**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**



**BÁO CÁO KẾT THÚC HỌC PHẦN**

**HỌC MÁY**

**ĐỀ TÀI: NHẬN DIỆN KHẨU TRANG VÀ CẢM XÚC**

**Giáo viên: TS.Phạm Tiến Lâm**

**Học phần:** Học máy

**Sinh viên:** Nguyễn Đình Quang

Phạm Xuân Tài

Hoàng Minh Quân

Trần Duy Nghĩa

Nguyễn Thu Hà

**HÀ NỘI, NĂM 2023**

MỤC LỤC

[LỜI CẢM ƠN 3](#_Toc136961853)

[Chương 1: Giới thiệu đề tài 4](#_Toc136961854)

[1.1. Giới thiệu về đề tài: 4](#_Toc136961855)

[1.2. Mô tả bài toán thực tế 4](#_Toc136961856)

[Chương 2 5](#_Toc136961857)

[1. Mô hình phát hiện khuôn mặt: 5](#_Toc136961858)

[2. Các thư viện được sử dụng: 7](#_Toc136961859)

[2.2 . Thư viện Numpy 7](#_Toc136961860)

[2.3 . Thư viện OpenCV 8](#_Toc136961861)

[2.4. Thư viện Tensorflow 9](#_Toc136961862)

[2.5. Thư viện Sklearn 10](#_Toc136961863)

[2.6 .Thư viện Matplotlib 11](#_Toc136961864)

[Chương 3. Chương trình thực nghiệm 13](#_Toc136961865)

[3.1 Bộ dữ liệu 13](#_Toc136961866)

[3.2 Chỉ số đánh giá hiệu suất 13](#_Toc136961867)

[3.3 . Huấn luyện mô hình và kết quả thực nghiệm 15](#_Toc136961868)

[3.3.1 Tiền xử lý dữ liệu 15](#_Toc136961869)

[3.3.2 Mô hình huấn luyện CNN 17](#_Toc136961870)

[3.3.3 Tiêu chí ảnh hưởng đến môi hình 21](#_Toc136961871)

[3.3.4 Tiêu chí đánh giá mô hình 21](#_Toc136961872)

[3.3.5 Kết quả thực nghiệm 22](#_Toc136961873)

[Chương 4. Kết luận 27](#_Toc136961874)

[3.3.5.1.1.1.1 Ưu điểm 27](#_Toc136961875)

[4 Nhược điểm 27](#_Toc136961876)

[5 Hướng phát triển trong tương lai 27](#_Toc136961877)

[TÀI LIỆU THAM KHẢO 28](#_Toc136961878)

# LỜI CẢM ƠN

Trong quá trình nghiên cứu môn Học máy, nhóm chúng em đã nhận được sự giúp đỡ nhiệt tình của Thầy cùng với anh trợ giảng. Các Thầy đã cung cấp cho em những kiến thức cơ bản quan trọng để phục vụ trong quá trình nghiên cứu. Chúng em rất trân trọng khoảng thời gian cùng thầy nghiên cứu môn học .

Tuy nhiên do còn có nhiều hạn chế về kiến thức, dự án này của chúng em cũng không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự đóng góp, phê bình của các thầy để đề tài của chúng em được hoàn thiện hơn. Cuối cùng chúng em xin kính chúc hai Thầy nhiều sức khỏe , thành công và hạnh phúc ,…

Chúng em xin chân thành cảm ơn!

# Chương 1: Giới thiệu đề tài

## Giới thiệu về đề tài:

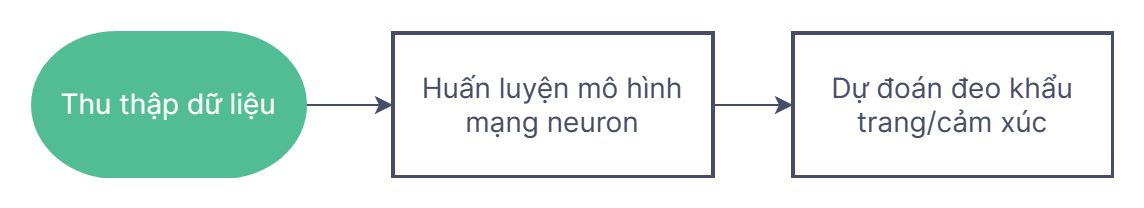
Tuy dịch bệnh Covid 19 đã dần được đẩy lùi nhưng vẫn có nguy cơ bùng phát dịch bệnh cùng với nhiều biến thể phức tạp của nó và sự xuất hiện nhiều loại bệnh lây truyền qua đường hô hấp đã thúc đẩy việc tạo ra những ứng dụng góp phần hỗ trợ việc phòng bệnh, giúp cho người dân dễ dàng kiểm soát việc đeo khẩu trang đúng cách để dễ dàng kiểm soát nguy cơ lây lan dịch bệnh trên diện rộng.

Việc áp dụng các công nghệ deep learning vào bài toán chẩn đoán hình ảnh Covid 19 thông qua ảnh nhận diện khuôn mặt về việc có hay không đeo khẩu trang đã và đang được phát triển. Trong đó mô hình mạng tích chập (Convolutional Neural Network – CNN) đã được sử dụng rất nhiều trong lĩnh vực dự đoán hình ảnh. Trong báo cáo này, nhóm đã xây dựng một mô hình học máy sử dụng mô hình mạng tích chập CNN để nhận diện việc đeo hay không đeo khẩu trang cũng như kết hợp xây dựng mô hình nhận diện cảm xúc.

## Mô tả bài toán thực tế

CNN (Convolutional Neural là một kiến trúc mạng nơ-ron nhân tạo được thiết kế để xử lý, phân loại dữ liệu ảnh, video và âm thanh.

CNN hoạt động bằng cách sử dụng các bộ lọc (filters) để thực hiện phép tích chập (convolution) trên hình ảnh đầu vào. Bộ lọc này sẽ di chuyển trên toàn bộ hình ảnh, tìm kiếm các đặc trưng của hình ảnh bằng cách so sánh các khu vực nhỏ của hình ảnh với một mẫu được học trước. Các đặc trưng được tìm thấy sau đó sẽ được đưa vào một hoặc nhiều lớp ẩn để xử lý và trích xuất thông tin từ các đặc trưng này.

Sau khi thông tin được trích xuất, CNN sử dụng một số lớp Fully Connected (FC) để xử lý dữ liệu và cuối cùng đưa ra kết quả dự đoán về việc đeo có đeo khẩu trang không, đeo khẩu trang có đúng hay không

Việc xây dựng một mô hình CNN cho chẩn đoán đeo khẩu trang yêu cầu một tập dữ liệu lớn về hình ảnh khuôn mặt, trong đó phải có đầy đủ các trường hợp đeo khẩu trang, không đeo khẩu trang, đeo khẩu trang không đúng cách và tập ảnh về cảm xúc ,. Sau đó, mô hình sẽ được huấn luyện trên tập dữ liệu này để học cách phân biệt các đặc trưng của hình ảnh đeo khẩu trang cũng như không đeo và đeo sai cách.

# Chương 2

# Mô hình phát hiện khuôn mặt:

Việc sử dụng mô hình phát hiện khuôn mặt có sẵn như ResnetSSD là một cách tiếp cận hiệu quả để nhận dạng và phát hiện khuôn mặt trong các ứng dụng thực tế. Mô hình này được xây dựng trên nền tảng deep learning và có khả năng phát hiện khuôn mặt với độ chính xác cao.

Để sử dụng mô hình này, bạn cần tải xuống mã nguồn từ đường dẫn trên và cài đặt các thư viện cần thiết. Sau đó, bạn có thể sử dụng mô hình để phát hiện khuôn mặt trong ảnh hoặc video.

Một số lưu ý khi sử dụng mô hình này là:

- Đảm bảo rằng ảnh hoặc video đầu vào đủ lớn và độ phân giải cao để đảm bảo độ chính xác của kết quả phát hiện.

