TRƯỜNG ĐẠI HỌC ĐÀ LẠT

KHOA CÔNG NGHỆ THÔNG TIN

TIỂU LUẬN

XÂY DỰNG ỨNG DỤNG ĐIỂM DANH BẰNG PHƯƠNG PHÁP NHẬN DIỆN KHUÔN MẶT

Môn Lập trình Python

Giảng viên hướng dẫn: **ThS. Phan Thị Thanh Nga**

Sinh viên thực hiện: Vũ Bá Đông - 2111818

Lâm Đồng, ngày 28 tháng 10 năm 2023

MỤC LỤC

❀LỜI CẢM ƠN❀

Em xin cảm ơn cô đã tận tình chỉ bảo và góp ý em trong quá trình thực hiện đề tài này. Sự nhiệt tình của cô và các bạn cũng là động lực để em hoàn thiện đề tài này đúng hạn.

Trong quá trình thực hiện báo cáo cũng như thuyết trình về đề tài em vẫn còn nhiều thiếu sót, em mong nhận được sự góp ý của cô để có thể hoàn thiện hơn về kĩ năng viết báo cáo cũng như thuyết trình trước đám đông của mình.

Một lần nữa em xin chân thành cảm ơn cô đã hỗi trợ em hoàn thiện bài báo cáo này.

I. GIỚI THIỆU

1.1 Giới thiệu đề tài

Theo một cách truyền thống đã được sử dụng khi điểm danh các sinh viên khi đi học sẽ có một số cách như: lớp trưởng điểm danh báo cáo lại cho giáo viên, giáo viên tự điểm danh bằng danh sách sinh viên hoặc một số cách khác. Tuy nhiên các cách điểm danh này đều mang nhược điểm chẳng hạn có thể điểm danh thiếu sót, các sinh viên điểm danh hộ nhau, hay sự mất tập trung khiến lỡ mất phiên điểm danh. Cải tiến hơn một chút là điểm danh bằng các nền tảng trực tuyến, hay điểm danh bằng thẻ từ tuy nhanh hơn điểm danh truyền thống nhiều lần nhưng vẫn không tránh khỏi một số nhược điểm của phương pháp điểm danh cũ, kèm theo các nhược điểm mới như là do kết nối mạng chập chờn dẫn đến lỗi khi điểm danh, thiết bị không đủ hiện đại để thực hiện các thao tác điểm danh, hay là các trường hợp đột xuất do lỗi trang web của nhà trường, hay là sinh viên làm mất thẻ, máy từ bị lỗi, đối với các trường hợp phát sinh việc này đôi khi còn mất nhiều thời gian của sinh viên và giảng viên hơn cách truyền thống. Tuy nhiên khi mà trong thời điểm hiện tại, công nghệ thông tin ngày càng phát triển, kèm theo trí tuệ nhân tạo cũng ngày càng thông minh hơn, việc sử dụng hai phương pháp điểm danh trên cũng trở nên lỗi thời, tốn thời gian.

Để khắc phục những hạn chế trên, việc ứng dụng công nghệ để xây dựng phần mềm điểm danh tự động là một bước đi rất mới mẻ, và có hiệu quả cao. Trong đề tài tiểu luận môn Python này em chọn đề tài "Điểm danh sinh viên bằng phương pháp nhận diện khuôn mặt".

1.2 Mục tiêu đề tài

Học và ứng dụng được ngôn ngữ lập trình Python vào các dự án.

Nghiên cứu các thư viện, kỹ thuật xử lý ảnh và nhận diện khuôn mặt trong python

Xây dựng trương trình điểm danh bằng phương pháp nhận diện khuôn mặt

Xây dựng giao diện đơn giản, thân thiện với người dùng.

Xây dựng được ứng dụng nhằm đáp ứng được các yêu cầu mà người sử dụng muốn.

Phân tích ưu nhược điểm của phương pháp nhận diện và các đề xuất cải hiệu suất của phương pháp nhận diện.

Hoàn thiện ứng dụng với các chức năng và có khả năng chạy ổn định.

Qua đó việc xây dựng ứng dụng giúp tiết kiệm thời gian, nâng cao tính chính xác tin cậy của quá trình điểm danh cũng như giảm thiểu công việc cho giảng viên khi thực hiện công tác điểm danh lớp học.

Phương pháp nghiên cứu:

Nghiên cứu cơ sở lý thuyết về công nghệ nhận diện khuôn mặt và các ứng dụng tương tự.

Phân tích yêu cầu, thiết kế ứng dụng và cơ sở dữ liệu cho ứng dụng điểm danh

Xây dựng ứng dụng với các chức năng cơ bản cho việc điểm danh: Tìm kiếm sinh viên, điểm danh sinh viên bằng tay hoặc điểm danh sinh viên bằng mở camera và ghi lại thông tin.

Trực tiếp sử dụng và kiểm thử các chức năng của ứng dụng.

Đánh giá các chỉ số của phương pháp nhận diện: tốc độ xử lý, độ chính xác khi nhận dạng, tính dễ dàng sử dụng, triển khai.

Phân tích điểm mạnh, yếu và đề xuất của ứng dụng.

II. CƠ SỞ LÝ THUYẾT

2.1 Tìm hiểu công nghệ nhận diện khuôn mặt

Nhận diện khuôn mặt là một trong các lĩnh vực của trí tuệ nhân tạo (AI) liên quan đến việc xác định và nhận dạng một người dựa trên hình ảnh của người đó. Phương pháp này dựa trên việc phân tích các đặc trưng của khuôn mặt như kích thước, hình dạng, tỉ lệ các bộ phận trên khuôn mặt, cũng như vị trí của các bộ phận.

Trong quá trình phát triển phương pháp nhận diện khuôn mặt, việc nhiều thuật toán nhiều phương pháp nhận diện khuôn mặt là điều hiển nhiên, nhưng một số phương pháp sẽ có độ phổ biến cao hơn và có tần suất sử dụng lớn hơn. Nhưng sẽ được chia thành hai phương pháp lớn là "Phương pháp dựa trên đặc trưng" và phương pháp dựa trên học máy"

2.2 Một số phương pháp nhận diện dựa trên đặc trưng khuôn mặt

Vì mỗi khuôn mặt đều có nét đặc trưng riêng biệt như khoảng cách mũi, miêng, cằm,... khi trích xuất các nét đặc trưng này thành vector số hóa thì tệp vector này của mỗi người sẽ khác nhau từ đó có thể nhận được các khuôn mặt khác nhau.

Phương pháp này sử dụng một số thuật toán trích xuất khuôn mặt phổ biến:

- Eigenface: Thuật toán dựa trên ý tưởng phân tích thành các thành phần chính (PCA: hiểu cơ bản là phân tích dữ liệu và sau đó tìm ra các thành phần chính của dữ liệu và giữ lại các thành phần đó việc này sẽ rút gọn số lượng dữ liệu cần phải xử lý) để rút trích các đặc trưng khuôn mặt từ các khuôn mặt đã được tiền xử lý bằng cách cắt, xoay chỉnh sáng của hình ảnh và chuyển thành ma trận vector. Áp dụng PCA lên ma trận từ hình ảnh trước đó để tìm ra các khuôn mặt riêng, các vector riêng có giá trị riêng lớn nhất sẽ đại diện cho đặc trưng khuôn mặt. Sau đó với mỗi ảnh khuôn mặt, chiếu ảnh khuôn mặt lên không gian *n* chiều của các khuôn mặt đã được phân tích thành các vector chính trước đó và tính trọng số của ảnh. Trọng số *n* tạo thành vector đặc trưng *n* chiều của khuôn mặt. So sánh 2 vector tương ứng với 2 ảnh khuôn mặt để xác định xem có phải cùng một người nay không.

