

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA VẬT LÝ



NGUYỄN TRIỆU VƯƠNG

**NGHIÊN CỨU DEEP LEARNING ĐỂ TÍNH TOÁN
CÁC ĐẶC TRƯNG CỦA MẶT NGƯỜI**

Tiểu luận

Ngành Kỹ thuật Điện tử và Tin học
(Chương trình đào tạo chuẩn)

Hà Nội - 2023

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA VẬT LÝ



NGUYỄN TRIỆU VƯƠNG

**NGHIÊN CỨU DEEP LEARNING ĐỂ TÍNH TOÁN
CÁC ĐẶC TRƯNG CỦA MẶT NGƯỜI**

Tiểu luận

Ngành Kỹ thuật Điện tử và Tin học
(Chương trình đào tạo chuẩn)

Giảng viên hướng dẫn: TS. PHẠM TIẾN LÂM
ĐẶNG VĂN BÁU
ThS. NGUYỄN VĂN QUYỀN

Hà Nội - 2023

“Học hỏi từ quá khứ, sống cho hiện tại, hy vọng cho tương lai. Điều quan trọng là đừng bao giờ ngừng đặt câu hỏi.”

Albert Einstein

Lời cảm ơn

Lời đầu tiên, tôi xin bày tỏ sự biết ơn chân thành và sâu sắc nhất tới TS. Phạm Tiến Lâm - Giáo viên hướng dẫn khoa học, người đã tận tình hướng dẫn, hỗ trợ và giúp đỡ tôi trong quá trình nghiên cứu và hoàn thiện tiểu luận của mình...

Tôi xin gửi lời cảm ơn chân thành tới anh Đặng Văn Báu, anh Nguyễn Văn Quyền và các thầy cô, anh chị em công tác tại phòng Tin học Vật lý - Khoa Vật Lý - Trường Đại học Khoa học Tự nhiên - Đại học Quốc gia Hà Nội đã tận tình truyền đạt kiến thức và hướng dẫn cho tôi trong suốt quá trình học tập tại trường.

Trong quá trình hoàn thành tiểu luận do thời gian và khả năng kiến thức còn hạn chế cũng như chưa có nhiều kinh nghiệm làm bài luận nên khó tránh khỏi những sai sót. Kính mong nhận được sự cảm thông, góp ý của các thầy các cô.

Hà Nội, tháng 1 năm 2023

Sinh viên

Nguyễn Triệu Vương

Mục lục

Lời cảm ơn	ii
Danh sách hình vẽ	v
Danh sách tên viết tắt	vii
MỞ ĐẦU	1
1 Tổng quan về nhận dạng khuôn mặt	2
1.1 Tổng quan về xử lý ảnh	2
1.1.1 Một số khái niệm	2
1.1.2 Các vấn đề của xử lý ảnh	2
1.1.3 Ứng dụng của xử lý ảnh trong thực tế	3
1.2 Bài toán nhận dạng khuôn mặt	4
1.2.1 Khái niệm	4
1.2.2 Một số trở ngại của công nghệ nhận dạng khuôn mặt	4
1.2.3 Các ứng dụng đặc trưng của bài toán nhận dạng khuôn mặt	5
1.2.4 Xây dựng hệ thống nhận dạng khuôn mặt	5
1.2.5 Một số phương pháp nhận dạng khuôn mặt	6
2 Tìm hiểu về học sâu và mô hình mạng neural tích chập	7
2.1 Tổng quan về Học máy (Machine learning)	7
2.2 Các thuật toán Học máy	7
2.2.1 Học có giám sát (Supervised learning)	7
2.2.2 Học không giám sát (Unsupervised learning)	7
2.3 Tìm hiểu về Học sâu (Deep learning)	8
2.4 Tìm hiểu về CNN (Convolutional Neural Network)	8
2.5 Cấu trúc của CNN	9
2.5.1 Lớp tích chập (Convolution)	9
2.5.2 Lớp phi tuyến ReLU	11
2.5.3 Lớp Pooling	11
2.5.4 Lớp Fully-connected (FC)	12
2.6 Huấn luyện mô hình CNN	12
2.7 Tìm hiểu về Multi-task Cascaded Convolutional Networks	15
2.7.1 Multi-task Cascaded Convolutional Networks là gì ?	15
2.7.2 Quy trình làm việc của MTCNN	15

3 Giới thiệu về mô hình FaceNet trong nhận diện khuôn mặt	19
3.1 Khái quát về FaceNet	19
3.2 Tóm tắt	22
3.3 Siamese network	24
3.4 Thuật toán FaceNet	25
3.4.1 Khái quát thuật toán	25
3.4.2 Triplet Loss	26
3.4.3 Lựa chọn Triplet	28
3.4.4 Mạng tích chập sâu	29
4 Thực nghiệm	32
4.1 Giới thiệu về thư viện DeepFace	32
4.2 Xây dựng ứng dụng Flask nhận diện khuôn mặt	32
4.2.1 Xây dựng ứng dụng Flask so sánh 2 người	32
4.2.2 Xây dựng ứng dụng Flask nhận diện 1 trong 9 người	34
4.3 Tính Accuracy và Precision của mô hình	37
KẾT LUẬN	39
Tài liệu tham khảo	40
A Liệt kê mã nguồn	42
A.1 Mã nguồn xây dựng ứng dụng Flask so sánh 2 người	42
A.2 Mã nguồn xây dựng ứng dụng Flask nhận diện 1 trong 9 người	43
A.3 Mã nguồn xây dựng model FaceNet	44

Danh sách hình vẽ

1.1	Quy trình xử lý ảnh	2
1.2	Các bước cơ bản trong một hệ thống xử lý ảnh	2
1.3	Hệ thống nhận dạng khuôn mặt	5
2.1	Mô hình CNN [2]	9
2.2	Phép tính Convolution	10
2.3	Mô tả hàm MaxPooling với cửa sổ 2x2 mà bước trượt bằng 2	12
2.4	Cấu trúc MTCNN [3]	15
2.5	Xử lý MTCNN [3]	16
2.6	P-Net [4]	17
2.7	R-Net (bên trái) và O-Net (bên phải) [4]	18
3.1	FaceNet lấy hình ảnh khuôn mặt làm đầu vào và xuất vector nhúng [5] . .	19
3.2	Hình ảnh khuôn mặt được vẽ ở dạng 2D [5]	20
3.3	Nhận dạng người bằng cách tính khoảng cách nhúng [5]	20
3.4	Trạng thái ban đầu trước khi huấn luyện [5]	21
3.5	Quá trình huấn luyện FaceNet [5]	22
3.6	Hình minh họa khoảng cách đầu ra khi sử dụng FaceNet giữa các cặp khuôn mặt. [6]	23
3.7	Tóm tắt quy trình nhận dạng khuôn mặt sử dụng FaceNet [7]	23
3.8	Siamese network [8]	24
3.9	Cấu trúc mô hình FaceNet [6]	26
3.10	Triplet Loss [6]	26
3.11	Mô phỏng Triplet Loss [8]	26
3.12	Nhúng FaceNet thành vector 128 chiều [6]	30
3.13	Module Inception dạng nguyên thuỷ (ảnh trái) và dạng giảm chiều (ảnh phải) [6]	31
3.14	Cấu trúc mạng Inception [6]	31
4.1	Trang chủ.	33
4.2	Hai ảnh đầu vào cùng người.	33
4.3	Kết quả 1.	33
4.4	Hai ảnh đầu vào khác người.	34
4.5	Kết quả 2.	34
4.6	Trang chủ.	35
4.7	Ba ảnh đầu vào.	35
4.8	Kết quả 1.	36

4.9	Kết quả 2.	36
4.10	Kết quả 3.	37
4.11	Accuracy và Precision ở tập train và tập test.	38

Danh sách tên viết tắt

AI	Artificial Intelligence
IoT	Internet of Things
CNN	Convolutional Neural Network
k-NN	k-Nearest Neighbors
SVM	Support Vector Machine
ReLU	Rectified Linear Unit
DCNN	Deep Convolution Neural Network
MTCNN	Multi-task Cascaded Convolutional Neural Networks
SGD	Stochastic Gradient Descent
VIP	Very Important Person

MỞ ĐẦU

Học sâu đã và đang rất phát triển, được ứng dụng rộng rãi trong các bài toán nhận dạng như: nhận dạng hình ảnh, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên ... và thu được những thành tựu to lớn với độ chính xác ngày càng cao. Trong đó nhận dạng khuôn mặt để xác định danh tính, giao dịch, kiểm soát an ninh ... ngày càng trở nên phổ biến.

Xuất phát từ đó, bài báo cáo này trình bày về một số thuật toán cũng như hướng phát triển cho đề tài "Nghiên cứu Deep Learning để tính toán các đặc trưng của mặt người". Nội dung của bài báo cáo này gồm 4 chương:

- Chương 1: Tổng quan về nhận dạng khuôn mặt

Nội dung chính của chương 1 là tìm hiểu khái quát về xử lý ảnh và bài toán nhận dạng khuôn mặt.

- Chương 2: Tìm hiểu về học sâu và mô hình mạng neural tích chập

Nội dung chính của chương 2 là tìm hiểu tổng quan về học máy, Học sâu, mô hình mạng neural tích chập (CNN) cũng như cách hoạt động, cấu trúc và việc huấn luyện của mô hình mạng neural tích chập.

- Chương 3: Giới thiệu về mô hình FaceNet trong nhận diện khuôn mặt

Nội dung chính của chương 3 là trình bày Siamese network và các thuật toán của FaceNet.

- Chương 4: Thực nghiệm

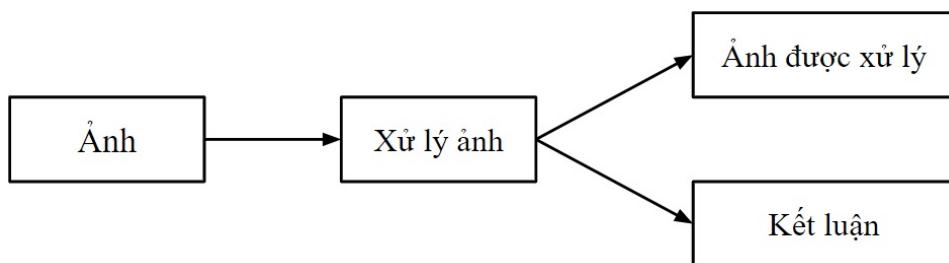
Thư viện DeepFace và mô hình có sẵn FaceNet sẽ được áp dụng để xây dựng các ứng dụng Flask trong bài báo cáo này. Bên cạnh đó chương này sẽ trình bày về độ chính xác và độ chụm của mô hình.

Chương 1 Tổng quan về nhận dạng khuôn mặt

1.1 Tổng quan về xử lý ảnh

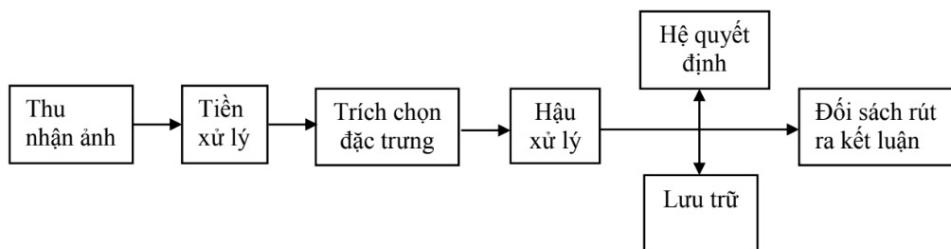
1.1.1 Một số khái niệm

Xử lý ảnh là từ một ảnh đầu vào qua quá trình xử lý (thông qua các thuật toán) ta thu được một ảnh đã được xử lý hoặc một kết luận.



Hình 1.1: Quy trình xử lý ảnh.

Thông thường các ảnh tự nhiên, ảnh chụp có các tín hiệu ảnh đặc trưng bởi 2 đại lượng là biên độ và dải tần số. Nó chính là các đối tượng của xử lý ảnh.



Hình 1.2: Các bước cơ bản trong một hệ thống xử lý ảnh.

1.1.2 Các vấn đề của xử lý ảnh

a) Điều chỉnh mức xám của ảnh

Có 2 hướng chính là tăng số mức xám hoặc giảm số mức xám với mục đích chính là tăng cường độ mịn cho ảnh hoặc in ảnh màu ra máy in đen trắng.

b) Trích chọn đặc điểm

Tùy theo mục đích nhận dạng trong quá trình xử lý ảnh mà các đối tượng được trích trọng. Một số đặc điểm của ảnh như đặc điểm biến đổi (dựa vào lọc vùng), đặc điểm không gian (điểm uốn, phân bố xác suất, biên độ, mức xám ...) hay đặc điểm biên và đường biên.

c) Nhận dạng

Hệ thống nhận dạng tự động bao gồm ba khâu tương ứng với ba giai đoạn chủ yếu sau đây:

- Bước 1: Thu nhận dữ liệu và tiền xử lý.
- Bước 2: Biểu diễn dữ liệu.
- Bước 3: Nhận dạng, ra quyết định.

Bốn cách tiếp cận khác nhau trong lý thuyết nhận dạng là:

- Đối sánh mẫu dựa trên các đặc trưng được trích chọn
- Phân loại thống kê
- Đối sánh cấu trúc
- Phân loại dựa trên mạng neural

1.1.3 *Ứng dụng của xử lý ảnh trong thực tế*

- *Xử lý và phục hồi hình ảnh*: Ứng dụng này tương tự như photoshop: từ một hình ảnh được chụp từ máy ảnh, ta có thể chỉnh sửa, xử lý để làm ảnh đẹp hơn hoặc phù hợp nhu cầu người dùng như: làm mờ, lấy biên, chỉnh độ nét, chỉnh độ phân giải, phục hồi và nhận dạng ảnh...

- *Truyền và mã hóa*: Ngày nay con người sử dụng internet để truyền nhận các ảnh, video một cách nhanh chóng. Hình ảnh khi ta chụp sẽ được mã hóa và truyền theo internet. Rất nhanh sau vài giây là người bạn có thể nhận được một bức ảnh.

- *Thị giác máy tính và robot*: Hiện tại công nghệ robot đang phát triển nhanh chóng, và càng ngày càng giống con người hơn. Thị giác của máy tính cũng là một phần quan

trọng. Làm thế nào để robot có thể nhìn mọi thứ, tránh vật cản, nhận dạng các vật...? Đó chính là nhờ một hệ thống quá trình xử lý ảnh phức tạp.

- *Phát hiện vật cản*: Phát hiện vật cản cũng là một lĩnh vực mới và được thực hiện bởi xử lý ảnh: tính toán khoảng cách từ robot tới vật cản bằng cách xác định được các đối tượng khác nhau trong hình ảnh sau đó xử lý và tính toán chúng.

- *Công nghệ nhận dạng*: Xử lý ảnh dùng để xác định, nhận dạng các đối tượng, các mối nguy hiểm, nhận dạng vân tay, khuôn mặt, hoặc các loại bệnh trong lĩnh vực y tế.