- Có thể cần điều chỉnh các tham số của mô hình để phù hợp với ứng dụng cụ thể của bạn.

- Để đạt được hiệu quả cao nhất, nên sử dụng mô hình này kết hợp với các công cụ nhận dạng khuôn mặt khác để xác định danh tính của người được phát hiện.

Với mô hình phát hiện khuôn mặt ResnetSSD, bạn có thể dễ dàng tích hợp vào các ứng dụng thực tế như hệ thống an ninh, ứng dụng nhận dạng khuôn mặt và nhiều ứng dụng khác.

Trong đề tài này, chúng ta sử dụng một mô hình phát hiện khuôn mặt có sẵn để có thể phát hiện khuôn mặt, từ đó chúng ta có thể dự đoán việc đeo khẩu trang và nhận diện cảm xúc.

**CNN**

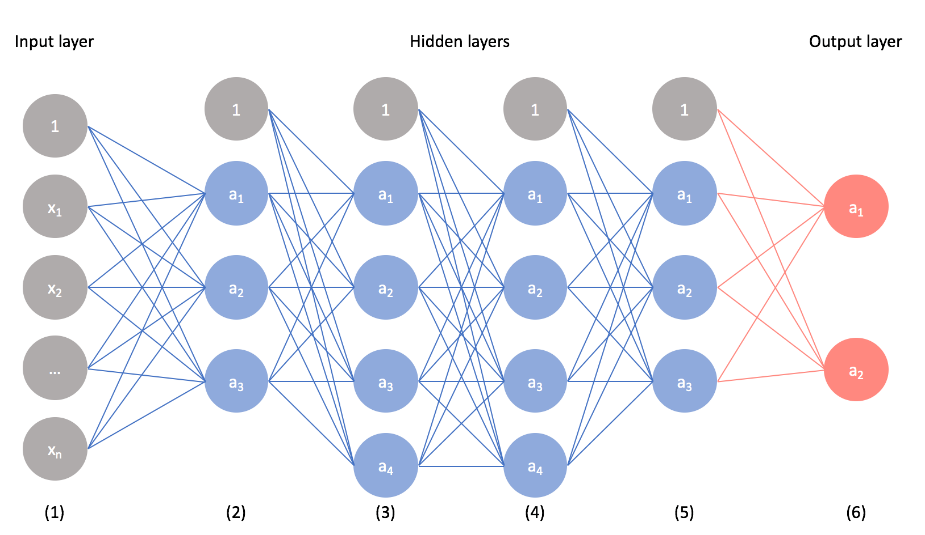
Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. CNN được sử dụng nhiều trong các bài toán nhận dạng các object trong ảnh.

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và Tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo. Các layer liên kết được với nhau thông qua cơ chế convolution(tích chập). Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

Mỗi một lớp được sử dụng các filter khác nhau thông thường có hàng trăm hàng nghìn filter như vậy và kết hợp kết quả của chúng lại. Ngoài ra có một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong quá trình huấn luyện mạng (traning) CNN tự động học các giá trị qua các lớp filter dựa vào cách thức thực hiện. Ví dụ trong tác vụ phân lớp ảnh, CNNs sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.



1. Mô hình cấu trúc của một CNN.

# 2. Các thư viện được sử dụng:

## . Thư viện Numpy

Thư viện NumPy (Numerical Python) là một thư viện Python phổ biến được sử dụng để làm việc với mảng đa chiều và các phép toán số học trên chúng. NumPy cung cấp một cấu trúc dữ liệu mảng mạnh mẽ và các hàm tiện ích để thực hiện các phép toán số học, xử lý mảng và thao tác trên dữ liệu số.

Dưới đây là một số khái niệm và tính năng chính của NumPy:

* Mảng NumPy (ndarray): Mảng NumPy là cấu trúc dữ liệu chính trong NumPy, cho phép lưu trữ và xử lý các mảng đa chiều. Mảng NumPy có kích thước cố định và các phần tử trong mảng cùng kiểu dữ liệu.
* Các phép toán số học: NumPy cung cấp các hàm và toán tử cho phép thực hiện các phép toán số học trên mảng như cộng, trừ, nhân, chia, lũy thừa, căn bậc hai, logarit, v.v.
* Truy cập phần tử: Bạn có thể truy cập và thay đổi giá trị các phần tử trong mảng NumPy bằng cách sử dụng chỉ mục và cắt mảng (slicing).
* Hàm toán học và thống kê: NumPy cung cấp nhiều hàm toán học và thống kê tiện ích như min, max, mean, sum, std, v.v. để thao tác trên mảng.
* Hàm điều kiện: NumPy cung cấp các hàm và phương thức cho phép kiểm tra và lựa chọn các phần tử trong mảng dựa trên các điều kiện như np.where, np.logical\_and, np.logical\_or, v.v.
* Thao tác trên mảng: NumPy cung cấp các hàm và phương thức để thực hiện các phép biến đổi mảng như reshape, transpose, flatten, concatenate, v.v.
* Tích hợp C/C++: NumPy cho phép tích hợp mã C/C++ vào mã Python thông qua giao diện C API của nó, giúp tăng tốc độ xử lý các phép toán số học.
* NumPy được sử dụng rộng rãi trong nhiều lĩnh vực như khoa học dữ liệu, máy học, tính toán khoa học, xử lý ảnh và âm thanh, mô phỏng vật lý, v.v. Thư viện này là một phần quan trọng của hệ sinh thái Python cho tính toán số và xử lý dữ liệu.

## . Thư viện OpenCV

Thư viện cv2 (OpenCV) là một thư viện mã nguồn mở được sử dụng rộng rãi trong xử lý ảnh và thị giác máy tính trong ngôn ngữ lập trình Python. OpenCV cung cấp nhiều chức năng và công cụ mạnh mẽ để xử lý, phân tích và trực quan hóa ảnh và video.

Dưới đây là một số chức năng chính của thư viện cv2:

* Đọc và ghi ảnh và video: OpenCV cung cấp các hàm để đọc và ghi các tệp tin ảnh và video từ các nguồn khác nhau, bao gồm ổ đĩa, camera, v.v.
* Xử lý ảnh: OpenCV cung cấp nhiều chức năng để xử lý ảnh, bao gồm chuyển đổi màu sắc, cắt, xoay, thu phóng, lật, v.v. Bạn cũng có thể áp dụng các bộ lọc, làm mờ, lọc nhiễu, phát hiện cạnh và nhiều phép biến đổi khác cho ảnh.
* Xử lý video: OpenCV hỗ trợ xử lý video, bao gồm khung hình theo thời gian, trích xuất khung hình, ghi lại video và áp dụng các hiệu ứng video.
* Xử lý thị giác máy tính: OpenCV cung cấp các chức năng và thuật toán để phân tích và nhận dạng các đối tượng trong ảnh, bao gồm nhận dạng khuôn mặt, phát hiện vật thể, trích xuất đặc trưng, theo dõi đối tượng, v.v.
* Xử lý điểm ảnh: OpenCV cho phép bạn truy cập và chỉnh sửa các điểm ảnh trực tiếp, bao gồm xử lý pixel, tạo hiệu ứng đồ họa và thực hiện các phép tính điểm ảnh phức tạp.
* Hiển thị ảnh: OpenCV cung cấp các chức năng để hiển thị ảnh và video trực tiếp trên màn hình, tạo cửa sổ đồ họa và tương tác với ảnh.

Thư viện cv2 có tính ổn định, tốc độ xử lý nhanh và rất phổ biến trong cộng đồng xử lý ảnh và thị giác máy tính.

