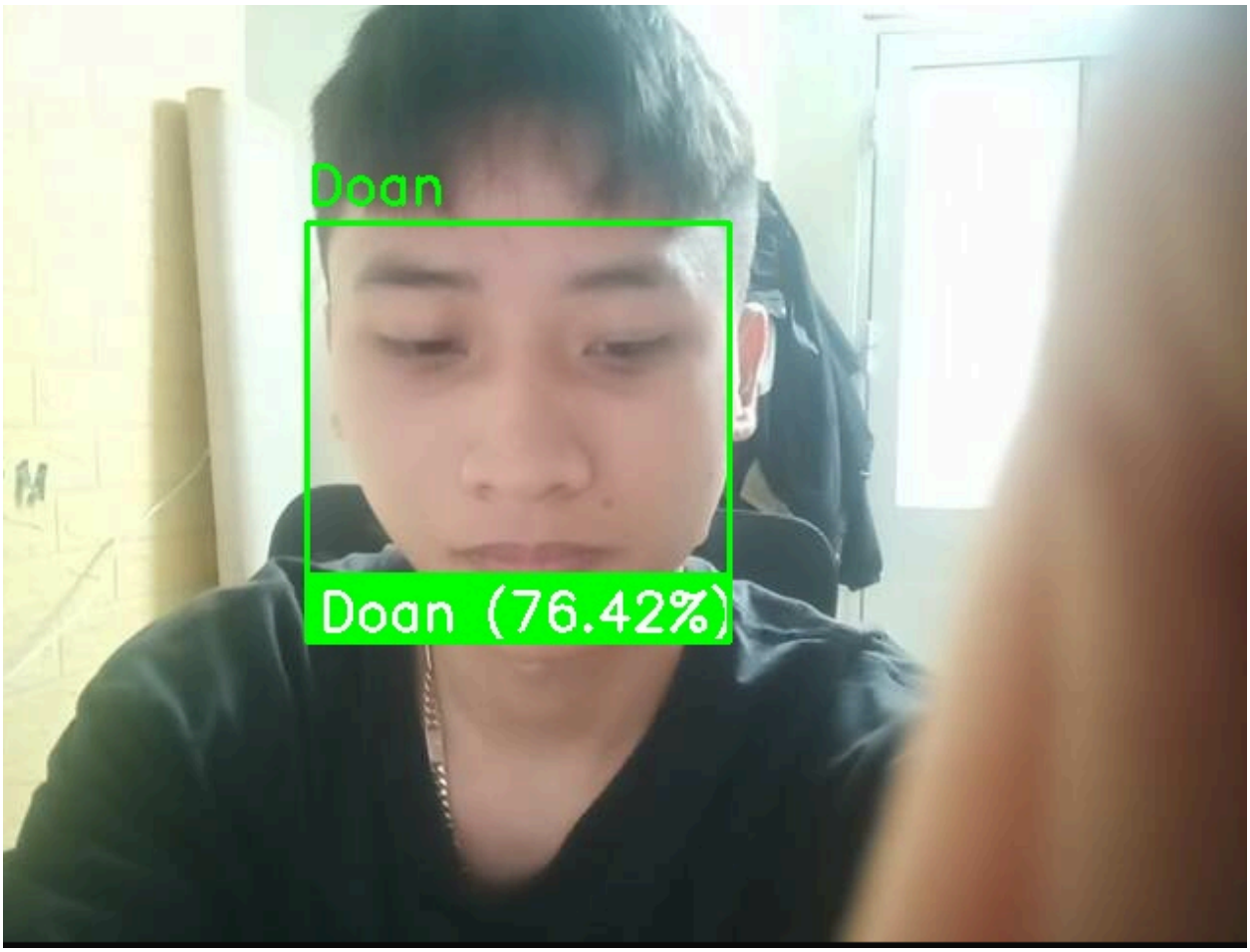
7. Xây dựng giao diện

- Sử dụng HTML/CSS trong thư mục templates và static để thiết kế giao diện web:
  - Trang chủ (index.html): Hiển thị luồng video
  - Trang điểm danh (attendance.html): Hiển thị lịch sử điểm danh
  - Trang thêm người (add_person.html): Form để thêm ảnh mới
- CSS trong static/css/styles.css để trang trí giao diện

8. Triển khai hệ thống

- Đẩy mã nguồn lên GitHub tại https://github.com/mDoanzz43/Face_Recognition_System.
- Chạy ứng dụng bằng lệnh python main.py và truy cập qua trình duyệt (localhost:5000).

