|  |
| --- |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| ĐỒ ÁN TỐT NGHIỆP  **NGHIÊN CỨU, XÂY DỰNG ỨNG DỤNG NHẬN DIỆN KHUÔN MẶT TRÊN ANDROID** |
| Ngành: An toàn thông tin  Mã số: |
| Hà Nội, 2020 |
| BAN CƠ YẾU CHÍNH PHỦ  **HỌC VIỆN KỸ THUẬT MẬT MÃ**  ¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯¯  Logo HvKTMM |
| ĐỒ ÁN TỐT NGHIỆP  **NGHIÊN CỨU, XÂY DỰNG ỨNG DỤNG NHẬN DIỆN KHUÔN MẶT TRÊN ANDROID** |
| Ngành: An toàn thông tin  Mã số:  *Sinh viên thực hiện*:  Mã SV: AT120626 Họ và tên: **Phạm Quang Huy**    *Người hướng dẫn 1*:  **TS. Phạm Duy Trung**  Khoa An toàn thông tin – Học viện Kỹ thuật mật mã  *Người hướng dẫn 2*:  **TS. Nguyễn Nhất Hải**  Viện Công nghệ thông tin và Truyền Thông – Đại học Bách Khoa Hà Nội |
| Hà Nội, 2020 |

Mục lục

MỤC LỤC

[Danh mục kí hiệu và viết tắt 5](#_Toc40450657)

[Danh mục hình vẽ 6](#_Toc40450658)

[Danh mục bảng 8](#_Toc40450659)

[Lời cảm ơn 9](#_Toc40450660)

[Lời nói đầu 10](#_Toc40450661)

[CHƯƠNG 1. CƠ SỞ LÝ THUYẾT VÀ BÀI TOÁN TÌM KIẾM KHUÔN MẶT 1](#_Toc40450662)

[1.1. Khái quát về xử lý ảnh 1](#_Toc40450663)

[1.1.1. Khái niệm xử lý ảnh 1](#_Toc40450664)

[1.1.2. Một số vấn đề trong xử lý ảnh 1](#_Toc40450665)

[1.1.3. Các hình thái của ảnh 2](#_Toc40450666)

[1.1.4. Một số ứng dụng của xử lý ảnh 3](#_Toc40450667)

[1.2. Bài toán phát hiện khuôn mặt 4](#_Toc40450668)

[1.2.1. Một số đặc trưng về khuôn mặt 4](#_Toc40450669)

[1.2.2. Tìm hiểu về bài toán phát hiện khuôn mặt 5](#_Toc40450670)

[1.2.3. Các phương pháp chính phát hiện mặt người 6](#_Toc40450671)

[1.2.4. Những khó khăn trong việc phát hiện khuôn mặt người 7](#_Toc40450672)

[1.3. Tìm hiểu về phương pháp phát hiện khuôn mặt của Viola và Johns 8](#_Toc40450673)

[1.3.1. Tổng quan về phương pháp 8](#_Toc40450674)

[1.3.2. Tiền xử lý ảnh 9](#_Toc40450675)

[1.3.3. Đặc trưng Haar-like 9](#_Toc40450676)

[1.3.4. Cách áp dụng đặc trưng Harr-like để phát hiện khuôn mặt trong ảnh 11](#_Toc40450677)

[1.3.5. Ảnh tích hợp (Integral image) 12](#_Toc40450678)

[1.3.6. Mô hình phân loại theo tầng Cascade of classifier 14](#_Toc40450679)

[CHƯƠNG 2. BÀI TOÁN NHẬN DẠNG MẶT NGƯỜI 17](#_Toc40450680)

[2.1. Tổng quan về bài toán nhận dạng mặt người 17](#_Toc40450681)

[2.1.1. Giới thiệu 17](#_Toc40450682)

[2.1.2. Thách thức trong việc nhận dạng mặt người 18](#_Toc40450683)

[2.1 3. Ứng dụng của nhận diện khuôn mặt 18](#_Toc40450684)

[2.2. Tiếp cận bài toán nhận diện khuôn mặt 19](#_Toc40450685)

[2.2.1. Một số bài toán về khuôn mặt 19](#_Toc40450686)

[2.2.2. Mô hình cơ bản của bài toán nhận diện khuôn mặt 21](#_Toc40450687)

[2.2.3. Một số thuật toán nhận diên khuôn mặt 24](#_Toc40450688)

[2.3. Mô hình FaceNet trong nhận dạng khuôn mặt 27](#_Toc40450689)

[2.3.1. Mạng CNN – Convolutional Neural Network 27](#_Toc40450690)

[2.3.2. Tìm hiểu mạng FaceNet 32](#_Toc40450691)

[2.3.3. Đánh giá thực nghiệm mô hình FaceNet 36](#_Toc40450692)

[CHƯƠNG 3. XÂY DỰNG HỆ THỐNG NHẬN Diện KHUÔN MẶT TRÊN NỀN TẢNG ĐIỆN THOẠI ANDROID 38](#_Toc40450693)

[3.1. Tổng quan về hệ thống nhận diện khuôn mặt trên thiết bị di động 38](#_Toc40450694)

[3.1.1. Yêu cầu của hệ thống 38](#_Toc40450695)

[3.1.2. Đặc điểm nhận dạng khuôn mặt trên thiết bị di động 38](#_Toc40450696)

[3.1.3. Công cụ, thư viện sử dụng 39](#_Toc40450697)

[3.1.4. Mô hình hoạt động của hệ thống 39](#_Toc40450698)

[3.2. Thiết kế các chức năng chính và giao diện ứng dụng 40](#_Toc40450699)

[3.2.1. Các chức năng chính của hệ thống 40](#_Toc40450700)

[3.2.2. Giao diện ứng dụng 41](#_Toc40450701)

[3.3. Xây dựng hệ thống 44](#_Toc40450702)

[3.3.1. Các lớp dữ liệu 44](#_Toc40450703)

[3.3.2. Quá trình thực hiện trong ứng dụng 48](#_Toc40450704)

[3.3.3. Thực hiện xây dựng các chức năng 49](#_Toc40450705)

[3.4. Thử nghiệm ứng dụng 52](#_Toc40450706)

[3.4.1. Môi trường thực hiện ứng dụng 52](#_Toc40450707)

[3.4.2. Kết quả thử nghiệm 52](#_Toc40450708)

[KẾT LUẬN 53](#_Toc40450709)

[Kết quả đã thực hiện được của đồ án 53](#_Toc40450710)

[Hướng phát triển đồ án 53](#_Toc40450711)

[TÀI LIỆU THAM KHẢO 54](#_Toc40450712)

Danh mục kí hiệu và viết tắt

CNN: Convolutional Neural Network – Mạng noron tích chập

Danh mục hình vẽ

[Hình 1.1 Sự khác nhau của khuôn mặt người 5](#_Toc40450713)

[Hình 1.2 Bốn đặc trưng Haar-like 10](#_Toc40450714)

[Hình 1.3 Các đặc trưng Haar-like được đặt lên tại khung hình 11](#_Toc40450715)

[Hình 1.4 Ảnh tích hợp 12](#_Toc40450716)

[Hình 1.5 Vùng giá trị điểm ảnh cần tính 13](#_Toc40450717)

[Hình 1.6 Kết hợp các bộ phân loại yếu thành bộ phân loại mạnh 14](#_Toc40450718)

[Hình 1.7 Mô hinh Cascade of classifier 15](file:///D:\DoAn\FaceRecognize\Bao%20cao.docx#_Toc40450719)

[Hình 2. 1 Khuôn mặt đại diện nữ của một số quốc gia 20](#_Toc40450720)

[Hình 2. 2 Mô hình hệ thống nhận diện khuôn mặt 21](#_Toc40450721)

[Hình 2. 3 Cân bằng sáng cho ảnh 22](#_Toc40450722)

[Hình 2. 4 Ảnh bị nhiễu 22](#_Toc40450723)

[Hình 2. 5 Xử lý nhiễu của ảnh 23](#_Toc40450724)

[Hình 2. 6 Mô tả phương pháp Learning similarity 24](#_Toc40450725)

[Hình 2. 7 Mô hình Siam network 26](#_Toc40450726)

[Hình 2. 8 Mô hình mạng CNN 28](#_Toc40450727)

[Hình 2. 9 Ma trận đầu vào và kernel 28](#_Toc40450728)

[Hình 2. 10 Tính toán feature map 29](#_Toc40450729)

[Hình 2. 11 Kết quả quá trình Convolution 29](#_Toc40450730)

[Hình 2. 12 Max Pooling 31](#_Toc40450731)

[Hình 2. 13 Layer Fully connected 31](#_Toc40450732)

[Hình 2. 14 Mục tiêu của hàm Triplet loss 34](#_Toc40450733)

[Hình 2. 15 Mô hình hoạt động nhận dạng khuôn mặt bằng mạng FaceNet 36](file:///D:\DoAn\FaceRecognize\Bao%20cao.docx#_Toc40450734)

[Hình 3. 1 Mô hình hoạt động nhận diện khuôn mặt 39](#_Toc40450735)

[Hình 3. 2 Các chức năng của hệ thống 40](#_Toc40450736)

[Hình 3. 3 Màn hình chính của ứng dụng 41](#_Toc40450737)

[Hình 3. 4 Màn hình thêm người mới 42](#_Toc40450738)

[Hình 3. 5 Màn hình nhận diện khuôn mặt 43](#_Toc40450739)

[Hình 3. 6 Màn hình quản lý 44](#_Toc40450740)

[Hình 3. 7 Class MainActivity 45](#_Toc40450741)

[Hình 3. 8 Class TrainActivity 45](#_Toc40450742)

[Hình 3. 9 Class RecognizeActivity 46](#_Toc40450743)

[Hình 3. 10 Class ManageActivity 46](#_Toc40450744)

[Hình 3. 11 Class PersonAdapter 47](#_Toc40450745)

[Hình 3. 12 Class ImageUtils 47](#_Toc40450746)

[Hình 3. 13 Class FileUtils 47](#_Toc40450747)

[Hình 3. 14 Class RecognizeFaceUtils 48](#_Toc40450748)

[Hình 3. 15 Class DetectFaceUtils 48](#_Toc40450749)

[Hình 3. 16 Quá trình huấn luyện khuôn mặt 48](#_Toc40450750)

[Hình 3. 17 Quá trình nhận dạng khuôn mặt 49](#_Toc40450751)

Danh mục bảng

[Bảng 2.1 Bảng so sánh hiệu suất của các kiến trúc trong mạng FaceNet 33](#_Toc40450752)

[Bảng 2.2 Bảng đánh giá độ chính xác mô hình FaceNet 37](#_Toc40450753)

Lời cảm ơn

Trước tiên em xin được bày tỏ sự trân trọng và lòng biết ơn đối với thầy giáo TS. Phạm Duy Trung – Khoa An toàn thông tin và thầy giáo TS. Nguyễn Nhất Hải - Viện Công nghệ thông tin và Truyền Thông Đại học Bách Khoa Hà Nội. Trong suốt thời gian làm đồ án tốt nghiệp, các thầy đã dành rất nhiều thời gian quý báu để tận tình chỉ bảo, hướng dẫn, định hướng cho em thực hiện đồ án.

Em xin được cảm ơn các thầy cô giáo Học viện Kỹ thuật Mật Mã đã giảng dạy trong quá trình học tập, thực hành, làm bài tập, giúp em hiểu thấu đáo hơn các nội dung học tập và những hạn chế cần khắc phục trong việc học tập, nghiên cứu và thực hiện bản đồ án này.

Em xin cảm ơn các bạn bè và nhất là các thành viên trong gia đình đã tạo mọi điều kiện tốt nhất, động viên, cổ vũ trong suốt quá trình học tập và đồ án tốt nghiệp.

|  |  |
| --- | --- |
|  | **SINH VIÊN THỰC HIỆN ĐỒ ÁN**  Phạm Quang Huy |
|  |  |

Lời nói đầu

Ngày nay, cùng với những tiến bộ vuợt bậc của khoa học kỹ thuật nói chung, bộ môn khoa học xử lý ảnh đã và đang thu được những thành tựu lớn lao và chứng tỏ vài trò không thể thiếu với những ứng dụng sâu rộng trong khoa học kỹ thuật cũng như đời sống xã hội. Một bộ phận của khoa học xử lý ảnh là lĩnh vực thị giác máy tính hiện đang thu hút rất nhiều sự quan tâm của các nhà nghiên cứu xử lý ảnh với mục tiêu xây dựng nên một thế giới trong đó hệ thống thị giác kỳ diệu của con người có thể được mô phỏng bởi các hệ thống máy tính, đem lại khả năng cảm nhận bằng thị giác cho các hệ thống về môi trường xung quanh.

Mơ ước về một hệ thống máy tính có thể hoà nhập vào thế giới con người với đầy đủ các giác quan trong đó thị giác đóng vai trò quan trọng đang dần dần được hiện thực hoá với những đóng góp nghiên cứu của các nhà khoa học trên phạm vi toàn thế giới. Một trong những bài toán nhận được nhiều sự quan tâm của lĩnh vực thị giác máy tính là bài toán nhận dạng khuôn mặt.

Đồng thời trong những năm gần đây, điện thoại di động dần trở thành một phần không thể thiếu trong cuộc sống của con người hiện đại, đồng thời xu hướng phát triển các ứng dụng trên các nền tảng di động đang ngày càng phổ biến. Hiện nay, các thế hệ điện thoại thông minh mới có tốc độ xử lý và khả năng lưu trữ ngày càng lớn. Nhận dạng khuôn mặt trên thiết bị di động vì thế sẽ có tiềm năng để phát triển để trợ giúp con người trong nhiều tác vụ khác nhau.

Vì vậy, tôi đã chọn đề tài tốt nghiệp của mình là “*Nghiên cứu, xây dựng ứng dụng nhận diện khuôn mặt trên Android*” nhằm tạo ra một ứng dụng có thể nhận dạng một người nào đó thông qua khuôn mặt của họ. Việc tạo ra một sản phẩm thực tế sẽ giúp tiết kiệm thời gian và công sức cho các công việc như điểm danh sinh viên, xác thực bảo vệ quyền riêng tư cá nhân, …

Đồ án của tôi gồm có 3 phần chính:

Phần 1: Cơ sở lý thuyết và bài toán tìm kiếm khuôn mặt.

Ở phần này chúng ta sẽ tìm hiểu về xử lý ảnh và thực hiện bài toán tìm kiếm khuôn mặt có trong ảnh.

Phần 2: Bài toán nhận diện khuôn mặt.

Tìm hiểu cách thức nhận diện các khuôn mặt sau khi đã tìm được trong ảnh.

Phần 3: Xây dựng phần mềm nhận diện khuôn mặt.

Thực hiện triển khai việc xây dựng hệ thống nhận diện khuôn mặt trên nền tảng điện thoại chạy hệ điều hành Android.

Sau thời gian khoảng ba tháng thực hiện đồ án, các mục tiêu về cơ bản đã đạt được. Tuy nhiên thị giác máy tính là lĩnh vực khoa học còn mới và phức tạp, thời gian thực hiện đồ án tương đối ngắn nên chắc chắn không tránh khỏi thiếu sót. Rất mong được sự góp ý của các thầy cô, cũng như các bạn học để đồ án này được hoàn thiện hơn.

**SINH VIÊN THỰC HIỆN ĐỒ ÁN**

**Phạm Quang Huy**

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT VÀ BÀI TOÁN TÌM KIẾM KHUÔN MẶT

## 1.1. Khái quát về xử lý ảnh

### 1*.1.1. Khái niệm xử lý ảnh*

Quá trình xử lý nhận dạng ảnh được xem như là quá trình thao tác ảnh đầu vào nhằm cho ra kết quả mong muốn. Xử lý ảnh là một trong những mảng quan trọng nhất trong kỹ thuật thị giác máy tính, là tiền đề cho nhiều nghiên cứu thuộc lĩnh vực này.

Hai nhiệm vụ cơ bản của quá trình xử lý ảnh là nâng cao chất lượng thông tin hình ảnh và xử lý số liệu cung cấp cho các quá trình khác trong đó có việc ứng dụng thị giác vào điều khiển.

Quá trình bắt đầu từ việc thu nhận ảnh nguồn (từ các thiết bị thu nhận ảnh dạng số hoặc tương tự) gửi đến máy tính. Dữ liệu ảnh được lưu trữ ở định dạng phù hợp với quá trình xử lý. Người lập trình sẽ tác động các thuật toán tương ứng lên dữ liệu ảnh nhằm thay đổi cấu trúc ảnh phù hơp với các ứng dụng khác nhau.