## 2.4. Thư viện Tensorflow

Thư viện TensorFlow là một thư viện mã nguồn mở phát triển bởi Google AI và được sử dụng rộng rãi trong lĩnh vực học máy và trí tuệ nhân tạo. TensorFlow cung cấp một cấu trúc dữ liệu gọi là "đồ thị tính toán" (computational graph) để biểu diễn và thực thi các phép tính toán số trên dữ liệu. Đồ thị tính toán trong TensorFlow bao gồm các nút (nodes) đại diện cho các phép tính toán và các cung (edges) đại diện cho luồng dữ liệu giữa các nút.

Dưới đây là một số khái niệm và tính năng chính của TensorFlow:

* Đồ thị tính toán (Computational graph): TensorFlow sử dụng đồ thị tính toán để biểu diễn và thực thi các phép tính toán. Đồ thị tính toán giúp tối ưu hóa và tận dụng được tính song song của phần cứng để thực hiện các phép tính nhanh chóng.
* Tensors: TensorFlow sử dụng cấu trúc dữ liệu tensor để lưu trữ và xử lý dữ liệu. Tensor có thể là một vector, ma trận, hay một mảng đa chiều với các phần tử cùng kiểu dữ liệu.
* Các lớp và phép tính: TensorFlow cung cấp một loạt các lớp và phép tính để xây dựng các mô hình học máy. Các lớp bao gồm các lớp mạng nơ-ron, lớp tích chập, lớp tổng hợp, lớp tái tạo, v.v. Phép tính bao gồm các phép tính toán như cộng, trừ, nhân, chia, lũy thừa, v.v.
* Tối ưu hóa và tạo mô hình: TensorFlow cung cấp các tối ưu hóa để tối đa hóa hiệu suất và tăng tốc độ huấn luyện mô hình. Ngoài ra, TensorFlow cũng cung cấp khả năng tạo và lưu trữ các mô hình đã huấn luyện để sử dụng sau này.
* Tích hợp và mở rộng: TensorFlow cho phép tích hợp và mở rộng với các công cụ và thư viện khác như Keras, scikit-learn, OpenCV, v.v. Điều này giúp thực hiện các tác vụ phức tạp và kết hợp các công cụ và thuật toán khác nhau.
* Tính tương thích và di động: TensorFlow hỗ trợ nhiều phiên bản và cung cấp tính tương thích đa nền tảng, cho phép chạy trên các thiết bị di động, máy tính cá nhân, máy chủ, và cụm máy tính phân tán.
* Cộng đồng và tài liệu: TensorFlow có một cộng đồng đông đảo và phong phú, cung cấp các tài liệu hướng dẫn, ví dụ mã, bài viết và thảo luận trên các diễn đàn, giúp người dùng tìm hiểu và giải quyết các vấn đề khi sử dụng thư viện này.

TensorFlow là một công cụ mạnh mẽ trong lĩnh vực học máy và trí tuệ nhân tạo, cho phép xây dựng và huấn luyện các mô hình phức tạp và giải quyết các bài toán thực tế.

## 2.5. Thư viện Sklearn

Scikit-learn, hay còn được gọi là sklearn, là một thư viện mã nguồn mở phổ biến trong ngôn ngữ lập trình Python được sử dụng cho Machine Learning và Data Mining. Scikit-learn cung cấp nhiều công cụ và thuật toán tiện ích để xây dựng và đánh giá các mô hình học máy, phân tích dữ liệu và thực hiện các tác vụ tiền xử lý dữ liệu.

Dưới đây là một số chức năng và tính năng chính của thư viện scikit-learn:

* Cung cấp các thuật toán học máy tiêu chuẩn: Scikit-learn cung cấp các thuật toán học máy phổ biến như hồi quy tuyến tính, hồi quy logistic, máy vector hỗ trợ (SVM), cây quyết định, Random Forest, K-means clustering, v.v. Các thuật toán này được triển khai một cách hiệu quả và dễ sử dụng.
* Tích hợp các công cụ tiền xử lý dữ liệu: Scikit-learn cung cấp nhiều công cụ tiền xử lý dữ liệu như chuẩn hóa dữ liệu, mã hóa biến định danh, xử lý giá trị thiếu, trích xuất đặc trưng, v.v. Điều này giúp chuẩn bị dữ liệu cho việc huấn luyện mô hình.
* Đánh giá và tối ưu hóa mô hình: Scikit-learn cung cấp các phép đo và công cụ để đánh giá hiệu suất của mô hình học máy, bao gồm chia dữ liệu thành tập huấn luyện và tập kiểm tra, cross-validation, tính toán độ chính xác, độ phân loại, mất mát, v.v. Ngoài ra, scikit-learn cũng cung cấp các công cụ tối ưu hóa để điều chỉnh các siêu tham số của mô hình.
* Tiện ích cho việc xử lý dữ liệu và giám sát: Scikit-learn cung cấp các công cụ và tiện ích để khám phá và xử lý dữ liệu, như phân tích thành phần chính (PCA), giảm chiều dữ liệu, trực quan hóa dữ liệu, tìm kiếm siêu tham số tốt nhất, v.v.
* Tích hợp với các thư viện khác: Scikit-learn tích hợp tốt với các thư viện khác trong hệ sinh thái của Python như NumPy, Pandas và Matplotlib, tạo điều kiện thuận lợi cho xử lý dữ liệu và trực quan hóa kết quả.

Scikit-learn là một công cụ mạnh mẽ và phổ biến trong lĩnh vực Machine Learning, đặc biệt là trong các tác vụ phân loại, hồi quy và gom cụm dữ liệu. Với tư cách là một thư viện mã nguồn mở, scikit-learn đang tiếp tục được phát triển và cung cấp cập nhật mới để phục vụ cộng đồng Machine Learning ngày càng lớn.

## .Thư viện Matplotlib

Matplotlib là một thư viện trong Python được sử dụng để tạo và hiển thị đồ thị, biểu đồ, hình ảnh và các loại visualizations khác. Nó cung cấp các công cụ mạnh mẽ để tạo ra các biểu đồ chất lượng cao, giúp bạn trực quan hóa dữ liệu một cách dễ dàng và linh hoạt.

Dưới đây là một số thành phần chính trong thư viện matplotlib:

* pyplot: Giao diện API cơ bản của Matplotlib, cung cấp các hàm để tạo và tùy chỉnh đồ thị và biểu đồ.
* Figure: Đại diện cho một hình ảnh hoặc một tệp tin hình ảnh.
* Axes: Cung cấp các phương thức để tạo các đối tượng đồ thị như các trục, điểm, đường, v.v.
* Subplots: Cho phép tạo và quản lý nhiều đồ thị trong cùng một hình ảnh.
* Plot: Hàm để tạo các loại đồ thị và biểu đồ, bao gồm đồ thị đường (line plot), biểu đồ cột (bar plot), biểu đồ hộp (box plot), đồ thị điểm (scatter plot), v.v.
* Title, Label, Legend`: Cung cấp các phương thức để thêm tiêu đề, nhãn và chú giải vào đồ thị.
* Axis: Quản lý và tùy chỉnh các trục của đồ thị, bao gồm các giá trị trục, nhãn, v.v.
* Colorbar: Hiển thị thanh màu để giải thích giá trị của màu sắc trong đồ thị.

Matplotlib cũng hỗ trợ nhiều kiểu đồ thị và biểu đồ, cho phép bạn tùy chỉnh màu sắc, kích thước, kiểu đường, điểm, các đánh dấu trục, v.v. Thư viện này rất linh hoạt và mạnh mẽ, cho phép bạn tạo ra các visualizations phức tạp và tùy chỉnh chúng theo ý muốn.

**2.7. Thư viện Keras**

Keras là một thư viện mã nguồn mở cho Python được sử dụng để xây dựng các mô hình học máy và mạng nơ-ron. Nó được thiết kế để làm cho việc xây dựng mô hình học máy trở nên dễ dàng và nhanh chóng hơn.

Keras có nhiều tính năng hữu ích, bao gồm:

- Dễ dàng sử dụng: Keras được thiết kế để làm cho việc xây dựng mô hình học máy trở nên dễ dàng và trực quan hơn.

