**ĐẠI HỌC ĐÀ NẴNG**

**KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



**ĐỒ ÁN CƠ SỞ 4**

**ĐỀ TÀI:**

**NHẬN DIỆN KHUÔN MẶT ĐEO KHẨU TRANG**

**ĐẢM BẢO AN TOÀN TRONG GIAI ĐOẠN DỊCH COVID-19**

Sinh viên thực hiện : **TRẦN DƯƠNG QUANG**

**HỒ NGUYỄN XUÂN TÂY**

Giảng viên hướng dẫn : **TS.** **HUỲNH NGỌC THỌ**

Lớp : **18IT3**

***Đà nẵng, tháng 12 năm 2020***

**ĐẠI HỌC ĐÀ NẴNG**

**KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

**ĐỒ ÁN CƠ SỞ 4**

**ĐỀ TÀI:**

**NHẬN DIỆN KHUÔN MẶT ĐEO KHẨU TRANG**

**ĐẢM BẢO AN TOÀN TRONG GIAI ĐOẠN DỊCH COVID-19**

Sinh viên thực hiện : **TRẦN DƯƠNG QUANG**

**HỒ NGUYỄN XUÂN TÂY**

Giảng viên hướng dẫn : **TS.** **HUỲNH NGỌC THỌ**

Lớp : **18IT3**

# 

# **LỜI MỞ ĐẦU**

Theo số liệu thống kê của Google cho đến thời điểm hiện tại, về tình hình dịch bệnh trên toàn thế giới.

Trên toàn thế giới có hơn 81,2 triệu ca nhiễm, đã bình phục 45,9 và tổng số ca tử vong đã lên đến 1,77 triệu ca. Con số ca nhiễm và số tử vong vẫn không có dấu hiệu ngừng tăng.

Riêng đối với Việt Nam tính ngày 29/12/2020 đã có tổng số ca nhiễm là 1,451 triệu ca, trong đó đã có 1,303 triệu ca đã bình phục và 35 ca đã tử vong. Đó là một con số không hề nhỏ và đáng bảo động.

Con số thông kê về số ca nhiễm trên toàn thế giới vẫn chưa có dấu hiệu suy giảm mà ngày càng gia tăng. Chính là vẫn đề nóng hổi toàn cầu và đáng quan tâm của các quốc gia đang có số ca nhiễm dẫn đầu như Mỹ, Ấn Độ, Bra-xin,…

Ở nước ta, từ đầu năm 2020 đến nay đã sảy ra nhiều ca tử vong đáng tiết và là lời cảnh tỉnh trong công tác phòng chống dịch bệnh. Công dân Việt Nam còn khá chủ quan và lơ là về việc tự trang bị và bảo vệ sức khỏe bản thân.

Virus corona xuất hiện tại Vũ Hán (Trung Quốc) làm lây nhiễm bệnh cho hơn 4.600 người và lấy đi sinh mạng của 169 người tính đến ngày 30-1-2020. Vậy virus corona là gì mà khiến cả thế giới lo sợ sẽ trở thành đại dịch toàn cầu?

Giống như dịch SARS và MERS trước đây, virus corona mới từ Vũ Hán cũng gây dịch bệnh viêm phổi cấp và thậm chí gây chết người. Triệu chứng coronavirus chủng mới thường không thể hiện rõ ở một số người trong thời gian ủ bệnh, một số người khác thì có biểu hiện phổ biến như sốt, ho khan, khó thở, đau cơ và mệt mỏi.

Hiện tại vẫn chưa rõ ràng virus SARS-CoV-2 đến từ khu chợ có nhiều động vật nhiễm virus sinh sống hay không nhưng loại virus này cực kỳ nguy hiểm. Virus gây bệnh viêm phổi lạ lây từ người sang người, thậm chí có thể đột biến trên đường đi và dần trở nên độc hại hơn.

Các virus corona ở người bị nhiễm bệnh thường lây sang người khác qua những con đường dưới đây:

* Lây qua đường máu
* Không khí do ho và hắt hơi
* Lây qua đường phân hay đường tiêu hóa
* Tiếp xúc với người bệnh như chạm hoặc bắt tay
* Chạm vào một vật hoặc bề mặt có virus rồi chạm tay vào mặt

Chúng ta nên trang bị đầy đủ khẩu trang, nước xịt khuẩn để đảm bảo an toàn cho bản thân và mọi người xung quanh.

Nghiên cứu cho thấy, việc sử dụng khẩu trang có thể hạn chế được sự lây lan và truyền bệnh qua đường hô hấp mỗi khi chúng ta giao tiếp hay hắc hơi ở khoảng cách.

Tuy nhiên thực trang cho thấy hiện nay vẫn còn nhiều người chủ quan với sự nguy hiểm của dịch bệnh, và khả năng lây lan của chuẩn virus này. Nhiều người khi ra khỏi nhà, ở nơi công cộng, gặp gỡ người xung quanh vẫn không đeo khẩu trang và điều này gây gia tăng khả năng lây lang dịch bệnh trong khi dịch vẫn còn hoành hành.

Chính vì tất cả những lí do trên mà ứng dụng nhận diện đeo khẩu trang này được đưa ra để nghiên cứu và thực hiện.

**LỜI CẢM ƠN**

Nhóm đồ án chúng em xin gửi lời cảm ơn đến thầy Huỳnh Ngọc Thọ– giảng viên hướng dẫn đồ án cơ sở 4 đã trang bị cho em những kiến thức, kỹ năng cơ bản cần có để hoàn thành đề tài này.

Tuy nhiên trong quá trình nghiêm cứu đề tài, do kiến thức chuyên ngành còn hạn chế nên nhóm chúng em vẫn còn nhiều thiếu sót khi tìm hiểu, đánh giá và trình bày về đề tài. Rất mong nhận được sự quan tấm và góp ý của thầy/cô bộ môn để tài của em được đầy đủ và hoàn chỉnh hơn

Xin chân thành cảm ơn.



**MỤC LỤC**

[**LỜI MỞ ĐẦU** 3](#_Toc60263128)

[**Chương 1: Giới thiệu** 10](#_Toc60263129)

[**1.1** **Tổng quan** 10](#_Toc60263130)

[**1.2** **Mục đích** 10](#_Toc60263131)

[**1.3** **Phương pháp** 11](#_Toc60263132)

[**1.3.1 Đối tượng** 11](#_Toc60263133)

[**1.3.2 Phạm vi nghiên cứu** 11](#_Toc60263134)

[**1.4** **Cấu trúc đồ án** 11](#_Toc60263135)

[**Chương 2: Nghiên cứu tổng quan** 12](#_Toc60263136)

[**2.1 Các phương pháp** 12](#_Toc60263137)

[**2.1.1 Bài toán nhận diện khuôn mặt** 12](#_Toc60263138)

[**2.1.2 Tổng quan kiến trúc của một hệ thống nhận dạng khuôn mặt** 13](#_Toc60263139)

[**2.1.3 Xác định khuôn hình khẩu trang trên khuôn mặt** 13](#_Toc60263140)

[**2.2 Các công nghệ thực hiện và các khái niệm liên quan** 14](#_Toc60263141)

[**2.2.1 Giới thiệu ngôn ngữ lập trình Python** 15](#_Toc60263142)

[**2.2.2 Giới thiệu về thư viện OpenCV** 15](#_Toc60263143)

[**2.2.3 Deep Learning** 16](#_Toc60263144)

[**2.2.4** **Thư viện của Deep Learning – Keras/Deep Learning** 17](#_Toc60263145)

[**2.3 Hạn chế, tồn tại của các phương pháp** 18](#_Toc60263146)

[**Chương 3: Phân tích thiết kế hệ thống** 20](#_Toc60263147)

[**3.1 Mô hình tổng quan của hệ thống nghiên cứu** 20](#_Toc60263148)

[**3.1.1 Luồng hoạt động của hệ thống** 20](#_Toc60263149)

[**3.2 Phân tích chi tiết luồng hoạt động của hệ thống** 21](#_Toc60263150)

[**3.2.1 Giai đoạn huấn luyện dữ liệu cho hệ thống** 21](#_Toc60263151)

[**3.2.2 Giai đoạn áp dụng và triển khai phân tích nhận diện của hệ thống** 22](#_Toc60263152)

[**3.3 Phân tích tập dữ liệu (dataset)** 23](#_Toc60263153)

[**3.3.1 Xây dựng tập dữ liệu** 23](#_Toc60263154)

[**Chương 4: Triển khai xây dựng** 28](#_Toc60263155)

[**4.1 Cấu trúc dự án** 28](#_Toc60263156)

[**4.2 Triển khai kịch bản huấn luyện (training) hệ thống nhận diện khẩu trang** 28](#_Toc60263157)

[**4.3 Tiến hành huấn luyện hệ thống nhận diện khẩu trang với Keras/Tensorflow** 35](#_Toc60263158)

[**4.4 Triển khai nhận diện khẩu trang với video stream qua webcam** 36](#_Toc60263159)

[**Chương 5: Kết luận và hướng phát triển** 42](#_Toc60263160)

[**5.1 Kết luận** 42](#_Toc60263161)

[**5.2 Hướng phát triển** 42](#_Toc60263162)

**DANH MỤC HÌNH**

[**Hình 1-Hệ thống nhận diện khuôn mặt** 13](file:///E:\TAI%20LIEU%20HOC%20TAP%20NAM%203\HK1%20NAM%203\Do%20an%20co%20so%204\Báo%20cáo%20đồ%20án%20ứng%20dụng%20nhận%20diện%20khuôn%20mặt%20mang%20khẩu%20trang.docx#_Toc60260967)

[**Hình 2-Các bước chính trong hệ thống nhận dạng** 14](file:///E:\TAI%20LIEU%20HOC%20TAP%20NAM%203\HK1%20NAM%203\Do%20an%20co%20so%204\Báo%20cáo%20đồ%20án%20ứng%20dụng%20nhận%20diện%20khuôn%20mặt%20mang%20khẩu%20trang.docx#_Toc60260968)

[**Hình 3-Giới thiệu về thư viện OpenCV** 17](file:///E:\TAI%20LIEU%20HOC%20TAP%20NAM%203\HK1%20NAM%203\Do%20an%20co%20so%204\Báo%20cáo%20đồ%20án%20ứng%20dụng%20nhận%20diện%20khuôn%20mặt%20mang%20khẩu%20trang.docx#_Toc60260969)

[**Hình 4-Sơ đồ luồng hoạt động của hệ thống** 21](#_Toc60260970)