1.2 Bài toán nhận dạng khuôn mặt

1.2.1 Khái niệm

Nhận dạng khuôn mặt là một loại phần mềm sinh trắc học ánh xạ các đặc điểm khuôn mặt của một cá nhân về mặt toán học và lưu trữ dữ liệu dưới dạng dấu khuôn mặt (faceprint). Công nghệ AI nhận dạng khuôn mặt là phần mềm sử dụng các thuật toán Học sâu để so sánh ảnh chụp trực tiếp hoặc hình ảnh kỹ thuật số với ảnh được lưu trữ trong cơ sở dữ liệu để xác minh danh tính của một cá nhân.

1.2.2 Một số trở ngại của công nghệ nhận dạng khuôn mặt

- Góc chụp khuôn mặt: Chụp thẳng, chụp nghiêng, chụp hất lên ... làm cho các thành phần trên khuôn mặt như mắt, mũi, miệng có thể bị khuất một phần hoặc thậm chí khuất hết, đều là những khó khăn rất lớn trong bài toán nhận dạng mặt người.

- Một số thành phần xuất hiện thêm hoặc không xuất hiện trên khuôn mặt như: đeo kính, đeo khẩu trang, trang điểm, mọc râu ... làm cho việc nhận dạng khuôn mặt thiếu chính xác.

- Khi con người thể hiện sự biểu cảm như: cười, khóc, nhăn mặt ... cũng ảnh hưởng đến kết quả nhận dạng.

- Ngoài ra một số tác nhân khác cũng gây ảnh hưởng đến kết quả như: Ánh sáng quá sáng, quá mờ, chất lượng ảnh ...

- Nền ảnh phức tạp: Nền của ảnh phức tạp là một trong những khó khăn nhất trong bài toán nhận dạng khuôn mặt người trong ảnh, khuôn mặt người sẽ dễ bị nhầm lẫn với

nhiều khung cảnh phức tạp xung quanh và ảnh hưởng rất nhiều đến quá trình phân tích và rút trích các đặc trưng của khuôn mặt trong ảnh, có thể dẫn đến không nhận ra khuôn mặt hoặc là nhận nhầm các khung cảnh xung quanh thành khuôn mặt người.

- Màu sắc của da mặt: Màu sắc của da mặt cũng đóng vai trò quan trọng trong nhận dạng khuôn mặt. Nếu màu sắc của da người quá tối hoặc gần với màu sắc của khung cảnh môi trường thì thuật toán sẽ gặp khó khăn trong việc nhận dạng các đặc trưng và có thể không tìm ra được khuôn mặt người.

1.2.3 Các ứng dụng đặc trưng của bài toán nhận dạng khuôn mặt

- Ứng dụng trong trường học: điểm danh khuôn mặt, đăng ký, kiểm soát an ninh các khu vực cần theo dõi... là các giải pháp nhận dạng khuôn mặt cho trường học.

- Ứng dụng trong các thiết bị IoT: thiết bị kiểm soát ra vào bằng khuôn mặt, thiết bị đọc giấy tờ tùy thân...

- Ứng dụng trong doanh nghiệp, công sở: chấm công khuôn mặt, bảo mật máy tính, quản lý ra/vào.

- Ứng dụng trong giám sát an ninh: các giải pháp kiểm soát an ninh (kiểm soát ra vào), nhận dạng khách lạ, khách VIP và đối tượng trong danh sách đen, tình nghi xuất hiện trong khu vực giám sát...

1.2.4 Xây dựng hệ thống nhận dạng khuôn mặt

Một hệ thống nhận dạng khuôn mặt có thể khái quát chung gồm có 3 bước cơ bản sau:



Hình 1.3: Hệ thống nhận dạng khuôn mặt.

- Phát hiện khuôn mặt: Hệ thống nhận vào một ảnh tĩnh (từ bức hình hay một đoạn video), sau đó có thể xử lý ảnh cho chất lượng tốt hơn, như chỉnh lại độ sáng, giảm độ nhiễu ...

- Trích rút đặc trưng: Chính là việc phân tích và rút ra đặc điểm của khuôn mặt trong ảnh vì mỗi khuôn mặt có đặc điểm khác nhau (trừ các trường hợp sinh đôi cùng trứng).

- Nhận dạng khuôn mặt: Hệ thống sẽ so sánh các đặc điểm được trích rút với cơ sở dữ liệu khuôn mặt và sẽ quyết định kết quả so sánh có phù hợp hay không.

1.2.5 Một số phương pháp nhận dạng khuôn mặt

Dựa vào các tiêu chí mà người ta chia ra thành nhiều phương pháp nhận dạng khuôn mặt nhưng phổ biến hiện nay là các loại sau:

- Phương pháp tiếp cận toàn cục
- Phương pháp tiếp cận dựa trên các đặc điểm cục bộ
- Phương pháp lai

Khi làm việc trong điều kiện không có kiểm soát, phương pháp tiếp cận dựa trên các đặc điểm cục bộ (trích chọn đặc trưng) tỏ ra thích hợp hơn hai phương pháp kia. Đó chính là các hệ thống phát hiện khuôn mặt người dựa trên tính năng (feature based).

Người ta cũng có thể chia thành hai hướng nhận dạng như làm với dữ liệu video (từ camera) hoặc làm với dữ liệu ảnh.

Chương 2 Tìm hiểu về học sâu và mô hình mạng neural tích chập

2.1 Tổng quan về Học máy (Machine learning)

Học máy là một công nghệ phát triển từ lĩnh vực trí tuệ nhân tạo. Các thuật toán Học máy là các chương trình máy tính có khả năng học hỏi về cách hoàn thành các nhiệm vụ và cách cải thiện hiệu suất theo thời gian.

Học máy ra đời làm giảm bớt những hạn chế vốn có của AI khi nó mang lại cho máy tính khả năng có thể tìm ra mọi thứ mà không được lập trình rõ ràng. Học máy vẫn đòi hỏi sự đánh giá của con người trong việc tìm hiểu dữ liệu cơ sở và lựa chọn các kĩ thuật phù hợp để phân tích dữ liệu. Đồng thời, trước khi sử dụng, dữ liệu phải sạch, không có sai lệch và không có dữ liệu giả.

Các mô hình Học máy yêu cầu lượng dữ liệu đủ lớn để "huấn luyện" và đánh giá mô hình. Trước đây, các thuật toán Học máy thiếu quyền truy cập vào một lượng lớn dữ liệu cần thiết để mô hình hóa các mối quan hệ giữa các dữ liệu. Sự tăng trưởng trong dữ liệu lớn (big data) đã cung cấp các thuật toán Học máy với đủ dữ liệu để cải thiện độ chính xác của mô hình và dự đoán.

2.2 Các thuật toán Học máy

2.2.1 Học có giám sát (Supervised learning)

Trong học có giám sát, máy tính học cách mô hình hóa các mối quan hệ dựa trên dữ liệu được gán nhãn (labeled data). Sau khi tìm hiểu cách tốt nhất để mô hình hóa các mối quan hệ cho dữ liệu được gán nhãn, các thuật toán được huấn luyện được sử dụng cho các bộ dữ liệu mới.

2.2.2 Học không giám sát (Unsupervised learning)

Trong học không giám sát, máy tính không được cung cấp dữ liệu được dán nhãn mà thay vào đó chỉ được cung cấp dữ liệu mà thuật toán tìm cách mô tả dữ liệu và cấu trúc của chúng.

2.3 Tìm hiểu về Học sâu (Deep learning)

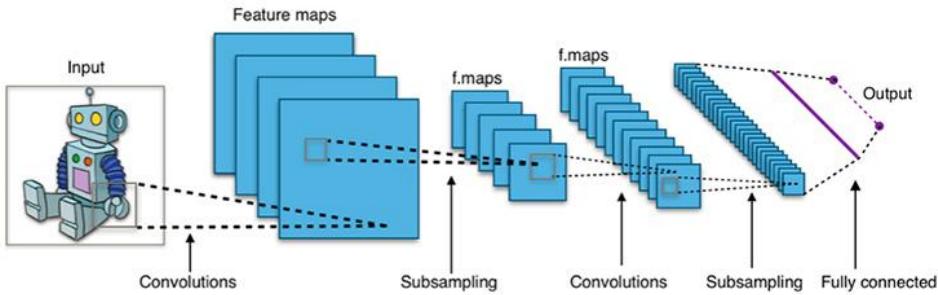
Học sâu là một tập hợp con của Học máy và đề cập đến các mạng thần kinh có khả năng học dữ liệu đầu vào với các biểu diễn trừu tượng. Học sâu vượt trội hơn so với học máy truyền thống trong các vấn đề phức tạp như nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên, phân loại hình ảnh ...

Học sâu khác với Học máy là thuật toán học sâu cần được đào tạo trong một thời gian dài vì có một số lượng lớn các tham số trong khi các thuật toán Học máy truyền thống có thể được đào tạo chỉ trong vòng vài giờ.

Học sâu đóng vai trò ngày càng lớn trong công nghệ hiện đại. Về mặt lý thuyết, nó có thể vượt quá khả năng của bộ não con người, vì chúng có các thuật toán nền tảng rộng lớn, được gọi là mạng lưới thần kinh. Ngày nay, các kỹ thuật học sâu có thể được tìm thấy ở một mật độ khá cao, từ xe tự lái đến các nghiên cứu học thuật.

2.4 Tìm hiểu về CNN (Convolutional Neural Network)

Mạng neural tích chập (CNN) là một mô hình học sâu dựa trên phép tích chập của ảnh để tách các điểm dựa vào bộ lọc thu được trong quá trình huấn luyện. CNN có thể xây dựng được các hệ thống phân loại với độ chính xác cao. Ý tưởng của CNN được lấy cảm hứng từ khả năng nhận biết thị giác của bộ não người. Để có thể nhận biết được các hình ảnh trong vỏ não người có hai loại tế bào là tế bào đơn giản và tế bào phức tạp (Hubel & Wiesel, 1968). Các tế bào đơn giản phản ứng với các mẫu hình dạng cơ bản ở các vùng kích thích thị giác và các tế bào phức tạp tổng hợp thông tin từ các tế bào đơn giản để xác định các mẫu hình dạng phức tạp hơn. Khả năng nhận biết các hình ảnh của não người là một hệ thống xử lý hình ảnh tự nhiên mạnh mẽ và tự nhiên nên CNN được phát triển dựa trên ba ý tưởng chính: tính kết nối cục bộ (Local connectivity hay compositionality), tính bất biến (Location invariance) và tính bất biến đối với quá trình chuyển đổi cục bộ (Invariance to local transition) (LeCun et al., 2015). CNN là một dạng mạng neural chuyên dụng để xử lý các dữ liệu dạng lưới 1 chiều như dữ liệu âm thanh, dữ liệu MGE hoặc nhiều chiều như dữ liệu hình ảnh, video. [1]



Hình 2.1: Mô hình CNN. [2]

Cấu trúc cơ bản của CNN gồm các lớp tích chập (Convolution layer), lớp phi tuyến (Nonlinear layer) và lớp lọc (Pooling layer). Các lớp tích chập kết hợp với các lớp phi tuyến sử dụng các hàm phi tuyến như ReLU ($\text{ReLU}(x)=\max(x,0)$) hay TanH để tạo ra thông tin trừu tượng hơn (Abstract/higher-level) cho các lớp tiếp theo.

Các lớp liên kết trong CNN được với nhau thông qua cơ chế tích chập. Lớp tiếp theo là kết quả tích chập từ lớp trước đó vì vậy CNN có được các kết nối cục bộ vì mỗi neural ở lớp tiếp theo sinh ra từ một bộ lọc được áp đặt lên một vùng cục bộ của lớp trước đó. Nguyên tắc này được gọi là kết nối cục bộ (Local connectivity). Mỗi lớp như vậy được áp đặt các bộ lọc khác nhau. Một số lớp khác như lớp pooling/subsampling dùng để lọc lại các thông tin hữu ích hơn bằng cách loại bỏ các thông tin nhiễu. Trong suốt quá trình huấn luyện, CNN sẽ tự động học các tham số cho các lớp. Lớp cuối cùng được gọi là lớp kết nối đầy đủ (Fully connect layer) dùng để phân lớp.

2.5 Cấu trúc của CNN

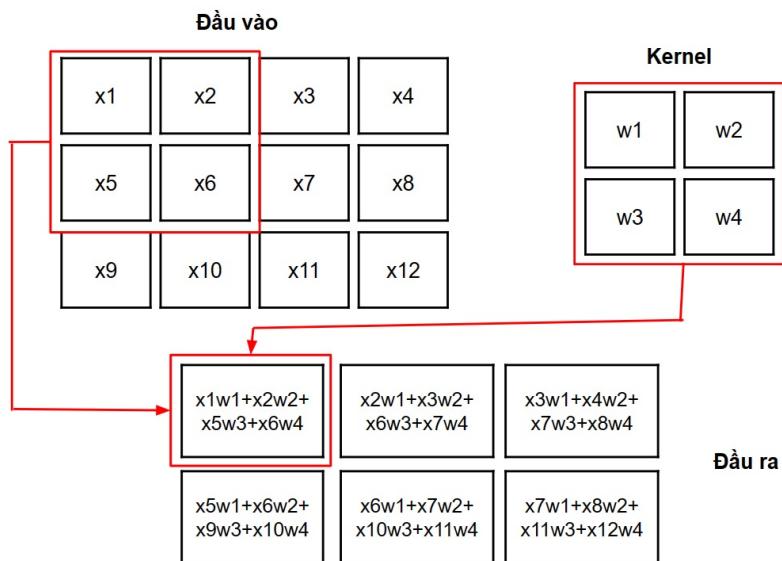
CNN là một kiểu mạng truyền thẳng, trong đó kiến trúc chính gồm nhiều thành phần được ghép nối với nhau theo cấu trúc nhiều lớp đó là: Convolution, Pooling, ReLU và Fully connected.

2.5.1 Lớp tích chập (Convolution)

Lớp tích chập (Convolution) là lớp quan trọng nhất trong cấu trúc của CNN. Tích chập được sử dụng đầu tiên trong xử lý tín hiệu số. Nhờ vào nguyên lý biến đổi thông tin có thể áp dụng kỹ thuật này vào xử lý ảnh và video số. Trong lớp Convolution sử dụng một bộ các bộ lọc có kích thước nhỏ hơn với ma trận đầu vào và áp lên một vùng của ma trận

đầu vào và tiến hành tính tích chập giữa bộ filter và giá trị của ma trận trong vùng cục bộ đó. Các filter sẽ dịch chuyển một bước trượt (Stride) chạy dọc theo ma trận đầu vào và quét toàn bộ ma trận. Trọng số của filter ban đầu sẽ được khởi tạo ngẫu nhiên và sẽ được học dần trong quá trình huấn luyện mô hình.

