World Academy of Science, Engineering and Technology
International Journal of Electronics and Communication Engineering
Vol:17, No:3, 2023

# Fitness Action Recognition Based on MediaPipe

Zixuan Xu, Yichun Lou, Yang Song, Zihuai Lin



*Abstract*—MediaPipe is an open-source machine learning computer vision framework that can be ported into a multi-platform environment, which makes it easier to use it to recognize human activity. Based on this framework, many human recognition systems have been created, but the fundamental issue is the recognition of human behavior and posture. In this paper, two methods are proposed to recognize human gestures based on MediaPipe, the first one uses the Adaptive Boosting algorithm to recognize a series of fitness gestures, and the second one uses the Fast Dynamic Time Warping algorithm to recognize 413 continuous fitness actions. These two methods are also applicable to any human posture movement recognition.

*Keywords*—Computer Vision, MediaPipe, Adaptive Boosting, Fast Dynamic Time Warping.

## I. Introduction

THE dynamic recognition of human action has always been a popular topic in the field of machine vision, and it has a variety of uses in many fields, such as human-computer interaction, somatosensory games, medical detection, and motion detection. The current mainstream method is to use a depth camera or wearable devices to read the skeletal and joint information of the human body, so as to identify the dynamic trajectory of the human body. However, these current methods also face some potential problems. First of all, it will be very cumbersome to use IOT equipment to read information from important spots on the human body. Each test requires wearing corresponding equipment, must be remeasured for individuals of various sizes, and is unsuitable for extensive multi-person detection. While a depth camera can solve this issue, its relatively high cost prevents it from being extensively utilized by the general population. It is also somewhat susceptible to environmental light, and specialized equipment is needed to reliably read the critical details of the human body.

With the rapid advancement of artificial intelligence in recent years, a different approach to addressing these possible issues is to employ certain developed AI model frameworks to directly read the three-dimensional skeleton coordinates of the human body using a regular monocular camera in place of conventional depth cameras and wearable devices. At present, the mainstream mature algorithm frameworks for reading key coordinates of the human body include OpenPose, AlphaPose, MediaPipe, etc.

The main contribution of this paper is to propose two algorithms for recognizing human motion poses based on the MediaPipe algorithm framework. The first one uses the Adaptive Boosting algorithm to recognize a series of fixed poses, and the second one uses the Fast Dynamic Time Warping algorithm to recognize 413 consecutive fitness movements. Based on these two methods, recognition can be made for any human posture movements.

Zixuan Xu and Yichun Lou are with the School of Electrical and Information Engineering, The University of Sydney, Australia (e-mail: zixu2040@uni.sydney.edu.au, ylou0722@uni.sydney.edu.au).

Song Yang is with the Hangzhou Irealcare Health Techno Ltd, China (e-mail: syang@irealcare.com).

Zihuai Lin is with the School of Electrical and Information Engineering, The University of Sydney, Australia (corresponding author, e-mail:zihuai.lin@sydney.edu.au).

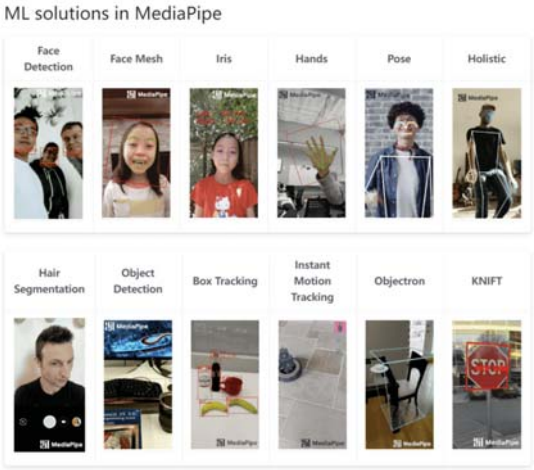



Fig. 1 MediaPipe machine learning solutions [2]

## II. Releated Work

In this section, we will focus on the relevant existing algorithmic frameworks used in this paper, including MediaPipe, Adaptive Boosting, and FastDTW. Also, since this experiment is conducted in a Python environment, some of the algorithmic frameworks used have been integrated into some libraries in Python, so the parameters of these libraries will be described.

### A. MediaPipe

The Google-created MediaPipe framework enables the creation of cross-platform applications by combining pre-existing perceptual components [1]. Numerous Machine Learning solutions, including face detection, face mesh, iris, hands, pose, etc, have so far been made available across different platforms such as Android, iOS, Python, JavaScript, etc, as shown in Fig. 1. This study primarily uses the Pose solution to track and detect the joint points of the human body in order to retrieve the three-dimensional coordinate points of the real world. The following will describe this solution in detail in the Python environment.

MediaPipe Pose Solution: The MediaPipe Pose solution, as seen in Fig. 2, can infer 33 3D coordinate points of the human body from a real-time video stream or image. It outputs x, y, z, and visibility; x and y are the ratio of the mark

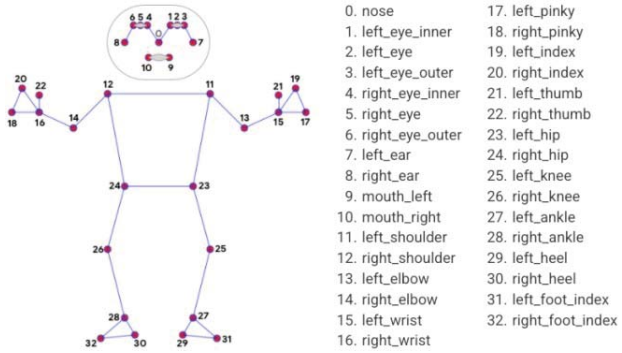



Fig. 2 MediaPipe Pose landmarks [3]

coordinates following the normalization of the image's length and width, while z stands for the depth coordinate. A visibility value indicates the likelihood that the sign will be visible in the photograph between 0 and 1. The midpoint of points 23 and 24 represents the location of the origin, which is at the buttocks of the human body. It should be mentioned that the official API claims that the scale of the z-axis is nearly equal to the x-axis, with a certain inaccuracy, given the value of the longitudinal coordinate of the z-axis (Some corresponding improvement methods will be proposed in the later part of the Methodology).

