**Báo cáo dự án: Hệ thống điểm danh khuôn mặt**

**1. Giới thiệu**

Dự án này trình bày một hệ thống điểm danh tự động sử dụng công nghệ nhận dạng khuôn mặt. Hệ thống được phát triển bằng Python, kết hợp các thư viện face\_recognition (dựa trên dlib), OpenCV để xử lý hình ảnh và Tkinter để xây dựng giao diện người dùng đồ họa (GUI). Mục tiêu chính là cung cấp một giải pháp hiệu quả để theo dõi sự có mặt của sinh viên hoặc nhân viên một cách tự động và chính xác.

Hệ thống bao gồm hai giai đoạn chính:

1. **Giai đoạn huấn luyện (Training):** Thu thập và trích xuất các đặc trưng khuôn mặt từ dữ liệu ảnh đã biết, sau đó lưu trữ chúng.
2. **Giai đoạn điểm danh thời gian thực (Real-time Attendance):** Sử dụng webcam để phát hiện và nhận dạng khuôn mặt, cập nhật trạng thái điểm danh trên giao diện người dùng.

**2. Nguyên lý hoạt động**

Hệ thống hoạt động dựa trên nguyên lý so sánh các đặc trưng khuôn mặt. Mỗi khuôn mặt được biểu diễn dưới dạng một tập hợp các con số (encoding) mà máy tính có thể hiểu và so sánh được.

1. **Trích xuất đặc trưng (Feature Extraction):** Khi một khuôn mặt được đưa vào hệ thống (từ ảnh huấn luyện hoặc từ webcam), thuật toán sẽ phân tích khuôn mặt đó để tạo ra một "dấu vân tay" số duy nhất, gọi là face encoding. Encoding này là một vector 128 chiều.
2. **So sánh (Comparison):** Khi một khuôn mặt mới được phát hiện, encoding của nó sẽ được so sánh với tất cả các encoding đã biết trong cơ sở dữ liệu.
3. **Quyết định nhận dạng (Recognition Decision):** Dựa trên "khoảng cách" giữa các encoding (thường là khoảng cách Euclidean), hệ thống sẽ xác định xem khuôn mặt mới có khớp với bất kỳ khuôn mặt nào đã biết hay không. Nếu khoảng cách nhỏ hơn một ngưỡng nhất định, khuôn mặt được coi là đã nhận dạng thành công.

**3. Chi tiết lý thuyết và hoạt động**

**3.1. Trích xuất đặc trưng khuôn mặt (Face Feature Extraction)**

Thư viện face\_recognition sử dụng các thuật toán mạnh mẽ từ thư viện dlib để xử lý khuôn mặt.

* **Phát hiện vị trí khuôn mặt (face\_recognition.face\_locations):**
  + Hàm này được sử dụng để xác định tọa độ (top, right, bottom, left) của tất cả các khuôn mặt trong một hình ảnh.
  + Mặc định, nó sử dụng mô hình "HOG" (Histogram of Oriented Gradients). HOG là một bộ mô tả đặc trưng được sử dụng trong thị giác máy tính và xử lý ảnh để phát hiện đối tượng. Nó hoạt động bằng cách đếm số lần xuất hiện của gradient hướng trong các vùng cục bộ của hình ảnh.
  + Mô hình HOG nhanh và hiệu quả cho nhiều trường hợp sử dụng, đặc biệt là trên CPU. Đối với hiệu suất cao hơn trên GPU, có thể sử dụng mô hình "CNN" (Convolutional Neural Network) bằng cách thêm model="cnn", tuy nhiên nó yêu cầu cài đặt dlib với hỗ trợ CUDA.
* **Trích xuất Encoding (face\_recognition.face\_encodings):**
  + Sau khi vị trí khuôn mặt được xác định, hàm này sẽ lấy vùng khuôn mặt đó và tạo ra một vector 128 chiều. Vector này được tạo ra bởi một mạng nơ-ron sâu đã được huấn luyện trên một tập dữ liệu lớn để học cách phân biệt các khuôn mặt khác nhau.
  + Mạng này được huấn luyện để tạo ra các encoding sao cho khuôn mặt của cùng một người sẽ có các encoding rất gần nhau trong không gian 128 chiều, trong khi khuôn mặt của những người khác nhau sẽ có các encoding cách xa nhau. Đây là bước quan trọng nhất, biến hình ảnh khuôn mặt thành một dạng dữ liệu có thể so sánh được về mặt toán học.

