TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

**KHOA ĐIỆN TỬ**

**BỘ MÔN: TIN HỌC CÔNG NGHIỆP**



**BÀI TẬP LỚN**

**THỊ GIÁC MÁY**

*Đề tài: Nhận dạng hoạt động thể chất*

Sinh viên: Đặng Hoàng Nam

Lớp: K55KMT

Giáo viên hướng dẫn: Đặng Thị Hiên

**THÁI NGUYÊN – 2023**

|  |  |
| --- | --- |
| **TRƯỜNG ĐHKTCN** | **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| **KHOA ĐIỆN TỬ** | ***Độc lập - Tự do - Hạnh phúc*** |

**BÀI TẬP LỚN**

**BÀI TẬP LỚN : MÔN THỊ GIÁC MÁY**

BỘ MÔN: TIN HỌC CÔNG NGHIỆP

*Sinh viên: Đặng Hoàng Nam Lớp: K55KMT*

*MSSV: K1954 8010 6015 Ngành: Kỹ thuật máy tính*

*Ngày giao đề: 09/01/2023 Ngày hoàn thành: 23/02/2023*

1.Tên đề tài : Nhận dạng hoạt động thể chất

.

2. Yêu cầu : Đọc được ảnh của một người đang thực hiện động tác thể dục và nhận dạng được 5 bài tập thể dục :

1. Chống đẩy

2. Gập bụng

3. Lên xà

4. Jumping Jack

5. Squat

|  |  |
| --- | --- |
| TỔ TRƯỞNG BỘ MÔN | GIÁO VIÊN HƯỚNG DẪN |
| *(Ký và ghi rõ họ tên)* | *(Ký và ghi rõ họ tên)* |

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

*Thái Nguyên, ngày.….tháng…..năm 2023*

**GIÁO VIÊN HƯỚNG DẪN**

*(Ký ghi rõ họ tên)*

NHẬN XÉT CỦA GIÁO VIÊN CHẤM

*Thái Nguyên, ngày.….tháng…..năm 2023*

**GIÁO VIÊN CHẤM**

*(Ký ghi rõ họ tên)*

**CÁC HÌNH ẢNH SỬ DỤNG TRONG BÁO CÁO**

***Hình 1. Đếm số lần chống đẩy sử dụng Python và Mediapipe***

***Hình 2. Logo Visual Studio Code và giao diện làm việc***

***Hình 3. Ngôn ngữ lập trình Python***

***Hình 4. Thư viện OpenCV cho xử lý ảnh***

***Hình 5. Sử dụng thư viện MediaPipe để phát hiện cử chỉ và khuôn mặt***

***Hình 6. Tọa độ X, Y, Z của các điểm tư thế người trong file CSV***

***Hình 7. Keras và TensorFlow***

***Hình 8. Ảnh kết quả là hình ảnh gốc với tên động tác dự đoán***

***Hình 9. 33 điểm đặt mốc trên cơ thể người***

***Hình 10. Hình ảnh kết quả sau khi vẽ khung xương người Mediapipe***

***Hình 11. Terminal -> New Terminal***

***Hình 12. Biểu đồ được vẽ và lưu vào ảnh metrics.png***

***Hình 13. Hình ảnh sau khi nhận diện***

***Hình 14 + 15. Kết quả của chương trình***

**MỤC LỤC**

[**LỜI MỞ ĐẦU** 6](#_Toc128834979)

[**CHƯƠNG I. GIỚI THIỆU TỔNG QUAN VỀ ĐỀ TÀI** 7](#_Toc128834980)

[**1.1. Mục tiêu của đề tài** 7](#_Toc128834981)

[**1.2. Quá trình hoạt động của chương trình** 8](#_Toc128834982)

[**1.3. Ngôn ngữ lập trình và môi trường làm việc** 8](#_Toc128834983)

[***1.3.1. Visual Studio Code*** *8*](#_Toc128834984)

[***1.3.2. Python*** *9*](#_Toc128834985)

[***1.3.3. Thư viện OpenCV*** *11*](#_Toc128834986)

[***1.3.4. Thư viện Google MediaPipe*** *12*](#_Toc128834987)

[**CHƯƠNG II. NHẬN DẠNG HOẠT ĐỘNG THỂ CHẤT** 14](#_Toc128834988)

[**2.1. Thuật toán** 14](#_Toc128834989)

[***2.1.1. Tạo bộ dữ liệu Landmark*** *14*](#_Toc128834990)

[***2.1.2. Khởi tạo mô hình tuần tự Sequential Model*** *15*](#_Toc128834991)

[***2.1.3. Sử dụng mô hình vừa tạo để suy luận ra tư thế người*** *16*](#_Toc128834992)

[**2.2. Các hàm sử dụng trong chương trình** 17](#_Toc128834993)

[***2.2.1. Pose Landmark*** *17*](#_Toc128834994)

[***2.2.2. MediaPipe Pose Model*** *18*](#_Toc128834995)

[**CHƯƠNG III. CODE VÀ CHẠY CHƯƠNG TRÌNH** 21](#_Toc128834996)

[**3.1.Đọc bộ dữ liệu ảnh và ghi vào file CSV** 21](#_Toc128834997)

[**3.2. Khởi tạo Sequential Model và Training** 24](#_Toc128834998)

[**3.3. Dự đoán tư thế người trong ảnh** 26](#_Toc128834999)

[**3.4. Chạy chương trình** 31](#_Toc128835000)

[***3.4.1. Download bộ dữ liệu hình ảnh*** *31*](#_Toc128835001)

[***3.4.2. Tạo file CSV với các lớp tương ứng*** *31*](#_Toc128835002)

[***3.4.3. Thực hiện train file CSV và xuất ra biểu đồ*** *32*](#_Toc128835003)

[***3.4.4. Nhận diện ảnh*** *33*](#_Toc128835004)

[***3.4.5. Kết quả chạy chương trình*** *34*](#_Toc128835005)

[**CHƯƠNG IV. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 35](#_Toc128835006)

[**4.1. Nhận xét và kết luận** 35](#_Toc128835007)

[**4.2. Hướng phát triển của đề tài** 35](#_Toc128835008)

[**TÀI LIỆU THAM KHẢO** 36](#_Toc128835009)

**LỜI MỞ ĐẦU**

Ngày nay, khi xã hội dần phát triển, con người sẽ muốn chú trọng hơn vào sức khỏe và sắc đẹp. Từ đó, nhu cầu tập Gym ngày càng gia tăng. Tập Gym mang lại rất nhiều lợi ích như giúp giảm cân, vóc dáng cân đối, khỏe mạnh, giúp cơ bắp săn chắc, cải thiện sức khỏe và tinh thần. Chính những lợi ích mà Gym đem lại, ngày nay không chỉ có giới trẻ tham gia, đây dường như là phương pháp rèn luyện bản thân được cả những người lớn tuổi lựa chọn.

Mặc dù tập Gym có lợi ích rất lớn đối với sức khỏe mỗi người, song nếu luyện tập không đúng cách và chưa có những kiến thức cơ bản khi luyện tập, chúng ta có thể sẽ gặp phải những rủi ro không đáng có như chấn thương, suy giảm sức khỏe, tập quá sức, tập sai động tác, chế độ ăn uống không điều độ…Vì vậy, giải pháp được sử dụng để giảm thiểu các sai sót không đáng có trong quá trình luyện tập đó là sử dụng các phần mềm tập luyện với trí tuệ nhân tạo để cùng đồng hành với người tập.

Cùng với sự phát triển của xã hội, các hệ thống giúp nhận dạng con người, đồ vật, hoạt động… được ra đời và phát triển với độ tin cậy ngày càng cao. Những hệ thống này giúp ghi lại các hoạt động của con người, đưa ra dữ liệu và điều chỉnh sao cho phù hợp với bài tập. Với cách nhận dạng hoạt động theo phương pháp này, chúng ta có thể thu nhập được nhiều thông tin hơn, đảm bảo tính chính xác, an toàn, thuận tiện.

