

ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN - ĐIỆN TỬ
BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG



ĐỒ ÁN TỐT NGHIỆP

POSE ESTIMATION AND VERIFICATION FOR YOGA AT HOME

COMPUTER ENGINEERING

Sinh viên: **Lâm Tấn Phát**

MSSV: 20119264

Nguyễn Trần Thành Trung

MSSV: 2011302

Hướng dẫn: TS. **Huỳnh Thế Thiện**

TP. HỒ CHÍ MINH, 08/2023

ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA ĐIỆN - ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG



HCMUTE

ĐỒ ÁN TỐT NGHIỆP

POSE ESTIMATION AND VERIFICATION FOR YOGA AT HOME

COMPUTER ENGINEERING

Sinh viên: **Lâm Tấn Phát**

MSSV: 20119264

Nguyễn Trần Thành Trung

MSSV: 2011302

Hướng dẫn: TS. **Huỳnh Thế Thiện**

TP. HỒ CHÍ MINH, 08/2023

LỜI CẢM ƠN

Trong quá trình chúng em hoàn thành đồ án môn học 2 với đề tài "Pose estimation and verification for yoga at home", chúng em đã nhận được rất nhiều sự tư vấn và lời khuyên từ Thầy Huỳnh Thế Thiện. Chúng em xin chân thành cảm ơn!

Nhóm thực hiện đồ án môn học
(*Ký và ghi rõ họ tên*)

Nguyễn Trần Thành Trung Lâm Tấn Phát

LỜI CAM ĐOAN

Nhóm thực hiện đồ án môn học 2 cam đoan đề tài thực hiện dựa vào một số tài liệu trước đó và không sao chép nội dung, kết quả của đồ án khác. Các nội dung tham khảo đã được trích dẫn đầy đủ.

Nhóm thực hiện đồ án môn học
(Ký và ghi rõ họ tên)

Nguyễn Trần Thành Trung Lâm Tấn Phát

TÓM TẮT NỘI DUNG

Trong bối cảnh dịch bệnh COVID-19, việc tập thể dục đã trải qua những sự biến đổi đáng kể. Khi đó việc tập luyện thể dục tại nhà đã trở thành lựa chọn phổ biến trong khoảng thời gian hạn chế ra khỏi nhà. Tuy nhiên, việc thực hiện các bài tập một cách đúng có thể gặp khó khăn đối với những người mới bắt đầu nếu thiếu sự hướng dẫn từ người huấn luyện chuyên nghiệp, điều này tăng nguy cơ chấn thương không mong muốn. Do vậy, cần phải có các hệ thống hỗ trợ giám sát hiệu suất tập thể dục để giúp ngăn ngừa những chấn thương.

Trong nghiên cứu này, chúng em trình bày một phương pháp tiếp cận tiên tiến để phát hiện chính xác các tư thế yoga bằng cách sử dụng kỹ thuật ước tính tư thế với sự hỗ trợ của thư viện OpenCV và mediapipe. Để đảm bảo việc đo và điều chỉnh tư thế cơ thể một cách chính xác trong quá trình tập luyện, giải pháp mà chúng em đề xuất sẽ tổ hợp các thuật toán dựa trên mô hình máy học cùng với thị giác máy tính.

Thông qua việc kết hợp sức mạnh của các kỹ thuật máy học và thị giác máy tính, chúng em hy vọng rằng phương pháp này sẽ cung cấp cho những người tập luyện một công cụ hữu ích để giúp họ thực hiện các tư thế yoga một cách chính xác và an toàn.

Mục lục

1	TỔNG QUAN	1
1.1	Giới thiệu	1
1.2	Mục tiêu	2
1.3	Tình hình nghiên cứu	2
1.4	Phương pháp nghiên cứu	3
1.5	Bố cục nội dung	3
2	CƠ SỞ LÝ THUYẾT	4
2.1	Máy học (Machine Learning)	4
2.2	Các mô hình máy học	5
2.2.1	Hồi quy logistic (Logistic Regression)	5
2.2.2	Bộ phân loại Ridge (Ridge Classifier)	6
2.2.3	Random Forest Classifier	7
2.2.4	Gradient Boosting Classifier	8
2.2.5	So sánh các mô hình máy học	9
2.3	Các thư viện cần thiết	10
2.3.1	Mediapipe	10
2.3.2	OpenCV	11
2.3.3	CSV	11
2.3.4	Os	12
2.3.5	Numpy	12
2.3.6	Pickle	13
2.3.7	Pandas	13

2.3.8	Sklearn	14
3	THIẾT KẾ HỆ THỐNG	15
3.1	Cơ chế hoạt động	15
3.2	Chi tiết hệ thống	16
3.2.1	Tải các thư viện cần thiết và chạy Mediapipe	16
3.2.2	Thu thập và lưu trữ điểm đặc trưng	19
3.3	Huấn luyện mô hình phân loại	23
3.3.1	Xử lý dữ liệu thông qua pandas	23
3.3.2	Huấn luyện mô hình	24
3.4	Đánh giá và lưu trữ mô hình	27
3.5	Phân loại mô hình	28
4	KẾT QUẢ	30
5	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	33
5.1	Kết luận	33
5.2	Hướng phát triển	34

Danh sách bảng

Danh sách hình vẽ

2.1	Đồ thị hàm sigmoid	6
2.2	Nguyên lí hoạt động	8
3.1	Sơ đồ khối cách hoạt động của mô hình	15
3.2	Thư viện Mediapipe và OpenCV	16
3.3	Tạo biến để vẽ các điểm (points)	17
3.4	Lấy khung hình từ webcam	17
3.5	Hiển thị và nối các landmarks trên cơ thể	18
3.6	Thư viện CSV, Os, Numpy	19
3.7	Thư viện CSV, Os, Numpy	19
3.8	Mô hình pose landmarks	20
3.9	Lưu trữ mô hình vào file csv	20
3.10	Tạo file csv	21
3.11	Dữ liệu được ghi trong file csv	21
3.12	Đặt nhãn cho động tác Bound Angle	22
3.13	Nhận diện động tác	22
3.14	Thư viện Pandas và Sklearn	23
3.15	Đọc dữ liệu từ file csv	23
3.16	Tách dữ liệu đặc trưng và nhãn tên	23
3.17	Chia dữ liệu thành 2 tập	24
3.18	Import 4 mô hình máy học	24
3.19	Tạo mảng pipeline	25
3.20	Tạo mảng fitmodels	26

3.21	Chạy fitmodels với mô hình rf	26
3.22	Tính độ chính xác của mô hình	27
3.23	Chạy fitmodels với 4 mô hình	27
3.24	Module Pickle	28
3.25	Mở tệp body-language.pkl	28
3.26	Dùng mô hình đã được huấn luyện để dự đoán dáng tập yoga	29
4.1	Kết quả trả về Tree Pose	31
4.2	Kết quả trả về Goddess Pose	32

Danh mục các từ viết tắt

Dưới đây là danh mục các từ viết tắt được sử dụng trong luận văn.

Các từ viết tắt	Định nghĩa
ML	Machine Learning
FPS	Frame per second
API	Application Programming Interface
cn	Confidence score
wn	Weak learner

Chương 1

TỔNG QUAN

1.1 Giới thiệu

Một trong những lĩnh vực được các nhà nghiên cứu và quan tâm hàng đầu trong nhiều năm gần đây được biết đến như nhận diện vật thể, con người thông qua thị giác máy tính do khả năng ứng dụng linh hoạt. Nhận diện hoạt động, tư thế của con người tác động đến các lĩnh vực khác nhau như tương tác người-máy, robot, trò chơi thực tế ảo, giám sát thông qua video, phân tích hoạt động trong thể thao, theo dõi sức khỏe và một số lĩnh vực liên quan. Gần đây, các hệ thống nhận dạng có những bước tiến nhảy vọt, điển hình nhất là công nghệ video 3D áp dụng trong bóng đá và có thể thay thế vị trí của công nghệ VAR. Mặt khác việc sử dụng Machine Learning nhận dạng các tư thế trong yoga vẫn đang trong quá trình nghiên cứu và phát triển. Dù đã có một số nỗ lực trong việc phát triển nhưng vẫn cần thêm nhiều nghiên cứu và phát triển để cải thiện độ chính xác và độ tin cậy của các mô hình. Đặc biệt, việc xử lý các tư thế yoga phức tạp và có sự thay đổi trong môi trường, ánh sáng, góc chụp và phong nền nơi hình ảnh được chụp là thách thức lớn cho các nhà nghiên cứu và nhà phát triển. [1]

Yoga không chỉ là một hình thức rèn luyện thể chất mà còn là một bộ môn phát triển tâm hồn và tinh thần. [2] Bộ môn này giúp cải thiện sự tập trung, tâm trí được thư giãn, giảm căng thẳng, và nhờ đó sức khỏe sẽ được cải thiện. Hiện tại yoga đang trở nên phổ biến trên toàn thế giới. Ý nghĩa ngày càng tăng của yoga trong y học là do tác dụng chữa bệnh phi thường trong nhiều bệnh lý ảnh hưởng đến cơ thể con người như các vấn đề về

hô hấp, bệnh tim, các vấn đề về cơ xương và ứng dụng học sâu trong chăm sóc sức khỏe. Tuy nhiên, tồn tại khoảng cách giữa sự hiểu biết và nhận thức của một số người về lợi ích của yoga, điều này dẫn đến hàng loạt các vấn đề sức khỏe. Bên cạnh đó có thể hạn chế chúng bằng cách áp dụng yoga như một phần trong thói quen hằng ngày. Áp dụng các công nghệ ngày càng tiến bộ mang đến khả năng giải quyết vấn đề, tăng khả năng tương tác của người dùng đến bài học thông qua công cụ hỗ trợ tự học theo thời gian thực, có thể phát hiện được các động tác yoga khác nhau.

