Contents

[1. Haar Casade: 3](#_Toc5707148)

[1.1. Haar Casade là gì: 3](#_Toc5707149)

[1.2. Phân tích Haar Casade 3](#_Toc5707150)

[1.3. Sử dụng Haar Cascade trong nhận diện khuôn mặt: 4](#_Toc5707151)

[2. Histogram of Oriented Gradients (HOG): 5](#_Toc5707152)

[2.1. Lý thuyết: 5](#_Toc5707153)

[2.2. Ví dụ: 6](#_Toc5707154)

[3. SVM: 7](#_Toc5707155)

[3.1. SVM là gì 7](#_Toc5707156)

[3.2. SVM làm việc như thế nào 7](#_Toc5707157)

[4. CNN 11](#_Toc5707158)

[4.1. Kỹ thuật xác định dựa vào các kiến thức con người (Knowledge-based): 11](#_Toc5707159)

[4.2. Kỹ thuật xác định dựa vào đặc tính khuôn mặt (Feature-based): 12](#_Toc5707160)

[4.3. Kỹ thuật xác định dựa vào mẫu cho trước (Template-Matching): 12](#_Toc5707161)

[4.4. Kỹ thuật xác định dựa vào hình dáng (Appearance-Base): 12](#_Toc5707162)

[5. So sánh: 13](#_Toc5707163)

[6. Facial Landmark Localization 14](#_Toc5707164)

[6.1. Giới thiệu: 14](#_Toc5707165)

[6.2. Framework for Landmark Localiztion 14](#_Toc5707166)

[6.3. Eye Localization 14](#_Toc5707167)

[6.4. Midline of Eyes 15](#_Toc5707168)

[6.5. Eye Candidate Detection 16](#_Toc5707169)

[6.6. Eye Candidate Subsampling 16](#_Toc5707170)

[6.7. Eye-Pair Classification 17](#_Toc5707171)

[6.8. Shape Modeling 17](#_Toc5707172)

[6.9. Example: 18](#_Toc5707173)

[6.10. References 20](#_Toc5707174)

[7. Affine transformations in Opencv: 20](#_Toc5707175)

[7.1. Transformations: 20](#_Toc5707176)

[7.2. Scaling: 20](#_Toc5707177)

[7.3. Rotation: 21](#_Toc5707178)

[7.4. Affine Transformation: 21](#_Toc5707179)

[7.5. Perspective Transformation: 22](#_Toc5707180)

[7.6. References; 22](#_Toc5707181)

[8. Encoding faces and Face recognition: 22](#_Toc5707182)

[8.1. Triplet Loss: 22](#_Toc5707183)

[8.2. Face Recognition với Facenet: 23](#_Toc5707184)

[8.3. Example: 24](#_Toc5707185)

[8.3.1. Training: 24](#_Toc5707186)

[8.3.2. Testing: 25](#_Toc5707187)

[8.4. References: 26](#_Toc5707188)

[9. 26](#_Toc5707189)

1. Haar Casade:
   1. Haar Casade là gì:

Haar Casade là một thuật toán dành cho máy học với chức năng nhận diện vật thể có trong ảnh hay 1 đoạn video. Thuật toán này dựa trên đề xuất của Paul Viola và Michael Jones

* 1. Phân tích Haar Casade



Thuật toán này bao gồm 4 giai đoạn:

* Lựa chọn đặc tính:



Đặc tính là những hình chữ nhật liền kề nhau nằm tại một vùng nhất định từ khung hình đang được xem xét, sau đó tính tổng các pixel lại để ra được những tổng khác nhau.

* Tạo ảnh tích hợp:

Những đặc tính sau khi tính toán thường không liên quan nhau.

Cho ví dụ với hình dưới đây:



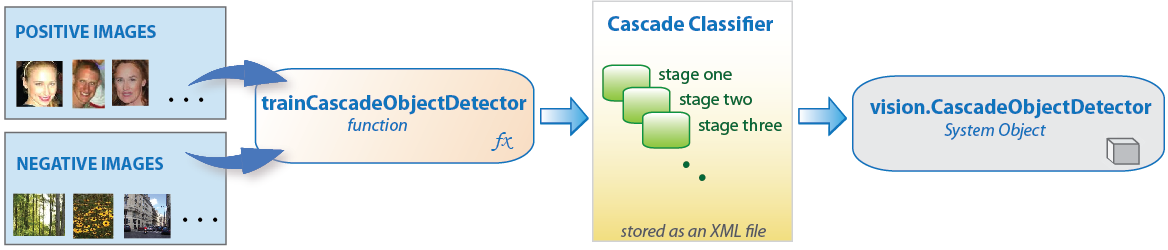
Ở đặc tính đầu tiên, tập trung vào vùng dưới trán. Vùng mắt sẽ nằm trong phần màu đen (tối hơn) và vùng mũi, má sẽ trong phần sang

Ở đặc tính thứ 2, tập trung từ trán tới sống mũi. Vùng mắt sẽ nằm 2 bên (màu đen) và sống mũi sẽ màu trắng (sáng hơn).

* Nhìn 2 đặc tính, không có gì liên quan nhau.
* Adaboost training:

Đây là một concept có chức năng chọn ra đặc tính tốt nhất và phân loại.

* Cascading Classifier:



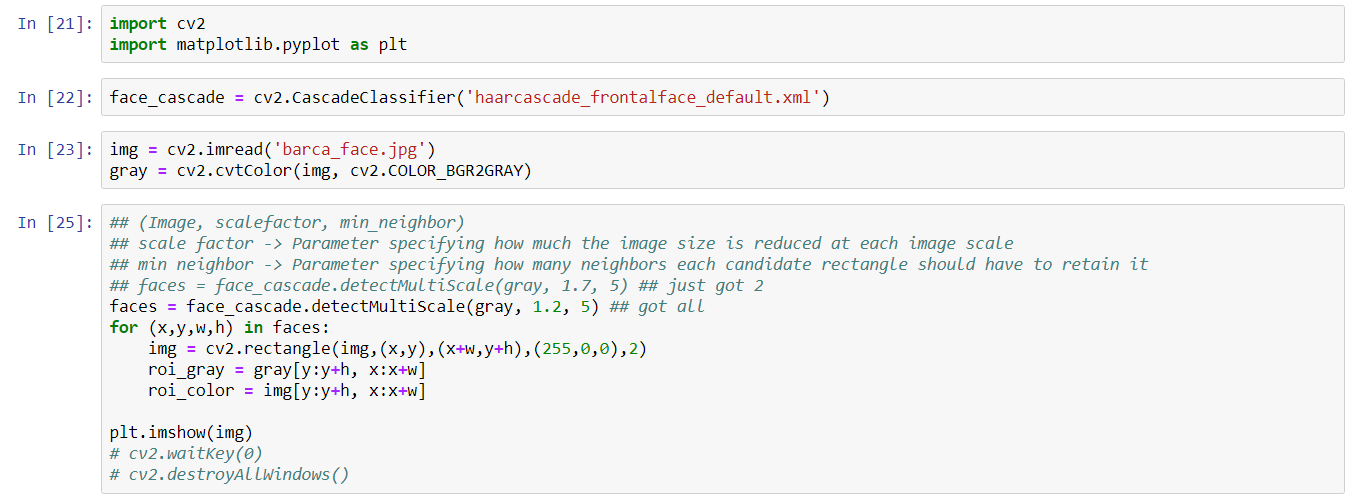
Đây là một chuỗi những giai đoạn khác nhau, những giai đoạn này là những đoạn máy học đơn giản. Những đoạn máy học này là bộ phân loại đơn giản gọi là *decision stumps.*

Mỗi giai đoạn gán mã cho một vùng hình đã cắt ra từ bức ảnh, và gán cho nó theo dạng nhị phân (Có hoặc không). Có ở đây là khung hình này chính là 1 vật thể. Còn không thì ngược lại.

Những gia đoạn này sẽ có 1 tỉ lệ đoán sai, điều này dùng để xem xét cho những giai đoạn sau đó (nên xét tiếp hay không).

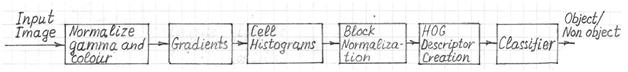
🡺Tổng hợp lại ta sẽ có những khung hình cắt thỏa yêu cầu.

* 1. Sử dụng Haar Cascade trong nhận diện khuôn mặt:





1. Histogram of Oriented Gradients (HOG):
   1. Lý thuyết:



Bước 1:

Chia ảnh thành những khối (ví dụ 16x16). Mỗi khối này lại chia nhỏ hơn (8x8). Thường những khối này đè chồng nhau, và từ mỗi khối đó ta thu được gradient. Để làm được điều này ta sử dụng thuật toán 1-D Sobel theo chiều dọc và chiều ngang như sau:

*Gx*(*y*,*x*) = *Y*(*y*,*x*+1) – *Y*(*y*,*x*-1); *Gy*(*y*,*x*) = *Y*(*y*+1,*x*) – *Y*(*y*-1,*x*)

Với:

Y(y, x) là cường độ điểm ảnh tại x, y.

*Gx*(*y*,*x*) là gradient theo chiều ngang.