Trong phạm vi bài báo cáo này chúng em xin được trình bày quá trình thực hiện bài tập **“Nhận dạng hoạt động thể chất”** qua thư viện OpenCV và Mediapipe. Cuối cùng, mặc dù đã cố gắng rất nhiều nhưng do thời gian có hạn, khả năng dịch và hiểu tài liệu chưa tốt nên nội dung đồ án này không thể tránh khỏi những thiếu sót, rất mong được sự chỉ bảo, góp ý của các thầy cô và các bạn.

# **CHƯƠNG I. GIỚI THIỆU TỔNG QUAN VỀ ĐỀ TÀI**

## **1.1. Mục tiêu của đề tài**

Mục tiêu của đề tài “Nhận dạng hoạt động thể chất” là nhận dạng và phân biệt được 5 bài tập thể dục cơ bản:

* **Push-up** (Chống đẩy)
* **Pull-up** (Lên xà đơn)
* **Sit-up** (Gập bụng)
* **Jumping Jack**
* **Squat**

Đề tài sẽ nhận một ảnh đầu vào (định dạng jpg, png, jpeg,…) là hình ảnh của một người đang thực hiện một động tác thể dục nào đó trong 5 động tác kể trên. Chương trình sẽ phân tích và đưa ra kết quả là hình ảnh đó và tên của động tác thể dục trong ảnh.



***Hình 1. Đếm số lần chống đẩy sử dụng Python và Mediapipe***

## **1.2. Quá trình hoạt động của chương trình**

* **Nhận mẫu:** Sau khi nhận ảnh đầu vào, chương trình sẽ lấy mẫu khung xương của đối tượng trong ảnh, sử dụng thư viện MediaPipe Pose.
* **Phân tích dữ liệu:** Các dữ liệu sau khi được thu thập sẽ được lưu trữ và trích xuất thành các vị trí khung xương đã thu thập dưới dạng tọa độ x, y
* **So sánh dữ liệu:** Các dữ liệu hệ thống vừa thu thập sẽ được đối chiếu với tập dữ liệu đã được lưu trữ và training trước đó.
* **Hiển thị kết quả:** Hệ thống sẽ xuất ra hình ảnh và tên của động tác thể dục mà người trong ảnh đang thực hiện.

## **1.3. Ngôn ngữ lập trình và môi trường làm việc**

### **1.3.1. Visual Studio Code**

**Visual Studio Code** là một [trình soạn thảo mã nguồn](https://vi.wikipedia.org/wiki/Tr%C3%ACnh_so%E1%BA%A1n_th%E1%BA%A3o_m%C3%A3_ngu%E1%BB%93n) được phát triển bởi [Microsoft](https://vi.wikipedia.org/wiki/Microsoft) dành cho [Windows](https://vi.wikipedia.org/wiki/Microsoft_Windows), [Linux](https://vi.wikipedia.org/wiki/Linux) và [macOS](https://vi.wikipedia.org/wiki/MacOS). Nó hỗ trợ chức năng debug, đi kèm với [Git](https://vi.wikipedia.org/wiki/Git_(ph%E1%BA%A7n_m%E1%BB%81m)), có chức năng nổi bật cú pháp ([syntax highlighting](https://vi.wikipedia.org/w/index.php?title=Syntax_highlighting&action=edit&redlink=1)), tự hoàn thành mã thông minh, [snippets](https://vi.wikipedia.org/w/index.php?title=Snippets&action=edit&redlink=1), và [cải tiến mã nguồn](https://vi.wikipedia.org/wiki/C%E1%BA%A3i_ti%E1%BA%BFn_m%C3%A3_ngu%E1%BB%93n). Nó cũng cho phép tùy chỉnh, do đó, người dùng có thể thay đổi theme, phím tắt, và các tùy chọn khác. Nó miễn phí và là [phần mềm mã nguồn mở](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m_t%E1%BB%B1_do_ngu%E1%BB%93n_m%E1%BB%9F) theo [giấy phép MIT](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_MIT), mặc dù bản phát hành của [Microsoft](https://vi.wikipedia.org/wiki/Microsoft) là theo [giấy phép phần mềm](https://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_ph%E1%BA%A7n_m%E1%BB%81m) miễn phí.

**Visual Studio Code** được dựa trên [Electron](https://en.wikipedia.org/wiki/Electron_(software_framework)), một nền tảng được sử dụng để triển khai các ứng dụng [Node.js](https://vi.wikipedia.org/wiki/Node.js) [máy tính cá nhân](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh_c%C3%A1_nh%C3%A2n) chạy trên động cơ bố trí Blink. Mặc dù nó sử dụng nền tảng Electronnhưng phần mềm này không phải là một bản khác của [Atom](https://vi.wikipedia.org/wiki/Atom), nó thực ra được dựa trên trình biên tập của [Visual Studio Online](https://vi.wikipedia.org/wiki/Microsoft_Visual_Studio) (tên mã là "Monaco").

Graphical user interface, text, application

Description automatically generated

***Hình 2. Logo Visual Studio Code và giao diện làm việc***

**Visual Studio Code** là một trình biên tập mã. Nó hỗ trợ nhiều ngôn ngữ và chức năng tùy vào ngôn ngữ sử dụng theo như trong bảng sau. Nhiều chức năng của Visual Studio Code không hiển thị ra trong các menu tùy chọn hay [giao diện người dùng](https://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_ng%C6%B0%E1%BB%9Di_d%C3%B9ng). Thay vào đó, chúng được gọi thông qua khung nhập lệnh hoặc qua một tập tin [.json](https://vi.wikipedia.org/wiki/JSON) (ví dụ như tập tin tùy chỉnh của người dùng). Khung nhập lệnh là một giao diện theo dòng lệnh. Tuy nhiên, nó biến mất khi người dùng nhấp bất cứ nơi nào khác, hoặc nhấn tổ hợp phím để tương tác với một cái gì đó ở bên ngoài đó. Tương tự như vậy với những dòng lệnh tốn nhiều thời gian để xử lý. Khi thực hiện những điều trên thì quá trình xử lý dòng lệnh đó sẽ bị hủy.

**Visual Studio Code** có thể được mở rộng qua [plugin](https://vi.wikipedia.org/wiki/Plugin). Điều này giúp bổ sung thêm chức năng cho trình biên tậpvà hỗ trợ thêm ngôn ngữ. Một tính năng đáng chú ý là khả năng tạo phần mở rộng để phân tích mã, như là các linter và công cụ phân tích, sử dụng [Language Server Protocol](https://vi.wikipedia.org/w/index.php?title=Language_Server_Protocol&action=edit&redlink=1).

### **1.3.2. Python**

**Python** là một [ngôn ngữ lập trình](https://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh) bậc cao cho các mục đích lập trình đa năng, do [Guido van Rossum](https://vi.wikipedia.org/w/index.php?title=Guido_van_Rossum&action=edit&redlink=1) tạo ra và lần đầu ra mắt vào năm [1991](https://vi.wikipedia.org/wiki/1990). 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 và là ngôn ngữ lập trình dễ học; được dùng rộng rãi trong phát triển [trí tuệ nhân tạo](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o). Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu.

**Python** luôn được xếp hạng vào những ngôn ngữ lập trình phổ biến nhất.

**Python** hoàn toàn [tạo kiểu động](https://vi.wikipedia.org/w/index.php?title=T%E1%BA%A1o_ki%E1%BB%83u_%C4%91%E1%BB%99ng&action=edit&redlink=1) và dùng cơ chế [cấp phát bộ nhớ tự động](https://vi.wikipedia.org/wiki/Qu%E1%BA%A3n_l%C3%BD_b%E1%BB%99_nh%E1%BB%9B); do vậy nó tương tự như [Perl](https://vi.wikipedia.org/wiki/Perl), [Ruby](https://vi.wikipedia.org/wiki/Ruby_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)), [Scheme](https://vi.wikipedia.org/wiki/Scheme), [Smalltalk](https://vi.wikipedia.org/wiki/Smalltalk), và [Tcl](https://vi.wikipedia.org/wiki/Tcl). Python được phát triển trong một dự án mã mở, do [tổ chức phi lợi nhuận](https://vi.wikipedia.org/wiki/T%E1%BB%95_ch%E1%BB%A9c_phi_l%E1%BB%A3i_nhu%E1%BA%ADn) Python Software Foundation quản lý.

