TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



BÁO CÁO ĐỒ ÁN CUỐI KÌ:

DỰ ÁN CÔNG NGHỆ THÔNG TIN 2

**TÌM HIỂU PYTHON VÀ OPENCV CHO NHẬN DẠNG MẶT NGƯỜI**

Người hướng dẫn: **ĐẶNG MINH THẮNG**

Người thực hiện: **LÊ TRẦN NHẬT DUY – 51503044**

Lớp : **15050304**

Khóa : **19**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

# LỜI CẢM ƠN

Em xin chân thành cảm ơn thầy Đặng Minh Thắng đã dành thời gian tận tình hướng dẫn, cũng như hết mình giải đáp toàn bộ các thắc mắc, đồng thời định hướng, cho em được tiếp xúc với các công nghệ cũng như các vấn đề mới trong lĩnh vực Computer Vision nói chung và nhận diện khuôn mặt nói riêng.

**BÀI TẬP LỚN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Chúng tôi xin cam đoan đây là sản phẩm bài tập lớn của riêng chúng tôi. Các nội dung nghiên cứu, kết quả trong bài tập lớn này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong bài tập lớn còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung bài tập lớn của mình**. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm 2019*

*Tác giả*

Duy

*Lê Trần Nhật Duy*

# PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

# TÓM TẮT

Trong bài cáo cáo này, sẽ được trình bày chia ra làm 2 phần. Phần 1 là cơ sở lý thuyết, giới thiệu bài toán nhận diện khuôn mặt và trọng tâm của phương pháp áp dụng. Và phần 2 là quá trình hiện thực chạy demo, dựa vào quá trình huấn luyện trên tập dữ liệu có sẵn bằng cách áp dụng các phương pháp học máy, thư viện xử lý ảnh openCV trên python và deep learning.

# MỤC LỤC

[LỜI CẢM ƠN 2](#_Toc9797757)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 4](#_Toc9797758)

[TÓM TẮT 5](#_Toc9797759)

[MỤC LỤC 6](#_Toc9797760)

[DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT 7](#_Toc9797761)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 8](#_Toc9797762)

[CHƯƠNG 1 – BÀI TOÁN NHẬN DIỆN KHUÔN MẶT 9](#_Toc9797763)

[1.1 Giới thiệu bài toán 9](#_Toc9797764)

[1.2 OpenCV 10](#_Toc9797765)

[1.3 Phát hiện khuôn mặt bằng module DNN của OpenCV 11](#_Toc9797766)

[1.4 Facenet 12](#_Toc9797767)

[CHƯƠNG 2 – QUÁ TRÌNH ÁP DỤNG VÀ THỰC NGHIỆM 14](#_Toc9797768)

[2.1 Cấu trúc của project 14](#_Toc9797769)

[2.2 Quá trình thực hiện 17](#_Toc9797770)

[2.2.1 Bước 1: Extract\_embeddings.py 17](#_Toc9797771)

[2.2.2 Bước 2: Train\_model.py 22](#_Toc9797772)

[2.2.3 Bước 3: Recognize\_video.py 24](#_Toc9797773)

[2.3 Kết quả thực nghiệm 26](#_Toc9797774)

# DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

**CÁC KÝ HIỆU**

**CÁC CHỮ VIẾT TẮT**

DNN Deep Neural Network

CNN Convolutional Neural Network

SVM Support Vector Machine

SVC Support Vector Classifier

# DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1: Bài toán nhận diện khuôn mặt 9](#_Toc9797905)

[Hình 2: OpenCV 10](#_Toc9797906)

[Hình 3: Cơ chế hoạt động của OpenCV DNN 11](#_Toc9797907)

[Hình 4: Nguyên lý hàm Triplet Loss 13](#_Toc9797908)

[Hình 5: Cấu trúc các file của project 14](#_Toc9797909)

[Hình 6: Ví dụ nội dụng của một file pickle 15](#_Toc9797910)

[Hình 7: Cấu trúc dạng cây của project 16](#_Toc9797911)

[Hình 8: Các thư viện cần thiết 17](#_Toc9797912)

[Hình 9: Các biến cần truyền vào 17](#_Toc9797913)

[Hình 10: Sử dụng Caffe làm model phát hiện khuôn mặt 18](#_Toc9797914)

[Hình 11: Khai báo mô hình trích xuất vector đặc trưng 18](#_Toc9797915)

[Hình 12: Khai báo đường dẫn tới dataset 19](#_Toc9797916)

[Hình 13: Tạo 2 list để chứa 19](#_Toc9797917)

[Hình 14: Lấy nhãn tên từ thư mục dataset 19](#_Toc9797918)

[Hình 15: Áp dụng DNN để phát hiện khuôn mặt 20](#_Toc9797919)

[Hình 16: Xử lý phát hiện khuôn mặt 21](#_Toc9797920)

[Hình 17: Khoanh vùng khuôn mặt phát hiện được để trích xuất vector đặc trưng 21](#_Toc9797921)

[Hình 18: Import các thư viện học máy cần thiết 22](#_Toc9797922)

[Hình 19: Các biến trong file train\_model.py 22](#_Toc9797923)

[Hình 20: Huấn luyện mô hình SVM 23](#_Toc9797924)

[Hình 21: Các thư viện cần trong file recognize\_video.py 24](#_Toc9797925)