*Gy*(*y*,*x*) là gradient theo chiều dọc.

Độ lớn và pha được định nghĩa như sau:

hog-object-detection-eq1

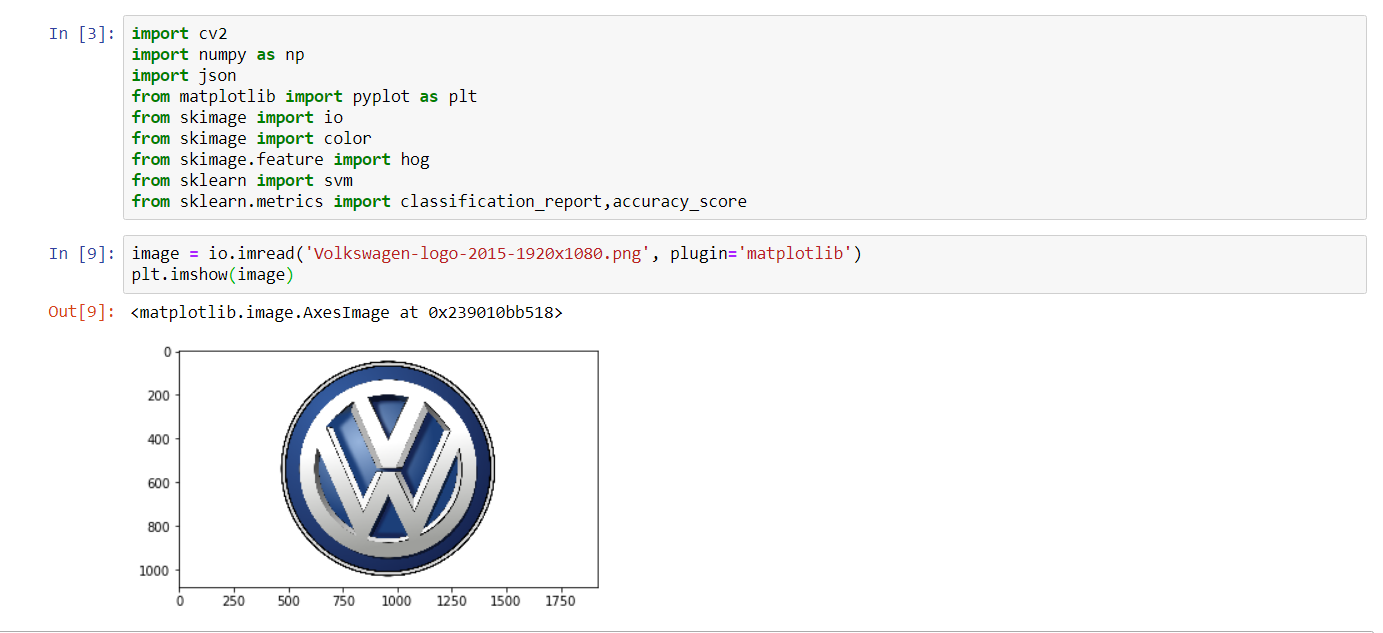
HOG được tạo ra cho từng cell, Chọn Q và hướng thì vô định cho nên góc sẽ quay từ nhỏ hơn 00 và tang đến 1800

Vì từng bức ảnh có từng độ tương phản khác nhau cho nên cân bằng độ tương phản có thể rất cần thiết. Cân bằng sử dụng vecto v với từng khối, một trong những công thức được dung như sau:

hog-object-detection-eq2

Và cuối cùng những đặc tính này được đưa vào SVM để huấn luyện phân lớp.

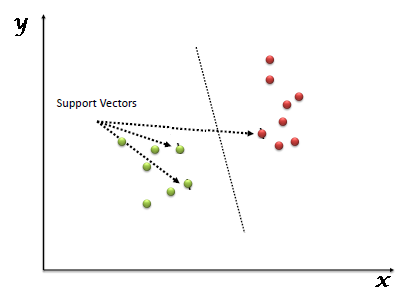
* 1. Ví dụ:





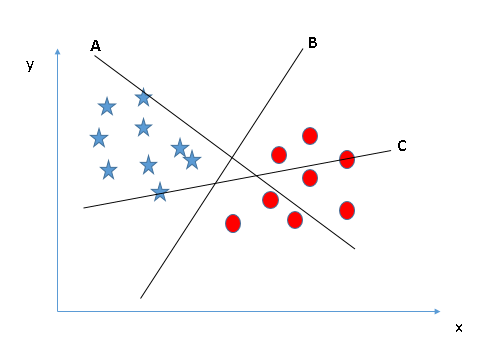
1. SVM:
   1. SVM là gì

SVM là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồi thị dữ liệu là các điểm trong n chiều ( ở đây n là số lượng các tính năng bạn có) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng ta thực hiện tìm "đường bay" phân chia các lớp. Đường bay - nó chỉ hiểu đơn giản là 1 đường thằng có thể phân chia các lớp ra thành hai phần riêng biệt.



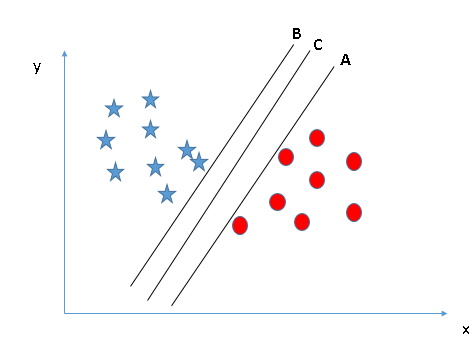
* 1. SVM làm việc như thế nào

- Identify the right hyper-plane (Scenario-1):



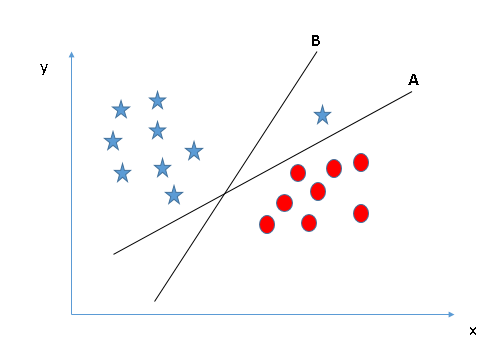
Quy tắc số một để chọn 1 hyper-lane, chọn một hyper-plane để phân chia hai lớp tốt nhất. Trong ví dụ này chính là đường B

- Identify the right hyper-plane (Scenario-2):



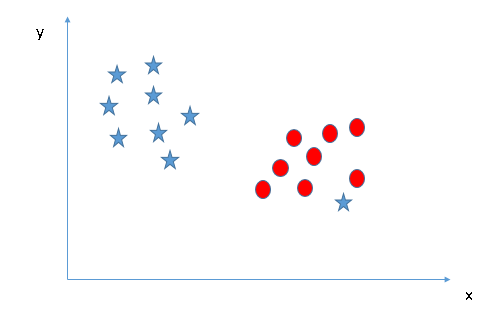
Quy tắc thứ hai chính là xác định khoảng cách lớn nhất từ điểu gần nhất của một lớp nào đó đến đường hyper-plane. Khoảng cách này được gọi là "Margin", Hãy nhìn hình bên dưới, trong đấy bạn có thể nhìn thấy khoảng cách margin lớn nhất đấy là đường C. Ở đây bạn nhớ nếu chọn lầm hyper-lane có margin thấp hơn thì sau này khi dữ liệu tăng lên thì sẽ sinh ra nguy cơ cao về việc xác định nhầm lớp cho dữ liệu

- Identify the right hyper-plane (Scenario-3):



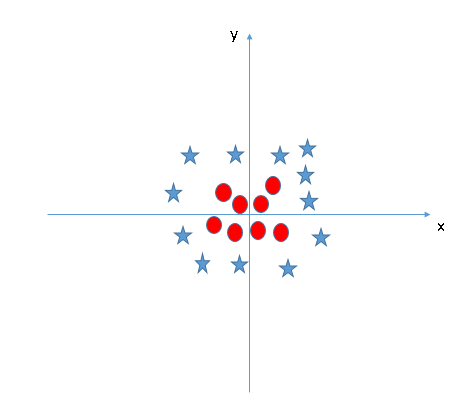
Có thể có một vài bạn sẽ chọn đường B bởi vì nó có margin cao hơn đường A, nhưng đấy sẽ không đúng bởi vì nguyên tắt đầu tiên sẽ là nguyên tắc số 1., chúng ta cần chọn hyper-plane để phân chia các lớp thành riêng biệt. Vì vậy đường A mới là lựa chọn chính xác.

- Can we classify two classes (Scenario-4)?

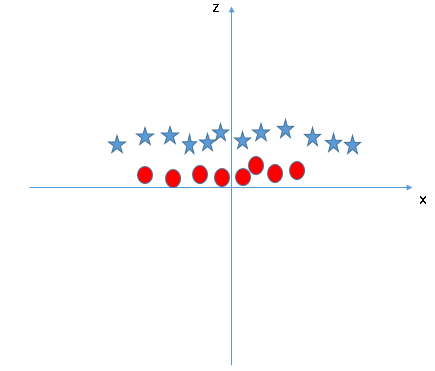


Một ngôi sao ở bên ngoài cuối được em như một ngôi sao phía ngoài hơn, SVM có tính năng cho phép bỏ qua các ngoại lệ và tìm ra hyper-plane có biên giới tối đa . Do đó chúng chúng ta có thể nói, SVM có khả năng mạnh trong việc chấp nhận ngoại lệ.