Logo, company name

Description automatically generated

***Hình 3. Ngôn ngữ lập trình Python***

**Python** là một [ngôn ngữ lập trình đa mẫu hình](https://vi.wikipedia.org/wiki/M%E1%BA%ABu_h%C3%ACnh_l%E1%BA%ADp_tr%C3%ACnh). [Lập trình hướng đối tượng](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng) và [lập trình cấu trúc](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_c%E1%BA%A5u_tr%C3%BAc) được hỗ trợ hoàn toàn, và nhiều tính năng của nó cũng hỗ trợ lập trình hàm và [lập trình hướng khía cạnh](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_kh%C3%ADa_c%E1%BA%A1nh) (bao gồm [siêu lập trình](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_meta) và [siêu đối tượng](https://vi.wikipedia.org/w/index.php?title=Si%C3%AAu_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng&action=edit&redlink=1). Các mẫu hình khác cũng được hỗ trợ thông qua các phần mở rộng, bao gồm [thiết kế theo hợp đồng](https://vi.wikipedia.org/w/index.php?title=Thi%E1%BA%BFt_k%E1%BA%BF_theo_h%E1%BB%A3p_%C4%91%E1%BB%93ng&action=edit&redlink=1) và [lập trình logic](https://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_logic).

**Python** sử dụng [kiểu động](https://vi.wikipedia.org/wiki/H%E1%BB%87_th%E1%BB%91ng_ki%E1%BB%83u) và một dạng kết hợp giữa [đếm tham chiếu](https://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BA%BFm_tham_chi%E1%BA%BFu&action=edit&redlink=1) và bộ dọn rác kiểm tra theo chu kì để [quản lí bộ nhớ](https://vi.wikipedia.org/wiki/Qu%E1%BA%A3n_l%C3%AD_b%E1%BB%99_nh%E1%BB%9B). Nó cũng có tính năng [phân giải tên](https://vi.wikipedia.org/w/index.php?title=Ph%C3%A2n_gi%E1%BA%A3i_t%C3%AAn&action=edit&redlink=1) động ([liên kết muộn](https://vi.wikipedia.org/w/index.php?title=Li%C3%AAn_k%E1%BA%BFt_mu%E1%BB%99n&action=edit&redlink=1)), cho phép liên kết các tên biến và phương thức trong quá trình thực thi chương trình.

Triết lý căn bản của ngôn ngữ Python được trình bày trong tài liệu *The*[*Zen of Python*](https://vi.wikipedia.org/wiki/Zen_of_Python) (*PEP 20*), có dạng [thơ Haiku](https://vi.wikipedia.org/wiki/Haiku), tóm gọn như sau:

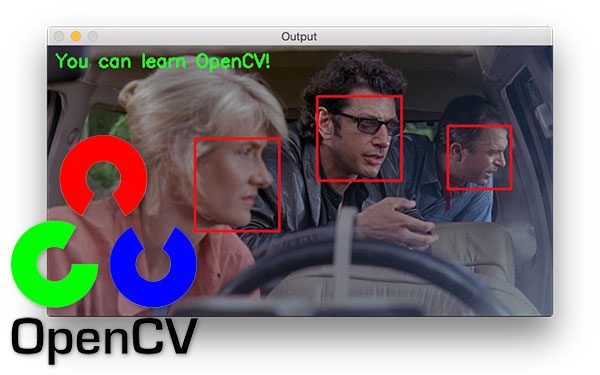
* Đẹp đẽ tốt hơn xấu xí
* Minh bạch tốt hơn ngầm định
* Đơn giản tốt hơn phức tạp
* Phức tạp tốt hơn rắc rối
* Tính dễ đọc rất quan trọng.

Thay vì tích hợp hết tất cả các tính năng vào phần cốt lõi, Python được thiết kế để [dễ dàng mở rộng](https://vi.wikipedia.org/w/index.php?title=T%C3%ADnh_m%E1%BB%9F_r%E1%BB%99ng&action=edit&redlink=1) (bằng các mô đun). Tính mô đun nhỏ gọn này đã làm cho Python trở nên phổ biến như là một cách thêm các giao diện lập trình được vào các ứng dụng hiện có. Tầm nhìn của Van Rossum về một ngôn ngữ có phần lõi nhỏ với một thứ viện chuẩn rộng lớn và một trình thông dịch dễ dàng mở rộng bắt nguồn từ việc ông nản lòng trước [ABC](https://vi.wikipedia.org/w/index.php?title=ABC_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)&action=edit&redlink=1), một ngôn ngữ lập trình tán thành hướng tiếp cận ngược lại. Python thường được mô tả là một ngôn ngữ "tặng kèm pin" nhờ vào [thư viện chuẩn](https://vi.wikipedia.org/wiki/Th%C6%B0_vi%E1%BB%87n_chu%E1%BA%A9n) bao quát của nó.

**Python** nỗ lực hướng đến một cú pháp đơn giản hơn, gọn gàng hơn trong khi vẫn cho các nhà phát triển lựa chọn phương pháp viết mã của họ. Đối lập với khẩu hiệu "[có nhiều hơn một cách để làm việc này](https://vi.wikipedia.org/w/index.php?title=C%C3%B3_nhi%E1%BB%81u_h%C6%A1n_m%E1%BB%99t_c%C3%A1ch_%C4%91%E1%BB%83_l%C3%A0m_vi%E1%BB%87c_n%C3%A0y&action=edit&redlink=1)", triết lý thiết kể của Python lại nằm trong châm ngôn "chỉ nên có một— và tốt nhất là chỉ một—cách rõ ràng để làm việc này". [Alex Martelli](https://vi.wikipedia.org/w/index.php?title=Alex_Martelli&action=edit&redlink=1), một Viện sĩ (Fellow) tại Tổ chức Phần mềm Python (Python Software Foundation) và là một tác giả viết sách Python, viết rằng "Mô tả một thứ gì đó là "tài tình" *không* được coi là một lời khen ngợi trong văn hoá Python."

### **1.3.3. Thư viện OpenCV**

**OpenCV** (**Open Source Computer Vision Library**) do Intel phát triển, được giới thiệu năm 1999 và hoàn thiện thành phiên bản 1.0 năm 2006. Thư viện OpenCV – gồm khoảng 500 hàm – được viết bằng ngôn ngữ lập trình C và tương thích với các hệ điều hành Windows, Linux, Mac OS… đóng vai trò xác lập chuẩn giao tiếp, dữ liệu, thuật toán cho lính vực CV và tọa điều kiện cho mọi người tham gia nghiên cứu và phát triển ứng dụng.



***Hình 4. Thư viện OpenCV cho xử lý ảnh***

Trước **OpenCV** không có một công cụ chuẩn nào cho lĩnh vực xử lí ảnh. Các đoạn code đơn lẻ do các nhà nghiên cứu tự viết thường không thống nhất và không ổn định. Các bộ công cụ thương mại như Matlab, Simulink,..v.v.. lại có giá cao chỉ thích hợp cho các công ty phát triển các ứng dụng lớn. Ngoài ra còn có các giải phảp kèm theo thiết bị phần cứng mà phần lớn là mã đóng và được thiết kế riêng cho tứng thiết bị, rất khó khan cho việc mở rộng ứng dụng.

