**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**NHẬN DẠNG**

**ĐỀ TÀI**

**NHẬN DẠNG NGƯỜI NỔI TIẾNG**

**` GVHD : Đỗ Văn Tiến**

**Thành viên: Phạm Mạnh Cường - 16520156**

**Đỗ Ngọc Kim Đô - 17520346**

**Nguyễn Ngọc Minh -** **17520756**

**Phan Duy Nam -17520783**

**Tô Thanh Tiến - 17521130**

Mục Lục

[**I.** **Tổng quan** 2](#_Toc28718389)

[**II.** **Xây dựng chương trình** 2](#_Toc28718390)

[1. Xây dựng cơ sở dữ liệu ảnh khuôn mặt người nổi tiếng 2](#_Toc28718391)

[2. ResNet 3](#_Toc28718392)

[3. Thuật toán Single Shot Detector 5](#_Toc28718393)

[4. Hệ thống nhận diện khuôn mặt 8](#_Toc28718394)

[**III.** **Quá trình xử lý** 9](#_Toc28718395)

[**IV.** **Cấu trúc đồ án** 12](#_Toc28718396)

[**V.** **Kết luận** 13](#_Toc28718397)

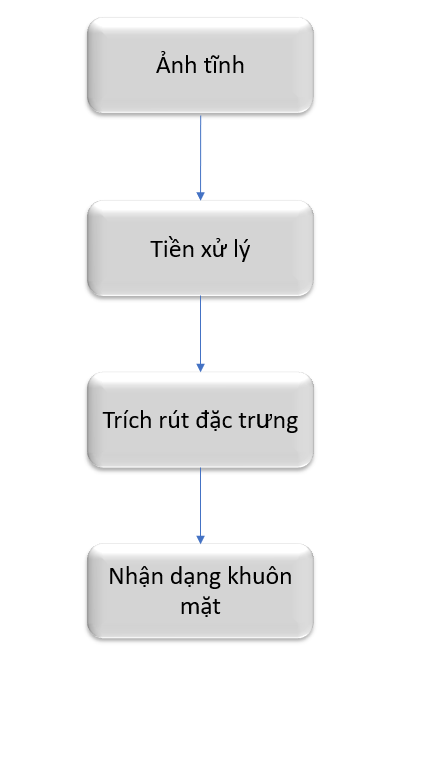
[**VI.** **Tài liệu tham khảo** 13](#_Toc28718398)

1. **Tổng quan**

Nhận dạng khuôn mặt là một trong những công nghệ đã và đang được nghiên cứu, phát triển và ứng dụng rộng rãi trong đời sống con người. Ví dụ như các hệ thống giám sát, quản lý đột nhập, nhận diện người nổi tiếng… Hiện tại, nhiều phương pháp nhận dạng khuôn mặt được đề ra để nâng cao hiệu suất. Tuy nhiện, những phương pháp này đôi khi còn gặp những vấn đề về độ sáng, hướng nghiêng, kích thước hay những ảnh hưởng của yếu tố môi trường như thời tiết và các thời điểm trong ngày.

Dựa vào hướng nhận dạng, người ta có thể chia chúng thành hai nhóm nhận dạng phổ biến hiện nay. Một là phương pháp nhận dạng dựa trên đặc trưng các phần tử trên khuôn mặt như mạng Neural Networks và SVM. Hai là phương pháp nhận dạng dựa trên xét tổng toàn thể khuôn mặt như phương pháp PCA và LDA.

1. **Xây dựng chương trình**

Hình bên mô tả các bước nhận dạng khuôn mặt cơ bản trong hệ thống.

1. Xây dựng cơ sở dữ liệu ảnh khuôn mặt người nổi tiếng

Đây là một bộ cơ sở dữ liệu về khuôn mặt của những người nổi tiếng khắp Việt Nam trong cuộc thi của AiaiVN. Bộ dữ liệu của 1000 người nổi tiếng trên khắp Việt Nam.