Find the hyper-plane to segregate to classes (Scenario-5)



 SVM có thể giải quyết vấn đề này, Khá đơn giản, nó sẽ được giải quyết bằng việc thêm một tính năng, Ở đây chúng ta sẽ thêm tính năng z = x^2+ y^2. Bây giờ dữ liện sẽ được biến đổi theo trục x và z như sau



* Tất cả dữ liệu trên trục z sẽ là số dương vì nó là tổng bình phương x và y
* Trên biểu đồ các điểm tròn đỏ xuất hiện gần trục x và y hơn vì thế z sẽ nhỏ hơn => nằm gần trục x hơn trong đồ thị (z,x)

Trong SVM, rất dễ dàng để có một siêu phẳng tuyến tính (linear hyper-plane) để chia thành hai lớp, Nhưng một câu hỏi sẽ nảy sinh đấy là, chúng ta có cần phải thêm một tính năng phân chia này bằng tay hay không. Không, bởi vì SVM có một kỹ thuật được gọi là kernel trick ( kỹ thuật hạt nhân), đây là tính năng có không gian đầu vào có chiều sâu thấm và biến đổi nó thành không gian có chiều cao hơn, tức là nó không phân chia các vấn đề thành các vấn đề riêng biệt, các tính năng này được gọi là kernel. Nói một cách đơn giản nó thực hiện một số biết đổi dữ liệu phức tạp, sau đó tìm ra quá trình tách dữ liệu dựa trên các nhãn hoặc đầu ra mà chúng ra đã xác định trước.

1. CNN
   1. Kỹ thuật xác định dựa vào các kiến thức con người (Knowledge-based):

ỹ thuật này dựa vào các hiểu biết của con người về khuôn mặt để xác định được một khuôn mặt trong ảnh (Ví dụ như việc một khuôn mặt sẽ phải có mắt, mũi, miệng và khoảng cách giữa chúng thường sẽ phải thoả mãn các ràng buộc nào đó,..). Điểm khó của kỹ thuật này nằm ở việc chúng ta sẽ phải xây dựng nên 1 bộ quy tắc. Nếu bộ quy tắc quá chung chung hay quá chặt chẽ thì đều không được vì nó sẽ dẫn tới việc nhận dạng nhầm hoặc không nhận dạng được.

* 1. Kỹ thuật xác định dựa vào đặc tính khuôn mặt (Feature-based):

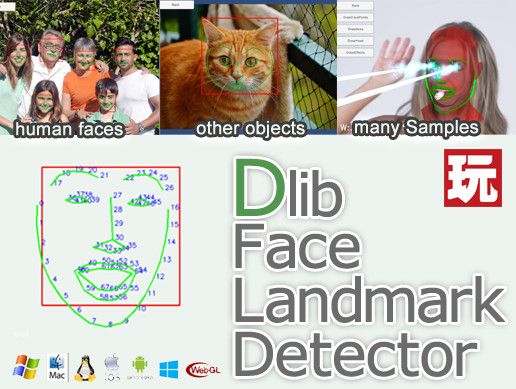
Kỹ thuật này tạo ra một mô hình, sau đó chúng ta sẽ huấn luyện mô hình đó như một mô hình để phân loại (classifier) nhằm xác định trong các khung hình cắt ra từ 1 ảnh ban đầu, đâu là các vùng của một khuôn mặt. Điểm yếu lớn nhất của kỹ thuật này là về mặt thời gian. Có vấn đề này là do chúng ta sẽ phải lấy ra rất nhiều vùng trong 1 bức ảnh nhằm đưa qua classifier.

* 1. Kỹ thuật xác định dựa vào mẫu cho trước (Template-Matching):

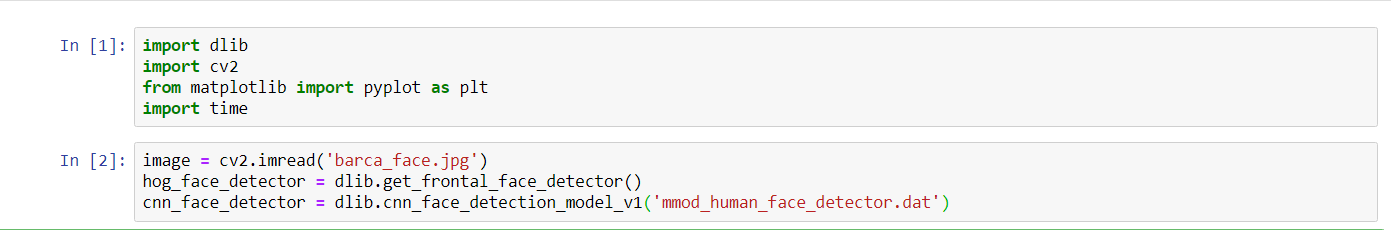
Kỹ thuật này xác định được vị trí của một khuôn mặt trong bức ảnh dựa vào việc so sánh giữa các bức ảnh khuôn mặt chúng ta cho trước (Feature template) và các khung hình được cắt ra. Template-Matching rất dễ dàng để sử dụng tuy nhiên cũng gặp phải vấn đề về thời gian tương tự như kỹ thuật Feature-base ở trên.

* 1. Kỹ thuật xác định dựa vào hình dáng (Appearance-Base):

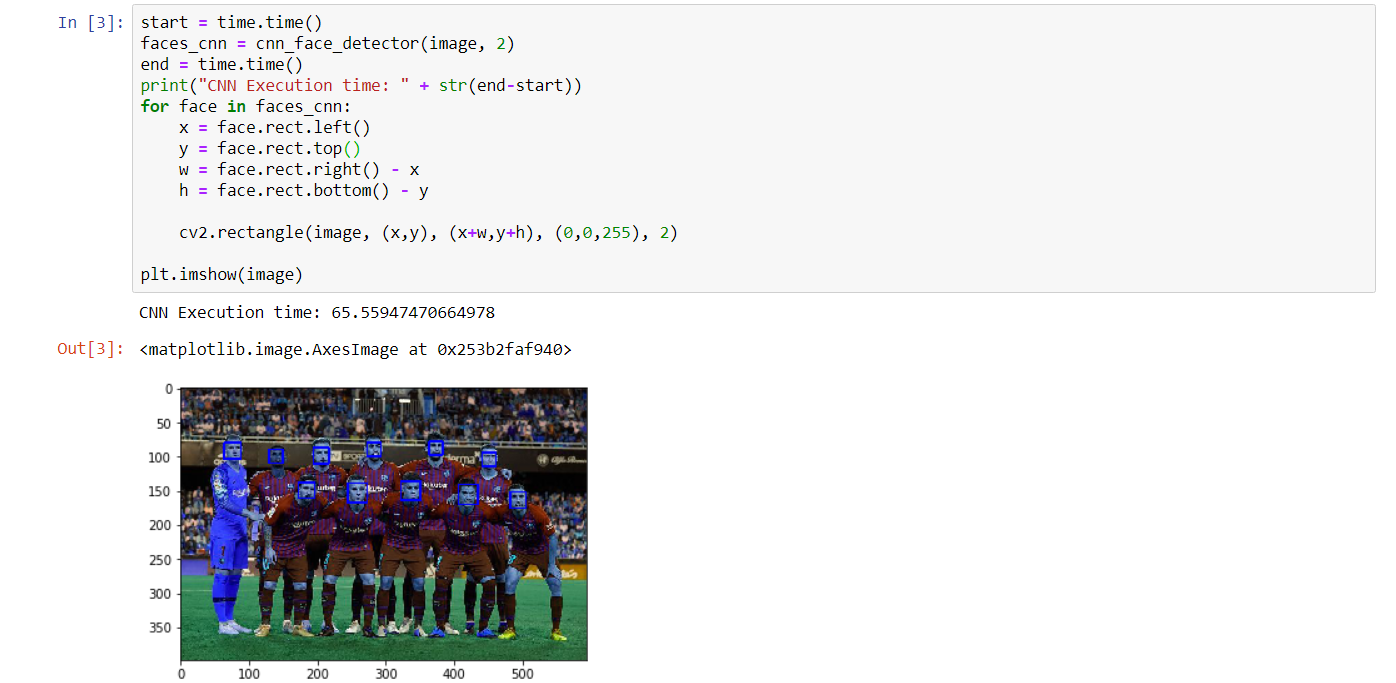
Đây là kỹ thuật mà Dlib sẽ sử dụng, phương pháp sử dụng các phương pháp hình thái học kết hợp với phân tích từ mô hình machine-leanring để xác định trực tiếp về vị trí của các vùng có khuôn mặt trong ảnh. Chúng ta sẽ nói kỹ hơn nữa về kỹ thuật này ở phần sau.