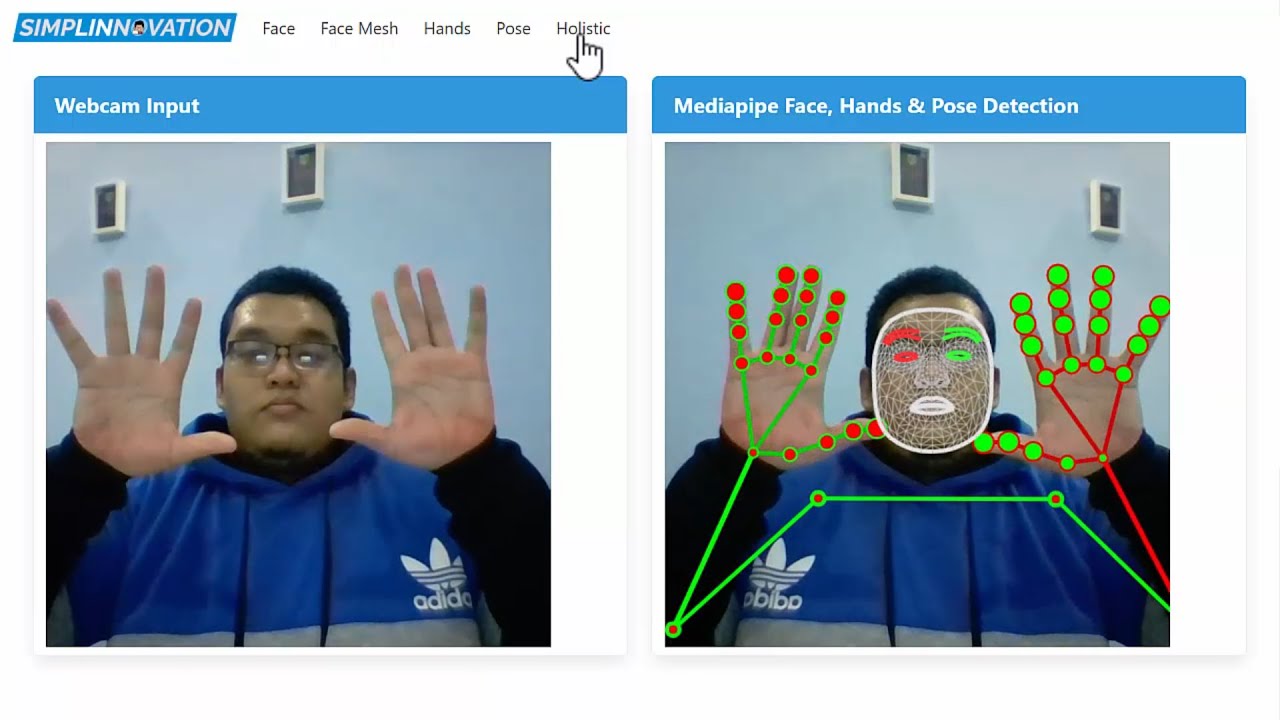
**OpenCV** là công cụ hữu ích cho những người bước đầu làm quen với xử lí ảnh số vì các ưu điểm sau:

* **OpenCV là công cụ chuyên dụng:** được Intel phát triển theo hướng tối ưu hóa cho các ứng dụng xử lí và phân tích ảnh, với cấu trúc dữ liệu hợp lí, thư viện tạo giao diện, truy xuất thiết bị phần cứng được tích hợp sẵn. OpenCV thích hợp để phát triển nhanh ứng dụng
* **OpenCV là công cụ mã nguồn mở:** Không chỉ là công cụ miễn phí, việc được xây dựng trên mã nguốn mở giúp OpenCV trở thành công cụ thích hợp cho nghiên cứu và phát triển, với khả năng thay đổi và mở rộng các mô hình, thuật toán
* **OpenCV đã được sử dụng rộng rãi:** Từ năm 1999 đến nay, OpenCV đã thu hút được một lượng lớn người dung, trong đó có các công ty lớn như Microsoft, IBM, Sony, Siemens, Google và các nhóm nghiên cứu ở Standford, MIT, CMU, Cambridge,… Nhiều forum hỗ trợ và cộng đồng người dung đã được thành lập, tạo nên kênh thông tin rộng lớn, hữu ích cho việc tham khảo, tra cứu.

### **1.3.4. Thư viện Google MediaPipe**

**MediaPipe** là thư viện phát hiện cử chỉ hình thể rất chính xác và nhẹ**.** [MediaPipe](https://github.com/google/mediapipe) là tập hợp của một loạt các giải pháp Machine Learning đa nền tảng, có thể can thiệp được. Một số ưu điểm có thể kể tới của giải pháp này bao gồm:

* **Cung cấp một giải pháp inference nhanh chóng**: Google khẳng định rằng bộ công cụ này có thể chạy ổn định trên hầu hết các cấu hình phần cứng thông dụng.
* **Dễ dàng cài đặt và triển khai:** Việc cài đặt cực kỳ dễ dàng và tiện lợi, có thể triển khai trên nhiều nền tảng khác nhau như Mobile (Android/iOS), Desktop/Cloud, Web và IoT devices.
* **Mã nguồn mở và miễn phí:** Toàn bộ source code được công khai trên [MediaPipe](https://github.com/google/mediapipe), người dùng hoàn toàn có thể sử dụng và tùy chỉnh trực tiếp để phù hợp với bài toán của mình.



***Hình 5. Sử dụng thư viện MediaPipe để phát hiện cử chỉ và khuôn mặt***

Các ứng dụng của thư viện MediaPipe bao gồm:

* **Phát hiện khuôn mặt**
* **Tìm lưới khuôn mặt**: Dùng cho các ứng dụng biến đổi khuôn mặt như Tiktok
* **Iris**: Tìm khoảng cách từ đồng tử của mắt đến camera mà không cần Depth webcam
* **Detect cử chỉ bàn tay**
* **Tìm hình dáng của cơ thể (MediaPipe Pose)**
* **Thay đổi màu tóc:** có thể hair salon sẽ cần
* **Object detection & Box tracking:** Tìm vật thể
* **Theo dõi chuyển động của vật thể**
* **Objectron**: tìm hình lập phương chứa vật thể
* **KNIFT**: Tìm vật thể bằng các đặc trưng đã biết

# **CHƯƠNG II. NHẬN DẠNG HOẠT ĐỘNG THỂ CHẤT**

## **2.1. Thuật toán**

Bài tập “Nhận dạng hoạt động thể chất” có thể được chia thành 3 phần:

1. Tạo bộ dữ liệu Landmark với tọa độ của 33 điểm cơ thể người và lưu vào file CSV
2. Khởi tạo mô hình tuần tự để dự đoán tư thế người
3. Sử dụng mô hình vừa tạo để suy luận ra tư thế người.

### **2.1.1. Tạo bộ dữ liệu Landmark**

Tạo bộ dữ liệu Landmark với tọa độ của 33 điểm cơ thể người. Bộ dữ liệu này bao gồm các file ảnh được chia ra thành các lớp khác nhau.

Ví dụ: Bộ dữ liệu “Các bài tập thể dục cơ bản” bao gồm 5 lớp: Chống đẩy, Hít đất, Lên xà đơn, Jumping Jack, Squat. Cấu trúc của bộ dữ liệu này như sau:

Filedata

Chong\_day

chong\_day\_1.jpg

chong\_day\_2.jpg

…

Hit\_dat

hit\_dat\_1.jpg

hit\_dat\_2.jpg

…

Squat

squat\_1.jpg

squat\_2.jpg

…

Các file ảnh sau khi được gán tọa độ sẽ được đưa vào một file csv. Ở đây sẽ chứa toàn bộ tọa độ của 33 điểm cơ thể người của từng ảnh. Dữ liệu này sẽ được chia ra thành các lớp tương ứng với số lớp của bộ dữ liệu ảnh.

Table

Description automatically generated

***Hình 6. Tọa độ X, Y, Z của các điểm tư thế người trong file CSV***

### **2.1.2. Khởi tạo mô hình tuần tự Sequential Model**

Để hiểu được mô hình tuần tự là gì, chúng ta sẽ cùng đi tìm hiểu về thư viện mạng nơ ron **Keras**.

**Keras** là một thư viện phần mềm mã nguồn mở cung cấp giao diện Python cho các mạng thần kinh nhân tạo. Keras hoạt động như một giao diện cho thư viện TensorFlow.