- Fisherface: Thuật toán này cải tiến từ thuật toán Eigenface, có các đặc điểm mới là sử dụng kĩ thuật phân tích phân biệt tuyến tính (Linear Discriminant Analysis - LDA). LDA lấy PCA làm tiền xử lý dữ liệu và lấy các vector mà PCA trả về làm đầu vào của LDA. Đầu ra của LDA cũng là các vector quan trọng như PCA nhưng lại các vector có thể phân biệt tốt nhất giữa các người khác với nhau, đồng thời các vector dữ liệu của cùng một người dùng sẽ được gom gọn lại, không bị phân tán như PCA. Tuy nhiên LDA phải xác định rõ ràng tên giữa các lớp. Tức là nếu có 100 người thì phải có 100 lớp với tên khác nhau. Vì vậy LDA dùng để tìm ra các đặc trưng có khả năng phân biệt cao nhất giữa các khuôn mặt.
- Local Binary Patterns Histogram: thuật toán này sử dụng phương pháp đặc trưng về độ sáng của khuôn mặt. Mẫu nhị phân cục bộ (Local Binary Pattern-LBP) mô tả độ sáng của mỗi pixel so với các pixel xung quanh nó và cho biết pixel đó sáng, tối hay phương phản với các pixel xung quanh. Kết quả của một LBP là một mã nhị phân mô tả mẫu độ sáng của pixel đó. Tổ hợp toàn bộ LBP của toàn bộ ảnh khuôn mặt sẽ cho ra một vector đặc trưng về độ sáng để nhận dạng và là đại diện cho ảnh của khuôn mặt. Việc so sánh và nhận diện khuôn mặt sẽ lấy 2 vector đại diện và so sánh với nhau.

2.3 Dựa trên học sâu

Là phương pháp sử dụng các mô hình học máy để nhân diện khuôn mặt. Các mô hình học máy này sử dụng mạng nơ-ron nhân tạo để học các đặc trưng phức tạo từ dữ liệu. Mô hình học sâu cho nhận diện khuôn mặt có kiến trúc nhiều tầng nơ-ron kết hợp với nhau. Mỗi tầng học một đặc trưng ở mức độ khác nhau và mức độ phức tạp về đặc trưng cũng cao hơn, mang nhiều thông tin hơn về khuôn mặt. Sau quá trình huấn luyện với lượng dữ liệu lớn, mạng nơ-ron có thể tự động trích xuất các đặc trưng cấp cao của khuôn mặt để xác định chính xác khuôn mặt đó.

2.3.1 Giải thích thông tin về chuyên môn

Mạng nơ-ron nhân tạo: lấy cảm hứng từ não người, bắt chước cách hoạt động của bộ não con người. Các hạt nơ-ron nhân tạo có bao gồm các thành phần chính là:

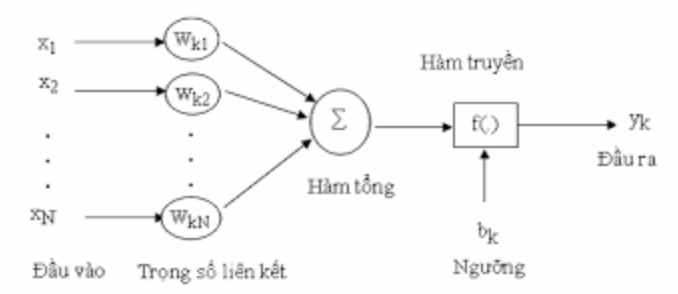
Đầu vào: dữ liệu trả về của các nơ-ron khác hoặc từ bên ngoài.

Trọng số: hệ số sẽ được nhân với mỗi đầu vào, hiển thị mức độ ảnh hưởng của đầu vào với nơ-ron đó. Trọng số thường được khởi tạo ngẫu nhiên lúc ban đầu, và dần được tự động căn chỉnh trong quá trình huấn luyện để cho ra giá trị tốt nhất.

Hàm kích hoạt: có thể coi là nhân của nơ-ron thực hiện xử lý (đoạn chương trình) các dữ liệu đầu vào bao gồm cả trọng số và trả dữ liệu ra.

Ngưỡng: giá trị điều chỉnh làm tăng hoặc giảm giá trị đầu ra của nơ-ron.

Đầu ra: kết quả sau khi thực hiện hàm kích hoạt và ngưỡng để chuyển tới các nơ-ron khác.



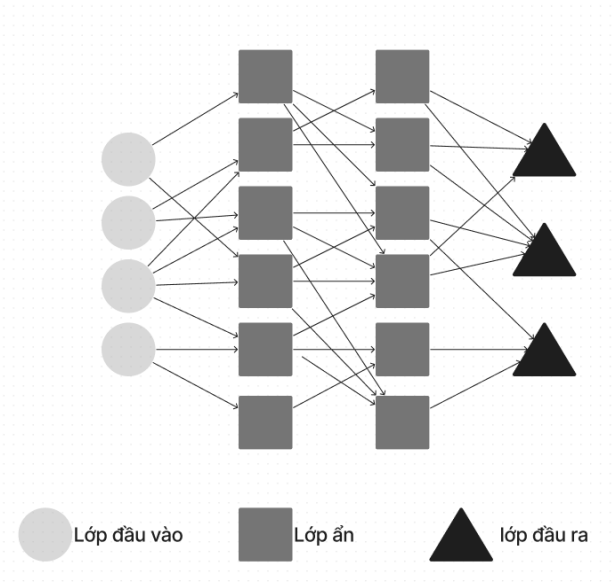
Hình 1. Mô tả cấu tạo của một nơ-ron nhân tạo.

Một mạng lưới nơ-ron nhân tạo bao gồm nhiều nơ-ron kết hợp với nhau thông qua trọng số, và một nơ-ron thường có nhiều đầu ở phía bên trái vào và nhiều đầu ra ở phía bên phải. Mỗi đầu vào phải kết nối với một trọng số quy định mức độ ảnh hưởng. Số lượng đầu vào và số lượng đầu ra phụ thuộc vào thiết kế và kiến trúc của mạng nơ-ron đó và vị trí của nơ-ron đó trong mạng lưới.