- Tích hợp với các thư viện tính toán số: Keras hỗ trợ các thư viện tính toán số như TensorFlow, Theano và CNTK, cho phép bạn chọn thư viện tính toán số phù hợp nhất cho dự án của mình.

- Hỗ trợ nhiều loại mô hình: Keras hỗ trợ nhiều loại mô hình, bao gồm mạng nơ-ron tiêu chuẩn, mạng nơ-ron tích chập và mạng nơ-ron tái tạo.

- Tích hợp với các công cụ tối ưu hóa: Keras cung cấp các công cụ tối ưu hóa để giúp bạn tìm kiếm các tham số tốt nhất cho mô hình của mình.

- Hỗ trợ các lớp và hàm kích hoạt tiêu chuẩn: Keras cung cấp các lớp và hàm kích hoạt tiêu chuẩn để giúp bạn xây dựng mô hình học máy nhanh chóng và dễ dàng.

Keras là một trong những thư viện phổ biến nhất cho việc xây dựng các mô hình học máy và mạng nơ-ron.

# Chương 3. Chương trình thực nghiệm

## 3.1 Bộ dữ liệu

Bộ dữ liệu hình ảnh được lấy từ:

## 3.2 Chỉ số đánh giá hiệu suất

Để đánh giá phân loại ảnh được chính xác, các mô hình trong phần trước sữ được chạy bằng cách sử dụng bộ dữ liệu dưới dạng hình ảnh, vì vậy cần có sự điều chỉnh để nâng cap độ chính xác của chúng.

Đối với mỗi mô hình 3 kết quả điển hình được hiển thị trong CNN là:

* Đồ thị độ chính xác (model ACC).

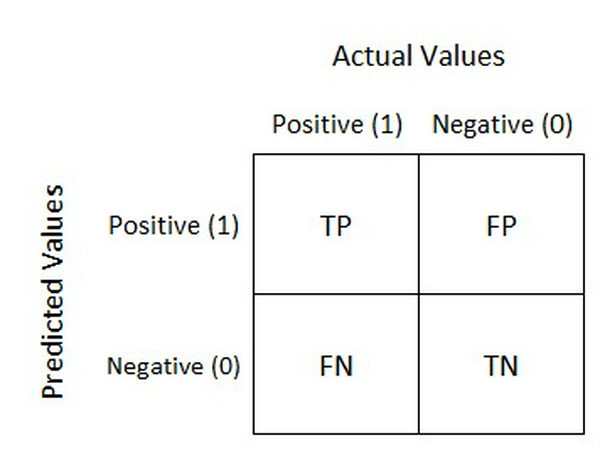
ACC cho biết sự thay đổi độ chính xác của mô hình qua các vòng lặp/epochs trong quá trình huấn luyện.

* Đồ thị loss (model loss).

Đồ thị loss biểu thị sự thay đổi của hàm loss của mô hình qua các vòng lặp hoặc epochs trong quá trình huấn luyện. Hàm loss tính toán sự sai khác giữa giá trị dự đoán của mô hình và giá trị thực tế.

* Confusion matrix.

Về cơ bản, confusion matrix thể hiện có bao nhiêu điểm dữ liệu thực sự thuộc vào một class, và được dự đoán là rơi vào một class. Một confusion matrix gồm 4 chỉ số sau đối với mỗi lớp phân loại:



* **TP (True Positive)**: Số lượng dự đoán chính xác.
* **TN (True Negative)**: Số lương dự đoán chính xác một cách gián tiếp. Là khi mô hình dự đoán đúng một người không bị bệnh tức là việc không chọn trường hợp bị bệnh là chính xác.
* **FP (False Positive - Type 1 Error)**: Số lượng các dự đoán sai lệch. Là khi mô hình dự đoán một người bị bệnh và người đó hoàn toàn khỏe mạnh.
* **FN (False Negative - Type 2 Error)**: Số lượng các dự đoán sai lệch một cách gián tiếp. Là khi mô hình dự đoán một người không bị bệnh nhưng người đó bị bệnh, tức là việc không chọn trường hợp bị bệnh là sai.

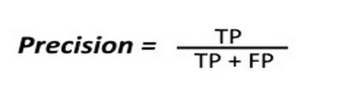
 Từ 4 chỉ số này, ta có 4 chỉ số để đánh giá mức độ tin cậy của một mô hình:

* **Accuracy:** Được tính bằng cách chia tổng số dự đoán đúng cho tất cả các dự đoán.

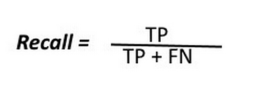
   



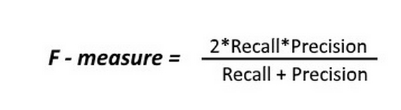
* **Precision**: Trong tất cả các dự đoán Positive được đưa ra, bao nhiêu dự đoán là chính xác?



* **Recall*:*** Trong tất cả các trường hợp Positive, bao nhiêu trường hợp đã được dự đoán chính xác?



* **F-score**: Dùng để đánh giá mức độ tin cậy chung của mô hình**:**



## . Huấn luyện mô hình và kết quả thực nghiệm

Do các bước tiền xử lý dữ liệu của 2 mô hình là như nhau thế nên trong báo cáo này em chỉ trình bày cách nhóm em xử lý dữ liệu của bộ dự hiệu khẩu trang

### Tiền xử lý dữ liệu

* Bước tiền xử lý dữ liệu gần như giống nhau ở cả 2 mô hình
* Import một số thư viện để xử lí và phân tích trực quan dữ liệu :
* import glob
* import shutil
* import cv2
* from PIL import Image
* import PIL
* import os
* import numpy as np
* import pandas as pd
* import matplotlib.pyplot as plt
* import seaborn as sns
* import random
* from sklearn.model\_selection import train\_test\_split
* import tensorflow as tf
* import keras
* from keras.preprocessing import image
* from keras.models import Sequential
* from keras.layers import Conv2D, MaxPool2D, Flatten,Dense,Dropout,BatchNormalization
* from keras import regularizers
* import warnings
* # Ignore waring
* warnings.filterwarnings('ignore')
* Tiền xử lí dữ liệu:

Tiền xử lý dữ liệu là một bước rất quan trọng trong việc giải quyết bất kỳ vấn đề nào trong lĩnh vực học máy. Hầu hết các bộ dữ liệu được sử dụng trong các vấn đề liên quan đến học máy cần được xử lý, làm sạch và biến đổi trước khi một thuật toán có thể được huấn luyện trên những bộ dữ liệu này.

* Load dữ liệu hình ảnh vào tập train và labels

+ Tạo đường dẫn :

mask\_path = ['/without\_mask/\*','/with\_mask/\*','/incorrect\_mask/\*' ]

train\_path = '/content/drive/MyDrive/Dataset/processed'

+ Load dữ liệu

X = []

y = []

for i, path in enumerate(mask\_path):

  for name in glob.glob(train\_path + path):

    img = cv2.imread(name)

    img = cv2.cvtColor(img,cv2.COLOR\_BGR2RGB)

    img = cv2.resize(img,(48,48))

    X.append((img))

    y.append(i)

len(X)

* Chuẩn hóa dữ liệu hình ảnh :

X = np.array(X)/255.

y = np.array(y)

Các giá trị pixel được đưa về các giá trị trong khoảng từ 0 đến 1 giúp mô hình học tốt hơn.

* Phân tích dữ liệu thành tập train và tập test :

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=2)

* Sau đó ta tạo 4 biến, gồm X\_train, y\_train và X\_test, y\_test. Với đối số

truyền vào là giá trị X, y ta đã lấy từ dữ liệu bên trên, test\_size trả về cho ta

phần trăm dữ liệu được chia, ví dụ 0.2 tương ứng với dữ liệu được chia thành

20% giá trị là test, còn lại là dữ liệu train. random\_state bằng một số tương ứng

nào đó để đảm bảo mỗi lần ta chạy lại mô hình, giá trị phân tách ngẫu nhiên

nhận được là giống nhau, bạn có thể cho số nào bất kỳ.