### 1.1.2. Một số vấn đề trong xử lý ảnh

a. Tiền xử lý

Tiền xử lý là giai đoạn đầu tiên trong xử lý ảnh số. Tuỳ thuộc vào quá trình xử lý tiếp theo trong giai đoạn này sẽ thực hiện các công đoạn khác nhau như: nâng cấp, khôi phục ảnh, nắn chỉnh hình học, khử nhiễu v.v..

b. Trích chọn đặc điểm

Các đặc điểm của đối tượng được trích chọn tuỳ theo mục đích nhận dạng trong quá trình xử lý ảnh. Trích chọn hiệu quả các đặc điểm giúp cho việc nhận dạng các đối tượng ảnh chính xác, với tốc độ tính toán cao và dung lượng nhớ lưu trữ giảm.

c. Đối sánh, nhận dạng

Nhận dạng tự động (automatic recognition), mô tả đối tượng, phân loại và phân nhóm các mẫu là những vấn đề quan trọng trong thị giác máy, được ứng dụng trong nhiều ngành khoa học khác nhau. Ví dụ mẫu có thể là ảnh của vân tay, ảnh của một vật nào đó được chụp, một chữ viết, khuôn mặt người hoặc một ký đồ tín hiệu tiếng nói. Khi biết một mẫu nào đó, để nhận dạng hoặc phân loại mẫu đó. Hệ thống nhận dạng tự động bao gồm ba khâu tương ứng với ba giai đoạn chủ yếu sau đây:

* Thu nhận dữ liệu và tiền xử lý.
* Biểu diễn dữ liệu.
* Nhận dạng, ra quyết định.

### 1.1.3. Các hình thái của ảnh

Đơn vị tế bào của ảnh số là pixel. Tùy theo mỗi định dạng là ảnh đen trắng, ảnh màu hay ảnh xám mà từng pixel có thông số khác nhau.

Đối với ảnh đen trắng, mỗi pixel được biểu diễn bởi 1 bit biểu thị màu đen hay màu trắng.

Đối với ảnh màu, từng pixel sẽ mang thông tin của ba màu cơ bản tạo ra bản màu khả kiến là Đỏ (R), Xanh lá (G) và Xanh biển (B). Trong mỗi pixel của ảnh màu, ba màu cơ bản R, G và B được bố trí sát nhau và có cường độ sáng khác nhau. Thông thường, mỗi màu cơ bản được biểu diễn bằng tám bit tương ứng 256 mức độ màu khác nhau. Như vậy mỗi pixel chúng ta sẽ có 256\*256\*256 màu (khoảng 16.78 triệu màu).

Đối với ảnh xám, thông thường mỗi pixel mang thông tin của 256 mức xám (tương ứng với tám bit) như vậy ảnh xám hoàn toàn có thể tái hiện đầy đủ cấu trúc của một ảnh màu tương ứng thông qua tám mặt phẳng bit theo độ xám.

### 1.1.4. Một số ứng dụng của xử lý ảnh

Các kỹ thuật xử lý ảnh trước đây chủ yếu được sử dụng để nâng cao chất lượng hình ảnh, chính xác hơn là tạo cảm giác về sự gia tăng chất lượng ảnh quang học trong mắt người quan sát. Thời gian gần đây, phạm vi ứng dụng xử lý ảnh mở rộng không ngừng, có thể nói hiện không có lĩnh vực khoa học nào không sử dụng các thành tựu của công nghệ xử lý ảnh số.

Trong y học các thuật toán xử lý ảnh cho phép biến đổi hình ảnh được tạo ra từ nguồn bức xạ X - ray hay nguồn bức xạ siêu âm thành hình ảnh quang học trên bề mặt film x - quang hoặc trực tiếp trên bề mặt màn hình hiển thị. Hình ảnh các cơ quan chức năng của con người sau đó có thể được xử lý tiếp để nâng cao độ tương phản, lọc, tách các thành phần cần thiết (chụp cắt lớp) hoặc tạo ra hình ảnh trong không gian ba chiều (siêu âm 3 chiều).

Trong lĩnh vực địa chất, hình ảnh nhận được từ vệ tinh có thể được phân tích để xác định cấu trúc bề mặt trái đất. Kỹ thuật làm nổi đường biên (image enhancement) và khôi phục hình ảnh (image restoration) cho phép nâng cao chất lượng ảnh vệ tinh và tạo ra các bản đồ địa hình 3-D với độ chính xác cao.

Trong ngành khí tượng học, ảnh nhận được từ hệ thống vệ tinh theo dõi thời tiết cũng được xử lý, nâng cao chất lượng và ghép hình để tạo ra ảnh bề mặt trái đất trên một vùng rộng lớn, qua đó có thể thực hiện việc dự báo thời tiết một cách chính xác hơn.

Xử lý ảnh còn được sử dụng rộng rãi trong lĩnh vực hình sự và các hệ thống bảo mật hoặc kiểm soát truy cập. Quá trình xử lý ảnh với mục đích nhận dạng vân tay hay khuôn mặt cho phép phát hiện nhanh các đối tương nghi vấn cũng như nâng cao hiệu quả hệ thống bảo mật cá nhân cũng như kiểm soát ra vào. Ngoài ra, có thể kể đến các ứng dụng quan trọng khác của kỹ thuật xử lý ảnh tĩnh cũng như ảnh động trong đời sống như tự động nhận dạng, nhận dạng mục tiêu quân sự, máy nhìn công nghiệp trong các hệ thống điều khiển tự động, nén ảnh tĩnh, ảnh động để lưu và truyền trong mạng viễn thông v.v.

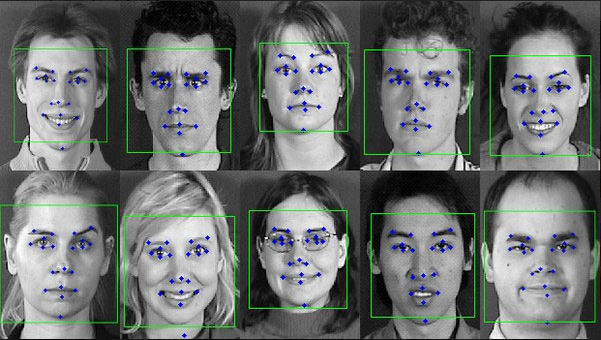
## 1.2. Bài toán phát hiện khuôn mặt

### 1.2.1. Một số đặc trưng về khuôn mặt

Chúng ta nhận diện gương mặt mọi người qua các đặc điểm, hệ thống nhận diện cũng vậy. Mỗi khuôn mặt đều có nhiều điểm mốc, những phần lồi lõm tạo nên các đặc điểm của khuôn mặt. Các hệ thống nhận diện gương mặt định nghĩa những điểm này là những điểm nút. Mỗi mặt người có khoảng 80 điểm nút. Có thể nhận diện một số điểm nút như sau:

* Khoảng cách giữa hai mắt
* Chiều rộng của mũi
* Độ sâu của hốc mắt
* Hình dạng của xương gò má
* Độ dài của xương hàm
* ...

Chuyên gia Michael Sheehan - chuyên gia thuộc Đại học California, Berkeley đã nghiên cứu cùng với các đồng nghiệp của mình đã tiến hành thu thập và phân tích dữ liệu hàng ngàn cá nhân khác nhau. Các chỉ số được tiến hành đo rất cụ thể và chi tiết: khoảng cách hai mắt, chiều cao mũi… cho tới chiều dài bắp chân.



Hình 1.1 Sự khác nhau của khuôn mặt người

Sau khi phân tích, họ nhận ra rằng khuôn mặt chính là bộ phận đặc biệt nhất. Nếu như đặc điểm ở chân, tay và các bộ phận khác hoàn toàn có thể dự đoán trước thì chúng ta không thể làm điều này với khuôn mặt. Đây là bộ phận duy nhất khác nhau hoàn toàn ở mỗi người, không ai giống ai và có thể coi là chứng minh thư tự nhiên của mỗi cá nhân [2].

### 1.2.2. Tìm hiểu về bài toán phát hiện khuôn mặt

Trong nhiều năm gần đây, có rất nhiều công trình nghiên cứu về bài toán xác định khuôn mặt người từ ảnh đen trắng, xám đến ảnh màu. Ban đầu chỉ là những bài toán đơn giản, mỗi ảnh chỉ có một khuôn mặt nhìn thẳng và đầu luôn phải ở tư thế thằng đứng trong ảnh đen trắng, không đáp ứng được nhu cầu ngày càng cao trong cuộc sống, khoa học ngày nay. Vì thế đã có những cải tiến nghiên cứu về bài toán phát hiện khuôn mặt người trong những môi trường phức tạp hơn, có nhiều khuôn mặt người trong ảnh hơn.

Về cơ bản, một hệ thống nhận dạng mặt người làm việc giống thị giác của con người khi cần nhận dạng (nhận ra) một ai đó khi nhìn vào một bức ảnh. Ở bước đầu tiên hệ thống cần xác định xem có mặt người trong bức ảnh đầu vào hay không, sau đó nếu có thì chỉ vùng ảnh đó sẽ được quan tâm, xử lý, bước này gọi là phát hiện khuôn mặt (face detection). Tiếp đến, các đặc điểm là đặc trưng nhất của khuôn mặt (giúp phân biệt ảnh mặt người này với người khác và các các ảnh thuộc về cùng một người) sẽ được trích chọn (extract) để nhận dạng, bước này gọi là trích chọn đặc trưng (feature extraction). Trong bước tiếp theo, gọi là nhận dạng (recognition), hệ thống sẽ đối sánh các đặc trưng vừa nhận được với một kho dữ liệu các đặc trưng của những người đã biết rõ danh tính để xác định xem các đặc trưng đó là của khuôn mặt nào và đưa ra kết luận về danh tính tương ứng với khuôn mặt có tỉ lệ tương đồng lớn nhất. Ở phần này chúng ta sẽ tìm hiểu về quá trình phát hiện khuôn mặt (face detection).

Phát hiện mặt người là một kỹ thuật để phát hiện vị trị và kích thước khuôn mặt người trong các ảnh bất kỳ. Kỹ thuật này chỉ nhận biết về các đặc trưng của khuôn mặt và bỏ qua những đặc trưng khác.

### 1.2.3. Các phương pháp chính phát hiện mặt người

Dựa vào tính chất của các phương pháp xác định mặt người trên ảnh, các phương pháp này được chia thành bốn loại chính, tương ứng với bốn hướng tiếp cận khác nhau. Ngoài ra cũng có rất nhiều nghiên cứu mà phương pháp xác định mặt người không chỉ dựa vào một hướng mà có liên quan đến nhiều hướng.

- Hướng tiếp cận dựa trên tri thức: Dựa vào các thuật toán, mã hoá các đặc trưng và quan hệ giữa các đặc trưng của khuôn mặt thành các luật. Đây là hướng tiếp cận theo kiểu top-down.

- Hướng tiếp cận dựa trên đặc trưng không thay đổi: Xây dựng các thuật toán để tìm các đặc trưng của khuôn mặt mà các đặc trưng này không thay đổi khi tư thế khuôn mặt hay vị trí đặt camera thay đổi.

- Hướng tiếp cận dựa trên so sánh khớp mẫu: Dùng các mẫu chuẩn của khuôn mặt (các mẫu này đã được chọn và lưu trữ) để mô tả các khuôn mặt hay các đặc trưng của khuôn mặt. Phương pháp này có thể dùng để xác định vị trí hay dò tìm khuôn mặt trên ảnh.

- Hướng tiếp cận dựa trên diện mạo: Trái với hướng tiếp cận dựa trên khuôn mẫu, các mô hình hay các mẫu sẽ được học từ một tập ảnh huấn luyện mà thể hiện tính chất tiêu biểu của sự xuất hiện của mặt người trong ảnh. Sau đó hệ thống sẽ xác định mặt người. Phương pháp này còn được biết đến với tên gọi tiếp cận theo hướng học máy.

### 1.2.4. Những khó khăn trong việc phát hiện khuôn mặt người

- Các khuôn mặt trong ảnh có thể có có những hướng nhìn khác nhau như: nhìn thẳng, nhìn nghiêng hay nhìn lên nhìn xuống. Cùng trong một ảnh nhưng sẽ có nhiều hướng nhìn khác nhau của khuôn mặt vì vậy việc phát hiện cũng sẽ gặp khó khăn.

- Trong một khuôn mặt không chỉ có những đặc trưng là khuôn mặt mà còn có một số chi tiết không phải là đặc trưng của khuôn mặt nên việc phát hiện khuôn mặt cũng có thể bị sai.

- Một số khuôn mặt bị che khuất bởi các đối tượng khác cũng gây cản trở cho việc phát hiện khuôn mặt.

- Sự biểu cảm của khuôn mặt có thể làm thay đổi đáng kể các đặc trưng và thông số của khuôn mặt. Như cùng một người nhưng khi cười, tức giận hay sợ hãi cũng dẫn đến sự khác biệt của khuôn mặt.

- Tập các ảnh khuôn mặt trong huấn luyện không thể bao quát được tất cả các biến đổi có thể có trên khuôn mặt của một người trong thế giới thực.

## 1.3. Tìm hiểu về phương pháp phát hiện khuôn mặt của Viola và Johns

### 1.3.1. Tổng quan về phương pháp

Phát hiện mặt người là bài toán cơ bản được xây dựng từ nhiều năm nay, có nhiều phương pháp được đưa ra như sử dụng template matching, neuron network… Cho tới nay bài toán này hầu như được giải quyết dựa trên phương pháp sử dụng trình phân loại Haar Cascades của Viola và Johns được cài đặt trong thư viện OpenCv. Phương pháp này được cho là đơn giản và kết quả phát hiện là tương đối cao, lên tới 95%. Các hãng sản xuất máy ảnh như Canon, Samsung… cũng đã tích hợp nó vào trong các sản phẩm của mình.

Trước năm 2000 các kỹ thuật để phát hiện khuôn mặt vẫn chưa đủ tin cậy, chậm và yêu cầu đầu vào thủ công. Đến năm 2001, Viola và Jones đã phát minh ra trình phân loại Haar Cascades, tạo nên sự cách mạng hoá trong phương pháp phát hiện khuôn mặt, nó có thể phát hiện các đối tượng ngay trong thời gian thực với độ chính xác 95% sau khi được Lienhart và Maydt cải thiện hơn năm 2002 [3].

Đây là một cách tiếp cận dựa trên máy học, chức năng xếp tầng được đào tạo từ rất nhiều hình ảnh tích cực và tiêu cực. Sau đó, nó được sử dụng để phát hiện các đối tượng trong các hình ảnh khác [4]. Khi training, chương trình sẽ rút ra đặc trưng từ các ảnh có mặt người (tích cực) rồi so sánh với các ảnh không có mặt người (tiêu cực), nếu đặc trưng đó được tìm thấy trong ảnh tiêu cực thì đặc trưng đó không phù hợp.

Về tổng quan bài toán phát hiện khuôn mặt cũng chính là bài toán phân loại 2 lớp mặt và không phải mặt. Trình phân loại Haar Cascades là bộ phân loại tầng với mỗi tầng là một bộ phân loại Adaboost. Ở các tầng sẽ sử dụng các đặc trưng Haar-like đặt vào các vùng ảnh để tính toán các giá trị đặc trưng, từ những giá trị đặc trưng này đưa vào bộ phân loại Adaboost sẽ xác định ảnh có khuôn mặt hay không.

### 1.3.2. Tiền xử lý ảnh

Phương pháp Haar Cascades được thực hiện trên ảnh xám (gray image). Mỗi điểm ảnh (pixel) sẽ có giá trị mức xám từ 0 đến 255 (không gian màu 8 bit). Như vậy phương pháp Haar Cascades sẽ không khai thác những đặc điểm về màu sắc khuôn mặt để nhận dạng song vẫn rất hiệu quả. Ảnh màu sẽ được chuyển về ảnh xám để nhận dạng.

Sau khi được chuyển thành ảnh xám, ảnh lại được tiếp tục chuyển thành “ảnh tích hợp” và các đặc trưng Haar-like sẽ làm việc trực tiếp trên ảnh tích hợp.

### 1.3.3. Đặc trưng Haar-like

a) Đối tượng nhận dạng

Trên ảnh, vùng khuôn mặt là tập hợp các điểm ảnh có những mối quan hệ khác biệt so với các vùng ảnh khác, những mối quan hệ này tạo nên các đặc trưng riêng của khuôn mặt. Tất cả khuôn mặt người đều có chung những đặc điểm sau khi đã chuyển qua ảnh xám, ví dụ như:

- Vùng hai mắt sẽ tối hơn vùng má và vùng trán, tức màu xám của vùng này cao hơn so với hai vùng còn lại.