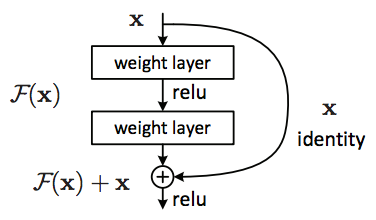


Nguồn ảnh được lấy từ trang [https://www.aivivn.com/contests/7](https://www.aivivn.com/contests/7?fbclid=IwAR3QW7o9kEfvdGszYfohYltQO4mgW0wGpbd0hPPNaUW7MD0oPCD8jFpZ5G8)

1. ResNet

ResNet (residual network), là mô hình deep learning nhận được quan tâm từ những năm 2012 sau cuộc thi LSVRC2012 (một cuộc thi về thị giác máy tính của ImageNet) và trở nên phổ biến trong lĩnh vực thị giác máy tính. ResNet khiến cho việc huấn luyện hàng trăm thậm chí hàng nghìn lớp của mạng nơ-ron trở nên khả thi và hiệu quả.

Khi train các mô hình Deep CNN (số lượng layers lớn, số lượng param lớn, etc) ta thường gặp phải vấn đề về vanishing gradient hoặc exploding gradient. Thực tế cho thấy khi số lượng layer trong CNN model tăng, độ chính xác của mô hình cũng tăng theo, tuy nhiên khi tăng số layers quá lớn (>50 layers) thì độ chính xác lại bị giảm đi.



Residual block ra đời nhằm giải quyết vấn đề trên, với Residual block, ta hoàn toàn có thể train các mô hình CNN có kích thước và độ phức tạp "khủng" hơn mà không lo bị exploding/vanishing gradient. Ý tưởng chính của ResNet là sử dụng kết nối tắt đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một residual block như trong hình trên.

Mấu chốt của Residual block là cứ sau 2 layer, ta cộng input với output: F(x) + x. Resnet là một mạng CNN bao gồm nhiều Residual block nhỏ tạo thành. Hiện tại trong Keras có pre-trained model của ResNet50 với weight được train trên tập ImageNet với 1000 class.

Đây là cách phân lớp một ảnh sử dụng mạng nơ-ron tích chập Resnet được đào tạo với hàng tram hoặc đôi khi hàng triệu tấm ảnh trước đó

* Tải mạng đã được đào tạo trước đó

Hình ảnh muốn được phân loại phải có cùng kích thước với kích thước đầu vào của mạng. Đối với ResNet, yếu tố đầu tiên của thuộc tính Layers của mạng là lớp đầu vào hình ảnh. Kích thước đầu vào mạng là thuộc tính InputSize của lớp đầu vào hình ảnh.

* Đọc và thay đổi kích thước ảnh

Thay đổi kích thước hình ảnh theo kích thước đầu vào của mạng bằng cách sử dụng imresiz. Điều này có một chút thay đổi tỷ lệ khung hình của hình ảnh.

* Phân loại ảnh

Hiển thị hình ảnh với nhãn dự đoán và xác suất dự đoán của hình ảnh có nhãn đó.

Chỉ trả về một lable dù cho trong ảnh có nhiều object.

* Hiển thị dự đoán hàng đầu

Trả về các dự đoán hàng đầu. Vì mạng phân loại ảnh thành nhiều đối tượng tương tự nhau, nên thường xem xét độ chính xác hang đầu khi đánh giá mạng. Kết quả mô hình trả về một trong những nhãn có độ chính xác cao nhất.

1. Thuật toán Single Shot Detector

Dựa trên ý tưởng sử dụng cửa sổ trượt. Chúng ta sẽ trượt trên feature map để nhận diện các đối tượng. Với mỗi loại đối tượng khác nhau, chúng ta sửa dụng các cửa sổ trượt có kích thước khác nhau. Cách này thoạt đầu trông có vẻ khá tốt, nhưng điểm yếu của nó là đã sử dụng cửa sổ trượt làm final boundary box. Do đó, giả sử chúng ta có nhiều đối tượng, và mỗi đối tượng có kích thước khác nhau, chúng ta sẽ có rất nhiều cửa sổ trượt để bao phủ hết toàn bộ đối tượng.

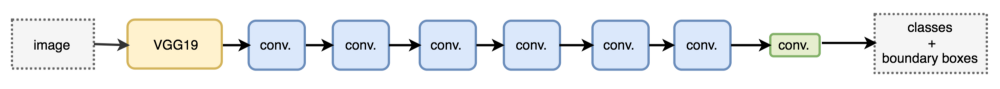
Một ý tưởng cải tiến là chúng ta sẽ định nghĩa trước các cửa sổ trượt, sau đó sẽ tiến hành dự đoán lớp và boundary box. Ý tưởng này, nhóm sử dụng thuật toán là single shot detectors. Ý tưởng này tương tự như việc sử dụng anchors trong Faster R-CNN, nhưng single shot detectors thực hiện dự đoán boundary box và class đồng thời cùng nhau.