* Chuẩn hóa dữ liệu output :

from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder()

encoder.fit(y\_train.reshape(-1,1))

y\_train = encoder.transform(y\_train.reshape(-1,1)).toarray()

y\_test = encoder.transform(y\_test.reshape(-1,1)).toarray()

print(y\_train)

* Chuyển về dạng onehot để phân biệt nhiều ảnh một cách dễ dàng

### Mô hình huấn luyện CNN

**1.Mô hình nhận dạng khẩu trang**

inp = Input(shape = (48,48,3))

cnn = Conv2D(filters =16,kernel\_size = 3,activation ='relu')(inp)

pooling = MaxPooling2D(pool\_size =(2,2))(cnn)

drop = Dropout(0.2)(pooling)

cnn = Conv2D(filters =32,kernel\_size = 3,activation ='relu')(inp)

pooling = MaxPooling2D(pool\_size =(2,2))(cnn)

drop = Dropout(0.2)(pooling)

cnn = Conv2D(filters =64,kernel\_size = 4,activation ='relu')(inp)

pooling = MaxPooling2D(pool\_size =(2,2))(cnn)

drop = Dropout(0.2)(pooling)

cnn = Conv2D(filters =128, kernel\_size =4,activation ='relu')(drop)

pooling = MaxPooling2D(pool\_size =(2,2))(cnn)

drop = Dropout(0.2)(pooling)

f =Flatten()(pooling)

fc1 = Dense(units =128, activation ='relu')(f)

fc2 = Dense(units =64, activation ='relu')(fc1)

fc3 = Dense(units =32, activation ='relu')(fc2)

out = Dense(units =3, activation ='softmax')(fc3)

model =Model(inputs = inp,outputs =out)

model.summary()

* Đoạn code này định nghĩa một mô hình mạng neural tích chập (CNN) để phân loại ảnh với ba lớp đầu ra.
* Biến inp định nghĩa một hình dạng đầu vào là 48x48 pixel với 3 kênh màu (RGB).
* Sau đó là bốn lớp tích chập (Conv2D) với số lượng bộ lọc, kích thước kernel và kích hoạt ReLU tăng dần.Lớp đệm padding để giữ nguyên kích thưởng đầu ra Mỗi lớp tích chập được theo sau bởi một lớp giảm kích thước (MaxPooling2D) để giảm kích thước không gian của đầu ra từ lớp tích chập. Các lớp giảm kích thước này giúp giảm thiểu quá khớp và cải thiện tốc độ và hiệu suất. Mỗi lớp giảm kích thước cũng được theo sau bởi một lớp dropout (Dropout) với tỷ lệ 0,2, loại bỏ ngẫu nhiên 20% của các neuron để giảm thiểu quá khớp.
* Sau các lớp tích chập và giảm kích thước, đầu ra đã được làm phẳng từ lớp giảm kích thước cuối cùng, được đưa vào ba lớp kết nối đầy đủ (Dense) với số lượng đơn vị giảm dần và kích hoạt ReLU. Lớp đầu ra có 3 đơn vị với hàm kích hoạt softmax, xuất ra xác suất lớp cho mỗi ảnh đầu vào.
* Mô hình được tạo ra bằng cách sử dụng hàm Model từ Keras, với lớp đầu vào và đầu ra được xác định là inp và out, tương ứng.
* Cuối cùng, hàm model.summary() được sử dụng để in ra tóm tắt kiến trúc mô hình, bao gồm các loại lớp, hình dạng và số lượng tham số.
* Sau khi đã xây dựng được kiến trúc của mô hình . Bước tiếp theo là thực hiện huấn luyện mô hình :
* optimizer1 = tensorflow.keras.optimizers.Adam(learning\_rate = 0.0001)
* model.compile(optimizer = optimizer1, loss = 'categorical\_crossentropy',metrics =['accuracy'])
* history = model.fit(X\_train,y\_train,batch\_size =16, epochs =25,validation\_data =(X\_test,y\_test))
  1. Mô hình nhận diện cảm xúc

model= tf.keras.models.Sequential()

model.add(Conv2D(32, kernel\_size=(3, 3), padding='same', activation='relu', input\_shape=(48, 48,1)))

model.add(Conv2D(64,(3,3), padding='same', activation='relu' ))

model.add(BatchNormalization())

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(128,(5,5), padding='same', activation='relu'))

model.add(BatchNormalization())

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel\_regularizer=regularizers.l2(0.01)))

model.add(BatchNormalization())

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu', kernel\_regularizer=regularizers.l2(0.01)))

model.add(BatchNormalization())

model.add(MaxPool2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256,activation = 'relu'))

model.add(BatchNormalization())

model.add(Dropout(0.25))

model.add(Dense(512,activation = 'relu'))

model.add(BatchNormalization())

model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))

model.summary()

- Đây là một mô hình CNN (Convolutional Neural Network) có kiến trúc phức tạp. Mô hình bao gồm 4 lớp tích chập (Conv2D) với các bộ lọc kích thước khác nhau và hàm kích hoạt là ReLU.

- Lớp đầu tiên có 32 bộ lọc kích thước 3x3, lớp thứ hai có 64 bộ lọc kích thước kernel 3x3, lớp thứ ba là lớp có 128 bộ lọc kích thước 5x5 và lớp cuối cùng là lớp với 512 bộ lọc kích thước 3x3 và kết hợp với hàm chính quy (regularization) để giúp mô hình tránh overfitting.

- Sau mỗi lớp tích chập, model sử dụng một lớp BatchNormalization để chuẩn hóa giá trị và tăng tốc độ hội tụ. Lớp MaxPool2D được sử dụng để giảm kích thước của đầu ra và giảm số lượng tham số của mô hình. Lớp Dropout được sử dụng để giảm hiện tượng overfitting.

- Cuối cùng, một lớp Dense với 256 đơn vị và hàm kích hoạt ReLU được thêm vào mô hình, sau đó theo sau bởi một lớp Dropout và BatchNormalization. Sau đó, một lớp Dense với 512 đơn vị và hàm kích hoạt ReLU được chỉ định kế tiếp và kết thúc bằng lớp Dense với 7 đơn vị đầu ra với hàm kích hoạt soft\_max (7 lớp tương ứng với 7 cảm xúc).

* **Sau khi có kiến trúc của mô hình cảm xúc bước tiếp theo là huấn luyện mô hình :**
* learning\_rate = 0.0001  # Tốc độ học ban đầu
* # Khởi tạo learning rate scheduler
* optimizer1 = tf.keras.optimizers.Adam(learning\_rate = 0.0001)
* model.compile(optimizer = optimizer1, loss = 'categorical\_crossentropy',metrics =['accuracy'])
* history = model.fit(X\_train, y\_train,
* epochs=40,validation\_data=(X\_test\_scaled, y\_test),batch\_size = 64,shuffle = True)
* optimizer1: Optimizer được khởi tạo bằng cách sử dụng Adam optimizer từ tensorflow.keras.optimizers, với learning\_rate = 0.0001
* model.compile(): Cấu hình mô hình với optimizer, hàm mất mát là categotical\_crossentropy và các metric là accuracy để đánh giá hiệu suất mô hình.
* model.fit(): Huấn luyện mô hình trên dữ liệu huấn luyện (X\_train, y\_train) với các tham số như batch\_size, epochs, và sử dụng dữ liệu kiểm tra (X\_test, y\_test) để đánh giá hiệu suất của mô hình trong quá trình huấn luyện.

### Tiêu chí ảnh hưởng đến môi hình

.Có nhiều yếu tố ảnh hưởng đến độ hiệu quả của một mô hình máy học. Dưới đây là một số tiêu chí quan trọng:

1. Độ rõ ràng và độ phong phú của dữ liệu: Mô hình sẽ hoạt động tốt hơn khi dữ liệu huấn luyện rõ ràng, có độ phân loại cao và đủ đa dạng. Dữ liệu không rõ ràng, nhiễu và thiếu thông tin có thể làm giảm độ hiệu quả của mô hình.