Ví dụ:







1. So sánh:

Với Hog: ta được kết quả chính xác và thời gian xử lý rất nhanh

Với CNN: ta cũng được kết quả chính xác nhưng thời gian xử lý và dự đoán rất chậm

🡺Quyết định sử dụng Hog vì lí do ta cần real-time

1. Facial Landmark Localization
   1. Giới thiệu:

Phát hiện và nhận diện khuôn mặt là một lĩnh vực lớn trong AI về vấn đề sinh trắc học và đã thương mại hóa thành công 20 năm nay

* 1. Framework for Landmark Localiztion

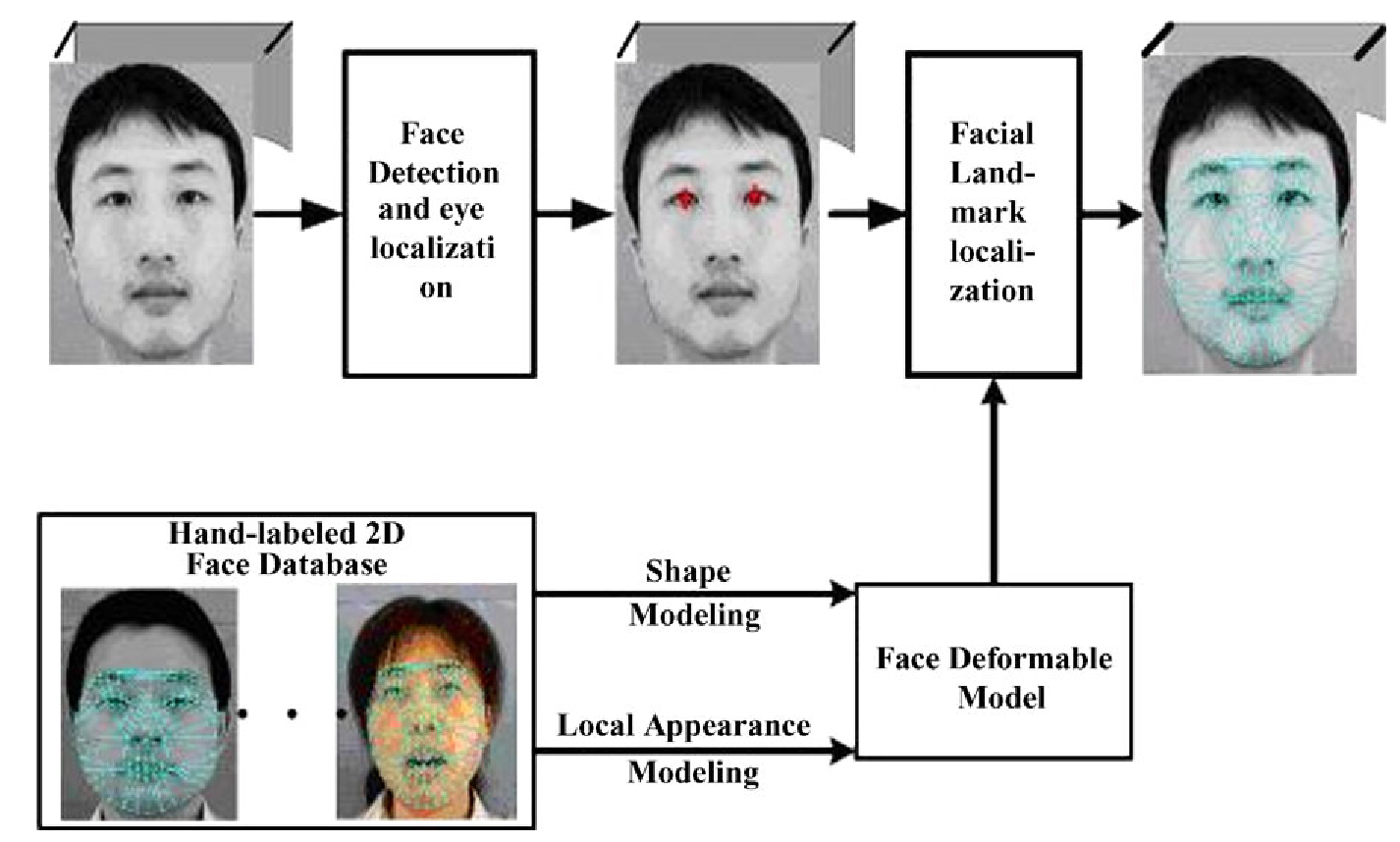
Quy trình huấn luyện: xây dựng một khuôn mặt với hình dạng khác dựa trên những model đã được chứng minh giải thích toán học. Quy trình này cần một lượng lớn data được gán mã bằng tay (có nghĩa là người sẽ đánh giá nó trước), và nó bao gồm: phát hiện khuôn mặt, sau đó đến định vị mắt, và đến những mốc vị trí đã được quy định của khuôn mặt đã được thay đổi hình dạng. Từ những điểm này, ta sẽ xác định được là khuôn mặt hay không.

* 1. Eye Localization

Việc định vị mắt rất là quan trọng vì những vị trí liên quan được đo lường và tính toán hay là việc trích xuất đặc tính dựa vào mắt.

Mắt cũng có đa dạng biểu hiện để nhận diện như là: kích thước, tư thế, góc xoay, độ gần, mở mắt nhắm mắt, điều kiện ánh sáng, đeo mắt kiếng.

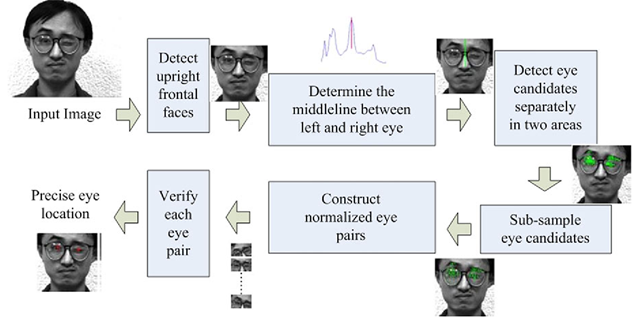
Mặc dù tìm được khuôn mặt một cách rõ rang, chính xác tuy nhiên định vị mắt vẫn là một thử thách lớn. Số lượng khá lớn những thuật toán phát hiện mắt đã được đăng, nhưng hầu hết trong một điều kiện nhất định mới nhận diện được.



Hình 1

* 1. Midline of Eyes

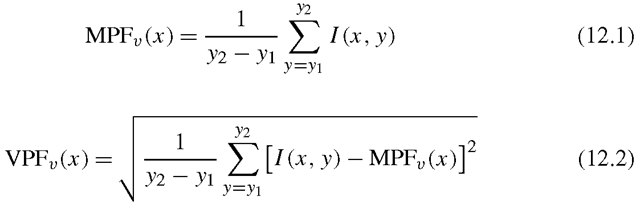
Hình dưới mô tả việc khi một tấm hình thô được đưa vào hệ thống, hệ thống sẽ chỉ ra đường giữa 2 mắt. Sau đó, lọc kĩ vùng mắt bằng những điểm đại diện cho mắt (con ngươi, lòng trắng, mí mắt, …) và cuối cùng qua một bộ phân lớp để thu nhỏ lại phần chính xác của mắt.



Hình 2

Theo như quan sát, ta thấy cường độ **thay đổi màu xám** của vùng mắt sẽ rõ ràng hơn sống mũi và vùng mắt thường sẽ tối hơn vùng sống mũi và được biểu diễn theo công thức sau:

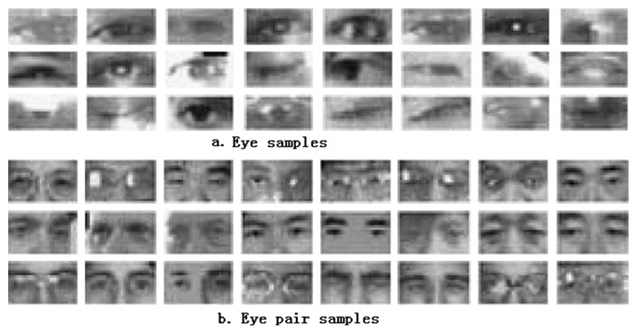
***Suppose I(x, y)****is the intensity of a pixel at location (x, y), the vertical mean projection function[tmp35b0-447_thumb](http://what-when-how.com/wp-content/uploads/2012/06/tmp35b0447_thumb.png)and    vertical    variance    projection function[tmp35b0-448_thumb](http://what-when-how.com/wp-content/uploads/2012/06/tmp35b0448_thumb.png)of I(x,y) in intervals [y1,y2] can be defined respectively, as:*



Áp dụng 2 hàm vào phần trên của mặt, ta nhận được kết quả rõ ràng xung quanh sống mũi. Sau đó ta tìm được đường giữa chia nửa mắt trái và mắt phải và áp dụng bộ phát hiện lên từng phần riêng biệt.