2.3.2 Kiến trúc mạng nơ-ron và mạng nơ-ron tích chập

Kiến trúc của mạng nơ-ron nhân tạo bao gồm ba lớp chính:

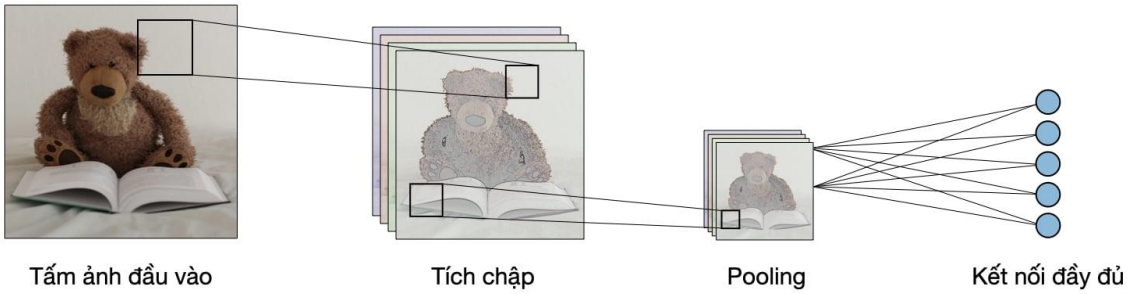
- Lớp đầu vào: Nhận các dữ liệu đầu vào và truyền cho lớp phía sau. Các nơ-ron ở lớp này tương ứng với các chiều tức là khi có một bức ảnh 10x10 pixel thì sẽ có 100 nơ-ron nhận giá trị của mỗi pixel ảnh và gửi qua lớp tiếp theo.
- Lớp ẩn: Là lớp quan trọng trong mạng lưới, nó thực hiện xử lý và trích chọn thông tin từ đầu vào, lớp ẩn này có thể bao gồm nhiều lớp khác, và mỗi nơ-ron ở lớp này kết hợp đồng thời với nơ-ron ở lớp trước nó và nơ-ron ở lớp sau nó.
- Lớp đầu ra: Chứa các nơ-ron đưa ra kết quả cuối cùng, mỗi nơ-ron tương ứng với một tên được phân biệt. Các nơ-ron ở lớp này nhân giá trị đầu vào từ lớp ẩn thông qua trọng số. Số lượng nơ-ron ở lớp này bằng số lượng các lớp mà muốn phân loại. Mỗi nơ-ron tương ứng với một lớp duy nhất, và giá trị của nơ-ron nào càng cao thì độ chính xác của nó càng lớn. Lớp đầu ra sẽ không liên kết ngược lại với các lớp trước đó thay vào đó nó truyền thông tin theo một chiều.



Hình 2. Mô tả cấu trúc mạng nơ-ron

Mạng nơ-ron tích chập (Convolutional Neural Network-CNN): là kiến trúc mạng nhân tạo dựa trên mạng nơ-ron nhân tạo ban đầu được thiết kế để xử lý dữ liệu liên quan tới hình ảnh. Bao gồm một số đặc điểm đặc trưng chính:

- Các lớp tích chập (Convolutional layers): Sử dụng bộ lọc tích chập để trích xuất các đặc trưng ảnh như góc, cạnh, màu sắc. Các lớp tích chập thường đi kèm với lớp polling giúp giảm chiều dữ liệu và nằm ở đầu của lớp ẩn.
- Lớp kết nối đầy đủ (Fully-Connected): Lớp này đi sau lớp tích chập và lớp pooling và thực hiện phân loại dựa trên các đặc trưng mà lớp tích chập đã trích xuất.



Hình 3. Mô tả cấu trúc mạng nơ-ron tích chập

Mạng nơ-ron tích chập tự động hóa các bộ lọc tích chập thông qua quá trình huấn luyện. Ưu điểm của mạng nơ-ron này là khả năng trích xuất các đặc trưng từ ảnh hoặc video mà không cần tới thiết kế thủ công của con người. Đạt hiệu quả cao trong các ứng dụng về phân loại ảnh, phát hiện đối tượng, nhận diện khuôn mặt.

2.4 So sánh thuật toán

Dựa trên đặc trưng	Học sâu
Đơn giản, dễ hiểu, dễ triển khai	Phức tạp, khó hiểu, khó triển khai
Tốc độ xử lý nhanh	Yêu cầu dữ liệu lớn và tài nguyên tính toán mạnh
Không yêu cầu dữ liệu lớn để huấn luyện	Thời gian huấn luyện lâu
Độ chính xác tương đối	Độ chính xác rất cao
Dễ bị ảnh hưởng bởi góc chụp, ánh sáng	Ít bị ảnh hưởng bởi yếu tố ngoại cảnh
Khó xử lý với khuôn mặt bị biến dạng	Có thể xử lý khuôn mặt biến dạng tốt hơn

2.5 Ngôn ngữ Python

Python là một ngôn ngữ lập trình bậc cao, có cú pháp ngắn gọn, dễ học và sử dụng. Một số đặc điểm nổi bật của Python:

- Là ngôn ngữ thông dịch, có thể chạy các dòng lệnh từng phần một để kiểm tra chương trình.
- Hỗ trợ nhiều phong cách lập trình: hướng đối tượng, lập trình hàm, lập trình thủ tục.
- Cú pháp đơn giản, dễ đọc và ít từ khóa. Sử dụng dấu cách, thụt đầu dòng thay cho các ký hiệu như ; hay {}.
- Là ngôn ngữ đa năng, có thể sử dụng cho nhiều lĩnh vực: web, khoa học dữ liệu, AI, xử lý ảnh, automation,...
- Hỗ trợ các kiểu dữ liệu phong phú như số, chuỗi, list, dict, set,... và các cấu trúc dữ liệu người dùng.
- Có hệ thống quản lý package (thư viện) và dependencies rất tốt, dễ cài đặt và sử dụng các thư viện.

2.6 Thư viện face_recognition python

2.6.1 Giới thiệu khái quát

Thư viện được xây dựng bằng mô hình học với độ chính xác 99,38% dựa trên tiêu chuẩn **Labeled Face in Wild** **(** tiêu chuẩn công khai để xác định khuôn mặt) theo công bố của nhóm phát triển. Thư viện được xây dựng dựa trên việc kết hợp cả hai phương pháp học sâu và phương pháp nhận diện dựa trên đặc trưng.

2.6.2 Các bước thực hiện nhận diện

Phát hiện khuôn mặt:Thư viện sử dụng thuật toán học máy Haar Cascades để nhận biết khuôn mặt trong một khung hình.

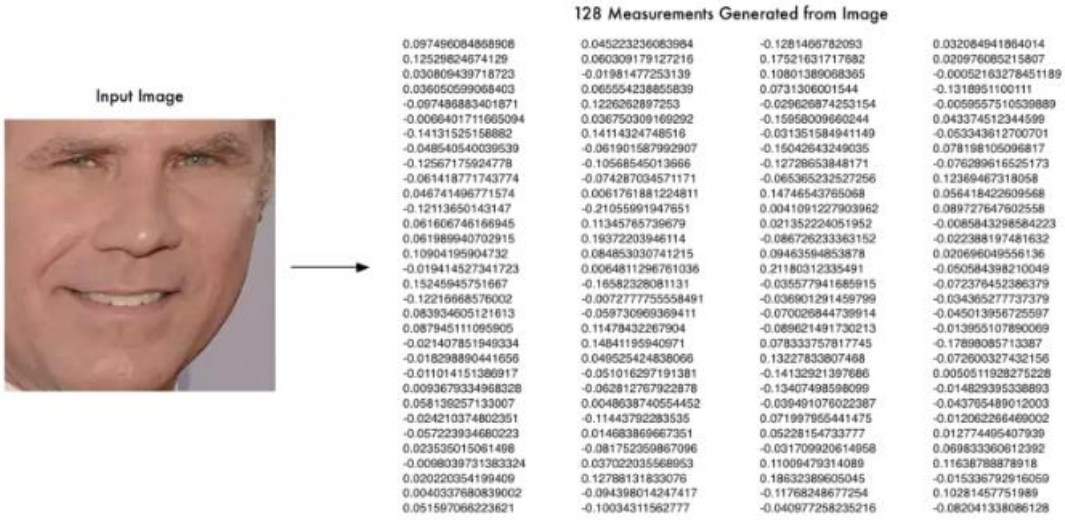
Tiềm xử lý ảnh khuôn mặtCăn chỉnh, xoay để có thể lấy hình ảnh khuôn mặt thẳng hàng và có góc nhìn đối diện. Chuyển ảnh về hệ xám và thay đổi kích thước ảnh thành kích thước chuẩn đã quy định.