### B. Adaptive Boosting

Adaptive Boosting (AdaBoost) is an ensemble learning method that has been extensively applied in object detection in recent years, and it also plays a significant role in face detection and human detection. AdaBoost was first introduced by Freund and Schapire to perform a binary classification [4]. Generally, it implements an iterative method to learn from the errors of weak classifiers and turn them into strong ones. It is proved that AdaBoost is rather effective in increasing the classification margins of training data [5].

For the AdaBoost algorithm, the input is the training dataset, and the output is the final classifier, namely the strong leaner. The first step is to initialize the weight of each training data. All the data are assigned an equal sample weight. Then, since AdaBoost uses an iterative approach, there will be several rounds of iteration. In each round, using the weighted training data to get a weak classifier and then calculating the error rate. Next, updating the weights based on the mistakes means more weight will be assigned to the incorrectly classified samples so that they can be classified correctly in the next round. After finishing all the iterations, a strong classifier is constructed by combining the selected weak classifiers linearly [6]. AdaBoost is originally designed for binary classification. Regarding multi-classification problems, statisticians proposed variants of AdaBoost including AdaBoost.M1 [7], AdaBoost.MH, AdaBoost. MR [8], etc.

### C. Fast Dynamic Time Warping

Fast Dynamic Time Warping (FastDTW) is an improved algorithm that implements the Dynamic Time Warping algorithm in the Python environment. FastDTW provides a

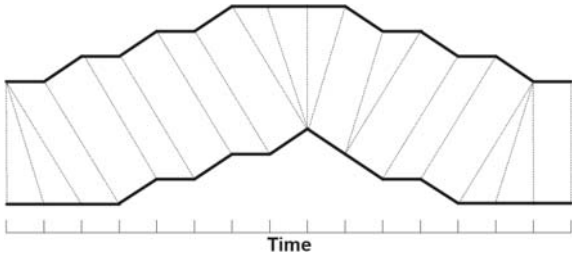



Fig. 3 A warping between two time series [9]

DTW approximation with linear time and space complexity. As shown in Fig. 3, the upper and lower lines represent two sequences in the time domain. Although they are shifted in the time domain, they have a certain similarity on the y-axis. If the corresponding points in time are appropriately translated to calculate the similarity of two-time series by this algorithm, the resulting DTW distance is 0 [9].

The dynamic time warping (DTW) algorithm does a great job of solving time series problems. Speech matching is the first application of this method. When people talk at varying rates, for instance, some people will draw out the sound of the letter "H" in the word "Hello." The time series is consistent overall, despite the fact that different pronunciations may just be time-shifted. By lengthening and shortening the time series, DTW determines how similar the two-time series are. The recognition of dynamic actions falls under this as well. To identify the action, the most similar action can be computed by comparing it to the eigenvalues of standard actions. This not only addresses the issue of different video data lengths but also simply calls for a model with a smaller set of standard data.

## III. Methodology

In this section, two methods for recognizing various types of human postures will be systematically described. The first method uses the AdaBoost algorithm to recognize fixed postures, while the second method uses the Fast Dynamic Time Warping algorithm to recognize a number of sequential movements.

### A. Fixed Pose Recognition

The objective of fixed pose recognition is to determine when the user reaches the standard posture. Since AdaBoost is applied for classification, the first step is preparing the dataset. The key data are collected from a dozen short videos through a self-designed method which will be introduced in the part Data Collection. Then, after data pre-processing, a classification model will be trained with AdaBoost by the training set, along with the performance evaluation which calculates the accuracy of the prediction on the test set. Part *B*. Model Construction and Evaluation will discuss model training and evaluation in detail. Finally, the real-time prediction can be realized based on the trained model.

*1) Data Collection:* The videos for data acquisition have been put in the same folder. For convenience, the video names are the pose names. To avoid misclassification, each video

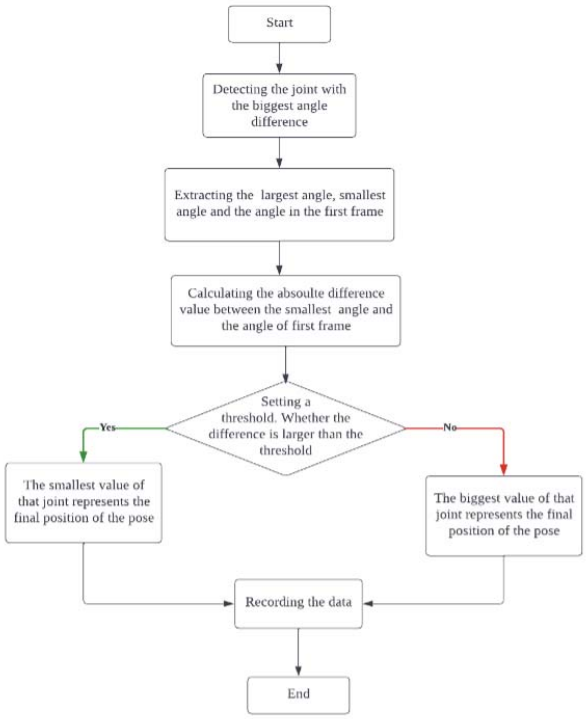



Fig. 4 Data collection flowchart

is equipped with a dataset separately rather than combining all the data into one file. With regard to a single video, the key data can be obtained by the methodology shown in the flowchart in Fig. 4.