Với thuật toán Faster R-CNN, chúng ta sử dụng một convolution filter trả ra 5 kết quả dự đoán: 4 giá trị là toạ độ của boundary box, và giá trị còn lại là xác suất xuất hiện đối tượng. Tổng quát hơn, ta có input là D feature map 8x8, output là 8x8x5, số convolution filter trong Faster R-CNN là 3x3xDx8.

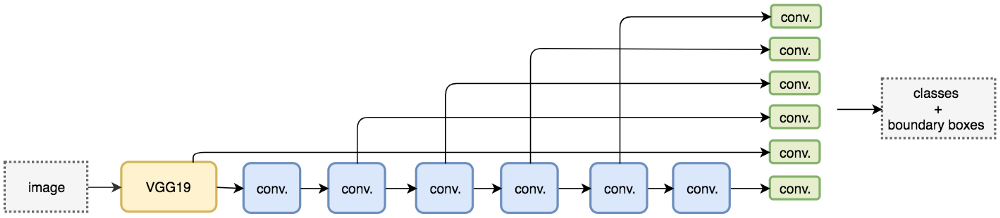
Với single shot detector, input của ta cũng tương tự là 8x8xD, output là 8x8x (4 + C) (với 4 tương ứng với 4 điểm boundary box, và C là số lượng lớp đối tượng), vậy ta cần một convolution filter là 3x3xDx(4+C).

Thuật toán Single Shot Detector chạy khá nhanh, nhưng độ chính xác của nó không cao lắm (không bằng region proposal). Thuật toán có vấn đề về việc nhận dạng các đối tượng có kích thước nhỏ. May mắn thay, bài toán của chúng ta quay quanh việc phân lớp đối tượng nên hạn chế được yếu điểm này của Single Shot Dectector.

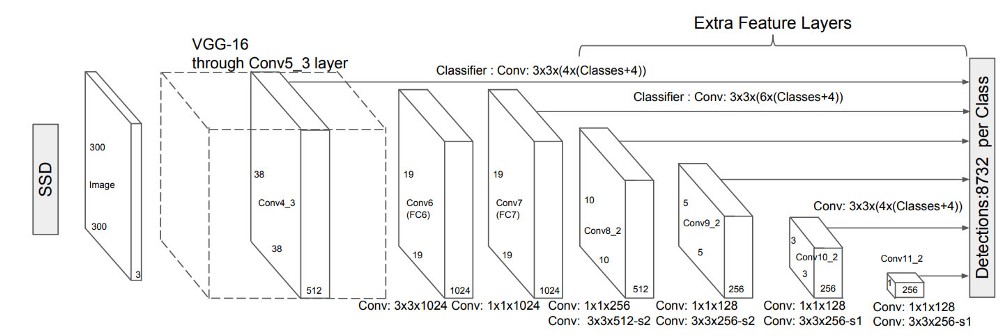
Mô hình single shot detector sử dụng rút trích đặc trưng. Mô hình như hình bên dưới. Trong đó, những conv có màu xanh nước biển nhạt là những custom convolution layer (ta có thể thêm bớt bao nhiêu tuỳ thích). Convolutional filter layer (là cục màu xanh lá cây) có nhiệm vụ tổng hợp các thông tin lại để đưa quyết định.



