

**ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**



**BÁO CÁO THỰC TẬP
CHỦ ĐỀ
NHẬN DIỆN KHUÔN MẶT
BẰNG PHƯƠNG PHÁP HỌC MÁY**

Học phần: **Thực tập thực tế về Khoa học dữ liệu**

Mã học phần: **MAT3381**

Người hướng dẫn: **Lại Tiên Đức - Công ty intX Việt Nam**

Sinh viên thực hiện: **Vũ Mạnh Đức – MSV: 20002046**

Phan Tấn Sơn – MSV: 20002090

Vũ Trọng Quân – MSV: 20002087

HÀ NỘI, 8/2023

LỜI CẢM ƠN

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến toàn thể các thầy cô trong khoa Toán – Cơ – Tin học – Trường Đại học Khoa học Tự nhiên - ĐHQGHN đã tạo điều kiện cho chúng em được đi thực tập và viết bài báo cáo này, đây là cơ hội tốt để chúng em có thể thực hành các kỹ năng học trên lớp và cũng giúp ích rất lớn để chúng em ngày càng tự tin về bản thân mình.

Chúng em xin gửi lời cảm ơn chân thành đến anh Trần Nam Trung và toàn thể công ty cổ phần giải pháp công nghệ intX, đặc biệt là anh Lại Tiến Độ đã trực tiếp hướng dẫn, chỉ bảo và đã cho chúng em nhiều kinh nghiệm quý báu trong suốt thời gian thực tập tại Công ty.

Em xin chân thành cảm ơn!

Hà Nội, ngày 18 tháng 08 năm 2023

Sinh viên thực hiện

Vũ Mạnh Đức

Phan Tấn Sơn

Vũ Trọng Quân

MỤC LỤC

CHƯƠNG I. GIỚI THIỆU	1
1. Phát biểu bài toán	1
2. Công nghệ sử dụng	1
2.1. <i>Python</i>	1
2.2. <i>Thư viện openCV</i>	2
2.3. <i>Thư viện dlib</i>	2
2.4. <i>Thư viện tkinter</i>	2
2.5. <i>Google colab</i>	3
2.6. <i>Visual Studio Code</i>	3
CHƯƠNG II. CÔNG VIỆC TRIỂN KHAI	4
1. Học agile scrum course	4
2. Phân tích bài toán nhận diện khuôn mặt	5
2.1. <i>Khó khăn</i>	5
2.2. <i>Các vấn đề cần giải quyết</i>	6
3. Phát hiện khuôn mặt	7
4. Tìm hiểu về YOLOv7	9
4.1. <i>Giới thiệu về YOLO</i>	9
4.2. <i>Hoạt động của YOLO</i>	9
4.3. <i>YOLOv7</i>	11
4.4. <i>Hạn chế của YOLOv7</i>	13
5. Tìm hiểu về Facial Landmarks	15
6. Tìm hiểu về MTCNN	18
6.1. <i>Giới thiệu MTCNN</i>	18
6.2. <i>Cấu trúc MTCNN</i>	18
6.3. <i>Ưu điểm của MTCNN</i>	19
6.4. <i>Hạn chế của MTCNN</i>	20
7. Face Classification	20
CHƯƠNG III. CÁC KẾT QUẢ ĐẠT ĐƯỢC	25
1. Chuẩn bị dữ liệu	25

<i>1.1. Thư viện</i>	25
<i>1.2. Xây dựng bộ dữ liệu</i>	25
2. Trích rút đặc trưng	26
3. Kết quả chạy demo.....	26
CHƯƠNG IV. PHỤ LỤC	29
1. Phân chia công việc	29
2. Tài liệu tham khảo.....	30
NHẬN XÉT THỰC TẬP	32

DANH MỤC HÌNH ẢNH

Hình 1. Phát hiện khuôn mặt trong ảnh.....	7
Hình 2. Phát hiện khuôn mặt qua HOG.....	9
Hình 3. Cấu trúc của YOLO.....	10
Hình 4. So sánh tốc độ của YOLOv7 với các phiên bản trước.....	12
Hình 5. Độ chính xác của YOLOv7.....	12
Hình 6. Yolov7 phát hiện khuôn mặt.....	14
Hình 7. Cấu trúc MTCNN.....	19
Hình 8. Các phép biến đổi hình học	15
Hình 9. Minh họa 68 điểm trên khuôn mặt con người.....	16
Hình 10. Kết quả Detecting facial landmarks qua camera	18
Hình 11. Lịch sử CNN trong nhận diện khuôn mặt	21
Hình 12. Lịch sử Loss Function trong nhận diện khuôn mặt	21
Hình 13. Minh họa CNN trong Triple Loss	22
Hình 14. Cấu trúc mạng Resnet.....	23
Hình 15. Triple Loss trong mạng Resnet	24
Hình 16. Giao diện lấy dữ liệu khuôn mặt.....	25
Hình 17. Một số hình ảnh trong thư mục với tên người đó	26
Hình 18. Trích rút đặc trưng khuôn mặt.....	26
Hình 19. Kết quả hiển thị khi người đó có trong bộ dữ liệu.....	27
Hình 20. Hiển thị kết quả khi người đó không có trong bộ dữ liệu.....	27

CHƯƠNG I. GIỚI THIỆU

1. Phát biểu bài toán

Hiện nay thị giác máy tính (computer vision) được áp dụng rộng rãi vào trong đời sống của con người. Một trong những ứng dụng phổ biến nhất của thị giác máy tính đó chính là nhận diện khuôn mặt. Bài toán nhận diện khuôn mặt là một bài toán phổ biến trong AI, nó có tên tiếng anh thường gọi là Face recognition. Bài toán nhận dạng khuôn mặt có thể áp dụng rộng rãi trong nhiều lĩnh vực khác nhau: Hệ thống điểm danh, chấm công, phát hiện tội phạm, tìm kiếm thông tin,...

Trong đề tài này, nhóm chúng em sẽ nghiên cứu các thuật toán liên quan đến nhận diện khuôn mặt và xây dựng hệ thống nhận diện khuôn mặt.

2. Công nghệ sử dụng

Để giải quyết bài toán này, chúng em đã sử dụng ngôn ngữ lập trình Python và các thư viện openCV, dlib, tkinter trên nền tảng Google Colab và Visual Studio Code.

2.1. Python

Python là một ngôn ngữ lập trình nâng cao, dùng để phát triển các ứng dụng cấp cao như: lập trình máy chủ web, tạo mẫu, phát triển trò chơi, đặc biệt là khoa học dữ liệu và học máy. Python hỗ trợ các module và package cho AI, computer vision.

Các đặc điểm của Python:

- Đơn giản, dễ học, dễ đọc
- Mã nguồn mở
- Hiệu suất cao
- Có thể tương tác với các module khác viết trên C/C++
- Có thể nhúng vào ứng dụng như một giao tiếp kịch bản.

2.2. Thư viện *openCV*

OpenCV (Open source computer vision library) là một thư viện mã nguồn mở hàng đầu cho thị giác máy tính (computer vision), xử lý ảnh và máy học, và các tính năng tăng tốc GPU trong hoạt động thời gian thực.

OpenCV cung cấp hơn 2500 thuật toán cổ điển và hiện đại, được sử dụng rộng rãi và hiệu quả trong các bài toán xử lý ảnh và video.

2.3. Thư viện *dlib*

Thư viện Dlib là một thư viện mã nguồn mở được sử dụng rộng rãi trong lĩnh vực thị giác máy tính và học máy. Nó cung cấp các công cụ và thuật toán mạnh mẽ để xử lý và phân tích ảnh, đặc biệt là trong việc nhận diện khuôn mặt, phát hiện đối tượng và theo dõi điểm đặc trưng.

Một trong những tính năng nổi bật của Dlib là khả năng nhận diện khuôn mặt chính xác và ổn định. Nó sử dụng các mô hình học máy và thuật toán như HOG (Histogram of Oriented Gradients), SVM (Support Vector Machines) và deep learning để xác định các điểm đặc trưng trên khuôn mặt và phân loại chúng. Điều này cho phép bạn nhận diện khuôn mặt, xác định các đặc điểm như mắt, mũi, miệng và xác định được vị trí và hướng của khuôn mặt.

Với tính linh hoạt và hiệu suất cao, Dlib đã trở thành một trong những thư viện quan trọng trong lĩnh vực thị giác máy tính và học máy, và được sử dụng rộng rãi trong các ứng dụng nhận diện khuôn mặt, phân loại ảnh và xử lý ảnh khác.

2.4. Thư viện *tkinter*

Thư viện Tkinter là một thư viện giao diện người dùng (GUI) mặc định của Python. Tkinter được xây dựng dựa trên thư viện Tk, một toolkit GUI đơn giản và mạnh mẽ. Với Tkinter, bạn có thể tạo ra các cửa sổ, nút bấm, hộp văn bản, hộp thoại, menu và nhiều thành phần giao diện khác để tương tác với người dùng. Nó

cung cấp các phương thức để xử lý sự kiện, truy cập và thay đổi các giá trị của các thành phần giao diện.

Tkinter dễ học và sử dụng, đặc biệt là đối với người mới học lập trình Python. Nó có tài liệu phong phú và cộng đồng phát triển mạnh mẽ, cung cấp các ví dụ và tài nguyên để hỗ trợ việc phát triển ứng dụng GUI.

2.5. *Google colab*

Google Colab là một dịch vụ miễn phí của Google cho phép bạn chạy và thực hiện các tác vụ lập trình trực tuyến thông qua trình duyệt web. Nó cung cấp môi trường Jupyter Notebook trên đám mây, cho phép bạn viết và chia sẻ mã nguồn Python một cách dễ dàng.

Một trong những lợi ích lớn nhất của Colab là khả năng sử dụng GPU và TPU miễn phí để thực hiện các tác vụ tính toán nặng. Điều này rất hữu ích khi bạn cần huấn luyện mô hình học máy phức tạp hoặc xử lý dữ liệu lớn.

Google Colab là một dịch vụ mạnh mẽ và tiện ích, cung cấp môi trường lập trình trực tuyến đáng tin cậy cho việc thực hiện các dự án khoa học dữ liệu, học máy và trí tuệ nhân tạo.

2.6. *Visual Studio Code*

Visual Studio Code (VSCode) là một trình soạn thảo mã nguồn mở và miễn phí được phát triển bởi Microsoft. Nó đã trở thành một trong những công cụ phổ biến nhất cho các nhà phát triển phần mềm và lập trình viên.

VSCode hỗ trợ đa nền tảng, tích hợp tốt với nhiều công cụ phát triển khác nhau và cung cấp trải nghiệm lập trình linh hoạt. Với sự kết hợp giữa tính năng mạnh mẽ và khả năng tùy chỉnh cao, VSCode đã trở thành một trong những lựa chọn phổ biến nhất cho việc phát triển phần mềm và lập trình.

CHƯƠNG II. CÔNG VIỆC TRIỂN KHAI

1. Học agile scrum course

Scrum là một phương pháp Agile (phát triển phần mềm linh hoạt) dựa trên cơ chế lặp và tăng trưởng. Scrum được thiết kế để hỗ trợ việc phát triển, cung cấp và cải tiến các sản phẩm phức tạp. Với Scrum, sản phẩm được xây dựng trong một chuỗi các quy trình lặp lại, có tên là vòng sprint. Qua đó, bạn có thể liên tục cải tiến sản phẩm, kỹ thuật, team (nhóm) và môi trường làm việc. Cũng nhờ vậy mà bạn có thể cung cấp giá trị cho khách hàng trong suốt quá trình phát triển.