Hình 2.2 mô tả lý thuyết và cách thức Convolution hoạt động trên một dữ liệu đầu vào được biểu diễn bằng một ma trận hai chiều. Ta có thể hình dung phép tính này được thực hiện bằng cách dịch chuyển một cửa sổ mà ta gọi là kernel trên ma trận đầu vào, trong đó kết quả mỗi lần dịch chuyển được tính bằng tổng tích chập (tích của các giá trị giữa 2 ma trận tại vị trí tương ứng). Khi được áp dụng phép tính Convolution vào xử lý ảnh người ta thấy rằng lớp Convolution sẽ giúp biến đổi các thông tin đầu vào thành các yếu tố đặc trưng (nó tương ứng như bộ phát hiện – detector các đặc trưng về cạnh, hướng, đốm màu ...). Hình 2.2 là minh họa việc áp dụng phép tính Convolution trên ảnh trong đó đầu vào là kết quả biến đổi hình ảnh khi thực hiện phép tích chập khác nhau cho ra kết quả khác nhau, đầu ra là trực quan hóa các kernel dùng để detect các đặc trưng về cạnh, hướng, đốm màu.



Hình 2.2: Phép tính Convolution.

2.5.2 Lớp phi tuyến ReLU

Về cơ bản, Convolution là một phép biến đổi tuyến tính. Nếu tất cả các neural được tổng hợp bởi các phép biến đổi tuyến tính thì một mạng neural đều có thể đưa về dưới dạng một hàm tuyến tính. Khi đó mạng CNN sẽ đưa các bài toán về Logistic Regression. Do đó tại mỗi neural cần có một hàm truyền dưới dạng phi tuyến. Có nhiều dạng hàm phi tuyến được sử dụng trong quá trình này. Tuy nhiên, các nghiên cứu gần đây chứng minh được việc sử dụng hàm ReLu (Rectified Linear Unit) cho kết quả tốt hơn ở các khía cạnh:

- + Tính toán đơn giản
- + Tạo ra tính thưa (sparsity) ở các neural ẩn. Ví dụ như sau bước khởi tạo ngẫu nhiên các trọng số, khoảng 50% các neural ẩn được kích hoạt (có giá trị lớn hơn 0)
- + Quá trình huấn luyện nhanh hơn ngay cả khi không phải trải qua bước tiền huấn luyện

Như vậy tầng ReLu cơ bản chỉ đơn giản là áp dụng hàm truyền ReLu.

2.5.3 Lớp Pooling

Xét về mặt toán học pooling thực chất là quá trình tính toán trên ma trận trong đó mục tiêu sau khi tính toán là giảm kích thước ma trận nhưng vẫn làm nổi bật lên được đặc trưng có trong ma trận đầu vào.

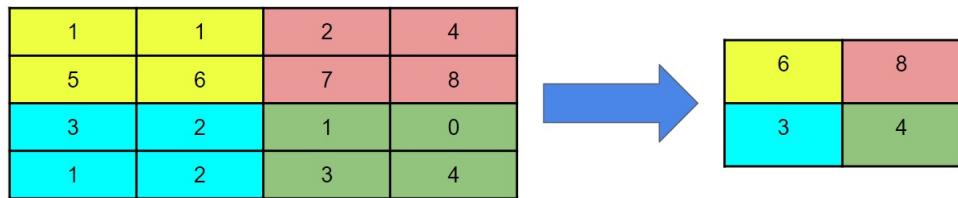
Có nhiều toán tử pooling như Sum-Pooling, Max-Pooling, L2-Pooling nhưng Max-Pooling thường được sử dụng. Về mặt ý nghĩa, Max-Pooling xác định vị trí cho tín hiệu mạnh nhất khi áp dụng một loại filter. Điều này cũng tương tự như là một bộ lọc phát hiện ví trí đối tượng bằng filter trong bài toán phát hiện đối tượng trong ảnh.

Về mặt lý thuyết với ma trận đầu vào có kích thước $W_1 \times H_1 \times D_1$ và thực hiện toán tử pooling trên ma trận con của ma trận đầu vào có kích thước $F \times F$ với bước nhảy S pixel, ta được ma trận đầu ra $W_2 \times H_2 \times D_2$ đó:

$$- W_2 = (W_1 - F)/S + 1$$

$$- H2 = (H1 - F)/S+1$$

$$- D2 = D1$$



Hình 2.3: Mô tả hàm Max-Pooling với cửa sổ 2×2 và bước trượt bằng 2.

Công dụng của lớp Pooling dùng để giảm kích thước dữ liệu, các tầng trong CNN chồng lên nhau có lớp Pooling ở cuối mỗi tầng giúp cho kích thước dữ liệu được co lại nhưng vẫn giữ được các đặc trưng để lấy mẫu. Ngoài ra giảm kích thước dữ liệu sẽ giảm số lượng tham số của mạng làm tăng tính hiệu quả và kiểm soát sự quá khứp dữ liệu (Overfitting).

2.5.4 Lớp Fully-connected (FC)

Lớp kết nối đầy đủ là một lớp giống như mạng neural truyền thẳng các giá trị được tính toán từ các lớp trước sẽ được liên kết đầy đủ vào trong neural của lớp tiếp theo. Tại lớp kết nối đầy đủ sẽ tiến hành phân lớp dữ liệu bằng cách kích hoạt hàm Softmax để tính xác suất ở lớp đầu ra.

2.6 Huấn luyện mô hình CNN

Ta chọn ngẫu nhiên trọng số và giá trị bộ lọc ngay từ ban đầu, tiếp theo ta gán nhãn cho mỗi hình ảnh đi qua mô hình CNN, Tập các ảnh phải đủ lớn và đã được gán nhãn cho biết cụ thể ảnh đó là gì. CNN sử dụng thuật toán lan truyền ngược (Back-propagation) để huấn luyện mô hình.

Mô tả huấn luyện một mạng CNN như sau:

Quá trình lan truyền thẳng: Truyền một hình ảnh cần đào tạo có kích thước $32 \times 32 \times 3$ qua toàn bộ mạng. Các trọng số và bộ lọc trong trường hợp này đã được khởi tạo ngẫu

nhiên, do đó sẽ không có kết luận hợp lý ở đây, do vậy dẫn đến hàm mất mát được lan truyền ngược.

Hàm mất mát (Loss function): Loss function có thể được định nghĩa theo nhiều cách khác nhau nhưng bài báo cáo này sẽ trình bày hàm mất mát cho Softmax Regression:

$$J(W, X, Y) = - \sum_{i=1}^N \sum_{j=1}^C y_{ji} \log(a_{ij}) = - \sum_{i=1}^N \sum_{j=1}^C y_{ji} \log\left(\frac{\exp(W_j^T x_i)}{\sum_{k=1}^C \exp(W_k^T x_i)}\right)$$

Giả sử biến L tương đương với đại lượng trên, chúng ta dễ dàng nhận thấy giá trị của hàm loss function sẽ là rất cao với một số hình ảnh đào tạo đầu tiên. Mục tiêu của mô hình là nhãn dự đoán giống nhãn đào tạo, để làm được điều này ta cần phải giảm thiểu giá trị loss function mà mô hình đang gặp phải. Để giải quyết vấn đề này ta cần tìm ra trọng số nào (weight) làm ảnh hưởng đến loss function của mô hình, tiếp theo ta cần thực hiện lan truyền ngược đi qua mô hình để xử lý vấn đề trên.

Lan truyền ngược (Back-propagation): Xác định trọng số ảnh hưởng nhất cho loss function và tìm cách để làm tối thiểu loss function, một khi chúng ta thực hiện được công việc này chúng ta sẽ đi đến bước cuối cùng là cập nhật trọng số. Để làm tối thiểu loss function ta phải tối ưu hàm mất mát.

Tối ưu hàm mất mát: Ta sử dụng Stochastic Gradient Descent (SGD) ở đây.

Với chỉ một cặp dữ liệu (x_i, y_i) , ta có:

$$\begin{aligned} J_i(W) &= J(W, x_i, y_i) = - \sum_{i=1}^N \sum_{j=1}^C y_{ji} \log\left(\frac{\exp(W_j^T x_i)}{\sum_{k=1}^C \exp(W_k^T x_i)}\right) \\ &= - \sum_{i=1}^N (y_{ji} W_j^T x_i - y_{ji} \log(\sum_{k=1}^C \exp(W_k^T x_i))) \\ &= - \sum_{i=1}^N y_{ji} W_j^T x_i + y_{ji} \log(\sum_{k=1}^C \exp(W_k^T x_i)) \end{aligned} \quad (2.1)$$

Trong biến đổi ở dòng cuối cùng, chúng ta đã sử dụng quan sát: $\sum_{k=1}^C y_{ki} = 1$ vì nó là tổng các xác suất.

Công thức tiếp theo ta sử dụng là:

$$\frac{\partial J_i(W)}{\partial W} = \left[\frac{\partial J_i(W)}{\partial W_1}, \frac{\partial J_i(W)}{\partial W}, \dots, \frac{\partial J_i(W)}{\partial W_C} \right] \quad (2.2)$$

Trong đó, Gradient theo từng cột có thể tính được dựa theo công thức (2.1):

$$\begin{aligned} \frac{\partial J_i(W)}{\partial W_j} &= -y_{ji}x_i + \frac{\exp(W_j^T x_i)}{\sum_{k=1}^C \exp(W_k^T x_i)} x_i \\ &= -y_{ji}x_i + a_{ji}x_i = x_i(a_{ji} - y_{ji}) = e_{ji}x_i \end{aligned} \quad (2.3)$$

trong đó $e_{ji}x_i = a_{ji} - y_{ji}$ là sai số dự đoán

Từ (2.2), (2.3):

$$\frac{\partial J_i(W)}{\partial W} = x_i[e_{1i}, e_{2i}, \dots, e_{Ci}] = x_i e_i^T$$

Ta dễ dàng suy ra:

$$\frac{\partial J(W)}{\partial W} = \sum_{i=1}^N x_i e_i^T = XE^T$$

với $E = A - Y$. Công thức tính gradient đơn giản thế này giúp cho cả Batch Gradient Descent, Stochastic Gradient Descent (SGD), và Mini-batch Gradient Descent đều có thể dễ dàng được áp dụng.

Giả sử rằng chúng ta sử dụng SGD, công thức cập nhật cho ma trận trọng số W sẽ là:

$$W = W_i + \eta x_i(y_i - a_i)^T$$

Trong đó:

- * W là trọng số lúc sau cập nhật
- * W_i là trọng số lúc đầu
- * η tỉ lệ học

Tỉ lệ học (Learning rate) là một tham số được khởi tạo bởi người lập trình. Tỉ lệ học quá cao dẫn đến bước nhảy lớn, như vậy khó xác định được các điểm tối ưu cần tìm.

Quá trình trên được lặp lại liên tục trong một thời gian nhất định cho mỗi tập hình ảnh đào tạo.

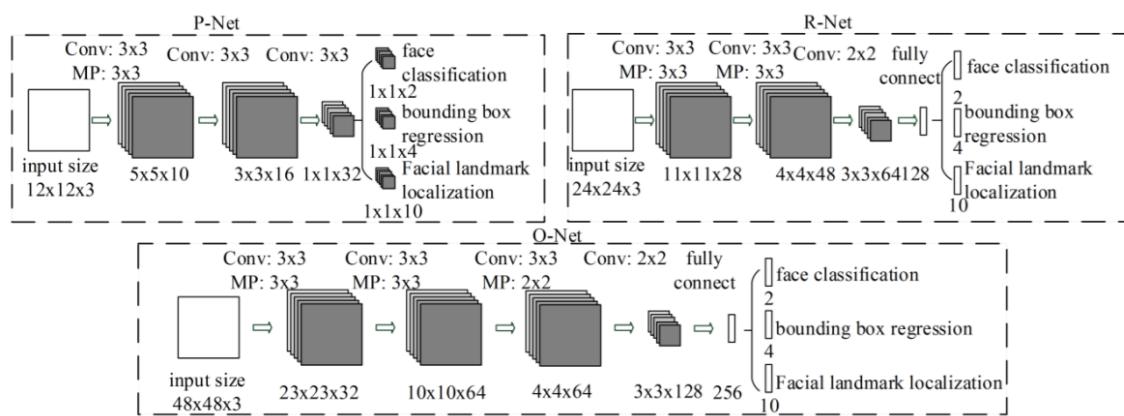
2.7 Tìm hiểu về Multi-task Cascaded Convolutional Networks

2.7.1 Multi-task Cascaded Convolutional Networks là gì ?

Multi-task Cascaded Convolutional Networks được viết tắt là MTCNN, là 1 mạng neural tích chập nhằm phát hiện khuôn mặt và facial landmark(vị trí mắt, mũi, miệng...). Nó bao gồm 3 mạng CNN xếp chồng và đồng thời hoạt động khi nhận diện khuôn mặt. Mỗi mạng có cấu trúc khác nhau và đảm nhiệm vai trò khác nhau trong việc phát hiện khuôn mặt. Kết quả dữ liệu đầu ra của MTCNN là vector đặc trưng biểu diễn cho vị trí khuôn mặt được xác định trong bức ảnh (mắt, mũi, miệng,...)