Kết quả:

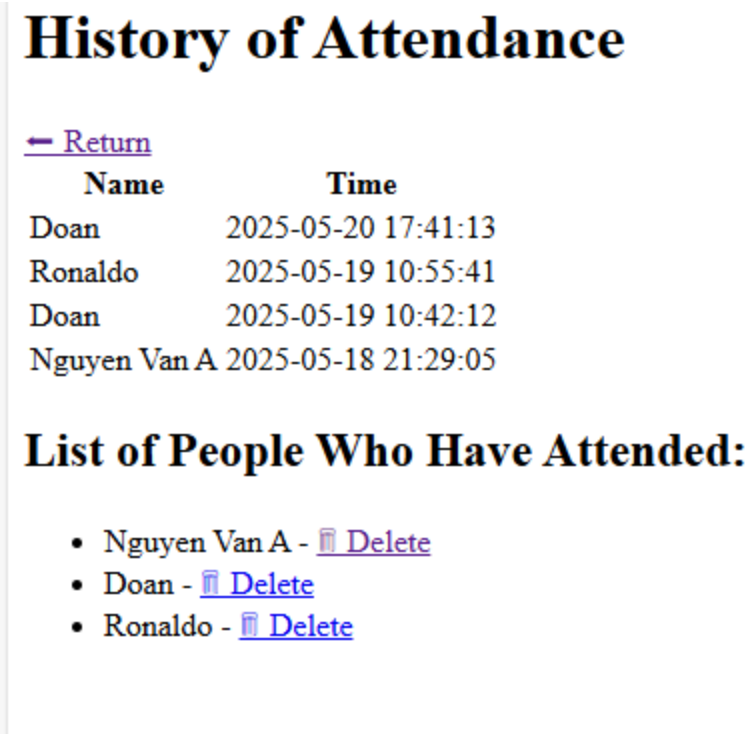


Khi kết nối webcam → nhận diện được mặt

```
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 727-119-354
127.0.0.1 - - [20/May/2025 17:40:47] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/May/2025 17:40:47] "GET /favicon.ico HTTP/1.1" 404 -
f[INFO] has attendance: {name} time {timestamp}
[INFO] Doan đã điểm danh lúc 17:41:13
127.0.0.1 - - [20/May/2025 17:41:13] "GET /video_feed HTTP/1.1" 200 -
f[INFO] {name} has attendance today
[INFO] Doan đã điểm danh lúc 17:41:23
f[INFO] {name} has attendance today
[INFO] Doan đã điểm danh lúc 17:41:34
f[INFO] {name} has attendance today
[INFO] Doan đã điểm danh lúc 17:41:44
```

→ Điểm danh

→ Lưu trữ được lịch sử điểm danh



History of Attendance

⬅ Return

Name	Time
Doan	2025-05-20 17:41:13
Ronaldo	2025-05-19 10:55:41
Doan	2025-05-19 10:42:12
Nguyen Van A	2025-05-18 21:29:05

List of People Who Have Attended:

- Nguyen Van A - 🗑 Delete
- Doan - 🗑 Delete
- Ronaldo - 🗑 Delete

IV. Kết luận

Dự án đã hoàn thành với các chức năng cơ bản, đạt được mục tiêu đề ra. Tuy nhiên, độ chính xác cần cải thiện bằng cách tăng dữ liệu và áp dụng transfer learning. Kinh nghiệm từ dự án giúp hiểu rõ hơn về nhận diện khuôn mặt và quản lý dự án