**Keras** chứa nhiều triển khai của các khối xây dựng mạng nơ-ron thường được sử dụng như lớp, [mục tiêu](https://en.wikipedia.org/wiki/Objective_function), [chức năng kích hoạt](https://en.wikipedia.org/wiki/Activation_function), [bộ tối ưu hóa](https://en.wikipedia.org/wiki/Mathematical_optimization) và một loạt công cụ giúp làm việc với dữ liệu hình ảnh và văn bản dễ dàng hơn nhằm đơn giản hóa mã hóa cần thiết để viết mã mạng nơ-ron sâu.

**Keras** cho phép người dùng sản xuất các mô hình sâu trên điện thoại thông minh ( iOS và [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) ), trên web hoặc trên [Máy ảo Java](https://en.wikipedia.org/wiki/Java_virtual_machine). Nó cũng cho phép sử dụng đào tạo phân tán các mô hình học sâu trên các cụm [Đơn vị xử lý đồ họa (GPU)](https://en.wikipedia.org/wiki/Graphics_processing_unit) và [Đơn vị xử lý tensor (TPU)](https://en.wikipedia.org/wiki/Tensor_processing_unit).

Logo

Description automatically generated

***Hình 7. Keras và TensorFlow***

Chương trình của chúng ta sẽ sử dụng thư viện Keras để tạo ra một mô hình Sequential model (mô hình tuần tự) gồm 3 lớp. Mô hình này sẽ có số chiều đầu ra bằng với số lớp cần phân loại trong file CSV.

Sau khi xây dựng xong model, chúng ta thực hiện training dữ liệu. Bước này cần đưa vào dữ liệu CSV đã khởi tạo từ trước. Máy tính sẽ thực hiện training với số lần lặp lại nhất định.

Cuối cùng là đánh giá mô hình. Ta sử dụng các hàm có sẵn của Keras để xuất ra số liệu như số lần training, độ chính xác, số lần bị sai…

### **2.1.3. Sử dụng mô hình vừa tạo để suy luận ra tư thế người**

Bước thứ 3 là sử dụng mô hình Sequential model vừa training để dự đoán tư thế người trong ảnh.

Ta nhập một ảnh đầu vào, máy tính sẽ gán khung xương MediaPipe Pose vào tư thế người trong ảnh, sau đó ghi ra dữ liệu đó. Tiếp theo sẽ so sánh với Sequential model vừa tạo và đưa ra kết quả dự đoán tư thế.

Trong chương trình này sẽ xuất ra ảnh mới là ảnh đầu vào và tên của động tác thể dục ở góc trái trên của ảnh.



***Hình 8. Ảnh kết quả là hình ảnh gốc với tên động tác dự đoán***

## **2.2. Các hàm sử dụng trong chương trình**

### **2.2.1. Pose Landmark**

Pose Landmark sẽ trả về danh sách các mốc tư thế. Mỗi mốc sẽ bao gồm:

* **X và Y:** Tọa độ mốc được chuẩn hóa [0.0, 1.0] theo chiều rộng và chiều cao của hình ảnh tương ứng.
* **Z:** Thể hiện độ sâu của mốc với độ sâu tại điểm giữa của hông là gốc và giá trị càng nhỏ thì mốc càng gần máy ảnh. Mức độ Z sử dụng gần giống với quy mô như X.
* **Visibility:** Một giá trị [0.0, 1.0] cho biết khả năng nhìn thấy mốc (hiện diện và không bị che khuất) trong ảnh.

Chart, radar chart

Description automatically generated

***Hình 9. 33 điểm đặt mốc trên cơ thể người***

### **2.2.2. MediaPipe Pose Model**

Bây giờ chúng ta cần khởi tạo một mô hình ước tính tư thế Mediapipe và chúng ta cũng sẽ sử dụng các tiện ích vẽ Mediapipe để dễ dàng vẽ các điểm trên hình ảnh. Mediapipe Pose cung cấp các tùy chọn khác nhau trong khi tạo đối tượng mô hình tư thế:

* **Static\_image\_mode:** Nếu chương trình chỉ có đầu vào hình ảnh duy nhất, ta sẽ đặt nó thành false. Nếu có một đầu vào video, ta có thể đặt nó thành true để nó cố gắng theo dõi từ các khung hình trước đó và sẽ tăng hiệu suất.
* **Model\_complexity:** Có thể là 0, 1 hoặc 2 trong đó 2 là model cao hơn
* **Enable\_segmentation:** Nếu được đặt thành true, nó cũng sẽ dự đoán mặt nạ cho hình ảnh nhất định
* **Min\_detection\_confidence:** Tin cậy của kết quả phát hiện, có thể được điều chỉnh theo yêu cầu và hình ảnh đầu vào

Tiếp theo chương trình sẽ đọc một hình ảnh từ thư mục bằng Opencv và cung cấp làm đầu vào cho Mediapipe. Opencv đọc hình ảnh dưới dạng BGR, vì vậy ta cần chuyển đổi hình ảnh sang định dạng RGB trước khi nhập vào mô hình.

image = cv2.imread("ronaldo.jpg")

#convert image to RGB (just for input to model)

image\_input = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

# get results using mediapipe

results = pose.process(image\_input)

Sử dụng hàm if-else để kiểm tra xem có ảnh đầu vào không và rút ra kết quả bằng cách sử dụng tiện ích vẽ Mediapipe.

if not results.pose\_landmarks:

print("no results found")

else:

mp\_drawing.draw\_landmarks(

image,

results.pose\_landmarks,

mp\_pose.POSE\_CONNECTIONS,

landmark\_drawing\_spec=mp\_drawing\_styles.get\_default\_pose\_landmarks\_style())

# write image to storage

cv2.imwrite("./ronaldo-processed.jpg", image)

Cuối cùng xuất ra hình ảnh và ghi lại vào bộ lưu trữ.

A person running on a field

Description automatically generated with medium confidence

***Hình 10. Hình ảnh kết quả sau khi vẽ khung xương người Mediapipe***

# **CHƯƠNG III. CODE VÀ CHẠY CHƯƠNG TRÌNH**

## **3.1.Đọc bộ dữ liệu ảnh và ghi vào file CSV**

import os

import cv2

import mediapipe as mp

import glob

import pandas as pd

import argparse

import numpy as np

import math

ap = argparse.ArgumentParser()

ap.add\_argument("-i", "--dataset", type=str, required=True,

                help="path to dataset/dir")

ap.add\_argument("-o", "--save", type=str, required=True,

                help="path to save csv file, eg: dir/data.csv")

args = vars(ap.parse\_args())

path\_data\_dir = args["dataset"]

path\_to\_save = args["save"]

##############

torso\_size\_multiplier = 2.5

n\_landmarks = 33

n\_dimensions = 3

landmark\_names = [

    'nose',

    'left\_eye\_inner', 'left\_eye', 'left\_eye\_outer',

    'right\_eye\_inner', 'right\_eye', 'right\_eye\_outer',

    'left\_ear', 'right\_ear',

    'mouth\_left', 'mouth\_right',

    'left\_shoulder', 'right\_shoulder',

    'left\_elbow', 'right\_elbow',

    'left\_wrist', 'right\_wrist',

    'left\_pinky\_1', 'right\_pinky\_1',

    'left\_index\_1', 'right\_index\_1',

    'left\_thumb\_2', 'right\_thumb\_2',

    'left\_hip', 'right\_hip',

    'left\_knee', 'right\_knee',

    'left\_ankle', 'right\_ankle',

    'left\_heel', 'right\_heel',

    'left\_foot\_index', 'right\_foot\_index',

]

##############

mp\_pose = mp.solutions.pose

pose = mp\_pose.Pose()

class\_list = os.listdir(path\_data\_dir)

class\_list = sorted(class\_list)

col\_names = []

for i in range(n\_landmarks):

    name = mp\_pose.PoseLandmark(i).name

    name\_x = name + '\_X'

    name\_y = name + '\_Y'

    name\_z = name + '\_Z'

    #name\_v = name + '\_V'

    col\_names.append(name\_x)

    col\_names.append(name\_y)

    col\_names.append(name\_z)