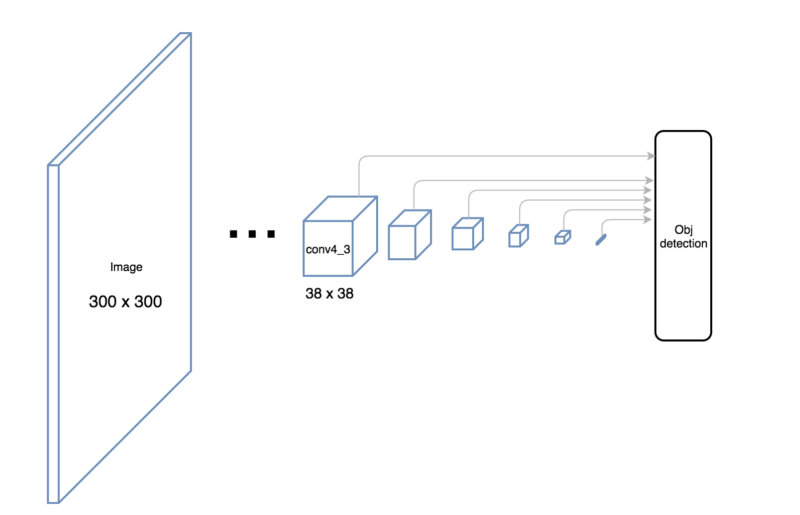
Khi sử dụng mô hình như hình ở trên, chúng ta thấy rằng các custom convolution layer có nhiệm vụ làm giảm chiều và giảm độ phân giải của bức ảnh. Cho nên, mô hình chỉ có khả năng nhận ra các đối tượng có kích thước lớn. Để giải quyết vấn đề này, chúng ta sẽ sử dụng các object detector khác nhau trên mỗi feature maps (xem output của mỗi custom convolution là một feature map).



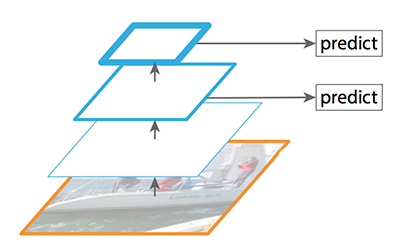
Ảnh bên dưới là sơ đồ số chiều của các feature maps.



SSD sử dụng các layer có kích thước giảm dần theo độ sâu để nhận dạng đối tượng. Nhìn vào hình vẽ sơ đồ bên dưới của SSD, chúng ra dễ dàng nhận thấy rằng độ phân giải giảm đáng kể qua mỗi layer và có lẽ (chắc chắn) sẽ bỏ sót những đối tượng có kích thước nhỏ ở những lớp có độ phân giải thấp. Nếu trong dự án thực tế của bạn có xảy ra vấn đề này, bạn nên tăng độ phân giải của ảnh đầu vào.



SSD đưa ra quyết định dựa vào nhiều feature map. Nhưng layer ở bottom không được sử dụng để nhận dạng đối tượng. Vì những layer này có độ phân giải cao nhưng giá trị ngữ nghĩa của chúng lại không đủ cao (thấp) nên những nhà nghiên cứu bỏ chúng đi để tăng tốc độ xử lý. Các nhà nghiêng cứu biện minh rằng các layer ở bottom chưa đủ mức ý nghĩa cần thiết để nâng cao độ chính xác, thêm các layer đó vào sẽ không nâng độ chính xác cao thêm bao nhiêu và họ bỏ chúng đi để có tốc độ tốt hơn. Cho nên, SSD chỉ sử dụng các layer ở lớp trên, và do đó sẽ không nhận dạng được các đối tượng có kích thước nhỏ.



1. Hệ thống nhận diện khuôn mặt

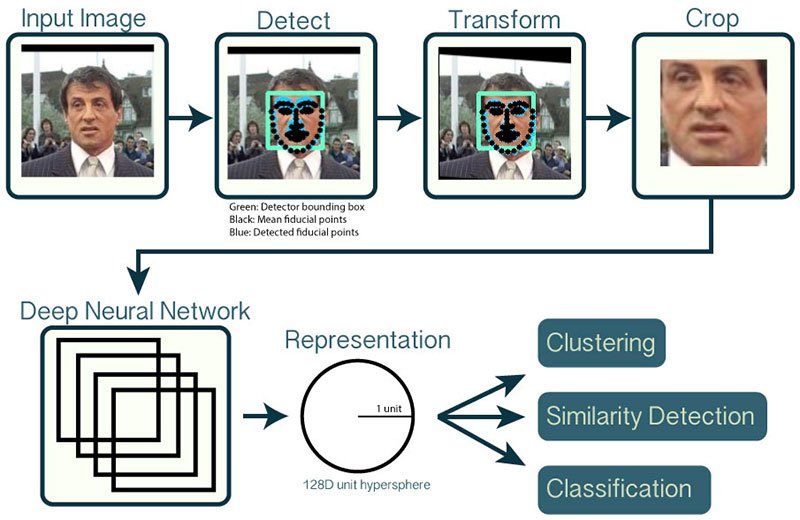
Trước khi muốn xác nhận danh tính một người nổi tiếng, ta phải đi xây dựng hệ thống nhận dạng khuôn mặt. Và để xây dựng hệ thống nhận dạng khuôn mặt bằng OpenCV, ta đi qua các bước chính