- Vùng sống mũi sẽ sáng hơn vùng mắt.

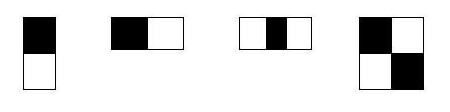
- …

Và còn rất nhiều những đặc điểm khác của khuôn mặt và các đặc trưng Haar-like dựa vào các đặc điểm này để nhận dạng.

Về tổng quát, các đặc trưng Haar-like không chỉ được sử dụng để nhận dạng khuôn mặt mà có thể dùng để nhận dạng bất kì một đối tượng nào trên ảnh (thân người, tay, chân, ô tô, đồ vật, ….). Bởi vì cũng giống như khuôn mặt, mỗi đối tượng có những đặc điểm riêng biệt đặc trưng bởi các vùng điểm ảnh, công việc của đặc trưng Haar-like là tính toán các giá trị tương quan giữa các vùng ảnh đó.

b) Các đặc trưng Haar-like

Đặc trưng Haar-like do Viola và Jones công bố gồm 4 đặc trưng cơ bản để xác định khuôn mặt người [5]. Mỗi đặc trưng Haar-like là một miền hình chữ nhật được chia thành 2,3 hoặc 4 hình chữ nhật nhỏ phân biệt quy ước bằng màu trắng và đen:



Hình 1.2 Bốn đặc trưng Haar-like

Để sử dụng các đặc trưng này vào việc xác định khuôn mặt người, 4 đặc trưng Haar-Like cơ bản được mở rộng ra và được chia làm 3 tập đặc trưng như sau:

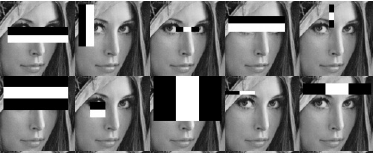
* Đặc trưng cạnh (edge feature) https://images.viblo.asia/47709099-37f8-48f2-be1b-4a5dc6a41c5b.png
* Đặc trưng đường (line feature) 
* Đặc trưng xung quanh tâm (center-surround features) https://images.viblo.asia/988200ee-7f89-4d58-bdb3-52479e62dc35.png

Giá giá trị của một đặc trưng Haar-Like là sự chênh lệch giữa tổng các giá trị xám của các pixel trong vùng đen với tổng các giá trị xám của các pixel trong vùng trắng như trong công thức sau:

Vậy khi được đặt lên một vùng ảnh, đặc trưng Haar-like sẽ tính toán và đưa ra giá trị đặc trưng f(x) của vùng đó.

### 1.3.4. Cách áp dụng đặc trưng Harr-like để phát hiện khuôn mặt trong ảnh

Để phát hiện khuôn mặt, hệ thống sẽ cho một cửa sổ con (sub-window) quét lên toàn bộ ảnh đầu vào. Sau mỗi lần quét hết toàn bộ ảnh đầu vào thì cửa sổ con sẽ tăng kích thước lên. Như vậy sẽ có rất nhiều ảnh con ứng với từng cửa sổ con, các đặc trưng Haar-like sẽ được đặt lên các cửa sổ con này từ đó tính ra các giá trị đặc trưng. Sau đó các giá trị này được bộ phân loại xác nhận xem khung hình đó có phải khuôn mặt hay không.



Hình 1.3 Các đặc trưng Haar-like được đặt lên tại khung hình

Như hình vẽ trên đây, tại mỗi khung hình các đặc trưng Haar-like được đặt lên ở các vị trí khác nhau và sau đó tính các giá trị đặc trưng. Ứng với mỗi đặc trưng trên, một bộ phân loại yếu (weak classifier) được định nghĩa như sau:

=

Trong đó:

- x: Cửa sổ con đang xét.

- : Giá trị đặc trưng Haar-like thứ k.

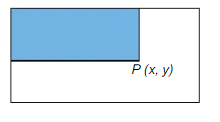
- : Ngưỡng (là giá trị đã được rút ra sau quá trình huấn luyện).

Ta hiểu công thức trên đơn giản như sau: Đặc trưng Haar-like có k vị trí đặt trong cửa sổ con x, khi giá trị của đặc trưng Haar-like tại vị trí thứ k vượt qua một ngưỡng thì bộ phân lớp sẽ kết luật cửa sổ con x là khuôn mặt, còn không qua ngưỡng đó thì không là khuôn mặt.

Sẽ có những vị trí trên ảnh không phải là khuôn mặt nhưng lại cho ra giá trị đặc trưng vượt ngưỡng và bộ phân lớp yếu sẽ kết luận đấy là khuôn mặt, nhưng ta không chỉ dùng một đặc trưng Haar-like mà dùng rất nhiều đặc trưng ở những vị trí và kích thước khác nhau nên sẽ có được những kết luận chính xác.

### 1.3.5. Ảnh tích hợp (Integral image)

Để tính các giá trị của đặc trưng Haar-like, ta phải tính tổng của các vùng pixel trên ảnh. Mà số lượng các đặc trưng Haar-like rất nhiều nên để tính toán đòi hỏi khối lượng tính toán rất lớn. Vì vậy ảnh tích hợp được đưa ra để nhằm tính toán nhanh chóng các đặc trưng, giảm thời gian xử lý. Giá trị của ảnh tích hợp tại ví trí (x, y) là tổng các điểm ảnh thuộc hình chữ nhật xác định bởi góc trên bên trái (0,0) và góc phải dưới (x,y).



Hình 1.4 Ảnh tích hợp

Ảnh tích hợp được định nghĩa theo công thức:

P(x,y) =

Trên thực tế để chuyển một ảnh thành ảnh tích hợp ta dùng công thức truy hồi sau:

s(x, y) = x(x, y-1) + i(x, y) với s(x, -1) = 0

P(x, y) = P(x-1, y) + s(x, y) P(-1, y) = 0

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 2 | 4 | 6 |
| 3 | 6 | 9 |

Ví dụ chuyển một ảnh 3x3 có giá trị xám như bên dưới thành ảnh tích hợp

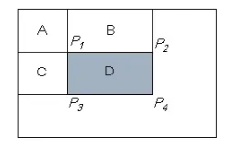
|  |  |  |
| --- | --- | --- |
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

==>

Ảnh ban đầu Ảnh tích hợp

Sau khi chuyển ảnh cần nhận dạng thành ảnh tích hợp, việc tính toán giá trị các đặc trưng Haar-like sẽ rất đơn giản.

Để tính giá trị đặc trưng Haar-like, ta phải tính được tổng giá trị điểm ảnh trong một vùng chữ nhật trên ảnh. Ví dụ như vùng D trong hình vẽ :



Hình 1.5 Vùng giá trị điểm ảnh cần tính

Với A, B, C, D: là tổng các giá trị điểm ảnh trong từng vùng.

P1, P2, P3, P4: là giá trị của ảnh tích hớp tại 4 đỉnh của D.

Nếu như là ảnh xám bình thường thì để tính D ta phải tính tổng tất cả các giá trị điểm ảnh trong D, miền D càng lớn thì số phép cộng càng nhiều. Nhưng với ảnh tích hợp dù miền D có kích thước như thế nào thì D cũng chỉ cần tính thông qua 4 giá trị tại 4 đỉnh.

Ta có: P1 = A, P2 = A+B, P3 = A+C, P4 = A+B+C+D

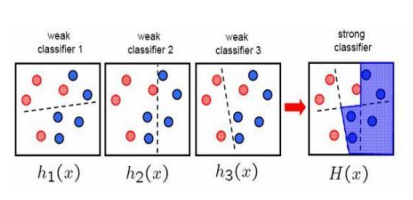
Suy ra :

D = (A+B+C+D) – (A+B) – (A+C) + A

= P4 – P2 – P3 + P1

Để chọn các đăc trưng Haar-like dùng cho việc thiết lập ngưỡng, Viola và Jones sử dụng một phương pháp máy học được gọi là Adaboost [6]. Adaboost sẽ kết hợp các bộ phân loại yếu để tạo thành một bộ phân loại mạnh.

Adaboost sử dụng thêm khái niệm trọng số (weight) để đánh dấu các mẫu khó nhận dạng. Trong quá trình huấn luyện, cứ mỗi bộ phân loại yếu (weak classifiers) được xây dựng, thuật toán sẽ tiến hành cập nhật lại trọng số để chuẩn bị cho việc xây dựng weak classifier kế tiếp: tăng trọng số của các mẫu bị nhận dạng sai và giảm trọng số của các mẫu được nhận dạng đúng bởi các weak classifier vừa xây dựng. Bằng cách này weak classifier sau có thể tập trung vào các mẫu mà các weak classifier trước nó làm chưa tốt. Sau cùng, các weak classifier sẽ được kết hợp tùy theo mức độ tốt của chúng để tạo nên strong classifier.



Hình 1.6 Kết hợp các bộ phân loại yếu thành bộ phân loại mạnh

### 1.3.6. Mô hình phân loại theo tầng Cascade of classifier

Giả sử chúng ta huấn luyện được một bộ phân lớp mạnh để phát hiện khuôn mặt thông qua 10 bộ phân lớp yếu. Nếu ta dùng bộ phân lớp mạnh này đưa vào các cửa sổ con (sub window) chạy khắp một tấm ảnh đưa vào để phát hiện khuôn mặt thì sẽ rất tốn chi phí tính toán và thời gian. Tại mỗi cửa sổ con trên tấm ảnh chúng ta lại phải dùng cả 10 bộ phân lớp yếu. Trong khi đó, ta thấy trong một tấm ảnh thì khuôn mặt chiếm tỉ lệ ít và tại những cửa sổ không phải là khuôn mặt ta có thể loại bỏ mà chỉ cần dùng một bộ phân lớp mạnh khác gồm ít bộ phân lớp yếu hơn.

Viola và Jones giải quyết vấn đề này bằng chuỗi các bộ phân lớp (Cascade of classifier). Ta sẽ có một chuỗi các bộ phân lớp, trong đó mỗi bộ phân lớp được xây dựng bằng thuật toán Adaboost.

Sub window

Không là khuôn mặt

Là khuôn mặt

Không là khuôn mặt

Không là khuôn mặt

Là khuôn mặt

Là khuôn mặt

Hình 1.7 Mô hinh Cascade of classifier

Mỗi cửa sổ con sẽ được cho đi qua các bộ phân lớp này:

- Bộ phân lớp đầu tiên sẽ loại bỏ phần lớn các cửa sổ không phải khuôn mặt và cho đi qua các cửa sổ được cho là khuôn mặt. Ở đây, bộ phân lớp này rất đơn giản nên độ phức tạp cũng rất thấp. Tất nhiên, vì nó đơn giản nên trong số các cửa sổ được nhận dạng là khuôn mặt sẽ có một số lượng lớn cửa sổ bị nhận dạng sai

- Những cửa sổ được cho đi qua bởi bộ phân lớp đầu sẽ được xem xét bởi bộ phân lớp sau đó. Nếu bộ phân lớp cho rằng đó không phải là khuôn mặt thì loại bỏ cửa sổ, nếu bộ phân lớp cho rằng đó là khuôn mặt thì ta lại cho đi qua và chuyển đến bộ phân lớp phía sau.

- Những bộ phân lớp càng về sau thì càng phức tạp hơn, đòi hỏi sự tính toán nhiều hơn. Cửa sổ nào không thoả mãn bộ phân lớp sẽ bị loại bỏ. Chỉ những cửa sổ đi qua được tất cả các bộ phân lớp thì ta mới quyết định đó là khuôn mặt.

Tóm lại, chuỗi các bộ phân lớp sẽ xử lý các mẫu (cửa sổ con) đi vào theo nguyên tắc sau: nếu một bộ phân lớp nào đó cho rằng đó không phải là mặt người thì ta loại bỏ ngay, còn nếu bộ phân lớp cho rằng đó là khuôn mặt thì ta chuyển tiếp đến bộ phân lớp sau. Nếu một mẫu trót lọt hết tất cả các bộ phân lớp thì ta mới quyết định đó là khuôn mặt.

CHƯƠNG 2. BÀI TOÁN NHẬN DẠNG MẶT NGƯỜI

## 2.1. Tổng quan về bài toán nhận dạng mặt người

### 2.1.1. Giới thiệu

Nhận dạng mặt người (face recognition) là một chủ đề nghiên cứu thuộc lĩnh vực thị giác máy tính (computer vision) đã được phát triển từ đầu những năm 90 của thế kỷ trước. Cho tới hiện nay, đây vẫn là một chủ đề nghiên cứu mở nhận được sự quan tâm của nhiều nhà nghiên cứu từ nhiều lĩnh vực nghiên cứu khác nhau như nhận dạng mẫu (Pattern Recognition), học máy (Machine Learning), thống kê (Statistics), sinh trắc học (Biometrics). Điều này là do có rất nhiều ứng dụng thực tế cần tới một hệ thống nhận dạng mặt, từ các hệ thống quản lý đăng nhập đơn giản cho tới các ứng dụng giám sát tại các địa điểm công cộng hoặc quản lý dân số và pháp lý. Bên cạnh đó, so với các hệ thống nhận dạng dựa trên các đặc điểm sinh trắc học khác của con người, như nhận dạng mống mắt và vân tay, dáng đi, nhận dạng khuôn mặt có nhiều ưu điểm:

+ Một hệ thống nhận dạng khuôn mặt không đòi hỏi có sự tương tác trực tiếp giữa đối tượng được nhận dạng và hệ thống.

+ Việc thu nhận dữ liệu (ảnh mặt) cho quá trình nhận dạng một con người dễ thực hiện hơn so với thu nhận các đặc điểm sinh trắc học khác (như thu nhận dấu vân tay và mống mắt).

+ Dữ liệu về mặt người phổ biến hơn so với các đặc điểm đặc trưng khác do sự bùng nổ của các mạng xã hội (facebook, twitter,…), các dịch vụ chia sẻ dữ liệu đa phương tiện (youtube, vimeo,…) và sự phát triển mạnh mẽ của các thiết bị thu nhận hình ảnh.

+ Từ ảnh khuôn mặt của một người ta có thể khai thác nhiều thông tin liên quan chứ không chỉ là danh tính, chẳng hạn như giới tính (gender), màu da (skin color), hướng nhìn (gaze direction),…

### 2.1.2. Thách thức trong việc nhận dạng mặt người

Việc xây dựng một hệ thống nhận dạng mặt người hoàn toàn tự động với khả năng nhận dạng chính xác cao thực sự là một thách thức đối với các nhà nghiên cứu. Điều này là do các yếu tố (chủ quan và khách quan) ảnh hưởng tới quá trình thu nhận ảnh và tạo ra các bức ảnh có độ khác biệt rất lớn của cùng một khuôn mặt. Có thể liệt kê ra một số yêu tố chủ yếu ảnh hưởng tới độ chính xác của một hệ thống nhận dạng khuôn mặt:

+ Tư thế góc chụp: Ảnh chụp khuôn mặt có thể thay đổi rất nhiều vì góc chụp giữa camera và khuôn mặt. Chẳng hạn như: chụp thẳng, chụp chéo, chụp từ trên xuống, từ dưới lên,… Với các tư thế khác nhau, các thành phần trên khuôn mặt như mắt, mũi, miệng có thể bị khuất một phần hoặc thậm chí khuất hết.

+ Sự xuất hiện hoặc thiếu một số thành phần của khuôn mặt: Các đặc trưng như râu mép, râu cằm, mắt kính, v.v… có thể xuất hiện hoặc không. Vấn đề này làm cho bài toán khó khăn hơn trong việc nhận dạng.

+ Sự biểu cảm của khuôn mặt: Biểu cảm của khuôn mặt người có thể làm ảnh hưởng đáng kể lên các thông số của khuôn mặt. Chẳng hạn, cùng một khuôn mặt của một người, nhưng có thể sẽ rất khác khi họ cười hoặc sợ hãi.

+ Sự che khuất: Khuôn mặt có thể bị che khuất bởi các đối tượng khác hoặc các khuôn mặt khác.

+ Điều kiện của ảnh: Ảnh được chụp trong các điều kiện khác nhau về chiếu sáng, tính chất camera (máy kỹ thuật số, máy hồng ngoại, v.v…) ảnh hướng rất nhiều đến chất lượng khuôn mặt.

+ Thay đổi về tuổi: Khuôn mặt người có các thay đổi lớn khi tuổi thay đổi và khó nhận dạng hơn ngay cả đối với hệ thống thị giác của con người.