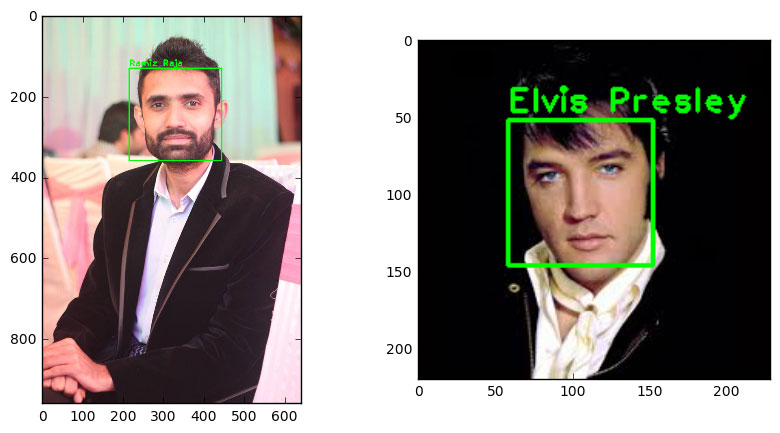
[Hình 22: Dùng mô hình để dự đoán xác suất 25](#_Toc9797926)

[Hình 23: Kết quả thu được 27](#_Toc9797927)

# CHƯƠNG 1 – BÀI TOÁN NHẬN DIỆN KHUÔN MẶT

# *1.1 Giới thiệu bài toán*

Hiện này, nhận diện khuôn mặt là một trong những bài toán rất phổ biến, được nhiều người quan tâm. Nội dung đằng sau bài toán này là việc làm sao cho máy tính có thể xác định vị trí khuôn mặt của người trong các ảnh hay video kĩ thuật số, sau đó xác định xem người đó là ai dựa bằng cách huấn luyện trên các tập dữ liệu. Trên thế giới đã có rất nhiều phương pháp đưa ra sử dụng như Haar Cascade, HOG và Deep Neural Network.



Hình 1: Bài toán nhận diện khuôn mặt

Mỗi hình ảnh đều được cấu tạo từ nhiều pixel (điểm ảnh), tức là các điểm rất nhỏ trong hình, được xếp thành nhiều cột và dòng,. Ví dụ rõ ràng nhất là trong thông số resolution (độ phân giải) của các TV kĩ thuật số hiện nay. 800 x 600 tức là rộng 800 pixel và dài 600 pixel.

Trong ảnh màu, pixel thường được thể hiện dưới kênh màu RGB (Red, Green, Blue). Do máy tính chỉ có thể hiểu được con số, nên toàn bộ pixel phải được thể hiện đúng với mỗi lượng màu đỏ, xanh lá, và xanh dương của nó. Trong ảnh xám, mỗi pixel là một con số duy nhất, thể hiện mức sáng, với 0 là màu đen, còn 255 là màu trắng. Vì thế nên phát hiện khuôn mặt thường được áp dụng lên ảnh xám.

# *1.2 OpenCV*

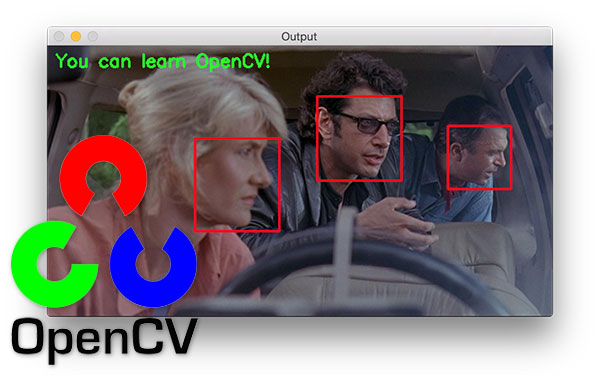
OpenCV (Open Source Computer Vision) là một thư viện lập trình mã nguồn mở cho phép người dùng thao tác chủ yếu trên các vấn đề liên quan tới thị giác máy tính (computer vision) thời gian thực và xử lý ảnh. Thư viện hiện đang được quản lý và phát triển bởi Intel. OpenCV thuộc dạng thư viện đa nền tảng, có thể hỗ trợ chung với các thư viện học sâu (deep learning) khác như TensorFlow, PyTorch, Caffe,…

OpenCV chủ yếu được viết và tối ưu bằng C++, nhưng vẫn có các giao diện hỗ trợ Python, Java, và các hệ điều hành thông dụng như Windows, Linux,…

Chức năng của OpenCV gồm: xử lí hiển thị, phát hiện hình ảnh, phân cụm và học máy, phát hiện vật thể,….

Nhờ vào các chức năng trên mà các ứng dụng của OpenCV cũng rất đa dạng, bao gồm:

* Hệ thống nhận diện khuôn mặt
* Điều khiển và tự động hóa robot
* Phân nhánh và nhận diện
* Thực tế ảo
* Phân tích hình ảnh y tế
* Kiểm tra và giám sát tự động



Hình 2: OpenCV

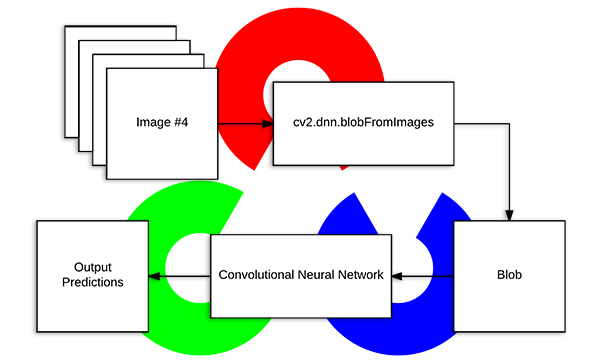
# *1.3 Phát hiện khuôn mặt bằng module DNN của OpenCV*

Có nhiều cách để phát hiện khuôn mặt trong ảnh hay video, trong bài báo cáo này sẽ áp dụng phương pháp Deep Neural Network của module OpenCV phiên bản 3.3.