    #col\_names.append(name\_v)

full\_lm\_list = []

target\_list = []

for class\_name in class\_list:

    path\_to\_class = os.path.join(path\_data\_dir, class\_name)

    img\_list = glob.glob(path\_to\_class + '/\*.jpg') + \

        glob.glob(path\_to\_class + '/\*.jpeg') + \

        glob.glob(path\_to\_class + '/\*.png')

    img\_list = sorted(img\_list)

    # Đọc phạm vi hình ảnh trong mỗi lớp

    for img in img\_list:

        image = cv2.imread(img)

        h, w, c = image.shape

        if image is None:

            print(

                f'[ERROR] Lỗi đọc ảnh {img} -- Skipping.....\n[INFO] Đang lấy ảnh tiếp theo')

            continue

        else:

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

            result = pose.process(img\_rgb)

            if result.pose\_landmarks:

                lm\_list = []

                for landmarks in result.pose\_landmarks.landmark:

                    # Sơ chế

                    max\_distance = 0

                    lm\_list.append(landmarks)

                center\_x = (lm\_list[landmark\_names.index('right\_hip')].x +

                            lm\_list[landmark\_names.index('left\_hip')].x)\*0.5

                center\_y = (lm\_list[landmark\_names.index('right\_hip')].y +

                            lm\_list[landmark\_names.index('left\_hip')].y)\*0.5

                shoulders\_x = (lm\_list[landmark\_names.index('right\_shoulder')].x +

                               lm\_list[landmark\_names.index('left\_shoulder')].x)\*0.5

                shoulders\_y = (lm\_list[landmark\_names.index('right\_shoulder')].y +

                               lm\_list[landmark\_names.index('left\_shoulder')].y)\*0.5

                for lm in lm\_list:

                    distance = math.sqrt(

                        (lm.x - center\_x)\*\*2 + (lm.y - center\_y)\*\*2)

                    if(distance > max\_distance):

                        max\_distance = distance

                torso\_size = math.sqrt(

                    (shoulders\_x - center\_x)\*\*2 + (shoulders\_y - center\_y)\*\*2)

                max\_distance = max(

                    torso\_size\*torso\_size\_multiplier, max\_distance)

                pre\_lm = list(np.array([[(landmark.x-center\_x)/max\_distance, (landmark.y-center\_y)/max\_distance,

                              landmark.z/max\_distance,

                              #landmark.visibility

                              ] for landmark in lm\_list]).flatten())

                full\_lm\_list.append(pre\_lm)

                target\_list.append(class\_name)

            print(f'{os.path.split(img)[1]} Landmarks added Successfully')

    print(f'[INFO] {class\_name} Successfully Completed')

print('[INFO] Landmarks from Dataset Successfully Completed')

data\_x = pd.DataFrame(full\_lm\_list, columns=col\_names)

data = data\_x.assign(Pose\_Class=target\_list)

data.to\_csv(path\_to\_save, encoding='utf-8', index=False)

print(f'[INFO] Successfully Saved Landmarks data into {path\_to\_save}')

## **3.2. Khởi tạo Sequential Model và Training**

import keras

import pandas as pd

from keras import layers, Sequential

import argparse

from sklearn.model\_selection import train\_test\_split

from matplotlib import pyplot as plt

import os

ap = argparse.ArgumentParser()

ap.add\_argument("-i", "--dataset", type=str, required=True,

                help="path to csv Data")

ap.add\_argument("-o", "--save", type=str, required=True,

                help="path to save .h5 model, eg: dir/model.h5")

args = vars(ap.parse\_args())

path\_csv = args["dataset"]

path\_to\_save = args["save"]

# Load .csv Data

df = pd.read\_csv(path\_csv)

class\_list = df['Pose\_Class'].unique()

class\_list = sorted(class\_list)

class\_number = len(class\_list)

# Create training and validation splits

x = df.copy()

y = x.pop('Pose\_Class')

y, \_ = y.factorize()

x = x.astype('float64')

y = keras.utils.to\_categorical(y)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y,

                                                    test\_size=0.2,

                                                    random\_state=0) #Lấy 20% dữ liệu csv để test

print('[INFO] Loaded csv Dataset')

# Khởi tạo model tuần tự với 3 Layer

model = Sequential([

    layers.Dense(512, activation='relu', input\_shape=[x\_train.shape[1]]),

    layers.Dense(256, activation='relu'),

    layers.Dense(class\_number, activation="softmax")

])

# Model Summary

print('Model Summary: ', model.summary())

model.compile(

    optimizer='adam',

    loss='categorical\_crossentropy',

    metrics=['accuracy']

)

# Add a checkpoint callback to store the checkpoint that has the highest

# validation accuracy.

#checkpoint\_path = path\_to\_save

#checkpoint = keras.callbacks.ModelCheckpoint(checkpoint\_path,

#                                             monitor='val\_accuracy',

#                                            verbose=1,

#                                             save\_best\_only=True,

#                                             mode='auto')

#earlystopping = keras.callbacks.EarlyStopping(monitor='val\_accuracy',

#                                              patience=50)

print('[INFO] Model Training Started ...')

# Start training

history = model.fit(x\_train, y\_train,

                    epochs=200,

                    batch\_size=None,

                    validation\_data=(x\_test, y\_test),

                    #callbacks=[checkpoint, earlystopping]

                    )

print('[INFO] Model Training Completed')

print(f'[INFO] Model Successfully Saved in /{path\_to\_save}')

# Plot History

metric\_loss = history.history['loss']

metric\_val\_loss = history.history['val\_loss']

metric\_accuracy = history.history['accuracy']

metric\_val\_accuracy = history.history['val\_accuracy']

# Construct a range object which will be used as x-axis (horizontal plane) of the graph.

epochs = range(len(metric\_loss))

# Plot the Graph.

plt.plot(epochs, metric\_loss, 'blue', label=metric\_loss)

plt.plot(epochs, metric\_val\_loss, 'red', label=metric\_val\_loss)

plt.plot(epochs, metric\_accuracy, 'blue', label=metric\_accuracy)

plt.plot(epochs, metric\_val\_accuracy, 'green', label=metric\_val\_accuracy)

# Add title to the plot.

plt.title(str('Model Metrics'))

# Add legend to the plot.

plt.legend(['loss', 'val\_loss', 'accuracy', 'val\_accuracy'])

# If the plot already exist, remove

plot\_png = os.path.exists('metrics.png')

if plot\_png:

    os.remove('metrics.png')

    plt.savefig('metrics.png', bbox\_inches='tight')

else:

    plt.savefig('metrics.png', bbox\_inches='tight')

print('[INFO] Successfully Saved metrics.png')

## **3.3. Dự đoán tư thế người trong ảnh**

import os

from keras.models import load\_model

import cv2

import mediapipe as mp

import numpy as np

import pandas as pd

import math

import argparse

ap = argparse.ArgumentParser()

ap.add\_argument("-m", "--model", type=str, required=True,

                help="path to saved .h5 model, eg: dir/model.h5")

ap.add\_argument("-c", "--conf", type=float, required=True,

                help="min prediction conf to detect pose class (0<conf<1)")

ap.add\_argument("-i", "--source", type=str, required=True,

                help="path to sample image")

ap.add\_argument("--save", action='store\_true',

                help="Save video")

args = vars(ap.parse\_args())

source = args["source"]

path\_saved\_model = args["model"]

threshold = args["conf"]

save = args['save']