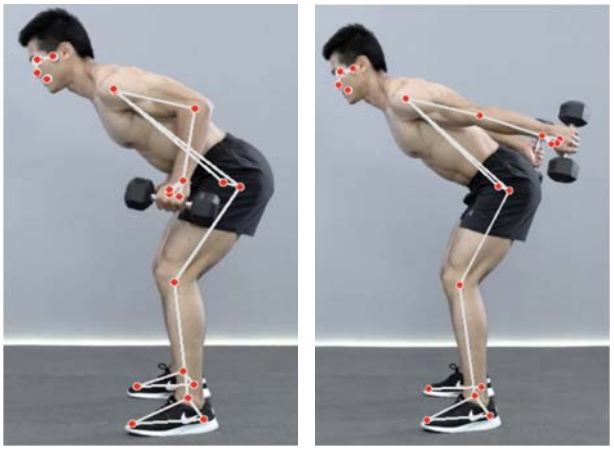
*Step 1:Key joint detection:* There are 10 selected joints in total and the first step is to determine which joint has the largest angle difference. This can be easily done by obtaining the maximum and minimum value of each joint while doing the exercise and then subtracting the two extrema. The joint with the biggest result is the key joint of the posture.

*Step 2: Key value extraction* Extracting three values of the key joint respectively, which is the maximum, the minimum and the value in the first frame.

*Step 3: Calculation* Calculating the absolute difference value of the minimum angle and the value in the first frame.

*Step 4: Key moment detection* Setting a threshold. The threshold in this methodology is 55 degrees. This value is an empirical value obtained from a series of testing on different videos. If the absolute difference is greater than the threshold, it means when the key joint reaches the minimum angle, the exercise is completed. If not, the moment when the key joint reaches the maximum angle represents the completion of the pose.

The logic behind such comparison is to judge whether the minimum or maximum value of the key joint is close to the initial value of the pose. In other words, the calculation in Step 3 represents whether the key joint is initially close to the minimum. If yes, the maximum value of the key joint indicates the end position of the posture and vice versa.



(a) Initial position (b) End position

Fig. 5 Triceps curl

Taking the exercise triceps curl as an example. Fig. 5 shows the initial and the final pose of the workout respectively. In the process of doing this exercise, the joint with the greatest angular variation is obviously the elbow. The maximum angle (E_max) of the elbow is calculated by MediaPipe which is 175 degrees while the minimum (E_min) is 91 degrees. The angle of the elbow in the first frame (E_first) is also 93 degrees. Then, the absolute difference is $|91 - 93| = 2$ which is smaller than the threshold (55 degrees). Therefore, at the initial stage of the exercise, the angle of the elbow is close to the minimum. As a result, the maximum angle is the result should be kept, which means the end position of the movement.

*Step5: Key data recording* Based on the key movement detected in Step 4, we write the angles of 10 selected joints into the dataset with the correct label (pose name). To enlarge this positive instance, all the data of key movement are repeated 100 times. Apart from key movement, other data are regarded as negative instances which are labeled as "Not Detected" (as shown in Fig. 6).

*2) Model Construction and Evaluation:* Before training the model, the data should be pre-processed. Firstly, datasets can be split into training sets and test sets. The training set accounts for 70% of the whole data. Secondly, the training set should go through with standardization which is a scaling technique wherein it makes the data scale-free by removing the mean and scaling to unit variance. With the processed data and label, a prediction model can be trained by the AdaBoost algorithm.

After getting the model, the next step is evaluation. The accuracy relates to the predicted label and the real label. The formula for accuracy is shown below, where TP means True Position (Predicting positive as a positive class), FN means False Negative (Predicting positive class to negative class), FP means False Positive (Predicting negative class as a positive class), and TN means True Negative (Predicting negative class to negative class). If the accuracy is not ideal, parameters must be adjusted manually to improve performance.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Pose | LeftElbow | RightElbow | LeftArmpit | RightArmpit | LeftShoulde | RightShoulc | LeftKnee | RightKnee | LeftHip | RightHip |
| 73 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 74 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 75 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 76 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 77 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 78 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 79 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 80 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 81 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 82 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 83 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 84 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 85 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 86 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 87 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 88 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 89 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 90 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 91 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 92 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 93 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 94 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 95 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 96 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 97 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 98 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | !4 | 53 | 151 | 51 |
| 99 | Deep Squati | 161 | 121 | 130 | 96 | 143 | 156 | !4 | 53 | 151 | 51 |
| 100 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | 5! | 53 | 151 | 51 |
| 101 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | 5! | 53 | 151 | 51 |
| 102 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | 5! | 53 | 151 | 51 |
| 103 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | 5! | 53 | 151 | 51 |
| 104 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | 5! | 53 | 151 | 51 |
| 105 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | 5! | 53 | 151 | 51 |
| 106 | Deep Squati | 161 | 121 | 130 | 95 | 143 | 155 | 5! | 53 | 151 | 51 |

