TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

**KHOA ĐIỆN TỬ**

**BỘ MÔN: TIN HỌC CÔNG NGHIỆP**



**BÀI TẬP LỚN**

**TRÍ TUỆ NHÂN TẠO**

*Đề tài: Nhận dạng hoạt động thể chất*

Sinh viên: Đặng Hoàng Nam

Lớp: K55KMT

Giáo viên hướng dẫn: Nguyễn Tuấn Linh

**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 TRÍ TUỆ NHÂN TẠO**

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: 11/01/2023 Ngày hoàn thành: 24/02/2023*

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. Ảnh đầu vào là ảnh hoặc video của người đang tập thể dục***

***Hình 2. Bộ dữ liệu phân biệt 5 hoạt động thể chất***

***Hình 3. Tọa độ X, Y, Z của các điểm tư thế người trong file CSV***

***Hình 4. Ảnh kết quả là hình ảnh gốc với tên động tác dự đoán***

***Hình 5. Hơn 1000 dữ liệu về tọa độ các mốc tư thế***

***Hình 6. 33 mốc tư thế trên cơ thể người***

***Hình 7. Mô hình tọa độ mốc tư thế người trong không gian***

***Hình 8. Hình ảnh kết quả sau khi vẽ khung xương người Mediapipe***

***Hình 9. Mạng thần kinh truyền thẳng đơn giản***

***Hình 10. Keras và TensorFlow***

***Hình 11. Visual Studio Code***

***Hình 12. Ngôn ngữ lập trình Python***

***Hình 13. Terminal -> New Terminal***

***Hình 14. Biểu đồ được vẽ và lưu vào ảnh metrics.png***

***Hình 15. Hình ảnh sau khi nhận diện***

***Hình 16. Mô hình thực hiện training 100 lần***

***Hình 17. Biểu đồ Training Model***

***Hình 18 + 19: Kết quả của chương trình***

**MỤC LỤC**

[**LỜI MỞ ĐẦU** 7](#_Toc128163385)

[**CHƯƠNG I. GIỚI THIỆU CHUNG** 8](#_Toc128163386)

[**1.1.Đặt vấn đề** 8](#_Toc128163387)

[**1.2.Hướng giải quyết vấn đề** 9](#_Toc128163388)

[**CHƯƠNG II. PHÂN TÍCH BÀI TOÁN** 10](#_Toc128163389)

[**2.1. Phân tích yêu cầu** 10](#_Toc128163390)

[***2.1.1. Tạo bộ dữ liệu Landmark*** *10*](#_Toc128163391)

[***2.1.2. Khởi tạo mô hình mạng nơ-ron*** *11*](#_Toc128163392)

[***2.1.3. Sử dụng mô hình vừa tạo để suy luận ra tư thế người*** *11*](#_Toc128163393)

[**2.2. Phân tích dữ liệu** 12](#_Toc128163394)

[***2.2.1. Bộ dữ liệu của bài toán*** *12*](#_Toc128163395)

[***2.2.2. Xử lý dữ liệu bằng MediaPipe Pose*** *13*](#_Toc128163396)

[***2.2.3. Tọa độ mốc tư thế Pose Landmark*** *15*](#_Toc128163397)

[***2.2.4. Vẽ khung xương người bằng mô hình MediaPipe Pose Model*** *16*](#_Toc128163398)

[**CHƯƠNG III. PHƯƠNG PHÁP HỌC MÁY** 17](#_Toc128163399)

[**3.1.Mạng thần kinh truyền thẳng** 17](#_Toc128163400)

[**3.2. Thư viện mạng nơ-ron Keras** 18](#_Toc128163401)

[**3.3. Mô hình tuần tự Sequential Model** 18](#_Toc128163402)

[**CHƯƠNG IV. THỰC NGHIỆM** 20](#_Toc128163403)

[**4.1. Tiền xử lý dữ liệu** 20](#_Toc128163404)

[**4.2. Môi trường thực nghiệm** 20](#_Toc128163405)

[**4.3. Code chương trình** 22](#_Toc128163406)

[***4.3.1. Đọc dữ liệu ảnh và tạo file CSV*** *22*](#_Toc128163407)