Module OpenCV 3.3 hỗ trợ rất nhiều các framework deep learning như Caffe, TensorFlow, Torch,… (trong bài sẽ sử dụng Caffe). Caffe là framework mã nguồn mở, viết bằng C++ trên giao diện Python, hỗ trợ rất nhiều phương pháp học máy như CNN, LSTM, RCNN,… và tăng tốc xử lý trên GPU hay CPU.

Phương pháp phát hiện khuôn mặt bằng DNN này dựa vào thuật toán Single Shot Detector (SSD) trên mạng lưới ResNet.

Cơ chế hoạt động của phương pháp này gồm 3 bước: đầu tiên ta đưa vào một input image tức là ảnh đầu vào, sau đó, module sẽ đảm nhận chuyển toàn bộ các hình thành một blob, tức là một khối ảnh. (một blob là ảnh đầu vào sau khi đã qua các bước tiền xử lý, mean substraction và chuẩn hóa). Cuối cùng, chuyển blob vào một mô hình CNN để xử lý và cho kết quả đầu ra.



Hình 3: Cơ chế hoạt động của OpenCV DNN

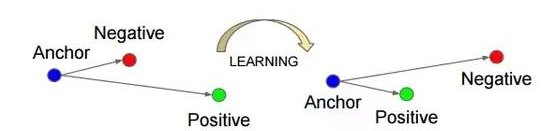
# *1.4 Facenet*

Sau bước phát hiện mặt người sẽ tới bước nhận dạng, ở đây sẽ áp dụng OpenFace, một module viết bằng thư viện Torch của Python dựa vào dự án nhận diện khuôn mặt FaceNet của Google.

Facenet, thực chất là một CNN có nhiệm vụ tách các đặc trưng của một ảnh mặt. Điểm đặc biệt tạo nên sự khác biệt của Facenet là nó sử dụng hàm lỗi Triplet (Triplet Loss) để tối thiểu hóa khoảng cách giữa các gương mặt tương đồng và tối đa hóa khoảng cách đến những gương mặt không tương đồng, vì vậy facenet có thể phân biệt rất chính xác người với người. Quá trình train CNN được thực hiện trên các bộ dataset lớn (ví dụ như vggface hoặc MS 1 million), kết quả là một tập embedding vector (vector đặc trưng), sau đó tách ra thành từng cụm để thực hiện phân loại bằng kNN hoặc SVM

Quá trình training dựa trên triplet loss như sau:

* Tiền xử lý data*:*Chú ý là các bộ dataset kể trên thường có dạng là các folder với tên là tên của một người, trong folder là hình của người đó, mỗi người tầm 100-1000 hình, một bộ data thì khoảng vài nghìn người
* Sử dung một bộ face detector để tách phần ảnh có gương mặt người ra (trong báo cáo sẽ sử dụng DNN của OpenCV như đã trình bày)
* 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.
* 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… với đầu ra là embbeding vector như đã nói, embedding vector này có thể có 128 chiều hoặc 512 chiều,…
* Chia vector đã train 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 khoảng cách từ các vector positive tới anchor và ngược lại đẩy các vector negative đi xa khỏi anchor)



Hình 4: Nguyên lý hàm Triplet Loss

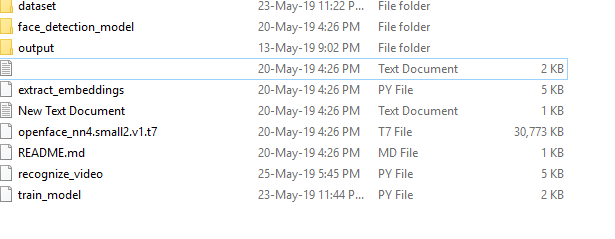
# CHƯƠNG 2 – QUÁ TRÌNH ÁP DỤNG VÀ THỰC NGHIỆM

Link github của project: <https://github.com/slenderman2211/Face-Reconigtion>

# *2.1 Cấu trúc của project*

Để giải quyết bài toán về nhận diện mặt người, ta sẽ áp dụng các phương pháp nêu trên để xử lý đoạn video truyền trực tiếp vào webcam máy tính theo thời gian thực, sau đó nhận diện các khuôn mặt xuất hiện trong khung hình.