### 2.1 3. Ứng dụng của nhận diện khuôn mặt

Bài toán nhận diện khuôn mặt là một bài toán phức tạp nhưng lại hấp dẫn rất nhiều người nghiên cứu về nó, có lẽ bởi tính ứng dụng thực tế của bài toán là rất lớn và đa dạng trong cuộc sống hiện đại. Có thể kể ra một số ứng dụng thông dụng nhất của bài toán nhận diện khuôn mặt người:

+ Hệ thống giao tiếp thông minh giữa người và máy: Con người có thể xây dựng những hệ thống giao tiếp giữa người và máy tính thông qua việc nhận diện khuôn mặt, biểu cảm trên khuôn mặt người để dự đoán, nhận biết trạng thái tâm lý hiện thời của người đó.

+ Nhận dạng tội phạm: Hệ thống có thể nhận diện ra một khuôn mặt ngay tức thì và đối chiếu với hàng triệu bản ghi có sẵn trong cơ sở dữ liệu để chỉ ra đó có thể là một tội phạm đang truy tìm hay không, hoặc đó có thể là một nhân vật đặc biệt nào đó cần quan tâm, v.v…

+ Giải trí: Trong hầu hết các máy ảnh hiện đại ngày nay đều có chức năng tự động nhận diện mặt người để có thể lấy độ nét, điều chỉnh ánh sáng cho phù hợp với khung cảnh xung quanh.

+ Hệ thống giám sát, theo dõi và bảo vệ: Các hệ thống camera sẽ xác định được đâu là người và theo dõi tập trung vào người đó,…

+ Bảo mật, xác nhận: Cho phép các nhân viên ra vào các khu vực mà không cần phải đăng nhập hay dùng thẻ,…

## 2.2. Tiếp cận bài toán nhận diện khuôn mặt

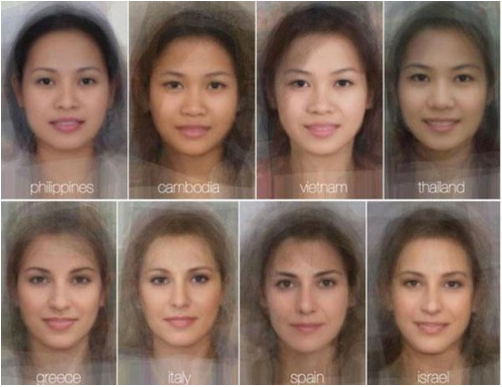
### 2.2.1. Một số bài toán về khuôn mặt

Có nhiều lớp bài toán khác nhau liên quan đến dữ liệu khuôn mặt. Bốn bài toán phổ biến nhất dựa trên nhu cầu thực tế cần áp dụng là:

- Nhận diện khuôn mặt (Face recognition): Đây là bài toán match one – many. Bài toán này sẽ trả lời cho câu hỏi người này là ai bằng cách nhận input là ảnh khuôn mặt và output là nhãn tên người trong ảnh. Tác vụ này thường được áp dụng trong các hệ thống chấm công, hệ thống giám sát, hệ thống camera thông minh.

- Xác thực khuôn mặt (Face verification): Đây là bài toán match one – one. Bài toán này trả lời cho câu hỏi có phải 2 ảnh đầu vào là cùng một người hay không? Kết quả output sẽ là yes hoặc no. Bài toán thường được dùng trong các hệ thống bảo mật. Xác thực khuôn mặt trên điện thoại trên điện thoại cũng là một bài toán như vậy.

- Tìm kiếm khuôn mặt đại diện (Face clustering): Chắc hẳn mọi người đã từng đọc hoặc xem video về người có khuôn mặt đặc trưng nhất thế giới. Đơn giản là ta chỉ cần tính ra trung bình của các ảnh khuôn mặt để thu được hình ảnh trung tâm (centroid image). Từ đó ta có thể tìm ra khuôn mặt đặc trưng nhất của nam, nữ các quốc gia.



Hình 2. 1 Khuôn mặt đại diện nữ của một số quốc gia

- Tìm kiếm khuôn mặt tương đương (Face similarity): Gần đây facebook hay có các ứng dụng vui như bạn giống với diễn viên điện ảnh nào nhất. Bài toán này sẽ tính toán xem khuôn mặt của bạn gần giống nhất với khuôn mặt của các diễn viên có sẵn trong cơ sở dữ liệu.

### 2.2.2. Mô hình cơ bản của bài toán nhận diện khuôn mặt

Một hệ thống nhận diện khuôn mặt người có thể được xây dựng qua rất nhiều công đoạn khác nhau và rất phức tạp, nhưng ta có thể khái quát chung mọi hệ thống nhận dạng khuôn mặt người gồm có 3 bước cơ bản sau:



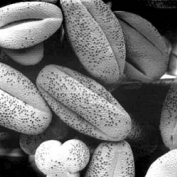
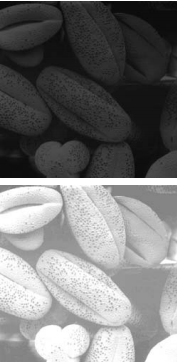
Hình 2. 2 Mô hình hệ thống nhận diện khuôn mặt

- Tiền xử lý: Sau khi đã xác định được vị trí có khuôn mặt trong hình ảnh đầu vào, chúng ta sẽ tiến hành một số bước tiền xử lý ảnh sau đây trước khi đưa vào trích xuất các đặc trưng của khuôn mặt:

+ Crop ảnh, resize ảnh: Chúng ta sẽ cắt khuôn mặt đã được phát hiện ra vào chỉnh sửa lại kích thước của khuôn mặt cho đồng nhất và phù hợp với kích thước đầu vào của bộ trích xuất đặc trưng.

+ Biến đổi hình ảnh sang dạng ảnh khác: Ảnh dùng để xử lý trong bài toán nhận dạng khuôn mặt là ảnh xám để loại bỏ các yếu tố về ảnh hưởng của màu sắc nên ta sẽ chuyển ảnh cần xử lý sang ảnh xám.

+ Cân bằng sáng: Các giải thuật xử lý ảnh thường nhạy cảm với ánh sáng, cùng nội dung ảnh nhưng với các điều kiện ánh sáng khác nhau có thể làm sai lệch kết quả xử lý (giả sử trong bài toán phát hiện đối tượng, bài toán nhận dạng, bài toán đếm đối tượng, …). Do đó, cân bằng sáng ở bước tiền xử lý là một trong những cách giúp làm giảm các ảnh hưởng này. Cân bằng sáng nhằm giảm sự ảnh hưởng do chiếu sáng (chói) và thiếu ánh sáng (ảnh tối).



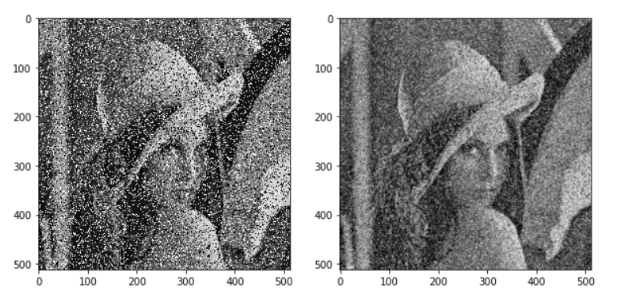
Hình 2. 3 Cân bằng sáng cho ảnh

+ Giảm nhiễu cho ảnh:



Hình 2. 4 Ảnh bị nhiễu

Hình bên trên là một ví dụ minh họa cho ảnh bị nhiễu. Một cách trực quan, bạn có thể định nghĩa điểm nhiễu là những điểm đốm trắng đen khiến cho hình khó nhìn. Nếu để ý kĩ hơn, bạn có thể thấy, các điểm nhiễu này có mức sáng (intensity) chênh lệch rõ ràng so với những điểm xung quanh (điểm trắng nổi lên giữa khu vực màu đen). Để loại bỏ những điểm nhiễu này, hay nói cách khác, loại bỏ những điểm có mức sáng chênh lệch lớn bất thường, Làm mờ ảnh (blurring) là một trong những phương pháp phổ biến nhất được áp dụng. Vậy làm thế nào để làm mờ ảnh ? Làm thế nào để làm mờ những điểm có độ sáng bất thường trên ảnh ? Biện pháp đơn giản nhất chính là thay thế những điểm có độ sáng bất thường bằng trung bình độ sáng của những điểm lân cận. Sau khi thực hiện blurring thì độ nhiễu của ảnh sẽ được giảm đáng kể.



Hình 2. 5 Xử lý nhiễu của ảnh

- Trích rút các đặc trưng: Trích rút đặc trưng là kỹ thuật sử dụng các thuật toán để lấy ra những thông tin mang những đặc điểm riêng biệt của một người.

- Nhận diện khuôn mặt: Sau khi đã trích rút các đặc trưng, sẽ đưa vào khối nhận dạng để phân lớp đối tượng.

### 2.2.3. Một số thuật toán nhận diên khuôn mặt

#### 2.2.3.1. One – shot learning

One – shot learning là thuật toán học có giám sát mà mỗi một người chỉ cần một vài, rất ít hoặc thậm chí chỉ một bức ảnh duy nhất. Từ đầu vào là bức ảnh của một người, chúng ta sử dụng một kiến trúc thuật toán CNN đơn giản để dự báo người đó là ai theo một phân phối xác suất.

Tuy nhiên, nhược điểm của phương pháp này là chúng ta phải huấn luyện lại thuật toán thường xuyên khi xuất hiện thêm một người vì lúc này số lượng output tăng lên một. Rõ ràng là không tốt đối với các hệ thống nhận diện khuôn mặt của các hệ thống mà số lượng thành viên thay đổi liên tục theo thời gian.

Để khắc phục được vấn đề này, chúng ta sử dụng phương pháp Learning similarity.

#### 2.2.3.2. Learning similarity

Phương pháp này dựa trên một phép đo khoảng cách giữa hai bức ảnh, nếu hai bức ảnh của cùng một người thì khoảng cách là nhỏ và nếu không thuộc về cùng một người thì khoảng cách sẽ lớn.

d = 0.3

d = 5

Hình 2. 6 Mô tả phương pháp Learning similarity

Phương pháp Learning similarity thay vì sự báo một phân phối xác suất để tìm ra nhãn phù hợp với ảnh đầu vào. Thuật toán sẽ so sánh khoảng cách giữa ảnh đầu vào với toàn bộ các ảnh có trong dữ liệu. Ta cần chọn một ngưỡng threshold để quyết định ảnh cùng một người hay hai người khác nhau. Giả sử ngưỡng threshold là 0.5. Trong sơ đồ trên thì ảnh input có khoảng cách với ảnh 1 là 0.3 nhỏ hơn 0.5, ảnh input có khoảng cách với ảnh 2 là 5. Do đó ảnh input với ảnh 1 là ảnh của cùng một người. Còn ảnh 2 và ảnh input là của hai người khác nhau.

Learning similarity có thể trả về nhiều hơn một ảnh là cùng loại với ảnh đầu vào tuỳ theo ngưỡng threshold. Ngoài ra phương pháp này không bị phụ thuộc vào số lớp. Do đó không cần phải huấn luyện lại khi xuất hiện lớp mới. Như vậy Learning similarity có ưu điểm hơn so với one – shot learning khi không phải huấn luyện lại model khi mà vẫn tìm ra được ảnh tương đồng.

Điểm mấu chốt là cần xây dựng được một model đủ tốt để chiếu các bức ảnh lên một không gian Euclide n chiều. Sau đó sử dụng khoảng cách để quyết định nhãn của chúng. Vậy làm thế nào để học được biểu diễn của ảnh trong không gian euclide n chiều ? Kiến trúc Siam network sẽ giúp chúng ta thực hiện điều này.

#### 2.2.3.3. Siam network

Những kiến trúc mạng mà khi bạn đưa vào hai bức ảnh và mô hình sẽ trả lời chúng có thuộc về cùng một người hay không được gọi chung là Siam network.

Kiến trúc của Siam network dựa trên một mạng CNN đã được loại bỏ output layer. Nó có tác dụng chuyển ảnh thành vector nhúng. Đầu vào của mạng Siam network là hai bức ảnh bất kì được lựa chọn ngẫu nhiên từ dữ liệu ảnh. Output của Siam network là hai vector tương ứng với biểu diễn của hai ảnh input. Sau đó chúng ta đưa hai vector vào hàm loss function để đo lường sự khác biệt giữa chúng.

Input 1

f(x1)

Input 2

f(x2)

Hình 2. 7 Mô hình Siam network

Từ mô hình CNN, mô hình trả về 2 vector là f(x1) và f(x2) biểu diễn cho ảnh input 1 và input 2 có cùng số chiều.

Khi input 1 và input 2 là cùng một người:

phải là một giá trị nhỏ.

Khi input 1 và input 2 không là cùng một người:

phải là một giá trị lớn.

## 2.3. Mô hình FaceNet trong nhận dạng khuôn mặt

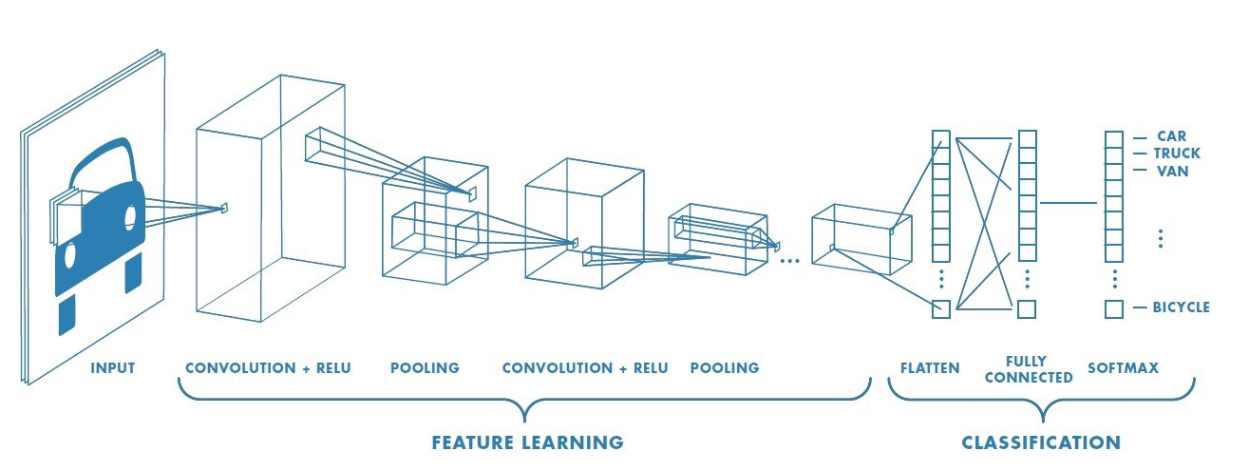
### 2.3.1. Mạng CNN – Convolutional Neural Network

Mạng nơron tích chập (CNN – Convolutional Neural Network) là một trong những mô hình mạng Học sâu phổ biến nhất hiện nay, có khả năng nhận dạng và phân loại hình ảnh với độ chính xác rất cao, thậm chí còn tốt hơn con người trong nhiều trường hợp. Mô hình này đã và đang được phát triển, ứng dụng vào các hệ thống xử lý ảnh lớn của Facebook, Google hay Amazon… cho các mục đích khác nhau như các thuật toán tagging tự động, tìm kiếm ảnh hoặc gợi ý sản phẩm cho người tiêu dùng.

Sự ra đời của mạng CNN dựa trên ý tưởng cải tiến cách thức các mạng nơrôn nhân tạo truyền thống học thông tin trong ảnh. Do sử dụng các liên kết đầy đủ giữa các điểm ảnh vào node, các mạng nơ-ron nhân tạo truyền thẳng bị hạn chế rất nhiều bởi kích thước của ảnh, ảnh càng lớn thì số lượng liên kết càng tăng nhanh và kéo theo sự bùng nổ khối lượng tính toán.

Ngoài ra sự liên kết đầy đủ này cũng là sự dư thừa khi với mỗi bức ảnh, các thông tin chủ yếu thể hiện qua sự phụ thuộc giữa các điểm ảnh với những điểm xung quanh nó mà không quan tâm nhiều đến các điểm ảnh ở cách xa nhau. Mạng CNN ra đời với kiến trúc thay đổi, có khả năng xây dựng liên kết chỉ sử dụng một phần cục bộ trong ảnh kết nối đến node trong lớp tiếp theo thay vì toàn bộ ảnh như trong mạng nơron truyền thẳng.