[***4.3.2. Tạo mô hình Sequential Model và training dữ liệu*** *25*](#_Toc128163408)

[***4.3.3. Nhận dạng hình ảnh*** *27*](#_Toc128163409)

[**4.4. Các module chương trình** 32](#_Toc128163410)

[***4.4.1. Tiền xử lý dữ liệu*** *32*](#_Toc128163411)

[***4.4.2. Thực hiện train file CSV và xuất ra biểu đồ*** *33*](#_Toc128163412)

[***4.4.3. Nhận diện hình ảnh*** *34*](#_Toc128163413)

[**4.5. Huấn luyện và kiểm thử mô hình** 35](#_Toc128163414)

[**CHƯƠNG V. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 37](#_Toc128163415)

[**5.1. Kết quả đạt được và hạn chế** 37](#_Toc128163416)

[**5.2. Hướng phát triển của đề tài** 37](#_Toc128163417)

[**TÀI LIỆU THAM KHẢO** 38](#_Toc128163418)

# **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 em xin được trình bày quá trình thực hiện bài tập tiểu luận **“Nhận dạng hoạt động thể chất”** của môn học Trí tuệ nhân tạo. 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 bài tập 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 CHUNG**

## **1.1.Đặt vấn đề**

Mục tiêu của bài tập **“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**

Để giải quyết được vấn đề bài tập đặt ra, ta cần thực hiện các yêu cầu sau:

* Đọc được ảnh hoặc video về người đang thực hiện động tác thể dục
* Phân biệt được động tác thể dục trong ảnh, phát hiện được vị trí bắt đầu và kết thúc của bài tập
* Hiển thị ra tên của hoạt động, hoạt động đó là vị trí bắt đầu hay kết thúc



***Hình 1. Ảnh đầu vào là ảnh hoặc video của người đang tập thể dục***

## **1.2.Hướng giải quyết vấn đề**

Ta sử dụng bộ dữ liệu tư thế người tập thể dục để train chương trình. Bộ dữ liệu gồm hơn 1000 dữ liệu thông số đo được từ tư thế các động tác thể dục khác nhau. Chương trình sau khi train sẽ ghi lại các thông số vào bộ nhớ.

Chương trình 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 hình ảnh về người trong hình, đưa ra dữ liệu về tư thế của người đó và so sánh với bộ dữ liệu đã train từ trước. Từ đó đưa ra dự đoán về hoạt động thể dục trong ảnh.

Graphical user interface, text, application, email

Description automatically generated

***Hình 2. Bộ dữ liệu phân biệt 5 hoạt động thể chất***

# **CHƯƠNG II. PHÂN TÍCH BÀI TOÁN**

## **2.1. Phân tích yêu cầu**

Bài toán “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 mạng nơ-ron 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 3. Tọa độ X, Y, Z của các điểm tư thế người trong file CSV***

### **2.1.2. Khởi tạo mô hình mạng nơ-ron**

Chương trình 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.

A picture containing ground, outdoor, beach, sky

Description automatically generated

***Hình 4. Ảnh kết quả là hình ảnh gốc với tên động tác dự đoán***

## **2.2. Phân tích dữ liệu**

### **2.2.1. Bộ dữ liệu của bài toán**

Dữ liệu trong bài tập được lấy tại trang web Kaggle. Bộ dữ liệu này đại diện cho 10 tư thế thể chất khác nhau dùng để phân biệt 5 bài tập. Các bài tập là Chống đẩy, Hít đất, Lên xà, Jumping Jack và Squat. Đối với mỗi bài tập, 2 lớp khác nhau đã được sử dụng để biểu thị hai vị trí đầu và cuối của bài tập đó (Ví dụ: Hai vị trí “lên” và “xuống” của động tác Chống đẩy).

Khoảng 500 video về những người tập thể dục đã được sử dụng để thu thập dữ liệu này. Các video từ Bộ dữ liệu [Countix](https://www.deepmind.com/open-source/kinetics) có chứa các liên kết YouTube của một số video hoạt động của con người. Sử dụng một tập lệnh Python đơn giản, các video về 5 bài tập thể chất khác nhau được tải xuống. Từ mỗi video, ít nhất 2 khung hình được trích xuất thủ công. Các khung được trích xuất đại diện cho các vị trí đầu cuối của bài tập.