1.2 Mục tiêu

Mục tiêu chính của đề tài là triển khai mô hình máy học để ước tính và nhận diện được các động tác yoga. Tạo các điểm (point) dựa theo các khớp xương của con người [3] trong hình ảnh, video thu thập thông qua camera của thiết bị và phản hồi kết quả động tác của người thực hiện. Trong đề tài này nhóm chúng em nghiên cứu một số động tác yoga tiêu biểu như Balasana (Child's Pose), Baddha Konasana (Bound Angle Pose), Utkata Konasana (Goddess Pose), Kumbhakasana (Plank Pose), Vrksasana (Tree Pose), Virabhadrasana II (Warrior II Pose).

1.3 Tình hình nghiên cứu

Về ước tính và xác minh tư thế Yoga đã trở thành một lĩnh vực quan trọng và đang nhận được sự quan tâm rộng rãi từ cộng đồng nghiên cứu và cả người học Yoga. Các nỗ lực nghiên cứu trong lĩnh vực này tập trung vào việc phát triển các phương pháp và công cụ để tự động ước tính và xác minh độ chính xác của các tư thế Yoga, mang lại nhiều lợi ích cho cả giảng dạy và tập luyện tại nhà.

1.4 Phương pháp nghiên cứu

Để đạt được mục tiêu của đề tài, chúng em sử dụng các phương pháp thu thập số liệu, thực nghiệm và phân tích tổng kết kinh nghiệm.

- Phương pháp thu thập số liệu:
 - Sử dụng phương pháp thu thập tài liệu: Nhóm em tìm những tài liệu liên quan đến chủ đề của mình để xây dựng cơ sở lý thuyết.
- Phương pháp thực nghiệm:
 - Thiết kế và triển khai hệ thống: Từ những thông tin đã được thu thập, nhóm em tiến hành chọn lọc, tổng hợp và cài đặt phần mềm, thư viện để thực hiện đề tài.
 - Tiến hành thử nghiệm: Nhóm em thực hiện các bài kiểm tra, chạy thử nghiệm mô hình và sửa lỗi.
- Phương pháp phân tích:
 - Đánh giá hiệu quả: Chúng em tiến hành đánh giá hiệu quả dựa trên độ chính xác của mô hình sau khi được huấn luyện.
 - Phân tích dữ liệu: Chúng em phân tích dữ liệu dựa vào thời gian thực nghiệm khi thay đổi các động tác và thay đổi đối tượng thực hiện động tác.

1.5 Bố cục nội dung

Trong bài báo cáo đồ án 2 bao gồm 5 chương:

- Chương 1: Tổng quan
- Chương 2: Cơ sở lý thuyết
- Chương 3: Thiết kế hệ thống
- Chương 4: Kết quả
- Chương 5: Kết luận và hướng phát triển

Chương 2

CƠ SỞ LÝ THUYẾT

2.1 Máy học (Machine Learning)

Machine Learning được xem như một nhánh của trí tuệ nhân tạo. Lĩnh vực này tập trung vào nghiên cứu, phát triển mô hình máy tính có khả năng học hỏi dựa trên dữ liệu mẫu hoặc những dữ liệu đã được học. Machine Learning có thể tự dự đoán và đưa ra quyết định mà không cần phải lập trình cụ thể. Từ đó tự động cải thiện hiệu suất trong công việc. [4]

Bài toán đặt ra cho Machine Learning xử lý chia làm 2 loại chính: Dự đoán (prediction) và phân loại (classification). Ví dụ phân loại chữ viết, nhận diện vật dụng,...

Các thuật toán phân loại cũng đối diện với vấn đề underfitting và overfitting tương tự như trong bài toán hồi quy. Vì vậy, việc thử nhiều phương pháp với các cách chọn đặc trưng và mức độ chính quy hóa khác nhau là cần thiết để tìm ra phương án tốt nhất.

Có nhiều cách để đánh giá hiệu suất của một phương pháp phân loại. Một cách đơn giản là sử dụng một tập dữ liệu thử nghiệm và tính tỉ lệ dự đoán đúng trên tập này. Phương pháp nào có tỉ lệ dự đoán đúng cao nhất sẽ được ưu tiên lựa chọn.

Từ đó nhóm đã đưa ra 4 mô hình máy học để sử dụng

2.2 Các mô hình máy học

2.2.1 Hồi quy logistic (Logistic Regression)

Logistic Regression là một thuật toán phân loại được sử dụng trong Machine Learning để dự đoán hoặc phân loại các đối tượng vào một tập hợp giá trị rời rạc. Thường, thuật toán này được áp dụng trong các bài toán phân loại nhị phân, tức là phân loại đối tượng vào hai nhóm khác nhau.

Thuật toán Logistic Regression hoạt động bằng cách tính toán giá trị dự đoán dựa trên các biến đầu vào (các biến độc lập). Kết quả dự đoán được biểu diễn dưới dạng xác suất rơi vào một nhóm cụ thể. Sau đó, giá trị dự đoán xác suất này được chuyển đổi thành các giá trị phân loại rời rạc (như 0 hoặc 1) thông qua một ngưỡng xác định trước. [5]

Phương trình toán học của Logistic Regression cho bài toán phân loại nhị phân (hai lớp) có dạng như sau:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Trong đó:

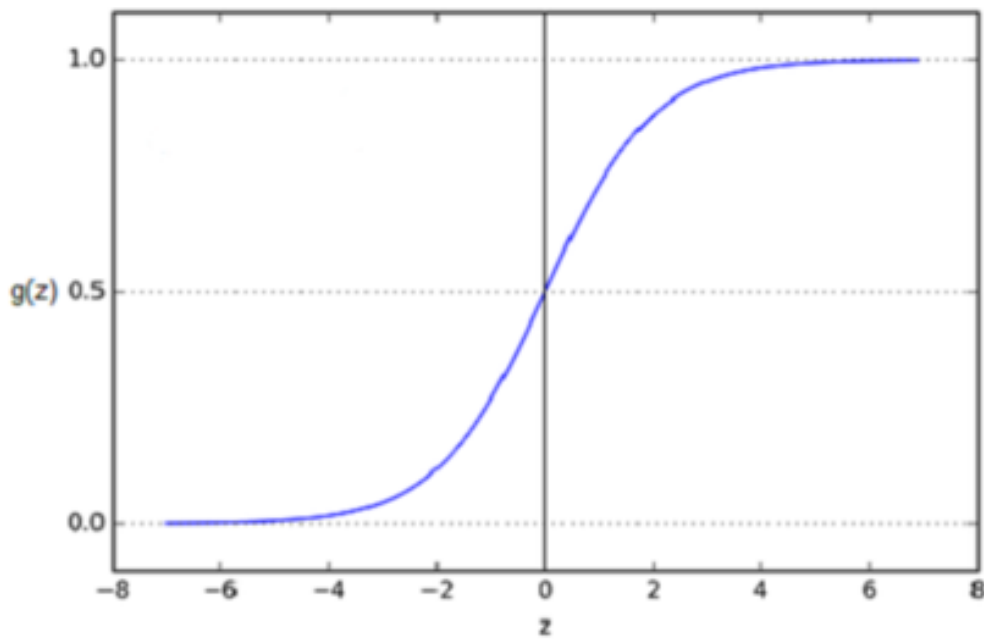
$g(z)$ là xác suất dự đoán rằng mẫu thuộc lớp 1 dựa trên đầu vào x .

w là vector trọng số (weight), là các tham số của mô hình cần được học từ dữ liệu huấn luyện.

x là vector đặc trưng (feature) của mẫu.

Hàm Sigmoid có dạng cong S và chuyển đổi giá trị của z từ khoảng -vô cùng đến vô cùng thành một khoảng giá trị từ 0 đến 1, biểu thị xác suất.

Để xây dựng một mô hình Logistic Regression, chúng ta cần sử dụng một tập dữ liệu huấn luyện đã được gán nhãn trước. Tập dữ liệu này bao gồm các biến đầu vào cùng với kết quả phân loại tương ứng. Quá trình huấn luyện mô hình tìm kiếm các tham số tối ưu để dự đoán kết quả phân loại cho các dữ liệu mới. Các tham số này thường được tối ưu thông qua các phương pháp như Gradient Descent.