##############

torso\_size\_multiplier = 2.5

n\_landmarks = 33

n\_dimensions = 3

landmark\_names = [

    'nose',

    'left\_eye\_inner', 'left\_eye', 'left\_eye\_outer',

    'right\_eye\_inner', 'right\_eye', 'right\_eye\_outer',

    'left\_ear', 'right\_ear',

    'mouth\_left', 'mouth\_right',

    'left\_shoulder', 'right\_shoulder',

    'left\_elbow', 'right\_elbow',

    'left\_wrist', 'right\_wrist',

    'left\_pinky\_1', 'right\_pinky\_1',

    'left\_index\_1', 'right\_index\_1',

    'left\_thumb\_2', 'right\_thumb\_2',

    'left\_hip', 'right\_hip',

    'left\_knee', 'right\_knee',

    'left\_ankle', 'right\_ankle',

    'left\_heel', 'right\_heel',

    'left\_foot\_index', 'right\_foot\_index',

]

class\_names = [

    'Squats Up', 'Squats Down', 'Situp Up', 'Situp Down', 'Pushups Up', 'Pushups Down', 'Pullups Up', 'Pullups Down', 'Jumping Jack Up', 'Jumping Jack Down'

]

##############

mp\_pose = mp.solutions.pose

pose = mp\_pose.Pose()

col\_names = []

for i in range(n\_landmarks):

    name = mp\_pose.PoseLandmark(i).name

    name\_x = name + '\_X'

    name\_y = name + '\_Y'

    name\_z = name + '\_Z'

    #name\_v = name + '\_V'

    col\_names.append(name\_x)

    col\_names.append(name\_y)

    col\_names.append(name\_z)

    #col\_names.append(name\_v)

# Load saved model

model = load\_model(path\_saved\_model, compile=True)

if source.endswith(('.jpg', '.jpeg', '.png')):

    path\_to\_img = source

    # Load sample Image

    img = cv2.imread(path\_to\_img)

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

    result = pose.process(img\_rgb)

    if result.pose\_landmarks:

        lm\_list = []

        for landmarks in result.pose\_landmarks.landmark:

            # Preprocessing

            max\_distance = 0

            lm\_list.append(landmarks)

        center\_x = (lm\_list[landmark\_names.index('right\_hip')].x +

                    lm\_list[landmark\_names.index('left\_hip')].x)\*0.5

        center\_y = (lm\_list[landmark\_names.index('right\_hip')].y +

                    lm\_list[landmark\_names.index('left\_hip')].y)\*0.5

        shoulders\_x = (lm\_list[landmark\_names.index('right\_shoulder')].x +

                       lm\_list[landmark\_names.index('left\_shoulder')].x)\*0.5

        shoulders\_y = (lm\_list[landmark\_names.index('right\_shoulder')].y +

                       lm\_list[landmark\_names.index('left\_shoulder')].y)\*0.5

        for lm in lm\_list:

            distance = math.sqrt((lm.x - center\_x)\*\*2 + (lm.y - center\_y)\*\*2)

            if(distance > max\_distance):

                max\_distance = distance

        torso\_size = math.sqrt((shoulders\_x - center\_x) \*\*

                               2 + (shoulders\_y - center\_y)\*\*2)

        max\_distance = max(torso\_size\*torso\_size\_multiplier, max\_distance)

        pre\_lm = list(np.array([[(landmark.x-center\_x)/max\_distance, (landmark.y-center\_y)/max\_distance,

                                 landmark.z/max\_distance ,

                                 #landmark.visibility

                                 ] for landmark in lm\_list]).flatten())

        data = pd.DataFrame([pre\_lm], columns=col\_names)

        predict = model.predict(data)[0]

        if max(predict) > threshold:

            pose\_class = class\_names[predict.argmax()]

            print('predictions: ', predict)

            print('predicted Pose Class: ', pose\_class)

        else:

            pose\_class = 'Unknown Pose'

            print('[INFO] Predictions is below given Confidence!!')

    # Show Result

    img = cv2.putText(

        img, f'{class\_names[predict.argmax()]}',

        (40, 50), cv2.FONT\_HERSHEY\_PLAIN,

        2, (255, 0, 255), 2

    )

    if save:

        os.makedirs('ImageOutput', exist\_ok=True)

        img\_full\_name = os.path.split(path\_to\_img)[1]

        img\_name = os.path.splitext(img\_full\_name)[0]

        path\_to\_save\_img = f'ImageOutput/{img\_name}.jpg'

        cv2.imwrite(f'{path\_to\_save\_img}', img)

        print(f'[INFO] Output Image Saved in {path\_to\_save\_img}')

    cv2.imshow('Output Image', img)

    if cv2.waitKey(0) & 0xFF == ord('q'):

        cv2.destroyAllWindows()

    print('[INFO] Inference on Test Image is Ended...')

else:

    # Web-cam

    if source.isnumeric():

        source = int(source)

    cap = cv2.VideoCapture(source)

    source\_width = int(cap.get(3))

    source\_height = int(cap.get(4))

    # Write Video

    if save:

        out\_video = cv2.VideoWriter('output.avi',

                            cv2.VideoWriter\_fourcc(\*'MJPG'),

                            10, (source\_width, source\_height))

    while True:

        success, img = cap.read()

        if not success:

            print('[ERROR] Failed to Read Video feed')

            break

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

        result = pose.process(img\_rgb)

        if result.pose\_landmarks:

            lm\_list = []

            for landmarks in result.pose\_landmarks.landmark:

                # Preprocessing

                max\_distance = 0

                lm\_list.append(landmarks)

            center\_x = (lm\_list[landmark\_names.index('right\_hip')].x +

                        lm\_list[landmark\_names.index('left\_hip')].x)\*0.5

            center\_y = (lm\_list[landmark\_names.index('right\_hip')].y +

                        lm\_list[landmark\_names.index('left\_hip')].y)\*0.5

            shoulders\_x = (lm\_list[landmark\_names.index('right\_shoulder')].x +

                           lm\_list[landmark\_names.index('left\_shoulder')].x)\*0.5

            shoulders\_y = (lm\_list[landmark\_names.index('right\_shoulder')].y +

                           lm\_list[landmark\_names.index('left\_shoulder')].y)\*0.5

            for lm in lm\_list:

                distance = math.sqrt((lm.x - center\_x) \*\*

                                     2 + (lm.y - center\_y)\*\*2)

                if(distance > max\_distance):

                    max\_distance = distance

            torso\_size = math.sqrt((shoulders\_x - center\_x) \*\*

                                   2 + (shoulders\_y - center\_y)\*\*2)

            max\_distance = max(torso\_size\*torso\_size\_multiplier, max\_distance)

            pre\_lm = list(np.array([[(landmark.x-center\_x)/max\_distance, (landmark.y-center\_y)/max\_distance,

                                     landmark.z/max\_distance,

                                     #landmark.visibility

                                     ]for landmark in lm\_list]).flatten())

            data = pd.DataFrame([pre\_lm], columns=col\_names)

            predict = model.predict(data)[0]

            if max(predict) > threshold:

                pose\_class = class\_names[predict.argmax()]

                print('predictions: ', predict)

                print('predicted Pose Class: ', pose\_class)

            else:

                pose\_class = 'Unknown Pose'

                print('[INFO] Predictions is below given Confidence!!')

            # Show Result

            img = cv2.putText(

                img, f'{pose\_class}',

                (40, 50), cv2.FONT\_HERSHEY\_PLAIN,

                2, (255, 0, 255), 2

            )

        # Write Video

        if save:

            out\_video.write(img)

        cv2.imshow('Output Image', img)

        if cv2.waitKey(1) & 0xFF == ord('q'):

            break

    cap.release()

    if save:

        out\_video.release()

        print("[INFO] Out video Saved as 'output.avi'")

    cv2.destroyAllWindows()

    print('[INFO] Inference on Videostream is Ended...')

## **3.4. Chạy chương trình**

Để sử dụng chương trình trên Visual Studio Code, ta cần làm các bước sau:

### **3.4.1. Download bộ dữ liệu hình ảnh**

Bước đầu tiên đó là tạo ra một thư mục Dataset chứa các thư mục con là các hình ảnh của lớp tương ứng. Cấu trúc của thư mục như sau:

Filedata

Chongday

chongday1.jpg

chongday2.jpg

…

Hitdat

hitdat1.jpg

hitdat2.jpg

…

Squat

squat1.jpg

squat2.jpg

…

Vậy thư mục sẽ có tên là Filedata, các thư mục con tương ứng với 3 lớp cần phân biệt đó là Chongday, Hitdat và Squat.