Cụ thể, cấu trúc của project như hình sau:



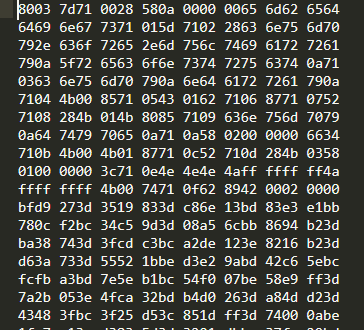
Hình 5: Cấu trúc các file của project

Tóm tắt quá trình hoạt động: đầu tiên, ta áp dụng phương pháp dò tìm bằng DNN của OpenCV (mục 1.3) để khoanh vùng khuôn mặt xuất hiện trong từng ảnh trong tập dữ liệu dataset, sau đó ta sẽ dùng mô hình OpenFace viết bằng thư viện PyTorch dựa trên FaceNet (mục 1.4) để trích xuất các vector đặc trưng (embedding) 128 chiều của từng khuôn mặt trong dataset trên. Sau khi đã thu được các vector, ta sẽ lấy chúng làm input đầu vào cho mô hình SVC để tiến hành train và phân loại, rồi áp dụng lên từng khung hình trong video truyền vào.

Thư mục dataset chứa các thư mục con, mỗi thư mục là một khuôn mặt cần nhận diện, với nhãn tên khác nhau. Trong project này sẽ, dataset sẽ bao gồm 2 người: Duy- 76 tấm, và Trâm- 61 tấm, tất cả đều có đuôi .jpg. Hai thư mục này sẽ là hai nhãn tên cần nhận diện, mỗi thư mục chứa một lượng ảnh nhất định để huấn luyện cho mô hình nhận diện.

Thư mục face\_detection\_model chứa các file mô hình Caffe và Tensorflow đã được huấn luyện sẵn, được hỗ trợ bởi OpenCV. Các mô hình này đảm nhận vai trò phát hiện và khoanh vùng khuôn mặt.

Thư mục ouput chứa các file .pickle sau khi đã hoàn thành các bước trên. Pickle là một dạng file được hỗ trợ bởi Python cho phép ta chuyển hầu hết mọi đối tượng và dữ liệu của Python thành dạng chuỗi. Điều này giúp thuận tiện trong quá trình thực thi, dễ dàng chuyển đổi qua lại giữa các máy và không cần phải lập đi lập lại các bước khi không có thay đổi.



Hình 6: Ví dụ nội dụng của một file pickle

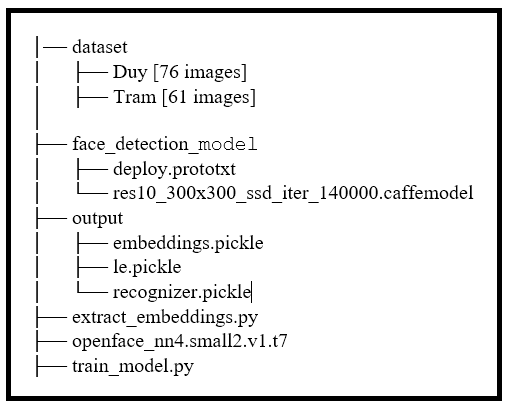
File extracting\_embeddings.py đảm nhận 2 vai trò: phát hiện khuôn mặt từ input và chuyển chúng qua các mạng neural để thành các vector đặc trưng 128 chiều bằng mô hình openFace (chính là file openface\_nn4.small2.v1. có đuôi thư viên Torch là t7)

File train\_model.py chính là mô hình SVM được huấn luyện bằng các vector đặc trưng có được từ file extracting\_embeddings.py để phân loại khuôn mặt theo các nhãn tên trong dataset.

Cuối cùng là file regconize\_video.py sẽ đảm nhận công việc tổng hợp các bước trên, áp dụng mô hình SVM đã được train sẵn và tiến hành trực tiếp nhận diện khuôn mặt webcam đưa vào.

Tóm lại, quá trình thực hiện có thể diễn tả ngắn gọn:

* File extracting\_embeddings.py sẽ đọc các hình ảnh trong dataset, nhận diện và chuyển nó thành các vector đặc trưng,
* Sau đó, file train\_model.py sẽ thu các vector đó để huấn luyện mô hình SVM để phân loại.
* Cuối cùng, file recognize\_video.py sẽ áp dụng SVM đã được train hoàn tất để nhận diện.



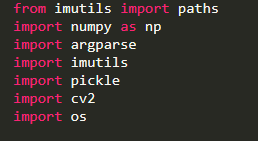
Hình 7: Cấu trúc dạng cây của project

# *2.2 Quá trình thực hiện*

Toàn bộ project này được chạy trên python 3.6.7, sử dụng thư viện OpenCV 4.0.0. với IDE là Sublime Text 3.

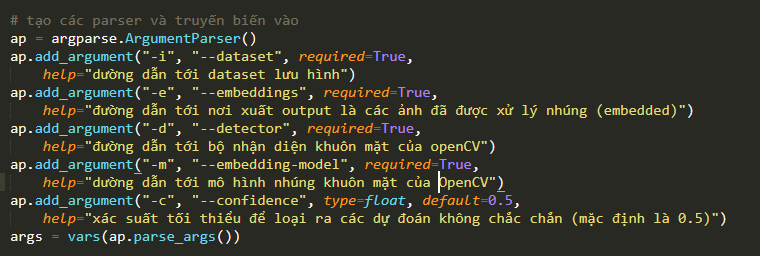
## ***2.2.1 Bước 1: Extract\_embeddings.py***

Theo như cấu trúc đã mô tả, bước 1 là phát hiện khuôn mặt trong dataset và trích xuất vector đặc trưng 128 chiều bằng file extract\_embedding.py.