[**Hình 5-Sơ đồ luồng hoạt động của giai đoạn Train dữ liệu** 22](#_Toc60260971)

[**Hình 6-Sơ đồ luồng hoạt động phân tích và nhận diện của hệ thống** 23](#_Toc60260972)

[**Hình 7-Bức ảnh một người không đeo khẩu trang** 24](#_Toc60260973)

[**Hình 8-Nhận diện khuôn mặt trên bức ảnh nhờ Deep Learning** 25](#_Toc60260974)

[**Hình 9-Trích xuất ROI của khuôn mặt với OpenCV và NumPy cắt** 26](#_Toc60260975)

[**Hình 10-Đánh dấu các điểm mốc trên khuôn mặt** 26](#_Toc60260976)

[**Hình 11-Hình khẩu trang** 27](#_Toc60260977)

[**Hình 12-Hình ảnh khẩu trang được đặt lên khuôn mặt** 27](#_Toc60260978)

[**Hình 13-Bộ dữ liệu ảnh đeo khẩu trang nhân tạo** 28](#_Toc60260979)

[**Hình 14-Kết quả nhận diện khuôn mặt không mang khẩu trang** 42](#_Toc60260980)

[**Hình 15-Kết quả nhận diện khuôn mặt có đeo khẩu trang** 42](#_Toc60260981)

# **Chương 1: Giới thiệu**

* 1. **Tổng quan**

Phổ biến trong đời sống con người đã làm cho lượng thông tin thu được bằng hình ảnh ngày càng tăng. Theo đó, lĩnh vực xử lý ảnh cũng được chú trọng phát triển, ứng dụng rộng rãi trong đời sống xã hội hiện đại. Không chỉ dừng lại ở việc chỉnh sửa, tăng chất lượng hình ảnh mà với công nghệ xử lý ảnh hiện nay chúng ta có thể giải quyết các bài toán nhận dạng chữ viết, nhận dạng dấu vân tay, nhận dạng khuôn mặt…

Một trong những bài toán được nhiều người quan tâm nhất của lĩnh vực xử lý ảnh hiện nay đó là nhận dạng khuôn mặt (Face Recognition). Như chúng ta đã biết, khuôn mặt đóng vai trò quan trọng trong quá trình giao tiếp giữa người với người, nó mang một lượng thông tin giàu có, chẳng hạn như từ khuôn mặt chúng ta có thể xác định giới tính, tuổi tác, chủng tộc, trạng thái cảm xúc, đặc biệt là xác định mối quan hệ với đối tượng (có quen biết hay không).

Trong đề tài, nhóm chúng em sẽ xây dựng hệ thống nhận diện khuôn mặt có đeo khẩu trang hoặc không đeo khẩu trang để đưa cảnh báo, là tiền đề cở sở để phát triển hệ thống cảnh báo mang khẩu trang đối với mọi người đặt biệt là ở nơi công cộng tập trung đông người như trung tâm thương mại, trường học, bệnh viện,…

Trong đề này nhóm chúng em xây dựng **“Ứng dụng nhận diện khuôn mặt đeo khẩu trang”** với thư viện OpenCV, Keras/Tensorflow, và Deep Learning.

* 1. **Mục đích**

Xây dựng một phần mềm nhận diện khuôn mặt có đang đeo khẩu trang hay là không, từ đó đưa ra cảnh báo trên mang hình giúp cho giám sát màn hình có thể giám sát tình hình đảm bảo an toàn trong giai đoạn dịch bệnh nguy hiểm hiện nay.

Tìm hiểu về thư viện OpenCV khi sử dụng trong môi trường lập trình bằng ngôn ngữ Python.

Tìm hiểu về Keras/Tensorflow.

Tìm hiểu về Deep Learning: cách hoạt động và ứng dụng của Deep Learning trong đề tài này.

* 1. **Phương pháp**

### **1.3.1 Đối tượng**

* Các phương pháp, thuật toán phục vụ cho việc phát hiện và nhận dạng khuôn mặt người trên ảnh.
* Bộ thư viện xử lý ảnh OpenCv
* Bộ thư viện xử lý và nhận dạng Tensorflow
* Phương pháp nhận diện và học sâu Deep Learning

### **1.3.2 Phạm vi nghiên cứu**

* Tập trung tìm hiểu nhận dạng khuôn mặt (Face Recognition)
* Công nghệ theo dõi nhận dạng tập trung ROI (Region Of Interest)
* Xử lí ảnh khuôn mặt từ bộ dữ liệu đầu vào có sẵn và huấn luyện để nhận diện.
* Nhận dạng khuôn mặt với webcam và phát hiện mang khẩu trang hoặc không, đưa ra cảnh báo.
  1. **Cấu trúc đồ án**

Cấu trung đồ án gồm 3 phần chính:

- Chương 1: Giới thiệu tổng quan

- Chương 2: Nghiên cứu tổng quan

- Chương 3: Phân tích thiết kế hệ thống

- Chương 4: Triển khai xây dựng

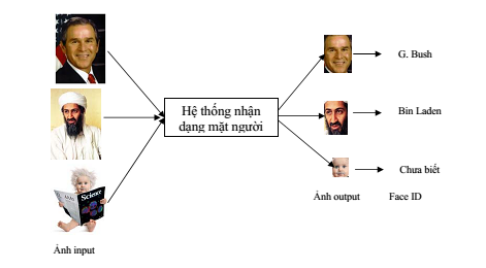
- Chương 5: Kết luận và hướng phát triển

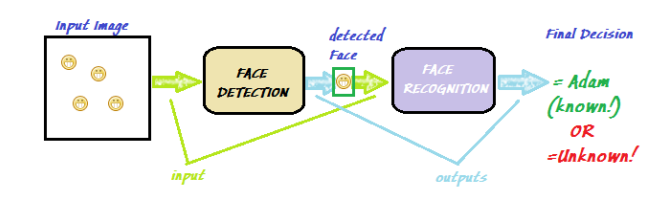
# **Chương 2: Nghiên cứu tổng quan**

## **2.1 Các phương pháp**

### **2.1.1 Bài toán nhận diện khuôn mặt**

Hệ thống nhận dạng khuôn mặt là một hệ thống nhận vào là một ảnh,một đoạn video (một dòng các hình ảnh liên tục) hoặc bằng webcam. Qua xử lý, tính toán hệ thống xác định được vị trí khuôn mặt (nếu có) trong ảnh và xác định là người nào trong số những người mà hệ thống đã được biết (qua quá trình học) hoặc là người lạ.

Trước khi nhận dạng mặt người, chương trình phải chuẩn bị trước một dữ liệu thô. Bao gồm các khuôn mặt của người cụ thể là sinh viên. Tiếp theo là training dữ liệu thô để trích xuất các embedding vector để phục vụ cho việc so sánh sau này. Do đó việc chuẩn bị dữ liệu ban đầu hết sức quan trọng.



**Hình 1-Hệ thống nhận diện khuôn mặt**

### **2.1.2 Tổng quan kiến trúc của một hệ thống nhận dạng khuôn mặt**

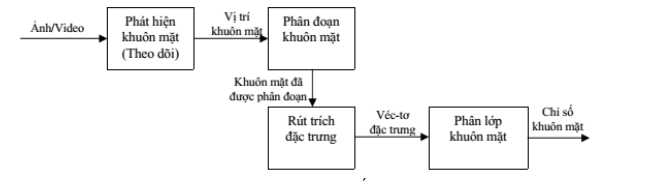
Một hệ thống nhận dạng mặt người thông thường bao gồm bốn bước xử lý sau:

1. Phát hiện khuôn mặt (Face Detection).

2. Phân đoạn khuôn mặt (Face Alignment hay Segmentation).

3. Trích chọn đặc trưng (Feature Extraction).

4. Nhận dạng (Recognition) hay Phân lớp khuôn mặt (Face Clasaification).

****

**Hình 2-Các bước chính trong hệ thống nhận dạng**

### **2.1.3 Xác định khuôn hình khẩu trang trên khuôn mặt**

Khi nhận diện khuôn mặt xác định vị trí và theo dõi cố dịnh trên khuôn.

Lúc này ta sử dụng ROI để có thể xác định một vật thể trên khuôn mặt sau khi ta đeo khẩu trang, các điểm ảnh trên khuôn mặt bị che ví dụ mắt, mũi, miệng,… Hệ thống sẽ xác định đó là khẩu trang.

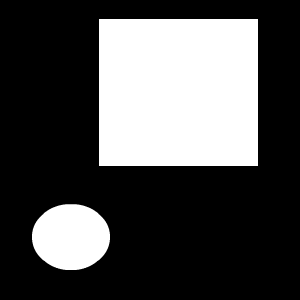
Vùng quan tâm (Region Of Interest, viết tắt ROI hoặc RoI) là vùng ảnh ta sẽ áp dụng các xử lý / biến đổi trên đó và bỏ qua các vùng còn lại.

Vùng quan tâm (ROI) trong xử lý ảnh

* ROI có thể ở bất cứ hình dạng nào, có thể là hình vuông, hình chữ nhật, hình tròn, nhiều vùng nhỏ trên ảnh hoặc thậm chí là các pixel trên ảnh.
* ROI là một KHẨU TRANG mà trên đó: pixel quan tâm có giá trị 1 (màu trắng) và pixel không quan tâm có giá trị zero.
* Vùng không quan tâm trên ảnh thường là BACKGROUND (ảnh nền).
* Áp dụng ROI lên ảnh ta lấy ảnh gốc nhân với khẩu trang ROI.

ROI được mô tả dữ dội quá, hãy cùng **Minh** xem qua ảnh minh họa của ROI nhé:

**ROI.png**



* Màu trắng thể hiện vùng ảnh ta quan tâm, cần sẽ xử lý.
* Màu đen là vùng ta KHÔNG QUAN TÂM, không xử lý hoặc bỏ đi.
* Khẩu trang ROI là ma trận những giá trị 0, 1 (nhị phân), do đó để trực quan hóa thành hình ảnh như hình trên, điểm không cần phải nói ra là ta scale về miền giá trị [0, 255] cho hiển thị!!

## **2.2 Các công nghệ thực hiện và các khái niệm liên quan**

Để giải quyết bài toán nhận diện khuôn mặt và nhận diện đeo khẩu trang, sử dụng thư việc xư lý ảnh OpenCV, thư viện Keras/Tensorflow và sử dụng Deep Learning trong quá trình thực hiện đề tài này.

Trong đồ án này ngôn ngữ lập trình được sử dụng là Python, được viết trên môi trường của Python là IDLE và Visual Studio Code.

### **2.2.1 Giới thiệu ngôn ngữ lập trình Python**

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu.

Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.

Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian, Python dần mở rộng sang mọi hệ điều hành từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các hệ điều hành khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của Python. Ông giữ vai trò chủ chốt trong việc quyết định hướng phát triển của Python.

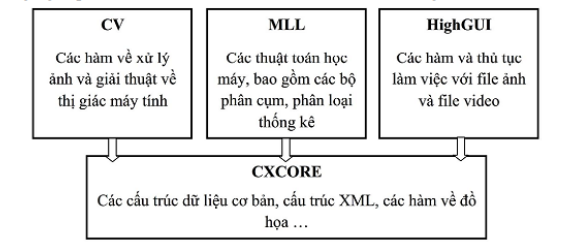
### **2.2.2 Giới thiệu về thư viện OpenCV**

OpenCv (Open Source Computer Vision) là một thư viện mã nguồn mở về thị giác máy với hơn 500 hàm và hơn 2500 các thuật toán đã được tối ưu về XLA, và các vấn đề liên quan tới thị giác máy. OpenCv được thiết kế một cách tối ưu, sử dụng tối đa mạnh của các dòng chip đa lõi… để thực hiện các phép tính toán trong thời gian thực, nghĩa là tốc độ đáp ứng của nó thể đủ nhanh cho các ứng dụng thông thường.

OpenCv là thư viện được thiết kế để chạy trên nhiều nền tảng khác nhau (cross-platform),nghĩa là nó có thể chạy trên hệ điều hành Window, Linux, Mac, iOS… Việc sử dụng thư viện OpenCv tuân theo các quy định về sử dụng phần mềm mã nguồn mở BSD do đó bạn có thể sử dụng thư viện này một cách miễn phí cho các mục đích phi thương mại lẫn thương mại.

Dự án về OpenCv được khởi động từ những năm 1999, đến năm 2000 nó được giới thiệu trong một hội nghị của IEEE về các vấn đề trong thị giác máy và nhận dạng, tuy nhiên bản OpenCV 1.0 mãi tới tận năm 2006 mới chính thức được công bố và năm 2008 bản 1.1 (prerelease) mới được ra đời. Tháng 10 năm 2009, bản OpenCV thế hệ thứ hai ra đời (thường gọi là phiên bản 2.x), phiên bản này có giao diện của C++ (khác với phiên bản trước có giao diện của C) và có nhiều điểm khác biệt so với phiên bản thứ nhất.

Thư viện OpenCV ban đầu được sự hỗ trợ từ Intel, sau đó được hỗ trợ bởi Willow Garage, một phòng thí nghiệm chuyên nghiên cứu về công nghệ robot. Cho đến nay, OpenCV vẫn là thư viện mở, được phát triển bởi nguồn quỹ không lợi nhuận (none-profit foundation) và được sự hưởng ứng rất lớn của cộng đồng.



**Hình 3-Giới thiệu về thư viện OpenCV**

### **2.2.3 Deep Learning**

Trí tuệ nhân tạo đang len lỏi vào trong cuộc sống và ảnh hưởng sâu rộng tới mỗi chúng ta, các cụm từ “Artificial Intelligence”, “Machine Learing”, “Deep Learning” đã không còn quá xa lạ gì. Chúng ta cùng xem hình vẽ để mô tả lại mối quan hệ giữa artificial intelligence, machine learning, và deep learning:

Deep learning đã và đang là một chủ đề AI được bàn luận sôi nổi. Là một phạm trù nhỏ của machine learning, deep learning tập trung giải quyết các vấn đề liên quan đến mạng thần kinh nhân tạo nhằm nâng cấp các công nghệ như nhận diện giọng nói, tầm nhìn máy tính và xử lý ngôn ngữ tự nhiên. Deep learning đang trở thành một trong những lĩnh vực hot nhất trong khoa học máy tính. Chỉ trong vài năm, deep learning đã thúc đẩy tiến bộ trong đa dạng các lĩnh vực như nhận thức sự vật (object perception), dịch tự động (machine translation), nhận diện giọng nói,… – những vấn đề từng rất khó khăn với các nhà nghiên cứu trí tuệ nhân tạo.

Để hiểu hơn về deep learning, hãy nhìn lại một số khái niệm cơ bản về trí tuệ nhân tạo.

Trí tuệ nhân tạo có thể được hiểu đơn giản là được cấu thành từ các lớp xếp chồng lên nhau, trong đó mạng thần kinh nhân tạo nằm ở dưới đáy, machine learning nằm ở tầng tiếp theo và deep learning nằm ở tầng trên cùng.

### **2.2.4** **Thư viện của Deep Learning – Keras/Deep Learning**

Keras được coi là một thư viện ‘high-level’ với phần ‘low-level’ (còn được gọi là backend) có thể là TensorFlow, CNTK, hoặc Theano. Keras có cú pháp đơn giản hơn TensorFlow rất nhiều. Với mục đích giới thiệu về các mô hình nhiều hơn là các sử dụng các thư viện deep learning, tôi sẽ chọn Keras với TensorFlow là ‘backend’.

Những lý do nên sử dụng Keras để bắt đầu:

* Keras ưu tiên trải nghiệm của người lập trình
* Keras đã được sử dụng rộng rãi trong doanh nghiệp và cộng đồng nghiên cứu
* Keras giúp dễ dàng biến các thiết kế thành sản phẩm
* Keras hỗ trợ huấn luyện trên nhiều GPU phân tán
* Keras hỗ trợ đa backend engines và không giới hạn bạn vào một hệ sinh thái

Tensorflow - Với sự bùng nổ của lĩnh vực Trí Tuệ Nhân Tạo – A.I. trong thập kỷ vừa qua, machine learning và deep learning rõ ràng cũng phát triển theo cùng. Và ở thời điểm hiện tại, TensorFlow chính là thư viện mã nguồn mở cho machine learning nổi tiếng nhất thế giới, được phát triển bởi các nhà nghiên cứu từ Google. Việc hỗ trợ mạnh mẽ các phép toán học để tính toán trong machine learning và deep learning đã giúp việc tiếp cận các bài toán trở nên đơn giản, nhanh chóng và tiện lợi hơn nhiều.

Kiến trúc TensorFlow hoạt động được chia thành 3 phần:

– Tiền xử lý dữ liệu

– Dựng model

– Train và ước tính model

Cách TensorFlow hoạt động:

TensorFlow cho phép các lập trình viên tạo ra dataflow graph, cấu trúc mô tả làm thế nào dữ liệu có thể di chuyển qua 1 biểu đồ, hay 1 sê-ri các node đang xử lý. Mỗi node trong đồ thị đại diện 1 operation toán học, và mỗi kết nối hay edge giữa các node là 1 mảng dữ liệu đa chiều, hay còn được gọi là ‘tensor’.

TensorFlow cung cấp tất cả những điều này cho lập trình viên theo phương thức của ngôn ngữ Python. Vì Python khá dễ học và làm việc, ngoài ra còn cung cấp nhiều cách tiện lợi để ta hiểu được làm thế nào các high-level abstractions có thể kết hợp cùng nhau. Node và tensor trong TensorFlow là các đối tượng Python, và các ứng dụng TensorFlow bản thân chúng cũng là các ứng dụng Python.

Các operation toán học thực sự thì không được thi hành bằng Python. Các thư viện biến đổi có sẵn thông qua TensorFlow được viết bằng các binary C++ hiệu suất cao. Python chỉ điều hướng lưu lượng giữa các phần và cung cấp các high-level abstraction lập trình để nối chúng lại với nhau.

TensorFlow 2.0, được ra mắt vào tháng 10 năm 2019, cải tiến framework theo nhiều cách dựa trên phản hồi của người dùng, để dễ dàng và hiệu quả hơn khi làm việc cùng nó (ví dụ: bằng cách sử dụng các Keras API liên quan đơn giản cho việc train model). Train phân tán dễ chạy hơn nhờ vào API mới và sự hỗ trợ cho TensorFlow Lite cho phép triển khai các mô hình trên khá nhiều nền tảng khác nhau. Tuy nhiên, nếu đã viết code trên các phiên bản trước đó của TensorFlow thì bạn phải viết lại, đôi lúc 1 ít, đôi lúc cũng khá đáng kể, để tận dụng tối đa các tính năng mới của TensorFlow 2.0.

## **2.3 Hạn chế, tồn tại của các phương pháp**

Bài toán nhận dạng mặt người là bài toán đã được nghiên cứu từ những năm 70. Tuy nhiên, đây là một bài toán khó nên những nghiên cứu hiện tại vẫn chưa đạt được những kết quả mong muốn. Chính vì thế, vấn đề này vẫn đang được nhiều nhóm trên thế giới quan tâm nghiên cứu. Khó khăn của bài toán nhận dạng mặt người có thể kể đến như sau:

a. Tư thế chụp, góc chụp: Ảnh chụp khuôn mặt có thể thay đổi rất nhiều bởi vì góc chụp giữa camera và khuôn mặt. Chẳng hạn như: chụp thẳng, chụp chéo bên trái 45o hay chụp chéo bên phải 45o , chụp từ trên xuống, chụp từ dưới lên, v.v… Với các tư thế khác nhau, các thành phần trên khuôn mặt như mắt, mũi, miệng có thể bị khuất một phần hoặc thậm chí khuất hết.

b. Sự xuất hiện hoặc thiếu một số thành phần của khuôn mặt: Các đặc trưng như: râu mép, râu hàm, mắt kính, v.v… có thể xuất hiện hoặc không. Vấn đề này làm cho bài toán càng trở nên khó hơn rất nhiều.

c. Sự biểu cảm của khuôn mặt: Biểu cảm của khuôn mặt con người có thể làm ảnh hưởng đáng kể lên các thông số của khuôn mặt. Chẳng hạn, cùng một khuôn mặt một người, nhưng có thể sẽ rất khác khi họ cười hoặc sợ hãi, v.v…

d. Sự che khuất: Khuôn mặt có thể bị che khuất bởi các đối tượng khác hoặc các khuôn mặt khác.