Hình 2.1: Đồ thị hàm sigmoid

Trong quá trình huấn luyện, mục tiêu là điều chỉnh vector trọng số w để tối ưu hóa việc dự đoán xác suất theo thực tế và giá trị dự đoán. Để làm điều này, thường sử dụng các thuật toán tối ưu hóa như gradient descent để tìm giá trị w tối ưu.

Đối với bài toán phân loại đa lớp, Logistic Regression có thể được mở rộng bằng cách sử dụng một phương pháp như One-vs-Rest (OvR) để phân loại vào nhiều lớp khác nhau.

2.2.2 Bộ phân loại Ridge (Ridge Classifier))

Ridge Classifier là một thuật toán phân loại quan trọng trong lĩnh vực Machine Learning. Thuật toán này thường được sử dụng để dự đoán hoặc phân loại các đối tượng vào các lớp khác nhau dựa trên thông tin từ các biến đầu vào. Ridge Classifier không chỉ có khả năng phân loại đa lớp mà còn giúp giảm thiểu quá khớp và cải thiện hiệu suất của mô hình phân loại. [6]

Ridge Classifier hoạt động bằng cách tính toán giá trị dự đoán cho mỗi lớp dựa trên các biến đầu vào. Thay vì dự đoán một giá trị nhị phân như trong Logistic Regression, Ridge Classifier tính toán một vector chứa các giá trị xác suất rơi vào mỗi lớp cụ thể. Điều này giúp mô hình thể hiện sự không chắc chắn trong việc phân loại và tránh tình

trạng quá tự tin.

Phương trình toán học của Ridge Classifier cho bài toán phân loại đa lớp có dạng tương tự như trong Logistic Regression. Đối với mỗi lớp, Ridge Classifier tính toán một hàm chuyển đổi xác suất, thường là hàm softmax, để biểu diễn khả năng rơi vào từng lớp.

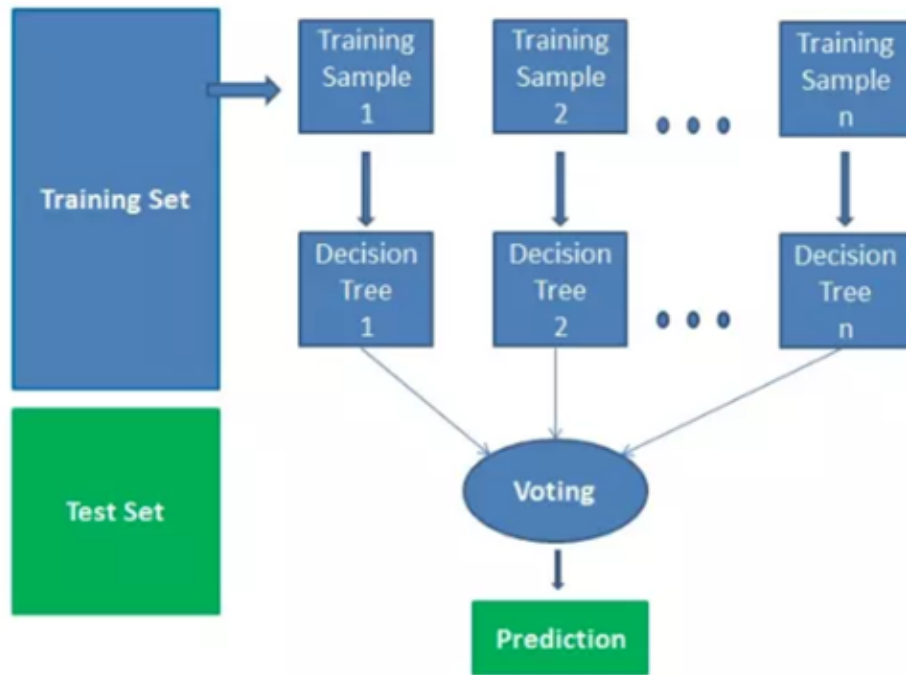
Quá trình huấn luyện mô hình Ridge Classifier liên quan đến tối ưu hóa các tham số để tối đa hóa khả năng dự đoán đúng trên tập dữ liệu huấn luyện. Điều này bao gồm việc điều chỉnh các trọng số của mô hình cho mỗi biến đầu vào và tham số chính quy (regularization parameters) để tránh quá khớp. Thuật toán thường sử dụng các phương pháp tối ưu hóa như Gradient Descent để tìm các giá trị tham số tối ưu.

2.2.3 Random Forest Classifier

Random Forest Classifier là một thuật toán phân loại trong lĩnh vực học máy. Nó là một biến thể của mô hình Random Forest, một phương pháp mạnh mẽ để giải quyết bài toán phân loại và hồi quy. Mô hình này dựa trên nguyên tắc của "rừng cây quyết định" (decision tree ensemble), trong đó nhiều cây quyết định được kết hợp để tạo thành một mô hình phân loại mạnh mẽ và ổn định. [7]

Thuật toán hoạt động theo 4 bước:

1. Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
2. Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
3. Bỏ phiếu cho mỗi kết quả dự đoán.
4. Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.



Hình 2.2: Nguyên lí hoạt động

2.2.4 Gradient Boosting Classifier

Boosting (Mục tiêu là giảm bias - áp dụng cho các model có variance thấp và bị bias cao): Xây dựng một lượng lớn các model (thường là cùng loại). Mỗi model sau sẽ học cách sửa những errors của model trước (dữ liệu mà model trước dự đoán sai) -> tạo thành một chuỗi các model mà model sau sẽ tốt hơn model trước bởi trọng số được update qua mỗi model (cụ thể ở đây là trọng số của những dữ liệu dự đoán đúng sẽ không đổi, còn trọng số của những dữ liệu dự đoán sai sẽ được tăng thêm) . Chúng ta sẽ lấy kết quả của model cuối cùng trong chuỗi model này làm kết quả trả về.

Gradient Boosting Classifier là một mô hình phân loại trong lĩnh vực học máy, thuộc vào họ thuật toán boosting. Mô hình Gradient Boosting Classifier là một phương pháp mạnh mẽ để giải quyết các bài toán phân loại, nơi mục tiêu là xây dựng một mô hình dự đoán tốt hơn bằng cách tập trung vào việc cải thiện các điểm yếu của mô hình trước đó. [7]

Gradient Boosting đều xây dựng thuật toán nhằm giải quyết bài toán tối ưu sau:

$$\min_{c_n=1:N, w_n=1:N} L(y, \sum_{n=1}^N c_n w_n)$$

Trong đó :

L : giá trị loss function

y : label

c_n : confidence score của weak learner thứ n (hay còn gọi là trọng số)

w_n : weak learner thứ n

Thay vì cố gắng quét tìm tất cả các giá trị c_n, w_n để tìm nghiệm tối ưu toàn cục - một công việc tốn nhiều thời gian và tài nguyên, chúng ta sẽ cố gắng tìm các giá trị nghiệm cục bộ sau khi thêm mỗi một mô hình mới vào chuỗi mô hình với mong muốn dần đi đến nghiệm toàn cục.

$$\min_{c_n, w_n} L(y, W_{n-1} + c_n w_n)$$

Trong đó: $W_{n-1} = \sum_{n=1}^{N-1} c_n w_n$

2.2.5 So sánh các mô hình máy học

Logistic Regression (Hồi quy Logistic):

- Ưu điểm: Đơn giản, dễ hiểu, thường hoạt động tốt khi các lớp dữ liệu là tách biệt tốt và không có sự phức tạp lớn.

- Hạn chế: Khả năng tách biệt lớp không tốt, không thể xử lý tốt với dữ liệu phi tuyến.

Ridge Classifier:

- Ưu điểm: Giúp kiểm soát overfitting, phù hợp với các bộ dữ liệu có nhiễu, là một biến thể của Logistic Regression nhưng có khả năng xử lý dữ liệu tốt hơn.

- Hạn chế: Vẫn có khả năng không phù hợp với các bài toán có độ phức tạp cao.

Random Forest Classifier:

- Ưu điểm: Mạnh mẽ, khả năng xử lý dữ liệu phi tuyến, có khả năng xử lý nhiễu, không yêu cầu chuẩn hóa dữ liệu.

- Hạn chế: Dễ bị overfitting trên dữ liệu nhỏ, khó để hiểu cách hoạt động nội tại của cây quyết định.

Gradient Boosting Classifier:

- Ưu điểm: Hiệu suất cao, khả năng xử lý dữ liệu tốt, tạo ra một mô hình mạnh bằng cách kết hợp nhiều cây quyết định yếu.

- Hạn chế: Dễ bị overfitting trên dữ liệu nhỏ, đòi hỏi nhiều thời gian và tài nguyên tính toán hơn.

2.3 Các thư viện cần thiết

2.3.1 Mediapipe

Mediapipe là một thư viện phần mềm được phát triển bởi Google, được thiết kế để hỗ trợ xây dựng ứng dụng liên quan đến xử lý hình ảnh và video, thường dựa trên dữ liệu hình ảnh hoặc video từ máy ảnh hoặc nguồn media khác.