(a) Positive instances

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Pose | LeftElbow | RightElbow | LeftArmpit | RightArmpit | LeftShoulde | RightShoulc | LeftKnee | RightKnee | LeftHip | RightHip |
| 2 | Not Detecte | 172 | 124 | 39 | 13 | 93 | 117 | 163 | 164 | 94 | 107 |
| 3 | Not Detecte | 172 | 123 | 38 | 14 | 93 | 116 | 163 | 165 | 97 | 108 |
| 4 | Not Detecte | 172 | 123 | 38 | 13 | 94 | 116 | 163 | 166 | 97 | 109 |
| 5 | Not Detecte | 171 | 124 | 39 | 13 | 95 | 115 | 163 | 166 | 97 | 109 |
| 6 | Not Detecte | 170 | 125 | 39 | 13 | 95 | 114 | 164 | 166 | 96 | 109 |
| 7 | Not Detecte | 173 | 123 | 40 | 13 | 96 | 114 | 163 | 166 | 96 | 111 |
| 8 | Not Detecte | 174 | 123 | 44 | 13 | 99 | 114 | 164 | 167 | 95 | 115 |
| 9 | Not Detecte | 171 | 123 | 45 | 12 | 100 | 115 | 165 | 168 | 95 | 116 |
| 10 | Not Detecte | 174 | 124 | 48 | 12 | 102 | 115 | 166 | 169 | 96 | 119 |
| 11 | Not Detecte | 175 | 124 | 52 | 10 | 103 | 116 | 167 | 170 | 96 | 121 |
| 12 | Not Detecte | 174 | 125 | 52 | 10 | 103 | 116 | 167 | 171 | 96 | 120 |
| 13 | Not Detecte | 174 | 127 | 57 | 10 | 106 | 116 | 167 | 171 | 95 | 122 |
| 14 | Not Detecte | 172 | 127 | 59 | 10 | 106 | 116 | 167 | 171 | 94 | 123 |
| 15 | Not Detecte | 169 | 126 | 56 | 12 | 103 | 116 | 164 | 170 | 91 | 121 |
| 16 | Not Detecte | 170 | 122 | 54 | 14 | 101 | 118 | 165 | 171 | 89 | 118 |
| 17 | Not Detecte | 170 | 120 | 52 | 17 | 100 | 119 | 166 | 169 | 89 | 116 |
| 18 | Not Detecte | 168 | 116 | 52 | 20 | 100 | 120 | 163 | 169 | 88 | 114 |
| 19 | Not Detecte | 166 | 113 | 53 | 22 | 101 | 120 | 163 | 167 | 88 | 109 |
| 20 | Not Detecte | 165 | 107 | 53 | 27 | 104 | 120 | 161 | 162 | 87 | 105 |
| 21 | Not Detecte | 164 | 105 | 56 | 30 | 107 | 121 | 160 | 158 | 87 | 102 |
| 22 | Not Detecte | 164 | 101 | 59 | 32 | 109 | 123 | 157 | 157 | 88 | 103 |
| 23 | Not Detecte | 164 | 99 | 63 | 34 | 111 | 123 | 155 | 145 | 89 | 94 |
| 24 | Not Detecte | 165 | 95 | 68 | 39 | 116 | 126 | 149 | 129 | 96 | 79 |
| 25 | Not Detecte | 164 | 92 | 73 | 45 | 120 | 131 | 142 | 125 | 101 | 77 |
| 26 | Not Detecte | 164 | 92 | 74 | 48 | 121 | 131 | 137 | 120 | 108 | 69 |
| 27 | Not Detecte | 166 | 90 | 78 | 51 | 123 | 135 | 133 | 121 | 111 | 72 |
| 28 | Not Detecte | 167 | 88 | 85 | 53 | 128 | 135 | 129 | 127 | 116 | 80 |
| 29 | Not Detecte | 166 | 88 | 90 | 56 | 133 | 134 | 126 | 122 | 120 | 77 |
| 30 | Not Detecte | 167 | 88 | 94 | 59 | 136 | 135 | 120 | 116 | 126 | 72 |

(b) Negative instances

Fig. 6 Data example

*3) Real-time Prediction:* The last step is applying the model to real-time prediction. The user can stand in front of the camera and do different poses. The result of whether the pose is detected or not will be shown on the screen.

### *B. Continuous Action Recognition*

The first method describes in detail how to identify a fixed pose using the AdaBoost algorithm, but for most actions, it is a series of consecutive actions rather than a single pose, so this part will primarily describe the use FastDTW algorithm to identify a series of consecutive actions.

*1) Action recognition mechanism:* The FastDTW similarity distance between the input sequence and the model sequence is typically determined directly for conventional discriminant algorithms. The degree of similarity between the two increases with decreasing distance. We select the model with the shortest distance, that is, the most matching model name. Alternately, we set a certain threshold for the calculated similarity distance. If the distance is within the threshold, it is judged that the action recognition is successful. However, the coordinates of the z-axis (depth coordinate) will have some drift errors in the process of recognizing three-dimensional coordinates by MediaPipe. So, if we directly select the most matching model, there may be some similar actions that may have errors in recognition. Similarly, if the threshold is set, due to the different lengths of each action video and the error of the MediaPipe depth coordinate, it may be impossible to accurately find a suitable threshold to distinguish the input and standard action. This paper, therefore, suggests a different approach. Fig. 7 shows the entire flow chart of this section. The left part represents the data of each dynamic action. Each action has three videos from different perspectives which are the front, left front, and right front viewpoints. This indicates that one action contains a typical video with three points of perspective, which can somewhat mitigate the effect of the MediaPipe depth coordinates inaccuracy. Then we process the video of each action, extract all the feature values of this action and store them in the file, read the user's input data in real-time, compare it with the data just stored, and calculate the FastDTW distance. In this way, we get the names of the five most similar actions. The action with the shortest FastDTW distance is used as the standard hypothesis is action B. If at least three actions in the first five actions are action B, Then we judge that the result of this action recognition is action B, and if it is not satisfied, the recognition fails.

*2) Extraction of features:* The MediaPipe Pose Solution model provided 33 coordinate points for one person, but because the distance between the subject and the camera varies, there may be a significant variation in the coordinates of the same action. Therefore, these coordinates must be improved and transformed into fixed position angles. The angle's value will be roughly similar when the same action specification is carried out, making it easier to apply the FastDTW algorithm to determine the similarity afterward. However, if all angles are directly calculated based on these 33 detected points and three points make an angle, then we use:

$$C_{33}^{3} * 3 = 16368 \tag{2}$$

We can get 16368 angles which is an excessive number of angles and a lot of them are worthless, just basic composition. As a result, the eigenvalues of each frame of the continuous action are directly retrieved from the reasonably significant eleven angles. Table I and Fig. 8 depict a summary of all eleven fixed joint angles. These angles can basically cover most of the dynamic movements of the whole body. According to these angles, by analyzing each frame of a continuous action video, the complete feature value of an action can be obtained.