Hình 8: Các thư viện cần thiết

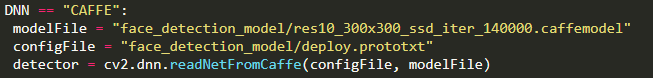
Hình trên là các thư viện cần import để tiến hành. Thư viện imutils có vai trò trích xuất đường dẫn tới các hình trong dataset, numpy để thực hiện các tính toán, argparse tạo các parser để truyền biến vào, pickle để xuất ra các file pickle, cv2 chính là openCV, thư viện OS để cung cấp các hàm giao tiếp với hệ điều hành.



Hình 9: Các biến cần truyền vào

Ở đây, dataset chính là đường dẫn tới tập dữ liệu, embeddings là đường dẫn để xuất output file pickle sau khi chạy chương trình, detector là dường dẫn tới các mô hình nhận diện khuôn mặt của OpenCV. Embedding\_model là dường dẫn tới mô hình xử lý quá trình trích xuất vector đặc trưng. Và cuối cùng là confidence, chính là xác suất tối thiểu để loại ra các dự đoán yếu, nếu xác suất tính ra thấp hơn xác suất này, thì sẽ loại bỏ kết quả đó (ở đây nếu không nhập thì cho mặc định là 0.5)

Ta bắt đầu tiến hành phát hiện khuôn mặt trong ảnh bằng cách sử dụng mô hình Caffe. Bất cứ mô hình nào hỗ trợ bởi OpenCV đều có 2 file, một file đuôi .protox chính là cấu trúc của mô hình (bao gồm các layers, tham số,…) và file .model chưa trọng số W. (ở đây ta dùng Caffe nên sẽ có đuôi là .caffemodel)



Hình 10: Sử dụng Caffe làm model phát hiện khuôn mặt

Cả 2 file cần đều được chứa trong thư mục face\_dectection\_model, sau đó ta truyền vào biến detector để chứa mô hình phát hiện.

Tiếp theo là tới mô hình trích xuất vector đặc trưng, do OpenFace được viết bằng Torch, nên ta dùng lệnh readNetFromTorch để đọc và truyền vào biến embedder



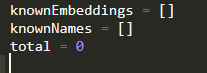
Hình 11: Khai báo mô hình trích xuất vector đặc trưng

Sau khi đã chuẩn bị sẵn sàng hết các mô hình, ta bắt khai báo đường dẫn tới thư mục dataset chứa các ảnh để huấn luyện. (chứa vào biến imagePaths)



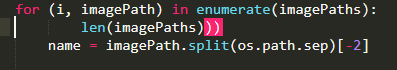
Hình 12: Khai báo đường dẫn tới dataset

Để theo dõi quá trình thực hiện cũng như thuận tiện trong quá trình xử lý, ta tạo 2 list, một list chứa các vector đặc trưng đã xử lý xong, và một list chưa nhãn tên đi kèm.



Hình 13: Tạo 2 list để chứa

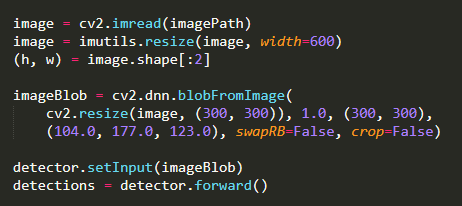
Sau khi xử lý phần ảnh và vector, ta phải chú ý tới phần nhãn tên. Ở đây trong dataset có 2 nhãn là Duy và Trâm, để lấy được 2 nhãn đó, ta thao tác như sau:



Hình 14: Lấy nhãn tên từ thư mục dataset

Ở đây, ta chạy một vòng lặp trong thư mục dataset để lấy hết các ảnh, còn phần nhãn tên, ta sẽ tách dường dẫn tới dataset thành một list có 3 thành phần: [ ‘dataset’, ‘Duy’, ‘0001.jpg’], sau đó lấy vị trí thứ 2 là Duy, cũng chính là nhãn tên ta cần, áp dụng tương tự với nhãn Trâm.

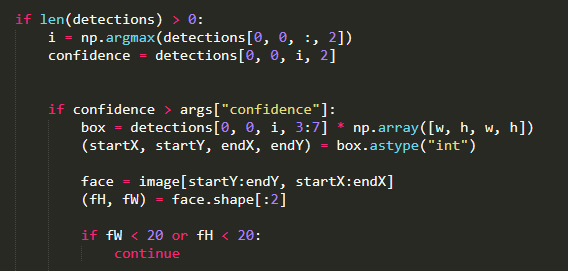
Tiếp theo là xử lý ảnh đầu vào, sau đó áp dụng DNN để phát hiện khuôn mặt.



Hình 15: Áp dụng DNN để phát hiện khuôn mặt

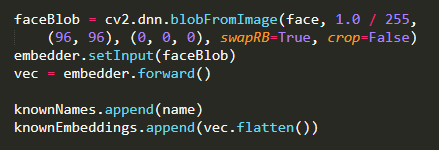
Ta bắt đầu sử dụng thư viện OpenCV để xử lý, đọc các file ảnh truyền vào, sau đó dùng thư viện imutils để chỉnh độ rộng ảnh thành 600, sau đó tiến hành thu chiều rộng và cao của ảnh. Cuối cùng, ta dùng hàm cv2.dnn.blobFromImage của OpenCV để thu blob từ hình, với các thông số như sau: kích cỡ hình 300x300, 3 kênh màu BGR để áp dụng mean substraction lần lượt là (104,177,123), swapRB=False để không đổi kênh màu thành RGB, và crop = False tức là không cắt sửa hình. Cuối cùng, ta chuyển blob tạo được thành đầu vào cho mạng nơ ron để tiến hành phát hiện.