Thư viện Mediapipe cung cấp các công cụ và khung làm việc để nhận diện và theo dõi các đối tượng, hình ảnh, cử chỉ và các khía cạnh khác trong thời gian thực. Nó giúp đơn giản hóa việc phát triển các ứng dụng về xử lý hình ảnh, như phát hiện khuôn mặt, theo dõi cử chỉ tay, xác định vị trí khung xương của cơ thể, và nhiều nhiệm vụ xử lý hình ảnh và video khác.

Mediapipe cung cấp các công cụ API, các mô hình mạng neural pre-trained và các thành phần xử lý hình ảnh chất lượng cao, giúp các nhà phát triển tập trung vào việc xây dựng ứng dụng thú vị và độc đáo mà không cần phải đầu tư quá nhiều vào việc triển khai và tinh chỉnh các phương pháp xử lý hình ảnh cơ bản.

2.3.2 OpenCV

OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở được phát triển để hỗ trợ xử lý hình ảnh và thị giác máy tính. Thư viện này được thiết kế để cung cấp các công cụ và chức năng cho việc xử lý hình ảnh, từ việc đọc và ghi ảnh, biến đổi hình ảnh, phát hiện đối tượng, theo dõi vị trí, và nhiều tác vụ xử lý hình ảnh khác.

Các chức năng chính của OpenCV bao gồm:

1. Xử lý hình ảnh cơ bản: Đọc, ghi và biến đổi hình ảnh, điều chỉnh độ tương phản, độ sáng, lọc hình ảnh và các phép biến đổi cơ bản khác.
2. Nhận diện đối tượng: OpenCV cung cấp các thuật toán phổ biến để phát hiện và nhận diện đối tượng trong hình ảnh, như Haar cascades, HOG (Histogram of Oriented Gradients), và Deep Learning-based object detection.
3. Theo dõi vị trí và chuyển động: Theo dõi vị trí của đối tượng trong chuỗi các khung hình, phát hiện chuyển động, và ước tính vận tốc.
4. Thị giác máy tính: Xử lý và phân tích hình ảnh để trích xuất thông tin hữu ích, như biểu đồ, đặc trưng, hoặc các đối tượng quan trọng.
5. Xử lý video: Xử lý và biến đổi video, áp dụng hiệu ứng hình ảnh và các phép biến đổi khác.

2.3.3 CSV

CSV (Comma-Separated Values) là một định dạng lưu trữ dữ liệu phổ biến được sử dụng để biểu diễn thông tin trong một bảng dữ liệu, trong đó các giá trị được phân tách bằng dấu phẩy (,) hoặc một ký tự phân tách khác. Định dạng CSV thường được sử dụng để trao đổi và lưu trữ dữ liệu tabular (theo dạng bảng) giữa các ứng dụng và hệ thống khác nhau.

Mỗi hàng trong tệp CSV thường tương ứng với một bản ghi hoặc một dòng trong bảng dữ liệu, và các cột tương ứng với các thuộc tính hoặc trường dữ liệu. Các giá trị trong các ô được phân cách bằng dấu phẩy hoặc ký tự phân tách khác, như dấu chấm phẩy (;) hoặc dấu tab (^).

2.3.4 Os

Trong Python, thư viện "os" là một thư viện chuẩn được cung cấp sẵn để tương tác với hệ điều hành và thực hiện các hoạt động liên quan đến hệ thống tệp và thư mục. Thư viện "os" cung cấp các chức năng để thực hiện các tác vụ như quản lý thư mục, tạo và xóa tệp, kiểm tra quyền truy cập, và nhiều hoạt động khác.

2.3.5 Numpy

NumPy (Numerical Python) là một thư viện mã nguồn mở trong Python, được sử dụng rộng rãi để thực hiện các phép toán số học và xử lý dữ liệu trong mảng đa chiều. Thư viện này giúp cho việc làm việc với dữ liệu số và phân tích số trở nên dễ dàng và hiệu quả hơn.

Các đặc điểm chính của NumPy:

Mảng Đa Chiều (Arrays): NumPy cung cấp cấu trúc dữ liệu mảng đa chiều, cho phép bạn làm việc với các tập dữ liệu nhiều chiều (ví dụ: ma trận) một cách hiệu quả. Mảng NumPy có thể có kích thước cố định và không thay đổi sau khi khởi tạo.

Phép Toán Số Học: NumPy cung cấp các hàm số học cơ bản (cộng, trừ, nhân, chia, lũy thừa) cũng như các hàm toán học cao cấp (sin, cos, exp, log, v.v.) để thực hiện các phép toán số học trên mảng.

Xử Lý Mảng: Thư viện này cung cấp các hàm cho việc thay đổi kích thước, cắt, chuyển vị và thao tác khác trên mảng.

Lập Trình Vectorized: NumPy khuyến khích việc sử dụng lập trình vectorized để thực hiện các phép toán trên mảng một cách nhanh chóng và hiệu quả, thay vì sử dụng vòng lặp.

Tích Hợp Các Dự Án Khác: NumPy tích hợp tốt với các thư viện và công cụ khác trong hệ sinh thái khoa học dữ liệu của Python như SciPy, pandas, matplotlib, và scikit-learn.

2.3.6 Pickle

Pickle là một module trong Python được sử dụng để serialize (chuyển đổi thành dạng bytes) và deserialize (chuyển đổi từ dạng bytes thành đối tượng ban đầu) các đối tượng Python. Quá trình pickle giúp bạn lưu trữ các đối tượng Python vào tệp hoặc truyền qua mạng và sau đó khôi phục chúng thành đối tượng Python ban đầu mà không mất đi thông tin hoặc cấu trúc.

Một số điểm quan trọng về Pickle:

Lưu Trữ Dữ Liệu: Pickle cho phép bạn lưu trữ và truyền các đối tượng Python phức tạp như danh sách, từ điển, lớp, và nhiều loại dữ liệu khác.

Không phải là Định Dạng Giao Tiếp Chung: Pickle thường không phải là định dạng giao tiếp chung vì nó có thể không an toàn đối với dữ liệu không được tin cậy từ nguồn bên ngoài. Điều này có thể dẫn đến các vấn đề về bảo mật hoặc tương thích trong tương lai.

Sử Dụng Pickle:

Để serialize một đối tượng thành dạng bytes: `pickle.dump(obj, file)`

Để deserialize một đối tượng từ dạng bytes: `obj = pickle.load(file)`

2.3.7 Pandas

Các Chức Năng Xử Lý Dữ Liệu: Pandas cung cấp nhiều chức năng để thực hiện các hoạt động xử lý dữ liệu như lọc, lấy dữ liệu con, sắp xếp, xử lý giá trị bị thiếu, và biến đổi dữ liệu.

Hỗ Trợ Đọc và Ghi Dữ Liệu: Pandas có khả năng đọc và ghi dữ liệu từ và vào nhiều định dạng dữ liệu phổ biến như CSV, Excel, SQL, JSON, và nhiều định dạng khác.

Thống Kê và Tính Toán: Pandas cung cấp các hàm tính toán thống kê, như tính trung bình, tổng, phương sai, và hỗ trợ cho các phép toán thống kê phức tạp.

Tích Hợp Dễ Dàng: Pandas có tích hợp tốt với các thư viện và công cụ khác trong hệ sinh thái khoa học dữ liệu của Python như NumPy, Matplotlib và SciPy.

2.3.8 Sklearn

Scikit-learn, thường được viết tắt là "sklearn", là một thư viện mã nguồn mở trong Python chuyên về machine learning (học máy) và data mining (khai phá dữ liệu). Scikit-learn cung cấp nhiều công cụ và thuật toán tiêu chuẩn để xây dựng và huấn luyện các mô hình học máy cho nhiều loại tác vụ khác nhau như phân loại, hồi quy, phân cụm, và nhiều tác vụ khác.

Một số đặc điểm chính của scikit-learn:

1. Thuật Toán Học Máy: Scikit-learn cung cấp một loạt các thuật toán học máy phổ biến như Decision Trees, Random Forests, Support Vector Machines (SVM), Naive Bayes, K-Means, và nhiều thuật toán khác.

2. Tiêu Chuẩn Hóa Giao Diện: Scikit-learn cung cấp một giao diện tiêu chuẩn cho việc xây dựng, huấn luyện và đánh giá mô hình học máy. Điều này giúp cho việc chuyển đổi giữa các thuật toán dễ dàng hơn.

3. Tiền Xử Lý Dữ Liệu: Scikit-learn cung cấp các công cụ để xử lý và tiền xử lý dữ liệu trước khi đưa vào huấn luyện mô hình. Điều này bao gồm chuẩn hoá dữ liệu, xử lý giá trị thiếu, và mã hóa dữ liệu.

4. Đánh Giá và Kiểm Định Mô Hình: Scikit-learn cung cấp các công cụ để đánh giá và kiểm định hiệu suất của mô hình học máy bằng cách sử dụng các phương pháp cross-validation và các chỉ số đánh giá.