* B1: Thực hiện nhận dạng khuôn mặt (trích xuất đặc trưng khuôn mặt từ mỗi khuôn mặt bằng cách học sâu)
* B2: Huấn luyện mô hình nhận dạng khuôn mặt dựa trên các trích xuất đặc trưng đó
* B3: Nhận diện khuôn mặt trên cả hình ảnh và video bằng OpenCV

Nhận dạng khuôn mặt với OpenCV dựa trên 2 thư viện quan trọng là dlib và face\_recognition. Với OpenCV tạo điều kiện cho phép chúng ta nhận dạng khuôn mặt chứ không xác định được khuôn mặt mà chúng ta muốn nhận dạng. Do vậy, ta tiến hành dùng deep learning và OpenCV với nhau để

* Phát hiện khuôn mặt.
* Tính toán nhúng 128-d để định hướng khuôn mặt.
* Huấn liện SVM trên các đặc trưng nổi bật các khuôn mặt.
* Nhận dạng khuôn mặt trong hình ảnh và video.

Vậy câu hỏi được đặt ra là OpenCV hoạt động nhận diện khuôn mặt như thế nào. Và đây là tổng quan về quá trình nhận dạng khuôn mặt của OpenCV.



Để xây dựng đường dẫn nhận dạng khuôn mặt OpenCV, áp dụng deep learning theo hai bước chính:

1. Áp dụng face detection, điều này giúp phát hiện sự hiện diện của khuôn mặt và vị trí của khuôn mặt trong ảnh. Tuy nhiên việc này không giúp xác định được khuôn mặt.
2. Trích xuất vector đặt trưng 128-d định lượng từng khuôn mặt trong một hình ảnh.

Model sau khi huấn luyện rất quan trọng trong việc phân chia lớp, tức là danh tính của người nổi tiếng. Thế nên việc training cần được chú ý một cách đặc biệt.

1. **Quá trình xử lý**

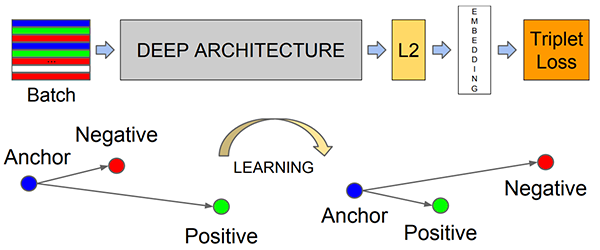
Đầu tiên, nhập một hình ảnh hoặc khung hình video vào đường ống nhận dạng khuôn mặt. Đưa ra hình ảnh đầu vào, áp dụng nhận diện khuôn mặt để phát hiện vị trí của khuôn mặt trong hình ảnh.

* Tùy chọn giúp chúng ta có thể tính toán các ranh giới trên khuôn mặt, cho phép chúng ta xử lý trước và căn chuẩn khuôn mặt.

Căn chỉnh khuôn mặt là quá trình xác định cấu trúc hình học của khuôn mặt và cố gắng để có được sự căn chính xác của khuôn mặt dựa trên dịch chuyển, xoay và tỉ lệ.

Việc căn chỉnh khuôn mặt để tăng độ chính xác nhận dạng khuôn mặt trong một số đường truyền

Sau đó, áp dụng căn chỉnh và cắt xén khuôn mặt bằng cách đưa input qua mạng lưới deep neural network



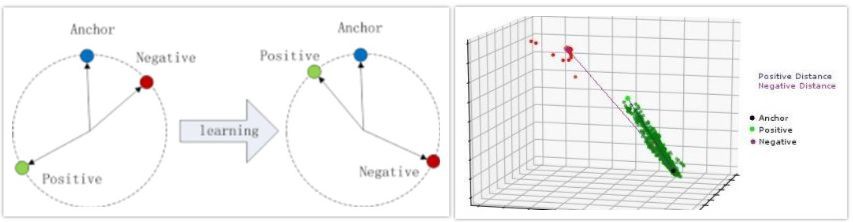
Mô hình học sâu FaceNet tính toán nhúng 128-d để định lượng khuôn mặt.