### **3.4.2. Tạo file CSV với các lớp tương ứng**

Bước tiếp theo ta sẽ sử dụng câu lệnh trong Terminal. Mở Visual Studio Code, chọn Terminal -> New Terminal

A screenshot of a computer

Description automatically generated with medium confidence

***Hình 11. Terminal -> New Terminal***

Powershell sẽ hiện ra. Bây giờ ta sẽ gõ lệnh:

python poseLandmark\_csv.py -i [đường dẫn đến thư mục ảnh Filedata] -o [đường dẫn đến file csv]

Ví dụ:

python poseLandmark\_csv.py -i pictures/Filedata -o /data.csv

Sau khi chạy xong, sẽ xuất ra file CSV có tên là data.csv với đường dẫn vừa tạo trong đoạn code trên.

### **3.4.3. Thực hiện train file CSV và xuất ra biểu đồ**

Sau khi chạy xong đoạn lệnh tạo file CSV, ta tiếp tục đánh dòng lệnh sau vào Powershell Terminal:

python poseModel.py -i [đường dẫn đến file csv]-o [đường dẫn đến file model]

Ví dụ:

python poseModel.py -i /data.csv -o MoHinh/model.h5

Chương trình sẽ đọc file CSV với đường dẫn như trên, train 100 lần theo mô hình nơ-ron truyền thẳng. Cuối cùng xuất ra file model.h5 trong thư mục có tên MoHinh. Biểu đồ sẽ được chương trình vẽ và lưu trong ảnh có tên metrics.png

Chart, line chart

Description automatically generated

***Hình 12. Biểu đồ được vẽ và lưu vào ảnh metrics.png***

### **3.4.4. Nhận diện ảnh**

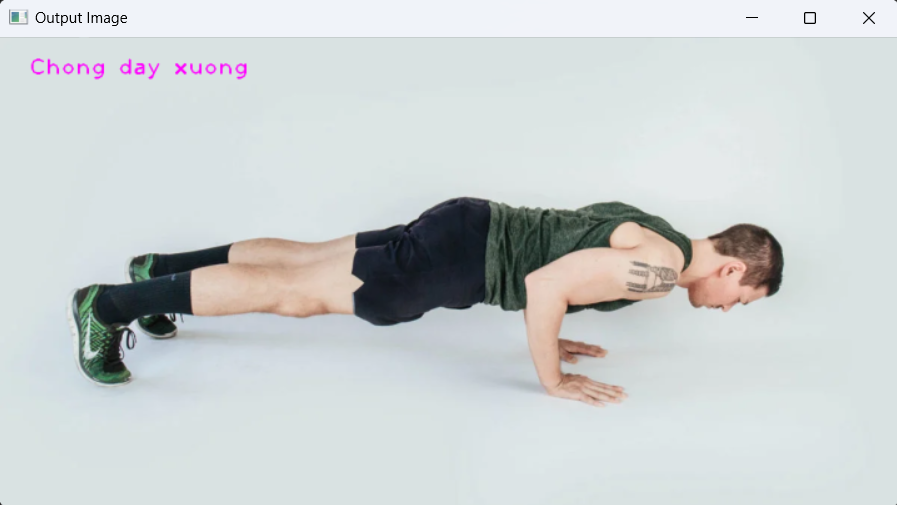
Cuối cùng là nhập hình ảnh vào chương trình để nhận diện. Ta tiếp dục dùng Powershell Terminal để gõ lệnh:

python inference.py --model [đường dẫn đến file model.h5] --conf 0.75 --source [đường dẫn đến hình ảnh cần nhận diện]

Ví dụ:

python inference.py --model model.h5 --conf 0.75 –source HinhAnh/chongday.jpg

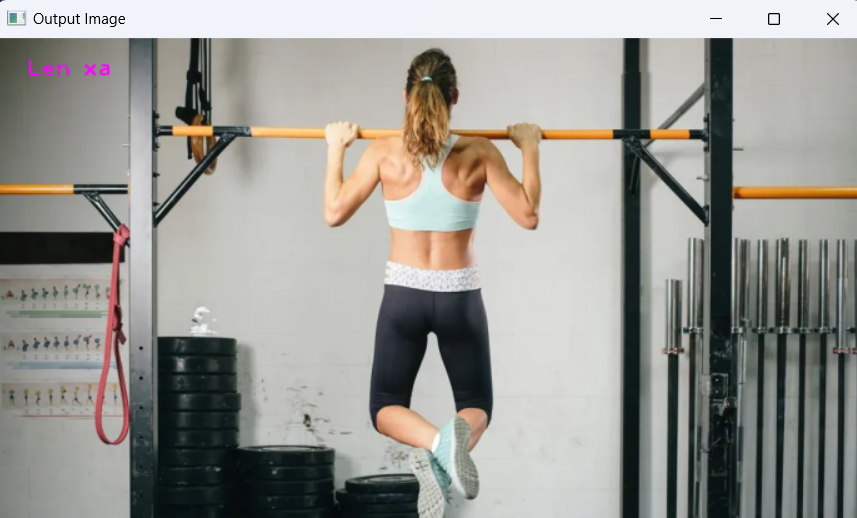
Chương trình sẽ đọc file model.h5 vừa train và thực hiện nhận diện ảnh với đường dẫn là hình ảnh tên chongday.jpg trong thư mục HinhAnh. Cuối cùng xuất ra hình ảnh chongday.jpg với dòng chữ tên hoạt động ở góc trái trên của ảnh.



***Hình 13. Hình ảnh sau khi nhận diện***

### **3.4.5. Kết quả chạy chương trình**





***Hình 14 + 15. Kết quả của chương trình***

# **CHƯƠNG IV. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## **4.1. Nhận xét và kết luận**

Đề tài **“Nhận dạng hoạt động thể chất”** nói chung và các đề tài ứng dụng học sâu (Deep Learning) là đề tài có tính ứng dụng trong thực tế. Có thể kể một số ứng dụng như nhận dạng tiếng nói, chữ viết tay, cử chỉ, biển số xe,… Đây là bài tập giúp sinh viên có thể hiểu hơn về lĩnh vực học máy và thị giác máy.

Chương trình đã phân loại được 5 hoạt động thể chất cơ bản với độ chính xác trên 95%. Với một số góc chụp bị khuất hoặc các bức ảnh nhiều người cùng thực hiện động tác, chương trình không thể dự đoán chính xác. Đề tài lấy nguồn dữ liệu CSV từ trang web Kaggle và đã ứng dụng thành công vào chương trình.

## **4.2. Hướng phát triển của đề tài**

Đề tài sẽ phát triển thêm tính năng đọc video đầu vào và cuối cùng là sử dụng camera để nhận dạng hoạt động trong thời gian thực. Ngoài ra đề tài cũng sẽ có giao diện phần mềm và trang web để người dùng có thể dễ dàng sử dụng.

Trong quá trình làm bài tập lớn, rất khó để tránh khỏi các sai sót không đáng có. Em xin cảm ơn cô **Đặng Thị Hiên** và các bạn đã giúp đỡ em trong quá trình nghiên cứu và hoàn thành bài tập.

# **TÀI LIỆU THAM KHẢO**

<https://github.com/naseemap47/CustomPose-Classification-Mediapipe>

<https://www.kaggle.com/datasets/muhannadtuameh/exercise-recognition>

<https://miai.vn/2022/02/14/nhan-dien-hanh-vi-con-nguoi-bang-mediapipe-pose-va-lstm-model-mi-ai/>

<https://keras.io/>

<https://mediapipe.dev/>

<https://www.tensorflow.org/?hl=vi>

<https://tek4.vn/api-mo-hinh-tuan-tu-sequential-don-gian-keras-co-ban>

<https://viblo.asia/p/lam-quen-voi-keras-gGJ59mxJ5X2>

<https://topdev.vn/blog/machine-learning-la-gi/>

<https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset>

<https://www.kaggle.com/datasets/ujjwalchowdhury/yoga-pose-classification>