**3.2. So sánh khuôn mặt (Face Comparison)**

* **Tính khoảng cách khuôn mặt (face\_recognition.face\_distance):**
  + Hàm này tính toán khoảng cách Euclidean giữa một encoding khuôn mặt mới và một danh sách các encoding khuôn mặt đã biết.
  + Công thức khoảng cách Euclidean giữa hai vector A=(a\_1,a\_2,...,a\_n) và B=(b\_1,b\_2,...,b\_n) là:

d(A,B)=i=1∑n​(ai​−bi​)2​

* + Khoảng cách này càng nhỏ, hai khuôn mặt càng giống nhau.
  + Trong code, min\_distance được tìm bằng cách lấy giá trị nhỏ nhất trong mảng các khoảng cách.
* **Ngưỡng nhận dạng (Thresholding):**
  + Một ngưỡng (threshold) được sử dụng để quyết định xem hai khuôn mặt có đủ giống nhau để được coi là cùng một người hay không. Trong code, ngưỡng này được đặt là 0.6.
  + Nếu min\_distance < 0.6, khuôn mặt được coi là khớp với match\_name. Ngược lại, nó được coi là "Unknown".
  + Việc lựa chọn ngưỡng là rất quan trọng:
    - Ngưỡng quá thấp có thể dẫn đến nhiều trường hợp "Unknown" (False Negative) vì hệ thống quá khắt khe.
    - Ngưỡng quá cao có thể dẫn đến nhận dạng sai người (False Positive) vì hệ thống quá dễ dãi.
  + Ngưỡng 0.6 là một giá trị phổ biến và thường hoạt động tốt với thư viện face\_recognition.

**3.3. Luồng xử lý**

**3.3.1. Giai đoạn huấn luyện (train\_faces function)**

1. **Duyệt thư mục dữ liệu:** Hàm quét qua thư mục dataset\_dir. Mỗi thư mục con trong đó được coi là một ID sinh viên (tên của sinh viên).
2. **Đọc và xử lý ảnh:** Trong mỗi thư mục sinh viên, hệ thống duyệt qua tất cả các file ảnh.
3. **Phát hiện và trích xuất:** Với mỗi ảnh, nó gọi face\_recognition.face\_locations để tìm khuôn mặt và sau đó face\_recognition.face\_encodings để tạo ra encoding 128 chiều.
4. **Lưu trữ encoding:** Các encoding của cùng một sinh viên được thu thập vào một danh sách.
5. **Lưu vào file:** Cuối cùng, tất cả các encoding đã biết (dưới dạng một từ điển {student\_name: [list\_of\_encodings]}) được lưu vào một file .pkl (student\_encodings.pkl) bằng pickle để sử dụng sau này mà không cần huấn luyện lại.

**3.3.2. Giai đoạn điểm danh thời gian thực (AttendanceGUI class và \_update\_frame method)**

1. **Tải encoding:** Khi ứng dụng GUI khởi động, nó tải các encoding đã lưu từ file student\_encodings.pkl.
2. **Khởi tạo Webcam:** Mở kết nối với webcam (cv2.VideoCapture(0)).
3. **Vòng lặp xử lý khung hình:**
   * Đọc từng khung hình từ webcam.
   * **Tiền xử lý:** Giảm kích thước khung hình (cv2.resize) để tăng tốc độ xử lý nhận dạng, sau đó chuyển đổi sang định dạng RGB.
   * **Phát hiện và trích xuất:** Phát hiện khuôn mặt và trích xuất encoding từ khung hình hiện tại.
   * **So sánh và nhận dạng:** So sánh encoding của khuôn mặt được phát hiện với tất cả các encoding đã biết để tìm ra khuôn mặt khớp nhất.
   * **Cập nhật trạng thái:** Nếu khuôn mặt được nhận dạng là một ID đã biết và chưa được điểm danh, trạng thái điểm danh của ID đó sẽ được cập nhật.
   * **Vẽ lên khung hình:** Vẽ một hình chữ nhật quanh khuôn mặt và hiển thị tên của người được nhận dạng (hoặc "Unknown").
   * **Hiển thị GUI:** Chuyển đổi khung hình OpenCV sang định dạng Tkinter và hiển thị lên video\_frame. Cập nhật nhãn "ID được phát hiện" và danh sách điểm danh.
   * **Lặp lại:** Hàm \_update\_frame tự gọi lại sau mỗi 10ms để tạo hiệu ứng video liên tục.
4. **Xử lý đóng ứng dụng:** Khi người dùng đóng cửa sổ, webcam được giải phóng và ứng dụng thoát.