* 1. Eye Candidate Detection

Sử dụng Adaboost training (Giải thích bên trên). Ví dụ: mắt với kích thước là 24x12 sẽ có khoảng 40000 đặc tính. Trong bộ data để huấn luyện có khoảng 6800 mẫu. Tất cả những mẫu này được cắt ra từ khuôn mặt và được đặt chính giữa bức ảnh mẫu. Sau đó ta sẽ thay đổi kích thước ảnh cần dự đoán về 24x12. Trong bước này, ta tránh đưa ra quyết định vị trí chính xác của mắt nhưng thay vào đó ta sẽ loại trừ những điểm nền ra và cuối cùng chọn ra những điểm có thể đại diện cho mắt nhằm giảm sai sót vì có thể nhận diện nhầm thành lông mày, rìa kiếng quá dày, …



* 1. Eye Candidate Subsampling

Do là biểu hiện của mắt không hoàn toàn riêng biệt cho nên sẽ nhận diện ra những “mắt giả” như là lông mày, những vùng sai (hình 2, ảnh 4 theo thứ tự). Và nếu ta lại lấy những mẫu cũ bỏ vào data input bước tiếp theo sẽ dẫn đến xử lý rất lâu, vì vậy để hợp nhất các mẫu cùng một nhóm, ta tạo mẫu phụ (subsample) với nhân tố N theo chiều ngang và dọc. Biến N được điều chỉnh theo độ rộng khuôn mặt. Và sau khi lấy mẫu phụ, số lượng data input sẽ giảm được khoảng 1/3 mẫu.

* 1. Eye-Pair Classification

Để loại trừ những vùng sai phạm hoặc không chính xác mà không đại diện cho mắt, ta xây dựng lớp nhận diện cặp mắt theo như cách phát hiện mắt. Mắt chỉ nằm trong khung hình vuông nhỏ.

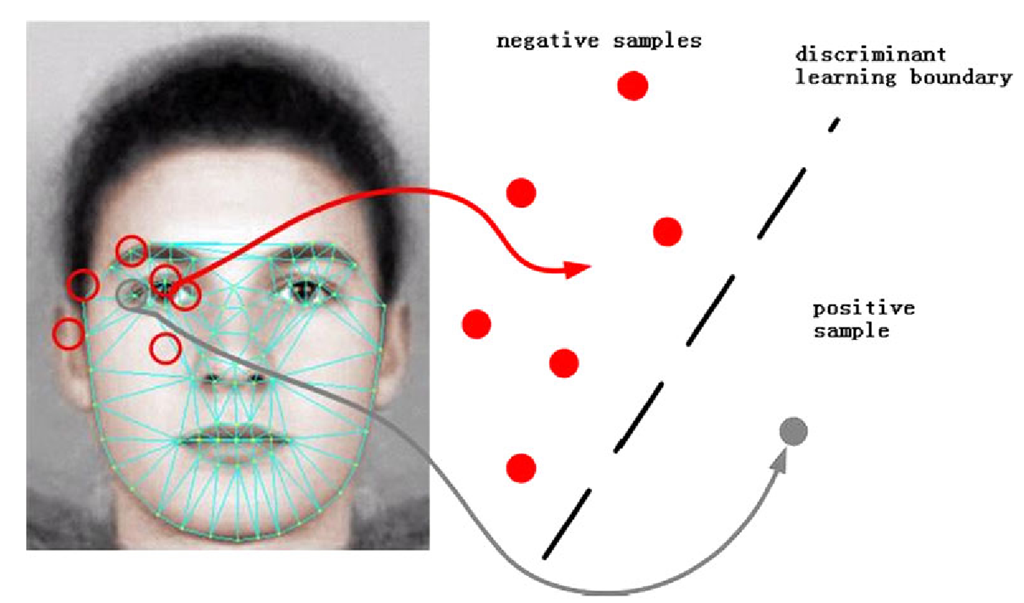
Những phần sai nằm ngoài mắt cũng được thu thập và sử dụng bootstrap method.

Những khả năng của việc ghép đôi được ước lượng và đánh giá, cuối cùng trung bình vị trí của 3 vùng tâm mắt đúng nhất được xem xét và đưa ra vị trí cuối cùng.

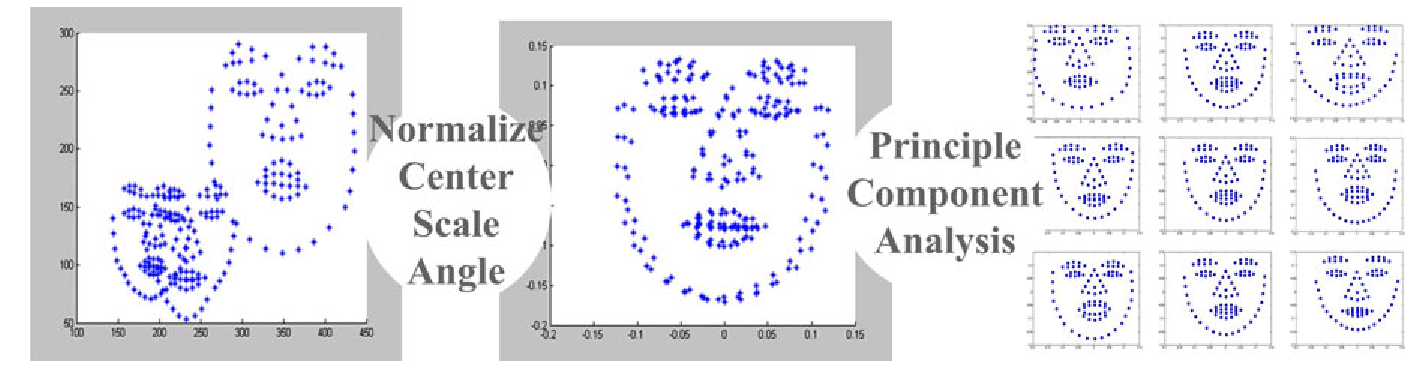
* 1. Shape Modeling

Định nghĩa hình dạng của 1 loạt những tọa độ của khuôn mặt như sau:

tmp35b0-453_thumb



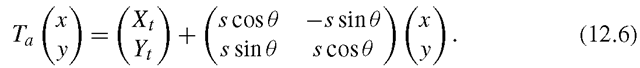
Hình . Ảnh 2d mô tả những phần đúng sai sau khi phân loại.



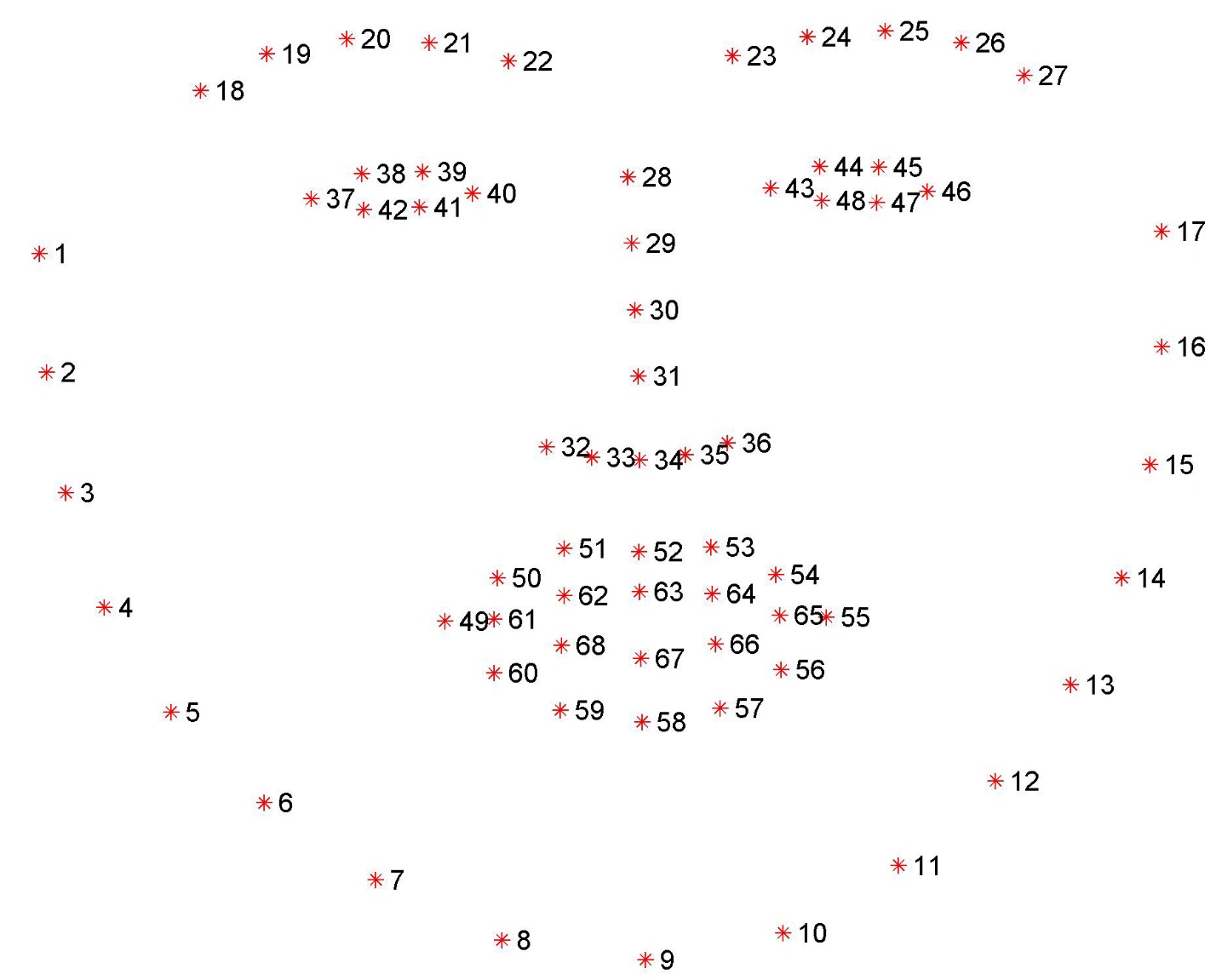
Hình . 2d shape modeling

**Principal component analysis (PCA)** phân tích những thành phần chính. PCA được dùng để đại diện cho những hình dạng dưới dạng vector bằng những phương trình tâm, tỉ lệ và góc được biểu diễn dưới những tham số hình học như sau:

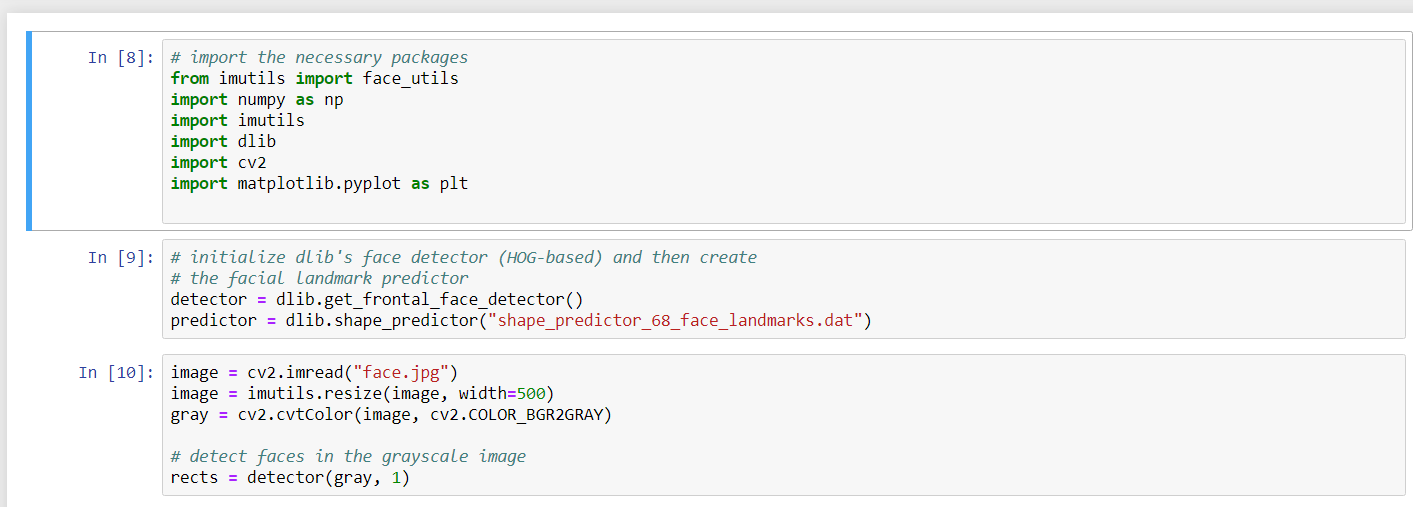


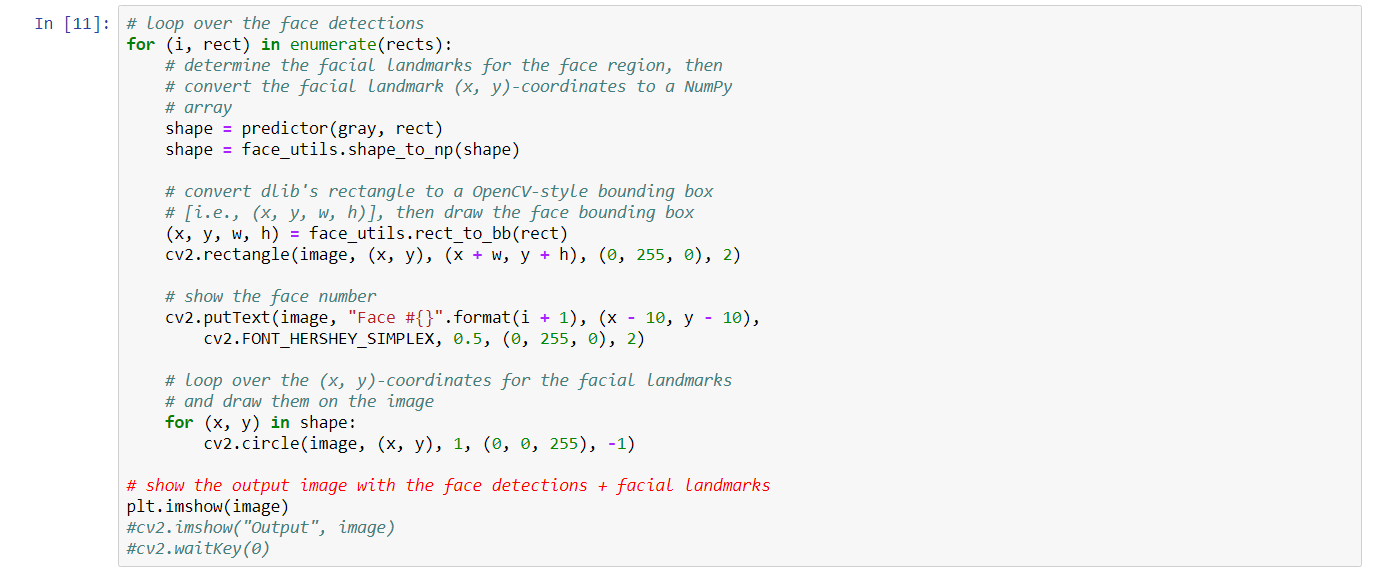


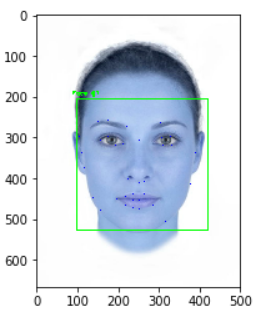
* 1. Example:



Hình . Mô tả về những vùng trên mặt







* 1. References

<http://what-when-how.com/face-recognition/facial-landmark-localization-face-recognition-techniques-part-1/>

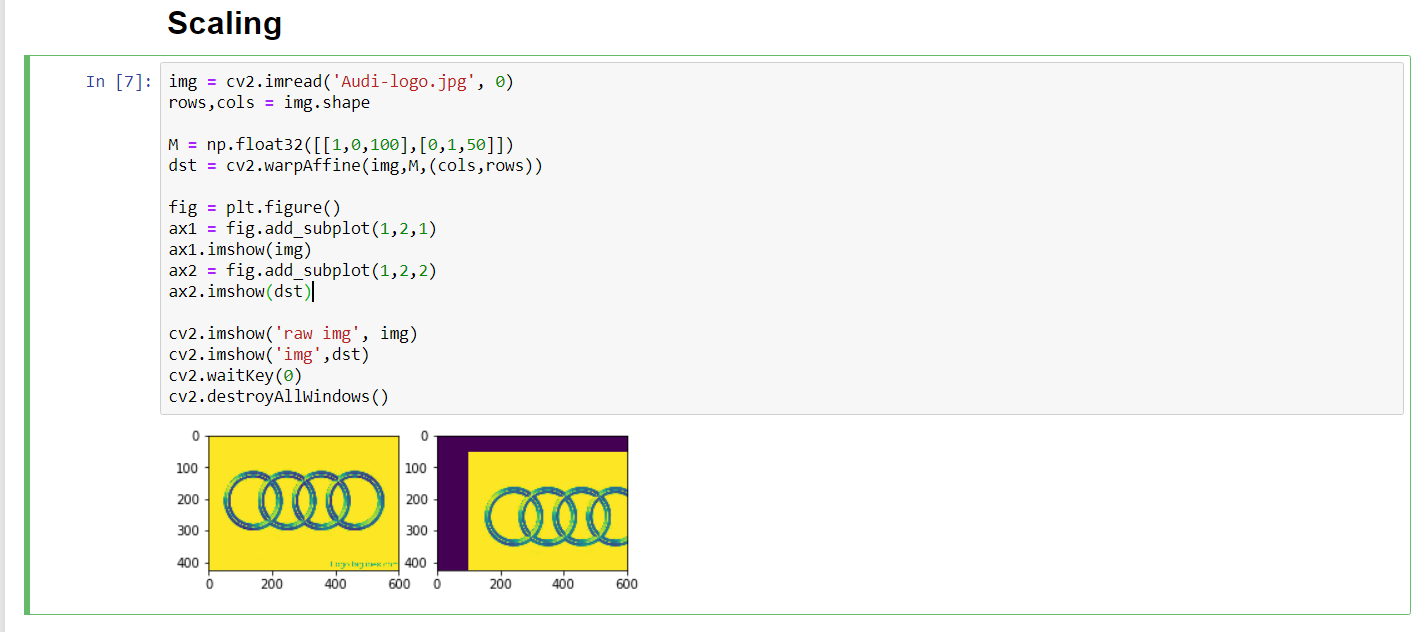
1. Affine transformations in Opencv:
   1. Transformations:

Opencv cung cấp 2 hàm: cv2.warpAffine và cv2.warpPerspective.

