AI Personal Trainer- Biceps curls

By:

242062004- Shriya Haral

242062019- Divit Saini

Introduction:

Health and Fitness are simply some of many the different words used to describe people being in good condition. There are many different services and products on offer which promise to improve or maintain a state of wellbeing, and any (or all) of these goods and services might be considered to be part of the health and fitness industry. These can include things as variable as medical services through to sport, recreation, food and natural therapies.

There are also many different forms of working with the body for better physical fitness. Modalities such as calisthenics, jogging, and weight training were some of the first forms that became popular. Through the years, many other forms have developed such as aerobics, yoga (which has actually been around for centuries), Pilates and then hybrids from each of these such as yogalates, body pump, and water aerobics as well as many other forms of exercising the body.

Hitting at a gym is like a trend these days, everyone is concerned about their physical health more rather than their mental health. For which they spend hours and hours at the gym to build a good physic. Also, they spend a huge amount of money for a personal trainer.

Source: Getty Images [3]

Isn’t it costly? Is it worth to spend so much money?

The answer is yes, it is costly and it is not worth it to spend a huge amount of money. Instead we can use an AI personal Trainer.

So let us understand what AI personal trainer?

“AI personal trainers are **virtual trainers powered by artificial intelligence that help you to achieve your fitness goals**. It monitors your performance in real-time. It also adapts your workout accordingly. Instead of being stuffed with hundreds of workouts, this personal trainer offers smart workouts to achieve fitness goals.

Computer vision enables fitness apps to provide corrective feedback

Source: Kaia Health [2]

Why do we need AI Personal Trainer?

It is not possible for all to have a personal trainer at the gym. Also, it becomes difficult for a person to spend their morning/ evening at the gym due to a busy life. AI personal trainer can assist you wherever you want and whenever you want.

Requirements:

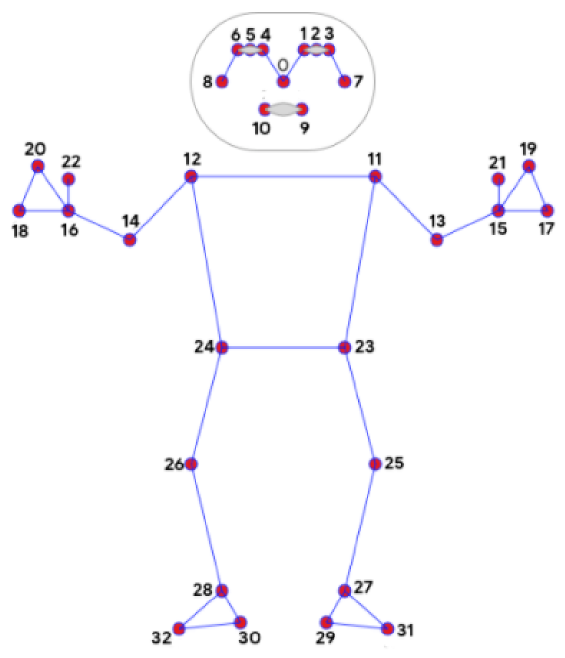
- Opencv
- Media Pipe
- Python Environment
- Web camera
- Time
- Math

Before implementation let us understand some concepts which are essential for implementation:

Let us understand each library in detail along with its installation in python.

Required Libraries:

- **Mediapipe:** Measurement of human pose from video plays a very important role in various programs such as exercise balance, sign language recognition, and full body control. For example, it can form the basis of yoga, dance, and fitness applications. It can also allow the overlay of digital content and information over the physical world to something that taxpayers dislike. MediaPipe Pose is an ML solution for tracking high physical fidelity, including 33 3D landmarks and a back-to-body separation mask from RGB video frames using our BlazePose-enabled research and ML Kit Pose Detection API. Current technologies are heavily dependent on powerful desktop environments, but our approach achieves real-time performance on many modern phones, desks / laptops, pythons and even the web.

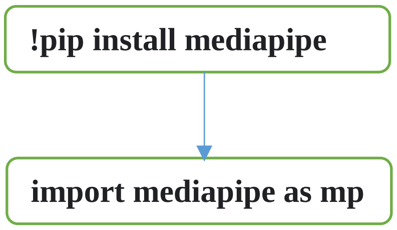


0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Pose landmarks [5]

Figure shows the points which are been allocated to a human body. This makes it easy while implementing any pose detection application. As with the help of these points one can easy build a model as per their needs.
For e.g. In our case we are implementing AI trainer – bices curls for which we require only hands, so we have used these lines of code to detect the points and to find the angle between them for each hand.
# **Right Arm:** angle = detector.findAngle(img, 12, 14, 16)
# **Left Arm:** angle = detector.findAngle(img, 11, 13, 15)

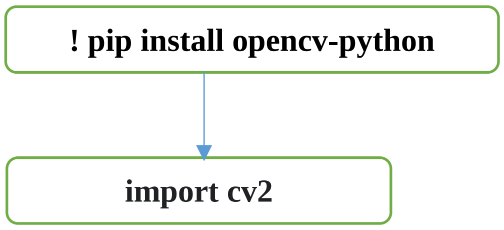
Steps for installing mediapipe in your machine:



```
!pip install mediapipe
import mediapipe as mp
```

- Opencv: **OpenCV** (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision. The library is cross-platform and free for use under the open-source Apache 2 License. It's features GPU acceleration for real-time operations. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human.

