**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HỒ CHÍ MINH**

**KHOA ĐIỆN - ĐIỆN TỬ**

**BỘ MÔN CÔNG NGHỆ KỸ THUẬT MÁY TÍNH-VIỄN THÔNG**



**BÁO CÁO MÔN HỌC CƠ SỞ VÀ ỨNG DỤNG AI**

**ĐỀ TÀI:**

**NHẬN DẠNG MỘT SỐ CỬ CHỈ BÀN TAY CON NGƯỜI**

**GVHD: Trương Ngọc Sơn**

**Sinh viên thực hiện:**

**Bùi Hữu Thịnh – MSSV:18119195**

**Trương Thành Lợi – MSSV:18119169**

**Tp.Hồ Chí Minh - 6/2021**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HỒ CHÍ MINH**

**KHOA ĐIỆN - ĐIỆN TỬ**

**BỘ MÔN CÔNG NGHỆ KỸ THUẬT MÁY TÍNH-VIỄN THÔNG**



**BÁO CÁO MÔN HỌC CƠ SỞ VÀ ỨNG DỤNG AI**

**ĐỀ TÀI:**

**NHẬN DẠNG MỘT SỐ CỬ CHỈ BÀN TAY CON NGƯỜI**

**GVHD: Trương Ngọc Sơn**

**Sinh viên thực hiện:**

**Bùi Hữu Thịnh – MSSV:18119195**

**Trương Thành Lợi – MSSV:18119169**

**Tp.Hồ Chí Minh - 6/2021**

**TÓM TẮT ĐỀ TÀI**

Chúng em sẽ ra một chương trình dùng để nhận dạng một số cử chỉ bàn tay con người mà chúng em lựa chọn từ trước . Để xây dựng chương trình đó chúng em cần tìm hiểu về ngôn ngữ lập trình Python , thư viện Keras, OpenCV… Tìm hiểu cách sử dụng các công cụ hỗ trợ như Google Colab, Spyder… Các bước để tiến hành sẽ là : tạo tập dữ liệu cho việc train-test, xây dựng và huấn luyện mạng CNN ( ở đây chúng em chọn theo kiến trúc mạng VGG ) , lưu những thông số sau khi huấn luyện và tiến hành chạy kiểm thử, hoàn thiện chương trình.

**LỜI CẢM ƠN**

Sau quá trình tìm hiểu, nghiên cứu và thực hiện đề tài này và hoàn thành đúng tiến độ, chúng em xin chân thành gửi lời cảm ơn sâu sắc nhất đến quý thầy cô, gia đình và những người anh, người em, người bạn đã giúp đỡ cho chúng em.

Chúng em xin gửi lời cảm ơn đến thầy Trương Ngọc Sơn, giảng viên trường Đại Học Sư Phạm Kỹ Thuật Tp.HCM đã hướng dẫn và tạo điều kiện để nhóm có thể hoàn thành được đề tài của mình.

Một lần nữa chúng em xin cảm ơn tất cả mọi người đã luôn giúp đỡ, hỗ trợ tận tình cho chúng em để chúng em hoàn thành tốt đề tài này.

Xin chân thành cảm ơn!

Người thực hiện đề tài:

Bùi Hữu Thịnh Trương Thành Lợi

MỤC LỤC

[Chương 1:Tổng quan 1](#_Toc75081010)

[1.1.Giới thiệu 1](#_Toc75081011)

[1.2.Mục tiêu đề tài 1](#_Toc75081012)

[1.3.Nội dung nghiên cứu 1](#_Toc75081013)

[1.4.Giới hạn đề tài 1](#_Toc75081014)

[1.5.Bố cục 1](#_Toc75081015)

[Chương 2: Cơ Sở Lý Thuyết 3](#_Toc75081016)

[2.1.Giới thiệu về mạng Neural tích chập 3](#_Toc75081017)

[2.1.1. Khái niệm 3](#_Toc75081018)

[2.1.2. Đặc điểm mạng Neural tích chập 3](#_Toc75081019)

[2.2. Giới thiệu ngôn ngữ Python và các thư viện 5](#_Toc75081020)

[2.2.1. Giới thiệu ngôn ngữ Python 5](#_Toc75081021)

[2.2.2. Giới thiệu thư viện Keras 6](#_Toc75081022)

[2.2.3. Giới thiệu thư viện OpenCV 7](#_Toc75081023)

[2.3. Giới thiệu Google Colab và Anaconda 8](#_Toc75081024)

[2.3.1. Google Colab 8](#_Toc75081025)

[2.3.2. Anaconda 8](#_Toc75081026)

[Chương 3: Ý tưởng và thiết kế 10](#_Toc75081027)

[3.1.Giới thiệu ý tưởng 10](#_Toc75081028)

[3.2.Tiến hành xây dựng tập dữ liệu 12](#_Toc75081029)

[3.2.1.Kỹ thuật xử lý ảnh 12](#_Toc75081030)

[3.2.2.Phân chia tập dữ liệu 14](#_Toc75081031)

[3.3. Xây dựng mô hình mạng CNN 15](#_Toc75081032)

[3.3.1.Giới thiệu mạng VGG-16 15](#_Toc75081033)

[3.3.2.Xây dựng mô hình CNN cho chương trình 16](#_Toc75081034)

[Chương 4:Thực thi và kết quả 18](#_Toc75081035)

[4.1.Huấn luyện mô hình mạng CNN 18](#_Toc75081036)

[4.2. Tiến hành chạy hệ thống sau khi mô hình đã được huấn luyện 21](#_Toc75081037)

[Chương 5: Kết luận và hướng phát triển 24](#_Toc75081038)

[5.1. Kết luận 24](#_Toc75081039)

[5.2. Hướng phát triển 24](#_Toc75081040)

[TÀI LIỆU THAM KHẢO 25](#_Toc75081041)

[PHỤ LỤC 26](#_Toc75081042)

**DANH MỤC HÌNH**

[Hình 2. 1.Minh họa một ví dụ sử dụng CNN để phân loại phương tiện 4](#_Toc75081043)

[Hình 2. 2.Tham số được chia sẻ cho 5 neural ở 5 layer khác nhau 4](#_Toc75081044)

[Hình 2. 3.Các ngôn ngữ phổ biến trong năm 2016 5](#_Toc75081045)

[Hình 2. 4.Các Framework sử dụng cho Deep Learning 6](#_Toc75081046)

[Hình 2. 5.Logo của OpenCV 7](#_Toc75081047)

[Hình 2. 6.Cấu hình phần cứng Google Colab cung cấp 8](#_Toc75081048)

[Hình 3. 1.Label 0 - Buffalo 10](#_Toc75081050)

[Hình 3. 2.Label 1 - Fist 10](#_Toc75081051)

[Hình 3. 3.Label 2 - Five 10](#_Toc75081052)

[Hình 3. 4.Label 3 - Okay 11](#_Toc75081053)

[Hình 3. 5.Label 5 - Hi 11](#_Toc75081054)

[Hình 3. 6.Các bước để hoàn thiện chương trình 12](#_Toc75081055)

[Hình 3. 7.Giải thuật Background Subtraction 13](#_Toc75081056)

[Hình 3. 8.Tập dữ liệu train 14](#_Toc75081057)

[Hình 3. 9.Tập dữ liệu validation 14](#_Toc75081058)

[Hình 3. 10.Tập dữ liệu test 15](#_Toc75081059)

[Hình 3. 11.Mô hình VGG-16 16](#_Toc75081060)

[Hình 3. 12.Mô hình CNN được chúng em xây dựng. 16](#_Toc75081061)

[Hình 3. 13.Triển khai mô hình trên Google Colab. 17](#_Toc75081062)

[Hình 4. 1.Liên kết Drive với Colab 18](#_Toc75081063)

[Hình 4. 2.Import thư viện cần thiệt vào Colab 18](#_Toc75081064)

[Hình 4. 3.Upload dữ liệu từ Drive và tiền xử lý dữ liệu 19](#_Toc75081065)

[Hình 4. 4.Mô hình CNN và đường dẫn xuất file sau khi train 20](#_Toc75081066)

[Hình 4. 5.Kết quả huấn luyện 20](#_Toc75081067)

[Hình 4. 6. File .h5 đã được sinh ra 20](#_Toc75081068)

[Hình 4. 7.Load mô hình dự đoán trong file .h5 21](#_Toc75081069)

[Hình 4. 8.Cửa số chương trình 21](#_Toc75081070)

[Hình 4. 9.Nhận dạng thành công Label 0 21](#_Toc75081071)

[Hình 4. 10.Nhận dạng thành công Label 1 22](#_Toc75081072)

[Hình 4. 11.Nhận dạng thành công Label 2 22](#_Toc75081073)