**4. Cấu trúc ứng dụng GUI**

Lớp AttendanceGUI quản lý toàn bộ giao diện và logic điểm danh:

* **\_\_init\_\_:** Khởi tạo cửa sổ Tkinter, tải encoding, khởi tạo trạng thái điểm danh và webcam, sau đó gọi \_create\_widgets và \_update\_frame.
* **\_load\_encodings:** Tải dữ liệu encoding từ file .pkl.
* **\_create\_widgets:**
  + Thiết lập cấu trúc layout chính với ttk.Frame và grid.
  + video\_frame: ttk.Label để hiển thị video từ webcam.
  + detected\_id\_frame: ttk.LabelFrame chứa detected\_id\_label để hiển thị ID được phát hiện.
  + attendance\_list\_frame: ttk.LabelFrame chứa tk.Canvas và ttk.Scrollbar để hiển thị danh sách điểm danh có thể cuộn.
* **\_populate\_attendance\_list:** Điền tên tất cả sinh viên đã biết vào danh sách điểm danh ban đầu với trạng thái "Chưa điểm danh".
* **\_update\_attendance\_list:** Cập nhật trạng thái điểm danh của một sinh viên cụ thể trên giao diện khi họ được nhận dạng.
* **\_update\_frame:** Hàm cốt lõi thực hiện vòng lặp xử lý video, nhận dạng khuôn mặt và cập nhật GUI.
* **on\_closing:** Xử lý khi người dùng đóng cửa sổ, đảm bảo giải phóng tài nguyên webcam.

**5. Hướng dẫn sử dụng**

1. **Cài đặt thư viện:**
2. pip install face\_recognition opencv-python Pillow
3. **Chuẩn bị dữ liệu huấn luyện:**
   * Tạo một thư mục gốc cho dữ liệu, ví dụ: AI1901\_face\_dataset.
   * Bên trong thư mục này, tạo các thư mục con với tên của từng sinh viên (ví dụ: NguyenVanA, TranThiB).
   * Đặt ít nhất vài ảnh khuôn mặt rõ nét của mỗi sinh viên vào thư mục tương ứng của họ. Đảm bảo ảnh có nhiều góc độ, điều kiện ánh sáng khác nhau để tăng độ chính xác.
4. AI1901\_face\_dataset/
5. ├── NguyenVanA/
6. │ ├── img1.jpg
7. │ ├── img2.png
8. │ └── ...
9. ├── TranThiB/
10. │ ├── img1.jpg
11. │ └── ...
12. └── ...
13. **Cập nhật đường dẫn:**
    * Trong file code, tìm dòng dataset\_dir = r'D:\Documents\Learning\FPT\SU25\CPV\excersice\Project\Code\CPV\AI1901\_face\_dataset' và thay đổi nó thành đường dẫn thực tế đến thư mục dữ liệu của bạn.
14. **Huấn luyện mô hình:**
    * Trong khối if \_\_name\_\_ == "\_\_main\_\_":, bỏ comment dòng train\_faces(dataset\_dir, output\_encodings\_file).
    * Chạy script. Quá trình này sẽ mất một thời gian tùy thuộc vào số lượng ảnh và tốc độ máy tính của bạn. Sau khi hoàn thành, một file student\_encodings.pkl sẽ được tạo ra.
    * Sau khi file student\_encodings.pkl đã được tạo, bạn nên comment lại dòng train\_faces(...) để không phải huấn luyện lại mỗi lần chạy ứng dụng điểm danh.
15. **Chạy ứng dụng điểm danh:**
    * Chạy lại script. Một cửa sổ GUI sẽ xuất hiện, hiển thị video từ webcam và bắt đầu nhận dạng khuôn mặt.
    * Khi một khuôn mặt được nhận dạng, tên của người đó sẽ hiển thị ở khung "ID được phát hiện" và trạng thái điểm danh trong danh sách sẽ được cập nhật.