Biến detections sau khi đã nhận input là một blob sẽ cho đầu ra là một list chứa các xác suất và tọa độ để khoanh vùng khuôn mặt lại. Ngay khi có phát hiện được ít nhất một khuôn mặt, ta sẽ chạy trường hợp if để trích vùng phát hiện nào có xác suất cao nhất (vì thế nên mỗi ảnh trong dataset nên tuyệt đối chỉ có một khuôn mặt), sau đó kiểm tra xem xác suất đó có cao hơn xác suất đã nhập vào không.



Hình 16: Xử lý phát hiện khuôn mặt

Nếu cao hơn thì tức là đã phát hiện khuôn mặt ngay tại ví trí đó, ta sẽ tiến hành cắt vùng khuôn mặt đó ra, sau đó tiếp tục đưa qua một hàm dnn để chuyển thành một blob thứ 2. Blob này sẽ là input của mô hình CNN trích xuất vector đặc trưng.



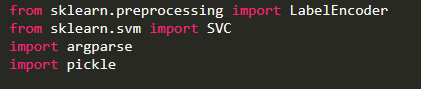
Hình 17: Khoanh vùng khuôn mặt phát hiện được để trích xuất vector đặc trưng

Với mỗi một vector đặc trưng đã xử lý, ta sẽ đưa vào trong list đã tạo lúc nãy, tương tự đối với các nhãn tên đi tương ứng.

Cuối cùng, ta chuyển toàn bộ các vector đặc trưng đã xử lý và các nhãn tên tương ứng vào trong một file pickle, đặt tên là **embedding.pickle** và được lưu trong thư mục ouput, file này sẽ đóng vai trò huấn luyện cho mô hình SVM trong bước tiếp theo.

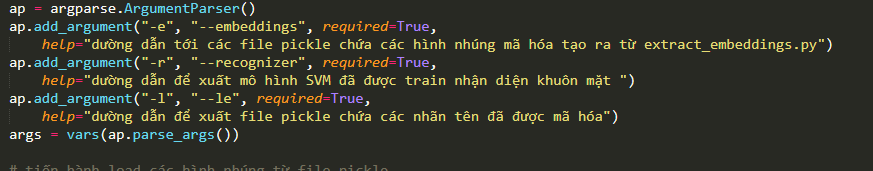
## ***2.2.2 Bước 2: Train\_model.py***

Sau khi hoàn thành bước 1, ta được kết quả là một file output chứa các vector đặc trưng của các khuôn mặt. Output này sẽ được đưa vào file train\_model.py để huấn luyện mô hình.



Hình 18: Import các thư viện học máy cần thiết

Ta sẽ dùng thư viện sklearn, và tiến hành huấn luyện trên mô hình học máy chuẩn SVM. Thư viện LabelEncoder giúp hỗ trợ tiền xử lý các nhãn tên để thuận tiện cho quá trình phân lớp.



Hình 19: Các biến trong file train\_model.py

Biến truyền vào của file train\_model.py có một số thay đổi, biến embeddings chính là dường dẫn tới file embedding.pickle ta thu được từ bước đầu tiên, cũng chính là dữ liệu cần để huấn luyện mô hình. Biến recognizer chính là đường dẫn để xuất output chính là mô hình sau khi đã huấn luyện hoàn chỉnh xong, điều này giúp thuận tiện trong quá trình thực thi, tránh phải huấn luyện nhiều lần mỗi khi chạy. Cuối cùng, biến le là đường dẫn xuất output là các nhãn tên đã được mã hóa. Hai file output này sẽ đóng vai trò quan trọng cho bước thứ 3 là nhận diện trực tiếp.

Đầu tiên, ta đọc file embedding.pickle để làm dữ liệu huấn luyện mô hình.



Sau đó, ta bắt đầu sử dụng thư viện LabelEncoder(), để đọc các nhãn tên, và truyền vào đó là list name đã tạo ra từ bước 1.



Tiếp theo tiến hành huấn luyện mô hình dựa trên dữ liệu các vector đặc trưng.



Hình 20: Huấn luyện mô hình SVM

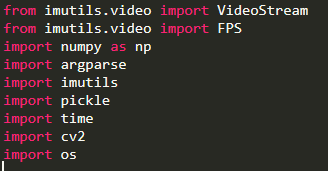
Ở đây, ta dùng một SVC (Suport Vector Classifier) để phân loại theo từng nhãn tên, chỉ số C là hàm lỗi, thể hiện mức độ “kĩ” của mô hình, biến C càng cao thì mô hình sẽ càng bám sát dataset, nhưng như vậy có thể gây ra quá khớp (overfit). Theo nghiên cứu, thì chỉ số C= 1 là lý tưởng nhất cho bài toán nhận diện khuôn mặt, do tính chất thay đổi liên tục của nó.

Sau khi thực thi xong, ta tiến hành xuất 2 file output là **recognizer.pickle**: chính là mô hình đã huấn luyện hoàn tất, và file **le.pickle**, chứa các nhãn tên đi kèm đã mã hóa. Đây sẽ là output cho bước 3.