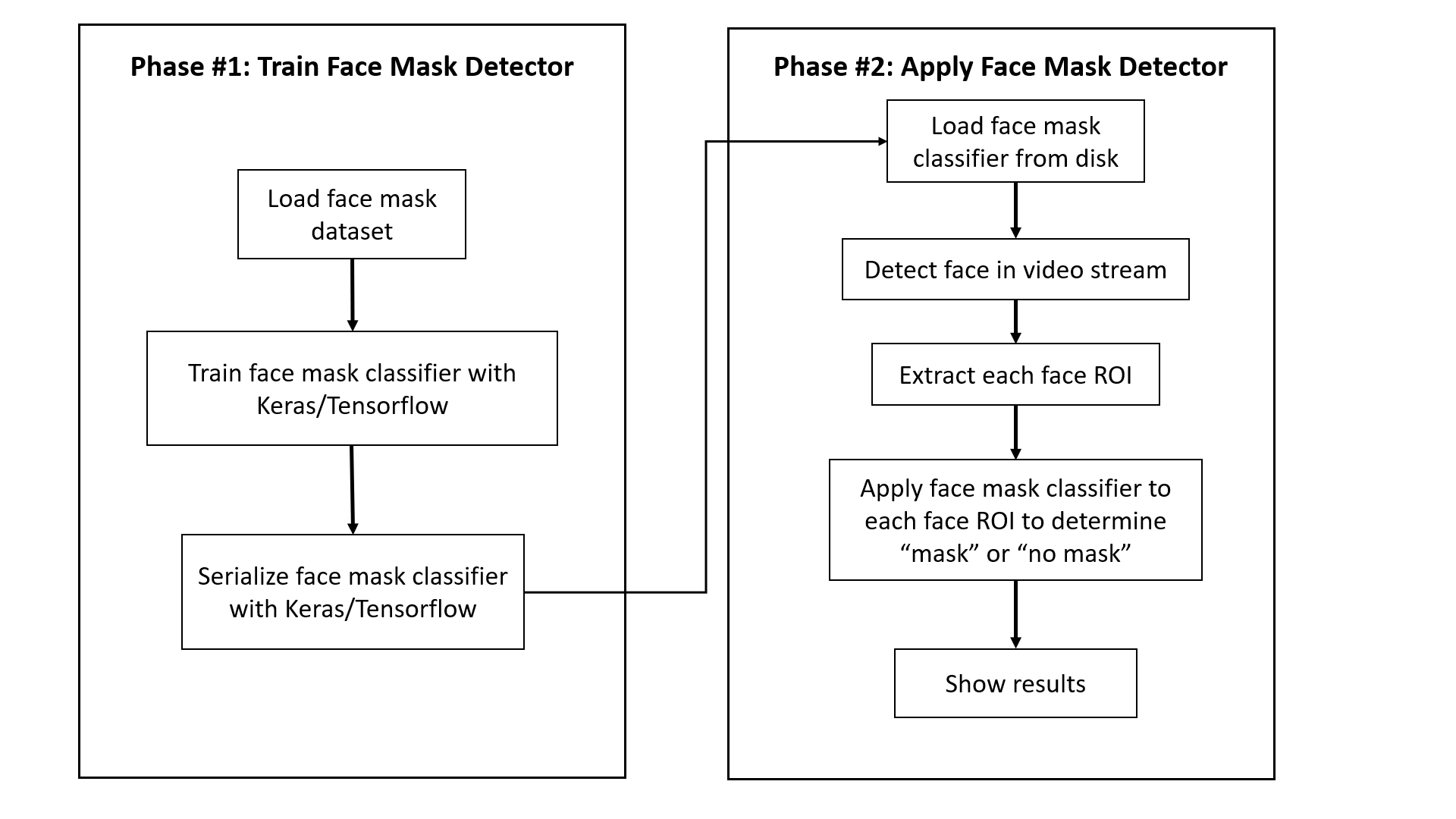
e. Hướng của ảnh (pose variations): Các ảnh khuôn mặt có thể biến đổi rất nhiều với các góc quay khác nhau của trục camera. Chẳng hạn chụp với trục máy ảnh nghiêng làm cho khuôn mặt bị nghiêng so với trục của ảnh.

f. Điều kiện của ảnh: Ảnh được chụp trong các điều kiện khác nhau về: chiếu sáng, về tính chất camera (máy kỹ thuật số, máy hồng ngoại, v.v…), ảnh có chất lượng thấp ảnh hưởng rất nhiều đến chất lượng ảnh khuôn mặt. g. Aging condition: Việc nhận dạng ảnh mặt thay đổi theo thời gian còn là một vấn đề khó khăn, ngay cả đối với khả năng nhận dạng của con người.

# **Chương 3: Phân tích thiết kế hệ thống**

## **3.1 Mô hình tổng quan của hệ thống nghiên cứu**

### **3.1.1 Luồng hoạt động của hệ thống**



**Hình 4-Sơ đồ luồng hoạt động của hệ thống**

Để có thể tùy chỉnh được việc huấn luyện trong nhận diện khuôn mặt đeo khẩu trang của hệ thống ta chia ra gồm 2 giai đoạn (phase) riêng biệt, mỗi giai đoạn có các bước nhiệm vụ riêng (Như ở ***Hình 3.1.1*** bên trên)

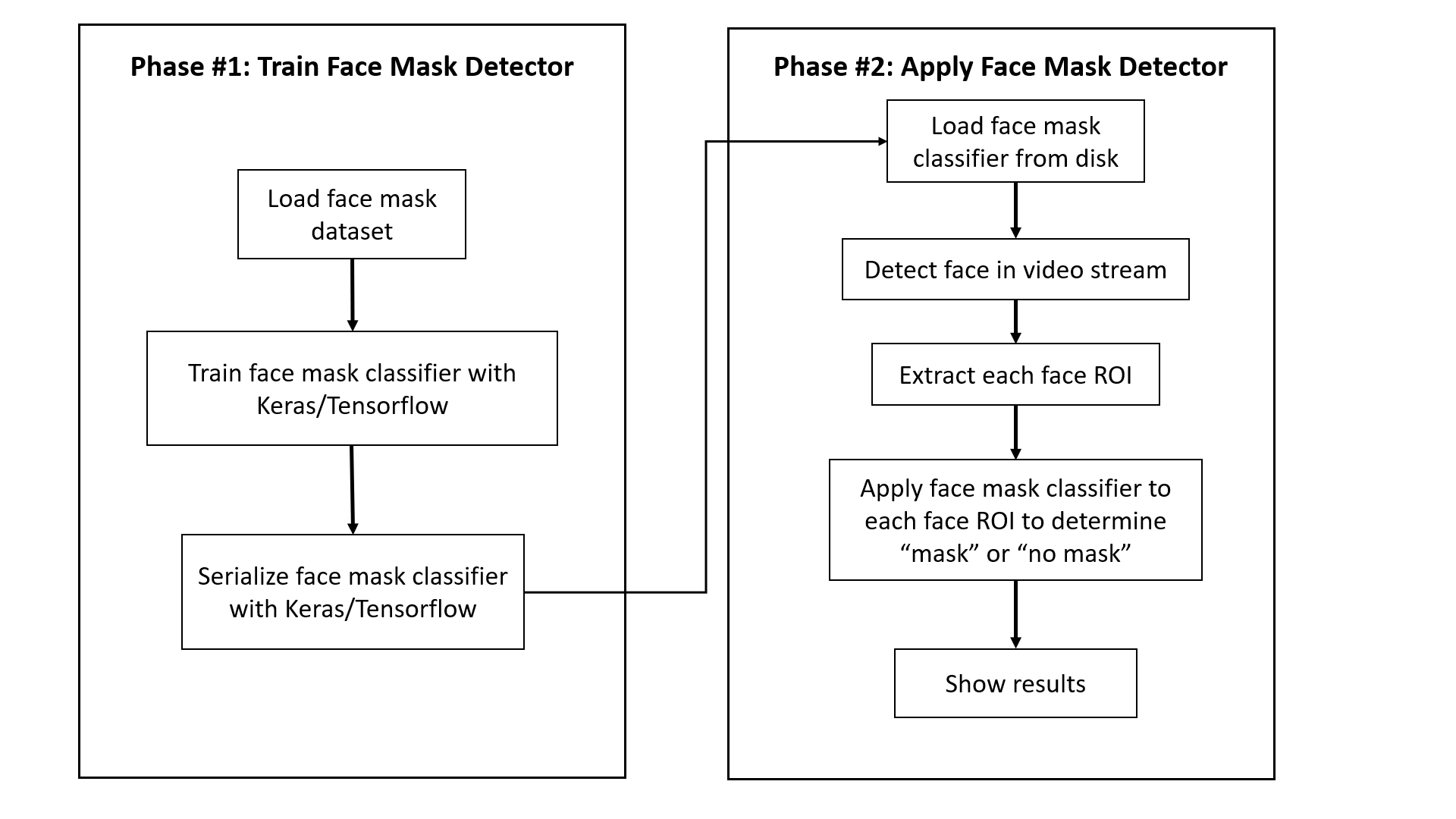
Ta có 2 giai đoạn chính như trên sơ đồ:

Giai đoạn 1: Huấn luyện dữ liệu cho hệ thống (Train Face Mask Detector)

Giai đoạn 2: Triển khai nhận diện và phân tích (Apply Face Mask Detector)

## **3.2 Phân tích chi tiết luồng hoạt động của hệ thống**

### **3.2.1 Giai đoạn huấn luyện dữ liệu cho hệ thống**



**Hình 5-Sơ đồ luồng hoạt động của giai đoạn Train dữ liệu**

Chúng ta sẽ tập trung vào việc tải tập dữ liệu phát hiện khẩu trang của mình từ bộ dữ liệu, đào tạo một mô hình (sử dụng Keras / TensorFlow) trên tập dữ liệu này và sau đó tuần tự hóa trình phát hiện khẩu trang vào bộ dữ liệu

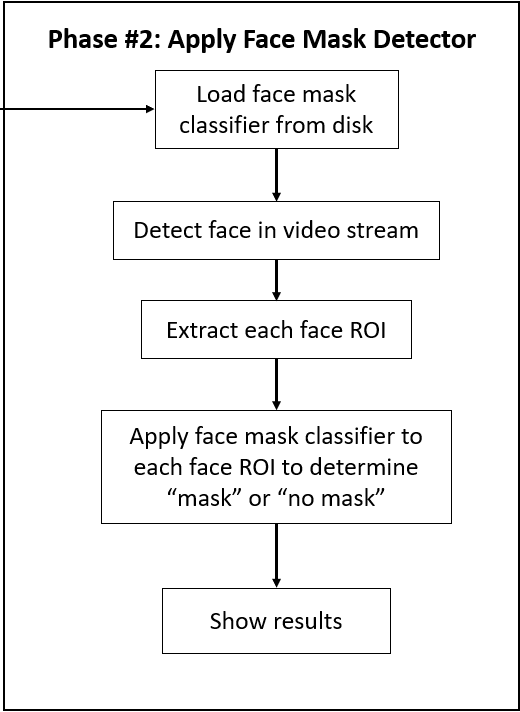
Phân tích luồng hoạt động của giai đoạn Train dữ liệu qua thứ tự các bước:

- Load face mask dataset: bước đầu tiên, hệ thống sẽ bắt đầu được load dữ liệu để dữ liệu được đưa vào, dữ liệu được lấy từ tập dữ liệu (dataset).