Quá trình đào tạo gồm:

* Dữ liệu đầu vào network
* **Triplet Loss Function**

Để đào tạo một mô hình nhận dạng với deep learning, mỗi dữ liệu đầu vào bao gồm ba ảnh: the anchor, the positive image, the negative image. Anchor là khuôn mặt hiện tại và có bản chất A. Hình thứ 2 là positive image, hình ảnh này chứa một khuôn mặt của người A.The negative image thì khác, nó không có cùng bản chất và có thể thuộc về người khác. Điểm nổi bật của anchor và positive image là đều thuộc một người, trong khi negative image không chứa cùng một khuôn mặt.

**-Triplet Loss Function:**



*Distance b/w anchor embedding & positive embedding = ||F(A)-F(P)||²*

*Distance b/w anchor embedding & negative embedding =||F(A)-F(N)||²*

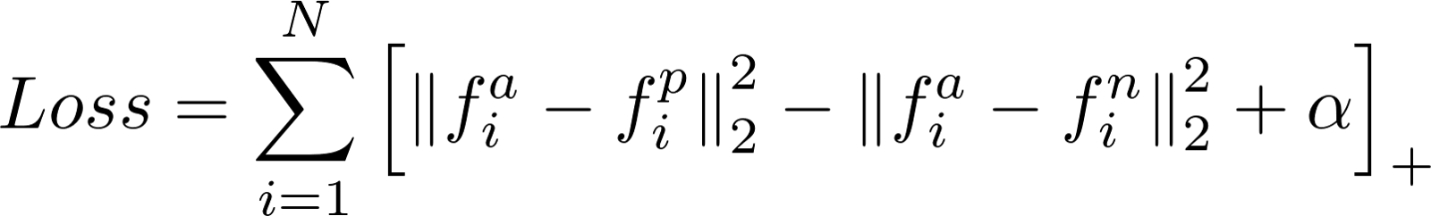
Ta có phương trình:

||F(A)-F(P)||² — ||F(A)-F(N)||² ≤ 0

Ta muốn  **dist(A,P)  nhỏ lại và**  **dist(A,N) cao lên. Để chắc chắn rằng Neural Network không làm cho dist(A,P) & dist(A,N) bằng 0 nên giảm về một giá trị** (α):

||F(A)-F(P)||² — ||F(A)-F(N)||² ≤ - α

Cho N triplet image:



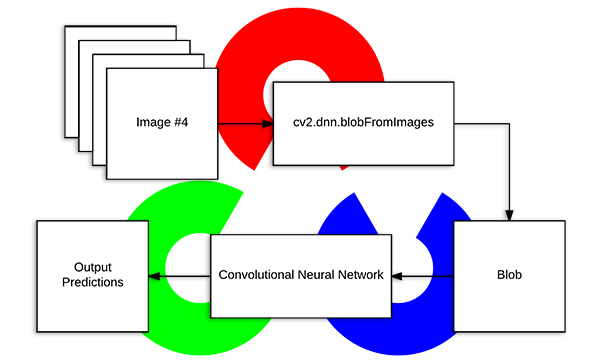
Nhưng một khi **Neural Network** đã được train, nó có thể tạo ra các phép đo cho bất kỳ khuôn mặt nào, ngay cả những khuôn mặt mà nó chưa từng thấy trước đây! Vì vậy, bước này chỉ cần được thực hiện một lần.

The neural network tính toán các nhúng 128-d cho mỗi mặt và chỉnh trọng số của network thông qua hàm chức năng mất mát bộ ba sao cho:

* Các nhúng 128-d của anchor và positive image nằm gần nhau hơn
* Đồng thời, đẩy các nhúng của negative image đi

Chúng ta có thể sử dụng lại mô hình OpenFace cho các ứng dụng mà không cần đào tạo nó một cách rõ ràng.