2. Số lượng và chất lượng dữ liệu: Một số lượng dữ liệu huấn luyện đủ lớn có thể giúp mô hình học được các mẫu và mối quan hệ phức tạp. Đồng thời, chất lượng dữ liệu cũng quan trọng, bao gồm độ chính xác, đồng nhất và đại diện cho phân phối dữ liệu thực tế.

3. Chọn mô hình phù hợp: Một mô hình phải được lựa chọn dựa trên yêu cầu cụ thể của bài toán. Mô hình phải có khả năng học và biểu diễn các mẫu và quan hệ trong dữ liệu một cách hiệu quả.

4. Quá trình huấn luyện: Thời gian huấn luyện, tỷ lệ học tập, thuật toán tối ưu hóa và các siêu tham số khác cũng có thể ảnh hưởng đến hiệu quả của mô hình. Quá trình huấn luyện phải được điều chỉnh và tối ưu để đạt được kết quả tốt.

5. Tính diễn giải và khả năng áp dụng: Một mô hình có khả năng diễn giải tốt và có thể áp dụng vào thực tế sẽ có hiệu quả cao hơn. Sự diễn giải giúp người dùng hiểu rõ quyết định của mô hình và tin tưởng vào kết quả dự đoán.

- Lưu ý rằng các tiêu chí ảnh hưởng đến hiệu quả của mô hình có thể khác nhau tùy thuộc vào bài toán cụ thể và ngữ cảnh sử dụng mô hình.

### Tiêu chí đánh giá mô hình

- Khi đánh giá huấn luyện một mô hình máy học, có một số tiêu chí quan trọng cần xem xét. Dưới đây là một số tiêu chí phổ biến để đánh giá mô hình:

1. Độ chính xác (Accuracy): Đây là tiêu chí cơ bản để đánh giá khả năng dự đoán chính xác của mô hình trên tập dữ liệu kiểm tra. Độ chính xác được tính bằng tỷ lệ giữa số lượng dự đoán đúng và tổng số mẫu trong tập kiểm tra.

2. Mất mát (Loss): Mất mát đo lường mức độ sai khác giữa các dự đoán của mô hình và giá trị thực tế tương ứng. Một hàm mất mát được sử dụng để đo lường sự sai khác này và mục tiêu là tìm cách giảm mất mát trong quá trình huấn luyện.

3. Đồng nhất (Consistency): Mô hình cần cho ra kết quả tương tự khi được huấn luyện lại trên các tập dữ liệu khác nhau hoặc khi được huấn luyện nhiều lần trên cùng một tập dữ liệu. Sự đồng nhất giúp đảm bảo tính ổn định và tin cậy của mô hình.

4. Quá khớp (Overfitting) và thiếu khớp (Underfitting): Quá khớp xảy ra khi mô hình đã học nhớ quá mức từ dữ liệu huấn luyện, nhưng không thể tổng quát hóa tốt cho các dữ liệu mới. Thiếu khớp xảy ra khi mô hình không học được đủ thông tin từ dữ liệu huấn luyện và không thể dự đoán chính xác trên dữ liệu mới. Đánh giá sự quá khớp và thiếu khớp là một tiêu chí quan trọng để đảm bảo mô hình có khả năng tổng quát hóa tốt.

5. Thời gian huấn luyện (Training time): Thời gian huấn luyện là thời gian mô hình cần để học từ dữ liệu huấn luyện. Đánh giá thời gian huấn luyện là quan trọng đặc biệt khi xem xét các mô hình phức tạp hoặc dữ liệu lớn.

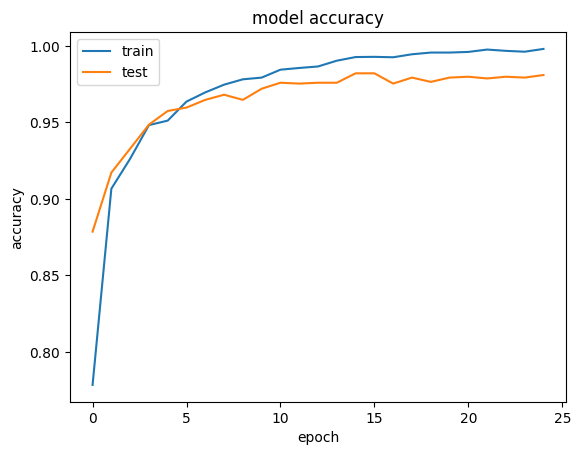
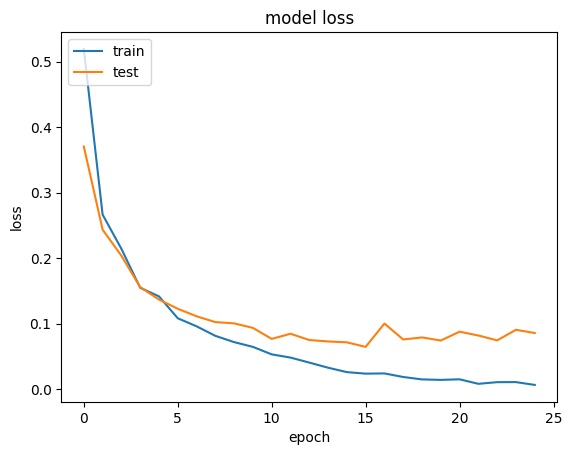
6. Khả năng diễn giải (Interpretability): Khả năng diễn giải của mô hình đo lường khả năng hiểu được lý do tại sao một dự đoán được thực hiện. Một mô hình diễn giải tốt có thể cung cấp giải thích rõ ràng và logic cho quá trình ra quyết định.

7. Hiệu suất tính toán (Computational performance): Đánh giá hiệu suất tính toán của mô hình là quan trọng, đặc biệt khi triển khai mô hình trên các hệ thống có tài nguyên hạn chế. Thời gian dự đoán và khả năng làm việc với dữ liệu lớn là những yếu tố quan trọng để xem xét.

* Những tiêu chí trên là chỉ một số ví dụ phổ biến. Đánh giá mô hình cần tuân thủ các tiêu chí phù hợp với bài toán cụ thể và ngữ cảnh sử dụng mô hình đó.

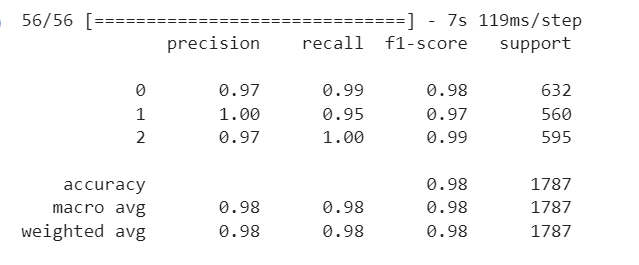
### Kết quả thực nghiệm

**1. Kết quả huấn luyện tập dữ liệu khẩu trang**

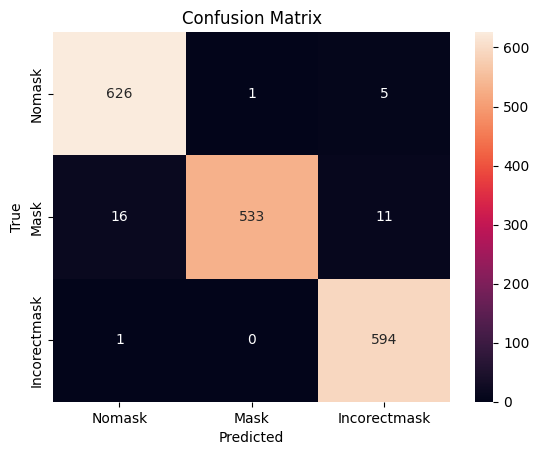


Đồ thị ACC, độ chính xác (accuracy) lên đến 98% và bắt đầu ổn định khi epoch =10, cho cả tập dữ liệu huấn luyện và tập test, đồ thị loss của cũng cho thấy sau mức này, loss của cả tập train và validation. Kết quả cho thấy mô hình có độ phù hợp tốt và không bị quá khớp (over-fitting).