Scrum là một khung tổ chức công việc (framework) dùng trong các dự án phát triển phần mềm với mục tiêu là chuyển giao các sản phẩm mới đều đặn, sau từ 1-4 tuần.

Ba giá trị cốt lõi của Scrum:

- Minh bạch.
- Thanh tra.
- Thích nghi.

Lợi ích Scrum mang lại:

- Cải thiện chất lượng phần mềm, dễ học và dễ sử dụng.
- Rút ngắn thời gian phát hành phần mềm, cho phép khách hàng sử dụng sản phẩm sớm hơn.
- Nâng cao tinh thần đồng đội, tối ưu hóa hiệu quả và nỗ lực của đội phát triển.
- Gia tăng tỷ suất hoàn vốn đầu tư (ROI).
- Tăng mức độ hài lòng của khách hàng.
- Kiểm soát dự án tốt, cải tiến liên tục.
- Giảm thiểu rủi ro khi xây dựng sản phẩm.

2. Phân tích bài toán nhận diện khuôn mặt

2.1. Khó khăn

Bài toán nhận diện khuôn mặt là một bài toán phức tạp trong lĩnh vực thị giác máy tính và học máy. Dưới đây là một số khó khăn chung mà người ta thường gặp phải khi làm việc với bài toán này:

Vấn đề chống giả mạo khuôn mặt: Với sự phát triển của các thiết bị công nghệ cũng như hiệu năng tính toán, xuất hiện ngày càng nhiều các dịch vụ sử dụng công nghệ nhận diện khuôn mặt. Kéo theo đó là những phương pháp cố gắng đánh lừa hệ thống này. Chống giả mạo khuôn mặt là một công việc khó khăn. Ngoài việc nhận diện xem người đó là ai hoặc người đó có tồn tại trong cơ sở dữ liệu của chúng ta không, còn phải xác định thêm người đó là thật hay là ảnh 2D hoặc 3D giả mạo. Với bài toán điểm danh sử dụng khuôn mặt trong lớp học, điều này càng quan trọng hơn.

Góc chụp: Với các tư thế khác nhau, các phần trên khuôn mặt có thể không rõ hoặc bị khuất hết. Với nhiều góc chụp, ví dụ chụp thẳng nhưng mặt lại quay về hướng khác hoặc là chụp một hướng khác của khuôn mặt đều là những khó khăn rất lớn trong việc nhận diện khuôn mặt.

Khuôn mặt thay đổi theo thời gian: Các chi tiết như râu ria mép hoặc kiểu tóc có thể thay đổi trong thời gian ngắn có thể làm thay đổi kết quả của bài toán. Ngoài ra còn gặp khó khăn khi khuôn mặt có nhiều trạng thái khác nhau, ví dụ như cùng một khuôn mặt nhưng có trạng thái rất khác nhau khi vui, buồn.

Điều kiện ảnh: Ảnh được chụp trong các điều kiện khác nhau: ánh sáng, các thiết bị khác nhau, nền ảnh phức tạp.

Trên đây đã đưa ra một số những khó khăn gặp phải khi thực hiện bài toán này. Để vượt qua những khó khăn này, người ta thường sử dụng các phương pháp và mô hình như deep learning, học máy, phân tích đặc trưng, và kỹ thuật xử lý ảnh để tăng cường độ chính xác và độ tin cậy của hệ thống nhận diện khuôn mặt.

2.2. Các vấn đề cần giải quyết

Bài toán nhận diện khuôn mặt (Face recognition) bao gồm các bài toán khác nhau như phát hiện khuôn mặt (Face detection), đánh dấu (Facial landmarking), trích chọn đặc trưng (Feature extration), gán nhãn, phân lớp (Classification).

Các vấn đề cần giải quyết:

Bài toán Face Recognition bắt buộc phải bao gồm tối thiểu 3 bước sau:

- Bước 1: Face Detection - Xác định vị trí của khuôn mặt trong ảnh (hoặc video frame). Vùng này sẽ được đánh dấu bằng một hình chữ nhật bao quanh.
- Bước 2: Face Extraction (Face Embedding) - Trích xuất đặc trưng của khuôn mặt thành một vector đặc trưng trong không gian nhiều chiều (thường là 128 chiều).
- Bước 3: Face Classification (Face Authentication - Face Verification - Face Identification).

Ngoài 3 bước trên, trong thực tế chúng ta thường bổ sung thêm một số bước để tăng độ chính xác nhận diện:

- Image Preprocessing: Xử lý giảm nhiễu, giảm mờ, giảm kích thước, chuyển sang ảnh xám, chuẩn hóa, ...
- Face Aligment: Nếu ảnh khuôn mặt bị nghiêng thì căn chỉnh lại sao cho ngay ngắn.
- Kết hợp nhiều phương pháp khác nhau tại bước 3.

Xác định được các vấn đề cần giải quyết, chúng ta sẽ tìm hiểu và giải quyết từng vấn đề.

3. Phát hiện khuôn mặt

Phát hiện khuôn mặt là quá trình tự động định vị khuôn mặt người trong phương tiện trực quan (hình ảnh kỹ thuật số hoặc video). Một khuôn mặt được phát hiện được báo cáo tại một vị trí có kích thước và hướng liên quan. Một khi khuôn mặt được phát hiện, nó có thể được tìm kiếm các điểm mốc như mắt và mũi.



Hình 1. Phát hiện khuôn mặt trong ảnh

Bất kì camera nào trong khoảng 7 năm trở lại đây thì bạn có thể thấy nhận diện khuôn mặt được tích hợp sẵn trong thiết bị. Nhận diện khuôn mặt là một tính năng tuyệt vời cho máy ảnh. Khi máy ảnh có thể tự động chọn ra các khuôn mặt, chúng có thể chắc chắn rằng tất cả các khuôn mặt được là nét trước khi nó ghi hình.

Nhận diện khuôn mặt đã trở thành xu hướng vào đầu những năm 2000 khi Paul Viola và Michael Jones phát minh ra cách để nhận diện khuôn mặt với tốc độ đủ để chạy trên con các dòng máy ảnh rẻ tiền. Tuy nhiên thì còn có các phương pháp đáng tin cậy cũng đã xuất hiện. Chúng ta sử dụng phương pháp được phát minh năm 2005 được gọi là “Histogram of Oriented Gradients” (rút gọn thành HOG)

Để tìm những khuôn mặt trong một tấm hình, chúng ta bắt đầu với việc biến tấm ảnh của chúng ta thành tấm ảnh xám, chỉ có đen và trắng vì chúng ta không cần màu sắc để tìm khuôn mặt.

Sau đó chúng ta nhìn vào từng pixel trong tấm hình cùng một lúc. Với mỗi pixel một, chúng ta lại nhìn vào những pixel lân cận nó. Mục đích của chúng ta là tìm ra pixel hiện tại có màu tối như thế nào so với các pixel lân cận nó. Khi đó chúng ta sẽ vẽ một mũi tên theo chiều mà màu trở nên tối hơn

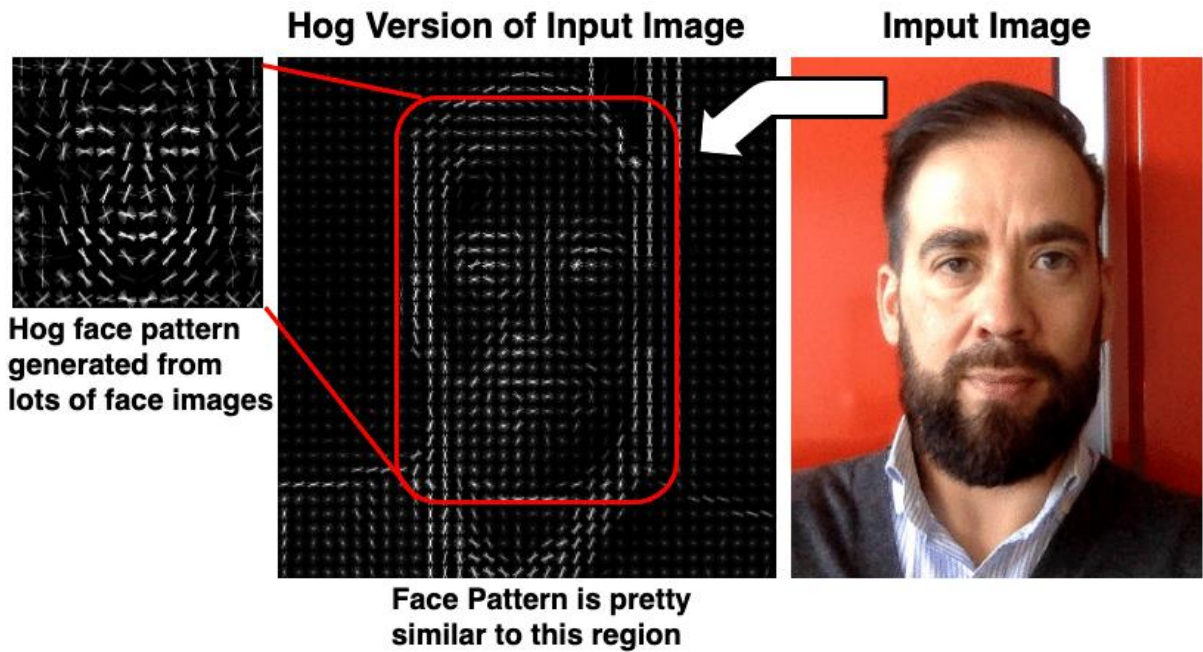
Nếu bạn lặp lại tiến trình đó với mỗi pixel một trong tấm ảnh, bạn sẽ kết thúc với mỗi pixel được thay thế bởi một mũi tên. Những mũi tên đó được gọi là “gradients”(vector độ dốc) và chúng chỉ ra dòng chảy(lưu lượng) từ những pixel sáng đến những pixel tối trên toàn bộ hình ảnh.

Điều này có vẻ là một việc làm ngẫu nhiên, nhưng đó là một lí do tốt để thay thế các pixel đó thành gradients. Nếu chúng ta phân tích trực tiếp các pixel tối và các pixel sáng trong bức ảnh của cùng một người sẽ nhận được tổng các giá trị hoàn toàn khác nhau. Nhưng nếu xem xét hướng sáng thay đổi, cả hai hình ảnh tối và hình ảnh sáng sẽ cho kết quả với cùng một đại diện(con số) chính xác. Điều đó làm cho vấn đề dễ giải quyết hơn.

Nhưng việc lưu gradient cho mỗi pixel đơn lẻ cho chúng ta quá nhiều chi tiết. Sẽ tốt hơn nếu chúng ta có thể nhìn thấy dòng chảy sáng / tối cơ bản ở mức cao hơn để chúng ta có thể thấy mô hình cơ bản của hình ảnh.

Để làm điều này, chúng tôi sẽ chia hình ảnh thành các ô vuông nhỏ 16x16 pixel mỗi hình. Trong mỗi ô vuông, chúng tôi sẽ đếm xem có bao nhiêu độ dốc theo từng hướng chính (có bao nhiêu điểm hướng lên, hướng lên phải, điểm phải, v.v ...). Sau đó, chúng tôi sẽ thay thế hình vuông đó trong hình ảnh bằng các hướng mũi tên nổi bật nhất.