- Train face mask classifier with Keras/Tensorflow: tiến hành huấn luyện phân loại với Keras/Tensoflow, việc huấn luyện lúc này có nhiệm vụ phân loại ra là khuôn mặt có đeo khẩu trang và không đao khẩu trang để khi việc phân tích sẽ đưa hai kết quả cho việc nhận diện.

- Serialize face mask classifier to disk: tuần tự hóa việc phân loại với Keras/Tensorflow, lúc này hệ thống sẽ có nhiệm vụ liên tục thực hiện việc tuần tự train dữ liệu đầu vào để duy trì việc huấn luyện dữ liệu.

### **3.2.2 Giai đoạn áp dụng và triển khai phân tích nhận diện của hệ thống**

****

**Hình 6-Sơ đồ luồng hoạt động phân tích và nhận diện của hệ thống**

Sau khi trình phát hiện khẩu trang được huấn luyện (train), chúng ta có thể chuyển sang tải trình phát hiện khẩu trang, thực hiện phát hiện khuôn mặt và sau đó phân loại từng khuôn mặt thành có đeo khẩu trang và không đeo khẩu trang.

Phân tích luồng hoạt động của giai đoạn áp dụng vào phân tích và nhận diện của hệ thống:

- Load face mask classifier from disk: Trước khi phân tích, chúng ta cho hệ thống load qua toàn bộ ảnh nhận dạng trong bộ nhớ cho chúng học và sau đó mới tiền hành sang cho việc nhận diện phân tích.

- Detect face in video stream: sau khi được load toàn bộ ảnh từ trong ảnh ta chuyện sang nhận diện khuôn mặt thông qua video stream sử dụng webcam từ laptop.

- Extract each face ROI: sau khi nhận dạng được khuôn mặt từ video stream ta tiến hành áp dụng ROI vào việc trích xuất một khuôn mặt tập trung nhất định.

- Apply face mask classifier to each face ROI to determine "mask" or "no mask": Áp dụng công cụ phân loại khẩu trang cho ROI của từng khuôn mặt để xác định "có khẩu trang" hoặc "không có khẩu trang".

- Show results: hiển thị kết quả sau khi đã tiến hành phân tích nhận diện.

## **3.3 Phân tích tập dữ liệu (dataset)**

### **3.3.1 Xây dựng tập dữ liệu**

Để xây dựng tập dữ liệu phục vụ cho việc huấn luyện (train) dữ liệu.

- Tập ảnh chân dung với khuôn mặt không đeo khẩu trang (*without mask*): chúng ta tạo ra một kho ảnh chụp chân dung của nhiều người khác nhau, sau đó training tập này với định danhh là không đeo khẩu trang

- Tập ảnh chân dung với khuôn mặt đeo khẩu trang (*with mask*): sử dụng tập ảnh chân dung không đeo khẩu trang, ta tiến hành tạo tâp ảnh nhân tạo bằng cách dùng tập lệnh Python và sử dụng OpenCV cho việc ghép hình khẩu trang vào các khuôn mặt không đeo khẩu trang. Từ đó ta có tập dữ liệu ảnh nhân tạo nhưng vẫn áp dụng thực tế được.

Các mốc trên khuôn mặt cho phép chúng ta suy ra và xác định được các cấu trúc trên khuôn mặt, bao gồm: Đôi mắt, Lông mày, Mũi, Mặt, Khuôn hàm mặt.

Để sử dụng các mốc trên khuôn mặt để xây dựng tập dữ liệu về các khuôn mặt đeo khẩu trang, trước tiên chúng ta cần bắt đầu bằng hình ảnh của một người không đeo khẩu trang:



**Hình 7-Bức ảnh một người không đeo khẩu trang**

Để xây dựng bộ dữ liệu về khẩu trang phòng đại dịch COVID-19 / Coronavirus, trước tiên chúng ta sẽ bắt đầu với một bức ảnh của một người không đeo khẩu trang.

Từ đó, chúng ta áp dụng tính năng phát hiện khuôn mặt để tính toán vị trí hộp giới hạn của khuôn mặt trong hình ảnh:



**Hình 8-Nhận diện khuôn mặt trên bức ảnh nhờ Deep Learning**

Hình 3.3.2 - Áp dụng tính năng nhận diện khuôn mặt. Ở đây, chúng ta đã sử dụng phương pháp học sâu (Deep Learning) để thực hiện nhận diện khuôn mặt với OpenCV.

.



**Hình 9-Trích xuất ROI của khuôn mặt với OpenCV và NumPy cắt**

Hình 3.3.3 - Khi chúng ta biết vị trí của khuôn mặt trong hình ảnh, chúng ta có thể trích xuất Vùng ưa thích (ROI) của khuôn mặt

Và từ đó, chúng ta áp dụng các điểm mốc trên khuôn mặt, cho phép chúng ta xác định vị trí của mắt, mũi, miệng, v.v.:



**Hình 10-Đánh dấu các điểm mốc trên khuôn mặt**

Sau đó, chúng ta phát hiện các điểm mốc trên khuôn mặt bằng cách sử dụng dlib để chúng ta biết vị trí đặt khẩu trang trên khuôn mặt.

Tiếp theo, chúng ta cần một hình ảnh của một khẩu trang (với nền trong suốt) như hình bên dưới:



**Hình 11-Hình khẩu trang**

Hình 3.3.5- Ví dụ về khẩu trang ngăn chặn lây lan COVID-19 Coronavirus qua đường hô hấp. Khẩu trang này sẽ tự động được phủ lên ROI của khuôn mặt ban đầu vì chúng ta biết các vị trí mốc khuôn mặt.

Khẩu trang này sẽ được tự động áp dụng cho khuôn mặt bằng cách sử dụng các điểm mốc trên khuôn mặt (cụ thể là các điểm dọc theo cằm và mũi) để tính toán vị trí khẩu trang sẽ được đặt.



**Hình 12-Hình ảnh khẩu trang được đặt lên khuôn mặt**

Trong hình này, khẩu trang được đặt trên khuôn mặt của người trong khung hình ban đầu. Nhìn thoáng qua, khó có thể biết được rằng khẩu trang COVID-19 đã được áp dụng với thị giác máy tính thông qua các mốc khuôn mặt OpenCV và dlib.

Sau đó, chúng ta có thể lặp lại quá trình này cho tất cả các hình ảnh đầu vào của mình, từ đó tạo bộ dữ liệu khẩu trang nhân tạo của chúng ta:



**Hình 13-Bộ dữ liệu ảnh đeo khẩu trang nhân tạo**

Hình 3.3.7- Hình ảnh khẩu trang COVID-19 nhân tạo được hiển thị. Bộ này sẽ là một phần của bộ dữ liệu "có khẩu trang" / "không có khẩu trang" của chúng ta để phát hiện khẩu trang COVID-19 với thị giác máy tính và học sâu bằng Python, OpenCV và TensorFlow / Keras.

# **Chương 4: Triển khai xây dựng**

## **4.1 Cấu trúc dự án**

Cấu trúc thư mục chính của dự án:

├── **dataset**

│ ├── with\_mask [690 entries]

│ └── without\_mask [686 entries]

├── **examples**

│ ├── example\_01.png

│ ├── example\_02.png

│ └── example\_03.png

├── **face\_detector**

│ ├── deploy.prototxt

│ └── res10\_300x300\_ssd\_iter\_140000.caffemodel

├── **detect\_mask\_image.py**

├── **detect\_mask\_video.py**

├── **mask\_detector.model**

├── **plot.png**

└── **train\_mask\_detector.py**

Tập thư mục **dataset** chứa dữ liệu được mô tả trong phần “Tập dữ liệu phát hiện khẩu trang COVID-19 của chúng ta”.

Ba hình ảnh **examples** được cung cấp để bạn có thể kiểm tra trình phát hiện khẩu trang hình ảnh tĩnh.

**- train\_mask\_detector.py** : Chấp nhận tập dữ liệu đầu vào của chúng ta và tinh chỉnh MobileNetV2 dựa trên nó để tạo **mask\_detector.model.** Một **plot.png** lịch sử huấn luyện cũng được tạo ra.

- **detect\_mask\_image.py :** Thực hiện phát hiện khẩu trang trong hình ảnh tĩnh.

**- detect\_mask\_video.py :** Sử dụng webcam của bạn, tập lệnh này áp dụng tính năng phát hiện khẩu trang cho mọi khung hình trong luồng.

## **4.2 Triển khai kịch bản huấn luyện (training) hệ thống nhận diện khẩu trang**

Chúng ta tiến hành xem xét tập dữ liệu, tìm hiểu cách để có thể sử dụng Keras và TensorFlow để đào tạo một bộ phân loại nhằm tự động phát hiện một người có đeo khẩu trang hay không.

Để hoàn thành nhiệm vụ này, chúng ta sẽ tinh chỉnh kiến ​​trúc MobileNet V2, một kiến ​​trúc hiệu quả cao có thể được áp dụng cho các thiết bị nhúng có dung lượng tính toán hạn chế (ví dụ: Raspberry Pi, Google Coral, NVIDIA Jetson Nano, v.v.).

Việc triển khai nhận diện khẩu trang của chúng ta cho các thiết bị nhúng có thể giảm chi phí sản xuất các hệ thống phát hiện khẩu trang như vậy, do đó chính là lí do chúng ta chọn sử dụng kiến ​​trúc này.

Tiến hành triển khai hệ thống:

- Chèn các thư viện có liên quan Deep Learning vào file train\_mask\_detector.py



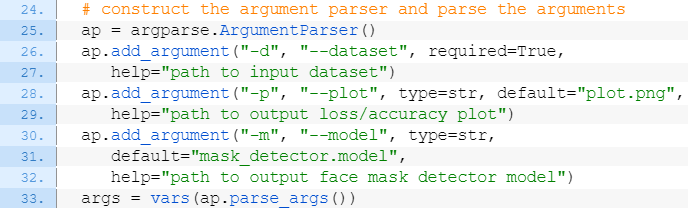
Tensorflow/Keras cho phép:

* Data augmentation
* Loading the MobilNetV2 classifier (we will fine-tune this model with pre-trained ImageNet weights)
* Building a new fully-connected (FC) head
* Pre-processing
* Loading image data

Chúng ta sử dụng scikit-learning (sklearn) để phân loại hai nhãn lớp, phân đoạn tập dữ liệu của chúng ta và in báo cáo phân loại.