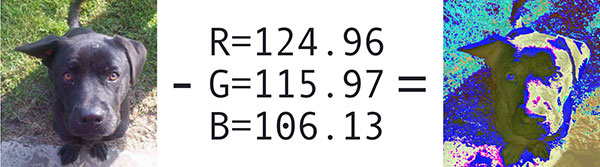
-Blob from image



Blob from image: dùng dể tiền xử lý hình ảnh và chuẩn bị cho việc phân lớp trong mô hình deep learning. Những nhiệm vụ tiền xử lý này thường liên quan đến:

* Mean subtraction
* Scaling by some factor

**Mean subtraction**



* Là một kĩ thuật dùng để chống lại sự thay đổi ánh sáng của ảnh và có thể xem đây là một kĩ thuật hỗ trợ cho CNN
* Các bước thực hiện :
* B1: Tính toán cường độ điểm ảnh trung bình trên tất cả các ảnh trong dataset với từng kênh màu R,G,B là: \mu_R,\mu_G và \mu_B.
* B2 : Tính mean subtract

R = R - \mu_R

G = G - \mu_G

B = B - \mu_B

Ngoài ra, có thể thêm chỉ số scaling \sigma

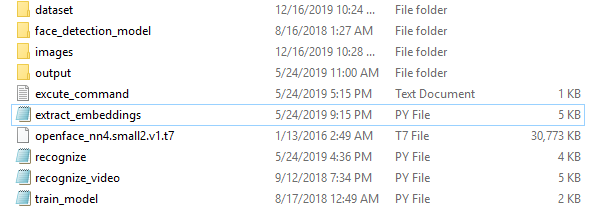
R = (R - \mu_R) / \sigma

G = (G - \mu_G) / \sigma

B = (B - \mu_B) / \sigma

Trong OpenCV có hai hàm dùng để tiền xử lý hình ảnh cho việc phân loại trong deep learning model: cv2.dnn.blobFromImage và cv2.dnn.blobFromImages.

1. **Cấu trúc project**



* dataset/: tập hợp các hình ảnh trong mỗi folder với tên của người đó
* images/: tập các hình ảnh test
* face\_detection\_model/: chứa Caffe model được cung cấp bởi OpenCV để detect khuôn mặt. Mô hình này dùng để phát hiện và khoanh vùng khuôn mặt.
* output/: chứa các output là các file pickle, gồm 3 file:
* embeddings.pickle: Một file chứa các embedding của khuôn mặt nối tiếp. Các embedding này đã được tính toán cho mọi khuôn mặt trong bộ dữ liệu và được lưu trữ trong tệp này
* le.pickle : bộ mã hóa nhãn. Chứa các label mà mô hình có thể nhận ra
* recognizer.pickle: mô hình SVM
* openface\_nn4.small2.v1.t7: Torch deep learning model tạo ra embedding 128-d
* extract\_embeddings.py: sử dụng deep learning để trích xuất đặt trưng tạo ra vector 128-d để biểu diễn khuôn mặt. Tất cả khuôn mặt trong dataset sẽ đi qua neural network để tạo thành các embedding.
* train\_model.py: phát hiện khuôn mặt, trích xuất embedding và làm cho mô hình phù hợp với dữ liệu.
* recognize.py: Phát hiện khuôn mặt, trích xuất embedding và truy vấn mô hình SVM để xác định là ai trong ảnh và vẽ các bounding box kèm chú thích.
* recognize\_video.py: Nhận diện khuôn mặt real time.

1. **Kết luận**

Mạng nơron là mạng có tính linh hoạt các nên ta có thể áp dụng nó không chỉ cho nhận dạng khuôn mặt, mà còn ứng dụng cho nhận dạng vân tay, điệu bộ mà chỉ cần thay đổi dữ liệu đầu vào với giá trị đầu ra mong muốn. Đây chính là một ưu thế của mạng nơron mà những thuật toán khó có được. Tuy nhiên kết quả nhận dạng ta thu được nằm trong những điều kiện cụ thể. Việc ứng dụng hệ thống vào thực tế sẽ gặp những vấn đề thách thức hiện nay như các thông số của môi trường, chất lượng ảnh thu được…. Đây là một hướng phát triển của đề tài để đưa hệ thống vào các ứng dụng thực tế như quản lý nhân viên, kiểm soát ra vào.

1. **Tài liệu tham khảo**

<https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/?fbclid=IwAR0bCPcFKmpUkTasKZf0iNQwnqmPpL3uuPs0gvSpTJd71-23hM0M5Ecadl4>

<https://www.mathworks.com/help/deeplearning/ref/resnet50.html>

<https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html>

<https://viblo.asia/p/gioi-thieu-ve-cac-pre-trained-models-trong-linh-vuc-computer-vision-3Q75wB1GlWb>

<https://trituenhantao.io/kien-thuc/resnet-mang-hoc-sau-dung-nghia/>