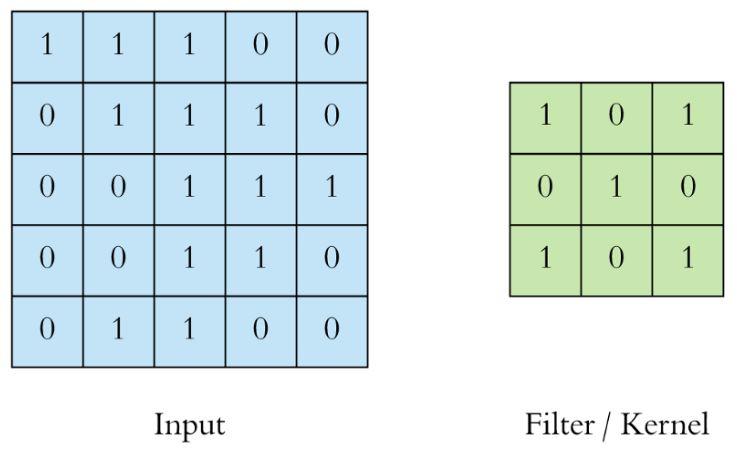
Các lớp cơ bản trong một mạng CNN bao gồm: Lớp tích chập (Convolutional Layer), Lớp kích hoạt phi tuyến, Lớp lấy mẫu (Pooling) và Lớp kết nối đầy đủ (Fully-connected), được thay đổi về số lượng và cách sắp xếp để tạo ra các mô hình huấn luyện phù hợp cho từng bài toán khác nhau [7].



Hình 2. 8 Mô hình mạng CNN

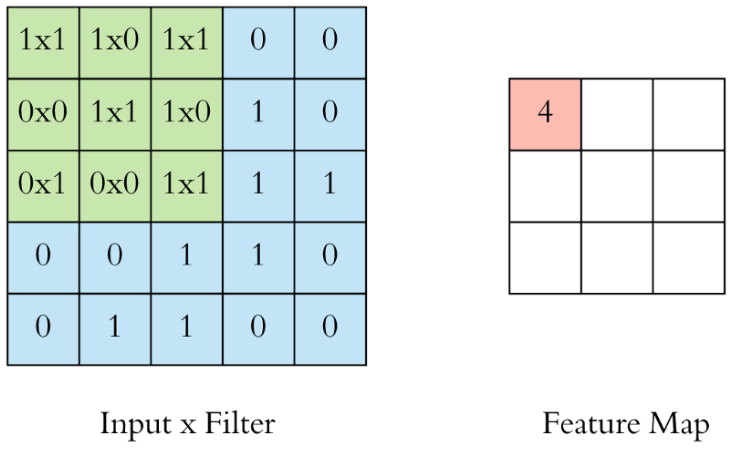
#### 2.3.1.1. Lớp tích chập

Lớp tích chập (Convolutional Layer) là một lớp quan trọng trong mạng CNN, nó giúp ta trích xuất được các đặc trưng của ảnh. Ta thực hiện tích chập bằng cách sử dụng kernel, đây là một ma trận sẽ quét qua ma trận dữ liệu đầu vào, từ trái qua phải, từ trên xuống dưới rồi nhân tương ứng từng giá trị của ma trận đầu vào với ma trận kernel rồi cộng lại, kết quả sẽ là một con số cụ thể, tập hợp các con số này sẽ là một ma trận gọi là feature map. Ví dụ ta có một ma trận đầu vào và một kernel như dưới đây.



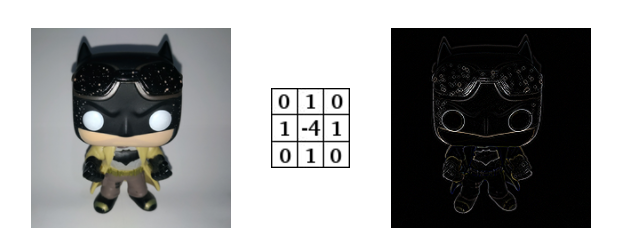
Hình 2. 9 Ma trận đầu vào và kernel

Ta sẽ quét kernel qua từng phần tử của input và tính toán như trên: Nhân tương ứng rồi cộng kết quả ta thu được một giá trị tại feature map.



Hình 2. 10 Tính toán feature map

Làm tương tự ta sẽ thu được một feature map hoàn chỉnh. Ví dụ với một hình đầu vào với một kernel như dưới ta thu được một feature map như sau.



Hình 2. 11 Kết quả quá trình Convolution

#### 2.3.1.2. Lớp kích hoạt phi tuyến

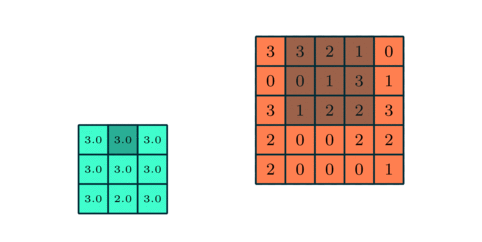
Lớp này được xây dựng với ý nghĩa đảm bảo tính phi tuyến của mô hình huấn luyện sau khi đã thực hiện một loạt các phép tính toán tuyến tính qua các lớp Tích chập. Lớp Kích hoạt phi tuyến nói chung sử dụng các hàm kích hoạt phi tuyến như ReLU hoặc sigmoid, tanh… để giới hạn phạm vi biên độ cho phép của giá trị đầu ra. Trong số các hàm kích hoạt này, hàm ReLU được chọn do cài đặt đơn giản, tốc độ xử lý nhanh mà vẫn đảm bảo được tính toán hiệu quả. Cụ thể, phép tính toán của hàm ReLU chỉ đơn giản là chuyển tất cả các giá trị âm thành giá trị 0.

Thông thường, lớp kích hoạt phi tuyến được áp dụng ngay phía sau lớp Tích chập, với đầu ra là một ảnh mới có kích thước giống với ảnh đầu vào, các giá trị điểm ảnh cũng hoàn toàn tương tự trừ các giá trị âm đã bị loại bỏ.

#### 2.3.1.3. Lớp lấy mẫu (Pooling)

Một thành phần tính toán chính khác trong mạng CNN là lấy mẫu (Pooling), thường được đặt sau lớp Tích chập và lớp kích hoạt phi tuyến để làm giảm kích thước ảnh đầu ra trong khi vẫn giữ được các thông tin quan trọng của ảnh đầu vào. Việc giảm kích thước dữ liệu có tác dụng làm giảm được số lượng tham số cũng như tăng hiệu quả tính toán. Lớp lấy mẫu cũng sử dụng một cửa sổ trượt để quét toàn bộ các vùng trong ảnh tương tự như lớp Tích chập, và thực hiện phép lấy mẫu thay vì phép tích chập – tức là ta sẽ chọn lưu lại một giá trị duy nhất đại diện cho toàn bộ thông tin của vùng ảnh đó. Có 2 loại pooling layer phổ biến là: max pooling và average pooling.

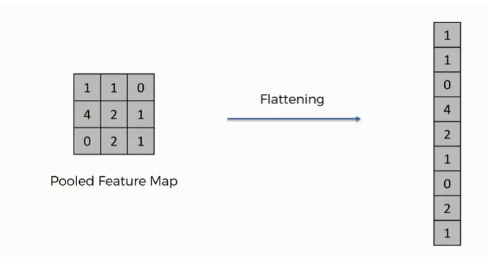
Loại pooling ta thường gặp nhất là max pooling, lấy giá trị lớn nhất trong một pooling window. Nó sử dụng một cửa sổ trượt qua từng giá trị của ma trận đầu vào (thường là các feature map trong convolutional layer), chọn ra một giá trị từ các gía trị nằm trong cửa sổ trượt (với max pooling ta sẽ lấy giá trị lớn nhất).



Hình 2. 12 Max Pooling

#### 2.3.1.4. Lớp kết nối đầy đủ (Fully-connected)

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer, thường sẽ tạo ra 2 layer Fully connected, 1 layer để tập hợp các feature layer mà ta đã tìm ra, chuyển đổi dữ liệu từ 3D, hoặc 2D thành 1D, tức chỉ còn là 1 vector. Còn 1 layer nữa là output, số neuron của layer này phụ thuộc vào số output mà ta muốn tìm ra.



Hình 2. 13 Layer Fully connected

### 2.3.2. Tìm hiểu mạng FaceNet

#### 2.3.2.1. Giới thiệu mạng FaceNet

Bước quan trọng nhất trong bài toán nhận dạng khuôn mặt chính là bước trích chọn các đặc trưng của khuôn mặt. Phương pháp trích chọn đặc trưng ở đây tôi đề xuất là mô hình mạng FaceNet. Được đề xuất bởi nhóm tác giả làm việc tại Google đó là Florian Schroff, Dmitry Kalenichenko và James Philbin năm 2015 [8].

FaceNet dựa trên việc nhúng mỗi ảnh vào không gian Euclide bằng cách sử dụng mạng CNN. Mạng được huấn luyện sao cho khoảng cách bình phương L2 trong không gian nhúng tương ứng với khoảng cách giữa các khuôn mặt: Khuôn mặt của cùng một người có khoảng cách nhỏ và khuôn mặt của những người khác nhau có khoảng cách lớn. Khi đó, mỗi khuôn mặt được đại diện bởi một vector đặc trưng 128 chiều được chuyển đổi thành 128 byte.

#### 2.3.2.2. Đặc trưng của mạng FaceNet

Hầu hết các thuật toán nhận diện khuôn mặt trước FaceNet đều tìm cách biểu diễn khuôn mặt bằng một vector nhúng để biểu diễn đặc điểm

+ Tuy nhiên hạn chế của các thuật toán đó là số lượng chiều của vector nhúng tương đối lớn (thường >= 1000) và ảnh hưởng tới hiệu suất của thuật toán. Thường chúng ta phải áp dụng thêm các thuật toán để giảm chiều dữ liệu.

+ Hàm chi phí (loss function) chỉ đo lường khoảng cách giữa hai bức ảnh. Như vậy trong một đầu vào huấn luyện chỉ học được một trong hai khả năng là sự giống nhau nếu chúng cùng 1 class hoặc sự khác nhau nếu chúng khác class giữa hai bức ảnh.

FaceNet đã giải quyết cả hai vấn đề trên bằng các hiệu chỉnh nhỏ nhưng mang lại hiệu quả rất lớn.

+ FaceNet sử dụng một mạng CNN và giảm chiều dữ liệu xuống chỉ còn 128 chiều. Do đó thuật toán phân loại ở những layer sau hoạt động nhanh hơn và đồng thời độ chính xác vẫn được đảm bảo.

+ Sử dụng một hàm chi phí (loss function) đặc biệt có khả năng đánh mức độ khác biệt giữa các bức ảnh sao cho giá trị loss function của chúng càng lớn, sự khác biệt giữa chúng càng cao. Đó chính là Triplet loss function. Hàm Triplet loss có khả năng học được đồng thời sự giống nhau giữa hai bức ảnh cùng nhóm và tìm cách phân biệt các bức ảnh nếu chúng không cùng nhóm. Do đó hiệu quả hơn rất nhiều so với các phương pháp trước đây.

Các tác giả đã sử dụng các loại kiến trúc mô hình CNN khác nhau và đã có những độ chính xác trong nhận diện giữa các kiến trúc [8].

Bảng 2.1 Bảng so sánh hiệu suất của các kiến trúc trong mạng FaceNet

|  |  |
| --- | --- |
| Architectures | Val |
| NN1 (Zeiler & Fergus 220x220) | 87.9 % 1.9 |
| NN2 (Inception 224x224) | 89.4 % 1.6 |
| NN3 (Inception 160x160) | 88.3 % 1.7 |
| NN4 (Inception 96x96) | 82.0 % 2.3 |
| NNS1 (mini Inception 165x165) | 82.4 % 2.4 |
| NNS2 (tiny Inception 140x116) | 51.9 % 2.9 |

#### 2.3.2.3. Hàm Triplet loss

Trong FaceNet, quá trình nhúng của mạng CNN đã giúp ta mã hóa bức ảnh về 128 chiều. Sau đó những vector này sẽ làm đầu vào cho hàm loss function mà có tác dụng phân biệt tốt các véc tơ giống hoặc khác nhau. Đó chính là hàm loss function.

Để áp dụng Triplet loss, chúng ta cần lấy ra ba bức ảnh, trong đó hai bức ảnh là của cùng một người, một bức là ảnh của người khác. Trong ba ảnh thì ra cố định một ảnh anchor. Chúng ta sẽ lựa chọn hai ảnh còn lại sao cho một ảnh là negative (người khác với anchor) và một ảnh là positive (cùng người với anchor).

Positive Anchor Negative

Kí hiệu ảnh Anchor, Positive, Negative lần lượt là A,P,N.

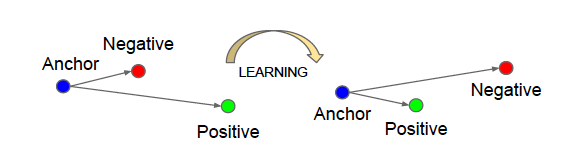
Khi đó khoảng cách của Anchor và Positive là :

Khoảng cách của Achor và Negative là:

Mục tiêu của hàm loss function là tối thiểu hóa khoảng cách giữa 2 ảnh khi chúng là negative và tối đa hóa khoảng cách khi chúng là positive. Như vậy chúng ta kì vọng rằng:

+ Ảnh Anchor và Positive giống nhau: khoảng cách d(A,P) sẽ nhỏ.

+ Ảnh Anchor và Negative khác nhau: khoảng cách d(A,N) sẽ lớn.



Hình 2. 14 Mục tiêu của hàm Triplet loss

Triplet loss function luôn lấy 3 bức ảnh làm input và trong mọi trường hợp ta kì vọng:

Và để tăng khoảng cách giữa d(A,P) và d(A,N) ta thêm vào một giá trị (gọi là margin)

Như vậy hàm loss function sẽ là:

Mục tiêu của chúng ta là giảm giá trị của biểu thức

tức là giảm khoảng cách giữa hai ảnh giống nhau, tăng giá trị của biểu thức tức là tăng khoảng cách giữa hai ảnh giống nhau. Như vậy giá trị của hàm loss function sẽ nhỏ hơn 0.

#### 2.3.2.4. Hoạt động của mạng FaceNet

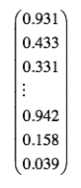
Hầu hết chúng ta khi xây dựng một thuật toán nhận diện khuôn mặt sẽ không cần phải train lại mô hình FaceNet mà tận dụng lại các mô hình Pre-train sẵn có. Bạn sẽ không cần phải tốn thời gian và công sức nếu không có đủ tài nguyên và dữ liệu.

Những mô hình Pre-train được huấn luyện trên các dữ liệu lên tới hàng triệu ảnh. Do đó có khả năng mã hóa rất tốt các bức ảnh trên không gian 128 chiều. Việc còn lại của chúng ta là sử dụng lại mô hình, tính toán, phân loại các khuôn mặt dựa trên vector đã được nhúng qua mô hình.

Vector embeddings

FaceNet model

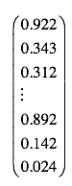
Input images





Khoảng cách Euclide giữa 2 bức ảnh

FaceNet





Phân loại

FaceNet

Hình 2. 15 Mô hình hoạt động nhận dạng khuôn mặt bằng mạng FaceNet

Khoảng cách Euclide giữa 2 vector được tính theo công thức:

Sau đó ta so sánh khoảng cách giữa 2 bức ảnh với một ngưỡng threshold, nếu nhỏ hơn ngưỡng thì 2 bức ảnh đó là cùng một người, ngược lại là hai người khác nhau.