Trích xuất đặc trưng: Sử dụng mô hình ResNet50 (Mô hình mạng nơ-ron tích chập với tổng cộng 50 lớp tích chập) đã được huấn luyện sẵn với tập dữ liệu lớn, có độ chính xác lên tới 99,38% theo công bố của nhóm nghiên cứu để trích xuất đặc trưng cấp cao (Hình 4).

Biểu diễn khuôn mặt thành vector 128 chiều dựa trên các đặc trưng đã được trích xuất (hình 5).

Nhận dang: Sử dụng thuật toán VP tree (Vantage-point tree - một dạng cây nhị phân) để tìm kiếm nhanh vector tương đồng gần nhất bằng cách so sánh khoảng cách vector của khuôn mặt cần nhận dạng với vector trong cơ sở dữ liệu.

Hình 4. 68 điểm điểm đặc trưng trên khuôn mặt



Hình 5: đầu ra của hình ảnh khi tìm được vector 128 chiều

2.6.3 Ưu, nhược điểm của thư viện

Ưu điểm	Nhược điểm
Độ chính xác nhận diện rất cao, vượt trội	Độ chính xác có thể bị giảm khi điều kiện

so với các thuật toán cơ bản	ánh sáng không thích hợp
Tốc độ nhanh nhờ mô hình ResNet50 đã được tập huấn sẵn.	Đòi hỏi phần cứng tương đối mạnh để có thể chạy mô hình
Có thể nhận diện được giới tính và tuổi tác, cảm xúc của người dùng.	Đòi hỏi cập nhật dữ liệu khi có sự thay đổi về khuôn mặt
Dễ dàng sử dụng thông qua các hàm có sẵn.	Khó kiểm soát, cải tiến mô hình do sử dụng mô hình đã được huấn luyện sẵn
Có thể kết hợp với các thư viện khác để cải thiện tốc độ ứng dụng.	

2.7 Thư viện OpenCV-Python

Thư viện mã nguồn mở được ứng dụng cực kỳ rộng rãi trong các tác vụ liên quan đến xử lý hình ảnh, video cũng như thị giác máy tính. Ngoài ra thư viện OpenCV còn có thể sử dụng để thực hiện các tác vụ về thị giác máy tính như nhận dạng khuôn mặt, theo dõi đối tượng.

Thư viện OpencV có hỗ trợ nhận diện khuôn mặt bằng phương pháp dựa trên đặc trưng của khuôn mặt và thuật toán Haar Cascade.

2.8 Thư viện QT6

Thư viện dùng để phát triển các ứng dụng giao diện người dùng cho nhiều nền tảng như Windows, Linux, MacOS, Android, IOS. Thư viện cung cấp bộ công cụ và API phong phú để phát triển các ứng dụng giao diện người dùng đẹp, hiệu quả, hiện đại. Có thể sử dụng kèm với thư viện "pyqt6-tools" và "pyqt6-tools designer" để thực hiện thiết kế giao diện một cách trực quan dễ dàng tương tự Windows Forms.

2.8 Thư viện mysql-connector-python

Thư viện dùng để kết nối cơ sở dữ liệu MySQL nhằm thực hiện các tác vụ của người dùng. Cung cấp đầy đủ các tác vụ tạo kết nối tới cơ sở dữ liệu, truy vấn, cập nhật thêm, xóa dữ liệu. Tương tự thư viện QT6 thì thư viện này cũng có thể triển khai trên nhiều nền tảng.

2.9 Thư viện openpyxl

Thư viện dùng để đọc và ghi tệp Excel bằng ngôn ngữ lập trình Python. Bao gồm đầy đủ các tác vụ đọc, ghi dữ liệu vào tệp Excel hoặc tạo mới chỉnh sửa tệp Excel hiện có. Hỗ trợ nhiều phiên bản Excel từ phiên bản 2007 đến nay cũng như hỗ trợ nhiều định dạng Excel phổ biến.

III. THIẾT KẾ VÀ TRIỂN KHAI

3.1 Mô tả chức năng hệ thống

Chức năng ứng dụng bao gồm ba chức năng chính:

- Sử dụng camera để điểm danh khuôn mặt.
- Xem, tìm kiếm và thông tin các buổi điểm danh.
- Xuất tệp điểm danh.

Các chức năng định hướng phát triển:

- Xem, sửa thông tin của học sinh.
- Thêm, xóa học sinh vào danh sách.
- Cho chọn lớp học trước khi bắt đầu sử dụng ứng dụng.
- Hệ thống tính điểm cho sinh viên.

3.2 Triển khai nhận diện khuôn mặt

3.2.1 Thông tin cơ bản

Sử dụng kết hợp hai thư viện chính là Face_recognition và thư viện OpenCV để thực hiện việc sử dụng camera để nhận diện khuôn mặt.

Tập dữ liệu để thực hiện kiểm tra là hình ảnh được trích xuất từ tệp dữ liệu hình ảnh khuôn mặt Face Recognition Dataset - Oneshot Learning, với hình dạng, góc chụp cũng như ánh sáng khác nhau.

Thực hiện kiểm thử trên 200 hình ảnh để xác định độ chính xác cũng như là tốc độ nhận diện của thư viện Face_recognition khi sử dụng kết hợp với thư viện OpenCV và một số thư viện phụ.

3.2.2 Các bước xây dựng

Mã hóa các khuôn mặt có trong cơ sở dữ liệu và lưu vào biến, có thể tùy chỉnh mã hóa thành một file json trước để quá trình có thể nhanh hơn trong khi triển khai thực tế.

Sử dụng OpenCV để khởi tạo và chạy camera, tùy chỉnh số lượng khung hình/s để tối ưu hóa về mặt tốc độ, xử lý cũng như độ trễ của video.

Xử lý tiền dữ liệu sử dụng OpenCV để thay đổi kích thước của hình ảnh cũng như chuyển ảnh thành hệ màu xám trước khi dùng làm đầu vào cho thư viện Face_recognition.

Tìm ra vị trí các khuôn mặt trong hình và mã hóa các khuôn mặt đó.

So sánh khuôn mặt mà camera quay được với các khuôn mặt đã được mã hóa lưu ở biến, trả về vị trí hình ảnh có điểm số cao nhất bằng cách dùng thư viện numpy.