Việc triển khai các đường dẫn imutils của ta sẽ giúp chúng ta tìm và liệt kê các hình ảnh trong tập dữ liệu của mình. Và chúng ta sử dụng matplotlib để vẽ các đường cong đào tạo của chúng ta.

Phân tích cú pháp một số đối số dòng lệnh cần thiết để khởi chạy tập lệnh của chúng ta từ terminal:



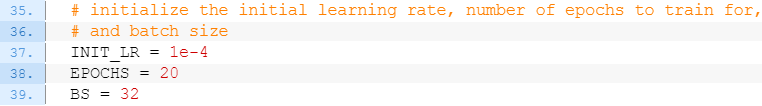
Các đối số dòng lệnh bao gồm:

--dataset : Đường dẫn đến tập dữ liệu đầu vào của khuôn mặt và khuôn mặt có khẩu trang.

--plot: Đường dẫn đến cốt truyện lịch sử đào tạo đầu ra của bạn, sẽ được tạo bằng cách sử dụng matplotlib.

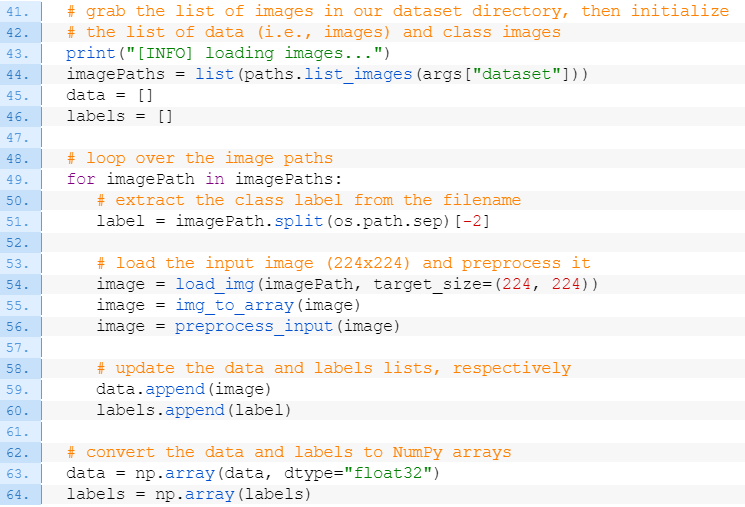
--model: Đường dẫn đến mô hình phân loại khẩu trang được tuần tự hóa kết quả.

Xác định các deep learning hyperparameters ở một nơi:



Chỉ định các hằng số siêu tham số bao gồm tốc độ học ban đầu, số kỷ nguyên đào tạo và kích thước lô. Sau đó, áp dụng lịch trình giảm tốc độ học tập, đặt tên biến tốc độ học tập là INIT\_LR.

Tại thời điểm này, chúng ta đã sẵn sàng tải và xử lý trước dữ liệu đào tạo của mình:



Trong khối lệnh, ta có :

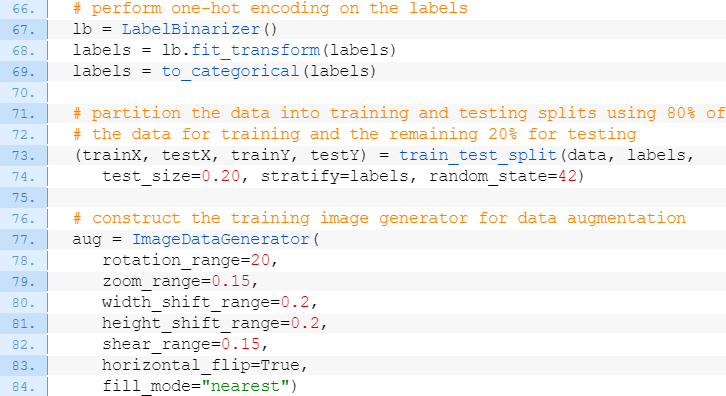
- Nắm lấy tất cả các imagePaths trong tập dữ liệu (dòng 44)

- Lặp qua các đường dẫn hình ảnh và tải + tiền xử lý hình ảnh (Dòng 49-60). Các bước tiền xử lý bao gồm thay đổi kích thước thành 224 × 224 pixel, chuyển đổi sang định dạng mảng và chia tỷ lệ cường độ pixel trong hình ảnh đầu vào thành phạm vi [-1, 1] (thông qua chức năng tiện lợi preprocess\_input)

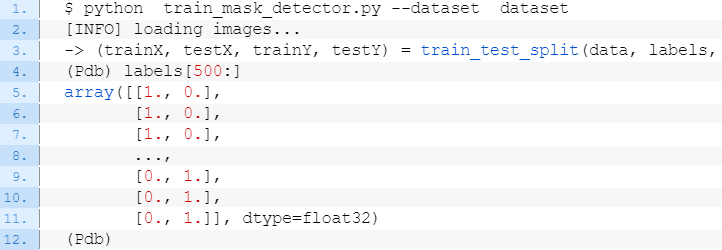
- Thêm hình ảnh được xử lý trước và nhãn được liên kết vào danh sách dữ liệu và nhãn, tương ứng (Dòng 59 và 60)

- Đảm bảo dữ liệu đào tạo của chúng ta ở định dạng mảng NumPy (Dòng 63 và 64)

Tiếp theo, chúng ta sẽ mã hóa nhãn, phân vùng tập dữ liệu và chuẩn bị cho việc tăng dữ liệu:



Các dòng 67-69 một nóng mã hóa các nhãn lớp của chúng ta, có nghĩa là dữ liệu của chúng ta sẽ ở định dạng sau:

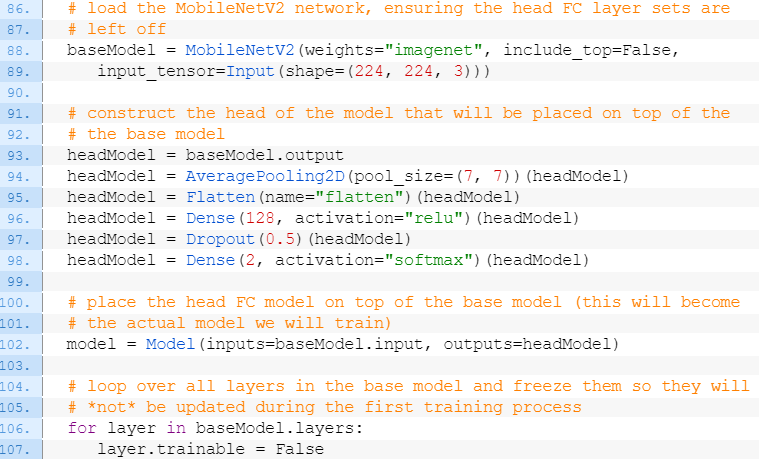


Như bạn có thể thấy, mỗi phần tử của mảng nhãn của chúng ta bao gồm một mảng trong đó chỉ có một chỉ mục là "hot" (tức là 1).

Sử dụng phương pháp tiện lợi của scikit-learning, Dòng 73 và 74 phân đoạn dữ liệu của chúng ta thành 80% đào tạo và 20% còn lại để kiểm tra.

Trong quá trình đào tạo, chúng ta sẽ áp dụng các đột biến nhanh chóng cho hình ảnh của chúng ta nhằm nỗ lực cải thiện khả năng tổng quát hóa. Điều này được gọi là tăng dữ liệu, trong đó các tham số xoay ngẫu nhiên, thu phóng, cắt, dịch chuyển và lật được thiết lập trên các Dòng 77-84.

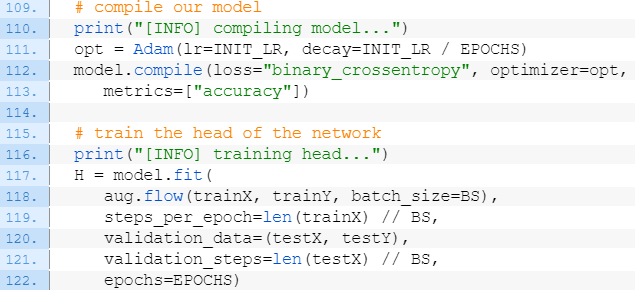
Chúng ta sẽ sử dụng đối tượng aug vào thời gian đào tạo. Nhưng trước tiên, chúng ta cần chuẩn bị MobileNetV2 để tinh chỉnh:



Tinh chỉnh thiết lập là một quy trình gồm ba bước:

* Tải MobileNet bằng các trọng số ImageNet đã được đào tạo trước, bỏ đầu mạng (Dòng 88 và 89)
* Tạo đầu FC mới và gắn nó vào đế thay cho đầu cũ ( Dòng 93-102)
* Cố định các lớp cơ sở của mạng (Dòng 106 và 107). Trọng lượng của các lớp cơ sở này sẽ không được cập nhật trong quá trình nhân giống ngược, trong khi trọng lượng của lớp đầu sẽ được điều chỉnh.

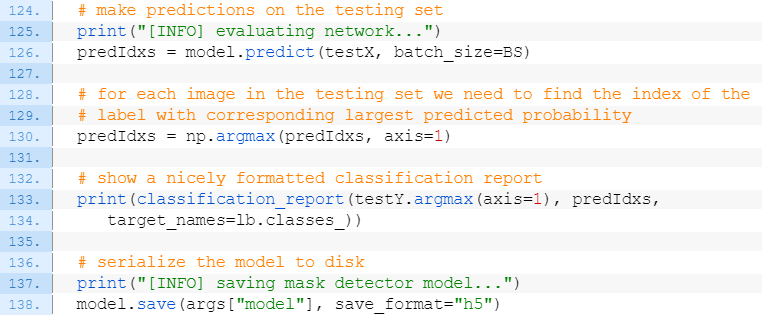
Với dữ liệu của chúng ta đã chuẩn bị và kiến ​​trúc mô hình sẵn sàng để tinh chỉnh, hiện chúng ta đã sẵn sàng biên dịch và đào tạo mạng phát hiện khẩu trang của mình:



Dòng 111-113 biên dịch mô hình của chúng ta với trình tối ưu hóa Adam, lịch trình phân rã tốc độ học tập và entropy chéo nhị phân. Nếu bạn đang xây dựng từ tập lệnh đào tạo này với> 2 lớp, hãy đảm bảo sử dụng entropy chéo phân loại.