Đường link dẫn đến trang web bộ dữ liệu:

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

**Table

Description automatically generated**

***Hình 5. Hơn 1000 dữ liệu về tọa độ các mốc tư thế***

### **2.2.2. Xử lý dữ liệu bằng MediaPipe Pose**

Đối với mọi khung hình, khung [MediaPipe](https://google.github.io/mediapipe) được sử dụng để áp dụng ước tính tư thế, giúp phát hiện bộ xương người của người trong khung hình. Vậy Mediapipe là gì ?

**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.

Chart, radar chart

Description automatically generated

***Hình 6. 33 mốc tư thế trên cơ thể người***

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

### **2.2.3. Tọa độ mốc tư thế 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.

Trong bài toán “Nhận diện hoạt động thể chất”, ta sẽ sử dụng hai tọa độ là X và Y đẻ nhận diện ảnh 2D. Hai thông số tọa độ là Z và Visibility được sử dụng để nhận diện video sẽ được phát triển thêm.

Chart, line chart

Description automatically generated

***Hình 7. Mô hình tọa độ mốc tư thế người trong không gian***

### **2.2.4. Vẽ khung xương người bằng mô hình 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

A person running on a field

Description automatically generated with medium confidence

***Hình 8. Hình ảnh kết quả sau khi vẽ khung xương người Mediapipe***

# **CHƯƠNG III. PHƯƠNG PHÁP HỌC MÁY**

## **3.1.Mạng thần kinh truyền thẳng**

**Mạng thần kinh truyền thẳng** (hay còn gọi là **mạng nơ-ron truyền thẳng**, viết tắt là FNN) là một [mạng thần kinh nhân tạo](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_th%E1%BA%A7n_kinh_nh%C3%A2n_t%E1%BA%A1o) trong đó kết nối giữa các nút không hình thành một chu trình. Theo cách đó, FNN khác với hậu duệ của nó: các [mạng thần kinh hồi quy](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_th%E1%BA%A7n_kinh_h%E1%BB%93i_quy).

[Mạng thần kinh](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_th%E1%BA%A7n_kinh) truyền thẳng là loại mạng thần kinh nhân tạo đầu tiên và đơn giản nhất được phát minh ra.

Trong mạng này, thông tin di chuyển chỉ một chiều - hướng đến - từ các nút đầu vào, thông qua các nút ẩn (nếu có) và đi đến các nút đầu ra. Không có chu trình (chu kỳ) hoặc vòng lặp trong mạng.

A picture containing vector graphics

Description automatically generated

***Hình 9. Mạng thần kinh truyền thẳng đơn giản***

## **3.2. 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 10. Keras và TensorFlow***

## **3.3. Mô hình tuần tự Sequential Model**

API mô hình tuần tự (sequential) là đơn giản nhất. Nó được gọi là “sequential” vì nó liên quan đến việc xác định một lớp Sequential và thêm từng lớp vào mô hình theo cách tuyến tính, từ đầu vào đến đầu ra.

Một mô hình tuần tự Sequential Model phù hợp với một chồng các lớp đơn giản, trong đó mỗi lớp có chính xác một tensor đầu vào và một tensor đầu ra.

Chương trình 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.

Text

Description automatically generated

Ta cấu trình quy trình học của model qua 3 đối số: Tối ưu hóa Optimizer, Hàm mất mát loss và danh sách các số liệu metrics.

Text

Description automatically generated

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 (epochs).

Text

Description automatically generated

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…

# **CHƯƠNG IV. THỰC NGHIỆM**

## **4.1. Tiền xử lý dữ liệu**

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, mỗi thư mục con chứa các hình ảnh về các hoạt động của từng lớp.