[Hình 4. 12.Nhận dạng thành công Label 3 22](#_Toc75081074)

[Hình 4. 13.Nhận dạng thành công Label 4 23](#_Toc75081075)

**DANH MỤC CÁC TỪ VIẾT TẮT**

AI – Artificial intelligence

CNN – Convolution Neural Network

CNTK – Computational Network Toolkit

IDE – Integrated Development Environment

MS-DOS – Microsoft Disk Operating System

MAC OS – Macintosh Operating System

# Chương 1:Tổng quan

## 1.1.Giới thiệu

Trong thời buổi công nghệ ngày càng phát triển thì những hệ thống trí thông minh nhân tạo được sinh ra một cách nhanh chóng giải quyết những bài toán: nhận dạng khuôn mặt, nhận dạng giọng nói …Với mong muốn tìm hiểu về những bài toán nhận dạng nói riêng và hệ thống trí thông minh nhận tạo nói chung. Nên chúng em quyết định chọn đề tài “Nhận dạng cử chỉ bàn tay con người”

Ở đề tài này chúng em sẽ xây dựng một một hình CNN và tiến hành huấn luyện nó để nó có thể nhận dạng được một số cử chỉ đơn giản như okay , chào…

## 1.2.Mục tiêu đề tài

-Nghiên cứu ngôn ngữ Python , các thư viện Keras, OpenCV.

-Thực thi việc xử lý ảnh ở mức cơ bản ( tách nền , phân biệt ảnh )

-Triển khai xây dựng và huấn luyện một mô hình CNN.

-Chạy thành công chương trình và nhận dạng thành công một số cử chỉ đơn giản mà chúng em chọn trong quá trình huấn luyện mô hình CNN.

## 1.3.Nội dung nghiên cứu

-Tìm hiểu mô hình mạng CNN.

-Tìm hiểu việc xử lý ảnh cơ bản.

-Tìm hiểu ngôn ngữ Python, thư viện Keras, OpenCV…

-Xây dựng, huấn luyện mô hình CNN , tạo chương trình nhận dạng cử chỉ bàn tay.

## 1.4.Giới hạn đề tài

-Xử lý ảnh ở mức cơ bản ( đổi ảnh màu thành ảnh xám, thay đổi kích thước ảnh…).

-Xây dựng mô hình CNN bằng cách sử dụng thư viện Keras.

- Huấn luyện mô hình trên Google Colab.

- Nhận dạng một vài cử chỉ bàn tay mà nhóm đã chọn trước.

## 1.5.Bố cục

Chương 1 :Tổng quan

Chương 2: Cơ sở lý thuyết

Chương 3: Ý tưởng và thiết kế

Chương 4: Thực thi và kết quả

Chương 5: Kết luận và hướng phát triển

# Chương 2: Cơ Sở Lý Thuyết

## 2.1.Giới thiệu về mạng Neural tích chập

### 2.1.1. Khái niệm

Neural tích chập, hay còn được gọi là CNN – viết tắt của “Convolutional Neural Network” – là một trong những mô hình Deep Learning phổ biến nhất và tiên tiến nhất hiện nay. Hầu hết các hệ thống nhận diện và xử lý ảnh hiện nay đều sử dụng mạng nơ ron tích chập vì tốc độ xử lý nhanh và độ chính xác cao.

Kiến trúc gốc của mô hình CNN được phát minh bởi một nhà khoa học máy tính người Nhật vào năm 1980. Sau đó vào năm 1998, Yann LeCun giới thiệu mô hình CNN đầu tiên (LeNet-5) và tiến hành bài toán nhận dạng ký tự viết tay. Mặc dù mô hình này hoạt động rất hiệu quả, tuy nhiên CNN cần rất nhiều dữ liệu và tài nguyên tính toán để hoạt động cho các hình ảnh lớn nên kỹ thuật này chỉ được áp dụng cho những hình ảnh có độ phân giải thấp. Mãi đến năm 2012, khi có sự xuất hiện của các tập dữ liệu lớn và nguồn tài nguyên máy tính lớn hơn rất nhiều so với trước đây, đã cho phép các nhà khoa học nghiên cứu ra AlexNet – đánh dấu bước phát triển quan trọng đầu tiên của Deep Learning.

### 2.1.2. Đặc điểm mạng Neural tích chập

a. Tính bất biến

Đối với một số loại dữ liệu, đặc biệt là dữ liệu ở dạng hình ảnh, đã gây khó khăn cho các mạng neural truyền thẳng chỉ nhận đầu vào là vector 1 chiều. Để có thể xử lí và tiến hành huấn luyện, chúng ta cần phải chuyển đổi được hình ảnh về dưới dạng vector, điều này thường gây ra sự mất mát nhiều thông tin so với dữ liệu gốc ban đầu.

Do đó với việc sử dụng các lớp tích chập và lớp gộp, CNN sẽ xử lý được các input ảnh 3 chiều gồm chiều cao, chiều rộng và chiều sâu, một sự cải tiến so với mạng neural thế hệ cũ.

b. Tính kết nối cục bộ

Một ưu điểm mà mạng CNN có được so với mạng neural thông thường đó là tính kết nối cục bộ. Các layer trong mạng CNN liên kết với nhau qua cơ chế convolution – layer tiếp theo là kết quả convolution từ layer phía trước, khác với việc liên kết trực tiếp qua trọng số w của neural truyền thống.



Hình 2. 1.Minh họa một ví dụ sử dụng CNN để phân loại phương tiện

Nhờ tính kết nối cục bộ, mô hình CNN thường có độ chính xác cao, và thông tin được phân loại theo nhiều mức độ giống như việc con người nhận biết các vật thể trong tự nhiên: ta phân biệt một con chó với một con mèo nhờ vào các đặc trưng từ mức độ thấp (có 4 chân, có đuôi…) đến mức độ cao (dáng đi, hình thể, màu lông…).

c. Tính chia sẻ tham số

Trong xử lý hình ảnh, thông tin của hình ảnh là các điểm ảnh (pixel). Nếu sử dụng mạng được kết nối đầy đủ, chúng ta sẽ tạo ra rất nhiều tham số. Ví dụ, một hình ảnh RGB có kích thước 512x512 pixel sẽ có 786432 (= 512 x 512 x 3) tham số đầu vào. Vì vậy, nếu chúng ta kết nối đầy đủ một mạng nơ ron thông thường, toàn bộ kiến trúc mạng sẽ cần tính toán hàng triệu nơ ron. Số lượng lớn các nơ ron làm cho toàn bộ quá trình học rất chậm và dẫn đến quá tải so với khả năng tính toán của máy tính hiện tại.



Hình 2. 2.Tham số được chia sẻ cho 5 neural ở 5 layer khác nhau

Với cấu trúc CNN, do sử dụng tích chập nên các tham số có thể được chia sẻ và sử dụng lại trong các layer, giảm thiểu đáng kể số lượng tham số và tăng tốc độ huấn luyện, nhưng không gây ra mất mát dữ liệu và thông tin.

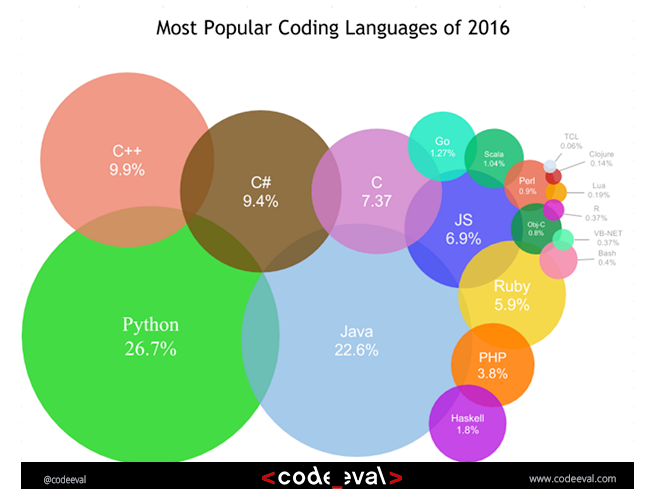
## 2.2. Giới thiệu ngôn ngữ Python và các thư viện

### 2.2.1. Giới thiệu ngôn ngữ Python

Python được Guido van Rossum tạo ra cuối năm 1990. Python hiện được duy trì bởi một nhóm phát triển cốt lõi tại viện nghiên cứu quốc gia về toán học và khoa học máy tính ở Hà Lan. Guido van Rossum vẫn giữ một vai trò quan trọng trong việc chỉ đạo tiến trình của nó.

Python được phát triển để chạy trên nền Unix. Nhưng theo thời gian, nó đã "bành trướng" sang mọi hệ điều hành từ MS-DOS đến MAC OS, OS/2, Windows, Linux và một số điều hành khác thuộc họ Unix.