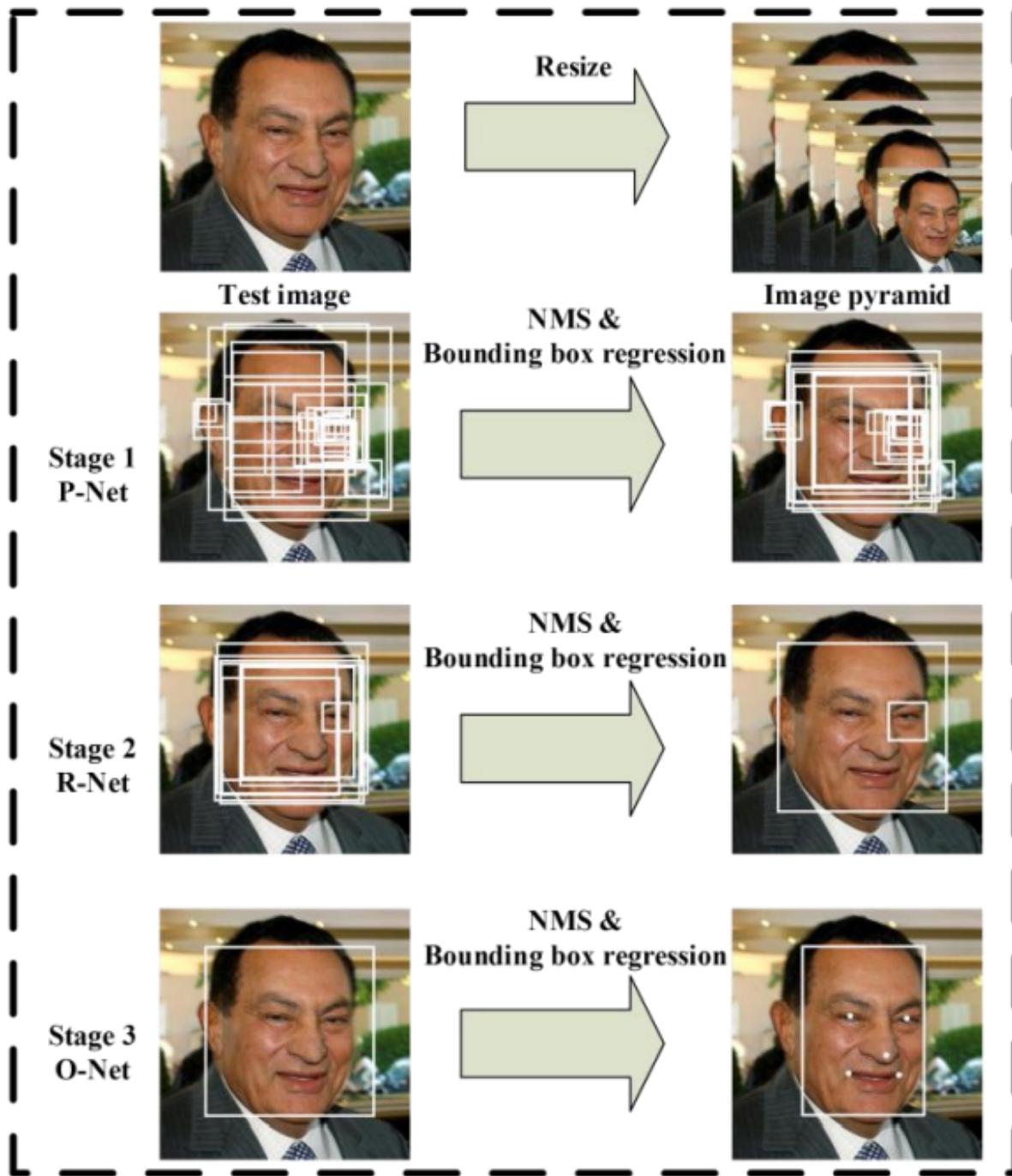
2.7.2 Quy trình làm việc của MTCNN

MTCNN hoạt động theo ba bước, mỗi bước dùng một mạng neural riêng lần lượt là: mạng đề xuất P-Net (Proposal Network) nhằm dự đoán các vùng trong ảnh ví dụ là vùng chứa khuôn mặt; mạng tinh chế R-Net (Refine Network) sử dụng đầu ra của P-Net để loại bỏ các vùng không phải khuôn mặt; và mạng đầu ra O-Net (Output Network): sử dụng đầu ra R-Net để đưa ra kết quả cuối cùng với 5 điểm đánh dấu khuôn mặt: 2 điểm mắt, 1 điểm mũi và 2 điểm khoé miệng (Hình 2.4).



Hình 2.4: Cấu trúc MTCNN. [3]

Hình 2.7 mô tả trực quan quy trình làm việc của MTCNN.



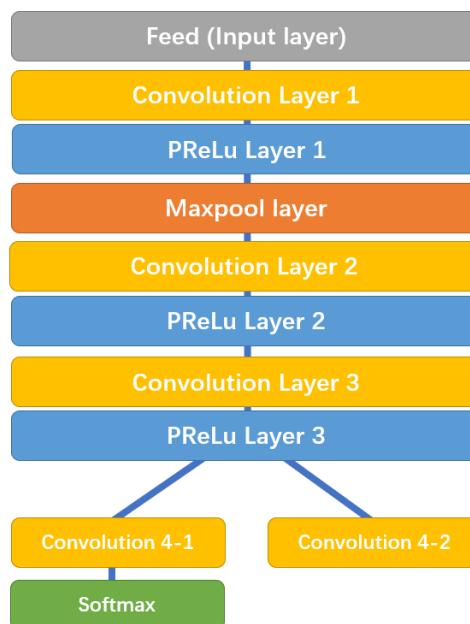
Hình 2.5: Xử lý MTCNN. [3]

Đối với mỗi hình ảnh truyền vào, mạng tạo ra một kim tự tháp hình ảnh nghĩa là nó sẽ tạo ra nhiều bản sao của hình ảnh đó ở các kích thước khác nhau. Mục đích của việc này là để phát hiện các khuôn mặt ở tất cả các kích thước khác nhau.

a. *Mạng P-Net*

Tại P-Net, đối với mỗi hình ảnh được chia tỉ lệ, hạt nhân (kernel) 12*12 chạy qua hình ảnh đó để tìm kiếm khuôn mặt.

Trong mỗi hạt nhân 12*12 này, có 3 cấu trúc được chạy qua với hạt nhân 3x3. Sau mỗi lớp convolution, một lớp PReLU được triển khai. Ngoài ra, một lớp max-pooling được đưa vào sau lớp PReLU đầu tiên (max-pooling lấy ra mọi pixel khác, chỉ để lại lớp lớn nhất trong vùng lân cận). Sau lớp convolution thứ ba, mạng chia thành 2 lớp. Các kích hoạt từ lớp thứ ba được chuyển đến hai lớp convolution riêng biệt và một lớp Softmax sau một trong các lớp convolution đó. Trong trường hợp này, nó tạo ra 2 xác suất: xác suất mà ở đó là một khuôn mặt trong khu vực và xác suất mà ở đó không phải là một gương mặt



Hình 2.6: P-Net. [4]

b. R-Net

R-Net có cấu trúc tương tự, nhưng có nhiều lớp hơn. Nó lấy đầu ra của P-Net làm đầu vào và đưa ra tọa độ hộp giới hạn chính xác hơn.

c. O-Net

Cuối cùng, O-Net lấy các đầu ra của R-Net làm đầu vào và đưa ra 3 bộ dữ liệu: xác suất của một mặt nằm trong hộp, tọa độ của hộp giới hạn và tọa độ của các mốc mặt (vị trí của mắt, mũi và miệng).

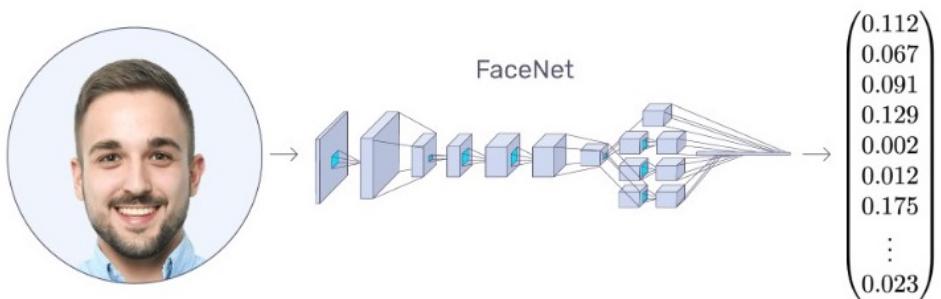


Hình 2.7: R-Net (bên trái) và O-Net (bên phải). [4]

Chương 3 Giới thiệu về mô hình FaceNet trong nhận diện khuôn mặt

3.1 Khái quát về FaceNet

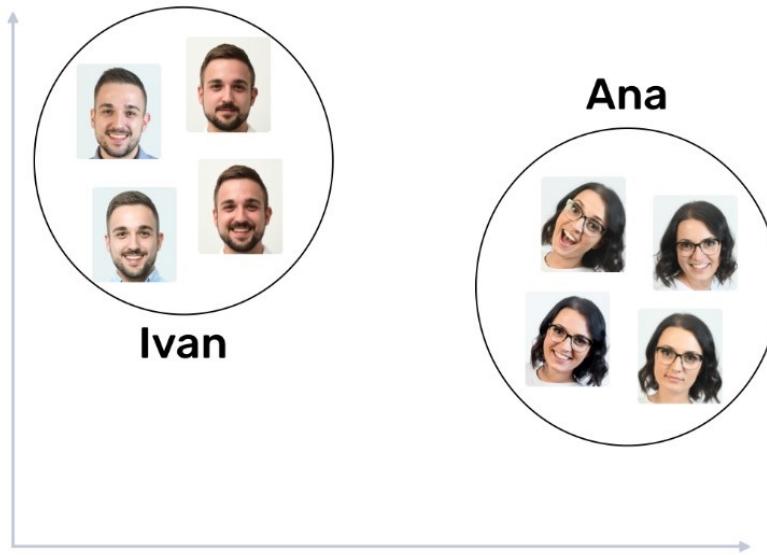
FaceNet là một mạng lưới thần kinh sâu được sử dụng để trích xuất các đặc điểm từ hình ảnh khuôn mặt của một người.



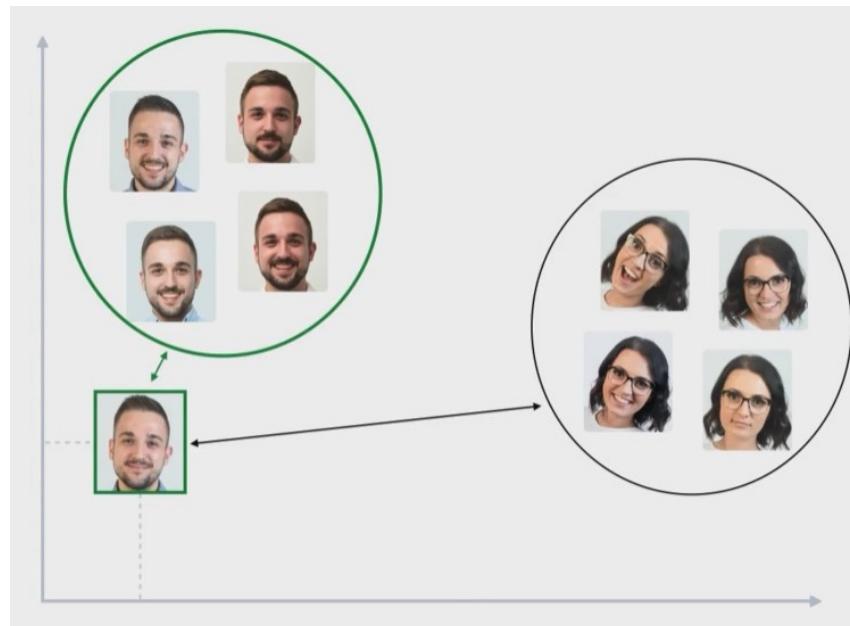
Hình 3.1: FaceNet lấy hình ảnh khuôn mặt làm đầu vào và xuất vector nhúng. [5]

FaceNet lấy hình ảnh khuôn mặt của người đó làm đầu vào và xuất ra một vector gồm 128 số đại diện cho các đặc điểm quan trọng nhất của khuôn mặt. Trong học máy, này được gọi là nhúng . Tại sao phải nhúng ? Bởi vì tất cả các thông tin quan trọng từ một hình ảnh được nhúng vào vector này. Về cơ bản, FaceNet lấy khuôn mặt của một người và nén nó vào một vector gồm 128 số. Lý tưởng nhất là nhúng các khuôn mặt tương tự cũng tương tự (Hình 3.1).

Một cách khả thi để nhận dạng một người trên một hình ảnh không nhìn thấy là tính toán phần nhúng của nó, tính toán khoảng cách đến hình ảnh của những người đã biết và nếu phần nhúng khuôn mặt đủ gần với phần nhúng của người A, chúng ta nói rằng hình ảnh này chứa khuôn mặt của người A (Hình 3.2).



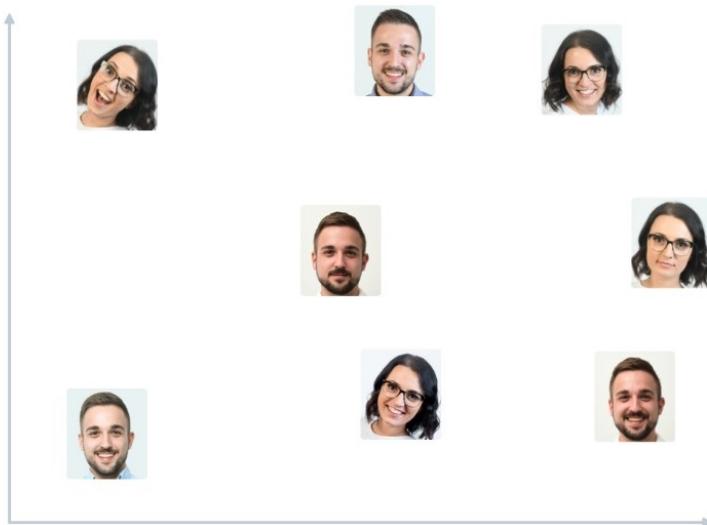
Hình 3.2: Hình ảnh khuôn mặt được vẽ ở dạng 2D. [5]



Hình 3.3: Nhận dạng người bằng cách tính khoảng cách nhúng. [5]

Để đào tạo FaceNet, cần rất nhiều hình ảnh của khuôn mặt. Để giữ cho mọi thứ đơn giản, chúng ta sẽ cho rằng chúng ta chỉ có một vài hình ảnh từ hai người. Logic tương tự có thể được áp dụng nếu chúng ta có hàng nghìn hình ảnh của những người khác nhau. Khi bắt đầu đào tạo, FaceNet tạo các vector ngẫu nhiên cho mọi hình ảnh, điều đó có nghĩa là các hình ảnh được phân tán ngẫu nhiên khi được vẽ (Hình 3.3).

FaceNet học theo cách sau:



Hình 3.4: Trạng thái ban đầu trước khi huấn luyện. [5]

Bước 1: Chọn ngẫu nhiên một hình ảnh anchor.

Bước 2: Chọn ngẫu nhiên một hình ảnh của cùng một người làm hình ảnh anchor (ví dụ positive).

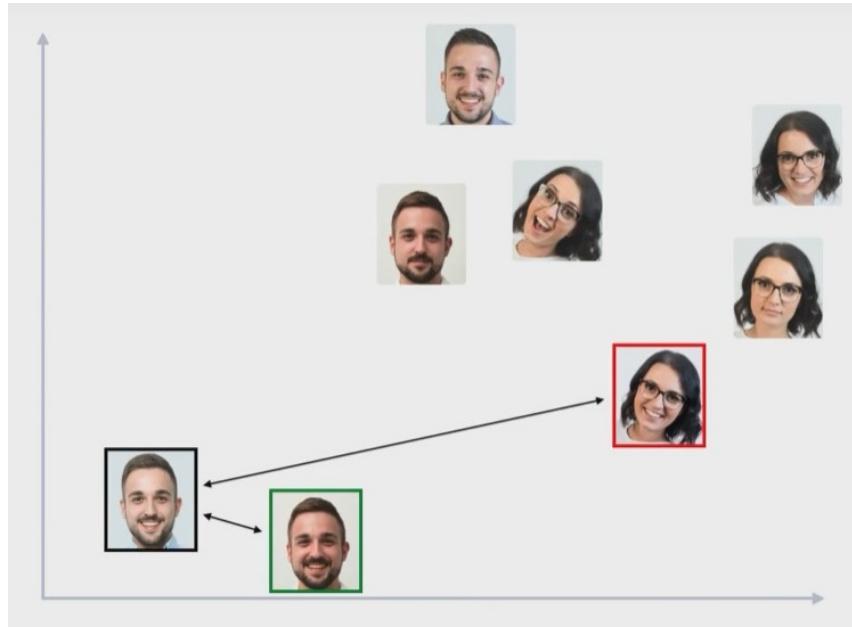
Bước 3: Chọn ngẫu nhiên một hình ảnh của một người khác với hình ảnh cố định (ví dụ negative).