**6. Ưu điểm**

* **Tự động hóa:** Giảm thiểu công sức và thời gian cho việc điểm danh thủ công.
* **Chính xác:** Sử dụng các thuật toán nhận dạng khuôn mặt tiên tiến để đạt độ chính xác cao.
* **Thời gian thực:** Khả năng nhận dạng và cập nhật trạng thái điểm danh ngay lập tức.
* **Giao diện thân thiện:** GUI trực quan, dễ sử dụng cho người vận hành.
* **Khả năng mở rộng:** Dễ dàng thêm mới sinh viên/nhân viên vào hệ thống bằng cách thêm ảnh vào thư mục dữ liệu và chạy lại huấn luyện.
* **Tiết kiệm chi phí:** Không yêu cầu phần cứng chuyên dụng ngoài webcam thông thường.

**7. Nhược điểm**

* **Yêu cầu dữ liệu huấn luyện:** Cần một lượng lớn ảnh chất lượng cao và đa dạng của mỗi người để huấn luyện hiệu quả. Chất lượng ảnh kém hoặc số lượng ảnh ít có thể làm giảm độ chính xác.
* **Điều kiện môi trường:** Hiệu suất có thể bị ảnh hưởng bởi điều kiện ánh sáng kém, góc nhìn khuôn mặt, vật cản (kính, khẩu trang), hoặc biểu cảm khuôn mặt thay đổi.
* **Tính riêng tư:** Việc thu thập và lưu trữ dữ liệu khuôn mặt có thể đặt ra các vấn đề về quyền riêng tư và bảo mật dữ liệu.
* **Tốc độ xử lý:** Mặc dù đã được tối ưu bằng cách giảm kích thước khung hình, việc xử lý nhận dạng khuôn mặt vẫn có thể tốn tài nguyên CPU, đặc biệt với số lượng khuôn mặt lớn hoặc trên các máy tính cấu hình thấp.
* **Độ nhạy cảm với ngưỡng:** Việc chọn ngưỡng nhận dạng phù hợp là rất quan trọng và có thể cần điều chỉnh thủ công để cân bằng giữa False Positive và False Negative.

**8. Hướng cải tiến**

* **Cải thiện tốc độ và hiệu suất:**
  + Sử dụng mô hình "CNN" của dlib (nếu có GPU) để tăng tốc độ phát hiện và trích xuất khuôn mặt.
  + Tối ưu hóa kích thước khung hình xử lý hoặc tần suất xử lý để cân bằng giữa hiệu suất và độ chính xác.
* **Quản lý dữ liệu nâng cao:**
  + Xây dựng cơ sở dữ liệu (ví dụ: SQLite, MySQL) thay vì file .pkl để quản lý thông tin sinh viên và encoding một cách có cấu trúc hơn.
  + Thêm chức năng thêm/xóa/cập nhật sinh viên trực tiếp từ GUI mà không cần thao tác thủ công với thư mục file.
* **Báo cáo và thống kê:**
  + Lưu trữ lịch sử điểm danh (thời gian, ID) vào file hoặc cơ sở dữ liệu.
  + Tạo các báo cáo điểm danh theo ngày, tuần, tháng.
  + Hiển thị thống kê trực quan trên GUI (ví dụ: số lượng đã điểm danh, chưa điểm danh).
* **Xử lý đa khuôn mặt:**
  + Hiện tại, hệ thống xử lý từng khuôn mặt một. Có thể tối ưu để xử lý đồng thời nhiều khuôn mặt trong một khung hình hiệu quả hơn.
  + Thêm khả năng nhận dạng và điểm danh nhiều người cùng lúc.
* **Giao diện người dùng:**
  + Cải thiện thẩm mỹ và trải nghiệm người dùng của GUI (ví dụ: thêm icon, hiệu ứng, tùy chỉnh theme).
  + Thêm các nút điều khiển (bắt đầu/dừng điểm danh, chụp ảnh, v.v.).
* **Xử lý trường hợp ngoại lệ:**
  + Cải thiện khả năng xử lý các trường hợp khuôn mặt bị che khuất, ánh sáng kém, hoặc chất lượng ảnh thấp.
  + Thêm cơ chế cảnh báo khi phát hiện khuôn mặt "Unknown" liên tục.
* **Tích hợp với hệ thống khác:**
  + Khả năng tích hợp với các hệ thống quản lý sinh viên hoặc hệ thống chấm công hiện có.
* **Độ bền và bảo mật:**
  + Thêm các biện pháp bảo mật để ngăn chặn việc giả mạo khuôn mặt (ví dụ: liveness detection để phân biệt ảnh/video với người thật).
  + Mã hóa dữ liệu encoding để tăng cường bảo mật.

Hy vọng báo cáo này cung cấp cái nhìn tổng quan và chi tiết về dự án của bạn!