Kết quả cuối cùng là chúng ta biến hình ảnh gốc thành một hình đại diện rất đơn giản, nắm bắt cấu trúc cơ bản của khuôn mặt một cách đơn giản hơn:



Hình 2. Phát hiện khuôn mặt qua HOG

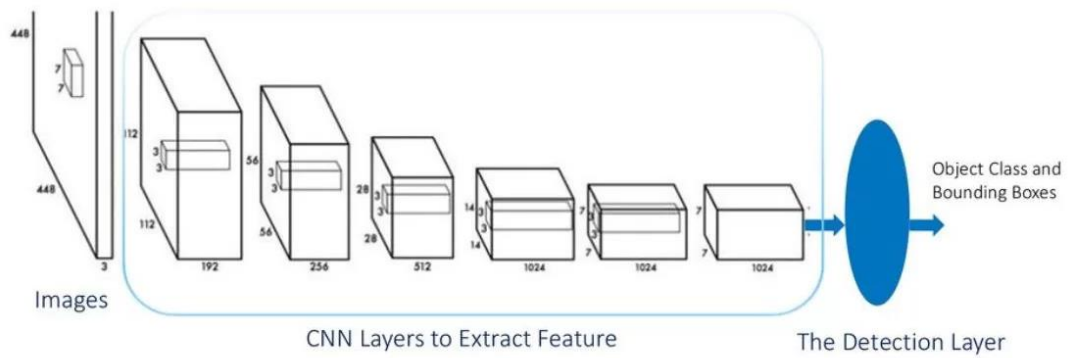
4. Tìm hiểu về YOLOv7

4.1. Giới thiệu về YOLO

YOLO (You Only Look Once) là mô hình phát hiện đối tượng phổ biến được biết đến với tốc độ nhanh và độ chính xác cao. Mô hình này lần đầu tiên được giới thiệu bởi Joseph Redmon và cộng sự vào năm 2016. Kể từ đó đến nay, đã có nhiều phiên bản của YOLO, một trong những phiên bản gần đây nhất là YOLOv7.

4.2. Hoạt động của YOLO

Thuật toán YOLO lấy hình ảnh làm đầu vào, sau đó sử dụng mạng nơ-ron tích chập sâu đơn giản để phát hiện các đối tượng trong ảnh. Kiến trúc của mô hình CNN tạo thành xương sống của YOLO được hiển thị bên dưới.



Hình 3. Cấu trúc của YOLO

20 lớp tích chập đầu tiên của mô hình được đào tạo trước với ImageNet bằng cách cắm vào một lớp tổng hợp trung bình tạm thời (temporary average pooling) và lớp được kết nối đầy đủ (fully connected layer). Sau đó, mô hình đào tạo trước này được chuyển đổi để thực hiện phát hiện. Lớp được kết nối đầy đủ cuối cùng của YOLO dự đoán cả xác suất của lớp và tọa độ hộp giới hạn.

YOLO chia hình ảnh đầu vào thành lưới $S \times S$. Nếu tâm của một đối tượng rơi vào một ô lưới thì ô lưới đó có nhiệm vụ phát hiện đối tượng đó. Mỗi ô lưới dự đoán các hộp giới hạn B và điểm tin cậy cho các hộp đó. Các điểm tin cậy này phản ánh mức độ tin cậy của mô hình rằng hộp chứa một đối tượng và mức độ chính xác mà mô hình cho rằng hộp được dự đoán.

YOLO dự đoán nhiều hộp giới hạn trên mỗi ô lưới. Tại thời điểm đào tạo, ta chỉ muốn một bộ dự đoán hộp giới hạn thể hiện cho từng đối tượng. YOLO chỉ định bộ dự đoán dựa trên chỉ số IOU hiện tại cao nhất với thực tế. Điều này dẫn đến sự chuyên môn hóa giữa các bộ dự đoán hộp giới hạn. Mỗi công cụ dự đoán trở nên tốt hơn trong việc dự báo các kích thước, tỷ lệ khung hình hoặc loại đối tượng nhất định, cải thiện tổng thể recall score.

Một kỹ thuật quan trọng được sử dụng trong các mô hình YOLO là NMS (non-maximum suppression). NMS là một bước hậu xử lý được sử dụng để cải thiện độ chính xác và hiệu quả của việc phát hiện đối tượng. Trong phát hiện đối tượng, thông thường có nhiều hộp giới hạn được tạo cho một đối tượng trong một

hình ảnh. Các hộp giới hạn này có thể chồng lên nhau hoặc nằm ở các vị trí khác nhau, nhưng tất cả chúng đều đại diện cho cùng một đối tượng. NMS được sử dụng để xác định và loại bỏ các hộp giới hạn dư thừa hoặc không chính xác và để xuất một hộp giới hạn duy nhất cho từng đối tượng trong ảnh.

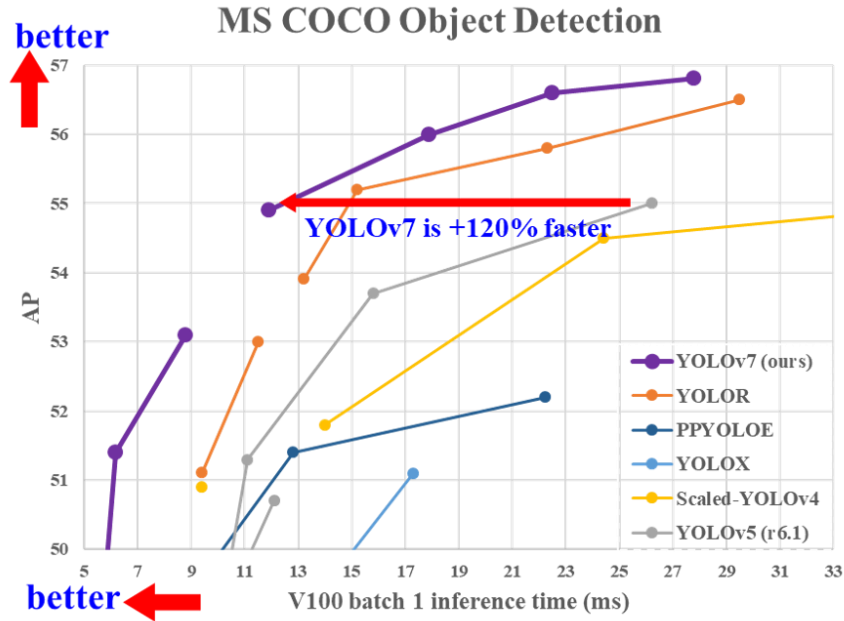
4.3. YOLOv7

YOLOv7, phiên bản mới nhất của YOLO, có một số cải tiến so với các phiên bản trước. Một trong những cải tiến chính là việc sử dụng các anchor box.

Các anchor box là một tập hợp các hộp được xác định trước với các tỷ lệ khung hình khác nhau được sử dụng để phát hiện các đối tượng có hình dạng khác nhau. YOLOv7 sử dụng chín anchor box, cho phép YOLO phát hiện phạm vi hình dạng và kích thước đối tượng rộng hơn so với các phiên bản trước, do đó giúp giảm số lượng xác định sai.

Một cải tiến quan trọng trong YOLOv7 là việc sử dụng một loss function mới gọi là “focal loss”. Các phiên bản trước của YOLO đã sử dụng cross-entropy loss function tiêu chuẩn, được biết là kém hiệu quả hơn trong việc phát hiện các đối tượng nhỏ. Focal loss giải quyết vấn đề này bằng cách giảm trọng số mất mát cho các ví dụ được phân loại tốt và tập trung vào các ví dụ khó—các đối tượng khó phát hiện.

Một trong những ưu điểm chính của YOLOv7 là tốc độ. Nó có thể xử lý hình ảnh với tốc độ 155 khung hình mỗi giây, nhanh hơn nhiều so với các thuật toán phát hiện đối tượng hiện đại khác. Ngay cả mô hình YOLO cơ bản ban đầu cũng có khả năng xử lý ở tốc độ tối đa 45 khung hình mỗi giây. Điều này làm cho nó phù hợp với các ứng dụng thời gian thực nhạy cảm như giám sát và ô tô tự lái, trong đó tốc độ xử lý cao hơn là rất quan trọng.



Hình 4. So sánh tốc độ của YOLOv7 với các phiên bản trước

Về độ chính xác, YOLOv7 thể hiện tốt so với các thuật toán phát hiện đối tượng khác. Nó đạt được độ chính xác trung bình là 73,8% ở ngưỡng IoU (giao điểm trên hợp nhất) là 0,5 trên bộ dữ liệu COCO phổ biến, có thể so sánh với các thuật toán phát hiện đối tượng hiện đại khác. So sánh định lượng của hiệu suất được hiển thị dưới đây.

Model	#Param.	FLOPs	Size	AP ^{val}	AP ^{val} ₅₀	AP ^{val} ₇₅	AP ^{val} _S	AP ^{val} _M	AP ^{val} _L
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOv4-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOv4-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	51.2%	69.7%	55.5%	35.2%	56.0%	66.7%
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOv4-CSP-X [81]	96.9M	226.8G	640	52.7%	71.3%	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	52.9%	71.1%	57.5%	36.9%	57.7%	68.6%
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	35.2%	52.8%	37.3%	15.7%	38.0%	53.4%
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	10.9%	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	30.8%	47.3%	32.2%	10.0%	31.9%	52.2%
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOv4-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	60.4%	69.2%
YOLOv7-E6	97.2M	515.2G	1280	55.9%	73.5%	61.1%	40.6%	60.3%	70.0%
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOv4-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	42.4%	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	56.8%	74.4%	62.1%	40.8%	62.1%	70.6%
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

Hình 5. Độ chính xác của YOLOv7

4.4. Hạn chế của YOLOv7

- YOLOv7 là một thuật toán phát hiện đối tượng mạnh mẽ và hiệu quả, nhưng nó có một số hạn chế.
- YOLOv7, giống như nhiều thuật toán phát hiện đối tượng, gặp khó khăn trong việc phát hiện các đối tượng nhỏ. Nó có thể không phát hiện chính xác các đối tượng trong các cảnh đông đúc hoặc khi các đối tượng ở xa máy ảnh.
- YOLOv7 cũng không hoàn hảo trong việc phát hiện các đối tượng ở các tỷ lệ khác nhau. Điều này có thể gây khó khăn cho việc phát hiện các đối tượng rất lớn hoặc rất nhỏ so với các đối tượng khác trong cảnh.
- YOLOv7 có thể nhạy cảm với những thay đổi về ánh sáng hoặc các điều kiện môi trường khác, vì vậy có thể bất tiện khi sử dụng trong các ứng dụng thực, nơi điều kiện ánh sáng có thể thay đổi.
- YOLOv7 có thể đòi hỏi nhiều tính toán, điều này gây khó khăn khi chạy trong thời gian thực trên các thiết bị hạn chế về tài nguyên như điện thoại thông minh hoặc các thiết bị biên khác.

Training model YOLOv7 phát hiện khuôn mặt

- Bộ dữ liệu: Wider face

Bộ dữ liệu WIDER FACE là bộ dữ liệu điểm chuẩn phát hiện khuôn mặt, trong đó hình ảnh được chọn từ bộ dữ liệu WIDER có sẵn công khai. Chúng tôi chọn 32.203 hình ảnh và gắn nhãn cho 393.703 khuôn mặt có mức độ thay đổi cao về tỷ lệ, tư thế và khớp căn như được mô tả trong các hình ảnh mẫu. Bộ dữ liệu WIDER FACE được tổ chức dựa trên 61 lớp sự kiện. Đối với mỗi lớp sự kiện, chúng tôi chọn ngẫu nhiên dữ liệu 40%/10%/50% làm tập huấn luyện, xác thực và kiểm tra.