Bước 4: Điều chỉnh các tham số mạng FaceNet sao cho ví dụ positive gần với anchor hơn ví dụ negative.

Lặp lại các bước này cho đến khi không còn thay đổi nào nữa được thực hiện. Tất cả các khuôn mặt của cùng một người ở gần nhau và ở xa những người khác (Hình 3.4, 3.5).

Phương pháp học này với các ví dụ anchor, positive và negative được gọi là triplet loss.

Vậy, những con số trong vector nhúng có nghĩa là gì ? Nhũng con số này có vẻ quan trọng đối với nhận dạng khuôn mặt, nhưng trên thực tế, chúng ta không thực sự biết những con số này đại diện cho điều gì và thực sự rất khó để diễn giải chúng.



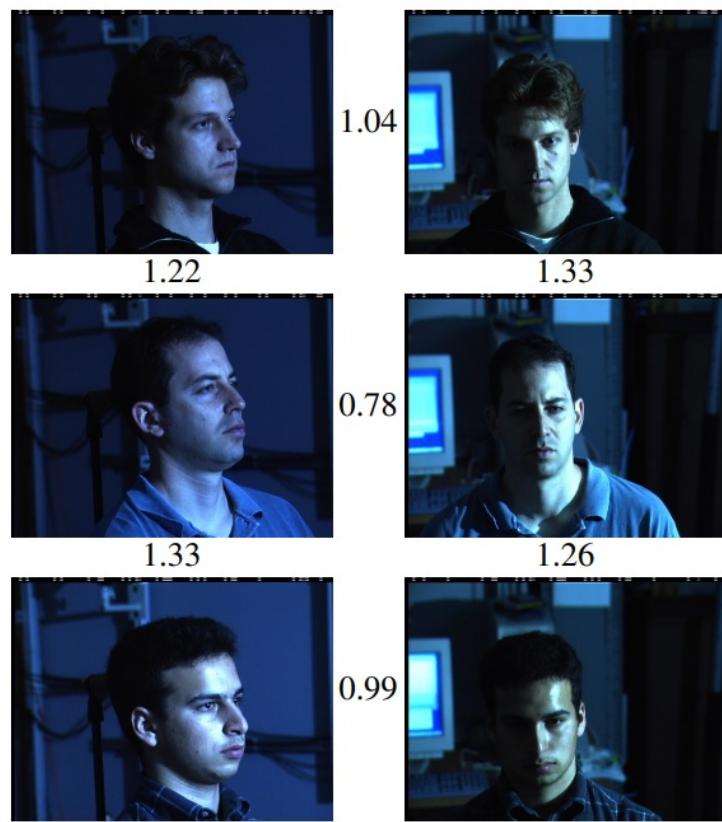
Hình 3.5: Quá trình huấn luyện FaceNet. [5]

3.2 Tóm tắt

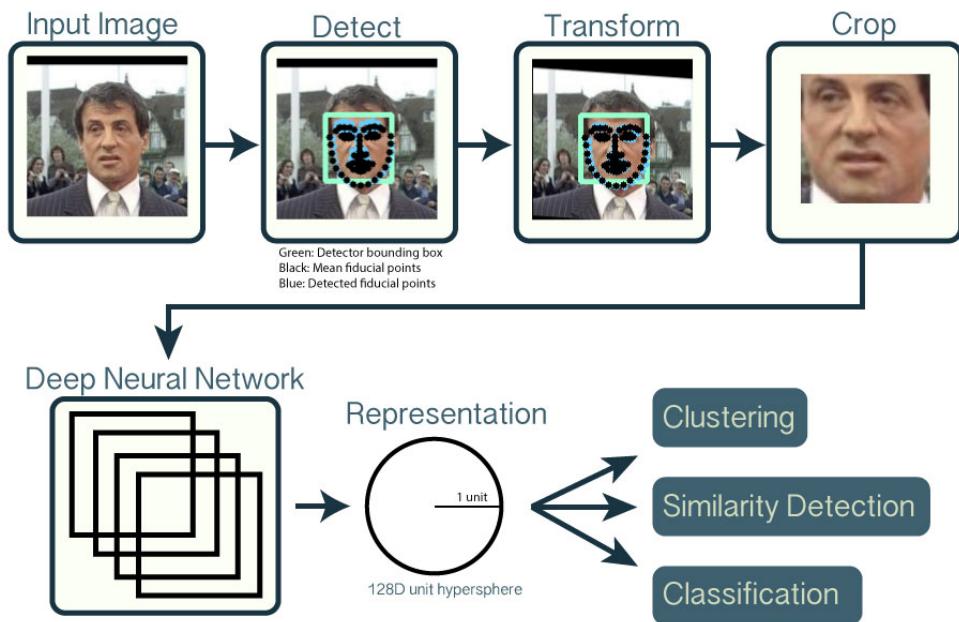
Nhóm tác giả [6] từ Google đề xuất một thuật toán có tên là FaceNet sẽ học cách ánh xạ từ ảnh vào khuôn mặt vào không gian chiết cosine với khoảng cách đo được tương ứng với độ tương đồng của khuôn mặt. Thuật toán này có thể tạo ra vector đặc trưng và nhúng vào bài toán nhận dạng khuôn mặt, kiểm tra khuôn mặt và phân cụm khuôn mặt. Nhóm tác giả đã sử dụng mạng neural tích chập sâu (Deep Convolutional Neural Network - DCNN) được huấn luyện để tự tối ưu hoá bài toán. Mạng được huấn luyện sao cho khoảng cách L_2 bình phương trong không gian nhúng tương ứng với mức độ tương đồng của khuôn mặt: Mặt cùng người sẽ có khoảng cách nhỏ, mặt khác người sẽ có khoảng cách lớn (Hình 3.6).

Sau khi thực hiện phép nhúng, thu được vector đặc trưng, ta có thể thực hiện được 3 bài toán (Hình 3.7):

- **Kiểm tra khuôn mặt**, ta chỉ cần phân ngưỡng khoảng cách giữa 2 vector đặc trưng của 2 khuôn mặt.
- **Nhận dạng khuôn mặt** là bài toán phân loại k-NN.
- **Phân cụm khuôn mặt** sử dụng k-mean.



Hình 3.6: Hình minh họa khoảng cách đầu ra khi sử dụng FaceNet giữa các cặp khuôn mặt. [6]

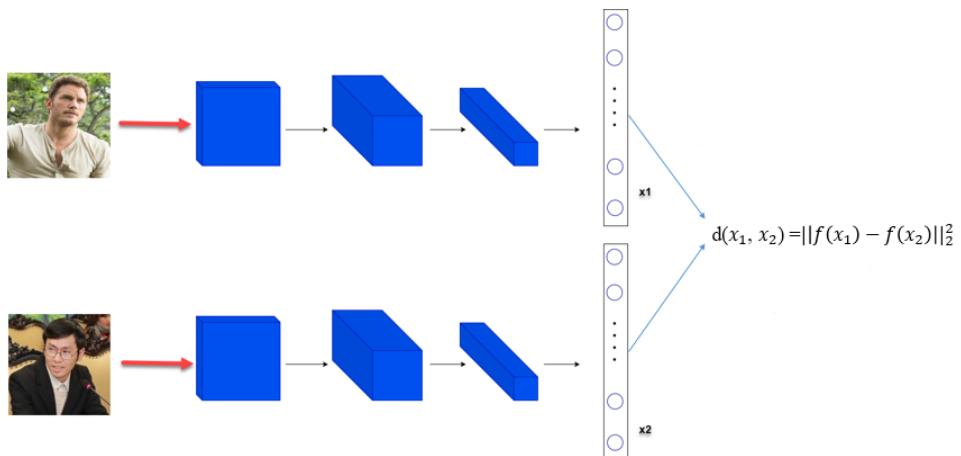


Hình 3.7: Tóm tắt quy trình nhận dạng khuôn mặt sử dụng FaceNet. [7]

3.3 Siamese network

Những kiến trúc mạng mà khi bạn đưa vào 2 bức ảnh và mô hình sẽ trả lời chúng thuộc về cùng 1 người hay không được gọi chung là Siamese network.

Kiến trúc của Siamese network dựa trên base network là một Convolutional neural network đã được loại bỏ lớp output có tác dụng encoding ảnh thành vector embedding. Đầu vào của mạng Siamese network là 2 bức ảnh bất kì được lựa chọn ngẫu nhiên từ dữ liệu ảnh. Output của Siamese network là 2 vector tương ứng với biểu diễn của 2 ảnh đầu vào. Sau đó chúng ta đưa 2 vector vào hàm loss function để đo lường sự khác biệt giữa chúng. Thông thường hàm loss function là một hàm norm chuẩn bậc 2.



Hình 3.8: Siamese network. [8]

Từ mô hình Convolutional neural network, mô hình trả ra 2 vector encoding là và biểu diễn cho lần lượt ảnh 1 và 2. x_1 và x_2 có cùng số chiều. Hàm $f(x)$ có tác dụng tương tự như một phép biến đổi qua lớp Fully connected trong mạng neural network để tạo tính phi tuyến và giảm chiều dữ liệu về các kích thước nhỏ. Thông thường là 128 đối với các mô hình pre-train.

$$d(x_1, x_2) = \|f(x_1) - f(x_2)\|_2^2 \quad (3.1)$$

- Khi x_1, x_2 là cùng 1 người: (3.1) phải là một giá trị nhỏ.
- Khi x_1, x_2 là 2 người khác nhau: (3.1) phải là một giá trị lớn.

3.4 Thuật toán FaceNet

FaceNet chính là một dạng Siamese network có tác dụng biểu diễn các bức ảnh trong một không gian cosine n chiều (thường là 128) sao cho khoảng cách giữa các vector embedding càng nhỏ, mức độ tương đồng giữa chúng càng lớn.

3.4.1 Khái quát thuật toán

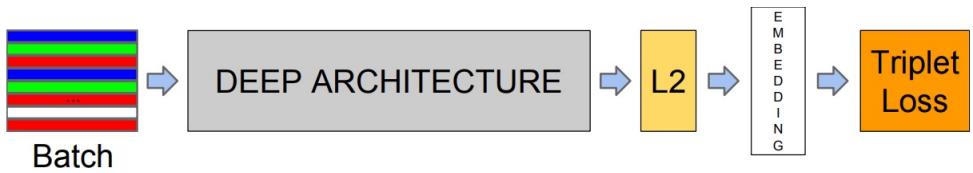
Hầu hết các thuật toán nhận diện khuôn mặt trước FaceNet đều tìm cách biểu diễn khuôn mặt bằng một vector embedding thông qua một lớp bottleneck có tác dụng giảm chiều dữ liệu.

Tuy nhiên hạn chế của các thuật toán này đó là số lượng chiều embedding tương đối lớn (thường ≥ 1000) và ảnh hưởng tới tốc độ của thuật toán. Thường chúng ta phải áp dụng thêm thuật toán PCA để giảm chiều dữ liệu để tăng tốc độ tính toán.

Hàm loss function chỉ đo lường khoảng cách giữa 2 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 mà không học được cùng lúc sự giống nhau và khác nhau trên cùng một lượt huấn luyện.

FaceNet đã giải quyết cả 2 vấn đề trên bằng các hiệu chỉnh nhỏ nhưng mang lại hiệu quả lớn:

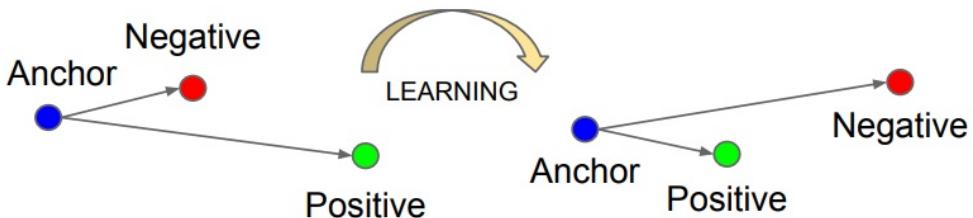
- Base network áp dụng một mạng convolutional neural network và giảm chiều dữ liệu xuống chỉ còn 128 chiều. Do đó quá trình suy diễn và dự báo nhanh hơn và đồng thời độ chính xác vẫn được đảm bảo.
- Sử dụng loss function là hàm triplet loss có khả năng học được đồng thời sự giống nhau giữa 2 bức ảnh cùng nhóm và phân biệt các bức ảnh không cùng nhóm. Do đó hiệu quả hơn rất nhiều so với các phương pháp trước đây.



Hình 3.9: Cấu trúc mô hình FaceNet. [6]

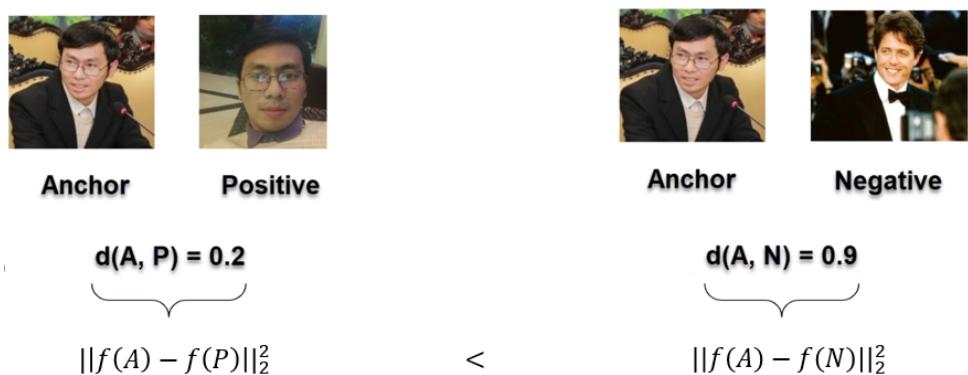
3.4.2 Triplet Loss

Trong FaceNet, quá trình encoding 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 đánh giá khoảng cách giữa các vector.



Hình 3.10: Triplet Loss. [6]

Để áp dụng triplet loss, chúng ta cần lấy ra 3 bức ảnh trong đó có một bức ảnh là anchor. Ảnh anchor cũng có tác dụng gần như vậy. Trong 3 ảnh, ảnh anchor được cố định trước. Chúng ta sẽ lựa chọn 2 ảnh còn lại sao cho một ảnh là negative (của một người khác với anchor) và một ảnh là positive (cùng một người với anchor).



$$\text{Loss function: } \|f(A) - f(P)\|_2^2 - \|f(A) - f(N)\|_2^2 < 0$$

Hình 3.11: Mô phỏng Triplet Loss [8]

Kí hiệu ảnh anchor, positive, negative lần lượt là A, P, N.