In the description of MediaPipe above, we know that it returns the coordinates (x,y,z) of the 33 key points of the whole body, suppose there are three points A (x1, y1, z1), B (x2, y2, z2), C (x3, y3, z3), we need to calculate ∠ACB. Based on the coordinates, the vector can be get:

$$\overrightarrow{a} = (x3 - x1, y3 - y1, z3 - z1) \tag{3}$$

$$\overrightarrow{b} = (x3 - x2, y3 - y2, z3 - z2) \tag{4}$$

Then ∠ACB formed by these two vectors can be calculated by using the following vector equation:

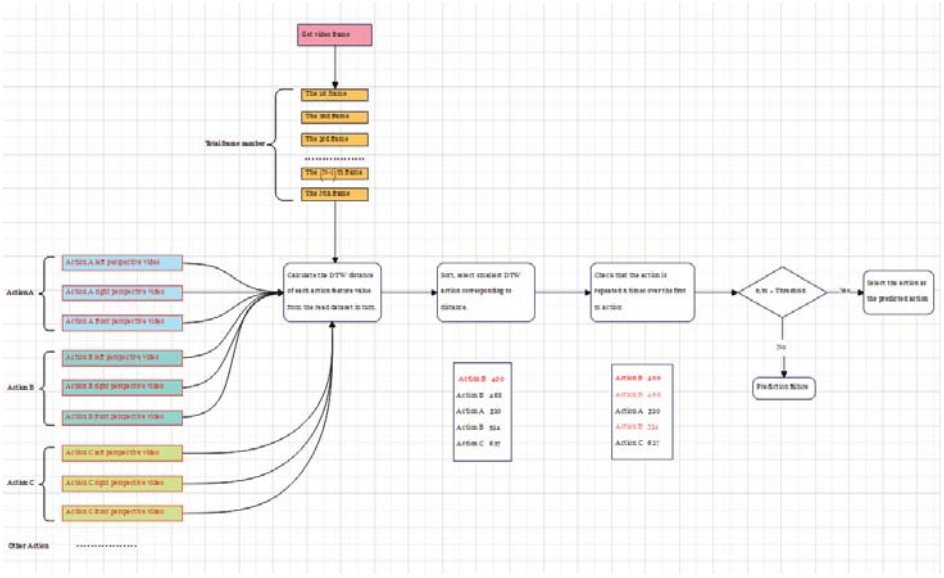


Fig. 7 Action recognition mechanism chart flow

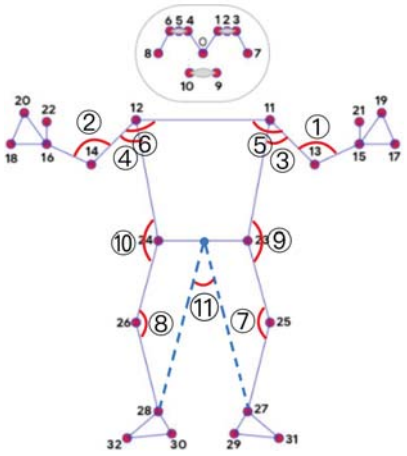


Fig. 8 MediaPipe landmarks and fixed joint angles

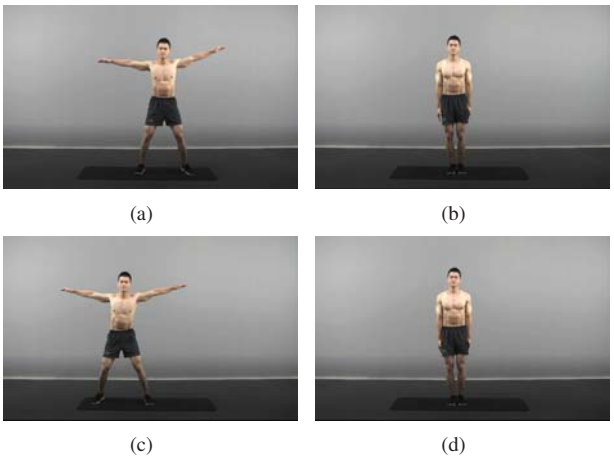


Fig. 9 Star step pictures

$$\angle ACB = \arccos \frac{\overrightarrow{a} \cdot \overrightarrow{b}}{|\overrightarrow{a}||\overrightarrow{b}|} \tag{5}$$

TABLE I
FIXED JOINT ANGLE NAME

| Index | Angle Name | Related MediapPipe Coordinate Number |
|---|---|---|
| 1 | left_elbow_angle | 11,13,15 |
| 2 | right_elbow_angle | 12,14,16 |
| 3 | left_shoulder_angle | 13,11,23 |
| 4 | right_shoulder_angle | 14,13,24 |
| 5 | left_chest_angle | 12,11,23 |
| 6 | right_chest_angle | 11,12,24 |
| 7 | left_knee_angle | 23,25,27 |
| 8 | right_knee_angle | 24,26,28 |
| 9 | left_waist_angle | 11,23,25 |
| 10 | right_waist_angle | 12,24,26 |
| 11 | crotch_angle | 27,23,24,28 |

*3) Storage and reading of eigenvalue file:* Taking the action star step as an example (see Fig. 9), it is clear from part *Extraction of features* that for a single video action, first we analyze each frame of the video to produce eleven angles of each frame of video, then sort by time series to get the complete feature value of the video, the first 30 frame features as shown in Fig. 10. It should be noted that each read's z-axis coordinates will vary significantly due to the MediaPipe's depth coordinates drifting. Even if the same video is read twice, the feature values are not the same. We repeat this operation for three videos of all fitness movements and use the NumPy library to store all the eigenvalues as a .npz file as the standard data model of the movement.

However, this also faces a problem. When the amount of data gradually increases, each retrieval time will also increase correspondingly. According to the test, when the feature values of all 413 videos are stored in the same file, each reading time is about 30 seconds, which is obviously unacceptable. For the design of fitness recognition, due to its particularity, some improved methods can be proposed. First