Python là một trong những ngôn ngữ nổi tiếng và ứng dụng tốt trong lĩnh vực AI , Machine Learning, Deep Learning…



Hình 2. 3.Các ngôn ngữ phổ biến trong năm 2016

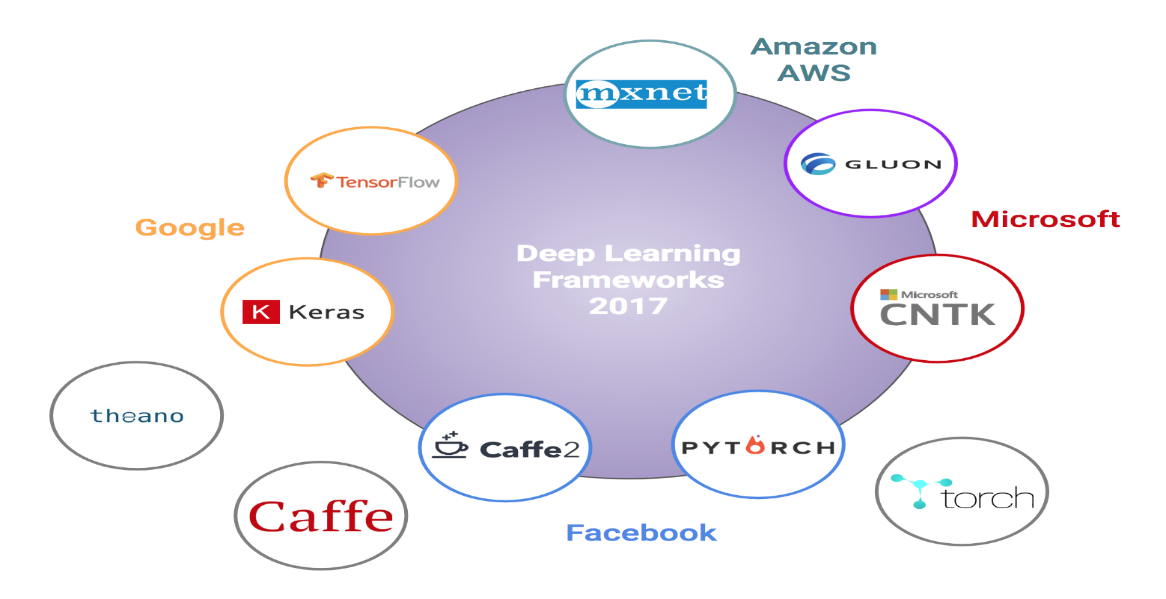
Đặc điểm của Python có thể kể đến:

* Ngữ pháp đơn giản, dễ đọc.
* Vừa hướng thủ tục (procedural-oriented), vừa hướng đối tượng (object-oriented)
* Hỗ trợ module và hỗ trợ gói (package)
* Xử lý lỗi bằng ngoại lệ (Exception)
* Kiểu dữ liệu động ở mức cao.
* Có các bộ thư viện chuẩn và các module ngoài, đáp ứng tất cả các nhu cầu lập trình.
* Có khả năng tương tác với các module khác viết trên C/C++ (Hoặc Java cho Jython, hoặc .Net cho IronPython).
* Có thể nhúng vào ứng dụng như một giao tiếp kịch bản (scripting interface).

### 2.2.2. Giới thiệu thư viện Keras

Keras là một thư viện được phát triển vào năm 2015 Đây là một thư viện mạnh mẽ và dễ sử dụng nhất của Python để tạo ra các mô hình học sâu.

Keras chạy trên các thư viện máy mã nguồn mở như TensorFlow, Theano hoặc Bộ công cụ nhận thức (CNTK). Theano là một thư viện dùng cho các tác vụ tính toán số nhanh còn TensorFlow là thư viện toán học nổi bật nhất được sử dụng để tạo mạng nơ-ron và mô hình học sâu. TensorFlow rất linh hoạt và mạnh về tính toán phân tán. CNTK là khung học sâu được phát triển bởi Microsoft, nó sử dụng các thư viện như Python, C #, C ++ hoặc các bộ công cụ học máy độc lập. Theano và TensorFlow là những thư viện rất mạnh nhưng khó hiểu trong việc tạo mạng nơ-ron. Chính vì vậy, Keras dựa trên cấu trúc tối thiểu, cung cấp một cách dễ dàng và dễ dàng để tạo các mô hình học sâu dựa trên TensorFlow hoặc Theano. Keras được thiết kế để xác định nhanh các mô hình học sâu.



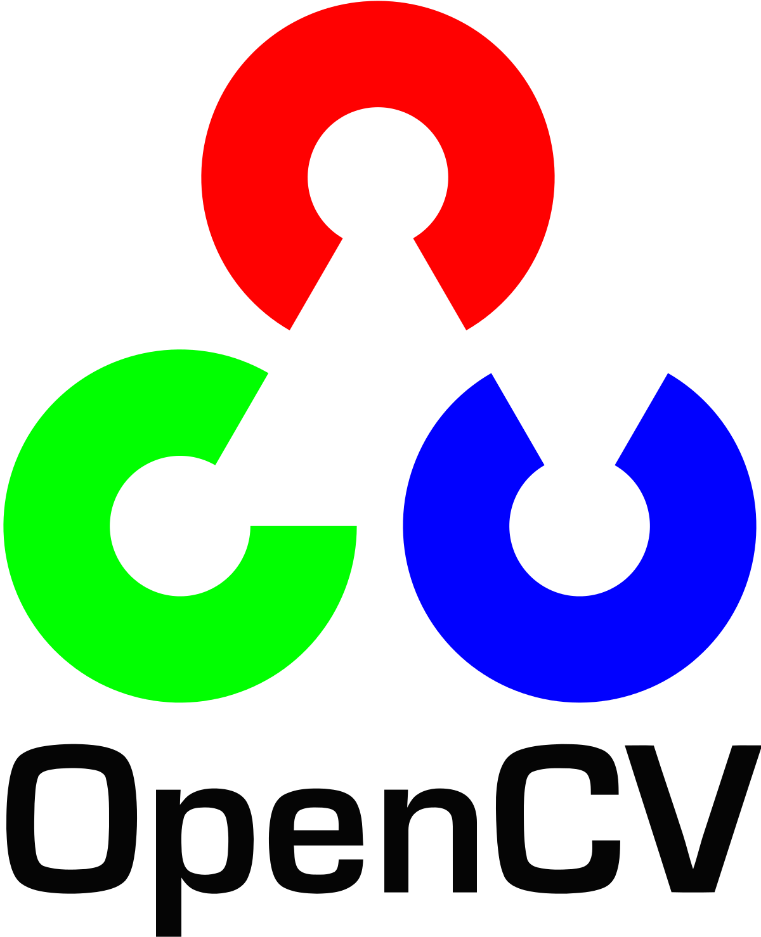
Hình 2. 4.Các Framework sử dụng cho Deep Learning

Lợi ích khi chúng ta dùng Keras được nêu trên trang chủ của họ, chúng ta sẽ nêu vài điều đặc trưng:

* 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

### 2.2.3. Giới thiệu thư viện OpenCV

OpenCV được Intel phát triển từ năm 1999 bởi Gary Bradsky. OpenCV viết tắt cho Open Source Computer Vision Library. OpenCV là thư viện nguồn mở hàng đầu cho Computer Vision và Machine Learning, và hiện có thêm tính năng tăng tốc GPU cho các hoạt động theo real-time.



Hình 2. 5.Logo của OpenCV

Thư viện này có trên các giao diện C++, C, Python và Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android. OpenCV được thiết kế để hỗ trợ hiệu quả về tính toán và chuyên dùng cho các ứng dụng real-time (thời gian thực).

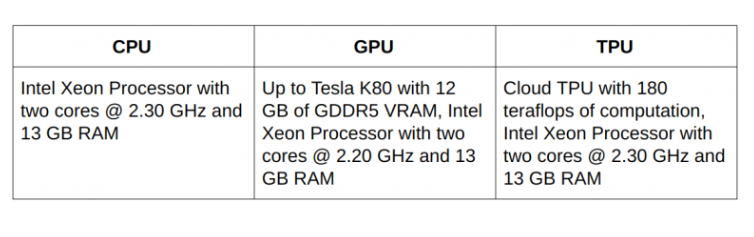
OpenCV được sử dụng cho đa dạng nhiều mục đích và ứng dụng khác nhau như là:

* Phân tích hình ảnh y học
* Robot và xe tự hành
* Kiểm tra và giám sát tự động
* Phục hồi hình ảnh/video
* Thực tế ảo
* …

## 2.3. Giới thiệu Google Colab và Anaconda

### 2.3.1. Google Colab

Google Colab, là một sản phẩm từ Google Research, là môi trường cho phép chạy các dòng code python thông qua trình duyệt, đặc biệt phù hợp với lĩnh vực phân tích dữ liệu, trí tuệ nhân tạo và cho mục đích giáo dục. Colab không cần yêu cầu cài đặt hay cấu hình máy tính, mọi thứ có thể chạy thông qua trình duyệt, bạn có thể sử dụng tài nguyên máy tính từ CPU tốc độ cao và cả GPUs và cả TPUs đều được cung cấp cho bạn. Vì là dịch vụ miễn phí, nên Colab sẽ có những thứ tự ưu tiên trong việc sử dụng tài nguyên hệ thống, cũng như giới hạn thời gian sử dụng, thời gian sử dụng tối đa lên tới 12 giờ.