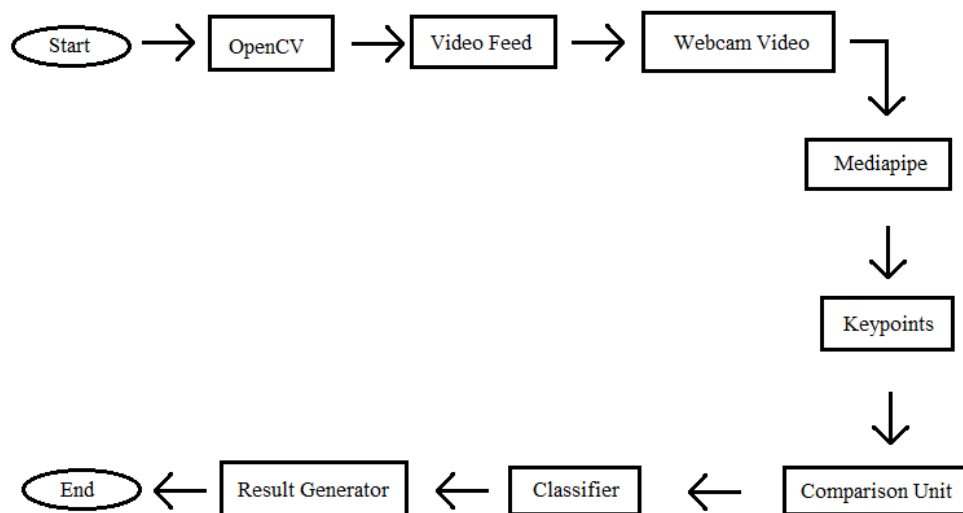
5. Hỗ Trợ Đa Loại Tác Vụ: Scikit-learn không chỉ hỗ trợ các tác vụ phân loại và hồi quy, mà còn hỗ trợ phân cụm, giảm chiều dữ liệu, học bán giám sát, và nhiều loại tác vụ khác.

Chương 3

THIẾT KẾ HỆ THỐNG

3.1 Cơ chế hoạt động

Người dùng có thể tìm hiểu về các tư thế yoga của hệ thống cung cấp. Dữ liệu được thu thập qua webcam của người dùng. Sử dụng mediapipe để phát hiện các tư thế yoga trong thời gian thực. Các điểm (point) được xác định và lưu trong mảng 3 chiều. Những điểm này dùng để so sánh tư thế người dùng với tư thế yoga tiêu chuẩn. Khi độ trùng khớp cao có thể kết luận tư thế của người dùng đang thực hiện.



Hình 3.1: Sơ đồ khối cách hoạt động của mô hình

- Thu thập dữ liệu: Các tư thế được sử dụng trong đề tài là Balasana (Child's Pose), Baddha Konasana (Bound Angle Pose), Utkata Konasana (Goddess Pose), Kumbhakasana (Plank Pose), Vrksasana (Tree Pose), Virabhadrasana II (Warrior II Pose). Tốc độ khung hình 30 FPS (30 khung hình /giây). Khoảng cách để webcam có thể thấy rõ toàn bộ cơ thể trong tầm 4m tính từ vị trí đặt webcam.
- Xử lý dữ liệu: Sử dụng thư viện mediapipe để trích xuất các điểm (point) của tư thế trong khung hình theo mảng 3 chiều, sau đó làm phẳng thành mảng 1 chiều và lưu trong file csv dưới dạng bảng.
Với tập dữ liệu từ file CSV ta sẽ huấn luyện mô hình máy học phân loại các động tác dựa vào tọa độ các điểm của tư thế và nhãn tên của tư thế đó.

3.2 Chi tiết hệ thống

3.2.1 Tải các thư viện cần thiết và chạy Mediapipe

```
import mediapipe as mp
import cv2
```

Hình 3.2: Thư viện Mediapipe và OpenCV

Đoạn mã này import thư viện Mediapipe và gán cho biến tên là mp. Mediapipe là một thư viện được phát triển bởi Google, cung cấp các công cụ và mô hình để xử lý dữ liệu hình ảnh và video, bao gồm việc nhận diện landmarks trên khuôn mặt, cơ thể, tay, v.v.

Thư viện OpenCV là một thư viện phổ biến cho xử lý dữ liệu hình ảnh và video, giúp ta có thể kết nối vào webcam và xử lý hình ảnh trên đó.

```
mp_drawing = mp.solutions.drawing_utils
mp_holistic = mp.solutions.holistic
```

Hình 3.3: Tạo biến để vẽ các điểm (points)

Tạo một biến mp-drawing để sử dụng các tiện ích liên quan đến việc vẽ landmarks trên hình ảnh.

Tạo một biến mp-holistic để sử dụng mô hình Holistic của Mediapipe. Mô hình Holistic có khả năng nhận diện khuôn mặt, cơ thể và tay cùng một lúc tuy nhiên đề tài chỉ sử dụng nhận diện cơ thể. Nhóm chọn xài holistic để cho nếu muốn phát triển đề tài xa hơn hoặc muốn sử dụng khuôn mặt và tay để đem lại kết quả chính xác hơn thì sẽ tiện chỉnh sửa cho sau này.

Ta sẽ test 1 đoạn nhỏ để đọc khung hình từ webcam:

```
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()
    cv2.imshow('Raw Webcam Feed', frame)

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Hình 3.4: Lấy khung hình từ webcam

Khởi tạo một đối tượng VideoCapture từ webcam bằng tham số 0 trong hàm VideoCapture. Tham số 0 này đại diện cho webcam đầu tiên được tìm thấy trên hệ thống.

Trong vòng lặp vô hạn, dòng “ret, frame = cap.read()” sẽ đọc một khung hình từ webcam. Biến “ret” sẽ chứa giá trị True nếu việc đọc thành công, False nếu không thành công và biến “frame” chứa khung hình đã đọc.

Sau đó ta hiển thị khung hình đang đọc (frame) trong cửa sổ tên “Raw Webcam Feed”.

Câu lệnh if sẽ kiểm tra nếu ta nhấn phím “q” thì sẽ kết thúc vòng lặp và đến bước cuối.

Hai dòng cuối được dùng để giải phóng tài nguyên của webcam để đảm bảo không để lại tài nguyên mở khi kết thúc chương trình và đóng tất cả các cửa sổ đang hiển thị, làm cho chương trình kết thúc sạch sẽ mà không còn cửa sổ đang chạy.

Sau khi ta đọc được khung hình từ webcam, bước tiếp theo cần làm chính là sử dụng Mediapipe để hiển thị và nối các landmark trên cơ thể:

```
cap = cv2.VideoCapture(0)
with mp_holistic.Holistic(min_detection_confidence=0.5, min_tracking_confidence=0.5) as holistic:
    while cap.isOpened():
        ret, frame = cap.read()

        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        results = holistic.process(image)

        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # Pose Detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS)

        cv2.imshow('Raw Webcam Feed', image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

Hình 3.5: Hiển thị và nối các landmarks trên cơ thể

Để hiển thị landmark trên cơ thể ta sử dụng mô hình Holistic của Mediapipe. Tham số min-detection-confidence và min-tracking-confidence xác định ngưỡng tối thiểu để xác định mức độ tin cậy của việc nhận diện và theo dõi.

Vì Mediapipe yêu cầu đầu vào là ảnh RGB nhưng màu của khung hình là dạng BGR nên sau khi lấy khung hình từ webcam về ta cần phải chuyển đổi màu của khung hình từ BGR sang RGB.

Dòng tiếp theo là dòng quan trọng nhất trong đề tài, “results = holistic.process(image)” vì nó sử dụng mô hình Holistic để xử lý ảnh và nhận diện các điểm đặc trưng (landmark) trong cơ thể và kết quả sẽ được lưu vào biến results.

Sau khi xử lý xong hình ảnh, ta chuyển đổi màu của khung hình từ RGB sang BGR lại để có thể hiển thị trên OpenCV.

Vẽ các landmarks của mô hình Holistic lên khung hình sử dụng hàm “draw-landmarks” của thư viện Mediapipe. Trong trường hợp này, chúng ta vẽ landmarks liên quan đến pose (tư thế cơ thể) và kết nối giữa chúng.

3.2.2 Thu thập và lưu trữ điểm đặc trưng

```
import csv
import os
import numpy as np
```

Hình 3.6: Thư viện CSV, Os, Numpy

Thư viện csv cung cấp các chức năng để làm việc với các tệp dữ liệu CSV. Dữ liệu CSV là một loại tệp văn bản dễ đọc được sử dụng để lưu trữ dữ liệu bảng dưới dạng hàng và cột.

Thư viện os cung cấp các chức năng để tương tác với hệ thống tệp và thư mục như tạo và xóa tệp, kiểm tra sự tồn tại của thư mục,...

Thư viện numpy cho tính toán số học và xử lý mảng đa chiều. Nó cung cấp nhiều chức năng hữu ích để thao tác và xử lý dữ liệu số.