Ràng buộc lại độ chính xác nếu cần và thực hiện lấy thông tin về hình ảnh đó.

Khi nhận diện, đúng hay sai đều sẽ sử dụng thư viện OpenCV để vẽ lên các khung hình của video vị trí khuôn mặt và thông tin. Thông tin của người được nhận diện sẽ có "mã số sinh viên" và người không được nhận diện sẽ là "Unknown".

Thực hiện hiển thị lên video để người dùng có thể nhận biết.

Thực hiện lưu lại thông tin của người đã được điểm danh vào một biến loại "Set", sau khi kết thúc phiên điểm danh sẽ ghi vào cơ sở dữ liệu. Thay vì trực tiếp vừa nhận diện khuôn mặt vừa mở và ghi cơ sở dữ liệu với tần suất có thể lên tới 30-60 lần trên giây, và liên tục truy vấn và ghi lại sẽ dẫn tới việc lãng phí tài nguyên phần cứng, giảm số khung hình trên giây và dễ làm lỗi kết nối cơ sở dữ liệu.

3.2.3 Đoạn chương trình mẫu cơ bản

```
import face_recognition
import cv2
import numpy as np
# Khởi tạo biến loại set để lưu lại người đã điểm danh
video_capture = cv2.VideoCapture(0)

# Đọc ảnh bằng thư viện face_recogintion và mã hóa
obama_image = face_recognition.load_image_file("obama.jpg")
obama_face_encoding = face_recognition.face_encodings(obama_image)[0]
biden_image = face_recognition.load_image_file("biden.jpg")
biden_face_encoding = face_recognition.face_encodings(biden_image)[0]

# Sau khi mã hóa thì lưu thông tin khuôn mặt mã hóa và tên các khuôn #mặt
known_face_encodings = [
    obama_face_encoding,
    biden_face_encoding
]
known_face_names = [
    "Barack Obama",
    "Joe Biden"
]

# Khởi tạo một số biến cần thiết

# Khởi tạo 1 biến set lưu thông tin những người đã điểm danh
registered_names=set()
# Mảng lưu vị trí các khuôn mặt trong khung hình
face_locations = []
# Mảng lưu khuôn mặt được mã hóa trong khung hình
face_encodings = []
# Mảng lưu khuôn mặt được nhận diện trong khung hình
face_names = []
```

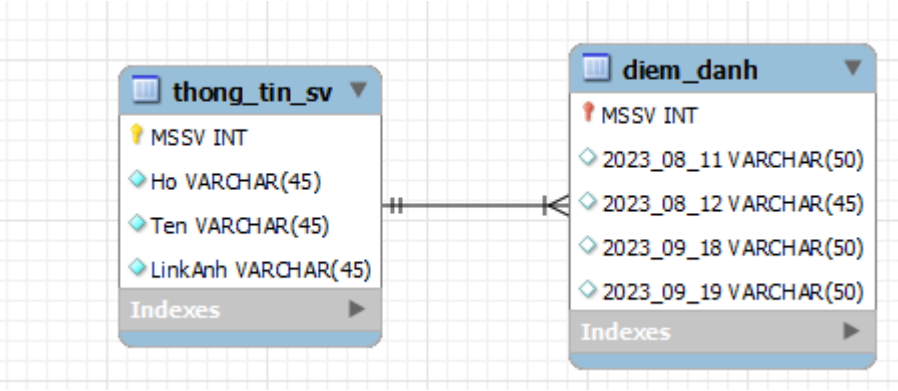
```
process_this_frame = True

while True:
    # Thực hiện đọc các khung hình mà OpenCV trả về
    ret, frame = video_capture.read()
    if process_this_frame:
        # Tiền xử lý dữ liệu bằng cách thay đổi kích thước (¼ kích thước ban đầu)
        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
        # Thay đổi màu của hình ảnh thành hệ RGB
        rgb_small_frame = small_frame[:, :, ::-1]
        # Tìm và lưu lại vị các trí khuôn mặt
        face_locations = face_recognition.face_locations(rgb_small_frame)
        # Mã hóa các khuôn mặt từ vị trí đã tìm
        face_encodings = face_recognition.face_encodings(rgb_small_frame,
face_locations)
        face_names = []
        # Thực hiện so sánh các khuôn mặt được mã hóa trong video
        for face_encoding in face_encodings:
            # So sánh để lấy ra các khuôn mặt có mức độ trùng với khuôn mặt cần
so sánh
            matches = face_recognition.compare_faces(known_face_encodings,
face_encoding)
            name = "Unknown"
            # Tìm ra vị trí khuôn mặt có độ chính xác cao nhất
            face_distances = face_recognition.face_distance(known_face_encodings,
face_encoding)
            best_match_index = np.argmin(face_distances)
            # Trả về giá trị tên khuôn mặt đã nhận diện được
            if matches[best_match_index]:
                name = known_face_names[best_match_index]
            face_names.append(name)
            # Lưu thông tên người được điểm danh vào biến
            registered_names.add(name)
    process_this_frame = not process_this_frame
    # Thực hiện vẽ thông tin cần thiết lên khung hình
    for (top, right, bottom, left), name in zip(face_locations, face_names):
        # Trả lại kích thước ban đầu khi resize
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255),
cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
```

```
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255,
255), 1)
    # Hiển thị lại kết quả cho người dùng xem
    cv2.imshow('Video', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
video_capture.release()
cv2.destroyAllWindows()
```

3.3 Triển khai cơ sở dữ liệu

Với dự án này, em tập trung chính vào việc điểm danh vì vậy cơ sở dữ liệu sẽ cực kì đơn giản (Hình 6).



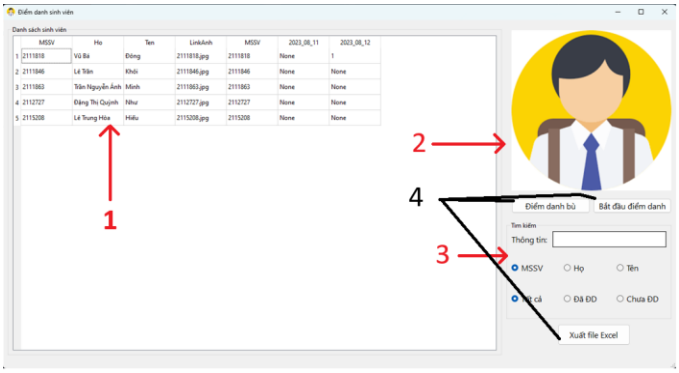
Hình 6: sơ đồ ER của cơ sở dữ liệu

Cơ sở dữ liệu bao gồm 2 bảng chính:

- Bảng thong_tin_sv lưu các thông tin cơ bản của sinh viên. Trường dữ liệu quan trọng đó là LinkAnh trường này sẽ trỏ tới ảnh mà sinh viên đó dùng để điểm danh, vì vậy nếu trường này sai, hoặc thiếu thì sẽ dẫn tới sai lệch thông tin điểm danh của sinh viên đó và sinh viên khác, ngoài ra nếu lỗi sẽ dẫn tới trường hợp ứng dụng bị crash. Trường hợp này có thể khắc phục bằng việc mã hóa khuôn mặt trước và khi nhận diện chỉ cần lấy tên ảnh (cũng là MSSV) để so sánh với MSSV có trong cơ sở dữ liệu.
- Bảng diem_danh: là bảng với duy nhất trường MSSV là được khởi tạo sẵn, các trường khác sẽ được tự động khởi tạo khi bắt đầu phiên điểm danh.