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | time frams | left elbow angle | right elbow angle | left shoulder angle | right shoulder angle | left chest angle | right chest angle | left knee angle | right knee angle | left waist angle | right waist angle | crotch angle | |
| 2 | 1 | 102. 9824282 | 111. 4017032 | 10. 07013236 | 11. 8484656 | 94. 59582575 | 92. 45894198 | 144. 0232463 | 139. 1411417 | 176. 5602049 | 172. 909158 | 5. 998613324 | |
| 3 | 2 | 103. 8375456 | 108. 972156 | 10. 34142372 | 11. 09258472 | 94. 14176812 | 93. 61897125 | 144. 9061429 | 136. 3205503 | 174. 2234178 | 171. 8378386 | 7. 009486598 | |
| 4 | 3 | 110. 4519642 | 115. 2449885 | 10. 47591402 | 12. 23191235 | 94. 42960052 | 94. 7352097 | 145. 0508814 | 139. 0311522 | 174. 6395166 | 176. 0908423 | 7. 407822602 | |
| 5 | 4 | 110. 7110361 | 115. 8747676 | 10. 82039906 | 11. 87693613 | 95. 05475663 | 94. 07018316 | 147. 5038511 | 141. 4926141 | 175. 18277 | 174. 4297654 | 7. 998100481 | |
| 6 | 5 | 111. 5149713 | 118. 5259362 | 10. 35984034 | 13. 17524223 | 96. 82938868 | 93. 18362622 | 150. 1400092 | 140. 8893434 | 174. 5018383 | 173. 8287467 | 7. 324708333 | |
| 7 | 6 | 111. 3992449 | 117. 9501986 | 9. 769271141 | 12. 66392412 | 95. 38745892 | 93. 50189493 | 147. 0072553 | 142. 1360715 | 175. 0897373 | 175. 1300787 | 7. 324108372 | |
| 8 | 7 | 112. 1676331 | 116. 0329966 | 9. 486648907 | 12. 11745645 | 94. 88427211 | 93. 79410562 | 145. 3959542 | 141. 0300817 | 174. 8475711 | 176. 1466895 | 7. 035495442 | |
| 9 | 8 | 112. 6764215 | 113. 8455953 | 8. 32535961 | 11. 30463854 | 91. 98897982 | 95. 40956318 | 147. 7736108 | 140. 2456317 | 173. 7496388 | 175. 0369787 | 7. 171332395 | |
| 10 | 9 | 112. 108935 | 114. 6299236 | 8. 620130004 | 11. 05806763 | 92. 22237516 | 95. 07699424 | 147. 1861535 | 142. 219378 | 173. 6106889 | 176. 7603747 | 7. 352717612 | |
| 11 | 10 | 112. 1893367 | 114. 030006 | 8. 682089923 | 11. 092185 | 92. 6457692 | 94. 75867562 | 146. 9377379 | 140. 6415545 | 172. 9928429 | 175. 7199236 | 7. 756027403 | |
| 12 | 11 | 111. 1787456 | 113. 6942547 | 8. 713041762 | 11. 21976927 | 93. 81005524 | 93. 94422141 | 148. 0052533 | 137. 412197 | 172. 8024563 | 170. 7151765 | 7. 977456636 | |
| 13 | 12 | 111. 5113863 | 113. 8305405 | 8. 766981854 | 11. 23556013 | 94. 12266281 | 93. 69235837 | 148. 3920586 | 136. 7760137 | 172. 1751249 | 169. 3038264 | 8. 183658616 | |
| 14 | 13 | 111. 9004366 | 114. 6331493 | 8. 942539889 | 11. 59700975 | 94. 34054415 | 93. 66518222 | 145. 2895088 | 137. 4373776 | 170. 0472714 | 169. 106642 | 9. 396270937 | |
| 15 | 14 | 115. 4324343 | 114. 0366443 | 9. 327822272 | 11. 21162945 | 94. 6243887 | 94. 31953569 | 146. 7279572 | 136. 9993422 | 169. 3230638 | 171. 0743474 | 12. 52794373 | |
| 16 | 15 | 115. 351978 | 112. 6620065 | 9. 68717171 | 11. 12945543 | 94. 68067007 | 94. 81828729 | 144. 1537012 | 138. 3349872 | 165. 8479301 | 171. 5673491 | 16. 11031028 | |
| 17 | 16 | 113. 1874723 | 113. 4511216 | 9. 861170624 | 10. 77120417 | 96. 07009625 | 93. 68116803 | 139. 1278504 | 137. 3068773 | 164. 0051536 | 167. 1625842 | 13. 83169608 | |
| 18 | 17 | 110. 4150809 | 113. 1949579 | 10. 14963811 | 11. 10792456 | 99. 13775701 | 91. 52259752 | 147. 2676515 | 135. 6405811 | 166. 8208262 | 167. 1790842 | 16. 2059529 | |
| 19 | 18 | 108. 2907945 | 113. 9686169 | 10. 860139 | 11. 41866744 | 99. 2302065 | 91. 97354781 | 147. 1023204 | 137. 2076931 | 166. 3856793 | 169. 8630469 | 16. 89874774 | |
| 20 | 19 | 104. 6588854 | 109. 9330536 | 13. 04146928 | 13. 61827458 | 99. 48543226 | 93. 94979506 | 145. 6952316 | 127. 513124 | 164. 0502322 | 158. 3321778 | 17. 84743582 | |
| 21 | 20 | 101. 8176858 | 105. 3470455 | 16. 19009412 | 16. 15986254 | 101. 7825015 | 94. 86873111 | 144. 0002987 | 131. 2506661 | 161. 8813904 | 158. 7458573 | 19. 4476741 | |
| 22 | 21 | 100. 3538 | 105. 4042369 | 17. 18041311 | 17. 29599303 | 103. 1109791 | 95. 45114106 | 147. 046347 | 133. 6458553 | 166. 1848173 | 160. 5943607 | 18. 25698147 | |
| 23 | 22 | 99. 87851139 | 105. 1186091 | 19. 35115637 | 18. 38058639 | 104. 5517196 | 97. 01977653 | 144. 4664193 | 125. 8565216 | 163. 3754297 | 154. 1957332 | 19. 23486134 | |
| 24 | 23 | 103. 5795471 | 104. 1220674 | 21. 02149156 | 20. 64349552 | 106. 4626926 | 100. 5190793 | 130. 745024 | 120. 3961147 | 151. 1619961 | 148. 4217362 | 22. 68648738 | |
| 25 | 24 | 101. 6429448 | 104. 3748887 | 22. 32298119 | 20. 96001592 | 107. 7555204 | 101. 0993142 | 133. 1246195 | 121. 8626872 | 152. 850751 | 149. 004821 | 23. 44972766 | |
| 26 | 25 | 109. 4358446 | 109. 853833 | 25. 00318612 | 23. 40366935 | 107. 8205178 | 106. 8273212 | 133. 9933776 | 120. 7271591 | 157. 335878 | 148. 1016655 | 25. 80834901 | |
| 27 | 26 | 109. 9057273 | 106. 6764938 | 28. 68915258 | 27. 25455565 | 109. 9751998 | 110. 9412164 | 133. 7849693 | 117. 824084 | 161. 0864475 | 150. 7767786 | 26. 63543835 | |
| 28 | 27 | 107. 5478122 | 108. 9769723 | 29. 71537099 | 27. 21092527 | 111. 3985584 | 111. 0882256 | 134. 4975019 | 120. 5241693 | 160. 5361017 | 152. 0093456 | 26. 12701369 | |
| 29 | 28 | 109. 4654655 | 103. 8844664 | 34. 17722043 | 39. 98426688 | 107. 3784822 | 118. 472114 | 134. 3434579 | 133. 3570003 | 163. 6433871 | 159. 8366341 | 26. 65114741 | |
| 30 | 29 | 105. 4821138 | 102. 7592387 | 41. 53825573 | 46. 15738568 | 111. 5215641 | 118. 5695517 | 126. 6278422 | 128. 4874777 | 162. 9509317 | 157. 9827815 | 27. 78488633 | |