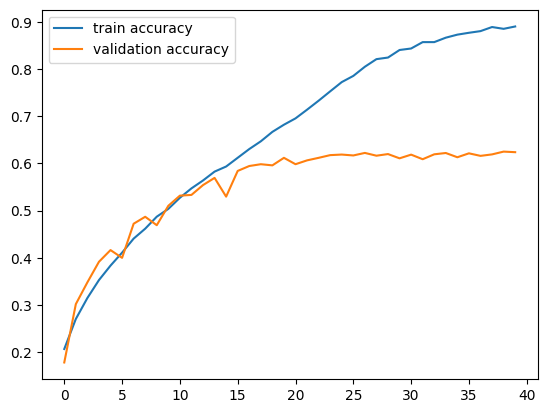
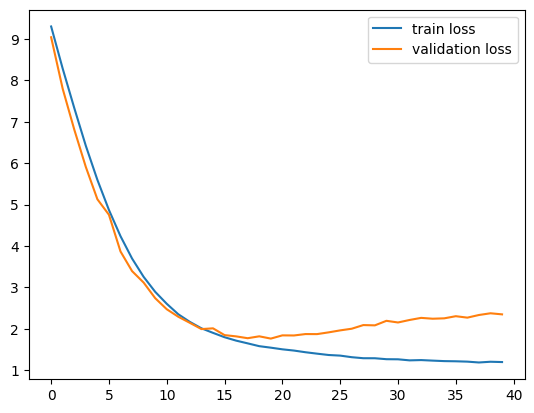
* Đánh giá hiệu suất của mô hình



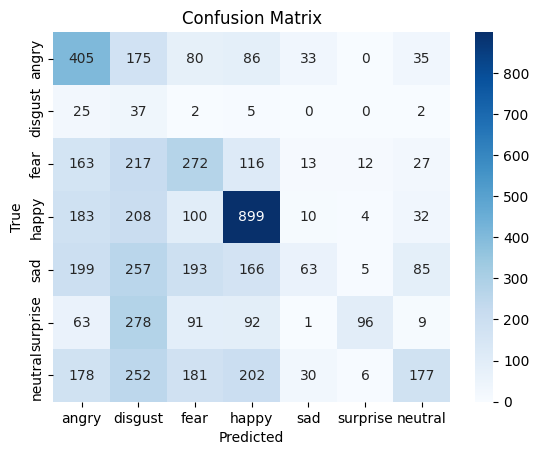
* Confusion matrix :



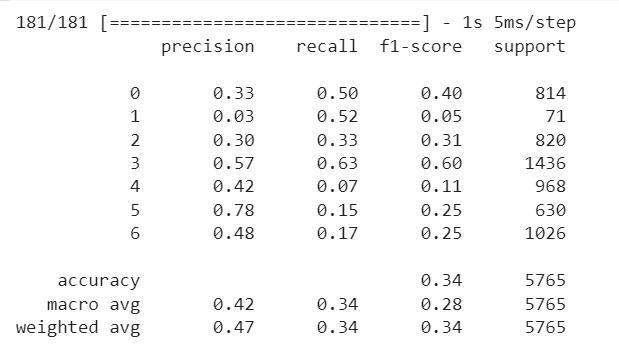
**2. Kết quả huất luyện tập dữ liệu cảm xúc**



* Đồ thị ACC của mô hình cho thấy sau epoch = 20, độ chính xác (accuracy) bắt đầu có sự phân biệt giữa tập dữ liệu huấn luyện và tập test, đồ thị loss của cũng cho thấy sau mức này, loss của cả tập train và validation. Kết quả cho thấy mô hình có độ phù hợp tốt và không bị quá khớp (over-fitting).
* Confusion matrix :



* Đánh giá hiệu suất của mô hình :



### Thử nghiệm với thời gian thực webcam

* Sử dụng thư viện open cv để test kết quả trên thời gian thực
* Bước 1 : Load model

from tensorflow.keras.models import load\_model

model1 = load\_model('model\_mask\_final.h5')

model2 = load\_model('model\_emotions\_final.h5')

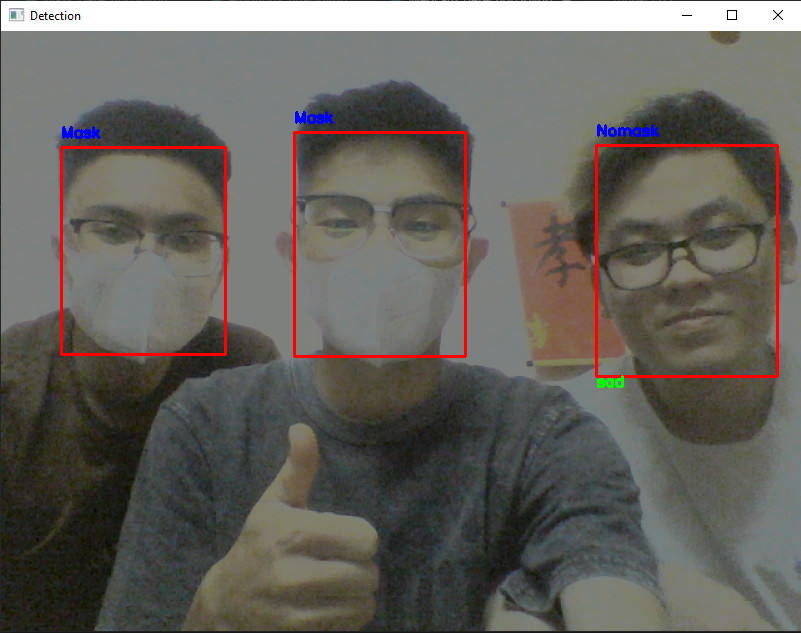
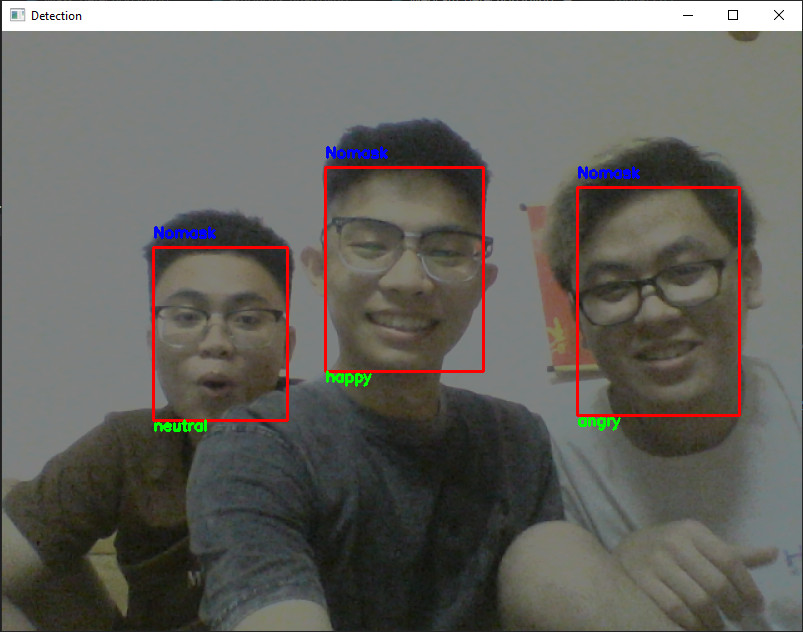
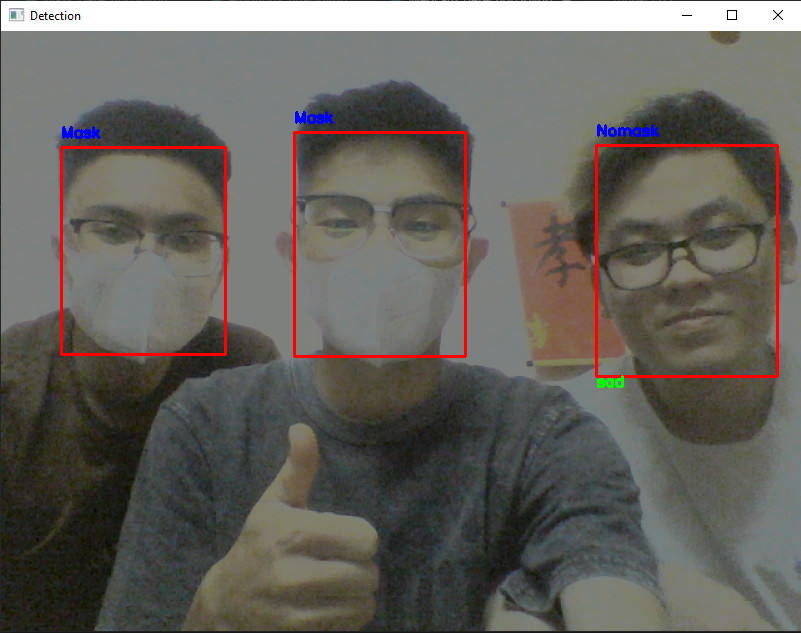
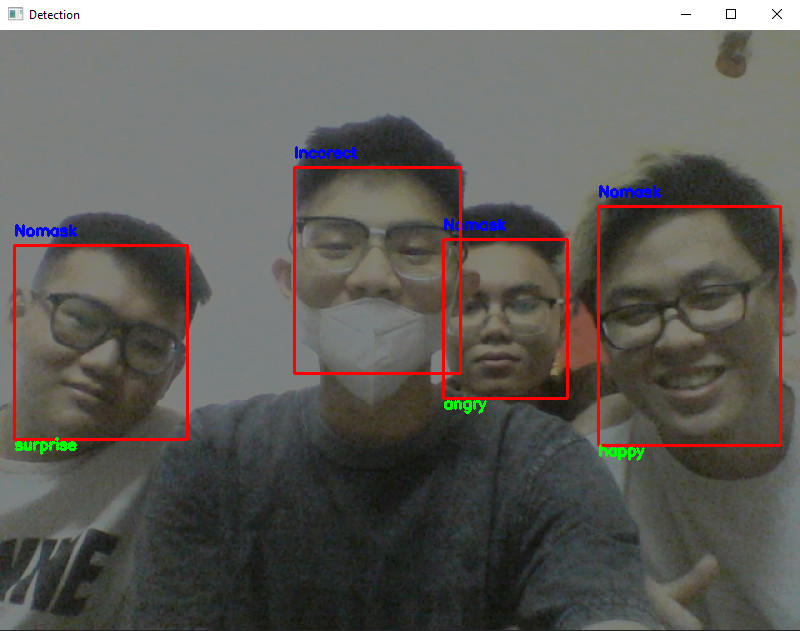
* Bước 2 : Phát hiện khuôn mặt