## ***2.2.3 Bước 3: Recognize\_video.py***

Sau khi đã có đủ các output, ta tới bước cuối cùng, nhận diện trực tiếp video truyền vào từ webcam theo thời gian thực. (được thực hiện trong file recognize\_video.py)

Quá trình thực hiện nhận diện khuôn mặt có lập lại vài quá trình ở bước 1, đó là thực hiện lại bước nhận diện mặt bằng DNN, sau đó tiếp tục dùng OpenFace để trích xuất vector đặc trưng, và truyền trực tiếp vào mô hình đã huấn luyện hoàn tất từ bước 2 để nhận diện.



Hình 21: Các thư viện cần trong file recognize\_video.py

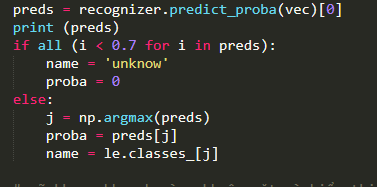
Các thư viện cần import tương tự như bước 1, tuy nhiên có thêm các thư viện videoStream và FPS để xử lý dòng video truyền vào từ webcam.

Các bước nhận diện và trích xuất vector đặc trưng đều tượng tư như bước 1

Để kích hoạt webcam và thu dòng Video, ta sử dụng các hàm sau:

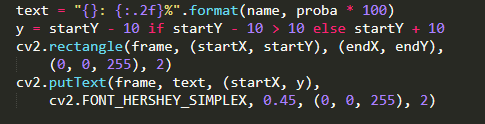


Ta để 2s để webcam có thời gian khởi động. Thay vì xử lý trên ảnh như bước 1, ta xử lý trên từng khung hình của video.



Hình 22: Dùng mô hình để dự đoán xác suất

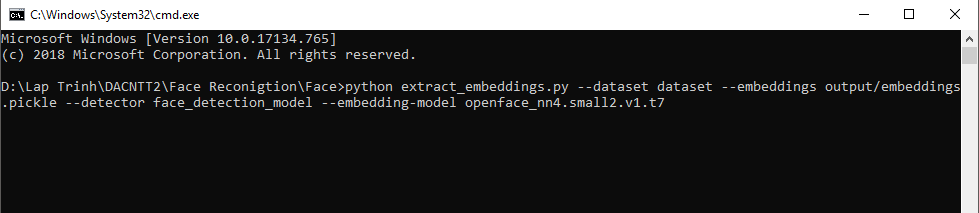
Sau khi đã qua bước trích xuất vector đặc trưng, ta được biến vec chứa kết quả. Sau đó ta chạy hàm predict\_proba bằng mô hình đã huấn luyện trước đó để dự đoán các xác suất của từng khung hình, ta được kết quả là biến preds dạng list chứa các xác suất của từng nhãn tên. Ta sẽ tìm xác suất của nhãn nào là cao nhất để hiển thị lên màn hình. Tuy nhiên, sẽ có những trường hợp gượng mặt xuất hiện không có trong dataset, vì thế, ta sẽ xét trong list xác suất nếu không có một con số nào vượt quá một ngưỡng cố định (ở đây ta cho là 70%), thì chứng tỏ gương mặt này không có trong dataset, ta sẽ chuyển name thành ‘unknow’ và cho xác suất là 0.



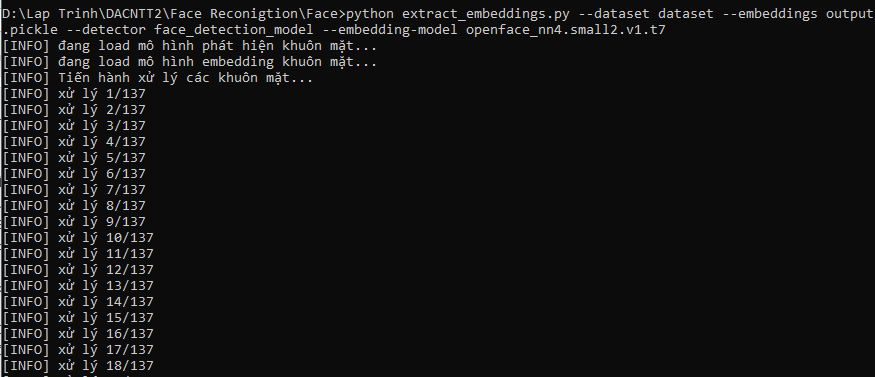
Sau khi đã tìm được, ta sẽ khoanh vùng khuôn mặt lại bằng một hình tứ giác đỏ, kèm theo hiển thị nhãn tên của người đã được phân loại và xác suất.

# *2.3 Kết quả thực nghiệm*

Thực hiện bước 1, chạy file extracting\_embeddings.py để tạo output embedding.pickle

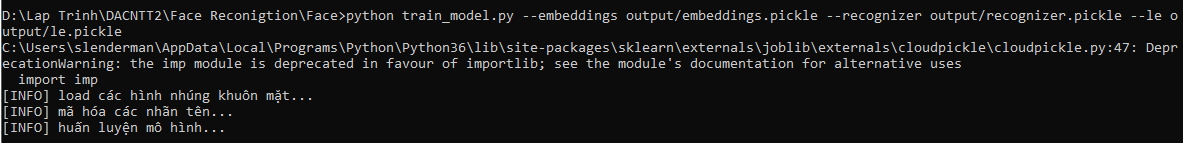


Ở đây, ta dẫn dường dẫn tới những file cần thiết cũng như dẫn cả output.