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 cần lựa chọn các bộ 3 ảnh sao cho:

- Ảnh anchor và positive khác nhau nhất: cần lựa chọn để khoảng cách $d(A, P)$ lớn. Điều này cũng tương tự như bạn lựa chọn một ảnh của mình hồi nhỏ so với hiện tại để thuật toán học khó hơn. Nhưng nếu nhận biết được thì nó sẽ thông minh hơn.
- Ảnh anchor và negative giống nhau nhất: cần lựa chọn để khoảng cách $d(A, N)$ nhỏ.

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

$$d(A, P) < d(A, N) \quad (3.2)$$

Để làm cho khoảng cách giữa vẽ trái và vẽ phải lớn hơn, chúng ta sẽ cộng thêm vào vẽ trái một hệ số α không âm rất nhỏ. Khi đó (3.2) trở thành:

$$\begin{aligned} d(A, P) + \alpha &\leq d(A, N) \\ \rightarrow \|f(A) - f(P)\|_2^2 + \alpha &\leq \|f(A) - f(N)\|_2^2 \\ \rightarrow \|f(A) - f(P)\|_2^2 - \|f(A) - f(N)\|_2^2 + \alpha &\leq 0 \end{aligned}$$

Như vậy hàm loss function sẽ là:

$$L = \sum_{i=0}^n [\|f(A_i) - f(P_i)\|_2^2 - \|f(A_i) - f(N_i)\|_2^2 + \alpha] \quad (3.3)$$

Trong đó n là số lượng các bộ 3 hình ảnh được đưa vào huấn luyện.

Sẽ không ảnh hưởng gì nếu ta nhận diện đúng ảnh negative và positive là cùng cặp hay khác cặp với anchor. Mục tiêu của chúng ta là giảm thiểu các trường hợp hợp mô hình nhận diện sai ảnh negative thành positive nhất có thể. Do đó để loại bỏ ảnh hưởng

của các trường hợp nhận diện đúng negative và positive lên hàm loss function. Ta sẽ điều chỉnh giá trị đóng góp của nó vào hàm loss function về 0.

3.4.3 Lựa chọn Triplet

Nếu lựa chọn triplet input một cách ngẫu nhiên có thể ảnh hưởng cho bất đẳng thức (3.2) dễ dàng xảy ra vì trong các ảnh ngẫu nhiên, khả năng giống nhau giữa 2 ảnh là rất khó. Hầu hết các trường hợp đều thỏa mãn bất đẳng thức (3.2) và không gây ảnh hưởng đến giá trị của loss function do giá trị của chúng được đặt về 0. Như vậy việc học những bức ảnh negative quá khác biệt với anchor sẽ không có nhiều ý nghĩa.

Để mô hình khó học hơn và đồng thời cũng giúp mô hình phân biệt chuẩn xác hơn mức độ giống và khác nhau giữa các khuôn mặt, chúng ta cần lựa chọn các input theo bộ 3 khó học (hard triplets).

Ý tưởng là chúng ta cần tìm ra bộ ba (A,P,N) sao cho là gần đạt được đẳng thức (xảy ra dấu =) nhất. Tức là $d(A,P)$ lớn nhất và $d(A,N)$ nhỏ nhất. Hay nói cách khác với mỗi anchor A cần xác định:

- **Hard Positive:** Bức ảnh positive có khoảng cách xa nhất với anchor tương ứng với nghiệm:

$$\text{argmax}_{P_i}(d(A_i, P_i))$$

- **Hard Negative:** Bức ảnh negative có khoảng cách gần nhất với anchor tương ứng với nghiệm:

$$\text{argmax}_{N_j}(d(A_j, N_j))$$

Với i,j là nhãn của người trong ảnh.

Lựa chọn triplet sẽ có ảnh hưởng rất lớn tới chất lượng của mô hình FaceNet. Nếu lựa chọn triplet tốt, FaceNet sẽ hội tụ nhanh hơn và đồng thời kết quả dự báo chuẩn xác hơn. Lựa chọn ngẫu nhiên dễ dẫn tới thuật toán không hội tụ.

Chọn đúng triplet sẽ giúp quá trình huấn luyện hội tụ nhanh. Mặt khác, nhóm tác giả sử dụng mini-batches nhỏ do khôi này giúp cải thiện khả năng hội tụ khi sử dụng kỹ

thuật Stochastic Gradient Descent - SGD bằng cách lấy phần biểu thức trong dấu Σ ở (3.3)

$$\|f(A_i) - f(P_i)\|_2^2 - \|f(A_i) - f(N_i)\|_2^2 + \alpha \quad (3.4)$$

sau đó lấy đạo hàm hàm L theo từng biến A_i, N_i, P_i

$$\frac{\partial L}{\partial A_i} = \sum_{i=1}^N \begin{cases} 2(f(N_i) - f(P_i)), & \text{nếu (3.4)} \geq 0 \\ 0, & \text{ngược lại} \end{cases}$$

$$\frac{\partial L}{\partial P_i} = \sum_{i=1}^N \begin{cases} -2(f(A_i) - f(P_i)), & \text{nếu (3.4)} \geq 0 \\ 0, & \text{ngược lại} \end{cases}$$

$$\frac{\partial L}{\partial N_i} = \sum_{i=1}^N \begin{cases} -2(f(A_i) - f(N_i)), & \text{nếu (3.4)} \geq 0 \\ 0, & \text{ngược lại} \end{cases}$$

sau đó ta cập nhật tham số hàm sai số

$$f(A_i) = f(A_i) - \delta \frac{\partial L}{\partial A_i}$$

$$f(P_i) = f(P_i) - \beta \frac{\partial L}{\partial P_i}$$

$$f(N_i) = f(N_i) - \gamma \frac{\partial L}{\partial N_i}$$

với δ, β, γ là tỉ lệ học.

Khi thực nghiệm, nhóm tác giả sử dụng khối gồm 1800 mẫu.

3.4.4 *Mạng tích chập sâu*

Khi thực nghiệm, nhóm chọn tỉ lệ học $\delta, \beta, \gamma = 0.05$, huấn luyện qua một cụm CPU trong 1000 - 2000 giờ, hàm chi phí giảm dần (tức độ chính xác tăng dần) sau 500 giờ huấn luyện. Sau huấn luyện, mỗi ảnh trả về vector đặc trưng 128 chiều, tính chất 2 ảnh cùng người dân đến khoảng cách hai vector gần hơn khoảng cách với ảnh của người khác.

Nhóm tác giả sử dụng 2 kiến trúc để học, một kiến trúc của Zeiler và Fergus. Nhóm tác giả thêm lớp tích chập $1 \times 1 \times d$, thu được mô hình sâu 22 lớp với 140 triệu tham số và cần 1.6 tỷ toán tử/giây/ảnh.

layer	size-in	size-out	kernel	param	FLPS
conv1	$220 \times 220 \times 3$	$110 \times 110 \times 64$	$7 \times 7 \times 3, 2$	9K	115M
pool1	$110 \times 110 \times 64$	$55 \times 55 \times 64$	$3 \times 3 \times 64, 2$	0	
rnorm1	$55 \times 55 \times 64$	$55 \times 55 \times 64$		0	
conv2a	$55 \times 55 \times 64$	$55 \times 55 \times 64$	$1 \times 1 \times 64, 1$	4K	13M
conv2	$55 \times 55 \times 64$	$55 \times 55 \times 192$	$3 \times 3 \times 64, 1$	111K	335M
rnorm2	$55 \times 55 \times 192$	$55 \times 55 \times 192$		0	
pool2	$55 \times 55 \times 192$	$28 \times 28 \times 192$	$3 \times 3 \times 192, 2$	0	
conv3a	$28 \times 28 \times 192$	$28 \times 28 \times 192$	$1 \times 1 \times 192, 1$	37K	29M
conv3	$28 \times 28 \times 192$	$28 \times 28 \times 384$	$3 \times 3 \times 192, 1$	664K	521M
pool3	$28 \times 28 \times 384$	$14 \times 14 \times 384$	$3 \times 3 \times 384, 2$	0	
conv4a	$14 \times 14 \times 384$	$14 \times 14 \times 384$	$1 \times 1 \times 384, 1$	148K	29M
conv4	$14 \times 14 \times 384$	$14 \times 14 \times 256$	$3 \times 3 \times 384, 1$	885K	173M
conv5a	$14 \times 14 \times 256$	$14 \times 14 \times 256$	$1 \times 1 \times 256, 1$	66K	13M
conv5	$14 \times 14 \times 256$	$14 \times 14 \times 256$	$3 \times 3 \times 256, 1$	590K	116M
conv6a	$14 \times 14 \times 256$	$14 \times 14 \times 256$	$1 \times 1 \times 256, 1$	66K	13M
conv6	$14 \times 14 \times 256$	$14 \times 14 \times 256$	$3 \times 3 \times 256, 1$	590K	116M
pool4	$14 \times 14 \times 256$	$7 \times 7 \times 256$	$3 \times 3 \times 256, 2$	0	
concat	$7 \times 7 \times 256$	$7 \times 7 \times 256$		0	
fc1	$7 \times 7 \times 256$	$1 \times 32 \times 128$	maxout p=2	103M	103M
fc2	$1 \times 32 \times 128$	$1 \times 32 \times 128$	maxout p=2	34M	34M
fc7128	$1 \times 32 \times 128$	$1 \times 1 \times 128$		524K	0.5M
L2	$1 \times 1 \times 128$	$1 \times 1 \times 128$		0	
total				140M	1.6B

Hình 3.12: Nhúng FaceNet thành vector 128 chiều. [6]

Cấu trúc mạng hình 3.12 do Zeiler và Fergus [9] đề xuất, với các lớp (layer), kích thước input (size-in) và output (size-out) có dạng rows * cols * #filter (dòng * cột, bước sải) và kích thước maxout pooling là $p = 2$.

Kiến trúc còn lại từ mô hình Inception của GoogLeNet [10] - một cấu trúc mạng CNN được giới thiệu vào năm 2014 của Google, với đặc trưng là các khối Inception. Khối này cho phép mạng được học theo cấu trúc song song, nghĩa là với 1 đầu vào có thể được đưa vào nhiều các lớp Convolution khác nhau để đưa ra các kết quả khác nhau, sau đó sẽ được Concatenate vào thành 1 output. Việc học song song này giúp mạng có thể học được nhiều chi tiết hơn, lấy được nhiều đặc trưng hơn so với mạng CNN truyền thống. Ngoài ra, mạng cũng áp dụng các khối Convolution 1×1 nhằm giảm kích thước của mạng, khiến việc huấn luyện trở nên nhanh hơn [11].

Ý tưởng chính của kiến trúc Inception nhằm quan sát cách các bộ phận có sẵn của đối tượng bao phủ và xấp xỉ cấu trúc thừa địa phương tối ưu hóa của mạng thị giác tích chập. Module Inception được xây dựng như trong hình 3.13. Mô hình này có khoảng 6.6 - 7.5 triệu tham số và khoảng 500 triệu - 1.6 tỷ toán tử. Chi tiết xem tại hình 3.14.

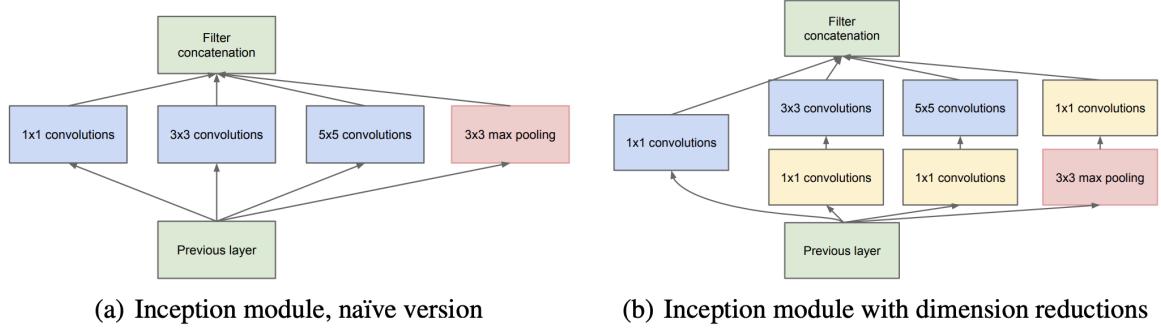


Figure 2: Inception module

Hình 3.13: Module Inception [10]

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 ($7 \times 7 \times 3, 2$)	$112 \times 112 \times 64$	1							9K	119M
max pool + norm	$56 \times 56 \times 64$	0						m 3×3, 2		
inception (2)	$56 \times 56 \times 192$	2		64	192				115K	360M
norm + max pool	$28 \times 28 \times 192$	0						m 3×3, 2		
inception (3a)	$28 \times 28 \times 256$	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	$28 \times 28 \times 320$	2	64	96	128	32	64	L ₂ , 64p	228K	179M
inception (3c)	$14 \times 14 \times 640$	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	$14 \times 14 \times 640$	2	256	96	192	32	64	L ₂ , 128p	545K	107M
inception (4b)	$14 \times 14 \times 640$	2	224	112	224	32	64	L ₂ , 128p	595K	117M
inception (4c)	$14 \times 14 \times 640$	2	192	128	256	32	64	L ₂ , 128p	654K	128M
inception (4d)	$14 \times 14 \times 640$	2	160	144	288	32	64	L ₂ , 128p	722K	142M
inception (4e)	$7 \times 7 \times 1024$	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	$7 \times 7 \times 1024$	2	384	192	384	48	128	L ₂ , 128p	1.6M	78M
inception (5b)	$7 \times 7 \times 1024$	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	$1 \times 1 \times 1024$	0								
fully conn	$1 \times 1 \times 128$	1							131K	0.1M
L2 normalization	$1 \times 1 \times 128$	0								
total									7.5M	1.6B

Hình 3.14: Cấu trúc mạng Inception [6]

FaceNet sử dụng mô hình Inception tương tự với [12]. Hai điểm khác biệt chính đó là FaceNet sử dụng L_2 pooling thay vì max pooling. Kích thước pooling luôn luôn 3×3 và tính song song với module tích chập trong mỗi module Inception.

Chương 4 Thực nghiệm

4.1 Giới thiệu về thư viện DeepFace

Thư viện DeepFace [13] là một thư viện đa năng được viết để sử dụng với python3 và nó được tích hợp sẵn rất nhiều model nhận diện, so sánh tốt nhất hiện nay (state of the art) như là: Google FaceNet, VGG-Face, OpenFace, Facebook DeepFace, DeepID, ArcFace, Dlib và SFace.

Ngoài nhận diện và so sánh khuôn mặt, DeepFace còn cung cấp cho chúng ta 1 số chức năng khá thú vị như phân tích khuôn mặt để dự đoán độ tuổi, giới tính, chủng tộc, cảm xúc.