Đào tạo về khẩu trang được triển khai qua Đường 117-122. Lưu ý cách đối tượng tăng dữ liệu của chúng ta (aug) sẽ cung cấp các lô dữ liệu hình ảnh bị đột biến.

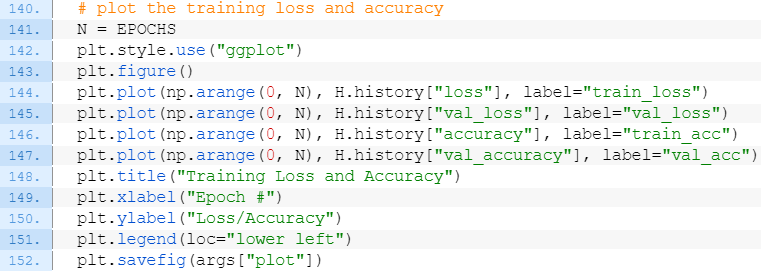
Sau khi đào tạo xong, chúng ta sẽ đánh giá mô hình kết quả trên bộ thử nghiệm:



Tại đây, Dòng 126-130 đưa ra dự đoán trên tập kiểm tra, lấy các chỉ số nhãn của lớp có xác suất cao nhất. Sau đó, chúng ta in một báo cáo phân loại trong thiết bị đầu cuối để kiểm tra.

Dòng 138 sắp xếp tuần tự mô hình phân loại khẩu trang của chúng ta vào đĩa.

Bước cuối cùng của chúng ta là vẽ các đường cong chính xác và mất mát của chúng ta:

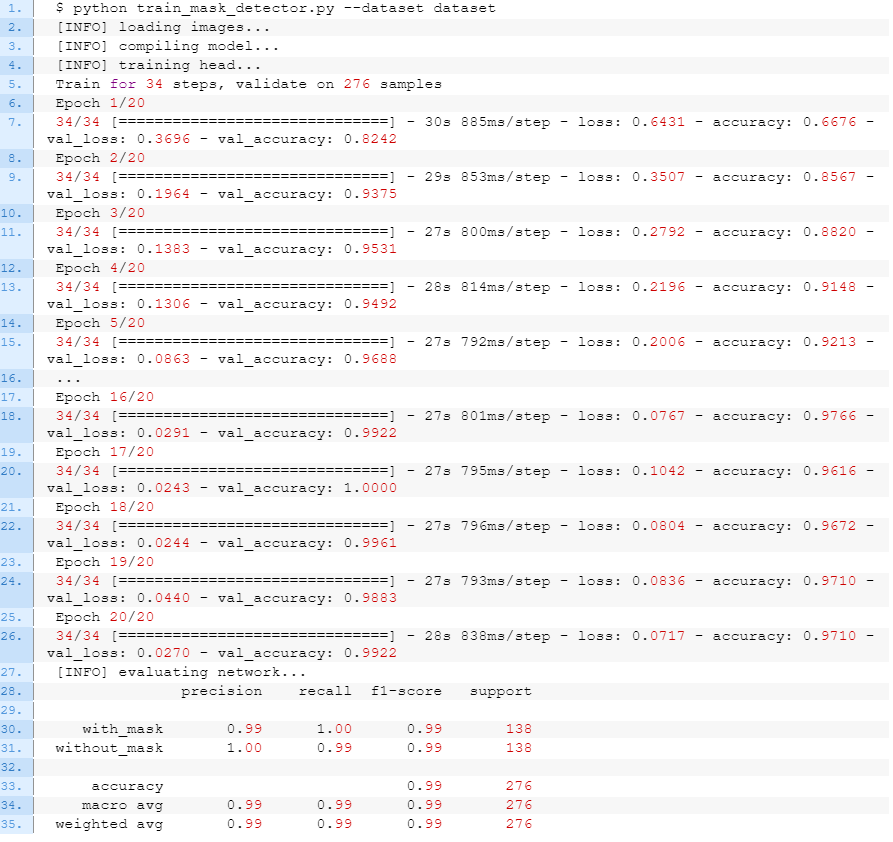


Khi âm mưu của chúng ta đã sẵn sàng, Dòng 152 lưu hình vào đĩa bằng cách sử dụng đường dẫn tệp --plot.

## **4.3 Tiến hành huấn luyện hệ thống nhận diện khẩu trang với Keras/Tensorflow**

Huấn luyện hệ thống nhận diện khẩu trang với Keras/Tensorflow và Deep Learning

Sử dụng trình Terminal để thực thi câu lệnh Train



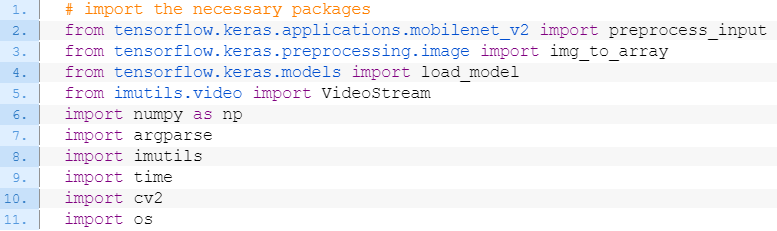
Như bạn có thể thấy, chúng ta đang đạt được độ chính xác ~ 99% trên bộ thử nghiệm của mình.

Với những kết quả này, chúng ta hy vọng rằng mô hình của chúng ta sẽ tổng quát hóa tốt cho các hình ảnh bên ngoài tập huấn luyện và thử nghiệm của chúng ta.

## **4.4 Triển khai nhận diện khẩu trang với video stream qua webcam**

Ta tiến hành xây dựng hệ thống nhận diện chạy với thời gian thực

Sử dụng file detect\_mask\_video.py để xây dựng cấu trúc tập lệnh nhận diện, ta tiến hành import các thư viện cần thiết như bên dưới

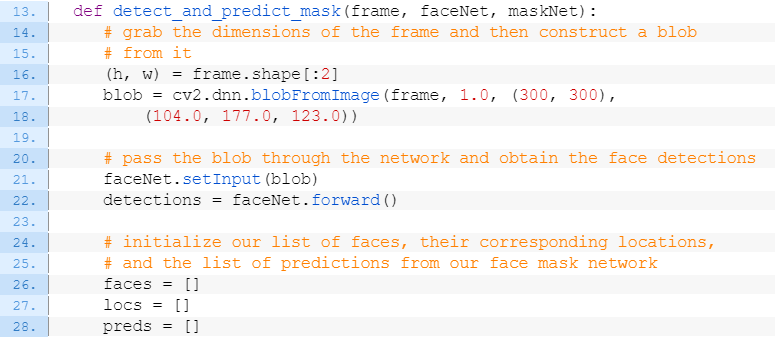


Thuật toán cho tập lệnh này giống nhau, nhưng nó được ghép lại với nhau theo cách để cho phép xử lý mọi khung hình trong luồng webcam của bạn.

Do đó, sự khác biệt duy nhất khi nói đến việc nhập là chúng ta cần một lớp VideoStream và thời gian. Cả hai điều này sẽ giúp chúng ta làm việc với luồng. Chúng ta cũng sẽ tận dụng imutils cho phương pháp thay đổi kích thước nhận biết khía cạnh của nó.

Chức năng này phát hiện khuôn mặt và sau đó áp dụng công cụ phân loại khẩu trang của chúng ta cho ROI của từng khuôn mặt. Một hàm như vậy hợp nhất mã của chúng ta - nó thậm chí có thể được chuyển sang một tệp Python riêng biệt.

Logic dự đoán khẩu trang / phát hiện khuôn mặt của chúng ta cho tập lệnh này nằm trong hàm detect\_and\_predict\_mask :

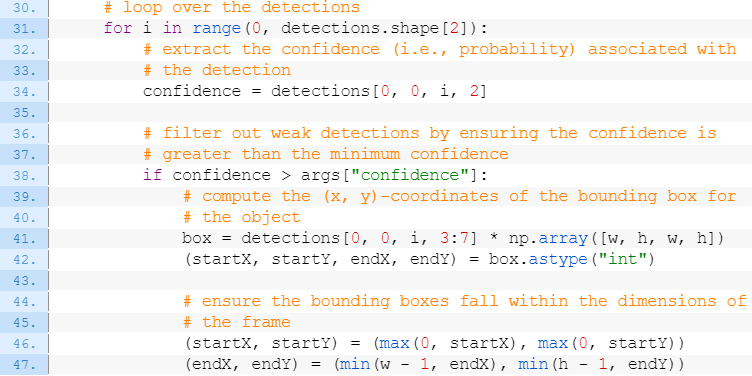


Hàm detector\_and\_posystem\_mask của chúng ta chấp nhận ba tham số:

* frame: Một khung từ khuôn mặt luồng của chúng ta
* faceNet: Mô hình được sử dụng để phát hiện vị trí trong khuôn mặt hình ảnh là khẩu trang
* maskNet: Mô hình phân loại khẩu trang COVID-19 của chúng ta

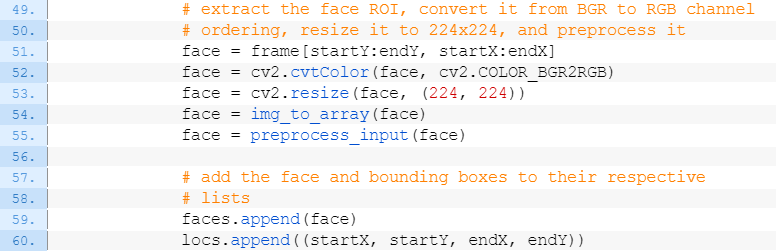
Bên trong, chúng ta xây dựng một đốm màu, phát hiện các khuôn mặt và khởi tạo danh sách, hai trong số đó hàm được đặt để trả về. Những danh sách này bao gồm khuôn mặt của chúng ta (tức là ROI), loc (vị trí khuôn mặt) và preds (danh sách dự đoán có khẩu trang / không có khẩu trang).

Từ đây, chúng ta sẽ lặp lại các bước phát hiện khuôn mặt:



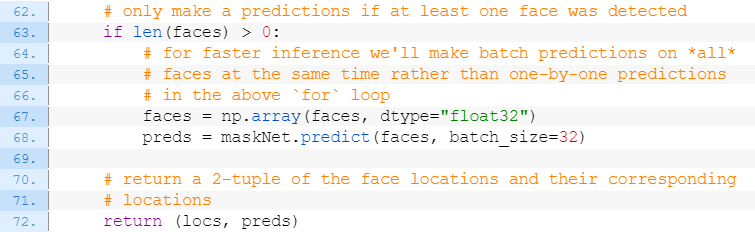
Bên trong vòng lặp, chúng ta lọc ra các phát hiện yếu (Dòng 34-38) và trích xuất các hộp giới hạn trong khi đảm bảo tọa độ hộp giới hạn không nằm ngoài giới hạn của hình ảnh (Dòng 41-47).