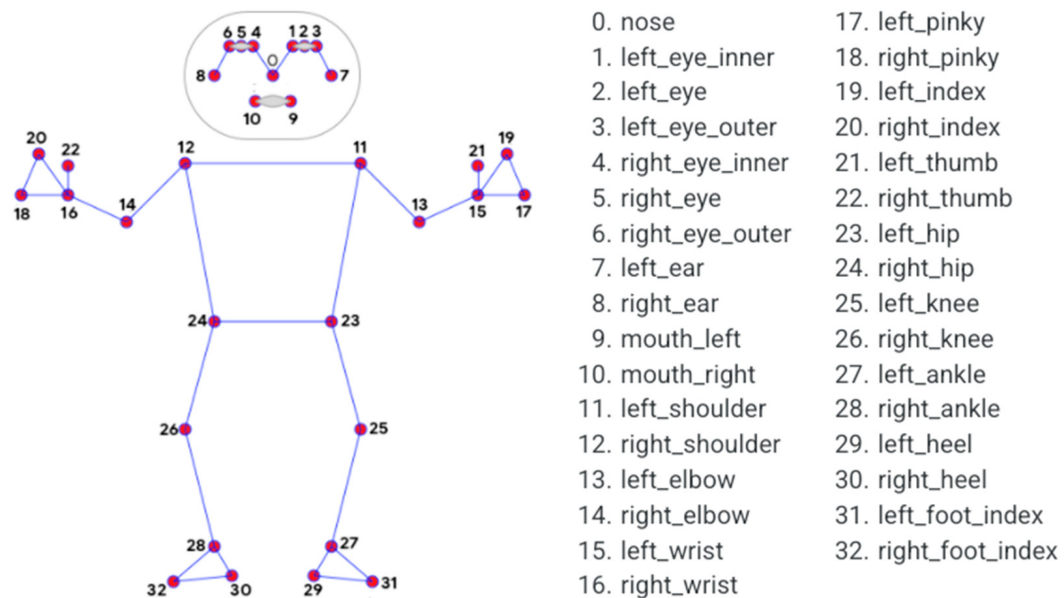
```
num_coords = len(results.pose_landmarks.landmark)
num_coords
```

33

Hình 3.7: Thư viện CSV, Os, Numpy

Ta sử dụng dòng code này để tính toán số lượng điểm đặc trưng được nhận diện trong kết quả của mô hình Holistic. Các điểm này được xác định bằng tọa độ trên khung hình.

Ở đây khi ta chạy “num-coords” thì nó sẽ trả về giá trị 33 điều đó có nghĩa là khung hình có 33 điểm đặc trưng. Để có thể biết rõ 33 điểm đó nằm ở đâu trên cơ thể thì ta có hình sau:



Hình 3.8: Mô hình pose landmarks

Tiếp theo ta cần chuẩn bị cho việc lưu dữ liệu từ mô hình nhận diện landmarks vào một tệp CSV để sau này có thể sử dụng trong quá trình huấn luyện mô hình machine learning.

```
landmarks = ['class']
for val in range(1, num_coords+1):
    landmarks += ['x{}'.format(val), 'y{}'.format(val), 'z{}'.format(val), 'v{}'.format(val)]
```

Hình 3.9: Lưu trữ mô hình vào file csv

Ban đầu, ta tạo ra một danh sách landmarks với một phần tử duy nhất “class”. Phần tử này sẽ là đại diện cho nhân tên ví dụ như: Goddess, Tree, ...

Tiếp theo đó ta vào vòng lặp để lặp ra từng số cho mỗi đặc trưng và đem vào danh sách.

Mỗi đặc trưng sẽ có 4 thông tin cụ thể sau:

- + x: Tọa độ ngang của điểm đặc trưng.
- + y: Tọa độ dọc của điểm đặc trưng.
- + z: Tọa độ sâu của điểm đặc trưng.
- + v: Mức độ tin cậy (visible) của điểm đặc trưng.

Như vậy, khi vòng lặp kết thúc thì ta sẽ có danh sách “landmarks” chứa các tên đặc trưng và thông tin liên quan đến tọa độ và mức tin cậy của từng điểm đặc trưng. Tiếp theo ta tạo tệp csv

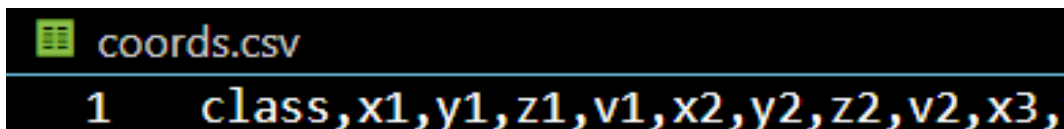
```
with open('coords.csv', mode='w', newline='') as f:  
    csv_writer = csv.writer(f, delimiter=',', quotechar='\"', quoting=csv.QUOTE_MINIMAL)  
    csv_writer.writerow(landmarks)
```

Hình 3.10: Tạo file csv

“with open('coords.csv', mode='w', newline='') as f:”: Dòng này mở tệp có tên 'coords.csv' để ghi dữ liệu. Tham số mode='w' cho biết chế độ ghi. Tham số newline="" giúp tránh vấn đề về xử lý ký tự xuống dòng khác nhau trên các hệ thống khác nhau.

“csv_writer = csv.writer(f, delimiter=',', quotechar='\"', quoting=csv.QUOTE_MINIMAL)”: Đoạn mã này tạo một đối tượng csv-writer để viết dữ liệu vào tệp. Tham số delimiter=',' chỉ định rằng dấu phẩy sẽ được sử dụng làm ký tự phân tách giữa các trường. Tham số quotechar='\"' chỉ định rằng dấu ngoặc kép sẽ được sử dụng để bao quanh các trường chứa dấu phẩy. Tham số quoting=csv.QUOTE_MINIMAL chỉ định rằng chỉ các trường chứa ký tự đặc biệt sẽ được bao quanh bởi dấu ngoặc kép.

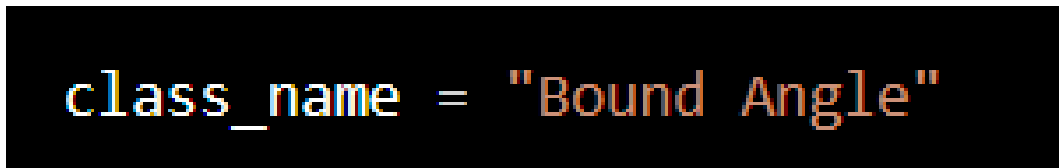
“csv_writer.writerow(landmarks)”: Dòng này ghi dòng tiêu đề (header) vào tệp CSV. Dòng này tạo ra một dòng trong tệp CSV chứa tên nhãn dán và các thông tin liên quan đến các đặc trưng đó, tách nhau bởi dấu phẩy. Ta sẽ được kết quả như này:



```
coords.csv  
1 class,x1,y1,z1,v1,x2,y2,z2,v2,x3,
```

Hình 3.11: Dữ liệu được ghi trong file csv

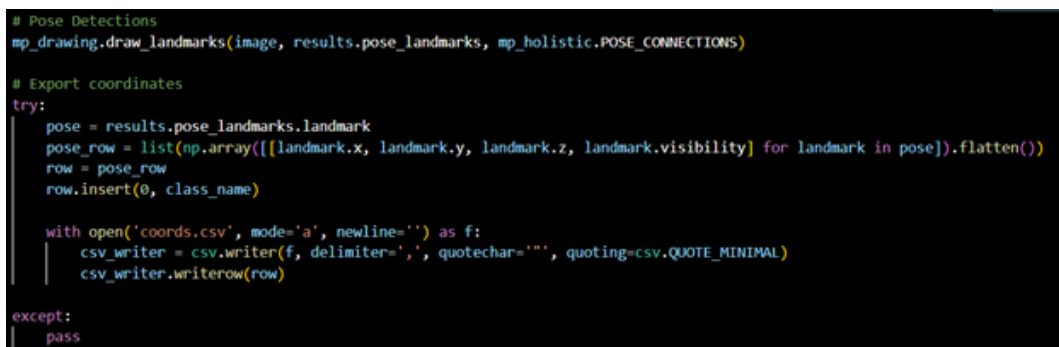
Sau khi tạo tệp CSV để chứa các thông tin, dữ liệu cần thiết thì đã đến lúc ta thu thập các điểm đặc trưng.



`class_name = "Bound Angle"`

Hình 3.12: Đặt nhãn cho động tác Bound Angle

Trước tiên ta cần nêu tên nhãn dán ra trước, đề tài em gồm 6 động tác cơ bản: Bound Angle, Tree, Goddess, Warrior II, Plank, Child's Pose. Ban đầu ta đặt class-name là Bound Angle.



```
# Pose Detections
mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS)

# Export coordinates
try:
    pose = results.pose_landmarks.landmark
    pose_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.visibility] for landmark in pose]).flatten())
    row = pose_row
    row.insert(0, class_name)

    with open('coords.csv', mode='a', newline='') as f:
        csv_writer = csv.writer(f, delimiter=',', quotechar='\"', quoting=csv.QUOTE_MINIMAL)
        csv_writer.writerow(row)
except:
    pass
```

Hình 3.13: Nhận diện động tác

Sau khi thu về khung hình và nối các điểm đặc trưng như bước trên, ta làm thêm động tác xuất dữ liệu tọa độ từ các landmark của cơ thể và ghi vào file CSV.