- Kết quả train model sau 55 epoch:

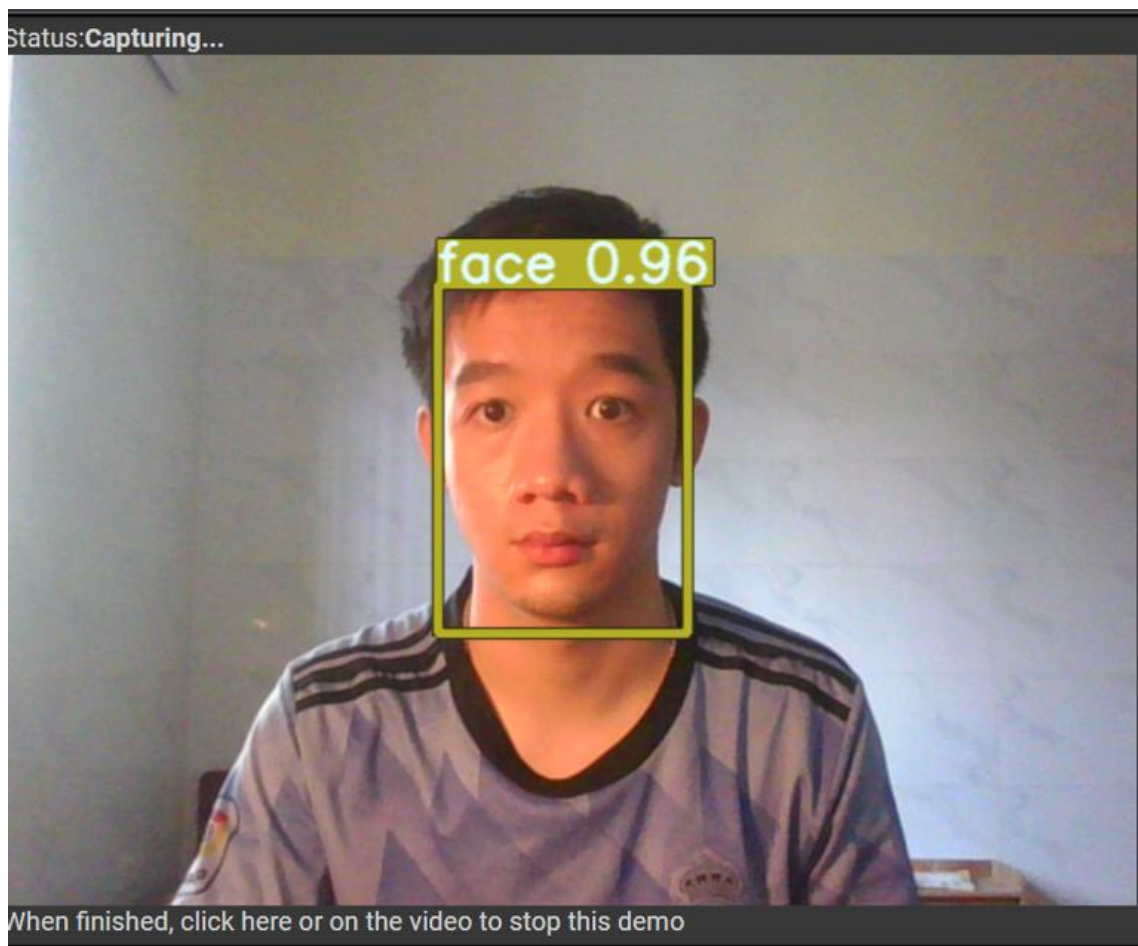
Epoch	gpu_mem	box	obj	cls	total	labels	img_size			
54/54	13.9G	0.01839	0.009006	0	0.02739	42	640:	100%	139/139	[02:21<00:00, 1.02s/it]
	Class	Images	Labels		P	R	mAP@.5	mAP@.5:.95:	100%	70/70 [00:46<00:00, 1.51it/s]
	all	2211	11321		0.98	0.962	0.985		0.831	

Giá trị Precision đạt 0.98.

Giá trị Recall đạt 0.96.

Giá trị trung bình chính xác (Average Precision) tại ngưỡng IoU (Giao nhau qua Hợp) đạt 0.98.

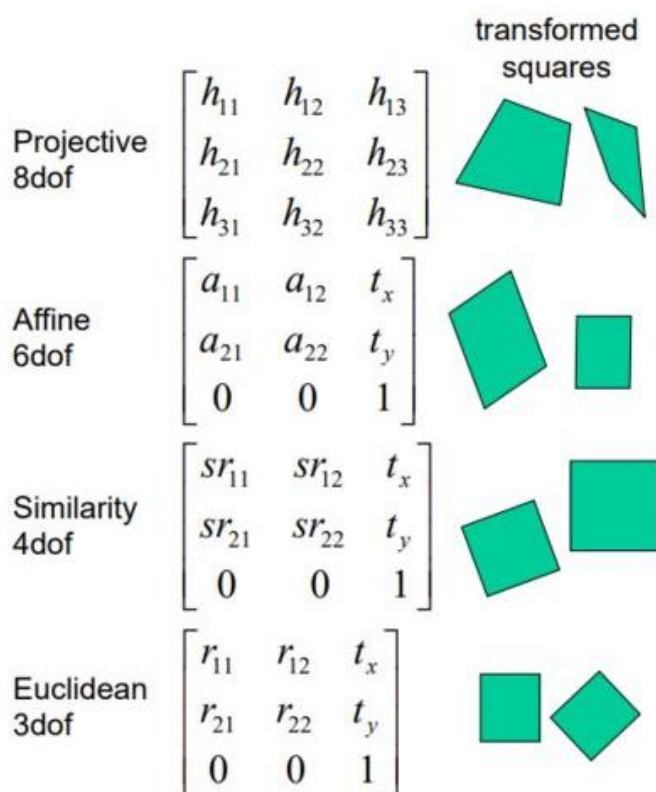
Nhìn chung, mô hình có vẻ đang hoạt động tốt, với các giá trị độ chính xác và độ nhớ cao và một giá trị trung bình chính xác trung bình (mAP) cao.



Hình 6. YOLOv7 phát hiện khuôn mặt

5. Tìm hiểu về Facial Landmarks

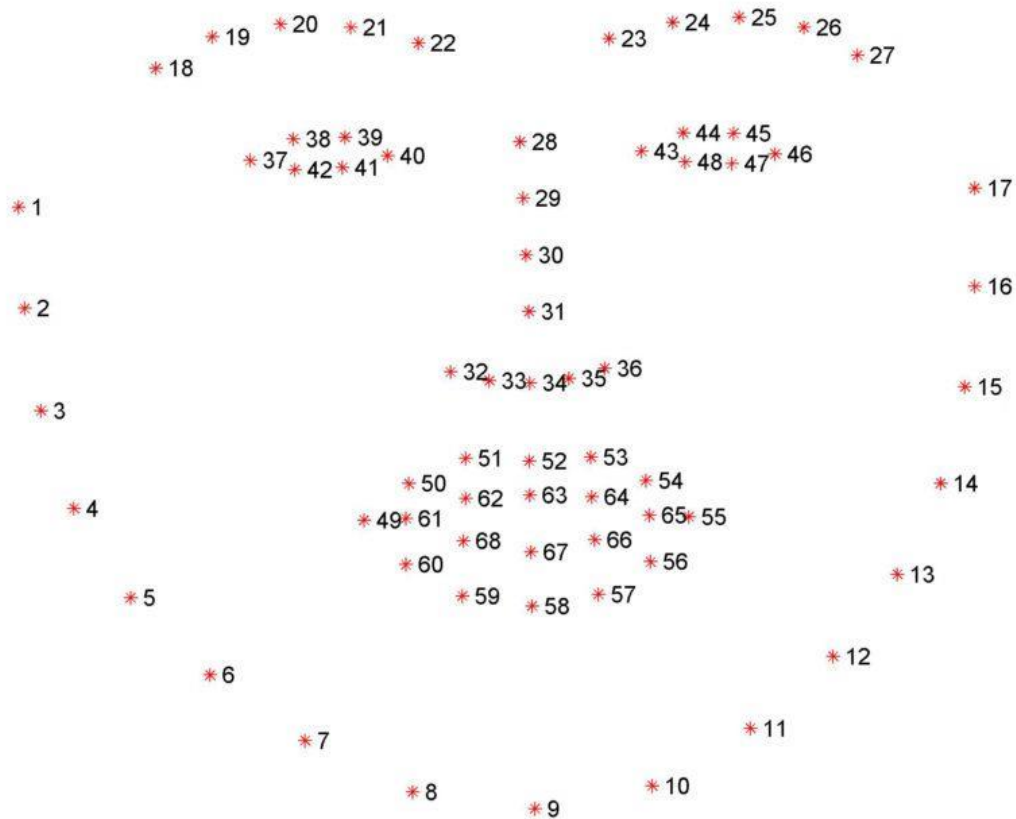
Sau khi phát hiện và xác định được vị trí khuôn mặt, vùng khuôn mặt được tách ra từ ảnh có thể ở các trạng thái khác nhau, các góc độ khác nhau. Có những khuôn mặt bị chéo và cũng có thể bị lệch do bước detect chưa chính xác trong việc lấy khung hình chuẩn của mặt. Vì thế việc căn chỉnh lại khuôn mặt là cần thiết để nâng cao độ chính xác khi nhận diện khuôn mặt. Để làm được điều này, chúng ta cần xác định được những điểm đặc biệt trên khuôn mặt như mắt, mũi, ... từ đó áp dụng các phương pháp biến đổi hình học trong toán học như: *Euclidean Transformation*, *Similarity Transformation*, *Affine Transformation*, *Projective Transformation* để khuôn mặt trở về theo quy chuẩn.



Hình 7. Các phép biến đổi hình học

Để làm được điều này cần xác định được cấu trúc khuôn mặt. Có rất nhiều kiểu cấu trúc khuôn mặt khác nhau nhưng về cơ bản, chúng ta sẽ xác định các phần miệng, lông mày phải, lông mày trái, mắt phải, mắt trái, mũi, quai hàm. Ý

tương cơ bản là chúng ta sẽ đưa ra 68 điểm cụ thể (được gọi là các mốc) tồn tại trên mỗi khuôn mặt - đỉnh cằm, cạnh ngoài của mỗi mắt, cạnh trong của mỗi lông mày, v.v. Sau đó, chúng ta sẽ huấn luyện một máy học thuật toán để có thể tìm thấy 68 điểm cụ thể này trên mọi mặt:



Hình 8. Minh họa 68 điểm trên khuôn mặt con người

The pre-trained facial landmark detector trong dlib được sử dụng để ước lượng 68 (x, y) -coordinates tương ứng với tọa độ các facial landmarks trên khuôn mặt.