Hình 2. 6.Cấu hình phần cứng Google Colab cung cấp

### 2.3.2. Anaconda

Anaconda là một nền tảng phân phối miễn phí của ngôn ngữ lập trình Python cho tính toán khoa học nhằm mục đích đơn giản hóa việc quản lý và triển khai gói. Nó hỗ trợ trên các nền tảng Windows, MacOS và Linux.

Anaconda chứa tất cả các gói (công cụ) phổ biến nhất mà một nhà khoa học dữ liệu cần. Các package trong Anaconda được quản lý bởi trình quản lý riêng của nền tảng này là conda. Ta thường dùng conda để tạo môi trường cô lập các dự án của mình, nhằm sử dụng các phiên bản Python khác nhau hoặc các phiên bản package khác nhau, cũng như dùng nó để cài đặt, gỡ cài đặt và cập nhật các package riêng trong từng dự án.

Lợi ích khi sử dụng Anaconda ta có thể kể tới:

* Anaconda không chỉ có Python: Ngoài ngôn ngữ Python ra, Anaconda cung cấp 600 package mở rộng đi kèm, đó là các package cơ bản cho các tác vụ xử lý khoa học dữ liệu như NumPy, Pandas,.... Vì vậy chúng ta không cần phải dùng terminal để tải chúng về nữa, tiết kiếm được thời gian.
* Anaconda hoạt động trên mọi nền tảng (Windows, Linux, MacOS)
* Anaconda giúp quản lý gói và môi trường ảo dễ dàng hơn: đây là một trong những ưu điểm lớn nhất.
* Anaconda cài đặt mà không có quyền của quản trị: Bạn có thể cài Anaconda từ hầu hết máy tính mà không cần lo tới quyền admin, như là sử dụng máy tính ở thư viện trường,.... Điều này khiến cho Anaconda có thể cài đặt ở bất cứ nơi đâu, miễn là có máy tính.
* Cung cấp đa dạng các IDE và Jupyter đi kèm

Trong các công cụ hỗ trợ cho Python mà Anaconda, nhóm sử dụng Spyther IDE cho mục đích viết chương trình và chạy mô phỏng. Giới thiệu một chút về Spyder thì đây là một IDE đa nền tảng mã nguồn mở. Python Spyder IDE được viết hoàn toàn bằng Python. Nó được thiết kế bởi các nhà khoa học và dành riêng cho các nhà khoa học, nhà phân tích dữ liệu và kỹ sư. Nó còn được gọi là IDE phát triển Python khoa học và có một lượng lớn các tính năng đáng chú ý sau đây:

* Đánh dấu cú pháp có thể tùy chỉnh
* Tính khả dụng của các điểm ngắt (điểm ngắt có điều kiện và gỡ lỗi)
* Thực thi tương tác cho phép bạn chạy dòng, tệp, ô, v.v.
* Chạy các cấu hình để lựa chọn thư mục làm việc, tùy chọn dòng lệnh, bảng điều khiển hiện tại / chuyên dụng / bên ngoài, v.v.
* Có thể xóa biến tự động (hoặc nhập gỡ lỗi)
* Điều hướng qua các ô, chức năng, khối, v.v. có thể đạt được thông qua Outline Explorer
* Nó cung cấp khả năng xem xét nội dung mã theo thời gian thực (Khả năng kiểm tra các chức năng, từ khóa và lớp là gì, chúng đang làm gì và chúng chứa thông tin gì)
* Tự động chèn dấu hai chấm sau if, while, v.v.
* Hỗ trợ tất cả các lệnh ma thuật IPython
* Hiển thị nội tuyến cho đồ họa được tạo bằng Matplotlib
* Cũng cung cấp các tính năng như trợ giúp, trình khám phá tệp, tìm tệp,…

# Chương 3: Ý tưởng và thiết kế

## 3.1.Giới thiệu ý tưởng

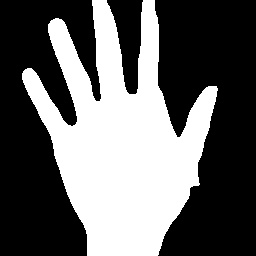
Chúng em sẽ tạo một chương trình dùng để nhận dạng một số cử chỉ của bàn tay.Ở đây chúng em chọn ra 5 cử chỉ .



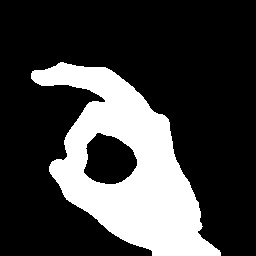
Hình 3. 1.Label 0 - Buffalo



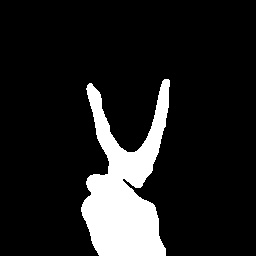
Hình 3. 2.Label 1 - Fist



Hình 3. 3.Label 2 - Five

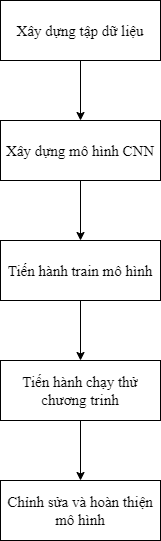


Hình 3. 4.Label 3 - Okay



Hình 3. 5.Label 5 - Hi

Chúng em sẽ sử dụng mạng CNN để tiến hành huấn luyện cho nó có thể nhận dạng được các cử chỉ đã đề ra ở trên. Toàn chương trình chúng em sẽ xây dựng theo các bước sau:



Hình 3. 6.Các bước để hoàn thiện chương trình

## 3.2.Tiến hành xây dựng tập dữ liệu

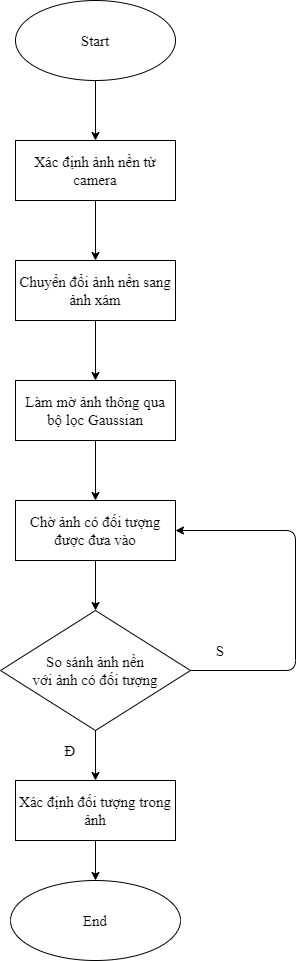
Chúng em sẽ xây dựng một tập dữ liệu ảnh trắng đen ( nền đen và tay trắng), để có thể dễ dàng nhận dạng. Kích thước của một ảnh là 256x256. Một vấn đề là ảnh nhận được từ WebCam ( Camera ) là ảnh màu . Chúng em phải chuyển đổi nó thành ảnh xám . Và dùng kỹ thuật xử lý ảnh để tách background .

### 3.2.1.Kỹ thuật xử lý ảnh

Chúng em sử dụng kỹ thuật Background Subtraction để xác định được các cử chỉ của tay. Giải thuật Background Subtraction là giải thuật mà ta sẽ cần có 2 ảnh, một ảnh nền và một ảnh có đối tượng, ta lấy 2 ảnh đó trừ nhau, điều này giúp chúng ta loại bỏ nền và giữ lại được đối tượng có trên ảnh.

Giải thuật này phù hợp cho các trường hợp xử lý ảnh khi đã biết trước, hoặc tính toán được ảnh nền từ camera. Điều này hoàn toàn phù hợp với mục tiêu của chương trình chúng em nên chúng em cảm thấy nó là một giải thuật phù hợp để chúng em sử dụng.