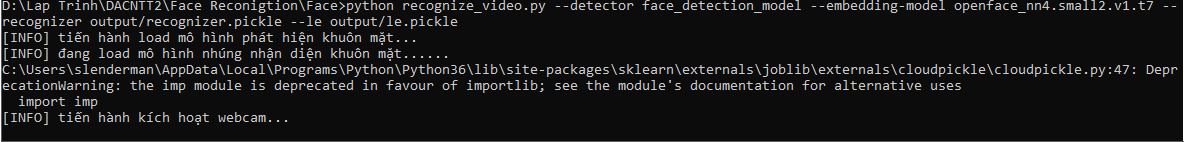
Tổng cộng sẽ xử lý 137 tấm hình trong dataset, sau khi hoàn thành, ta được file embeddings.pickle.

Tiếp theo qua bước 2, ta tiến hành huấn luyện mô hình.

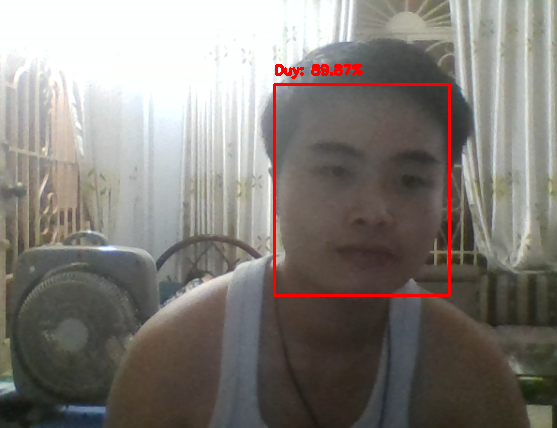


Ta dẫn dường dẫn tới file pickle đã tạo được từ bước 1, đồng thời được 2 output là recognizer.pickle và le.pickle

Sau khi đã có đủ output, ta tiến hành nhận diện khuôn mặt.



Kết quả thu được khi kích hoạt webcam:



Hình 23: Kết quả thu được

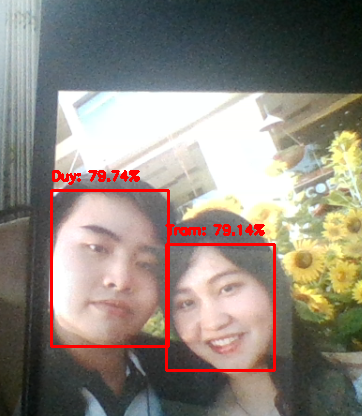
Thử trên nhiều góc độ khác nhau:

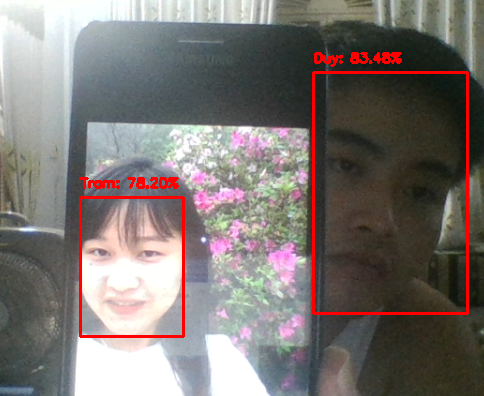




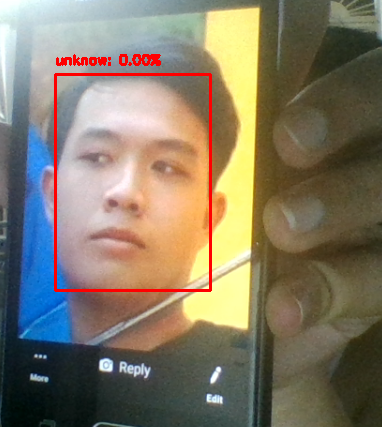


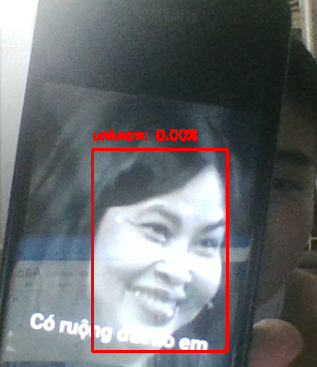
Kiểm tra trường hợp nhiều người (đều có trong dataset)



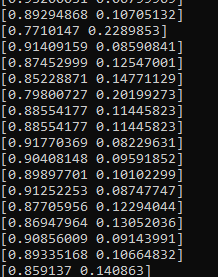


Kiểm tra trường hợp mặt không có trong dataset





Kết quả cho ra có độ chính xác khá tốt, tuy nhiên vẫn còn một số trường hợp sai sót nhưng không quá nhiều.



Mỗi list xuất hiện khi chương trình chạy chính là biến preds khi mô hình dự đoán trên từng khung hình, cũng chính là các list chứa xác suất của từng nhãn tên. Ở đây, nhãn tên Duy có xác suất cao nhất nên sẽ được chọn ra thay vì nhãn tên Trâm.

**TÀI LIỆU THAM KHẢO**

**Tiếng Việt**

1. <https://forum.machinelearningcoban.com/t/deep-learning-va-bai-toan-face-recognition/1776>
2. <https://www.facebook.com/338608676766270/posts/354364085190729/?app=fbl>

**Tiếng Anh**

1. <https://www.learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/>
2. [https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/.](https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/)