Trong bài báo cáo này sử dụng model FaceNet để nhận diện và so sánh khuôn mặt.

4.2 Xây dựng ứng dụng Flask nhận diện khuôn mặt

Toàn bộ phần mã nguồn và giải thích đoạn mã nguồn bằng tiếng Anh đi kèm sẽ được trình bày ở phần phụ lục A.

Mã nguồn và dữ liệu có thể được lấy từ đây:

[https://github.com/NguyenTrieuVuong/deepface1333.](https://github.com/NguyenTrieuVuong/deepface1333)

4.2.1 Xây dựng ứng dụng Flask so sánh 2 người

Mã nguồn và giải thích đoạn mã nguồn được đính kèm ở phần phụ lục A.1

Từ giao diện trang chủ, nếu ta chọn 2 ảnh đầu vào cùng người, kết quả sẽ như hình 4.3, còn nếu ta chọn 2 ảnh đầu vào cùng người, kết quả sẽ như hình 4.5. Quan sát 2 kết quả, ta thấy khi distance nhỏ hơn threshold, kết quả sẽ cho ra đúng người. Còn khi ta thấy khi distance lớn hơn threshold, kết quả sẽ cho ra khác người.

Using Facenet For verify image

Không tệp nào được chọn

Không tệp nào được chọn

Hình 4.1: Trang chủ.

Ta chọn 2 ảnh đầu vào cùng 1 người, nhấn "Nộp" trả về kết quả cùng 1 người.



Hình 4.2: Hai ảnh đầu vào cùng người.

```
{  
  "detector_backend": "opencv",  
  "distance": 0.09402392136123494,  
  "model": "Facenet",  
  "similarity_metric": "cosine",  
  "threshold": 0.4,  
  "verified": true  
}
```

Hình 4.3: Kết quả 1.

Ta chọn 2 ảnh đầu vào của 2 người khác nhau, nhấn "Nộp" trả về kết quả không cùng 1 người.



Hình 4.4: Hai ảnh đầu vào khác người.

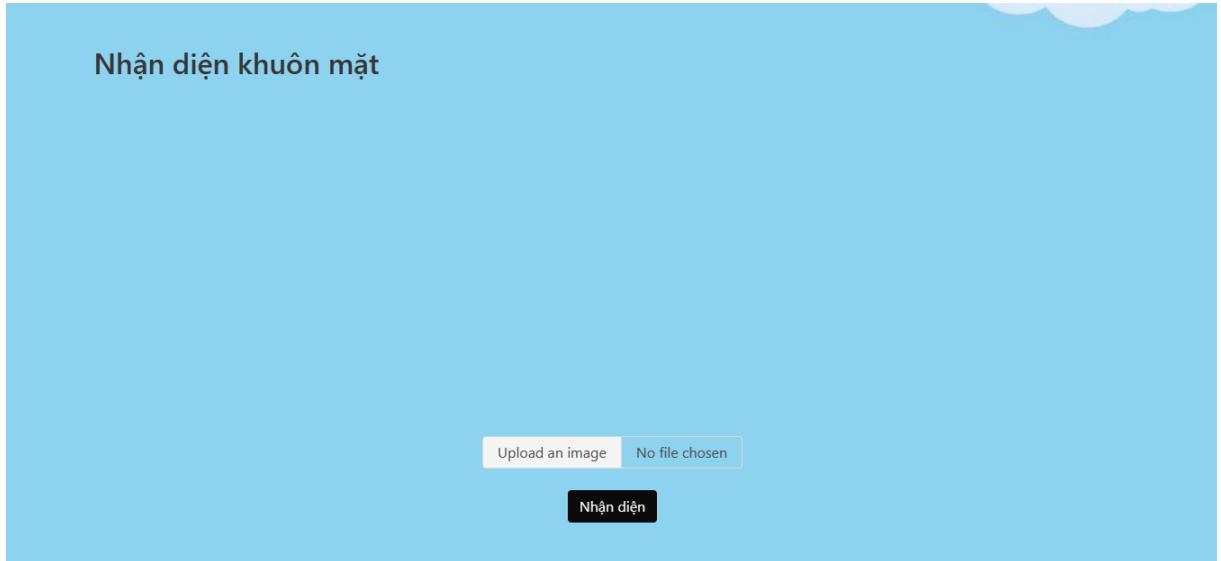
```
{  
    "detector_backend": "opencv",  
    "distance": 0.8739010290484228,  
    "model": "Facenet",  
    "similarity_metric": "cosine",  
    "threshold": 0.4,  
    "verified": false  
}
```

Hình 4.5: Kết quả 2.

4.2.2 Xây dựng ứng dụng Flask nhận diện 1 trong 9 người

Mã nguồn và giải thích đoạn mã nguồn được đính kèm ở phần phụ lục A.2

Sử dụng bộ dữ liệu gồm 9 người : Trump, Obama, Ben Affleck, Gautam Rode, Jerry Seinfeld, Madonna, Mindy Kaling, Dwayne Johnson và Cillian Murphy. Pre-train dựa trên model có sẵn của DeepFace: FaceNet.



Hình 4.6: Trang chủ.

Ta lần lượt tải lên 3 ảnh đầu vào khác nhau tương ứng với 3 người được thử nghiệm, khi ảnh được nhận diện, nó sẽ cho ra kết quả tương ứng với tên người được thử nghiệm và phần trăm độ chính xác của ảnh mặt người. Confidence là số phần trăm dự đoán tên người trên ảnh đó.



Hình 4.7: Ba ảnh đầu vào.

Nhận diện khuôn mặt



Đây là **Ben**

Affleck

Confidence:

68.19%

[Back to Home](#)

Nhận diện khuôn mặt



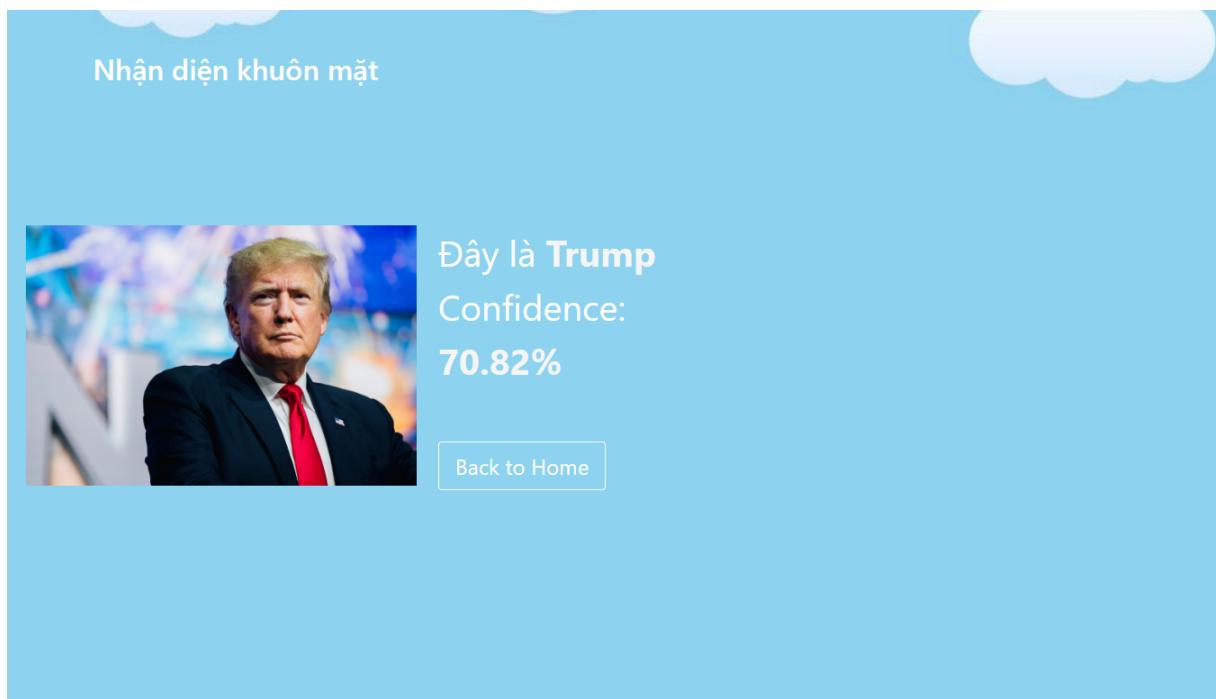
Đây là **Madonna**

Confidence:

71.63%

[Back to Home](#)

Hình 4.8: Kết quả 1.



Hình 4.10: Kết quả 3.

4.3 Tính Accuracy và Precision của mô hình

Để thống kê độ chính xác và mức độ tuyệt đối của sự chính xác của 2 ứng dụng trên, ta sẽ tính độ chính xác và độ chụm của mô hình.

Accuracy (độ chính xác) là mức độ chính xác, trong khi Precision (độ chụm) là mức độ tuyệt đối của sự chính xác đó – biểu hiện ở mức lặp của kết quả.

Ta tính Accuracy và Precision theo công thức sau đây:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

trong đó:

- TP : kết quả đúng trong cùng thư mục
- TN : kết quả đúng trong thư mục khác

- *FP*: kết quả sai trong cùng thư mục

- *FN*: kết quả sai trong thư mục khác

	Train		Test	
	Accuracy (%)	Precision (%)	Accuracy (%)	Precision (%)
Ben Affleck	94,51	50,62	94,44	55,56
Cillian Murphy	100	100	98,46	86,11
Dwayne Johnson	98,22	100	96,17	76,11
Gautam Rode	99,38	97,53	95,69	67,22
Jerry Seinfeld	99,07	95,06	96,67	79,44
Madonna	95,2	56,79	97,22	75
Mindy Kaling	97,81	80,25	94,94	54,44
Obama	97,53	80,25	94,44	65,56
Trump	95,06	60,49	96,17	65,56
Average	97,42	80,11	96,02	69,44

Hình 4.11: Accuracy và Precision ở tập train và tập test.

Từ thư mục **test** trong file zip **deepface1333.zip** gồm 9 người, mỗi người có 20 ảnh khác tập train và validation, ta tính được trung bình Accuracy đạt đến mức 96% và Precision đạt gần 70%.

KẾT LUẬN

Bài báo cáo đã trình bày chi tiết về nhận diện khuôn mặt sử dụng mô hình học sâu. Đầu tiên bài báo cáo khái quát về xử lý ảnh và bài toán nhận diện khuôn mặt, trình bày khái quát về học máy, học sâu, trình bày chi tiết về mô hình học sâu CNN. Tiếp đó bài báo cáo khái quát về mô hình FaceNet và sử dụng mô hình này để tính khoảng cách nhúng của 2 vector 128 chiều từ đó so sánh 2 khuôn mặt với nhau. Cuối cùng bài báo cáo khái quát về thư viện DeepFace, ứng dụng thư viện đó xây dựng ứng dụng Flask so sánh 1 trong 2 người, xây dựng Flask nhận diện 1 trong 9 người và tính các độ chính xác của mô hình.

Kết quả thực nghiệm đạt được 1 số thuận lợi như độ chính xác cao, khả năng nhận diện khá nhanh, nhưng không thể tránh khỏi một số nhược điểm như độ chụm chỉ ở mức trung bình, mô hình có độ phức tạp vừa phải, vẫn còn hiện tượng dự đoán sai, lượng dữ liệu số người còn hạn chế.

Từ bài toán nhận diện khuôn mặt sử dụng mô hình học sâu, có thể thấy được tầm quan trọng của nhận diện khuôn mặt trong nhiều hướng nghiên cứu như: Điểm danh sinh viên có mặt trong lớp học, ứng dụng trong hệ thống quan sát, theo dõi và bảo vệ ở cơ quan, tổ chức tìm kiếm liên quan đến con người thông qua khuôn mặt trên nhiều hệ cơ sở dữ liệu lớn, ứng dụng trong lĩnh vực giải trí (biên khuôn mặt thật thành mặt anime)...

Tài liệu tham khảo

- [1] Phuoc-Hai Huynh, Van Hoa Nguyen, and Thanh-Nghi Do. “SO SÁNH MÔ HÌNH HỌC SÂU VỚI CÁC PHƯƠNG PHÁP HỌC TỰ ĐỘNG KHÁC TRONG PHÂN LỐP DỮ LIỆU BIỂU HIỆN GEN MICROARRAY”. In: Aug. 2017, pp. 843–844. DOI: [10.15625/vap.2017.00099](https://doi.org/10.15625/vap.2017.00099).
- [2] ITNavi. *Những Thông Tin Về Cấu Trúc Mạng CNN Là Gì?* <https://itnavi.com.vn/blog/cnn-la-gi>. 2021.
- [3] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, et al. “Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks”. In: *IEEE Signal Processing Letters* 23.10 (2016), pp. 1499–1503. DOI: [10.1109/lsp.2016.2603342](https://doi.org/10.1109/lsp.2016.2603342). URL: <https://doi.org/10.1109/lsp.2016.2603342>.
- [4] Tienthegainz. *Hệ thống phát gạo nhận diện khuôn mặt(Phần 1)*. <https://viblo.asia/p/he-thong-phat-gao-nhan-dien-khuon-matphan-1-Qbq5Q0gw1D8>. 2020.
- [5] Luka Dulčić. *Face Recognition with FaceNet and MTCNN*. <https://arsfutura.com/magazine/face-recognition-with-facenet-and-mtcnn/>. 2019.
- [6] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “Facenet: A unified embedding for face recognition and clustering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.
- [7] Brandon Amos. *OpenFace*. <https://cmusatyalab.github.io/openface/>.
- [8] Phạm Đình Khánh. *Mô hình Facenet trong face recognition*. <https://phamdinhkhanh.github.io/2020/03/12/faceNetAlgorithm.html>. 2020.
- [9] Matthew D Zeiler and Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2013. DOI: [10.48550/ARXIV.1311.2901](https://arxiv.org/abs/1311.2901). URL: <https://arxiv.org/abs/1311.2901>.
- [10] Christian Szegedy, Wei Liu, Yangqing Jia, et al. *Going Deeper with Convolutions*. 2014. DOI: [10.48550/ARXIV.1409.4842](https://arxiv.org/abs/1409.4842). URL: <https://arxiv.org/abs/1409.4842>.
- [11] Quang Trần. *Nhận diện khuôn mặt với mạng MTCNN và FaceNet (Phần 1)*. <https://viblo.asia/p/nhan-dien-khuon-mat-voi-mang-mtcnn-va-facenet-phan-1-Qbq5QDN41D8>. 2021.
- [12] Christian Szegedy, Wei Liu, Yangqing Jia, et al. “Going deeper with convolutions”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1–9. DOI: [10.1109/CVPR.2015.7298594](https://doi.org/10.1109/CVPR.2015.7298594).

- [13] Sefik Ilkin Serengil and Alper Ozpinar. “LightFace: A Hybrid Deep Face Recognition Framework”. In: *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*. IEEE. 2020, pp. 23–27. DOI: [10.1109/ASYU50717.2020.9259802](https://doi.org/10.1109/ASYU50717.2020.9259802). URL: <https://doi.org/10.1109/ASYU50717.2020.9259802>.