3.4 Giao diện

Bằng cách kết hợp các thư viện QT6 làm thư viện xây dựng giao diện, và thư viện "pyqt6-tools" và "pyqt6-tools designer" giúp trực quan hóa hóa trình xây dựng giao diện và tiết kiệm thời gian trong việc này. Giao diện Bao gồm Màn hình chính lưu trữ thông tin điểm danh của sinh viên theo ngày (số 1 hình 7), một ô hiển thị video khi nhận diện (số 2 hình 7), vùng tìm kiếm dữ liệu (số 3 hình 7) và các nút thực thi chức năng (số 4 hình 7).



Hình 7: Giao diện người dùng của ứng dụng

3.5 Giải thích đoạn mã của ứng dụng

3.5.1 Đoạn mã sử dụng nhiều lần

- Đoạn mã kết nối cơ sở dữ liệu

```
 # Hàm khởi tạo chứa thông tin về chuỗi kết nối bao gồm:
    # địa chỉ máy chủ CSLD
    # tên người dùng
    # mật khẩu
    # tên cơ sở dữ liệu
    def __init__(self):
        self.connection_string = {
    'host': 'localhost',
    'user': 'root',
    'password': '0711',
    'database': 'faceattendace'
}
```

- Đoạn mã truy vấn cơ sở dữ liệu trả về dữ liệu

```
    def execute_query(self,query):

        try:
            # Kết nối cơ sở dữ liệu thông qua chuỗi kết nối
            db = mysql.connector.connect(**self.connection_string)
            # Tạo con trỏ để thực hiện truy vấn
            cursor = db.cursor()
            # Thực thi câu lệnh SQL (câu lệnh query)
            cursor.execute(query)
            # Lấy tất cả các liệu nhận được và ghi vào biến
            record = cursor.fetchall()
        # kiểm tra nếu xảy ra lỗi kết nối CSDL
        except mysql.connector.errors as error:
            print("error", error)
```

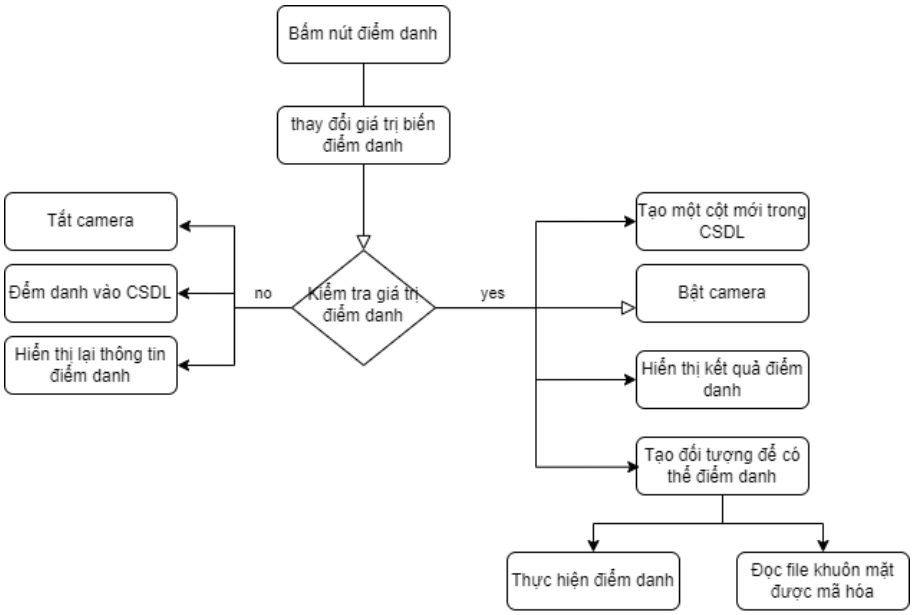
```
            return
        # thực hiện viêc đóng CDL và con trỏ cũng như trả về giá trị sau khi thực
hiện thành công
        finally:
            if db.is_connected():
                db.close()
                cursor.close()
                return record
```

- Đoạn mã truy vấn cơ sở dữ liệu không trả về kết quả

```
def execute_non_query(self,query):
        try:
            db = mysql.connector.connect(**self.connection_string)
            cursor = db.cursor()
            cursor.execute(query)
            # Thực hiện câu truy vấn
            db.commit()
            print("successful")
            # Đóng kết nối CSDL
            db.close()
        except mysql.connector.Error as e:
            print(f"Error: {e}")
            return
```

3.5.2 Đoạn mã xử lý chính

- Sơ đồ, đoạn mã thực hiện điểm danh



Hình 8. Sơ đồ thực hiện chức năng điểm danh

- Xử lý nút điểm danh

```
def btn_diem_danh_click(self):
        # Thay đổi giá trị nút điểm danh
        self.check_click_btn_diem_danh = not self.check_click_btn_diem_danh
        if self.check_click_btn_diem_danh:
            # Tạo một hàng mới để nghi điểm danh
            self.data.create_collumn(self.day)
            # Mở camera
            self.video_caputer = cv2.VideoCapture(0)
            # Khởi tạo đối tượng để thực hiện điểm danh
            self.face = face_attendace.face_recognition()
            # Hiển thị kết quả điểm danh, và điển danh
            # thực hiện kết nối biến timer để có thể trả về hình 30frams/s
            self.timer.timeout.connect(self.update_frame)
            self.timer.start(60)
            self.btn_diem_danh.setText(QtCore.QCoreApplication.translate("MainWin
dow", "Kết thúc điểm danh"))
        else:
            # Ghi thông tin điểm danh được vào cơ sở dữ liệu
            self.data.attendance(self.face.registered_names,self.day)
            # Đặt lại số dòng của bảng thành 0
            self.tw_danh_sach_sv.setRowCount(0)
            # Hiển thị thông tin người đã điểm danh thành công
            self.tw_danh_sach_sv=self.handle_app.load_data_to_table(self.tw_danh_
sach_sv)
            # Đóng camera
            self.close_evet(self.update_frame)
            self.lb_camera.setPixmap(
                QtGui.QPixmap(str(os.path.join(os.getcwd(),
"images/icon_app.png")))
            )
            self.btn_diem_danh.setText(QtCore.QCoreApplication.translate("MainWin
dow", "Bắt đầu điểm danh"))
```

- Tạo một cột mới trong CSDL

```
def create_collumn(self,column_name):
        #tạo câu truy vấn để thực hiện tạo cột với tên dd.MM.yyyy_hh:mm:ss
        query = f"ALTER TABLE diem_danh ADD COLUMN  `{column_name}` NVARCHAR(50)"
        print(query)
        self.execute_non_query(query)
```