prototxt = 'deploy.prototxt'

model = 'res10\_300x300\_ssd\_iter\_140000.caffemodel'

net = cv2.dnn.readNetFromCaffe(prototxt, model)

* Bước 3 : Chạy webcam
* cap = cv2.VideoCapture(0)
* while True:
* ret, frame = cap.read()
* if ret:
* frame = imutils.resize(frame, width=600)
* (h, w) = frame.shape[:2]
* blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)), 1.0, (300, 300), (104.0, 177.0, 123.0))
* net.setInput(blob)
* detections = net.forward()
* for i in range(0, detections.shape[2]):
* confidence = detections[0, 0, i, 2]
* if confidence > 0.5:
* box = detections[0, 0, i, 3:7] \* np.array([w, h, w, h])
* (startX, startY, endX, endY) = box.astype("int")
* text = "{:.2f}%".format(confidence \* 100)
* y = startY - 10 if startY - 10 > 10 else startY + 10
* minX, maxX = min(startX, endX), max(startX, endX)
* minY, maxY = min(startY, endY), max(startY, endY)
* face = frame[minY:maxY, minX:maxX].copy()
* img = cv2.resize(face,(50,50))
* img = np.array([img/255.])
* y\_hat = model1.predict(img)
* y\_hat = np.argmax(y\_hat)
* mask = ('Nomask', 'Mask', 'Incorect')
* predicted\_mask = mask[y\_hat]
* cv2.putText(frame,predicted\_mask, (startX,y),cv2.FONT\_HERSHEY\_SIMPLEX, 0.45, (0, 255, 255), 2)
* img = cv2.resize(face,(48,48))
* img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)
* img = np.array([img/255.])
* emotions = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral')
* y\_hat2 = model2.predict(img)
* y\_hat2 = np.argmax(y\_hat2)
* predicted\_emotion = emotions[y\_hat2]
* # predicted\_emotion = emotions[y\_hat2]
* cv2.rectangle(frame, (startX, startY), (endX, endY), (0, 0, 255), 2)
* if y\_hat == 0:
* #cv2.rectangle(frame, (startX, startY), (endX, endY), (0, 0, 255), 2)
* cv2.putText(frame,predicted\_emotion, (endX-10,endX),cv2.FONT\_HERSHEY\_SIMPLEX, 0.45, (0, 255, 0), 2)
* cv2.imshow("frame", frame)
* key = cv2.waitKey(1)
* if key == 27:
* break
* cap.release()
* cv2.destroyAllWindows()
* Demo kết quả test bằng webcam



# Chương 4. Kết luận

Kết quả đạt được : Chúng em đã ứng dụng thành công mạng neural tích chập trong việc phát hiện người có đeo khẩu trang hay không và nhận diện cảm xúc con người . Tuy nhiên dự án của chúng em vẫn còn 1 vài ưu / nhược điểm sau

## Ưu điểm

1. Tính ứng dụng : Tính ứng dụng cao , có thể linh hoạt sử dụng trong nhiều hoàn cảnh , địa điểm khác nhau .
2. Hiệu suất cao : CNN rất hiệu quả trong việc phân loại xác định hình ảnh , mô hình khẩu trang đã đạt được độ chính xác cao ( >98%) trong việc phân loại người có đeo khẩu trang hay không .
3. Xử lý được dữ liệu thời gian thực

## Nhược điểm

1. Yêu cầu dữ liệu đủ lớn : Để đạt được hiệu suất tốt , mô hình CNN yêu cầu một lượng lớn dữ liệu . Điều này gây khó khăn trong việc phân loại cảm xúc khi mà lượng dữ liệu không được cân đối trong các lớp .
2. Tính chất dữ liệu có nhiều điểm chung : Khi thực hiện xác nhận có đeo khẩu trang , không đeo và đeo sai cách thì đeo sai cách rất khó phân biệt do có nhiều điểm giống với có đeo .
3. Chưa được hoàn toàn chính xác và độ ổn định chưa được cao .

# Hướng phát triển trong tương lai

1. Trong tương lai , chúng em muốn tạo ra một app trên điện thoại để có thể đến được gần hơn với nhiều người .
2. Muốn phát triển lên thành sản phẩm hoàn chỉnh có tính thực tế cao hơn
3. Không chỉ dừng lại ở việc xác định khẩu trang thêm vào đó là cảnh báo bằng âm thanh nếu thấy đối tượng không đeo khẩu trang

# TÀI LIỆU THAM KHẢO

1.https://www.kaggle.com/datasets/spandanpatnaik09/face-mask-detectormask-not-mask-incorrect-mask

2. https://www.kaggle.com/datasets/ananthu017/emotion-detection-fer

3. https://www.ibm.com/topics/convolutional-neural-networks