Detecting facial landmarks with dlib, OpenCv and Python

Bước 1: Xác định vị trí faces trong bức ảnh (thông qua một số method of face detection)

```
detector = dlib.get_frontal_face_detector()
rects = detector(gray, 1)
```

- Detector nhận vào hai tham số:
 - Grayscale image
 - Tham số thứ hai là upsample image, mặc định để 1. Đây chính là số lần chúng ta upsample image trong quá trình detect faces giúp phát hiện các khuôn mặt có kích thước nhỏ. Tuy nhiên nếu tham số lớn quá sẽ khiến mô hình chạy chậm.
- Câu lệnh số 2 trả về list các rectangle tương ứng với các khuôn mặt phát hiện được. Lưu ý trong dlib rectangle trả lại có dạng (left, top, right, bottom) tương đương với (xmin, ymin, xmax, ymax). Ở đây có sử dụng hàm chuyển đổi rect này về dạng bounding box hay sử dụng trong OpenCV cho tiện (xmin, ymin, w, h). Thư viện imutils có tích hợp sẵn các hàm chuyển đổi này.

Bước 2: Sử dụng shape predictor để có được tọa độ (x,y) của các facial landmarks

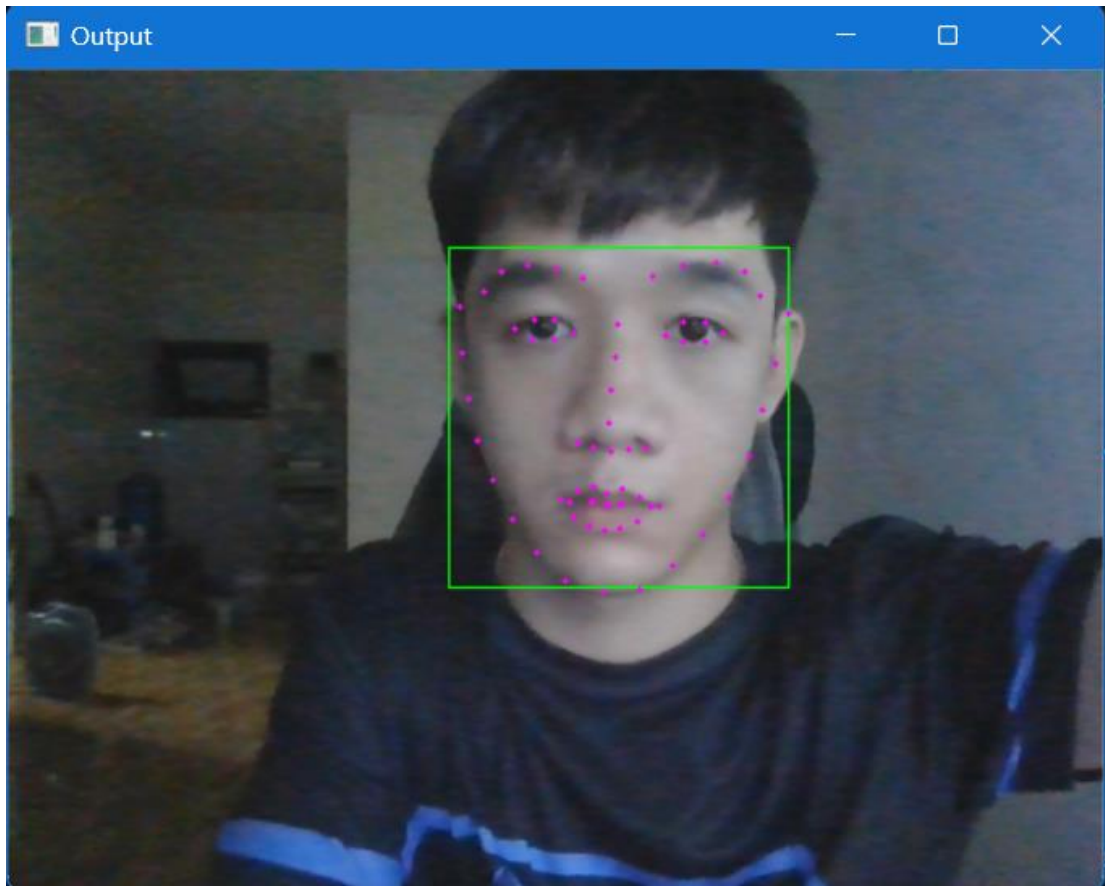
```
predictor = dlib.shape_predictor(args["shape_predictor"])
shape = predictor(gray, rect)
```

Đầu tiên chúng ta cần tải pre-trained model cho các facial landmark. Ở câu lệnh số 2 truyền vào 2 tham số là ảnh ban đầu và rect (từng rectangle cho từng khuôn mặt). Kết quả trả về là shape có chứa thông tin của 68 facial landmarks. Tọa độ của mỗi facial landmark có thể được trích xuất thông qua câu lệnh sau:

```
for i in range(68):
    x, y = shape.part(i).x, shape.part(i).y
```

Để đơn giản hơn chúng ta có thể chuyển đổi các tọa độ này về 2d-numpy array có chiều là (68, 2) - 68 là số điểm, 2 tương ứng hai tọa độ x, y cho mỗi điểm. Tương tự như trên thư viện imutils có tích hợp sẵn các hàm chuyển đổi này.

Kết quả:



Hình 9. Kết quả Detecting facial landmarks qua camera

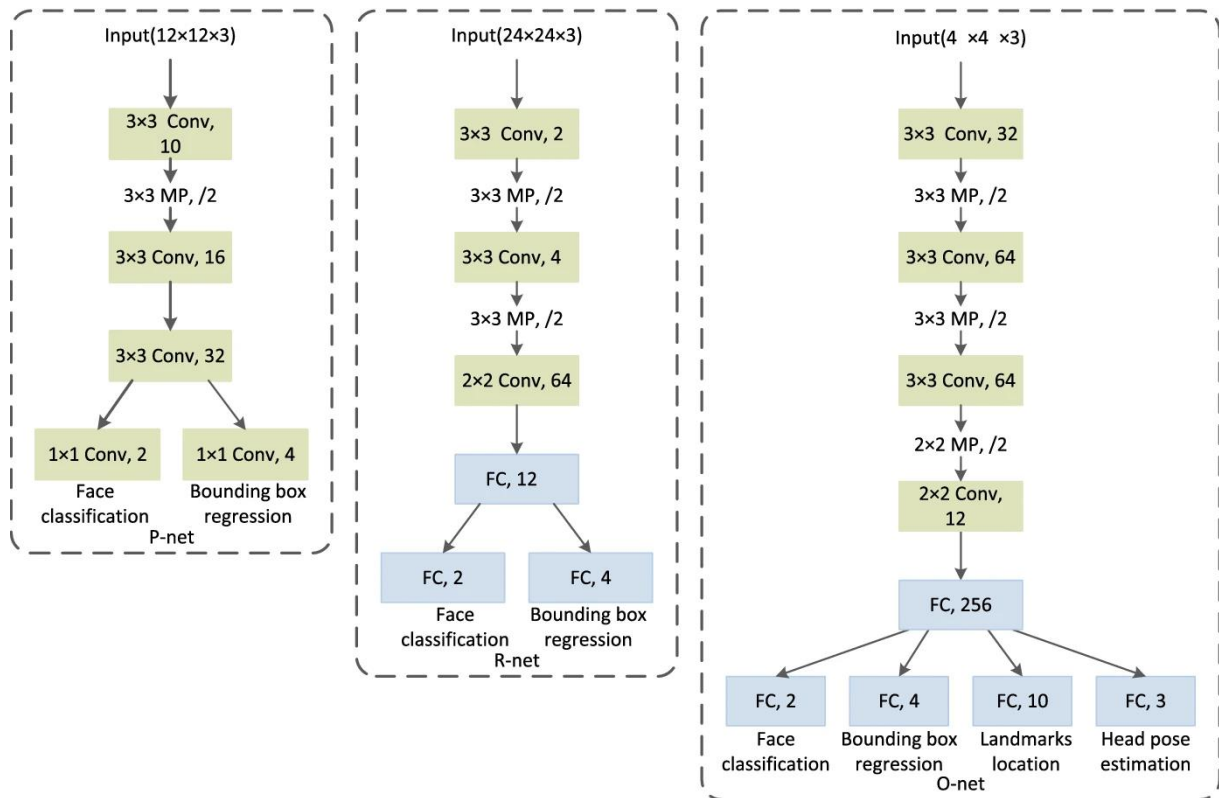
6. Tìm hiểu về MTCNN

6.1. Giới thiệu MTCNN

MTCNN (Multi – Cascaded Convolution Networks) là một mô hình được sử dụng phổ biến trong lĩnh vực nhận dạng khuôn mặt và phát hiện khuôn mặt trong ảnh. Nó được xây dựng trên cơ sở của Convolutional Neural Networks (CNN) và kết hợp nhiều tác vụ như phát hiện khuôn mặt (face detection), xác định các điểm đặc trưng trên khuôn mặt (landmark localization) và xác định vùng khuôn mặt (face alignment).

6.2. Cấu trúc MTCNN

MTCNN gồm 3 thành phần chính: P-net, R-net và O-net.



Hình 10. Cấu trúc MTCNN

- Phần P-net: nhận dạng toàn bộ số khuôn mặt trong bức ảnh bằng cách tạo nhiều bản copy từ ảnh gốc với kích thước khác nhau tạo thành một Image Pyramid để mạng dễ dàng nhận diện các khuôn mặt. Sau đó thiết lập ngưỡng và loại bỏ các hộp giới hạn có tỷ lệ trùng nhau.
- Phần R-net: cũng thực hiện như P-net và tinh chỉnh lại những hộp giới hạn.
- Phần O-net: xác định vị trí khuôn mặt và những điểm quan trọng trên khuôn mặt.

6.3. Ưu điểm của MTCNN

- Độ chính xác cao: MTCNN sử dụng một quy trình phát hiện khuôn mặt theo các lớp liên tiếp, giúp tăng cường độ chính xác của quá trình phát hiện khuôn mặt.
- Xử lý đa tác vụ: MTCNN có khả năng xác định vùng khuôn mặt, phát hiện khuôn mặt và xác định các điểm đặc trưng trên khuôn mặt.

trong một bước duy nhất, giúp tiết kiệm thời gian và tăng tốc độ xử lý.

- Thích ứng với tỷ lệ kích thước khuôn mặt: MTCNN có khả năng phát hiện và xử lý các khuôn mặt có kích thước khác nhau, từ nhỏ đến lớn.
- Tích hợp mô hình phát hiện khuôn mặt và xác định điểm đặc trưng: MTCNN không chỉ phát hiện khuôn mặt mà còn xác định các điểm đặc trưng như mắt, mũi, miệng. Điều này hữu ích cho các tác vụ như nhận dạng khuôn mặt hoặc phân tích biểu cảm khuôn mặt.

6.4. Hạn chế của MTCNN

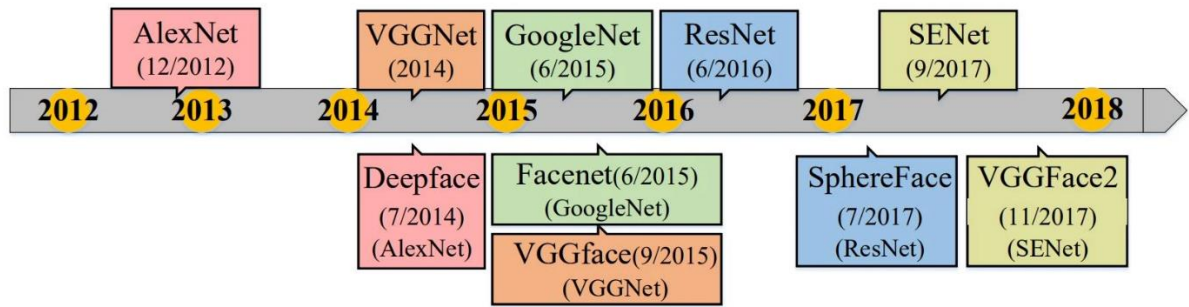
Đòi hỏi tài nguyên tính toán lớn: MTCNN dựa trên mạng CNN với nhiều lớp và tham số, do đó yêu cầu tài nguyên tính toán cao và thời gian xử lý tương đối lâu đối với ảnh có kích thước lớn.