### 2.3.3. Đánh giá thực nghiệm mô hình FaceNet

Hiệu quả của mô hình FaceNet được thực hiện trên các bộ cơ sở dữ liệu chuẩn (chứa các khung hình được thu nhận từ các thiết bị camera, webcam khác nhau), được công bố chung cho các nhóm nghiên cứu trên thế giới [9] như bộ cơ sở dữ liệu LFW bao gồm những khuôn mặt được gán nhãn trong tự nhiên. Bộ dữ liệu gồm 13233 hình ảnh khuôn mặt của 5749 người thu thập từ web. Hay bộ cơ sở dữ liệu JAFFE chứa các khuôn mặt nữ Nhật Bản, được chụp tại khoa tâm lý học của Đại học Kyushu, Nhật Bản, bao gồm 213 hình ảnh của 7 biểu hiện khuôn mặt. Sử dụng một số tập dữ liệu thử nghiệm ta thu được một bảng đánh giá độ chính xác khi nhận dạng với mô hình mạng FaceNet như sau [10]

Bảng 2.2 Bảng đánh giá độ chính xác mô hình FaceNet

|  |  |  |
| --- | --- | --- |
| Tập dữ liệu thử nghiệm | Số khuôn mặt cần nhận dạng | Độ chính xác (%) |
| Face96 | 3040 | 98.02 |
| FEI\_P1 | 700 | 98.16 |
| FEI\_P2 | 700 | 98.74 |
| FEI\_P3 | 700 | 97.55 |
| JAFFE | 213 | 99.02 |
| LFW | 13233 | 95.26 |

CHƯƠNG 3. XÂY DỰNG HỆ THỐNG NHẬN Diện KHUÔN MẶT TRÊN NỀN TẢNG ĐIỆN THOẠI ANDROID

## 3.1. Tổng quan về hệ thống nhận diện khuôn mặt trên thiết bị di động

### 3.1.1. Yêu cầu của hệ thống

Yêu cầu đặt ra là xây dựng một ứng dụng có khả năng nhận vào là dữ liệu trực tiếp từ camera của điện thoại, sau đó thực hiện việc phát hiện và nhận dạng danh tính của người có khuôn mặt xuất hiện trong camera một cách liên tục trong thời gian thực.

### 3.1.2. Đặc điểm nhận dạng khuôn mặt trên thiết bị di động

Hiện nay các thiết bị di động đang ngày càng được nâng cấp về cấu hình (vi xử lý, RAM, dung lượng lưu trữ), tuy nhiên khả năng xử lý, tính toán và lưu trữ của nó vẫn còn thua kém rất nhiều so với máy tính nên việc nhận dạng trên thiết bị di động chỉ nên được thực hiện với quy mô nhỏ.

Hầu hết các thư viện, các model dùng để xử lý quá trình phát hiện và nhận dạng khuôn mặt đều được viết cho các ngôn ngữ lập trình trên máy tính, vì vậy để có thể sử dụng các thư viện, các model này chúng ta phải đưa các thư viện này vào hệ thống trên điện thoại di động một cách thủ công, phải qua các bước chuyển đổi để có thể sử dụng được.

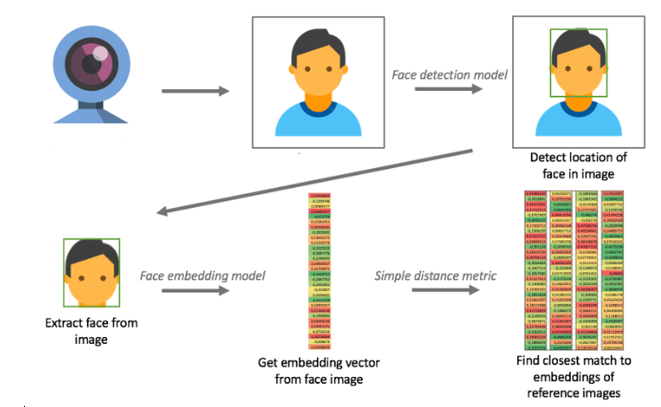
Một yếu tố ảnh hưởng đến khả năng nhận dạng chính là việc thu nhận ảnh trực tiếp bằng camera của thiết bị di động thường ảnh hưởng bởi quá trình lấy nét, đồng thời ảnh thu được thường chứa các nhiễu.

Việc nhận diện trên thiết bị di động có một ưu điểm rất lớn là tính thuật tiện của nó, chúng ta có thể thực hiện việc nhận dạng bất cứ lúc nào, bất cứ ở đâu. Ta chỉ việc lấy chiếc điện thoại di động của mình ra và thực hiện việc nhận dạng mà không cần mang theo cả một hệ thống máy tính.

### 3.1.3. Công cụ, thư viện sử dụng

Để xây dựng hệ thống nhận diện khuôn mặt trên nền tảng Andoird, trong đồ án này tôi lựa chọn ngôn ngữ lập trình là Android với bộ công cụ Android Studio 3.4.2 dùng để lập trình. Thư viện sử dụng là OpenCv dùng để xử lý ảnh, bộ phân loại Haar Cascades của Viola và Johns dùng để phát hiện khuôn mặt và cuối cùng là một model dùng để trích xuất các đặc trưng của ảnh chứa mặt người, model này được tạo ra bởi nhà nghiên cứu Hiroki Taniai, model này đã được huấn luyện trên bộ dữ liệu MS-Celeb-1M [11] bao gồm dữ liệu của một triệu người nổi tiếng trên thế giới do Microsoft tạo nên.

### 3.1.4. Mô hình hoạt động của hệ thống



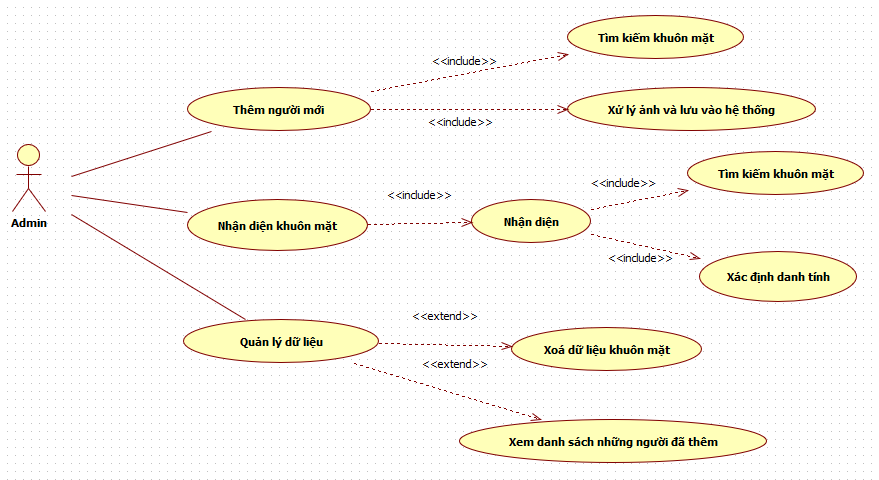
**Frame**

Hình 3. 1 Mô hình hoạt động nhận diện khuôn mặt

Công việc nhận diện khuôn mặt sẽ được thực hiện qua ba bước chính. Đầu tiên ta sẽ lấy dữ liệu từ camera là các khung hình (frame) có chứa khuôn mặt sau đó phát hiện và khoanh vùng vào vị trí có chứa khuôn mặt. Tiếp sau đó ta sẽ cắt vào tạo ra một khung hình chỉ chứa khuôn mặt người và đưa vào model để trích xuất ra các đặc trưng là một vector. Cuối cùng sau khi có các đặc trưng thì ta mang đi so sánh trong cơ sở dữ liệu xem vector này gần với khuôn mặt nào nhất.

## 3.2. Thiết kế các chức năng chính và giao diện ứng dụng

### 3.2.1. Các chức năng chính của hệ thống

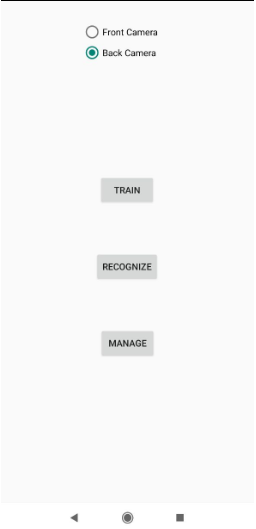


Hình 3. 2 Các chức năng của hệ thống

Hệ thống sẽ gồm 3 chức năng chính đó là thêm một người mới vào hệ thống, nhận diện khuôn mặt và quản lý danh sách những người có trong hệ thống. Để thêm một người mới ta phải xây dựng các chức năng tìm kiếm khuôn mặt sau đó xử lý ảnh và lưu dữ liệu khuôn mặt của người mới đó vào hệ thống cơ sở dữ liệu. Để nhận diện khuôn mặt thì ta cũng phải tìm kiếm được khuôn mặt của người trong các khung hình sau đó trích xuất các đặc trưng rồi từ đó xác định danh tính của khuôn mặt. Còn để quản lý danh sách những người có trong hệ thống ta phải xây dựng chức năng xem danh sách những người đã có trong hệ thống. Từ đó có thể xoá họ đi khỏi hệ thống.

### 3.2.2. Giao diện ứng dụng

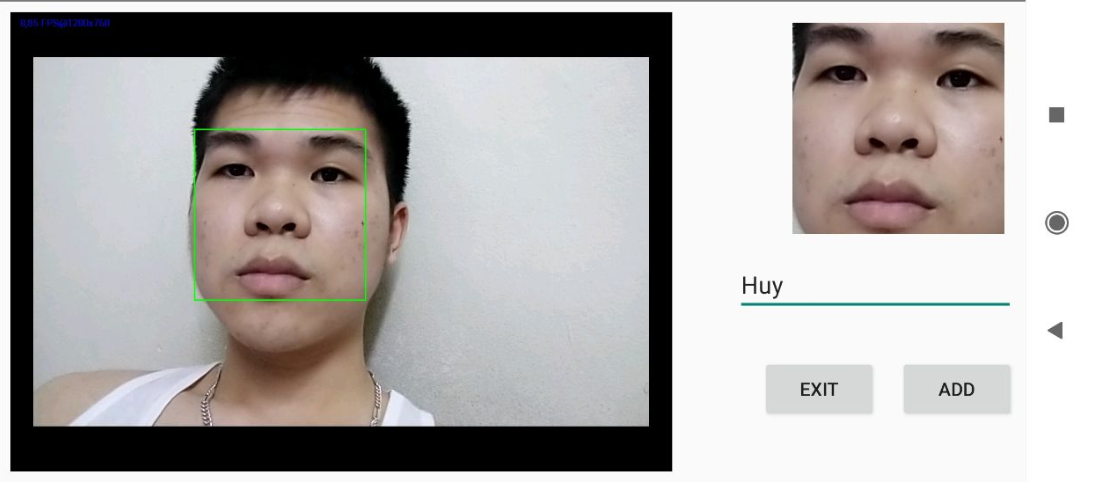
Màn hình đầu tiên sẽ là màn hình chính của ứng dụng



Hình 3. 3 Màn hình chính của ứng dụng

Màn hình chính của ứng dụng khi chúng ta mới truy cập vào sẽ cung cấp các chức năng như lựa chọn chiều của camera là camera trước hoặc camera sau. Sau đó là việc lựa chọn các chức năng chính của ứng dụng như thêm một người mới vào hệ thống (TRAIN), nhận dạng khuôn mặt (RECOGNIZE) hay quản lý danh sách những người có trong hệ thống (MANAGE).

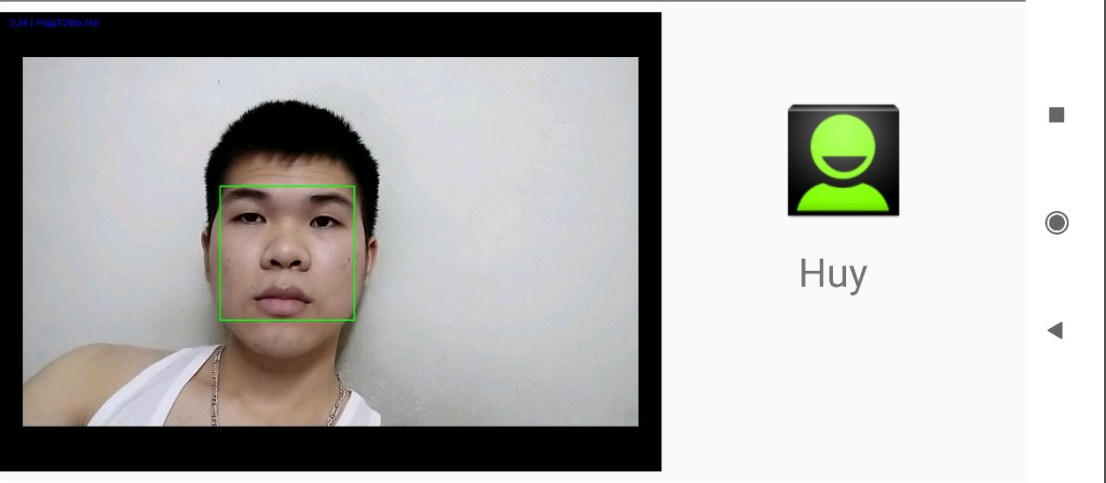
Tiếp theo sẽ là màn hình khi chúng ta lựa chọn chức năng thêm người mới (TRAIN)



Hình 3. 4 Màn hình thêm người mới

Tại đây ta sẽ có một khung để hiện thị camera trực tiếp, một imageView để hiện thị khuôn mặt sau khi được phát hiện và cắt ra khỏi khung hình, một editText để nhập tên của người chúng ta muốn thêm vào hệ thống, hai button thể hiện chức năng thêm dữ liệu của người mới vào hệ thống hoặc thoát ra ngoài màn hình chính.

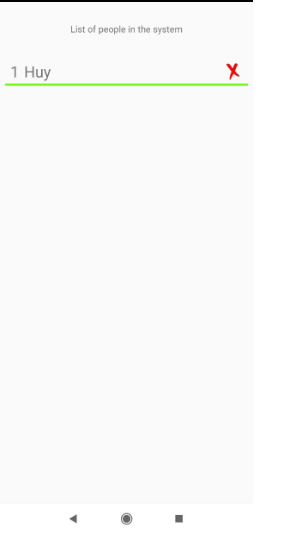
Tiếp theo là màn hình giúp chúng ta nhận diện khuôn mặt khi lựa chọn chức năng (RECOGNIZE)



Hình 3. 5 Màn hình nhận diện khuôn mặt

Tại đây ta cũng có một khung để hiện thi luồng dữ liệu trực tiếp từ camera của điện thoại, tại mỗi frame hệ thống sẽ nhận diện xem khuôn mặt này là của ai. Vì vậy ứng dụng này luôn nhận diện khuôn mặt tại thời gian thực chứ không cần phải chụp ảnh.

Cuối cùng sẽ là giao diện của màn hình quản lý danh sách những người đã được thêm vào hệ thống



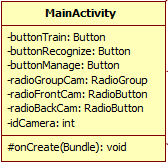
Hình 3. 6 Màn hình quản lý

Màn hình này giúp chúng ta có thể xem được trong hệ thống có bao nhiêu người từ đó có thể xoá bỏ những người không còn trong hệ thống một cách dễ dàng.

## 3.3. Xây dựng hệ thống

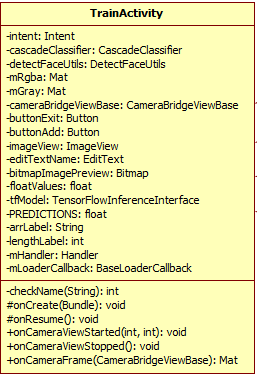
### 3.3.1. Các lớp dữ liệu

Ứng dụng sẽ được xây dựng từ các lớp dữ liệu sau:



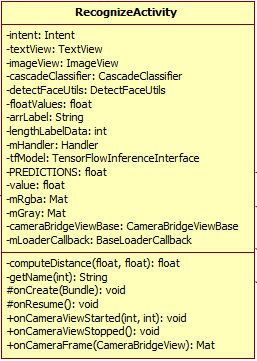
Hình 3. 7 Class MainActivity

Class MainActivity sẽ xử lý việc lựa chọn camera để nhận dạng khuôn mặt (camera trước hoặc sau), lắng nghe sự kiện của các nút để mở các màn hình chức năng tương ứng.



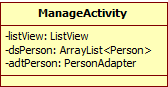
Hình 3. 8 Class TrainActivity

Class TrainActivity để thực hiện việc huấn luyện dữ liệu và thêm người mới vào hệ thống. Nhận frame từ camera sau đó phát hiện khuôn mặt, từ đó trích xuất đặc điểm và lưu trữ vào cơ sở dữ liệu.



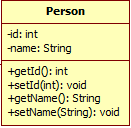
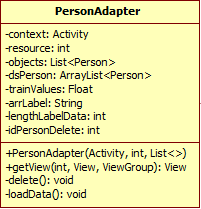
Hình 3. 9 Class RecognizeActivity

Class RecognizeActivity để quản lý màn hình nhận diện khuôn mặt. Nhận frame từ camera sau đó phát hiện khuôn mặt, đưa vào model để trích xuất đặc trưng. Từ đó so sánh khoảng cách với các khuôn mặt có trong cơ sở dữ liệu và hiện thị kết quả.



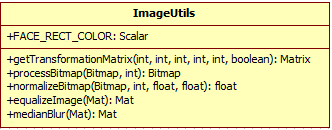
Hình 3. 10 Class ManageActivity

Class ManageActivity để thực hiện việc hiển thị danh sách những người có trong hệ thống và có thể xoá những người không còn trong hệ thống.