Cv2.warpAffine dùng matrix 2x3 và cv2.warpPerspective dùng 3x3

* 1. Scaling:

Sử dụng hàm cv2.resize(), có các loại nội suy khác nhau là cv2.INTER\_AREA, cv2.INTER\_CUBIC (chậm) và cv2.INTER\_LINEAR.



* 1. Rotation:

Góc quay của một bức hình \theta với ma trận biến đổi:

M = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta   \end{bmatrix}

Ngoài ra OpenCV cung cấp góc quay tại bất cứ điểm nào với ma trận:

\begin{bmatrix} \alpha &  \beta & (1- \alpha )  \cdot center.x -  \beta \cdot center.y \\ - \beta &  \alpha &  \beta \cdot center.x + (1- \alpha )  \cdot center.y \end{bmatrix}

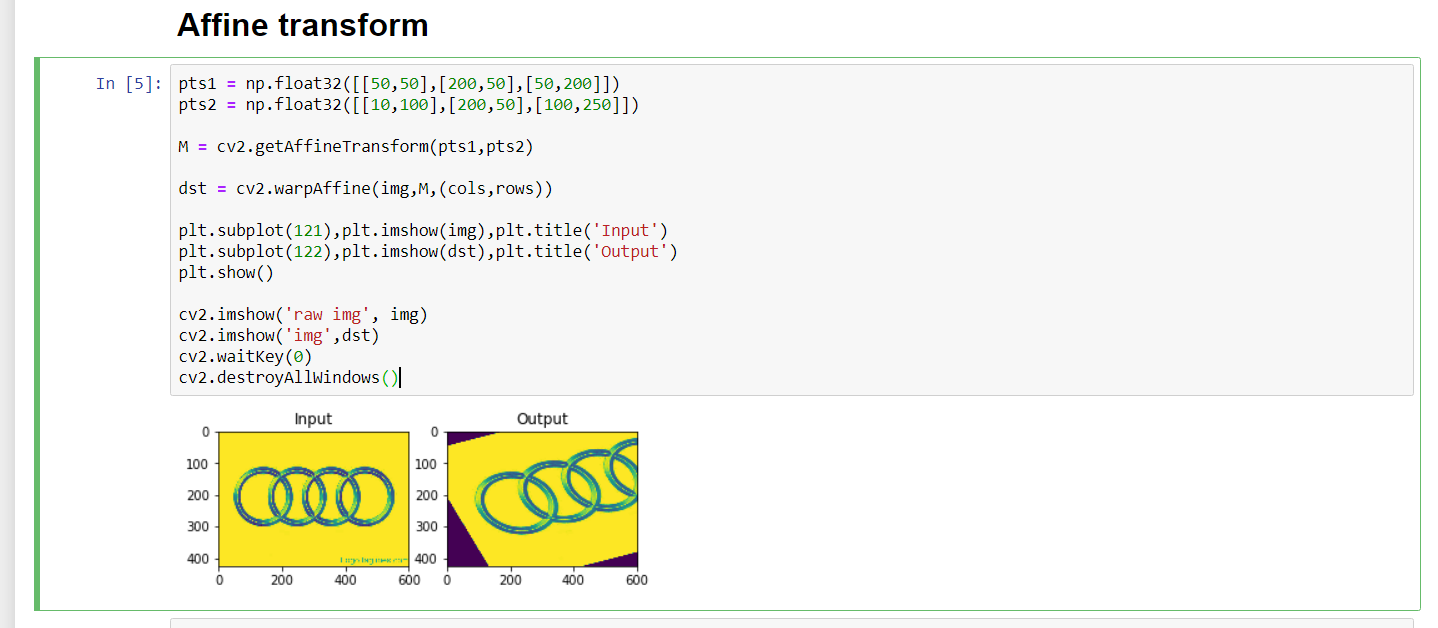
Tại

\begin{array}{l} \alpha =  scale \cdot \cos \theta , \\ \beta =  scale \cdot \sin \theta \end{array}



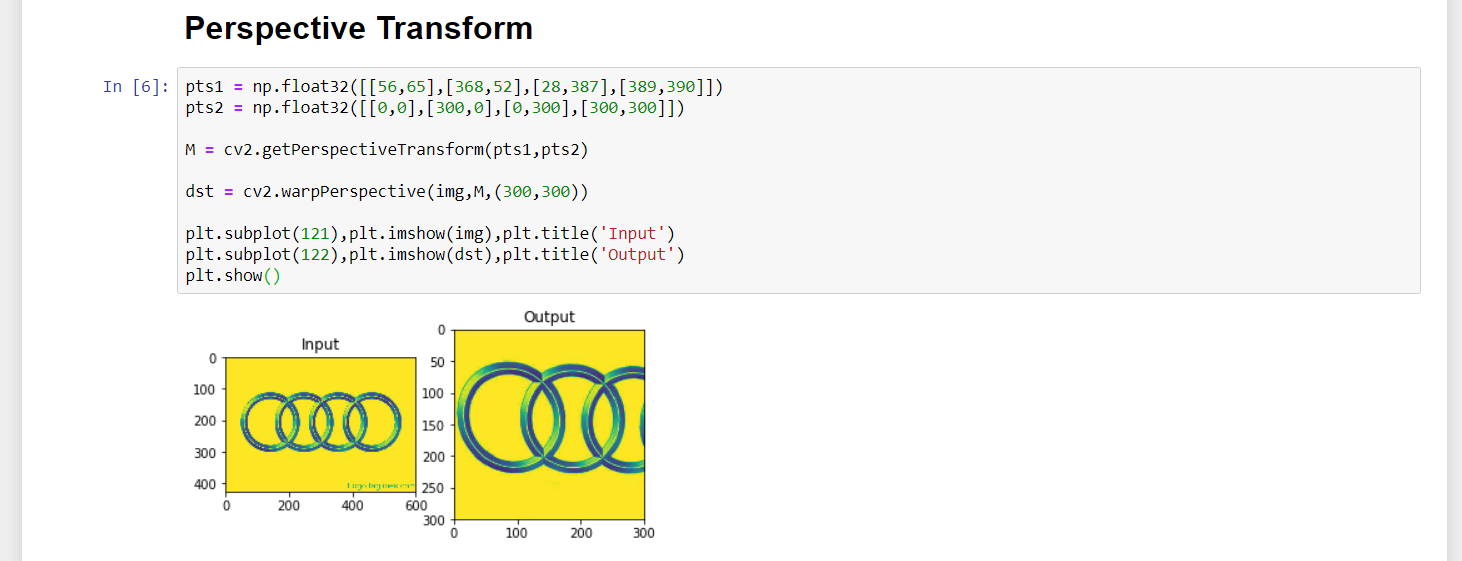
* 1. Affine Transformation:

Trong affine transformation, tất cả những đường song song trong hình gốc, sau khi xử lý thì output vẫn song song. Để tìm được ma trận biến đổi, ta cần 3 điểm từ hình input và sử dụng hàm cv2.getAffineTransform sẽ tạo ra ma trận đó và dùng cv2.warpAffine để xử lý ảnh gốc.



* 1. Perspective Transformation:

Để dùng, ta cần ma trận biến đổi 3x3. Đường thẳng sẽ vẫn là đưởng thẳng sau ghi xử lý. Để tìm được ma trận này ta cần 4 điểm từ ảnh input. Và 3 trong 4 điểm không nên thẳng hang. Để tạo ma trận ta dùng hàm cv2.getPerspectiveTransform. Sau đó sử dụng cv2.warpPerspective với ma trận 3x3 đã tìm được.