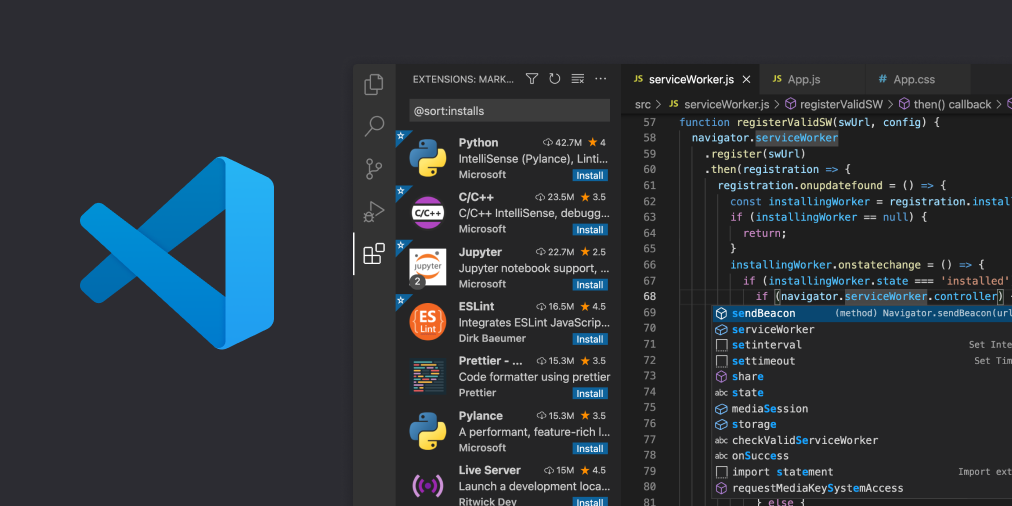
## **4.2. Môi trường thực nghiệm**

Chương trình chạy trên môi trường **Visual Studio Code**, ngôn ngữ lập trình được sử dụng là **Python**.

**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.

Visual Studio Code là một trình biên tập mã. 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).

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).



***Hình 11. Visual Studio Code***

**Python** là một ngôn ngữ lập trình được sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học (ML). Các nhà phát triển sử dụng Python vì nó hiệu quả, dễ học và có thể chạy trên nhiều nền tảng khác nhau. Phần mềm Python được tải xuống miễn phí, tích hợp tốt với tất cả các loại hệ thống và tăng tốc độ phát triển.

Các đặc điểm sau tạo nên sự độc đáo của ngôn ngữ lập trình Python:

* **Là một ngôn ngữ thông dịch:** Ngôn ngữ này trực tiếp chạy từng dòng mã. Nếu có lỗi trong mã chương trình, nó sẽ ngừng chạy. Do đó, lập trình viên có thể nhanh chóng tìm ra lỗi trong đoạn mã.
* **Dễ sử dụng:** Python sử dụng từ ngữ giống trong tiếng Anh. Không giống như các ngôn ngữ lập trình khác, Python không sử dụng dấu ngoặc ôm. Thay vào đó, ngôn ngữ này sử dụng thụt đầu dòng.
* **Là một ngôn ngữ linh hoạt:** Các lập trình viên không cần phải khai báo loại biến khi viết mã bởi vì Python sẽ xác định chúng vào thời điểm chạy. Vì vậy, bạn có thể viết các chương trình Python một cách nhanh chóng hơn.
* **Là ngôn ngữ cấp cao:** Python gần gũi với ngôn ngữ con người hơn các ngôn ngữ lập trình khác. Do đó, các lập trình viên không cần phải lo lắng về những chức năng cơ bản của nó như kiến trúc và quản lý bộ nhớ.
* **Là ngôn ngữ lập trình hướng đối tượng:** Python coi mọi thứ đều là đối tượng, nhưng ngôn ngữ này cũng hỗ trợ các phương thức lập trình khác như lập trình hàm và lập trình cấu trúc.