Triển khai giải thuật vào chương trình chúng em như sau :



Hình 3. 7.Giải thuật Background Subtraction

\*Giải thích lưu đồ giải thuật Background Subtration:

-Bắt đầu thì sẽ đợi hệ thống lấy ảnh từ camera và xác định đó là ảnh nền

-Sau đó ảnh sẽ chuyển sang thành ảnh xám và làm mờ thông qua bộ lọc Gausian

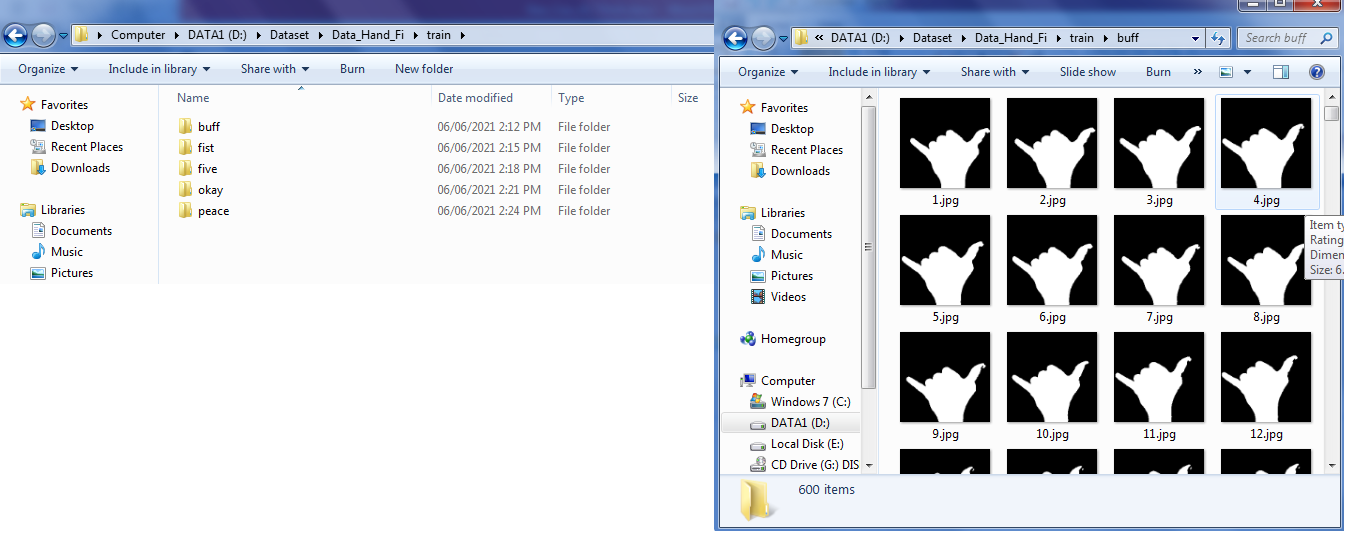
-Những ảnh sau đó mà camera nhận được sẽ được mang đi so sánh với giá trị ảnh nền ban đầu ( như một phép trừ ) .

-Những giá trị sai khác với ảnh nền sẽ cho giá trị sai lệnh từ đó sẽ xác định được vật thể mới xuất hiện trong ảnh .

### 3.2.2.Phân chia tập dữ liệu

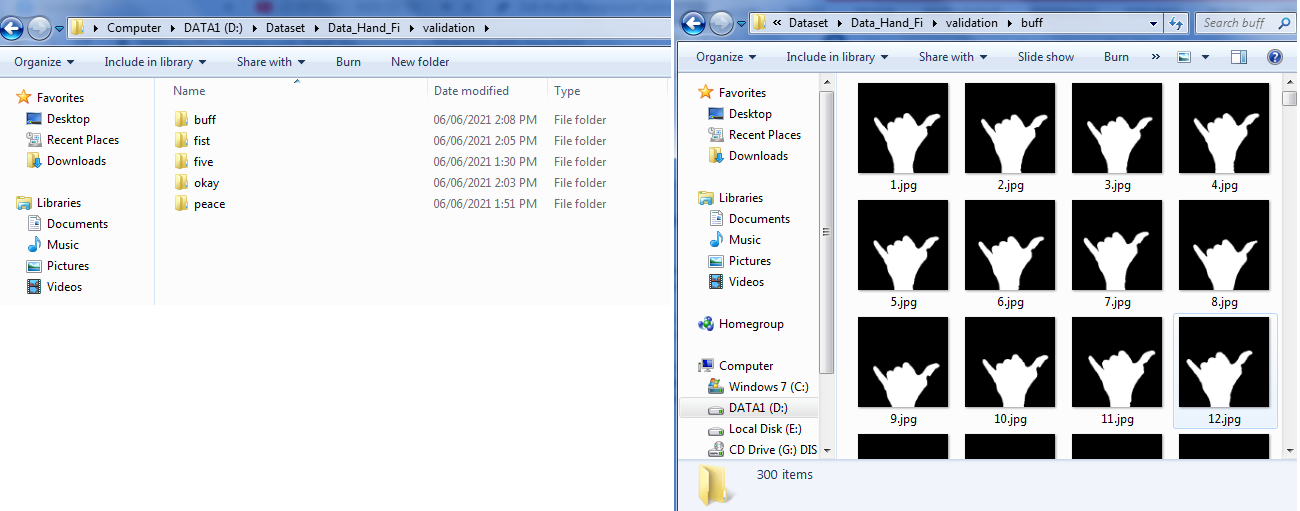
Chúng em muốn có một tập dữ liệu tốt để giúp cho việc huấn luyện mô hình đạt kết quả tốt nhất. Chúng em sẽ tiến hành xây dựng 3 tập dữ liệu bao gồm: train,validation và test :

-Train : Chúng em xây dựng bao gồm 600 ảnh cho mỗi cử chỉ ( gồm 5 cử chỉ) tổng cộng cả tập train là 3000 ảnh. Dùng để huấn luyện cho mô hình CNN.



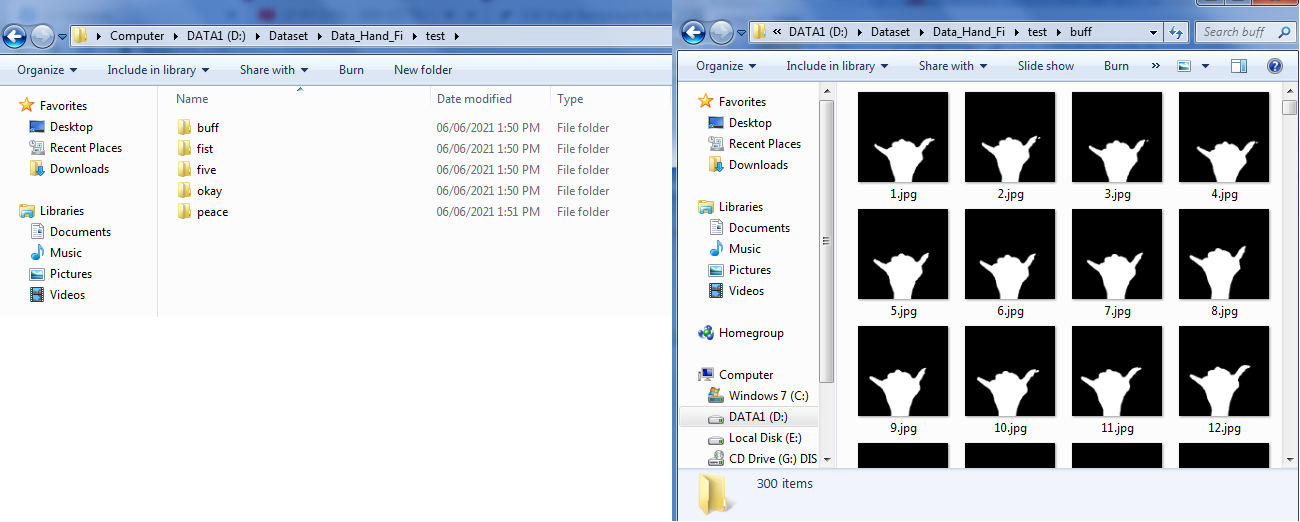
Hình 3. 8.Tập dữ liệu train

-Validation: Tập dữ liệu này dùng để xác thực mô hình trong khi huấn luyện, vì trong quá trình huấn luyện chúng em muốn quan sát xem mức cải thiện độ chính xác của mô hình qua từng vòng huấn luyện như thế nào.Nên chúng em cần tạo tập dữ liệu này để có thể đảm nhận việc đó. Tập dữ liệu này chúng em 1500 ảnh ( 300 ảnh cho mỗi loại cử chỉ ).