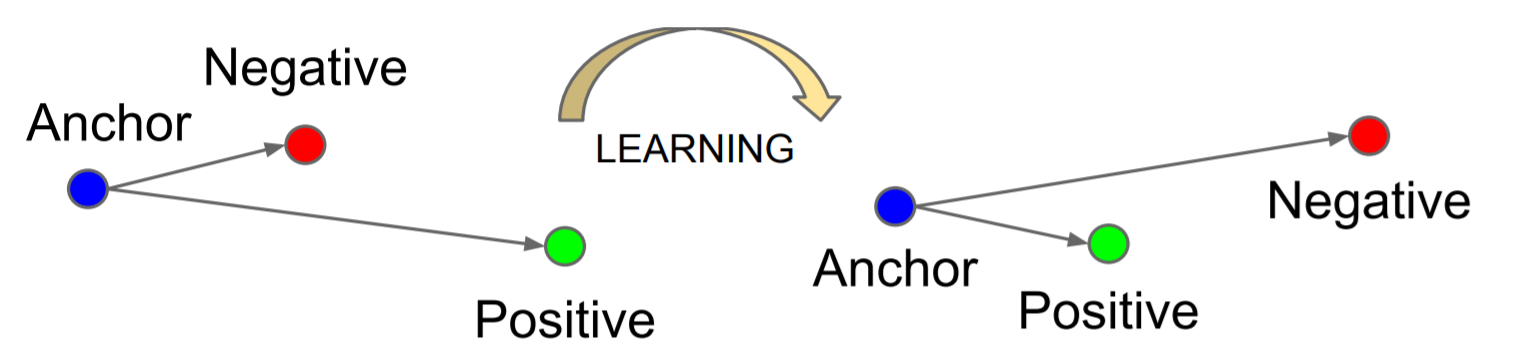


* 1. References;

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html>

1. Encoding faces and Face recognition:
   1. Triplet Loss:

Triple Loss sủ dụng cùng loại logic i.e dùng để giảm khoảng cách/ độ sai lệch giữa những thứ gần giống nhau và cùng 1 lúc cũng làm tang độ giống nhau

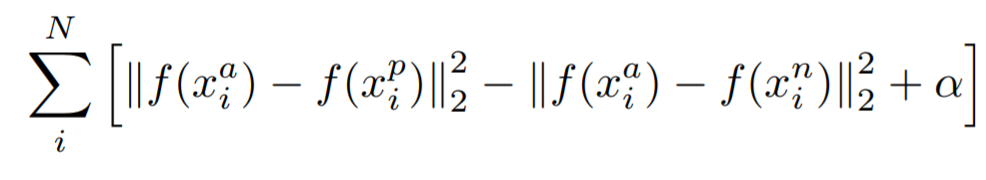


Khi huấn luyện, ta tinh chỉnh sao cho những thông số weight khác biệt ít nhất có thể. Và trong thời điểm đó cũng tăng mức khác biệt.

Để hiểu triplet loss dùng trong Facenet Model, ta có ví dụ như sau:

Input: 2 tấm hình khuôn mặt khác nhau của cùng 1 người và 1 tấm của người khác.

Facenet Model sẽ xử lý ảnh thành không gian 128 chiều và đưa ra vector với size 128 bao gồm tọa độ. Ta kiểm tra sự khác nhau thông qua những điểm khác nhau nhiều hay ít (nhiều => giống, ít => khác)

Công thức toán học thực hiện như sau:  


Hàm f(x) lấy x làm input và output là vector 128 chiều

Tóm lại:

* + Giảm khoảng cách giữa gốc và ảnh đúng (Đúng là label đúng)
  + Tăng khoảng cách giữa gốc và ảnh sai
  1. Face Recognition với Facenet:

Ta sẽ thực hiện tách feature và sử dụng mô hình máy học.

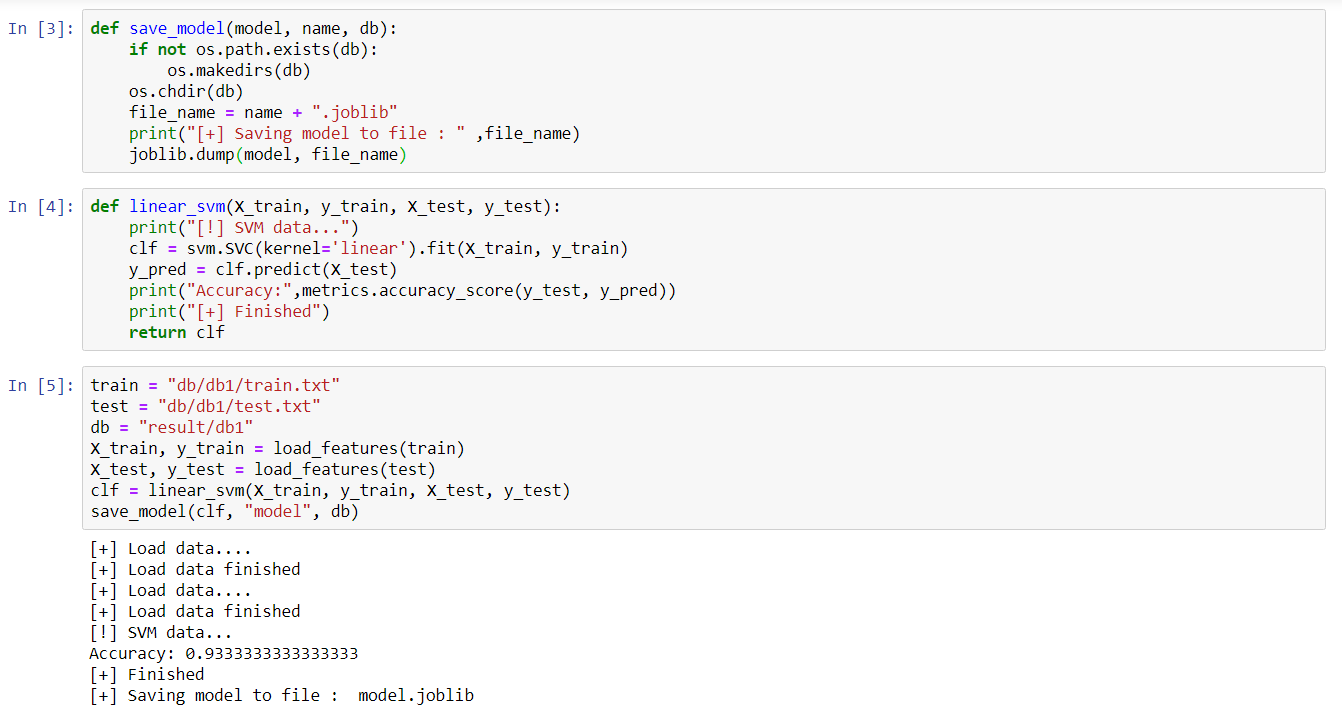
* + Sử dung một bộ face detector để tách phần ảnh có gương mặt người ra. Các phương pháp hiện đại sử dụng MTCNN (Multi-task Cascaded Convolutional Networks), ưu điểm là mô hình này detect được gương mặt ở nhiều góc độ, và có thể nhận diện cả face landmark với thời gian xử lý khá tốt (link paper[[1]](#footnote-1)). Sau đó ta resize hình lại một kích cỡ xác định (tùy chọn), facenet thì là 160x160, gần đây thì arcface dùng cỡ 112x112, việc chọn cỡ này là tùy theo yêu cầu về phần cứng hoặc độ chính xác.
  + Thực hiện các bước “lọc” bộ dataset. Thực tế là các bộ dataset lớn như vggface hay ms 1m đều có nhiều “nhiễu”, tức các hình không chứa gương mặt hoặc gương mặt khác với người cần xét. Vì vậy có thể dùng một mô hình nhận diện gương mặt khác để tìm và loại bỏ những hình “nhiễu” ấy. Hoặc đơn giản là ta có thể lấy clean list (danh sách các hình chuẩn) đã được lọc sẵn.
  + Thực hiện bước normalize cho tập dataset đã chuẩn bị. Cách normalize cũng giống như cách normalize cho CNN bình thường. Các bước train: *Đầu vào là tập dataset đã chuẩn bị, CNN có thể là các CNN như inception, mobilenet,… với đầu ra là embbeding vector như đã nói, embedding vector này có thể có 128 chiều hoặc 512 chiều,… tùy ý. Tóm lại ta chỉ cần lấy cấu trúc inception hoặc mobilenet, bỏ lớp cuối và thêm vào lớp embedding vector (không có activation)*
  + Forward pass trên tập data, lưu các embedding vector đã tính trên minibatch lại.
  + Chia minibatch thành 3 phần là positive, negative và anchor, trong đó anchor là phần “neo” giống như “tâm” của cụm embedding vector, sau đó chọn phần positive là những embedding vector là cùng người với anchor, negative là khác người
  + Tính Triplet loss là khoảng cách giữa các embbeding vector anchor và positive trừ cho khoảng cách giữa các embbeding vector anchor và negative (tức ta tìm cách làm giảm khoẳng cách từ các vector positive tới anchor và ngược lại đẩy các vector negative đi xa khỏi anchor)

Một số hàm loss softmax và tác dụng phân cụm của chúng (paper SphereFace) Để nói kĩ về các hàm này và triplet loss, có lẽ sẽ cần một bài viết khác.

Sau khi đã train trên các tập data lớn thì kết quả đạt được là một CNN có thể tách được các đặc trưng hữu ích để phân loại gương mặt. Lúc này để nhận diện người với người ta cần các bước: Huấn luyện:

1. Tạo một bộ dataset khác cho đối tượng mình muốn nhận diện.
2. Lưu bộ dataset đó theo cấu trúc như bộ dataset lớn đã train (để dễ tính lại các embedding vector), cấu trúc là thư mục chứ ảnh của cùng một người - một người tầm 10-20 ảnh, chú ý là đủ các góc độ sẽ cho hiệu quả hơn.
3. Tính embedding vector cho tập đã thu thập, lưu các embedding này vào file nào đó (tùy chọn)
4. Huấn luyện kNN hoặc SVM trên tập vector để phân loại người với người.
   1. Example:
      1. Training:





* + 1. Testing:



* 1. References:

Machinelearningcoban.com



1. <https://kpzhang93.github.io/MTCNN_face_detection_alignment/index.html> [↑](#footnote-ref-1)