Angle +

Fig. 10 The first 30 frames feature of the start step

of all, the movements of fitness exercises are usually divided by the training parts, and the movements of each training are completed in groups, not randomly selected from 413 movements. We can reduce the scope of each search based on this condition. All the movements depend on the training parts, and can be divided into 20 groups, each group has about 20 movements, which can ensure that the detection time in this group is about 1 second. Secondly, we can first give the user the name of the action that needs to be done, then retrieve the group of the action according to this name, thereby reducing the complexity of retrieval and reducing the recognition time.

## IV. Experiments

In this section, the steps to implement the experiments using the algorithmic approach will be described in detail. First, the equipment information and some data sources for the experiments will be explained, and then the results and analysis of the experiments made will be described in detail.

### A. The Specification of Hardware and Datasets

In *B. Results for Fixed Postures*, the camera mainly used in the experiment is the 1080P Facetime HD monocular camera. In *C. Results for Continuous Actions*, the camera mainly used is Aoni ordinary monocular camera with 1920*1080 resolution. The computer and Python environments are configured as follows:

Operating System: macOS 12.1(part B) / Windows11 home version(part C)

CPU: Apple M1 Pro(part B) / AMD Ryzen 7 6800H with Radeon Graphics 3.19 GHz(part C)

Python Version: Python 3.8.8

Main Python Libraries: MediaPipe, AdaBoost, FastDTW, OpenCV, NumPy

All the data used in the experiments come from the website [10], which includes a total of 413 fitness movements, each of which contains a video completed by a professional fitness trainer. The lengths of the videos vary, ranging from 32 to 6096 frames.

```
------ Start training models using Adaboost for different poses ------
Train for Abs1 is Completed. The accuracy is 1.0
Train for Abs2 is Completed. The accuracy is 1.0
Train for Abs3 is Completed. The accuracy is 1.0
Train for Abs4 is Completed. The accuracy is 1.0
Train for Bent Leg Raise1 is Completed. The accuracy is 1.0
Train for Bent Leg Raise2 is Completed. The accuracy is 1.0
Train for Deep Squat is Completed. The accuracy is 1.0
Train for Dimond Push ups is Completed. The accuracy is 1.0
Train for Double Dumbells Squat is Completed. The accuracy is 1.0
Train for Dumbell Squat is Completed. The accuracy is 1.0
Train for Elastic Band Front Raise is Completed. The accuracy is 1.0
Train for Elastic Band Lateral Raise is Completed. The accuracy is 1.0
Train for Elastic Band Up Raise2 is Completed. The accuracy is 1.0
Train for Half Squat is Completed. The accuracy is 1.0
Train for Push ups1 is Completed. The accuracy is 1.0
Train for Side Abs1 is Completed. The accuracy is 1.0
Train for Side Abs2 is Completed. The accuracy is 1.0
Train for Single Dumbell Row is Completed. The accuracy is 1.0
Train for Single Dumbell Squat is Completed. The accuracy is 0.98
Train for Single Leg Squat is Completed. The accuracy is 1.0
Train for Single Shoulder Press is Completed. The accuracy is 1.0
Train for Single Tricepts is Completed. The accuracy is 1.0
Train for Tricepts Curl is Completed. The accuracy is 1.0
-------------------- Completed --------------------
```

Fig. 11 Model accuracy

In *C. Results for Continuous Actions*, about the recognition of the continuous actions, in the previous method description, due to the unstable drift of the MediaPipe depth coordinates, it is proposed to input standard videos from three perspectives. However, due to the limited angle of the video, this study reversed the original video left and right as the second view. One copy of each video, so one action contains four videos from left and right perspectives.

### B. Results for Fixed Postures

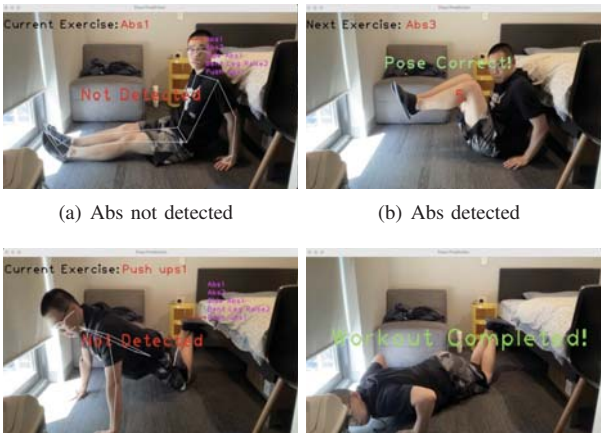
The accuracy of the models trained by AdaBoost is perfect. Most of the rates reach 100% while only one accuracy is 98%. Due to the high accuracy, there is no need to tune the parameters. All the parameters are set as default. The accuracy is shown in Fig. 11.