Độ chính xác bị ảnh hưởng bởi điều kiện ánh sáng và góc nhìn: Như hầu hết các thuật toán phát hiện khuôn mặt, MTCNN cũng có thể bị ảnh hưởng bởi điều kiện ánh sáng yếu hoặc góc nhìn khó khăn, dẫn đến giảm độ chính xác.

7. Face Classification

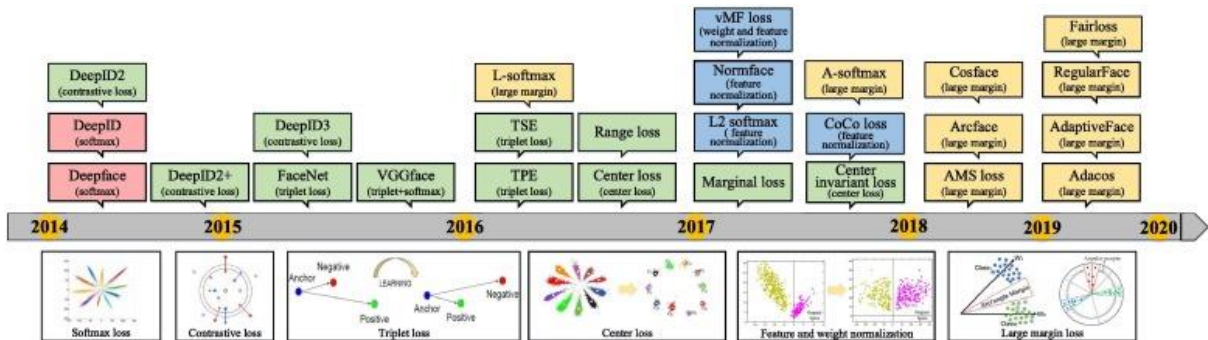
Sau khi căn chỉnh khuôn mặt, bước tiếp theo là trích chọn đặc trưng của khuôn mặt phục vụ cho nhận diện khuôn mặt. Trích xuất đặc trưng của khuôn mặt là một bước rất quan trọng trong các công nghệ như face tracking, nhận dạng cảm xúc khuôn mặt hay nhận diện khuôn mặt. Ảnh khuôn mặt sẽ được đi qua một bộ trích chọn đặc trưng mà đầu ra ở đây là một vector biểu thị cho khuôn mặt. Có rất nhiều phương pháp để trích chọn đặc trưng của một bức ảnh.

Các phương pháp truyền thống thường sử dụng các bộ trích chọn đặc trưng hand-craft feature (đặc trưng cứng) như: HOG, SIFT, LBP,... Tuy nhiên, các bộ trích chọn đặc trưng sử dụng mạng CNN lại có hiệu quả vượt bậc so với các bộ trích chọn đặc trưng truyền thống.



Hình 11. Lịch sử CNN trong nhận diện khuôn mặt

Cùng với sự phát triển của các kiến trúc mạng là sự phát triển của các hàm LossFunction để tối ưu kết quả cho bài toán. Nhiều loss function có thể kể đến như Euclidean Distance Based Loss, Angular/Cosine-Margin-Based Loss và SoftmaxLoss và một số biến thể của nó.



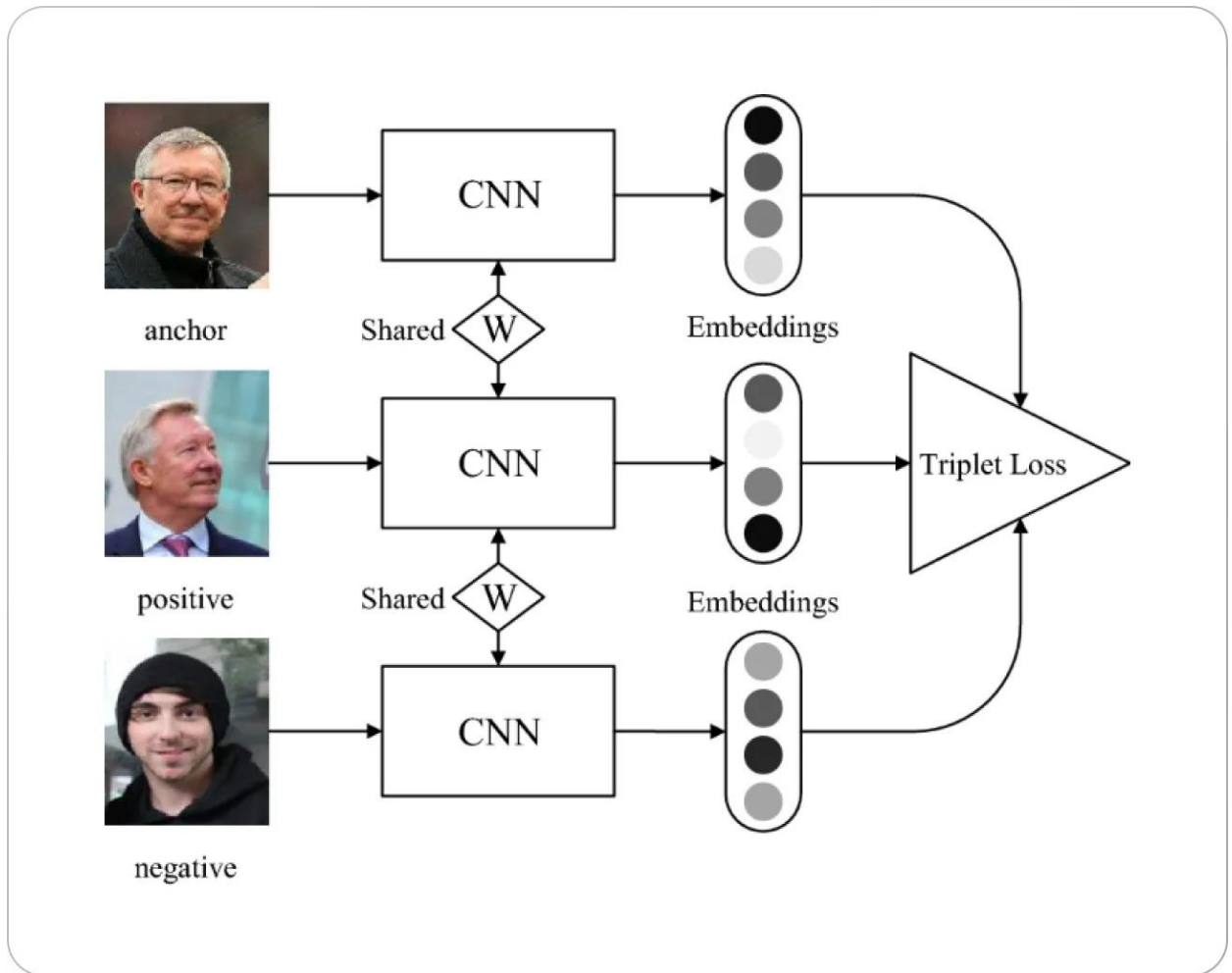
Hình 12. Lịch sử Loss Function trong nhận diện khuôn mặt

Triple loss:

Triple theo tiếng việt được gọi là “bộ ba”, thông thường, chúng ta sẽ cho lần lượt các ảnh một cách lần lượt và mô hình học sâu để train model, nhưng ở mô hình triple này chúng ta phải sử dụng từ “bộ ba”. Bộ ba của chúng ta bao gồm một ảnh của một người bất kỳ (query), một ảnh của người đó (positive), và một ảnh mặt của người khác (negative), để mô hình có thêm thông tin về bộ ba ảnh này để phù hợp với bài toán hơn.

Thay vì lấy hai giá trị đầu vào, Triple Loss đưa ra một công thức mới gồm 3 giá trị đầu vào: Anchor (ảnh đầu vào của mạng), Positive (ảnh cùng là của một người với Anchor), Negative (ảnh không cùng là một người với Anchor). Sau đó các ảnh đầu vào sẽ được trích xuất đặc trưng thành các embedding (embedding là

một vector với số chiều cố định nhỏ hơn feature vector bình thường, đại diện cho các feature và được dùng trong nhiệm vụ phân loại các đối tượng) rồi thực hiện phân loại.



Hình 13. Minh họa CNN trong Triple Loss

Công thức hàm Triple loss

$$\sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]$$

Hàm liên hệ giữa 2 embedding của Anchor và Positive với Anchor và Negative phải nhỏ hơn ít nhất một số. Để mạng học được nhiều hơn, điểm Positive được chọn phải nằm xa nhất có thể so với Anchor và điểm Negative phải nằm gần nhất có thể so với Anchor là những trường hợp xấu nhất mà mạng học cần vượt qua. Như vậy khi áp dụng Triple loss vào các mô hình convolutional neural network chúng ta có thể tạo ra các biểu diễn vector tốt nhất cho mỗi một bức ảnh.

Những biểu diễn vector này sẽ phân biệt tốt các ảnh Negative rất giống ảnh Positive. Và đồng thời các bức ảnh thuộc cùng một label sẽ trở nên gần nhau hơn trong không gian chiều euclidean.

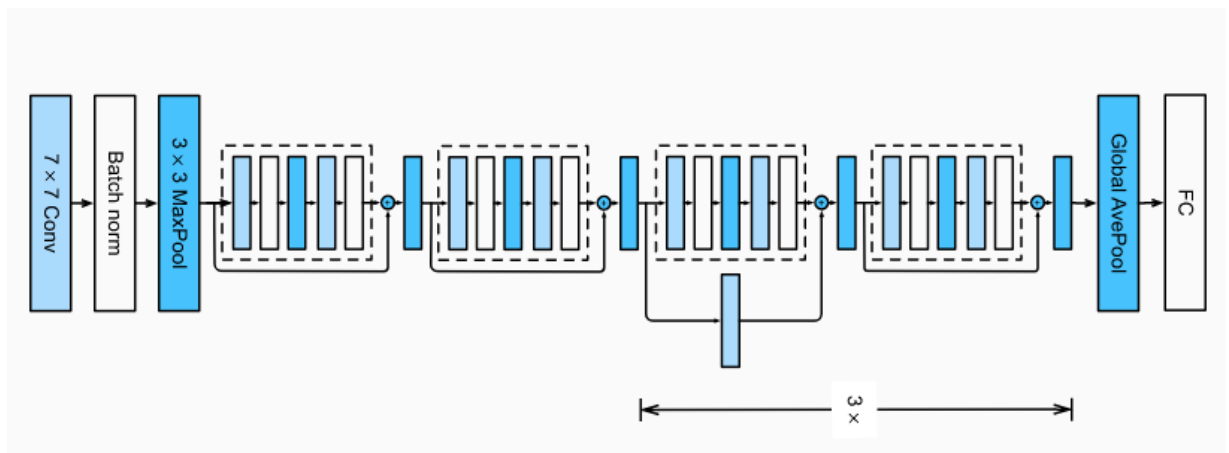
Tuy nhiên, khi gặp dataset lớn sẽ có sự bùng nổ về số lượng bộ ba tripletsamples dẫn đến sự lặp lại đáng kể các bước.