Khởi đầu, ta gán danh sách các landmarks của tư thế yoga được nhận diện từ kết quả mô hình Holistic cho biến “pose”.

Tiếp theo ta tạo một danh sách “pose-row” chứa thông tin về các tọa độ và mức độ tin cậy (visibility) của mỗi landmark trong tư thế. Dòng code này sử dụng một biểu thức danh sách để tạo ra danh sách 2D vì thế nên sau đó ta dùng `.flatten()` để biến nó thành một danh sách 1D.

Gán biến pose-row vào row, bước này không cần thiết vì mình chỉ xài mỗi landmark của cơ thể, nếu như trong trường hợp xài cả tay hoặc mặt thì biến row này sẽ là giá trị tổng của cơ thể, mặt, tay sau khi biến đổi sang danh sách 1D.

Thêm tên lớp (class-name) ở vị trí đầu tiên của danh sách row. Điều này giúp xác định tên của tư thế yoga tương ứng với dữ liệu đặc trưng. Ở ví dụ hiện tại thì tên đặc trưng là Bound Angle.

3.3 Huấn luyện mô hình phân loại

3.3.1 Xử lý dữ liệu thông qua pandas

```
import pandas as pd
from sklearn.model_selection import train_test_split
```

Hình 3.14: Thư viện Pandas và Sklearn

Ta sử dụng thư viện pandas vì nó là một thư viện mạnh mẽ cho xử lý và phân tích dữ liệu, chủ yếu dựa trên cấu trúc dữ liệu DataFrame.

Ngoài ra ta còn import hàm train-test-split từ model-selection của thư viện scikit-learn. Hàm này sẽ được sử dụng để chia dữ liệu thành tập huấn luyện và tập kiểm tra khi huấn luyện mô hình máy học.

```
df = pd.read_csv('coords.csv')
```

Hình 3.15: Đọc dữ liệu từ file csv

Sử dụng hàm read-csv từ thư viện pandas để đọc dữ liệu từ tệp csv và trả về một DataFrame chứa dữ liệu từ tệp csv này.

Sau khi trả về thì ta gán cho biến df để tiện cho việc sử dụng ở những code sau.

```
x = df.drop('class', axis=1)
y = df['class']
```

Hình 3.16: Tách dữ liệu đặc trưng và nhãn tên

Bước tiếp theo ta cần làm là tách tách dữ liệu đặc trưng và nhãn tên. Để làm được điều đó thì ta sẽ tạo 2 biến x và y. Biến x sẽ chứa các dữ liệu đặc trưng còn biến y sẽ chứa nhãn tên.

Sử dụng “df.drop”, ta bỏ lớp “class” ở DataFrame, “axis=1” là bỏ 1 cột và gán nó vào biến x.

Sau đó ta trích xuất cột “class” ở DataFrame và gán nó vào biến y.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=1234)
```

Hình 3.17: Chia dữ liệu thành 2 tập

Sử dụng hàm train-test-split từ thư viện scikit-learn. Ta sẽ chia dữ liệu thành 2 tập, tập huấn luyện và tập kiểm tra.

Biến x và biến y đều được chia thành 2 tập: x-train là tập huấn luyện, x-test là tập kiểm tra và tương tự cho y.

Tham số của “test-size” là 0,3 có nghĩa là trong dữ liệu, sẽ có 30% được chia vào tập kiểm tra, 70% còn lại được chia vào tập huấn luyện.

Tham số của “random-state” là 1234 có nghĩa là tham số này xác định hạt giống (seed) để đảm bảo rằng quá trình chia dữ liệu sẽ được thực hiện cùng một cách mỗi khi chạy lại mã. Điều này giúp đảm bảo kết quả của việc chia dữ liệu là nhất quán và dễ tái sản xuất.

3.3.2 Huấn luyện mô hình

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
```

Hình 3.18: Import 4 mô hình máy học

Ta import một số module từ thư viện scikit-learn để xây dựng và huấn luyện các mô hình máy học. Thông thường chỉ sử dụng 1 class nhưng để mang lại kết quả tốt nhất, nhóm chọn 4 class để chạy và xem kết quả class nào tốt nhất thì sẽ sử dụng class đó.

Import hàm `make_pipeline` từ module `pipeline` của thư viện `scikit-learn`. `make_pipeline` được sử dụng để tạo một chuỗi các bước xử lý dữ liệu và mô hình trong một pipeline (luồng công việc).

Import class `StandardScaler` từ module `preprocessing` của thư viện `scikit-learn`. `StandardScaler` được sử dụng để chuẩn hóa dữ liệu đặc trưng để có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1.

Import class `LogisticRegression` và `RidgeClassifier` từ module `linear-model` của thư viện `scikit-learn`. Đây là các mô hình học sử dụng trong phân loại.

Import class `RandomForestClassifier` và `GradientBoostingClassifier` từ module `ensemble` của thư viện `scikit-learn`. Đây là các mô hình học dựa trên phương pháp tập hợp (ensemble).

```
pipelines = {  
    'lr':make_pipeline(StandardScaler(), LogisticRegression()),  
    'rc':make_pipeline(StandardScaler(), RidgeClassifier()),  
    'rf':make_pipeline(StandardScaler(), RandomForestClassifier()),  
    'gb':make_pipeline(StandardScaler(), GradientBoostingClassifier())  
}
```

Hình 3.19: Tạo mảng pipeline

Tạo một mảng có tên `pipelines`, trong đó mỗi phần tử của từ điển đại diện cho một pipeline xử lý dữ liệu và mô hình học máy.

`'lr':make_pipeline(StandardScaler(), LogisticRegression())`: Đoạn code này tạo một phần tử trong mảng với khóa `'lr'` (tương ứng với `Logistic Regression`). Pipeline này bao gồm hai bước: chuẩn hóa dữ liệu sử dụng `StandardScaler()` và mô hình học máy `LogisticRegression()`.

Tương tự như trên, ta tạo ra phần tử `'rc'`, `'rf'`, `'gb'`, tương ứng với `RidgeClassifier()`, `RandomForestClassifier()`, `GradientBoostingClassifier()`.

Tạo một mảng `fit-models` chứa các mô hình đã được huấn luyện bằng cách lặp qua các pipelines và dùng chúng để huấn luyện trên tập dữ liệu huấn luyện.

`"for algo, pipeline in pipelines.items()"`: Bắt đầu một vòng lặp qua từng phần tử trong mảng `pipelines`. Biến `algo` sẽ lưu giá trị của khóa (key) trong mảng, tức là tên của mô hình (ví dụ: `'lr'`, `'rc'`, `'rf'`, `'gb'`), và biến `pipeline` sẽ lưu giá trị của giá trị (value), tức là

```
fit_models = {}
for algo, pipeline in pipelines.items():
    model = pipeline.fit(x_train, y_train)
    fit_models[algo] = model
```

Hình 3.20: Tạo mảng fitmodels

pipeline tương ứng với mô hình.

“model = pipeline.fit(x-train, y-train)”: Sử dụng pipeline để huấn luyện mô hình trên tập dữ liệu huấn luyện x-train và y-train. Biến model sẽ lưu trữ mô hình đã được huấn luyện.

“fit-models[algo] = model”: Gán mô hình đã huấn luyện vào từ điển fit-models với khóa tương ứng là algo, tức là tên của mô hình.

Sau khi chạy xong, ta có thể chạy thử dòng code để xem sau quá trình huấn luyện sẽ như nào:

```
fit_models['rf'].predict(x_test)
✓ 0.1s
array(['Plank', 'Warrior 1l', 'Warrior 1l', 'Tree', "Child's Pose",
       'Bound Angle', "Child's Pose", "Child's Pose", 'Goddess',
       'Warrior 1l', "Child's Pose", "Child's Pose", 'Warrior 1l',
       'Plank', 'Plank', 'Plank', 'Plank', 'Goddess', 'Plank', 'Goddess',
       'Bound Angle', 'Bound Angle', 'Goddess', 'Plank', 'Plank',
       "Child's Pose", 'Plank', 'Goddess', "Child's Pose", "Child's Pose",
       'Tree', 'Plank', 'Bound Angle', 'Bound Angle', 'Plank',
       "Child's Pose", 'Bound Angle', 'Plank', "Child's Pose", 'Tree',
       "Child's Pose", 'Tree', 'Goddess', 'Warrior 1l', 'Tree',
       "Child's Pose", 'Plank', 'Warrior 1l', 'Tree', 'Bound Angle',
       'Bound Angle', 'Tree', 'Plank', 'Tree', 'Tree', 'Bound Angle',
       'Tree', 'Warrior 1l', "Child's Pose", 'Plank', 'Plank',
```

Hình 3.21: Chạy fitmodels với mô hình rf

3.4 Đánh giá và lưu trữ mô hình

```
from sklearn.metrics import accuracy_score
import pickle
```

Hình 3.22: Tính độ chính xác của mô hình

Accuracy-score được sử dụng để tính độ chính xác của mô hình phân loại bằng cách so sánh dự đoán của mô hình với nhãn thực tế.