- Tạo đối tượng để điểm danh

```
    # Hàm khởi tạo
    def __init__(self):
        # Tạo các biến tương tự chương trình demo
```

```
        self.known_face_encodings = []
        self.known_face_ID = []
        self.data=DataBase()
        self.load_data()
        self.registered_names=set()
        pass
    # Hàm đọc dữ liệu từ file mã hóa trước đó
    def load_data(self):
        # Mở file data.json và đọc
        with open('data.json', 'r') as f:
            data=json.load(f)
        for key, value in data.items():
            # Ghi thông tin dữ liệu vào các biến được khởi tạo trước đó
            self.known_face_encodings.append(np.array(value))
            self.known_face_ID.append(key)
```

- Hàm điểm danh

```
def get_processed_frame(self, frame):
        # thay đổi thành một loại kích thước để sử lý
        small_frame = cv2.resize(frame, (0, 0), fx=1, fy=1)
        # Chuyển thành hệ màu RGB
        rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)
        face_locations = fr.face_locations(rgb_small_frame)
        face_encodings = fr.face_encodings(
            rgb_small_frame, face_locations
        )

        for (top, right, bottom, left), face_encoding in zip(
            face_locations, face_encodings
        ):
            matches = fr.compare_faces(
                self.known_face_encodings, face_encoding
            )
            face_distances = fr.face_distance(
                self.known_face_encodings, face_encoding
            )
            best_match_index = np.argmin(face_distances)
            name = "Unknown"
            if matches and face_distances[best_match_index]<0.4:
                # first_match_index = matches.index(True)
                name = self.known_face_ID[best_match_index]
            color = (0, 0, 0)
            if name != "Unknown":
                color = (255, 0, 0)
                self.registered_names.add(name)
```

```
                name=name+ 'successful'
            else:
                color = (0, 0, 255)

            cv2.rectangle(frame, (left, top), (right, bottom), color, 2)
            # Draw a label with a name below the face
            cv2.rectangle(
                frame, (left, bottom - 35), (right, bottom), color, cv2.FILLED
            )
            font = cv2.FONT_HERSHEY_DUPLEX
            cv2.putText(
                frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255),
1
            )
        # Trả về frame
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        frame = cv2.resize(frame, (311, 311))
        return frame
```

- Hiển thị kết quả điểm danh

```
 def update_frame(self):
        # Đọc các frame mà opnecv lấy được
        ret, frame = self.video_caputer.read()
        if ret:
            #gọi hàm điểm danh
            processed_frame = self.face.get_processed_frame(frame)
            #gọi hàm hiển thị hình ảnh
            self.display_image(processed_frame)

    def display_image(self, frame):
        # Format lại hình ảnh thành dạng BGR
        q_format = QtGui.QImage.Format.Format_BGR888
        # Chuyển ảnh dạng numpy thành Qimage
        out_image = QtGui.QImage(
            frame, frame.shape[1], frame.shape[0], frame.strides[0], q_format
        )
        # Format lại dạng RGB
        out_image = out_image.rgbSwapped()
        # chuyển ảnh thành dạng Pixmap để hiển thị lên lable
        self.lb_camera.setPixmap(QtGui.QPixmap.fromImage(out_image))
```

- Điểm danh vào cơ sở dữ liệu

```
def attendance(self,regiter_IDs,date):
        # Truyền thông tin regiter_IDs là thông tin những người đã được lưu trong
registered_names lúc điểm danh
        # truyền vào tên cột điểm danh
```

```
        query=""
        for id in regiter_IDs:
            # Chạy vòng for qua mảng regiter_IDs và tạo câu truy vấn
            query=f'''UPDATE `faceattendace`.`diem_danh` SET `{date}` = 'Đủ'
WHERE (`MSSV` = '{id}');
            '''
            self.execute_non_query(query)
```

- Hiển thị lại thông tin điểm danh

```
def load_data_to_table(self,table,query=None):
        # Lấy số cột trong CSDL
        number_of_columns=self.data.get_number_of_cloumns()
        # Lấy số dòng trong CSDL
        number_of_rows=self.data.get_number_of_rows()
        # Lấy tên cột trong CSDL
        columns_name=self.data.get_columns_name()
        # Đặt số cột cho bảng
        table.setColumnCount(number_of_columns)
        # Đặt số dòng cho bảng
        table.setRowCount(number_of_rows)
        # Đặt tên của các cột
        table.setHorizontalHeaderLabels(columns_name)
        if query:
            # Nếu truyền query thì đặt query_data= query truyền vào
            query_data=query
        else:
            # thực hiện truy vấn thông tin dưới
            query_data = "select * from thong_tin_sv,diem_danh where
thong_tin_sv.mssv=diem_danh.mssv"

        data_table=self.data.execute_query(query_data)
        table_row=0
        # Duyệt qua từng dòng và gán giá trị
        for row in data_table:
            cell_index=0
            for cell in row:
                table_item=QtWidgets.QTableWidgetItem(str(cell))
                if cell_index<5:
                   table_item.setFlags(QtCore.Qt.ItemFlag.ItemIsEnabled)
                table.setItem(table_row,cell_index,table_item)
                cell_index+=1
            table_row+=1
        # Trả về bảng sau khi hoàn thành
        return table
```

- Hàm xử lý nút điểm danh bù

```
def Diem_danh_bu(self):
        # Lấy hàng đang được chọn
        current_row = self.tw_danh_sach_sv.currentRow()
        # Lấy cột đang được chọn
        current_column = self.tw_danh_sach_sv.currentColumn()
        # Tạo một Qtable item (một ô giá trị) với giá trị "Trễ"
        tw_item = QtWidgets.QTableWidgetItem("Trễ")
        # Lấy tên của cột
        date=self.tw_danh_sach_sv.horizontalHeaderItem(current_column).text()
        #lấy mssv
        id=self.tw_danh_sach_sv.item(current_row,0).text()
        # gọi hàm điểm danh bù
        self.data.make_up_attendance(id,date,tw_item.text())
        # Thay đổi giá trị của ô
        self.tw_danh_sach_sv.setItem(current_row, current_column, tw_item)

def make_up_attendance(self,regiter_IDs,date,type=None):
        if type==None:
            type='Đủ'
        query= f"UPDATE diem_danh SET `{date}` = N'{type}' WHERE
mssv={regiter_IDs};"
        self.execute_non_query(query)
```

- hàm xử lý xuất excel

```
def write_excel(self):
        db = mysql.connector.connect(**self.connection_string)
        # columns_name=self.get_columns_name()
        query_data = "select * from thong_tin_sv,diem_danh where
thong_tin_sv.mssv=diem_danh.mssv"
        # sử dụng pandas để đọc dữ liệu trả về từ CSDL
        df = pd.read_sql(query_data, con=db)
        db.close()
        excel_file='D:\FaceAttendace\data.xlsx'
        #xuất file excel qua dường dẫn với tên tạo ở trên
        df.to_excel(excel_file)
        # print(df)
```