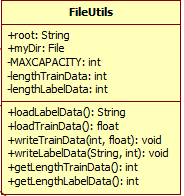
Hình 3. 11 Class PersonAdapter

Class PersonAdapter để quản lý danh sách những người trong hệ thống. Cập nhật cơ sở dữ liệu về những người trong hệ thống.



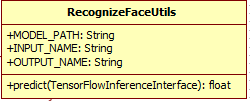
Hình 3. 12 Class ImageUtils

Class ImageUtils để thực hiện các thao tác tương tác với ảnh, chỉnh sửa, điều chỉnh kích thước, cân bằng sáng, giảm nhiễu.



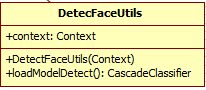
Hình 3. 13 Class FileUtils

Class FileUtils để thực hiện các thao tác với file, đọc ghi file cơ sở dữ liệu.



Hình 3. 14 Class RecognizeFaceUtils

Class RecognizeFaceUtils để thực hiện việc tương tác với model nhận dạng, trích xuất các đặc trưng của khuôn mặt.



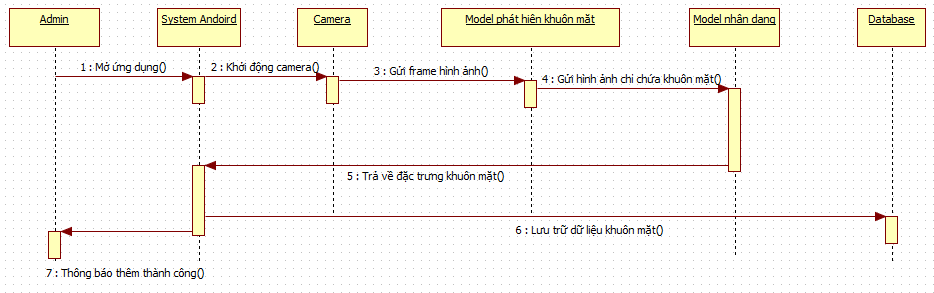
Hình 3. 15 Class DetectFaceUtils

Class DetectFaceUtils để thực hiện việc phát hiện khuôn mặt có trong các khung hình.

### 3.3.2. Quá trình thực hiện trong ứng dụng

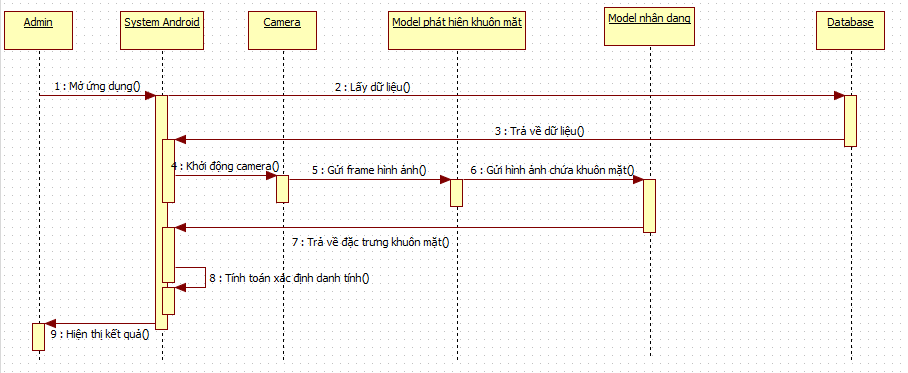
Quá trình tương tác của các thành phần trong hệ thống khi thực hiện hai chức năng chính của ứng dụng là thêm người mới và nhận dạng khuôn mặt được thực hiện như các sơ đồ dưới đây.

Quá trình huấn luyện để thêm người mới vào hệ thống



Hình 3. 16 Quá trình huấn luyện khuôn mặt

Quá trình nhận dạng khuôn mặt.



Hình 3. 17 Quá trình nhận dạng khuôn mặt

### 3.3.3. Thực hiện xây dựng các chức năng

#### 3.3.3.1. Thêm người mới

Đầu tiên từ khung hình của camera chúng ta sẽ lấy ra một ảnh màu và một ảnh xám.

**mRgba** = inputFrame.rgba();  
**mGray** = inputFrame.gray();

Sau đó ta sẽ phát hiện khuôn mặt có trong khung hình và đánh dấu nó bằng cách vẽ một khung hình chữ nhật quanh khuôn mặt.

MatOfRect list\_face = **new** MatOfRect();  
**cascadeClassifier**.detectMultiScale(**mGray**,list\_face,1.1,5,***CASCADE\_SCALE\_IMAGE***, **new** Size(70,70), **new** Size());

Rect[] list = list\_face.toArray();

Rect r = list[0];

**for**(**int** i = 0; i < list.**length**; i++){  
 Imgproc.*rectangle*(**mRgba**, **new** Point(list[i].**x**, list[i].**y**),  
 **new** Point(list[i].**x**+list[i].**width**,list[i].**y**+list[i].**height**),  
 ImageUtils.*FACE\_RECT\_COLOR*,2);  
}

Tiến hành cắt khung hình là một hình chỉ chứa khuôn mặt, xử lý cân bằng sáng và giảm nhiễu sau đó chuyển về kiểu dữ liệu Bitmap.

m = ImageUtils.*equalizeImage*(m);m = ImageUtils.*medianBlur*(m);  
  
Bitmap bitmap= Bitmap.*createBitmap*(m.width(), m.height(), Bitmap.Config.***ARGB\_8888***);  
Utils.*matToBitmap*(m, bitmap);  
**bitmapTrain** = bitmap;

Sau đó ta sẽ đưa hình ảnh chỉ chứa khuôn mặt vào model để trích xuất đặc trưng. Trước khi đưa hình ảnh vào model ta cần điều chỉnh kích thước của ảnh để phù hợp với đầu vào của model.

Bitmap resized\_image = ImageUtils.*processBitmap*(**bitmapTrain**,160);  
**floatValues** = ImageUtils.*normalizeBitmap*(resized\_image,160,80.5f,1.0f);

**PREDICTIONS** = RecognizeFaceUtils.*predict*(**tfModel**, **floatValues**);

Sau khi trích xuất được một mảng PREDICTIONS có 128 phần tử ta sẽ lưu lại vào file cơ sở dữ liệu cùng với tên của chủ khuôn mặt.

FileUtils.*writeTrainData*(id, **PREDICTIONS**);  
FileUtils.*writeLabelData*(name, id);

#### 3.3.3.2. Nhận diện khuôn mặt

Bước đầu tiên của quá trình nhận diện khuôn mặt cũng giống như quá trình thêm người mới đó là phát hiện, đánh dấu khuôn mặt, xử lý ảnh và đưa vào model để trích xuất đặc trưng.

Sau khi trích xuất được đặc trưng của khuôn mặt thì ta tiến hành so sánh khoảng cách của vector đặc trưng mới cần nhận dạng với các vector đặc trưng trong cơ sở dữ liệu.

**float** min = 99999;  
**int** idMin = 0;  
**for** (**int** j = 0; j < **lengthTrainData**; j++) {  
 **float** dis = computeDistance(**PREDICTIONS**, **value**[j]);  
 **if** (dis < min) {  
 min = dis;  
 idMin = (**int**) **value**[j][128];  
 }  
}

Sau khi tìm được vector đặc trưng gần giống với khuôn mặt cần nhận dạng thì ta cần kiểm tra xem khoảng cách giữa 2 vector đặc trưng gần nhau nhất có nhỏ hơn một ngưỡng THRESHOLD không. Nếu nhỏ hơn thì ta mới đi tìm trong cơ sở dữ liệu xem với idMin này thì khuôn mặt đó là của ai.

String name = getName(idMin);  
  
**if** (min < ***THRESHOLD***) {  
 msg.**obj** += name+**"\n"**;  
} **else** {  
 msg.**obj** += **""**;  
}

Cuối cùng ta sẽ hiển thị kết quả lên màn hình xem khuôn mặt này là của ai hoặc khuôn mặt này không có trong hệ thống.

**mHandler** = **new** Handler() {  
 @Override  
 **public void** handleMessage(Message msg) {  
 String tempName = msg.**obj**.toString();  
 **if**(!tempName.equals(**""**)){  
 **imageView**.setImageResource(R.drawable.***ic\_green***);  
 **textView**.setText(tempName);  
 }**else** {  
 **imageView**.setImageResource(R.drawable.***ic\_red***);  
 **textView**.setText(**"unknow"**);  
 }  
 }  
  
};

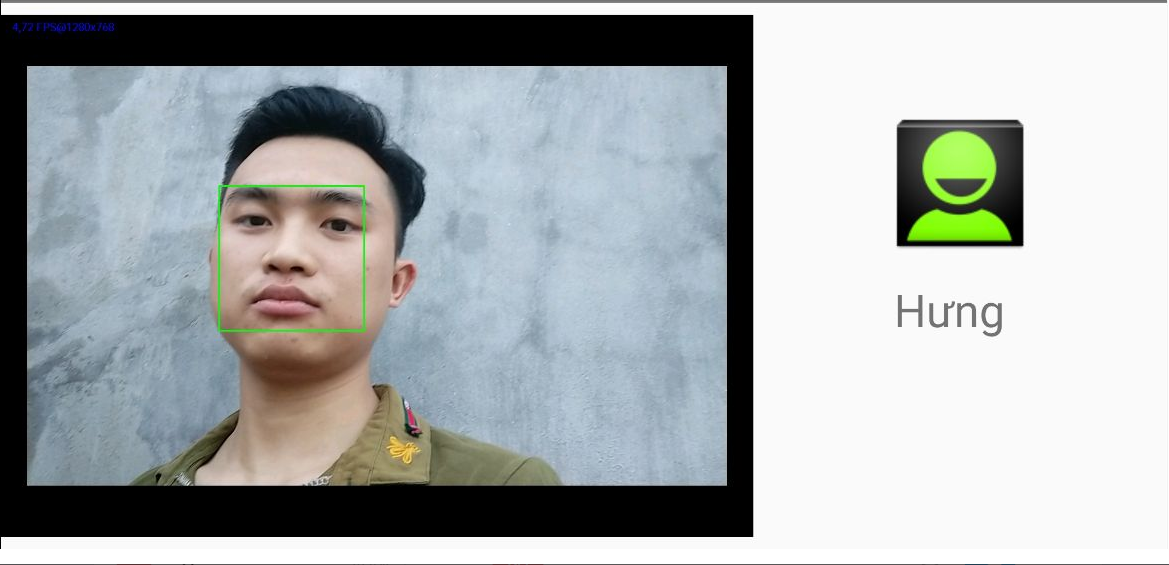
## 3.4. Thử nghiệm ứng dụng

### 3.4.1. Môi trường thực hiện ứng dụng

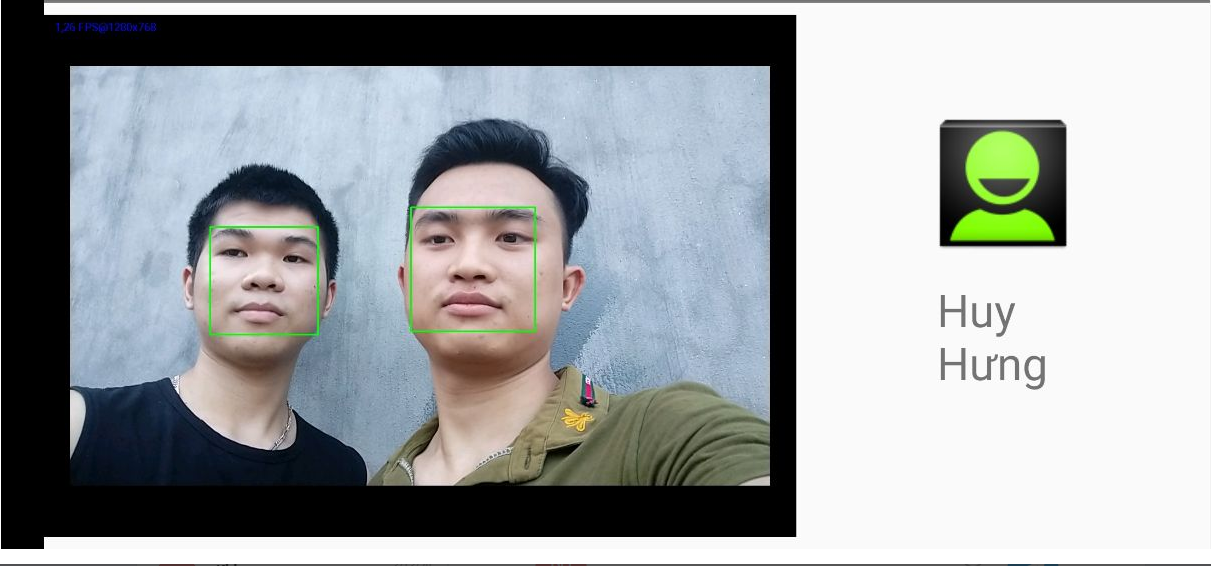
Ứng dụng được thử nghiệm trên điện thoại di động chạy hệ điều hành Android 9, RAM 3GB.

### 3.4.2. Kết quả thử nghiệm

Khi nhận diện 1 người



Khi nhận diện 2 người



KẾT LUẬN

## Kết quả đã thực hiện được của đồ án

Với ý tưởng áp dụng trí tuệ nhân tạo vào các nhu cầu của đời sống, nhằm giải phóng sức lao động của con người với những công việc đơn giản và góp phần xây dựng cách mạng công nghiệp 4.0, đề tài “*Nghiên cứu, xây dựng ứng dụng nhận diện khuôn mặt trên Android”* đã hoàn thành được những mục tiêu sau:

* Ở chương một, đồ án đã trình bày tổng quan về xử lý ảnh, cũng như các kiến thức về quá trình phát hiện khuôn mặt người có trong ảnh. Đưa ra một số cách tiếp cận cho bài toán phát hiện khuôn mặt. Tìm một giải pháp tối ưu để phát hiện được khuôn mặt một cách nhanh và chính xác.
* Ở chương hai, đồ án đã trình bày về bài toán nhận diện khuôn mặt sau khi đã được phát hiện. Giới thiệu tổng quan về nhận dạng khuôn mặt, cách công dụng phổ biến của nhận diện bằng khuôn mặt. Tìm hiều một số cách nhận diện khuôn mặt. Giới thiệu về cách nhận diện khuôn mặt bằng mô hình mạng FaceNet.
* Ở chương ba, đồ án đã đi vào quá trình xây dựng một ứng dụng nhận diện khuôn mặt được trên khai trên hệ điều hành Android. Phân tích, xây dựng các chức năng chính cần có của ứng dụng, ưu nhược điểm khi triên khai ứng dụng trên điện thoại di động.

## Hướng phát triển đồ án

Với rất nhiều những ứng dụng thực tế của bài toán nhận diện khuôn mặt. Đồ án có một số hướng phát triển trong tương lai, để tạo thành một hệ thống hoàn thiện hơn như:

* Phát triển thành ứng dụng điểm danh sinh viên trong một lớp học
* Thực hiện bảo mật dữ liệu thông qua xác thực khuôn mặt

TÀI LIỆU THAM KHẢO

[1]. *Mô hình màu RGB*, https://vi.wikipedia.org/wiki/M%C3%B4\_h%C3%ACnh\_m%C3%A0u\_RGB

[2]. *Nguyên nhân khiến khuôn mặt con người không ai giống ai*, https://khoahoc.tv/nguyen-nhan-khien-khuon-mat-con-nguoi-khong-ai-giong-ai-56493

[3]. Alrashed, H. H. *Face Recognition On Android,* Submitted in partial fulfilment of the requirements for the first year of Master of Information Sciences Massey University 2014

[4]. *Face Detection using Haar Cascades*, https://opencv-python-tutroals.readthedocs.io/en/latest/py\_tutorials/py\_objdetect/py\_face\_detection/py\_face\_detection.html

[5]. Viola and Jones. *Rapid object detection using a boosted cascade of simple feature*, Computer Vision and Pattern Recognition, 2001.

[6]. *Viola–Jones object detection framework*, https://en.wikipedia.org/wiki/Viola%E2%80%93Jones\_object\_detection\_framework

[7]. Sumit Saha. *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*, <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>.

[8]. Florian Schroff, Dmitry Kalenichenko, James Philbin. *FaceNet: A Unified Embedding for Face Recognition and Clustering,* 2015.