Resnet:

Công cụ Dlib dùng để extract feature sử dụng mô hình mạng ResNet-29 conv layers và triplet loss. Mạng được train trên bộ dữ liệu khoảng 3 triệu khuôn mặt The face scrub dataset, the VGG dataset, và các ảnh trên internet (Tổng số người là 7485).

Mô hình có độ chính xác lên đến 99,38% thử nghiệm trên các khuôn mặt của bộ dữ liệu LFW có 5749 ID với 64811 ảnh.

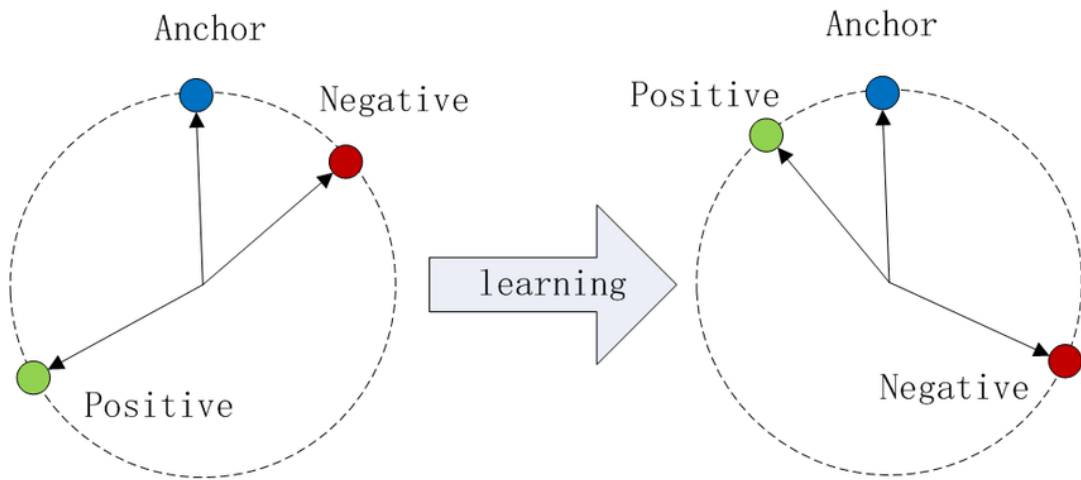
Mạng ResNet (R) - 29 là một mạng CNN được thiết kế để làm việc với 29 lớp chập. ResNet gồm có các lớp như convolution, pooling, activation và fully connected.



Hình 14. Cấu trúc mạng Resnet

Triple loss: Quá trình encoding của mạng CNN đã giúp ta mã hóa bức ảnh về 128 chiều. Sau đó những vectơ này sẽ làm đầu vào cho hàm loss function đánh giá khoảng cách giữa các véc tơ. Mục tiêu của hàm loss function là tối thiểu hóa

khoảng cách giữa 2 ảnh khi chúng là positive và tối đa hóa khoảng cách khi chúng là negative.



Hình 15. Triple Loss trong mạng Resnet

$$\mathcal{L}(A, P, N) = \sum_i^n \max (\|f(A_i) - f(p_i)\|_2^2 - \|f(A_i) - f(N_i)\|_2^2 + \alpha, 0)$$

CHƯƠNG III. CÁC KẾT QUẢ ĐẠT ĐƯỢC

Từ những kiến thức đã tìm hiểu ở chương 1 và 2, chương 3 sẽ trình bày về hệ thống nhận diện khuôn mặt do nhóm đã xây dựng.

1. Chuẩn bị dữ liệu

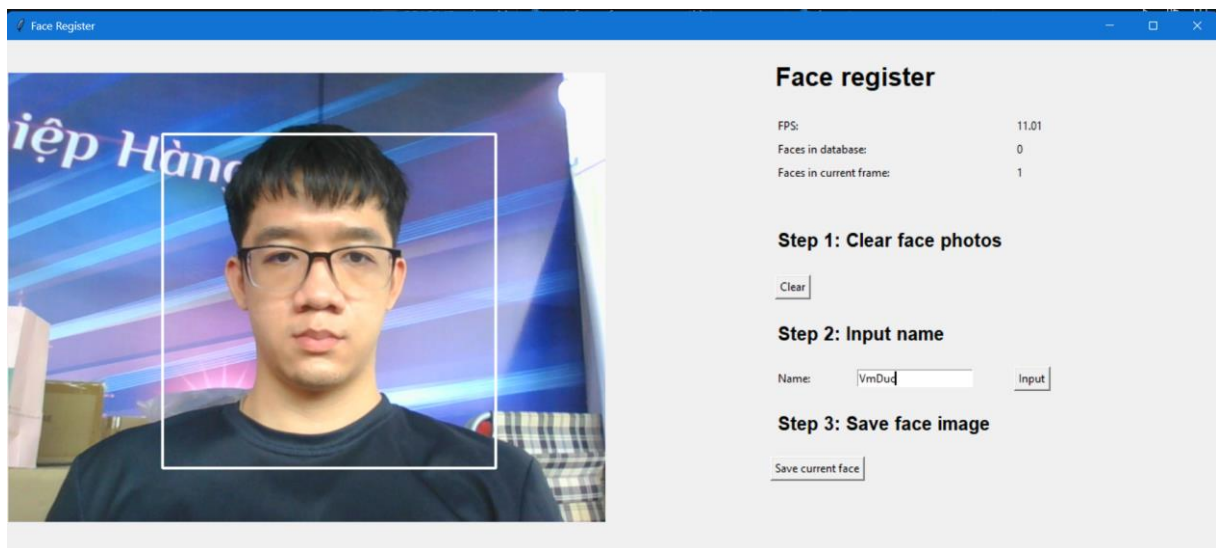
1.1. Thư viện

Tiến hành cài đặt các thư viện cần thiết (đã có trong file requirements.txt):

- Dlib
- numpy
- scikit-image
- pandas
- tk
- opencv-python
- tensorflow

1.2. Xây dựng bộ dữ liệu

Chạy chương trình “get_faces_from_camera_tkinter.py” để lấy khuôn mặt từ camera.



Hình 16. Giao diện lấy dữ liệu khuôn mặt

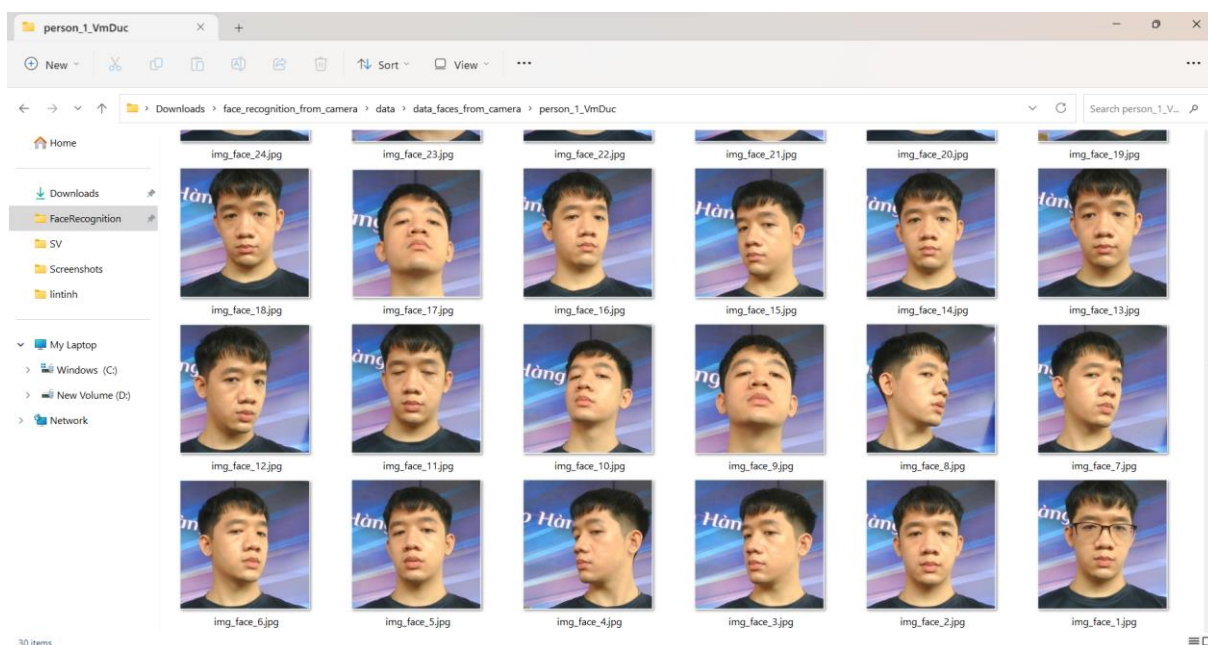
Step 1: clear face photo: Xóa toàn bộ data cũ

Step 2: Input name: Nhập tên người đang đăng ký nhận diện khuôn mặt

Step 3: Save face image: click Save current face để chụp ảnh

Các hình ảnh sẽ được lưu trong folder riêng của từng người

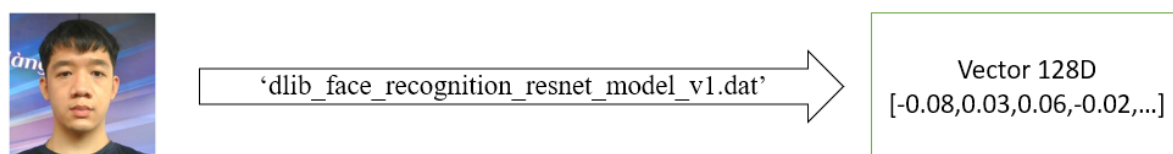
Một số hình ảnh trong bộ dữ liệu:



Hình 17. Một số hình ảnh trong thư mục với tên người đó

2. Trích rút đặc trưng

Ảnh khuôn mặt sẽ được đi qua một số bộ trích chọn đặc trưng mà đầu ra ở đây là vector biểu thị cho khuôn mặt. Sử dụng model “*dlib_face_recognition_resnet_model_v1.dat*” để extract thành vector 128D.