Hình 3. 9.Tập dữ liệu validation

-Test: Tập dữ liệu này dùng để thực hiện kiểm tra mô hình sau khi đã huấn luyện xong.Chúng em xây dựng tập dữ liệu này gồm 1500 ảnh (300 cho mỗi loại). Trong trường hợp chúng ta muốn dùng ảnh trực tiếp được đổi từ camera để kiếm tra thì chúng ta có thể dùng tập dữ liệu này để đảm nhận luôn nhiệm vụ của tập dữ liệu validation . Lúc này thì chúng ta chỉ cần sử dụng 2 tập dữ liệu Test và Train.

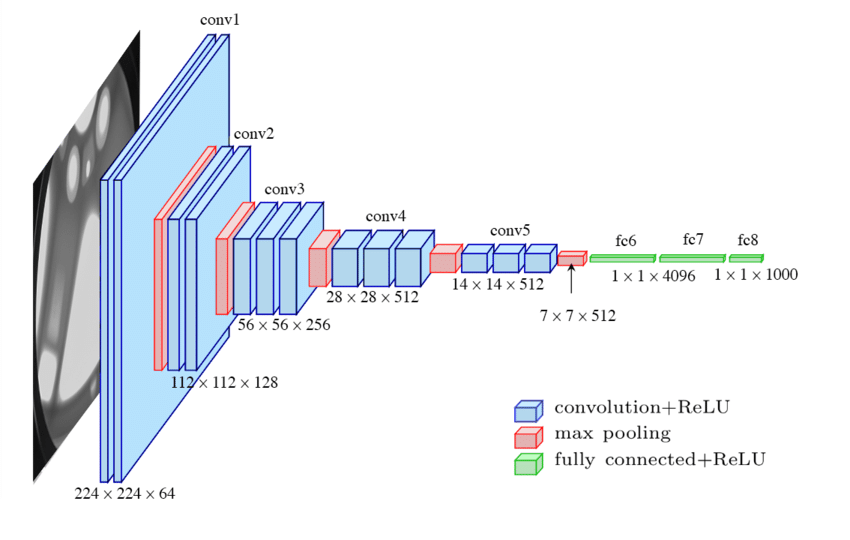


Hình 3. 10.Tập dữ liệu test

## 3.3. Xây dựng mô hình mạng CNN

### 3.3.1.Giới thiệu mạng VGG-16

Mạng VGG-16 là một mạng CNN với 16 lớp , được phát triển năm 2014. Là một biến thể sâu hơn nhưng lại đơn giản hơn so với kiến trúc convolution thường thấy ở kiến trúc CNN, nó sâu hơn nhưng lại đơn giản hơn LeNet, AlexNet… thu gọn về kích thước nhưng số lượng lại lớn hơn, sâu hơn.

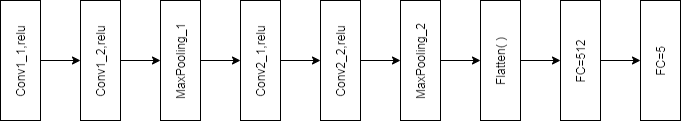


Hình 3. 11.Mô hình VGG-16

Mô hình sẽ gồm 13 lớp tích chập và 3 lớp ở tầng full-connect, tổng cộng là 16 nên được gọi là VGG-16 . Có một phiên bản nữa là VGG-19 được phát triển lúc sau.

### 3.3.2.Xây dựng mô hình CNN cho chương trình

Chúng em sẽ xây dựng mô hình CNN của riêng mình dựa trên mạng VGG-16 , chúng em sẽ giảm một số lớp so với mô hình VGG-16 ban đầu. Điều này giúp cho việc train mô hình có tốc độ nhanh hơn, nhằm giảm áp lực cho phần cứng và thời gian huấn luyện , vì chương trình nhận dạng của chúng em chỉ có 5 loại ảnh,và là ảnh xám, nên cũng không cần một mô hình quá sâu, quá phức tạp cho nó.



Hình 3. 12.Mô hình CNN được chúng em xây dựng.

Nhìn vào mô hình của chúng em có thể dễ dàng thấy rằng sẽ có 2 tầng tích chập và 1 tầng full-connect . Ở tầng tích chập thứ 1 và thứ 2 sẽ có cấu trúc như nhau đó là 2 lớp tích chập và 1 lớp MaxPooling . Tầng full-connect sẽ bao gồm 1 lớp Flatten, 1 lớp ẩn 512 và Output là 5 tương ứng với 5 loại ảnh cần phân biệt ( 5 cử chỉ ). Sau mỗi lớp tích chập chúng em sử dụng hàm kích hoạt là ReLU , ở lớp Output chúng em sử dụng hàm Softmax và có sử dụng kỹ thuật DropOut khi triển khai mô hình.

Sau đây là khi chúng em triển khai mô hình trên Google Colab bằng thư viện Keras:



Hình 3. 13.Triển khai mô hình trên Google Colab.

# Chương 4:Thực thi và kết quả

## 4.1.Huấn luyện mô hình mạng CNN

Do hạn chế về phần cứng để huấn luyện mô hình nên chúng em đã quyết định là đưa mô hình lên Google Colab để có thể mượn “sức mạnh” của Google để giúp việc huấn luyện nhanh chóng hơn.

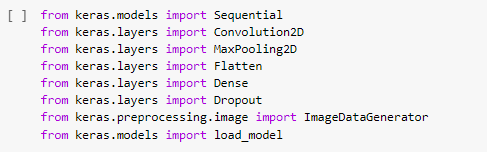
Và để huấn luyện mô hình trên Google Colab chúng ta tiến hành theo các bước sau:

-Bước 1: Tiến ảnh upload tập dữ liệu lên Google Drive và liên kết drive với Google Colab.



Hình 4. 1.Liên kết Drive với Colab

-Bước 2: Tiến hành import các thư viện cần thiết trên Colab



Hình 4. 2.Import thư viện cần thiệt vào Colab

-Bước 3: Tiền xử lý. Ở bước này ta tiến hành tải dữ liệu từ Drive và xử lý tập dữ liệu trước khi bắt đầu huấn luyện mô hình. Resize về đúng kích thước mà mô hình CNN chúng ta cần, gán label cho các ảnh trong tập dữ liệu . Ở đây chúng em dùng tập dữ liệu Test để đảm nhận luôn vai trò của tập dữ liệu Validation, vì chúng em sẽ tiến hành kiểm tra độ chính xác mô hình thông qua hình ảnh trực tiếp từ camera.



Hình 4. 3.Upload dữ liệu từ Drive và tiền xử lý dữ liệu

-Bước 4: Xây dựng mô hình CNN ( upload đoạn code mô hình CNN mà chúng ta đã thiết kế ở hình 3.13 ) và chọn đường dẫn để xuất file lưu thông số sau khi train.

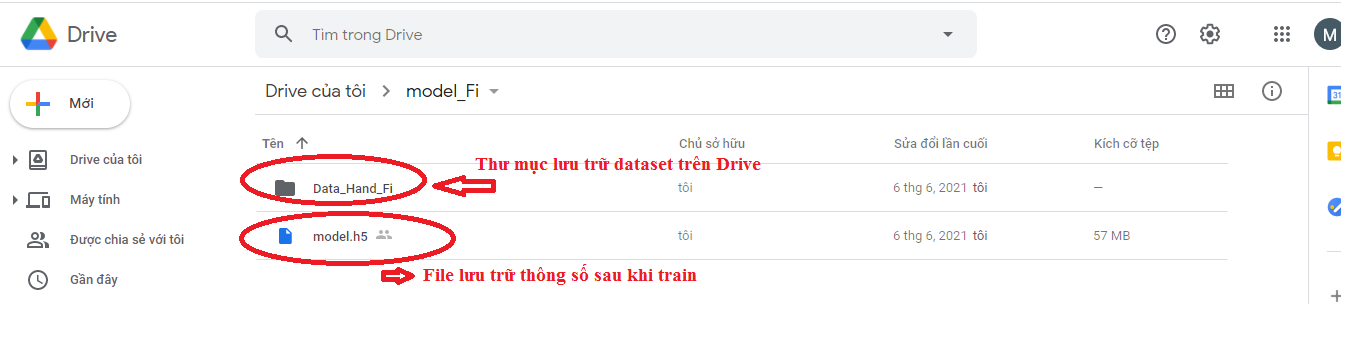


Hình 4. 4.Mô hình CNN và đường dẫn xuất file sau khi train

-Bước 5: Sau khi ta huấn luyện thì Colab sẽ sinh ra 1 file .h5 để lưu các thông số của mô hình sau khi đã huấn luyện xong. Việc của chúng ta là tải file này về và chuẩn bị cho việc chạy chương trình nhận dạng.