[9]. Face recongnition homepage, <https://face-rec.org/databases>

[10]. Nguyễn Thị Thanh Tân, Huỳnh Văn Huy, Ngô Quốc Tạo, *Hội thảo quốc gia lần XX: Một số vấn đề chọn lọc của Công nghệ thông tin và truyền thông*, 2017

[11]. Microsoft, *MS-Celeb-1M: Challenge of Recognizing One Million Celebrities in the Real World,* June 29, 2016.

PHỤ LỤC

Mã chương trình quá trình thêm người mới vào hệ thống

**public class** TrainActivity **extends** AppCompatActivity **implements** Serializable, CameraBridgeViewBase.CvCameraViewListener2 {  
 **private** Intent **intent**;  
 **private static final** String ***TAG*** = **"Train::Activity"**;  
 **private** CascadeClassifier **cascadeClassifier**;  
 **private** DetectFaceUtils **detectFaceUtils**;  
 **private** Mat **mRgba**, **mGray**;  
 **private** CameraBridgeViewBase **cameraBridgeViewBase**;  
 **private** Button **buttonExit**, **buttonAdd**;  
 **private** ImageView **imageView**;  
 **private** EditText **editTextName**;  
 **private** Bitmap **bitmapImagePreview**, **bitmapTrain**;  
 **private float**[] **floatValues** = **new float**[160 \* 160 \* 3];  
 **private** TensorFlowInferenceInterface **tfModel**;  
 **private float**[] **PREDICTIONS** = **new float**[128];  
 **private** String[][] **arrLabel**;  
 **private int lengthLabel**;  
 **private** Handler **mHandler**;  
 **static** {  
 System.*loadLibrary*(**"tensorflow\_inference"**);  
 }  
  
  
  
  
  
 **private** BaseLoaderCallback **mLoaderCallback** = **new** BaseLoaderCallback(**this**) {  
 @Override  
 **public void** onManagerConnected(**int** status) {  
 **switch** (status) {  
 **case** LoaderCallbackInterface.***SUCCESS***:  
 {  
 Log.*i*(***TAG***, **"OpenCV loaded successfully"**);**detectFaceUtils** = **new** DetectFaceUtils(getApplication());  
 **cascadeClassifier** = **detectFaceUtils**.*loadModelDetect*();  
 **cameraBridgeViewBase**.enableView();  
  
 } **break**;  
 **default**:  
 {  
 **super**.onManagerConnected(status);  
 } **break**;  
 }  
 }  
 };

**private int** checkName(String name){  
  
 **if**(**lengthLabel** == 0)  
 **return** 1;  
 **for**(**int** i = 0; i < **lengthLabel**; i++){  
 **if**(**arrLabel**[i][0].equalsIgnoreCase(name)){  
 **return** Integer.*parseInt*(**arrLabel**[i][1]);  
 }  
 }  
 **for**(**int** i = 1; i <= FileUtils.***MAXCAPACITY***; i++){  
 **boolean** check = **true**;  
 **for**(**int** j = 0; j < **lengthLabel**; j++){  
 **if**(Integer.*parseInt*(**arrLabel**[j][1]) == i) {  
 check = **false**;  
 **break**;  
 }  
 }  
 **if**(check == **true**){  
 **return** i;  
 }  
 }  
 **return** -1;  
 }  
  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 getWindow().addFlags(WindowManager.LayoutParams.***FLAG\_KEEP\_SCREEN\_ON***);  
 setContentView(R.layout.***activity\_train***);  
  
 **intent** = getIntent();  
  
 **cameraBridgeViewBase** = findViewById(R.id.***myCameraView***);  
 **cameraBridgeViewBase**.setCvCameraViewListener(**this**);  
 **cameraBridgeViewBase**.setVisibility(SurfaceView.***VISIBLE***);  
  
 **cameraBridgeViewBase**.setCameraIndex(**intent**.getIntExtra(**"idCamera"**, 0));  
  
 **buttonExit** = findViewById(R.id.***buttonExit***);  
 **buttonAdd** = findViewById(R.id.***buttonAdd***);  
 **editTextName** = findViewById(R.id.***editTextName***);  
 **imageView** = findViewById(R.id.***imageView2***);  
  
 **tfModel** = **new** TensorFlowInferenceInterface(getAssets(),RecognizeFaceUtils.***MODEL\_PATH***);  
 **arrLabel** = **new** String[FileUtils.***MAXCAPACITY***][2];  
  
 **if** (!FileUtils.*myDir*.exists()) {  
 FileUtils.*myDir*.mkdirs();  
 }  
 **arrLabel** = FileUtils.*loadLabelData*();  
 **lengthLabel** = FileUtils.*getLengthLabelData*();  
  
  
 **mHandler** = **new** Handler() {  
 @Override  
 **public void** handleMessage(Message msg) {  
 **if** (msg.**obj**==**"IMG"**)  
 {  
 **imageView**.setImageBitmap(**bitmapImagePreview**);  
 }  
 }  
 };  
  
  
 **buttonExit**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
  
 finish();  
 }  
 });  
  
 **buttonAdd**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **imageView**.setImageBitmap(**bitmapImagePreview**);  
 String name = **editTextName**.getText().toString().trim();  
  
 **if**(!**editTextName**.getText().toString().isEmpty()){  
 *//Resize ảnh về kích thước 160 x 160* Bitmap resized\_image = ImageUtils.*processBitmap*(**bitmapTrain**,160);  
  
 *//Chuyển các pixel của ảnh về thành mảng float* **floatValues** = ImageUtils.*normalizeBitmap*(resized\_image,160,80.5f,1.0f);  
  
 **PREDICTIONS** = RecognizeFaceUtils.*predict*(**tfModel**, **floatValues**);  
  
 name = name.substring(0,1).toUpperCase() + name.substring(1).toLowerCase();  
 **int** id = checkName(name);  
  
 FileUtils.*writeTrainData*(id, **PREDICTIONS**);  
  
 FileUtils.*writeLabelData*(name, id);  
  
 Toast.*makeText*(TrainActivity.**this**, **"Add succes: "**+name, Toast.***LENGTH\_SHORT***).show();  
  
 }**else**{  
 Toast.*makeText*(TrainActivity.**this**, **"Add a person name !"**, Toast.***LENGTH\_SHORT***).show();  
 }  
 }  
 });  
  
 }  
  
  
  
 @Override  
 **protected void** onResume() {  
 **super**.onResume();  
 **if** (!OpenCVLoader.*initDebug*()) {  
 Log.*d*(***TAG***, **"Internal OpenCV library not found. Using OpenCV Manager for initialization"**);  
 OpenCVLoader.*initAsync*(OpenCVLoader.***OPENCV\_VERSION***, **this**, **mLoaderCallback**);  
 } **else** {  
 Log.*d*(***TAG***, **"OpenCV library found inside package. Using it!"**);  
 **mLoaderCallback**.onManagerConnected(LoaderCallbackInterface.***SUCCESS***);  
 }  
 }  
  
 @Override  
 **public void** onCameraViewStarted(**int** width, **int** height) {  
 **mGray** = **new** Mat();  
 **mRgba** = **new** Mat();  
 }  
  
 @Override  
 **public void** onCameraViewStopped() {  
 **mGray**.release();  
 **mRgba**.release();  
 }  
  
 @Override  
 **public** Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {  
 **mRgba** = inputFrame.rgba();  
 **mGray** = inputFrame.gray();  
  
 MatOfRect list\_face = **new** MatOfRect();  
 **cascadeClassifier**.detectMultiScale(**mGray**,list\_face,1.1,5,***CASCADE\_SCALE\_IMAGE***, **new** Size(70,70), **new** Size());  
  
 Rect[] list = list\_face.toArray();  
  
  
 **if**(list.**length** == 1 && !**editTextName**.getText().toString().isEmpty()) {  
 Rect r = list[0];  
 Mat m = **mGray**.submat(r);  
  
 Mat preview = **mRgba**.submat(r);  
 Bitmap bitmapPreview= Bitmap.*createBitmap*(preview.width(), preview.height(), Bitmap.Config.***ARGB\_8888***);  
 Utils.*matToBitmap*(preview, bitmapPreview);  
 **bitmapImagePreview** = bitmapPreview;  
  
 *// tang do tuong phan, can bang sang* m = ImageUtils.*equalizeImage*(m);

*// Giam nhieu cua anh* m = ImageUtils.*medianBlur*(m);  
  
 Bitmap bitmap= Bitmap.*createBitmap*(m.width(), m.height(), Bitmap.Config.***ARGB\_8888***);  
 Utils.*matToBitmap*(m, bitmap);  
 **bitmapTrain** = bitmap;  
  
 Message msg = **new** Message();  
 String textTochange = **"IMG"**;  
 msg.**obj** = textTochange;  
 **mHandler**.sendMessage(msg);  
  
 }  
  
  
 **for**(**int** i = 0; i < list.**length**; i++){  
 Imgproc.*rectangle*(**mRgba**, **new** Point(list[i].**x**, list[i].**y**),  
 **new** Point(list[i].**x**+list[i].**width**,list[i].**y**+list[i].**height**),  
 ImageUtils.*FACE\_RECT\_COLOR*,2);  
 }  
  
 **return mRgba**;  
 }  
  
  
}

Mã chương trình quá nhận dạng khuôn mặt

**public class** RecognizeActivity **extends** AppCompatActivity **implements** CameraBridgeViewBase.CvCameraViewListener2 {  
 **private** Intent **intent** = getIntent();  
 **private** TextView **textView**;  
 **private** ImageView **imageView**;  
 **private static final** String ***TAG*** = **"Recognize::Activity"**;  
 **private** CascadeClassifier **cascadeClassifier**;  
 **private** DetectFaceUtils **detectFaceUtils**;  
 **private float**[] **floatValues** = **new float**[160 \* 160 \* 3];  
 **private** String[][] **arrLabel**;  
 **private int lengthLabelData**;  
 **private** Handler **mHandler**;  
 **private int lengthTrainData** = 0;  
 **private** TensorFlowInferenceInterface **tfModel**;  
 **private float**[] **PREDICTIONS** = **new float**[128];  
 **private float**[][] **value** = **new float**[FileUtils.***MAXCAPACITY***][129];  
 **private static final double *THRESHOLD*** = 0.2;  
  
  
 **private** Mat **mRgba**, **mGray**;  
 CameraBridgeViewBase **cameraBridgeViewBase**;  
  
 **static** {  
 System.*loadLibrary*(**"tensorflow\_inference"**);  
 }  
  
 **private** BaseLoaderCallback **mLoaderCallback** = **new** BaseLoaderCallback(**this**) {  
 @Override  
 **public void** onManagerConnected(**int** status) {  
 **switch** (status) {  
 **case** LoaderCallbackInterface.***SUCCESS***:  
 {  
 Log.*i*(***TAG***, **"OpenCV loaded successfully"**);  
 **detectFaceUtils** = **new** DetectFaceUtils(getApplication());  
 **cascadeClassifier** = **detectFaceUtils**.*loadModelDetect*();  
 **cameraBridgeViewBase**.enableView();  
  
 } **break**;  
 **default**:  
 {  
 **super**.onManagerConnected(status);  
 } **break**;  
 }  
 }  
 };  
  
  
 **private float** computeDistance(**float**[] x, **float**[] y){  
 **float** sum = 0;  
 **for**(**int** i = 0; i < 128; i++){  
 sum += (x[i]-y[i])\*(x[i]-y[i]);  
 }  
 **return** (**float**) Math.*sqrt*(sum);  
 }  
  
  
 **private** String getName(**int** id){  
 **for**(**int** i = 0; i < **lengthLabelData**; i++){  
 **if**(Integer.*parseInt*(**arrLabel**[i][1]) == id)  
 **return arrLabel**[i][0];  
 }  
 **return "Unknow"**;  
 }  
  
  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 getWindow().addFlags(WindowManager.LayoutParams.***FLAG\_KEEP\_SCREEN\_ON***);  
 setContentView(R.layout.***activity\_recognize***);  
  
 **intent** = getIntent();  
  
 **textView** = findViewById(R.id.***textView***);  
 **imageView** = findViewById(R.id.***imageView***);  
  
 **cameraBridgeViewBase** = findViewById(R.id.***myCameraView***);  
 **cameraBridgeViewBase**.setCvCameraViewListener(**this**);  
 **cameraBridgeViewBase**.setVisibility(SurfaceView.***VISIBLE***);  
 **cameraBridgeViewBase**.setCameraIndex(**intent**.getIntExtra(**"idCamera"**, 0));  
  
  
 **tfModel** = **new** TensorFlowInferenceInterface(getAssets(),RecognizeFaceUtils.***MODEL\_PATH***);  
  
 **arrLabel** = FileUtils.*loadLabelData*();  
 **lengthLabelData** = FileUtils.*getLengthLabelData*();  
 **value** = FileUtils.*loadTrainData*();  
 **lengthTrainData** = FileUtils.*getLengthTrainData*();  
  
 **mHandler** = **new** Handler() {  
 @Override  
 **public void** handleMessage(Message msg) {  
 String tempName = msg.**obj**.toString();  
 **if**(!tempName.equals(**""**)){  
 **imageView**.setImageResource(R.drawable.***ic\_green***);  
 **textView**.setText(tempName);  
 }**else** {  
 **imageView**.setImageResource(R.drawable.***ic\_red***);  
 **textView**.setText(**"unknow"**);  
 }  
 }  
  
 };  
  
 }  
  
  
  
 @Override  
 **protected void** onResume() {  
 **super**.onResume();  
 **if** (!OpenCVLoader.*initDebug*()) {  
 OpenCVLoader.*initAsync*(OpenCVLoader.***OPENCV\_VERSION***, **this**, **mLoaderCallback**);  
 } **else** {  
 **mLoaderCallback**.onManagerConnected(LoaderCallbackInterface.***SUCCESS***);  
 }  
 }  
  
  
 @Override  
 **public void** onCameraViewStarted(**int** width, **int** height) {  
 **mGray** = **new** Mat();  
 **mRgba** = **new** Mat();  
 }  
  
 @Override  
 **public void** onCameraViewStopped() {  
 **mGray**.release();  
 **mRgba**.release();  
 }  
  
 @Override  
 **public** Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {  
 **mRgba** = inputFrame.rgba();  
 **mGray** = inputFrame.gray();  
  
 MatOfRect list\_face = **new** MatOfRect();  
  
 **cascadeClassifier**.detectMultiScale(**mGray**,list\_face,1.2,6,0|***CASCADE\_SCALE\_IMAGE*** , **new** Size(70,70), **new** Size());  
  
 Rect[] list = list\_face.toArray();  
  
 **if**(list.**length** > 0) {  
 Message msg = **new** Message();  
 msg.**obj**=**""**;  
 **for**(**int** i = 0; i < list.**length**; i++) {  
 Rect r = list[i];  
 Mat m = **mGray**.submat(r);  
  
 *// tang do tuong phan, can bang sang* m = ImageUtils.*equalizeImage*(m);  
  
 *// Giam nhieu cua anh* m = ImageUtils.*medianBlur*(m);  
  
 Bitmap bitmapRecognize = Bitmap.*createBitmap*(m.width(), m.height(), Bitmap.Config.***ARGB\_8888***);  
 Utils.*matToBitmap*(m, bitmapRecognize);  
  
  
  
 *//Resize anh ve kich thuoc 160 x 160* Bitmap resized\_image = ImageUtils.*processBitmap*(bitmapRecognize, 160);  
  
 *//Dua cac pixel cua anh ve dang mang float* **floatValues** = ImageUtils.*normalizeBitmap*(resized\_image, 160, 80.5f, 1.0f);  
  
 **PREDICTIONS** = RecognizeFaceUtils.*predict*(**tfModel**, **floatValues**);  
  
 **float** min = 99999;  
 **int** idMin = 0;  
 **for** (**int** j = 0; j < **lengthTrainData**; j++) {  
 **float** dis = computeDistance(**PREDICTIONS**, **value**[j]);  
 **if** (dis < min) {  
 min = dis;  
 idMin = (**int**) **value**[j][128];  
 }  
 }  
   
 String name = getName(idMin);  
  
 **if** (min < ***THRESHOLD***) {  
 msg.**obj** += name+**"\n"**;  
 } **else** {  
 msg.**obj** += **""**;  
 }  
  
 }  
 **mHandler**.sendMessage(msg);  
  
 }  
  
 **for**(**int** i = 0; i < list.**length**; i++){  
 Imgproc.*rectangle*(**mRgba**, **new** Point(list[i].**x**, list[i].**y**),  
 **new** Point(list[i].**x**+list[i].**width**,list[i].**y**+list[i].**height**),  
 ImageUtils.*FACE\_RECT\_COLOR*,2);  
 }  
  
  
 **return mRgba**;  
 }  
}