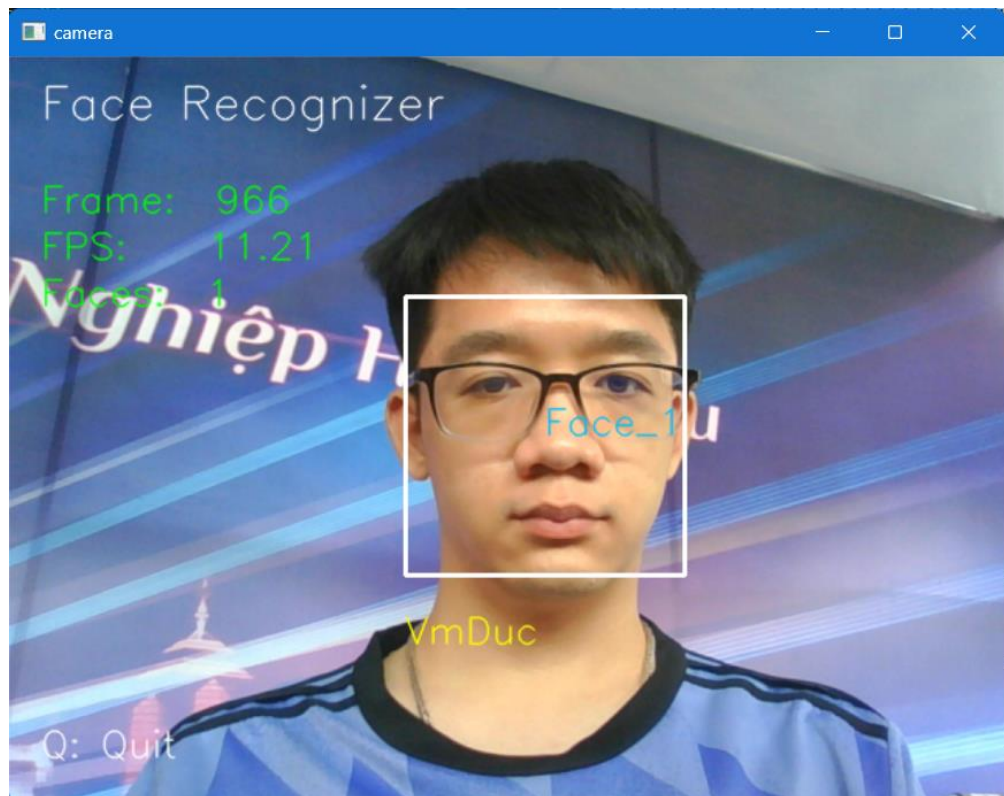
Hình 18. Trích rút đặc trưng khuôn mặt

Các vector đặc trưng sẽ được lưu trong file “*features_all.csv*”.

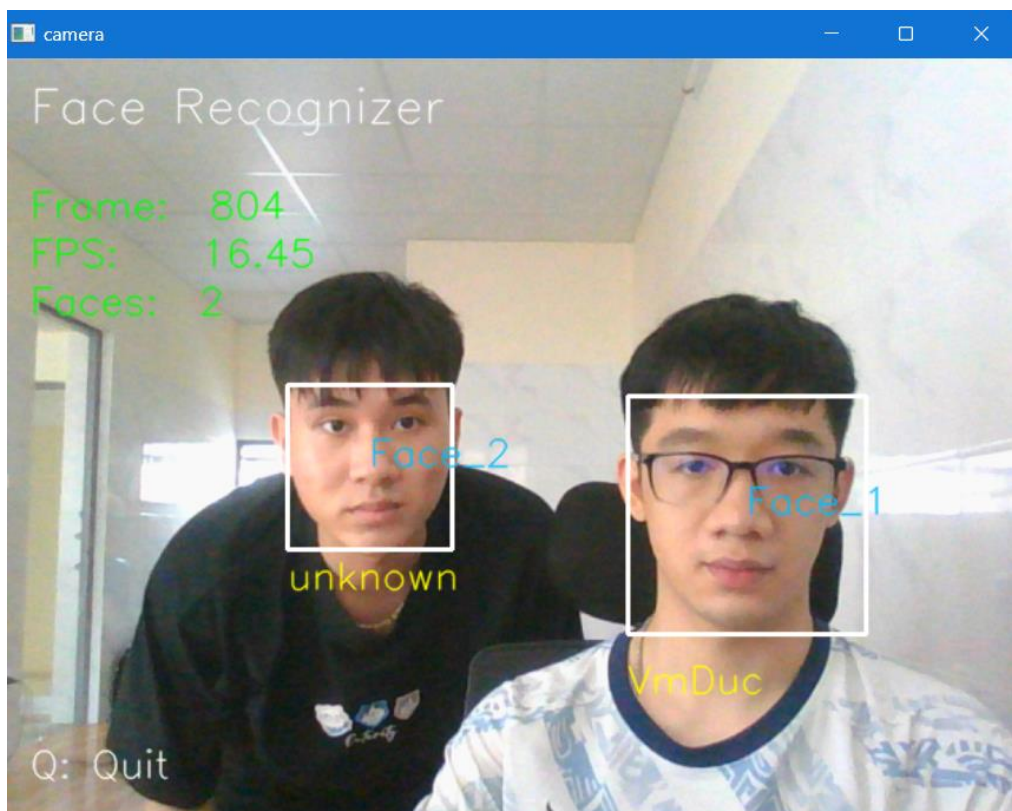
3. Kết quả chạy demo

Kết quả hiển thị tên người nếu có trong bộ dữ liệu, nếu không sẽ hiện unknown.

FPS nằm trong khoảng từ 10-21.



Hình 19. Kết quả hiển thị khi người đó có trong bộ dữ liệu



Hình 20. Hiển thị kết quả khi người đó không có trong bộ dữ liệu

4. Hướng phát triển

Với công nghệ ngày càng phát triển, các hệ thống nhận diện ngày càng phổ biến và có độ chính xác cao. Tuy nhiên trong vấn đề này vẫn còn nhiều khó khăn cần giải quyết như độ chính xác, tốc độ xử lý, sự mất mát không ổn định, vấn đề chống giả mạo, trường hợp khuôn mặt giống nhau (anh chị em ruột, sinh đôi), hay việc khuôn mặt bị che đi một phần. Cùng với đó là việc nâng cao chất lượng của bộ dữ liệu để tăng độ chính xác.

Để phát triển hơn nữa trong tương lai, chúng em đề xuất các hướng phát triển sau:

- Thực hiện tối ưu độ chính xác và tối thiểu hóa sự mất mát, tăng sự ổn định của hệ thống.
- Thêm chức năng chống giả mạo, nhằm tránh bị kẻ xấu lợi dụng kẻ hở đánh lừa hệ thống.
- Dùng các thuật toán khác để tối ưu bài toán về mặt thời gian.
- Xây dựng thành ứng dụng hoàn chỉnh để hỗ trợ nhà trường, công ty, khu công nghiệp điếm danh, chấm công, có các chức năng như thêm, sửa, xóa người, tối ưu thời gian huấn luyện mô hình.
- Tích hợp vào các camera và dữ liệu người dân để thực hiện các chức năng tìm kiếm tội phạm, tìm kiếm người trong đám đông,...

CHƯƠNG IV. PHỤ LỤC

1. Phân chia công việc

STT	Công việc	Người thực hiện
1	Học alige scrum course	Vũ Mạnh Đức, Vũ Trọng Quân, Phan Tấn Sơn
2	Phân tích bài toán	Vũ Mạnh Đức, Vũ Trọng Quân, Phan Tấn Sơn
3	Tìm hiểu YOLOv7, training YOLOv7	Vũ Mạnh Đức, Vũ Trọng Quân, Phan Tấn Sơn
4	Tìm hiểu Facial Landmarks	Vũ Mạnh Đức, Vũ Trọng Quân, Phan Tấn Sơn
5	Tìm hiểu MTCNN	Vũ Mạnh Đức, Vũ Trọng Quân, Phan Tấn Sơn
6	Tìm hiểu model face classification	Vũ Mạnh Đức, Vũ Trọng Quân, Phan Tấn Sơn
7	Xây dựng ứng dụng	Vũ Mạnh Đức, Vũ Trọng Quân, Phan Tấn Sơn
8	Viết báo cáo	Vũ Mạnh Đức
9	Thiết kế slide	Vũ Trọng Quân, Phan Tấn Sơn

2. Tài liệu tham khảo

- [1] Adarsh Ghimire, Naoufel Werghi, Sajid Javed, Jorge Dias, Real Time FaceRecognition System.
- [2] D. F. R. A. Survey, Mei Wang, Weihong Deng.
- [3] [Online]. Available: <https://viblo.asia/p/nhan-dien-khuon-mat-voi-mang-mtcnn-va-facenet-phan-1-Qbq5QDN4lD8>. [Accessed 20 08 2023].
- [4] [Online]. Available: <https://miai.vn/2019/09/11/face-recog-2-0-nhan-dien-khuon-mat-trong-video-bang-mtcnn-va-facenet/>. [Accessed 20 08 2023].
- [5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep ResidualLearning for Image Recognition.
- [6] [Online].Available: http://dlib.net/dnn_face_recognition_ex.cpp.html. [Accessed 20 08 2023].
- [7] [Online]. Available: <https://github.com/timesler/facenet-pytorch>. [Accessed 20 08 2023].
- [8] [Online]. Available: <https://github.com/WongKinYiu/yolov7>. [Accessed 20 08 2023].
- [9] Peng Zhang , Fuhao Zou , Zhiwen Wu, Nengli Dai, Skarpness Mark , MichaelFu , Juan Zhao, Kai Li, FeatherNets: Convolutional Neural Networks asLight as Feather for Face Anti-spoofing.
- [10] [Online]. Available: <https://github.com/davidsandberg/facenet>. [Accessed 20 08 2023].
- [11] [Online]. Available: <https://miai.vn/2019/08/13/face-recognize-thu-lam-he-thong-cham-cong-bang-nhan-dang-khuon-mat/>. [Accessed 20 08 2023].

- [12] [Online]. Available: <https://www.miai.vn/2020/07/08/tong-hop-cac-tuyet-chieu-chong-gia-mao-khuon-mat-bang-anh-video-trong-nhan-dien-khuon-mat/>. [Accessed 20 08 2023].
- [13] Aqeel Anwar , Arijit Raychowdhury, Masked Face Recognition for SecureAuthentication.
- [14] [Online]. Available: <https://github.com/JDAI-CV/FaceX-Zoo>. [Accessed 20 08 2023].
- [15] [Online].Available:
https://github.com/SamYuen101234/Masked_Face_Recognition.
[Accessed 20 08 2023].
- [16] [Online].Available:
https://github.com/ageitgey/face_recognition. [Accessed 20 08 2023].

NHẬN XÉT THỰC TẬP

Khoa Toán - Cơ - Tin học

Trường ĐH Khoa học Tự nhiên ĐHQGHN

NHẬN XÉT THỰC TẬP

Công ty thực tập: IntX Việt Nam

Người hướng dẫn: Lại Tiến Đệ

Thời gian thực tập: 01/06/2023 – 27/08/2023

Đề tài: Nhận diện khuôn mặt bằng phương pháp học máy

STT	Họ và tên SV	Ý thức	Kiến thức	Mức độ hoàn thành CV	Điểm (Thang điểm 10)
1	Vũ Mạnh Đức	Tốt	Khá	Tốt	10
2	Phan Tấn Sơn	Tốt	Khá	Tốt	9.5
3	Vũ Trọng Quân	Tốt	Khá	Tốt	9.5


Những vấn đề cần góp ý với thực tập sinh/ nhà trường để công việc có thể tiến hành tốt hơn:

Trong thời gian thực tập, 3 bạn trong nhóm được giao nhiệm vụ tìm hiểu và thực hành trên các mô hình nhận dạng khuôn mặt. Các có thái độ tốt trong thời gian thực tập, luôn tích cực, chủ động trong công việc, mạnh dạn đặt câu hỏi khi có thắc mắc cần giúp đỡ. Ba bạn đều có kiến thức nền tảng tốt, chăm chỉ học hỏi, tìm tòi và hoàn thành các yêu cầu theo đúng kế hoạch. Qua thời gian thực tập, các bạn đã thu được nhiều kiến thức bổ ích.

Nhận xét chung: Ba bạn Đức, Sơn và Quân đã hoàn thành tốt kỳ thực tập tại Công ty cổ phần Giải pháp công nghệ intX năm 2023.

Hà Nội, ngày 20 tháng 08 năm 2023

Xác nhận của người đánh giá


Lại Tiến Đệ

Xác nhận của doanh nghiệp



1


GIÁM ĐỐC
Trần Nam Trung