- hàm xử lý nút tìm kiếm

```
def search(self):
        # Khởi tạo câu truy vấn mặc đinh
        query = f"""SELECT *
        FROM thong_tin_sv,diem_danh
        WHERE thong_tin_sv.mssv=diem_danh.mssv
        """
        # Khởi tạo câu truy 1 vấn bổ trợ cho câu truy vấn mặc định
```

```
        sub_query1 = (
            # sub_query1 có giá trị AND thong_tin_sv.mssv like '%MSSV%' khi nút
MSSV được chọn
            " AND thong_tin_sv.mssv like '%{}%'
".format(self.txt_search.toPlainText())
            if self.rd_mssv.isChecked()
            # sub_query1 có giá trị AND thong_tin_sv.mssv like '%Họ%' khi nút Họ
được chọn
            else " AND ho like '%{}%'".format(self.txt_search.toPlainText())
            # sub_query1 có giá trị AND thong_tin_sv.mssv like '%Tên%' khi nút
Tên được chọn
            if self.rd_ho.isChecked()
            else " AND ten like '%{}%'".format(self.txt_search.toPlainText())
        )
        # Khởi tạo câu truy vấn 2 bổ trợ cho câu truy vấn mặc định
        sub_query2 = (
            # sub_query2 có giá trị rỗng khi nút Tất cả được chọn
            ""
            if self.rd_all_dd.isChecked()
            # sub_query2 có giá AND {Ngày được chọn} IS NULL trị rỗng khi nút Tất
cả được chọn
            else " AND {} IS NULL".format(self.day)
            if self.rd_chua_dd
            # sub_query2 có giá  AND {} IS NOT NULL AND {} <> 'Đủ' trị rỗng khi
nút Dã ĐD cả được chọn
            else " AND {} IS NOT NULL AND {} <> 'Đủ'".format(self.day,self.day)
        )
        # print("{}{}{}".format(query, sub_query1, sub_query2))
        # Đặt lại gá trị cho bảng
        self.tw_danh_sach_sv.setRowCount(0)
        self.tw_danh_sach_sv=self.handle_app.load_data_to_table(self.tw_danh_sach
_sv,"{}{}{}".format(query, sub_query1, sub_query2))
```

IV. KẾT QUẢ THỰC NGHIỆM

4.1 Độ chính xác của ứng dụng

Thực hiện kiểm tra 6 phiên với số lần nhận dạng khác nhau cho ra kết quả:

STT	Max (giây)	Min (giây)	Số lần kiểm thử	Trung bình	Đúng	Sai	Tỉ lệ
1	1.703637	0.360993	95	0.487399	94	1	98.95
2	1.09928	1.09928	95	0.455209	94	1	98.95

3	1.706521	0.3425	176	0.420114	174	2	98.86
4	1.91047	0.001	223	0.451531	223	0	100
5	4.651072	0.397732	125	0.526063	223	0	100
6	3.458012	0.408387	294	0.525175	294	1	99.65

Bảng tổng kết kết quả đạt được:

Tổng kết	Max	Min	Trung bình	Tỉ lệ	Nhà SX công bố	Sai lệch
	4.651072	0.001	0.478	99.37	99.38	0.01

Dựa trên bản số liệu thấy được rằng độ chính xác khi nhận diện là rất cao, lên tới con số 99,37.Tuy nhiên theo thực nghiệm, không thể nhận ra được người khi đeo kính (với hình ảnh trong cơ sở dữ liệu là không đeo kính) và ngược lại.

Vì vậy hạn chế lớn nhất của ứng dụng đó là nhận diện không chính xác khi khuôn mặt người dùng có sự biến đổi lớn.

4.2 Đánh giá hiệu năng xử lý của ứng dụng

Dựa theo bản ở trên có thể thấy được tốc độ sử lý là cực cực kì nhanh, có thể mất chưa tới 0,001 giây để nhận dạng. Tuy nhiên do ảnh hưởng bởi góc độ, ánh sáng thì việc nhận diện lại có thể mất rất nhiều thời gian có thể lên tới 4,65 giây.

4.3 Kết quả đạt được

Hiểu được cách hoạt động và phương pháp nhận diện bằng học máy.

Xây dựng thành công ứng dụng với các chức năng quan trọng đáp ứng được tốt các yêu cầu chính của đề tài. Và hoàn thiện các chức năng bao gồm:

- Chức năng tìm kiếm thông tin của sinh viên.
- Chức năng điểm danh bằng nhận diện khuôn mặt.
- Chức năng điểm danh bù.
- Chức năng xuất tập tin excel

4.4 Ưu, nhược điểm của ứng dụng

Ưu điểm của ứng dụng:

- Thời gian điểm danh sinh viên giảm thiểu rất nhiều.
- Giải phóng thời gian dư thừa cho việc điểm danh.
- Tỉ lệ chính xác cao khi nhận diện.
- Công bằng minh bạch với người dùng.
- Dễ dàng thao tác và sử dụng.

Nhược chế của ứng dụng

- Ứng dụng chỉ sử dụng được trên các máy có cấu hình trung bình hoặc mạnh và có yêu cầu phải có camera, webcam.
- Tức là tỉ lệ sai sót cao khi dung mạo người nhận diện có sự thay đổi lớn.
- Có thể ảnh hưởng bởi độ sáng, và góc nhìn.

4.5 Đề xuất cải tiến và hướng phát triển của ứng dụng

Cải tiến

- Tối ưu hóa các tham số của thuật toán để cải thiện tốc độ và độ chính xác.
- Sử dụng phần cứng mạnh hơn như GPU để xử lý song song nhanh hơn.
- Kết hợp thêm các thuật toán khác để nâng cao độ chính xác khi nhận diện.

Hướng phát triển

- Phát triển ứng dụng thành ứng dụng đa nền tảng.
- Sử dụng trên các thiết bị IoT để có thể điểm danh một cách tự động nhất có thể.
- Bổ sung các chức năng nhận diện cảm xúc cũng như hành động của sinh viên trong lớp học.
- Cho phép thống kê, phân tích các số liệu mà ứng dụng thu thập được

V. KẾT LUẬN

Trong phạm vi tiểu luận này, em đã nghiên cứu các phương pháp nhận dạng khuôn mặt và ứng dụng trong điểm danh tự động. Em đã xây dựng thành công ứng dụng với các chức năng cơ bản: nhận dạng điểm danh, điểm danh bù cũng như tìm kiếm và lưu trữ tệp tin. Ứng dụng sử dụng thư viện face_recognition của Python, một thư viện mạnh mẽ cho nhận dạng khuôn mặt.

Qua đó, em đã làm hiểu và biết cách vận hành, sử dụng được quy trình xây dựng một hệ thống nhận dạng khuôn mặt và ứng dụng trong thực tế. Kết quả của tiểu luận cũng chứng minh tính khả thi và hiệu quả của việc áp dụng công nghệ nhận dạng khuôn mặt trong điểm danh tự động. Tuy nhiên, do giới hạn về thời gian, ứng dụng em xây dựng vẫn còn có thể cải thiện và bổ sung thêm nhiều tính năng.

VI. TÀI LIỆU THAM KHẢO

Nguyễn Bảo Long (2022). Xây dựng hệ thống điểm danh sinh viên dựa trên nhận diện khuôn mặt. Trường Đại học Đà Lạt. Lấy từ https://dlu.edu.vn/wp-content/uploads/2022/06/BaoCaoToanVan.pdf

Adam Geitgey (2015). face_recognition. GitHub. Lấy từ https://github.com/ageitgey/face_recognition

Murtaza (2020). Face Recognition and Attendance Project. GitHub. Lấy từ https://github.com/murtazahassan/Face-Recognition#face-recognition-and-attendance-project