Hình 4. 5.Kết quả huấn luyện



Hình 4. 6. File .h5 đã được sinh ra

## 4.2. Tiến hành chạy hệ thống sau khi mô hình đã được huấn luyện

Đầu tiên chúng ta định dạng địa chỉ dẫn đến file .h5 đã tạo ở phần 4.7



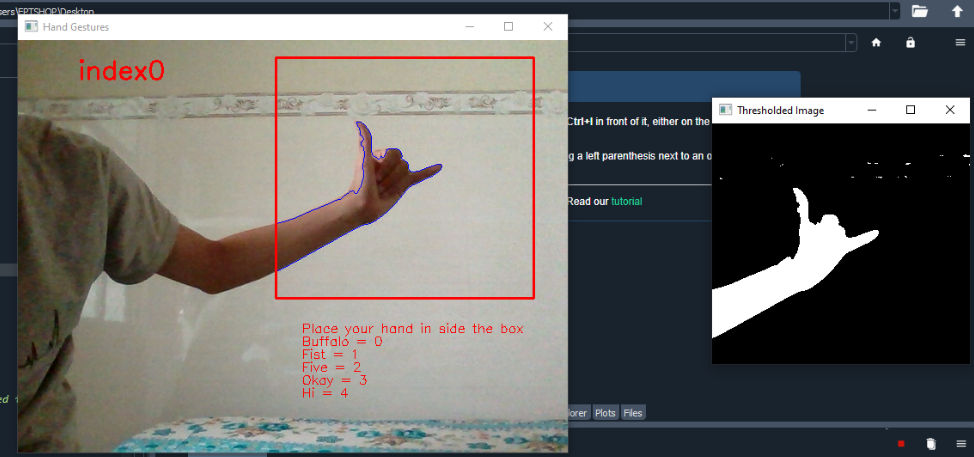
Hình 4. 7.Load mô hình dự đoán trong file .h5

Sau đó bấm F5 trên bàn phím để chạy chương trình, lúc này sẽ mở ra cửa sổ đầu tiên, chúng ta đợi một lát để chương trình xử lí Background, sau khi xử lí xong sẽ mở ra một cửa sổ thứ hai, lúc này chúng ta sẽ đưa tay vào Camera (WebCam) để chương trình dự đoán được cử chỉ tay của chúng ta.



Hình 4. 8.Cửa số chương trình

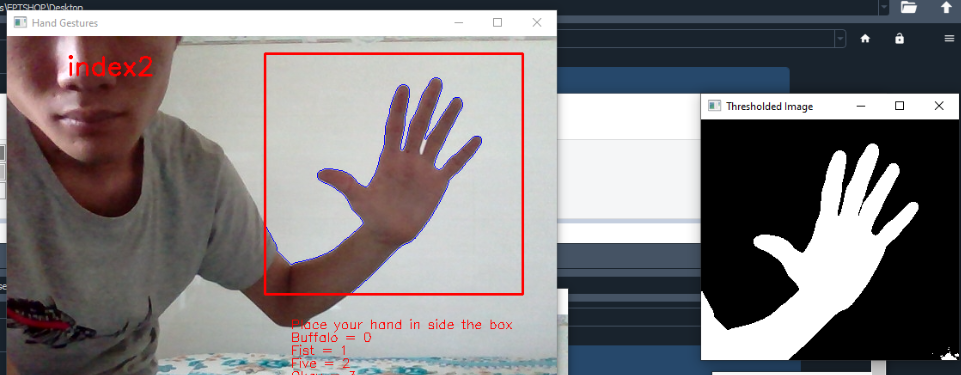
Đây là một số hình ảnh kết quả mà nhóm thu được:



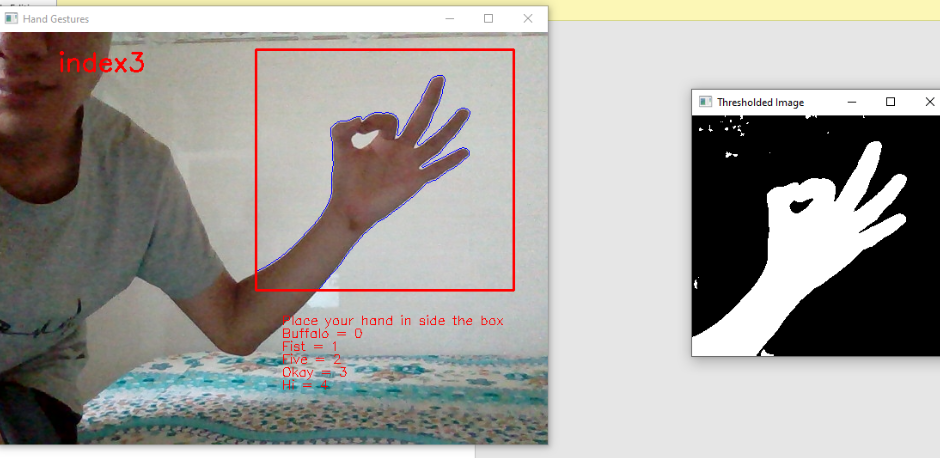
Hình 4. 9.Nhận dạng thành công Label 0



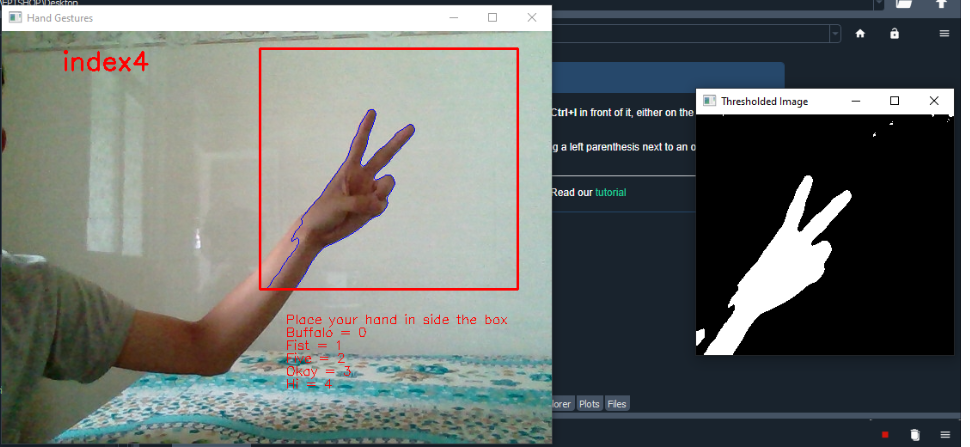
Hình 4. 10.Nhận dạng thành công Label 1



Hình 4. 11.Nhận dạng thành công Label 2



Hình 4. 12.Nhận dạng thành công Label 3



Hình 4. 13.Nhận dạng thành công Label 4

# Chương 5: Kết luận và hướng phát triển

## 5.1. Kết luận

- Nhóm đã hiểu được lý thuyết cần thiết và cách để thiết kế hệ thống cho việc nhận dạng cử chỉ tay.

- Train mô hình mạng CNN với độ chính xác ổn .

- Test thực tế bằng hình ảnh thông qua WebCam nhận dạng chính xác được các trường hợp cử chỉ đã được học.

- Chương trình còn có sự hạn chế về nhiễu do background gây ra khiến cho việc dự đoán trở nên sai.

## 5.2. Hướng phát triển

- Nhóm sẽ tiếp tục phát huy mô hình xử lí với độ chính xác cao hơn hiện tại, xử lý tốt phần background.

-Phát triển để mô hình có thể tự xác định bàn tay trong video thay vì phải đưa vào khung như hiện tại.

- Áp dụng hệ thống vào hệ thống nhúng để ứng dụng thực tiễn.

- Phát triển nhiều cử chỉ tay khác so với năm cử chỉ hiện có.

# TÀI LIỆU THAM KHẢO

[1]: “TRÍ TUỆ NHÂN TẠO CƠ SỞ VÀ ỨNG DỤNG” , tác giả Trương Ngọc Sơn, Trường ĐH SPKT TP HCM.

[2]: Trang web dịch từ sách “Dive into Deep Learning” <https://d2l.aivivn.com/index.html>

[3]: Giới thiệu ngôn ngữ Python , <http://tutorials.aiclub.cs.uit.edu.vn/index.php/2020/10/27/introductionpython/>

[4]: Giới thiệu thư viện Keras , <https://viblo.asia/p/gioi-thieu-ve-deep-learning-thu-vien-keras-63vKjDGAl2R>

[5]: Giới thiệu OpenCV, <https://topdev.vn/blog/opencv-la-gi-hoc-computer-vision-khong-kho/>

[6]: Giới thiệu Google Colab , <https://codelearn.io/sharing/google-colab-la-gi>

Và một số tài liệu tham khảo khác.