Tiếp theo, chúng ta sẽ thêm ROI của khuôn mặt vào hai danh sách tương ứng của chúng ta:



Sau khi trích xuất ROI của khuôn mặt và xử lý trước (Dòng 51-56), chúng ta nối ROI của khuôn mặt và các hộp giới hạn vào danh sách tương ứng của chúng.

Bây giờ chúng ta đã sẵn sàng để kiểm tra khuôn mặt của chúng ta thông qua dự đoán khẩu trang của chúng ta:

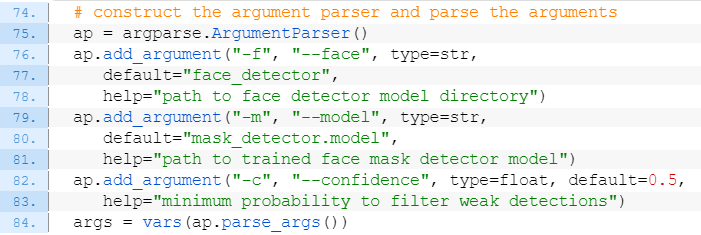


Logic ở đây được xây dựng cho tốc độ. Đầu tiên, chúng ta đảm bảo ít nhất một khuôn mặt đã được phát hiện (Dòng 63) - nếu không, chúng ta sẽ trả về các preds trống.

Chúng ta đang thực hiện suy luận trên toàn bộ loạt mặt trong khung để đường dẫn của chúng ta nhanh hơn (Dòng 68). Sẽ không hợp lý nếu bạn viết một vòng lặp khác để đưa ra dự đoán trên từng mặt riêng lẻ do chi phí cao (đặc biệt nếu bạn đang sử dụng GPU yêu cầu nhiều giao tiếp trên xe buýt hệ thống của bạn). Sẽ hiệu quả hơn nếu thực hiện các dự đoán hàng loạt.

Dòng 72 trả về các vị trí hộp giới hạn khuôn mặt của chúng ta và các dự đoán khẩu trang / không khẩu trang tương ứng cho người gọi.

Tiếp theo, chúng ta sẽ xác định các đối số dòng lệnh của chúng ta:



Các đối số dòng lệnh của chúng ta bao gồm:

--face: Đường dẫn đến thư mục dò tìm khuôn mặt

--model: Đường dẫn đến trình phân loại khẩu trang được đào tạo của chúng ta

--confidence: Ngưỡng xác suất tối thiểu để lọc các phát hiện khuôn mặt yếu

Với việc nhập, chức năng tiện lợi và chuỗi dòng lệnh của chúng ta đã sẵn sàng hoạt động, chúng ta chỉ có một số khởi tạo để xử lý trước khi chúng ta lặp qua các khung:

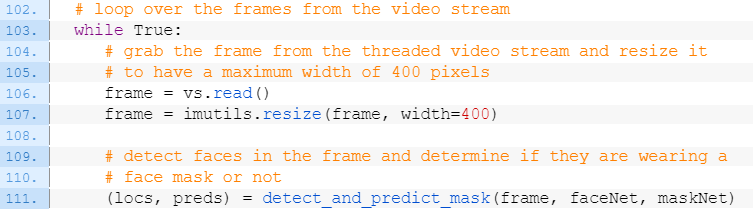
Ở đây chúng ta đã khởi tạo:

- Nhận diện khuôn mặt

- Nhận diện khẩu trang

- Webcam video stream

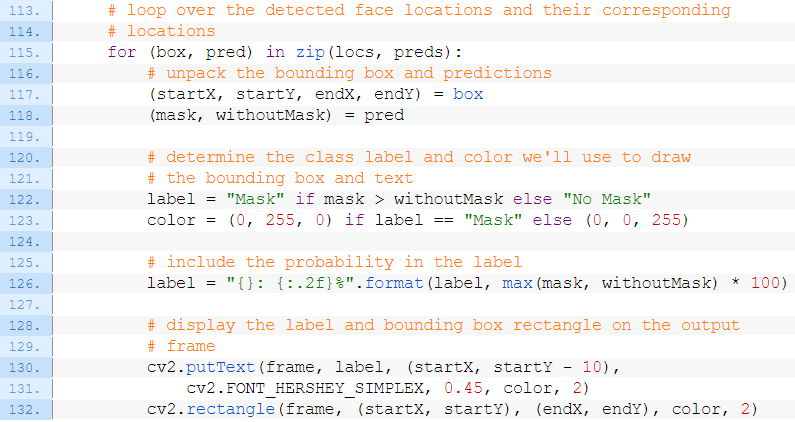
Tiếp tục lặp lại các khung trong luồng:



Chúng ta bắt đầu lặp qua các khung trên Dòng 103. Bên trong, chúng ta lấy một khung từ luồng và thay đổi kích thước của nó (Dòng 106 và 107).

Từ đó, chúng ta đưa tiện ích tiện lợi của mình vào sử dụng; Đường dây 111 phát hiện và dự đoán xem mọi người có đeo khẩu trang hay không.

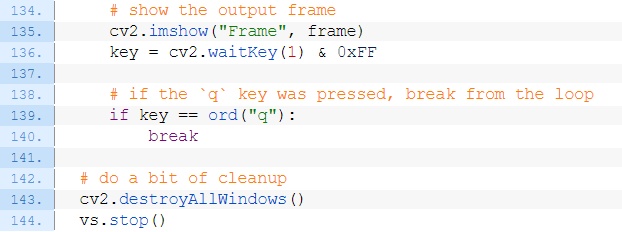
Xử lý hậu kỳ (tức là chú thích) kết quả phát hiện khẩu trang COVID-19:



Bên trong vòng lặp của chúng ta đối với các kết quả dự đoán (bắt đầu từ Dòng 115), chúng ta:

* Mở hộp giới hạn khuôn mặt và dự đoán khẩu trang / không khẩu trang (Dòng 117 và 118)
* Xác định nhãn và màu sắc (Dòng 122-126)
* Chú thích nhãn và giới hạn khuôn mặt (Dòng 130-132)

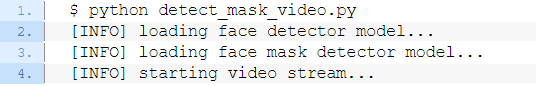
Cuối cùng, chúng ta hiển thị kết quả và thực hiện dọn dẹp:

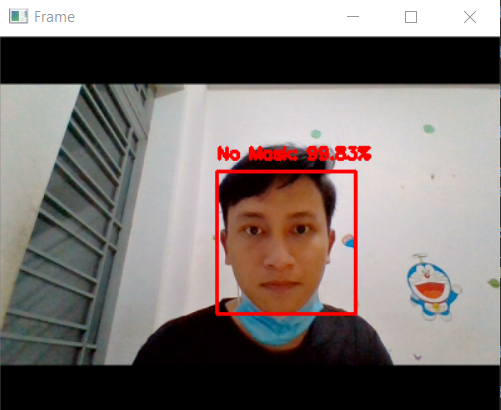


Sau khi khung được hiển thị, chúng ta chụp các phím bấm. Nếu người dùng nhấn q (thoát), chúng ta thoát ra khỏi vòng lặp và thực hiện công việc dọn phòng.

Nhận diện khẩu trang với OpenCV trong thời gian thực:

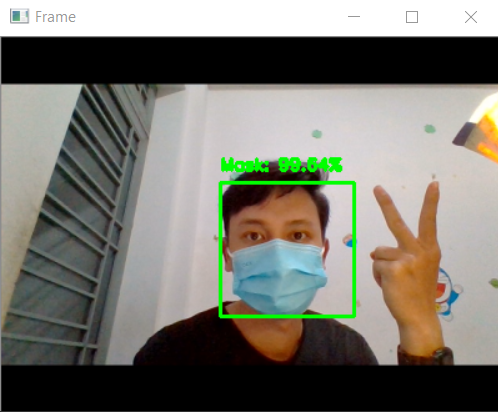
Ta có thể khởi chạy trình phát hiện khẩu trang trong các luồng video thời gian thực bằng lệnh sau:





**Hình 14-Kết quả nhận diện khuôn mặt không mang khẩu trang**

Hình 4.4 – Kết quả nhận diện khuôn mặt khi không đeo khẩu trang đã được hiện ra sau khi phân tích.



**Hình 15-Kết quả nhận diện khuôn mặt có đeo khẩu trang**

Hình 4.4.1 – Kết quả nhận diện khuôn mặt khi đeo khẩu trang đã được hiện ra sau khi phân tích.

Ở đây, Ta có thể thấy rằng máy dò khẩu trang của chúng ta có khả năng chạy trong thời gian thực (và cũng chính xác trong dự đoán của nó).

# **Chương 5: Kết luận và hướng phát triển**

## **5.1 Kết luận**

Sau thời gian khoảng 1 tháng vừa học vừa triển khai xây dựng thì cuối cùng cơ bản hệ thống cũng hoàn thành.

Hệ thống có thể load và training dữ liệu từ tập dữ liệu đã tạo ra gồm các ảnh người có đeo khẩu trang và không đeo khẩu trang.

Từ đó có thể phân tích và nhận diện phân biệt được khuôn mặt có đeo khẩu trang hoặc không. Ngoài ra nhận diện theo thời gian thực khi người dùng sử dụng webcam stream.

Tuy nhiên vì thời gian và kiến thức có hạn,nhóm em đã cơ bản hoàn thành hệ thống nên còn một vài hạn chế nhất định như sau:

- Hệ thống nhận diện hoạt động kém khi ánh sáng yếu, hoặc khoảng cách xa.

- Vì tập dữ liệu nhân tạo nên đôi khi có sự sai sót nhầm lẫn khi nhận diện các loại khẩu trang kiểu mẫu khác nhau

- Chưa thể đưa ra cảnh báo bằng âm thanh

## **5.2 Hướng phát triển**

Vì hệ thống còn một số hạn chế nên nhóm em sẽ tiếp tục cải thiện hệ thống về mặt nhận diện.

Sưu tập được bộ ảnh dữ liệu thật để nâng cao khả năng nhận diện và đưa ra kết quả chính xác hơn.

Bổ sung tính năng cho phép hệ thống đưa ra cảnh báo bằng âm thanh.

Xây dựng lắp đặt camera giám sát tích hợp việc nhận diện khuôn mặt mang khẩu trang.