Logo, company name

Description automatically generated

***Hình 12. Ngôn ngữ lập trình Python***

## **4.3. Code chương trình**

### **4.3.1. Đọc dữ liệu ảnh và tạo 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}')

### **4.3.2. Tạo mô hình Sequential Model và training dữ liệu**

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=100,

                    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, 'red', 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')

### **4.3.3. Nhận dạng hình 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 Len', 'Squats Xuong', 'Gap bung len', 'Gap bung xuong', 'Chong day len', 'Chong day xuong', 'Len xa', 'Xuong xa', 'Jumping Jack len', 'Jumping Jack xuong'

]

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

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}')

    scale\_percent = 60 # percent of original size

    width = int(img.shape[1] \* scale\_percent / 100)

    height = int(img.shape[0] \* scale\_percent / 100)

    dim = (width, height)

    # resize image

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

    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...')

## **4.4. Các module chương trình**

### **4.4.1. Tiền xử lý dữ liệu**

Mở Visual Studio Code, chọn Terminal -> New Terminal

A screenshot of a computer

Description automatically generated with medium confidence

***Hình 13. 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.

### **4.4.2. 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 14. Biểu đồ được vẽ và lưu vào ảnh metrics.png***

### **4.4.3. Nhận diện hình ảnh**

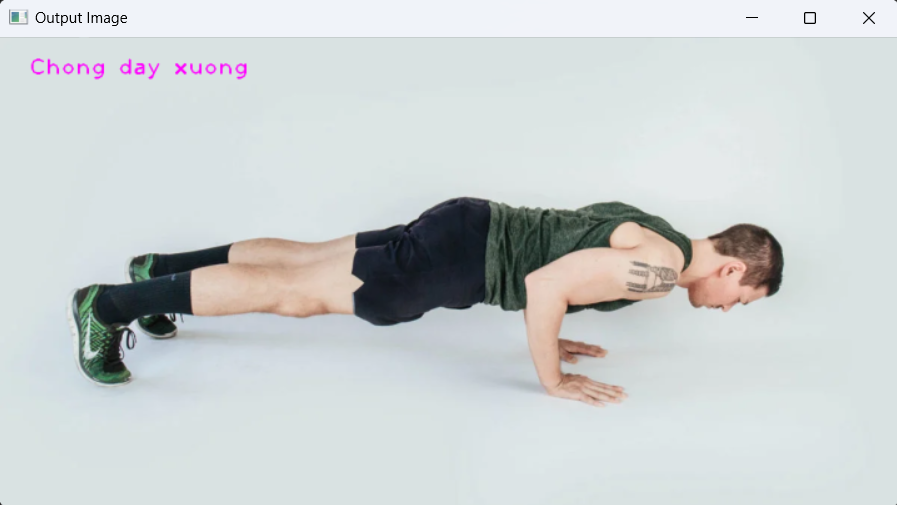
Cuối cùng là nhập hình ảnh vào chương trình để nhận diện. Ta tiếp dục dùng Powershell Terminal để gõ lệnh:

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 15. Hình ảnh sau khi nhận diện***

## **4.5. Huấn luyện và kiểm thử mô hình**

Sau khi thực hiện training mô hình 100 lần, kết quả cho ra độ chính xác cao (trên 95%).

Graphical user interface, text

Description automatically generated

***Hình 16. Mô hình thực hiện training 100 lần***

Sau khi training, chương trình xuất ra biểu đồ biểu thị độ chính xác và sai số của từng lần training. Có thể thấy rằng những lần training đầu tiên sẽ có sai số cao nhất. Đến lần train thứ 40 chương trình đã có độ ổn định. Lần train 60 – 80 có sự biến động với sai số.

Chart

Description automatically generated

***Hình 17. Biểu đồ Training Model***

Thử nghiệm chương trình với các hình ảnh tập thể dục, ta có các kết quả trả về như sau:

A picture containing text, person, indoor

Description automatically generated

A picture containing person, person, female

Description automatically generated

***Hình 18 + 19: Kết quả của chương trình***

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

## **5.1. Kết quả đạt được và hạn chế**

Đề tài **“Nhận dạng hoạt động thể chất”** nói chung và các đề tài ứng dụng mạng nơ-ron 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 trí tuệ nhân tạo và mạng nơ-ron nhân tạo.

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.

## **5.2. 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 tiểu luận, rất khó để tránh khỏi các sai sót không đáng có. Em xin cảm ơn thầy **Nguyễn Tuấn Linh** 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>