# PHỤ LỤC

**1/ Code tạo tập dữ liệu ( chụp ảnh từ camera ), dùng ở IDE Spyder**

import cv2

import imutils

#import numpy as np

#from sklearn.metrics import pairwise

#import time

bg = None

def run\_avg(image, accumWeight):

global bg

if bg is None:

bg = image.copy().astype("float")

return

cv2.accumulateWeighted(image, bg, accumWeight)

# Function - To segment the region of hand in the image

def segment(image, threshold=30):

global bg

diff = cv2.absdiff(bg.astype("uint8"), image)

thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH\_BINARY)[1]

(cnts, \_) = cv2.findContours(thresholded.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

if len(cnts) == 0:

return

else:

segmented = max(cnts, key=cv2.contourArea)

return (thresholded, segmented)

# Main function

if \_\_name\_\_ == "\_\_main\_\_":

accumWeight = 0.5

camera = cv2.VideoCapture("http://192.168.1.35:4747/video")

top, right, bottom, left = 44, 344, 300, 600

num\_frames = 0

imageNumber =1

calibrated = False

while(True):

(grabbed, frame) = camera.read()

frame = imutils.resize(frame, width=700)

frame = cv2.flip(frame, 1)

clone = frame.copy()

(height, width) = frame.shape[:2]

roi = frame[top:bottom, right:left]

gray = cv2.cvtColor(roi, cv2.COLOR\_BGR2GRAY)

gray = cv2.GaussianBlur(gray, (7, 7), 0)

if num\_frames < 30:

run\_avg(gray, accumWeight)

if num\_frames == 1:

print (">>> Please wait!...")

elif num\_frames == 29:

print (">>> Tiến Hành...")

else:

hand = segment(gray)

if hand is not None:

(thresholded, segmented) = hand

# show the thresholded image

cv2.imshow("Thesholded", thresholded)

keypress = cv2.waitKey(1) & 0xFF

if keypress == ord("a"):

# Set the directory CORRECTLY

directory = "D:/Dataset/Hand\_Fi/train/peace/"+str(imageNumber)+".jpg"

cv2.imwrite(directory,thresholded)

imageNumber += 1

print(directory)

cv2.rectangle(clone, (left, top), (right, bottom), (0,255,0), 2)

num\_frames += 1

cv2.imshow("Video Feed", clone)

keypress = cv2.waitKey(1) & 0xFF

if keypress == ord("q"):

break

camera.release()

cv2.destroyAllWindows()

**2/ Code mô hình CNN dùng trên google colab**

from google.colab import drive

drive.mount('/content/drive')

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

from keras.layers import Dropout

from keras.layers import Activation

from keras.preprocessing.image import ImageDataGenerator

from keras.models import load\_model

train\_datagen = ImageDataGenerator(

        rescale=1./255,

        shear\_range=0.2,

        zoom\_range=0.2,

        horizontal\_flip=True,

samplewise\_center=True,

vertical\_flip=True)

test\_datagen = ImageDataGenerator(rescale=1./255)

training\_set = train\_datagen.flow\_from\_directory(

        '/content/drive/MyDrive/model\_Fi/Data\_Hand\_Fi/train',

        target\_size=(64,64), #Resize phù hợp với mô hình CNN

        batch\_size=32,

        color\_mode='grayscale', #Ảnh xám

        class\_mode='categorical') #Gán label

test\_set = test\_datagen.flow\_from\_directory(

        '/content/drive/MyDrive/model\_Fi/Data\_Hand\_Fi/test',

        target\_size=(64,64),#Resize phù hợp với mô hình CNN

        batch\_size=32,

        color\_mode='grayscale', #Ảnh xám

        class\_mode='categorical')#Gán label

classifier = Sequential()

#step 1 - convolution and polling

classifier.add(Convolution2D(32,(3,3),input\_shape=(64,64,1),activation='relu',padding='same'))

classifier.add(Activation('relu'))

classifier.add(Convolution2D(32,(3,3),input\_shape=(64,64,1),activation='relu',padding='same'))

classifier.add(Activation('relu'))

classifier.add(MaxPooling2D(pool\_size=(3,3),strides=2))

classifier.add(Dropout(0.25))

#ADDING 2ND CONVOLUTION and polling

classifier.add(Convolution2D(64,(3,3),input\_shape=(64,64,1),activation='relu',padding='same'))

classifier.add(Activation('relu'))

classifier.add(Convolution2D(64,(3,3),input\_shape=(64,64,1),activation='relu',padding='same'))

classifier.add(Activation('relu'))

classifier.add(MaxPooling2D(pool\_size=(3,3),strides=2))

classifier.add(Dropout(0.25))

#step3 Flatten

classifier.add(Flatten())

#creating ANN

classifier.add(Dense(units=512,activation='relu'))

classifier.add(Dropout(0.5))

classifier.add(Dense(units=5,activation='softmax'))

#complie the CNN

classifier.compile(optimizer='rmsprop',loss='categorical\_crossentropy', metrics = ['accuracy'])

#Tiến hành train

classifier.fit\_generator(training\_set,

        steps\_per\_epoch=93,#no of images in training set/batch\_size

        epochs=3,

        validation\_data=test\_set,

        validation\_steps=46)

#Xuất file .h5 lưu trữ thông số sau khi train

classifier.save('/content/drive/MyDrive/model\_Fi/model.h5')

**3/ Code test model dùng Spyder**

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

import numpy as np

from keras.preprocessing import image

from keras.models import load\_model

import cv2

newmod=load\_model('C:/Users/FPTSHOP/hand\_gestures.h5')

background = None

accumulated\_weight = 0.4

pre\_top = 20

pre\_bot = 300

pre\_right = 300

pre\_left = 600

def segment(frame, threshold=20):

global background

diff = cv2.absdiff(background.astype("uint8"), frame)

\_ , thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH\_BINARY)

contours, hierarchy = cv2.findContours(thresholded.copy(), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

if len(contours) == 0:

return None

else:

hand\_segment = max(contours, key=cv2.contourArea)

return (thresholded, hand\_segment)

def calc\_accum\_avg(frame, accumulated\_weight):

global background

if background is None:

background = frame.copy().astype("float")

return None

cv2.accumulateWeighted(frame, background, accumulated\_weight)

def predict\_display(img):

width=64

height=64

dim=(width,height)

resized = cv2.resize(img, dim, interpolation = cv2.INTER\_AREA)

test\_img=image.img\_to\_array(resized)

test\_img=np.expand\_dims(test\_img,axis=0)

result= newmod.predict(test\_img)

val=[index for index,value in enumerate(result[0]) if value ==1]

return val

cam = cv2.VideoCapture(0)

num\_frames = 0

while True:

ret, frame = cam.read()

frame = cv2.flip(frame, 1)

frame\_copy = frame

roi = frame[pre\_top:pre\_bot, pre\_right:pre\_left]

gray = cv2.cvtColor(roi, cv2.COLOR\_BGR2GRAY)

gray = cv2.GaussianBlur(gray, (7, 7), 0)

if num\_frames < 60:

calc\_accum\_avg(gray, accumulated\_weight)

if num\_frames <= 59:

cv2.putText(frame\_copy, "DANG XU LI BACKGROUND.", (200, 400), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,0,255), 2)

else:

cv2.putText(frame\_copy, "Place your hand in side the box", (330, 340), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,0,255),1)

cv2.putText(frame\_copy, "Buffalo = 0", (330, 355), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,0,255),1)

cv2.putText(frame\_copy, "Fist = 1", (330, 370), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,0,255),1)

cv2.putText(frame\_copy, "Five = 2", (330, 385), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,0,255),1)

cv2.putText(frame\_copy, "Okay = 3", (330, 400), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,0,255),1)

cv2.putText(frame\_copy, "Hi = 4", (330, 415), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,0,255),1)

hand = segment(gray)

if hand is not None:

thresholded, hand\_segment = hand

cv2.drawContours(frame\_copy, [hand\_segment + (pre\_right, pre\_top)], -1, (255, 0, 0),1)

cv2.imshow("Thresholded Image", thresholded)

res=predict\_display(thresholded)

if len(res)==0:

cv2.putText(frame\_copy, str('None'), (70, 45), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,0,255), 2)

else:

x='index'+str(res[0])

cv2.putText(frame\_copy, str(x), (70, 45), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,0,255), 2)

cv2.rectangle(frame\_copy, (pre\_left, pre\_top), (pre\_right, pre\_bot), (0,0,255), 2)

num\_frames += 1

cv2.imshow("Hand Gestures", frame\_copy)

# Close windows with Esc

k = cv2.waitKey(1) & 0xFF

if k == 27:

break

cam.release()

cv2.destroyAllWindows()