Module pickle để sử dụng trong việc lưu và nạp (serialize và deserialize) các đối tượng Python, bao gồm cả mô hình máy học.

```
for algo, model in fit_models.items():
    yhat = model.predict(x_test)
    print(algo, accuracy_score(y_test, yhat))
```

✓ 0.1s

```
lr 1.0
rc 1.0
rf 1.0
gb 0.9979838709677419
```

Hình 3.23: Chạy fitmodels với 4 mô hình

Tương tự như code trước, “for algo, model in fit-model.items()” bắt đầu một vòng lặp qua các phần tử trong mảng fit-models. Biến algo lưu tên của mô hình và biến model lưu mô hình tương ứng.

Biến yhat được định nghĩa là y mũ, dùng để lưu kết quả dự đoán nhãn trên tập dữ liệu kiểm tra.

Dòng cuối được dùng để in ra tên của mô hình và độ chính xác của mô hình bằng cách sử dụng hàm accuracy-score so sánh nhãn thực tế y-test và nhãn dự đoán yhat.

Ở kết quả, ta thấy lr, rc, rf đều đạt chính xác 100%. Điều này rất ít khi xảy ra, có thể do các thông tin của các điểm đặc trưng quá khác nhau nên dễ phân loại hoặc nguyên nhân gì đó khác. Vì kết quả 3 mô hình máy bằng nhau nên có thể chọn bất kỳ cái nào, nhóm em chọn mô hình rf để sử dụng.

```
with open('body_language.pkl', 'wb') as f:  
    pickle.dump(fit_models['rf'], f)
```

Hình 3.24: Module Pickle

Sử dụng module pickle, ta lưu mô hình RandomForestClassifier đã được huấn luyện vào tệp. Mở một tệp có tên “body-language.pkl” để ghi dữ liệu. Tham số wb cho biết chế độ ghi dưới dạng nhị phân.

Ta sử dụng hàm pickle.dump để lưu mô hình RandomForestClassifier từ mảng fit-models (với rf) vào tệp. Đối số đầu tiên là mô hình cần lưu, và đối số thứ hai là tệp mà ta muốn ghi dữ liệu vào. Trường hợp ở đây là ghi mô hình rf đã được huấn luyện vào tệp “body-language.pkl”.

3.5 Phân loại mô hình

```
with open('body_language.pkl', 'rb') as f:  
    model = pickle.load(f)
```

Hình 3.25: Mở tệp body-language.pkl

Trước khi phân loại, ta cần phải nạp mô hình đã lưu vào trước. Mở một tệp có tên “body-language.pkl” để đọc dữ liệu. Tham số rb cho biết chế độ đọc dưới dạng nhị phân.

Sử dụng hàm pickle.load để nạp mô hình từ tệp. Kết quả nạp sẽ được gán cho biến model, ta có thể sử dụng mô hình đã huấn luyện mà không cần phải huấn luyện lại từ đầu.

Code vẫn tương tự như lúc trích xuất điểm đặc trưng sang tệp csv nhưng sẽ khác ở phần dưới.

```

# Pose Detections
mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS)

# Export coordinates
try:
    pose = results.pose_landmarks.landmark
    pose_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.visibility] for landmark in pose]).flatten())
    row = pose_row

    x = pd.DataFrame([row])
    body_language_class = model.predict(x)[0]
    body_language_prob = model.predict_proba(x)[0]
    print(body_language_class, body_language_prob)

```

Hình 3.26: Dùng mô hình đã được huấn luyện để dự đoán dáng tập yoga

Sau khi lấy được các thông tin của các điểm đặc trưng trên cơ thể, thay vì lưu vào tệp csv thì ta sẽ tạo một DataFrame x để chứa dữ liệu điểm đặc trưng.

Ta sử dụng mô hình đã huấn luyện để dự đoán tình trạng cơ thể dựa trên dữ liệu đặc trưng của tình trạng pose trong DataFrame x. Kết quả dự đoán được gán cho biến body-language-class.

Ta có thể tính xác suất dự đoán cho mỗi tình trạng cơ thể sử dụng hàm predict-proba của mô hình. Kết quả xác suất được gán cho biến body-language-prob.

Và cuối cùng ta sẽ in ra nhãn tên tình trạng cơ thể đã dự đoán (body-language-class) và xác suất dự đoán cho mỗi lớp (body-language-prob).

Chương 4

KẾT QUẢ

Sau khi hoàn thành mô hình, nhóm em đưa ra được một hệ thống phát hiện và nhận dạng động tác yoga theo thời gian thực. Sử dụng thư viện mediapipe để xác định các điểm (points) trong mô hình pose landmarks. Tọa độ các điểm được lưu trong file csv.

Bằng cách áp dụng và so sánh 4 mô hình máy học, hệ thống đạt được độ chính xác tương đối cao 100% trên tập dữ liệu thử nghiệm. Khi phân loại chính xác các tư thế, hệ thống phản hồi kết quả của động tác người dùng đang thực hiện.



Hình 4.1: Kết quả trả về Tree Pose



Hình 4.2: Kết quả trả về Goddess Pose

Chương 5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Chương trình thực hiện được tính năng chính là phân loại được động tác yoga người dùng đang thực hiện và chấm điểm.

Quá trình bắt đầu bằng việc thu thập dữ liệu và tiền xử lý hình ảnh để trích xuất các đặc trưng quan trọng. Nhóm em đã sử dụng mô hình Holistic từ thư viện Mediapipe để nhận diện các điểm đặc trưng quan trọng trên cơ thể, bao gồm các điểm trên tay, chân, và cơ thể.

Tiếp theo, chúng em đã xây dựng một chuỗi pipeline bao gồm việc chuẩn hóa dữ liệu và đã huấn luyện các mô hình máy học trên tập dữ liệu huấn luyện và đánh giá hiệu suất của chúng trên tập dữ liệu kiểm tra bằng cách sử dụng độ chính xác so sánh giữa y-test và yhat.

Lưu trữ mô hình bằng cách sử dụng module pickle, giúp ta có thể sử dụng lại mô hình đã huấn luyện mà không cần phải huấn luyện lại từ đầu.

5.2 Hướng phát triển

Đề tài này còn nhiều chỗ thiếu sót và có thể được bổ sung, phát triển xa hơn như là:

- + Hiện thị ảnh mẫu của từng động tác và chấm điểm người dùng dựa theo ảnh mẫu đó. Thay đổi động tác mới thông qua nút nhấn.

- + Thiết kế bộ đếm giờ. Nếu thực hiện đúng động tác trên 70% thì bộ đếm giờ sẽ bắt đầu đếm. Mục đích là đo khoảng thời gian mà người dùng có thể giữ nguyên động tác đó.

- + Kết hợp landmarks của mặt và tay để có thể phân loại nhiều động tác hơn và chính xác hơn.

- + Thiết kế 1 trang web để người dùng có thể dễ dàng sử dụng.

[1] [2] [3] [4] [5] [6] [7]

Tài liệu tham khảo

- [1] R. Gajbhiye, S. Jarag, P. Gaikwad, and S. Koparde, “Ai human pose estimation: Yoga pose detection and correction,” *International Journal of Innovative Science and Research Technology*, 2022.
- [2] D. Swain, S. Satapathy, B. Acharya, M. Shukla, V. C. Gerogiannis, A. Kanavos, and D. Giakovis, “Deep learning models for yoga pose monitoring,” *Algorithms*, vol. 15, no. 11, p. 403, 2022.
- [3] S. Muley, P. Mahajan, P. Medidar, H. Chavan, and P. Patil, “Yoga guidance using human pose estimation,” *IRJMETs*, vol. 2, pp. 1533–7, 2020.
- [4] S. B. Kotsiantis, I. Zaharakis, P. Pintelas *et al.*, “Supervised machine learning: A review of classification techniques,” *Emerging artificial intelligence applications in computer engineering*, vol. 160, no. 1, pp. 3–24, 2007.
- [5] F. Y. Hsieh, D. A. Bloch, and M. D. Larsen, “A simple method of sample size calculation for linear and logistic regression,” *Statistics in medicine*, vol. 17, no. 14, pp. 1623–1634, 1998.
- [6] K. Rakesh and P. N. Suganthan, “An ensemble of kernel ridge regression for multi-class classification,” *Procedia computer science*, vol. 108, pp. 375–383, 2017.
- [7] J. L. Speiser, M. E. Miller, J. Tooze, and E. Ip, “A comparison of random forest variable selection methods for classification prediction modeling,” *Expert systems with applications*, vol. 134, pp. 93–101, 2019.