Phụ lục A Liệt kê mã nguồn

Mã nguồn và dữ liệu có thể được lấy từ đây:

[https://github.com/NguyenTrieuVuong/deepface1333.](https://github.com/NguyenTrieuVuong/deepface1333)

A.1 Mã nguồn xây dựng ứng dụng Flask so sánh 2 người

Mã nguồn **FaceNetAPI.py** nằm trong thư mục **Face-Recog-Facenet-API**:

```
1 # Import all required libraries
2 import cv2
3 import matplotlib.pyplot as plt
4 import os
5 import numpy as np
6 from numpy.core.numeric import identity
7 from sklearn.preprocessing import Normalizer
8 from PIL import Image
9 from deepface.basemodels import VGGFace, Facenet
10 from flask import Flask, render_template, request, send_from_directory, url_for,
11     redirect, jsonify, json
12 import codecs
13 import base64
14 from deepface.extendedmodels import Age, Gender, Race, Emotion
15 from deepface.commons import functions, realtime, distance as dst
16 from deepface import DeepFace
17
18 # Run Flask
19 app = Flask(__name__)
20
21 # Home page
22 @app.route('/')
23 def home():
24     return render_template('verify_facenet.html')
25
26 #-----#
27 # VERIFY FACENET
28
29 @app.route('/verify/facenet/distance', methods=['POST'])
30 def distance1():
31     model = "Facenet"
32
33     # upload 2 images
34     img_upload = request.files['image']
35     img2_upload = request.files['image2']
36
37     # save 2 uploaded images to folder "upload"
38     img = os.path.join("upload", img_upload.filename)
39     print(img)
40     img_upload.save(img)
41     img2 = os.path.join("upload", img2_upload.filename)
42     print(img2)
43     img2_upload.save(img2)
44
45     # verify the comparation of 2 images
46     resp=DeepFace.verify(img1_path=img, img2_path=img2,model_name=model)
47
48     # return result
```

```

48     return jsonify(resp)
49 #-----
50
51 if __name__ == '__main__':
52     app.run(debug=True)

```

A.2 Mã nguồn xây dựng ứng dụng Flask nhận diện 1 trong 9 người

Mã nguồn `app.py` nằm trong thư mục **Face-Recog-1-in-9-people**:

```

1 # Import all required libraries
2 import os
3 import tensorflow as tf
4 from flask import Flask, render_template, request, send_from_directory
5 from deepface.basemodels import Facenet
6 from deepface.commons import functions, realtime, distance as dst
7 from deepface import DeepFace
8 from numpy import expand_dims
9 from sklearn.preprocessing import LabelEncoder
10 from sklearn.preprocessing import Normalizer
11 from deepface.basemodels import Facenet
12 import numpy as np
13 import pickle
14
15 # Run Flask
16 app = Flask(__name__)
17
18 dir_path = os.path.dirname(os.path.realpath(__file__))
19 UPLOAD_FOLDER = "uploads"
20 STATIC_FOLDER = "static"
21
22 # Load model and read "facenet.h5" from folder "static/models"
23 loaded_model = pickle.load(open(STATIC_FOLDER + "/models/" + "facenet.h5", 'rb'))
24
25 # Read the image from path and preprocess using model "Facenet"
26 def load_and_preprocess_image(path):
27     embedding = DeepFace.represent(img_path = path , model_name = 'Facenet')
28     samples = expand_dims(embedding, axis=0)
29     return samples
30
31
32 # Predict & classify image
33 def classify(model, image_path):
34     label = 0
35
36     preprocessed_imgage = load_and_preprocess_image(image_path)
37
38     yhat_class = loaded_model.predict(preprocessed_imgage)
39     yhat_prob = loaded_model.predict_proba(preprocessed_imgage)
40     class_index = yhat_class[0]
41     class_probability = yhat_prob[0,class_index] * 100
42
43     if class_index >= 0.5 :
44         label = "Trump"
45
46     else:
47         label = "Obama"
48
49     return label, class_probability
50
51 #-----
52 # FLASK
53
54 # Flask Home page

```

```

55 @app.route("/")
56 def home():
57     return render_template("home.html")
58
59 # Flask load image
60 @app.route("/classify", methods=["POST", "GET"])
61 def upload_file():
62
63     if request.method == "GET":
64         return render_template("home.html")
65
66     else:
67         # upload and save image to folder "uploads"
68         file = request.files["image"]
69         upload_image_path = os.path.join(UPLOAD_FOLDER, file.filename)
70         print(upload_image_path)
71         file.save(upload_image_path)
72
73         # classify and verify label of the two people
74         label, prob = classify_loaded_model, upload_image_path)
75
76         prob = np.round(prob, 2)
77
78 # Flask Classify page after recognition
79     return render_template(
80         "classify.html", image_file_name=file.filename, label=label, prob=prob
81     )
82
83 # Flask send image file to folder "uploads"
84 @app.route("/classify/<filename>")
85 def send_file(filename):
86     return send_from_directory(UPLOAD_FOLDER, filename)
87 -----
88
89 if __name__ == "__main__":
90     app.debug = True
91     app.run(debug=True)
92     app.debug = True

```

A.3 Mã nguồn xây dựng model FaceNet

Mã nguồn **testmodel.ipynb** được tạo ra nhằm trích xuất file **facenet.h5** để đưa vào thư mục **Face-Recog-1-in-9-people/static/models**, chạy ứng dụng Flask nhận diện 1 trong 2 người. Ta mở colab.research.google.com, mở file **test-model.ipynb**, tải file **deepface1333.zip** lên mục "Tệp" ở thư mục content và chạy tất cả:

```

1 # extract to folder name "deepface1333" in folder "content"
2 !unzip /content/deepface1333.zip -d "/content/"

1 # Install deepface library in GitHub
2 !pip install deepface

1 # Import all required libraries
2 import time
3 from os import path
4 import numpy as np
5 import pandas as pd
6 from tqdm import tqdm

```

```

7 import pickle
8
9 from deepface.basemodels import VGGFace, OpenFace, Facenet, Facenet512,
10    FbDeepFace, DeepID, DlibWrapper, ArcFace, Boosting
11 from deepface.extendedmodels import Age, Gender, Race, Emotion
12 from deepface.commons import functions, realtime, distance as dst
13 from deepface import DeepFace
14
15 import matplotlib.pyplot as plt
16 import tensorflow as tf
17
18
19 from os import listdir
20 from os.path import isdir
21 from PIL import Image
22 from numpy import savez_compressed
23 from numpy import asarray
24 from mtcnn.mtcnn import MTCNN
25
26
27 # load images and extract faces from path
28 def load_faces(directory):
29     faces = list()
30     # enumerate files
31     for filename in listdir(directory):
32         # path
33         path = directory + filename
34         # get face
35         # face = extract_face(path)
36         # store
37         faces.append(path)
38     return faces
39
40
41 # load a dataset
42 def load_dataset(directory):
43     X, y = list(), list()
44
45     for subdir in listdir(directory):
46         # path
47         path = directory + subdir + '/'
48         # skip any files that might be in the dir
49         if not isdir(path):
50             continue
51         # load all faces in the subdirectory
52         faces = load_faces(path)
53         # create labels
54         labels = [subdir for _ in range(len(faces))]
55         # summarize progress
56         print('>loaded %d examples for class: %s' % (len(faces), subdir))
57         # store
58         X.extend(faces)
59         y.extend(labels)
60     return asarray(X), asarray(y)
61
62 # load train dataset
63 trainX, trainy = load_dataset('/content/deepface1333/data/train/')
64 print(trainX.shape, trainy.shape)
65 # load test dataset
66 testX, testy = load_dataset('/content/deepface1333/data/val/')
67 print(testX.shape, testy.shape)
68 # save arrays to one file in compressed format
69 savez_compressed('5-celebrity-faces-dataset.npz', trainX, trainy, testX, testy)

```

```

>loaded 9 examples for class: jerry_seinfeld
>loaded 9 examples for class: ben_afflek
>loaded 9 examples for class: gautam_rode
>loaded 9 examples for class: mindy_kaling
>loaded 9 examples for class: trump
>loaded 9 examples for class: obama

```

```

>loaded 9 examples for class: dwayne_johnson
>loaded 9 examples for class: madonna
>loaded 9 examples for class: cillian_murphy
(81,) (81,)
>loaded 5 examples for class: jerry_seinfeld
>loaded 4 examples for class: ben_afflek
>loaded 3 examples for class: gautam_rode
>loaded 4 examples for class: mindy_kaling
>loaded 4 examples for class: trump
>loaded 4 examples for class: obama
>loaded 3 examples for class: dwayne_johnson
>loaded 3 examples for class: madonna
>loaded 5 examples for class: cillian_murphy
(35,) (35,)

```

```

1 from numpy import load
2 from numpy import expand_dims
3 from numpy import asarray
4 from numpy import savez_compressed
5 from keras.models import load_model
6 import os
7 # load the face dataset
8 data = load('5-celebrity-faces-dataset.npz')
9 trainX, trainy, testX, testy = data['arr_0'], data['arr_1'], data['arr_2'], data['arr_3']
10 print('Loaded: ', trainX.shape, trainy.shape, testX.shape, testy.shape)
11 # convert each face in the train set to an embedding
12 newTrainX = []
13 train_dir = '/content/deepface1333/data/train/'
14 for directory in os.listdir(train_dir):
15     for i in os.listdir(os.path.join(train_dir, directory)):
16         embedding = DeepFace.represent(img_path = os.path.join(train_dir, directory, i),
17                                         model_name = 'Facenet')
18         newTrainX.append(embedding)
19 newTrainX = asarray(newTrainX)
20 print(newTrainX.shape)
21 # convert each face in the test set to an embedding
22 newTestX = []
23 val_dir = '/content/deepface1333/data/val/'
24 for directory in os.listdir(val_dir):
25     for i in os.listdir(os.path.join(val_dir, directory)):
26         embedding = DeepFace.represent(img_path = os.path.join(val_dir, directory, i),
27                                         model_name = 'Facenet')
28         newTestX.append(embedding)
29 newTestX = asarray(newTestX)
30 print(newTestX.shape)
31 # save arrays to one file in compressed format
32 savez_compressed('5-celebrity-faces-embeddings.npz', newTrainX, trainy, newTestX,
33                  testy)

# develop a classifier for the 5 Celebrity Faces Dataset
1 from random import choice
2 from numpy import load
3 from numpy import expand_dims
4 from sklearn.preprocessing import LabelEncoder
5 from sklearn.preprocessing import Normalizer
6 from sklearn.svm import SVC
7 from matplotlib import pyplot
8 # load faces
9 data = load('/content/5-celebrity-faces-dataset.npz')
10 testX_faces = data['arr_2']
11 # load face embeddings
12 data = load('/content/5-celebrity-faces-embeddings.npz')
13 trainX, trainy, testX, testy = data['arr_0'], data['arr_1'], data['arr_2'], data['arr_3']
14 out_encoder = LabelEncoder()

```

```

16     out_encoder.fit(trainy)
17     trainy = out_encoder.transform(trainy)
18     testy = out_encoder.transform(testy)
19     # fit model
20     model = SVC(kernel='linear', probability=True)
21     model.fit(trainX, trainy)

```

```
SVC(kernel='linear', probability=True)
```

```

1 import matplotlib.image as mpimg
2 # test model on a random example from the test dataset
3 selection = choice([i for i in range(testX.shape[0])])
4 random_face_pixels = testX_faces[selection]
5 random_face_emb = testX[selection]
6 random_face_class = testy[selection]
7 random_face_name = out_encoder.inverse_transform([random_face_class])
8 # predict face
9 samples = expand_dims(random_face_emb, axis=0)
10 yhat_class = model.predict(samples)
11 yhat_prob = model.predict_proba(samples)
12 # get name
13 class_index = yhat_class[0]
14 class_probability = yhat_prob[0,class_index] * 100
15 predict_names = out_encoder.inverse_transform(yhat_class)
16 print('Predicted: %s (%.3f)' % (predict_names[0], class_probability))
17 print('Expected: %s' % random_face_name[0])
18 # plot image
19 image = mpimg.imread(random_face_pixels)
20 plt.imshow(image)
21 plt.show()

```

↳ Predicted: ben_afflek (68.937)
 Expected: ben_afflek



```

1 import pickle
2 # save the model to disk
3 filename = 'facenet.h5'
4 pickle.dump(model, open(filename, 'wb'))

```



```

1 # score result to model file "facenet.h5"
2 loaded_model = pickle.load(open('/content/facenet.h5', 'rb'))
3 result = loaded_model.score(newTestX, testy)
4 print(result)

```

1.0

```
1 # TEST
2 import glob
3 # upload and test 20 pictures of 1 tested person in folder "test" which is not like
4     those in folder "train" and "val" of zip file "deepface1333.zip"
5 path = '/content/test/cillian_murphy/*'
6 for i in glob.glob(path):
7     print(i)
8     embedding = DeepFace.represent(img_path = i , model_name = 'Facenet')
9     # predict face using saved model
10    labels = ["Ben Affleck", "Cillian Murphy", "Dwayne Johnson", "Gautam Rode", "
11        Jerry Seinfeld", "Madonna", "Mindy Kaling", "Obama", "Trump"]
12    samples = expand_dims(embedding, axis=0)
13    yhat_class = loaded_model.predict(samples)
14    yhat_prob = loaded_model.predict_proba(samples)
15    # get name
16    class_index = yhat_class[0]
17    predicted_label = labels[class_index]
18    print(predicted_label)
```

```
/content/test/cillian_murphy/cillian-murphy2.jpg
1/1 [=====] - 0s 107ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy8.jpg
1/1 [=====] - 0s 108ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy11.jpg
1/1 [=====] - 0s 109ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy12.jpg
1/1 [=====] - 0s 105ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy1.jpg
1/1 [=====] - 0s 114ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy19.jpg
1/1 [=====] - 0s 105ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy18.jpg
1/1 [=====] - 0s 137ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy6.jpg
1/1 [=====] - 0s 104ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy7.jpg
1/1 [=====] - 0s 103ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy17.jpg
1/1 [=====] - 0s 102ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy13.jpg
1/1 [=====] - 0s 118ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy15.jpg
1/1 [=====] - 0s 104ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy10.jpg
1/1 [=====] - 0s 110ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy4.jpg
1/1 [=====] - 0s 100ms/step
Madonna
/content/test/cillian_murphy/cillian-murphy5.jpg
1/1 [=====] - 0s 113ms/step
```

```
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy14.jpg
1/1 [=====] - 0s 99ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy9.jpg
1/1 [=====] - 0s 102ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy.jpg
1/1 [=====] - 0s 102ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy16.jpg
1/1 [=====] - 0s 114ms/step
Cillian Murphy
/content/test/cillian_murphy/cillian-murphy3.jpg
1/1 [=====] - 0s 106ms/step
Cillian Murphy
```