Before making the prediction in real time, the expected exercises should be determined for the user. Multiple exercises can be input at one time. The interface is shown in Fig. 12 (a). If the wrong name of the exercise is detected, the system will ask the user to input it again. The result of this is shown in Fig. 12 (b).

Then, the user must do each exercise one by one. There is a short pause of five seconds between two exercises and there


(a) Input the right name (b) Wrong name detected

Fig. 12 Input action name process




(a) Abs not detected (b) Abs detected

(c) Push ups not detected (d) Push ups detected

Fig. 13 Action recognition live test

(a) Left (b) Middle (c) Right

Fig. 14 User location detection

(a) Wait for start signal (b) Buffer time

Fig. 15 Start the game

is an exercise list to indicate to the user the current process of the whole workout. When the last exercise is completed, "Workout Completed" will be shown on the screen.

### C. Results for Continuous Actions

In this part, feature value files are first extracted and stored for all video groups by the method described before. It randomly selects one of the actions in the list, and the specified user completes it. Then it reads the user's action information from the camera, compares it with the feature value file, and calculates the FastDTW distance. And it uses the recognition algorithm to get the most similar action name, and checks whether it matches the specified action name, if it matches, it is considered successful recognition, if it does not match, it is considered that the action execution is not standard, and the recognition fails. Finally, all 413 action videos were recognized successfully.

Some specific steps and results will be described in detail below: First, we determine the specific location of the user and make sure that the user is in the center of the screen, so as to reduce errors as much as possible and reduce additional interference. This only needs to determine that all the coordinate points detected by MediaPipe are within the drawn box. As shown in Fig. 14, when it is not within the box, the user will be prompted to "Move into the box".

When it is detected that the user's whole body is within the frame, an action name in the action list is randomly selected (take Star step as an example). It locates the eigenvalue file where the action is located according to the action name so that it can be compared after the action is completed. Once the user gives the signal that they are ready to act, which is when their hands are pushed together for three seconds, waiting for the detection to begin. Following the three seconds of buffer time, the fitness action will then begin to be detected (see Fig. 15).

When starting to recognize fitness movements, the standard video of professional fitness trainers will also be displayed in the lower left corner for user reference. At the same time, the timing starts according to the standard video length of the action, and the user needs to complete the specified action within the timing range, as shown in Fig. 16.

When the time is over, we read the complete data of the user in the action and compare it with the pre-stored standard feature value file. The most similar action name is obtained through the FastDTW algorithm and the discrimination mechanism. If it is consistent with the name randomly selected before, the name of the recognized action will be displayed. If it is inconsistent, it will be displayed that the action is not standardized, and the recognition fails and starts the next set

(a) Detect position (b) Wait for the start signal

(c) Start game (d) Wait for the start signal

Fig. 16 Action recognition live test

(a) Recognition success (b) Recognition fail

Fig. 17 Recognition results

of workouts, as shown in Fig. 17.

### D. Analysis

In the first method proposed above, we use the AdaBoost algorithm to identify a series of fixed fitness postures, but the disadvantages of such an approach are obvious. Any fitness activity is a continuous process, and the recognition of only one pose is incomplete, and the trajectory and force process of many actions cannot be judged by a single pose. For the second method, the FastDTW algorithm is used to solve the problem of continuous movement recognition, and we can use this method to recognize some continuous movements. The advantage of this method compared with machine learning is that it does not require too much data. In the second experiment, we actually use only one standard video of each action as the data source, which greatly reduces the number of data samples required. However, there are also some potential problems. The Z-axis depth coordinates of MediaPipe have some errors and cannot be guaranteed to be 100% accurate, which also leads to the possibility of inaccurate recognition when the camera's viewpoint does not match the viewpoint of the pre-video data. This is the reason why the methodology above proposes the use of three views of the video, but this can only reduce its error to some extent and cannot completely solve the problem.

## V. Conclusion

In conclusion, this paper proposes two algorithms for recognizing human movements based on the MediaPipe framework. The first one uses the AdaBoost algorithm to recognize a specific pose, and the second one uses the FastDTW algorithm to recognize continuous fitness movements. In this paper, 413 complete videos of fitness movements are recognized using these methods, and their accuracy rate reaches more than 95% in the same viewpoint as the original video. Based on the methodology proposed, any system for action recognition can be developed.

## References

[1] C. Lugaresi et al., "MediaPipe: A Framework for Building Perception Pipelines." arXiv, Jun. 14, 2019. doi: 10.48550/arXiv.1906.08172.
[2] "Home," mediapipe. https://google.github.io/mediapipe/ (accessed Oct. 27, 2022).
[3] "Pose," mediapipe. https://google.github.io/mediapipe/solutions/pose.html (accessed Oct. 27, 2022).
[4] Y. Freund and R. E. Schapire, "A short introduction to boosting", J. Jpn. Soc. Artif. Intell., vol. 14, no. 5, pp. 771-780, Sep. 1999.
[5] R. E. Schapire, Y. Freund, P. Bartlett and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods", Ann. Statist., vol. 26, no. 5, pp. 1651-1686, 1998.
[6] S. Wu and H. Nagahashi, "Parameterized AdaBoost: Introducing a Parameter to Speed Up the Training of Real AdaBoost," in IEEE Signal Processing Letters, vol. 21, no. 6, pp. 687-691, June 2014, doi: 10.1109/LSP.2014.2313570.
[7] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119–139, 1997, doi: 10.1006/jcss.1997.1504.
[8] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence-rated predictions," Proceedings of the eleventh annual conference on Computational learning theory - COLT' 98, 1998.
[9] S. Salvador and P. Chan, "FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space," p. 11.
[10] "Hi Fitness Action Library: A complete collection of fitness actions, detailed action diagrams and action video teaching!" https://www.hiyd.com/dongzuo/ (accessed Oct. 27, 2022).