Steps for installing Opencv in your machine for python:



```
! pip install opencv-python
import cv2
```

Using Opencv document one can play around various functions used for image, videos, GUI, etc. The link for complete documentation is: https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

- **Numpy:** NumPy is the fundamental package for scientific computing in Python. It can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices. It provides standard trigonometric functions, functions for arithmetic operations, handling complex numbers, etc. NumPy has standard trigonometric functions which return trigonometric ratios for a given angle in radians.

  Steps for installing NumPy:

  ```
  !pip install numpy
  ```

  ```
  import numpy as np
  ```

- **Time:** Time provides various time-related functions. As we are using AI personal trainer for real time application, there is a need of time function.
  Steps for installing NumPy: Time is already installed in all the versions of python. So there is no need to install it. By passing the below command we can import time to our model.

  ```
  import time
  ```

- **Math:** It provides access to the mathematical functions defined by the C standard.
  These functions cannot be used with complex numbers; use the functions of the same name from the cmath module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don't is made since most users do not want to learn quite as much mathematics as required to understand complex numbers.
  Using math document one can understand various different functions using math as per the need.
  The link for complete documentation is: https://docs.python.org/3/library/math.html
  Steps for installing math function in python:

  ```
  !pip install python-math
  ```

  ```
  import math
  ```

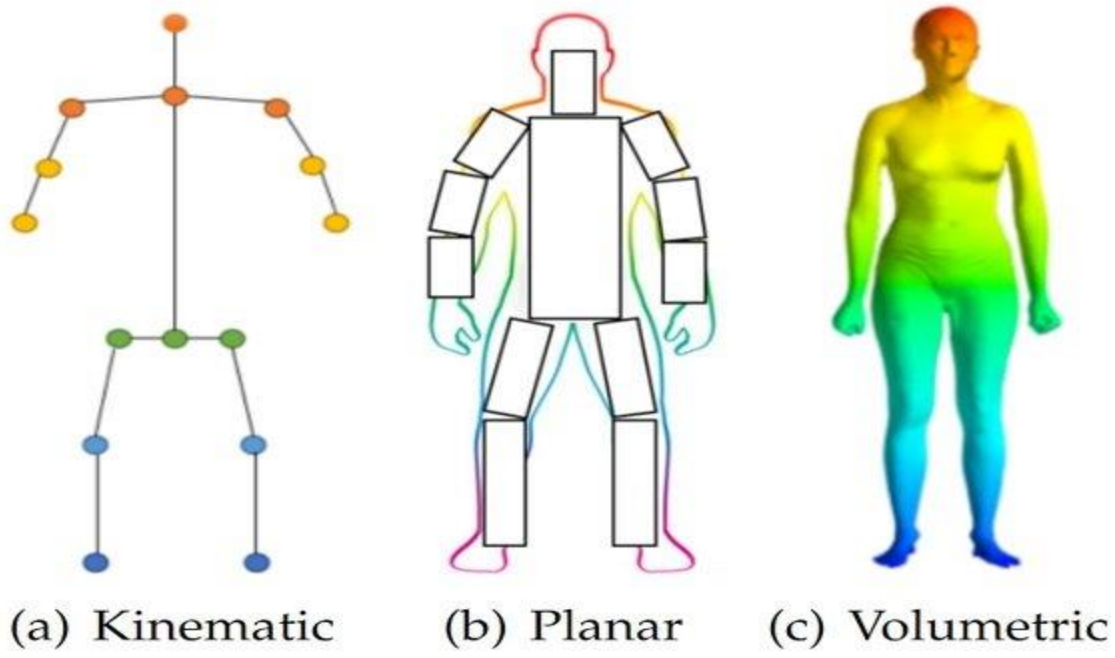
Our model's working:

Before going to the steps for coding, it is mandatory to understand some concepts that are essential for implementing AI personal trainer.

How will you detect poses using image, videos or real time data? Using Opencv it is easy to detect the several different objects, but detecting poses of a human being becomes bit tricky for that we use Pose estimator from media pipe. So let us first understand what is human pose estimator and its working.

What is Human Pose estimation?

Human pose estimation is a computer vision-based technology that detects and analyzes human posture. The main component of human pose estimation is the modeling of the human body. There are three of the most used types of human body models: skeleton-based model, contour-based, and volume-based.



Types of Human Pose models [6]

Kinematic-based model consists of a set of joints (keypoints) like ankles, knees, shoulders, elbows, wrists, and limb orientations comprising the skeletal structure of a human body. This model is used both in 2D and 3D human pose estimation techniques because of its flexibility.
Planar-based model consists of the contour and rough width of the body torso and limbs, where body parts are presented with boundaries and rectangles of a person's silhouette.
Volume-based model consists of 3D human body shapes and poses represented by volume-based models with geometric meshes and shapes, normally captured with 3D scans.

How Human Pose Estimator detects poses for our model?
It detects the poses based on these 3 things:

- Points
- Positions
- Angles

Points: The points are obtained using Pose landmarks which is present in Mediapipe- Pose. This can be seen in the figure Pose Landmarks to have a clear idea. Every part of the human body is assigned with some values which makes it easy for pose estimation of that desired point. Later these points help to find the angle between the desired points (i.e. right arm, left arm, left leg and so on.) In our case the points which are been used to find the angles are as follows:
# **Right Arm:** angle = detector.findAngle(img, 12, 14, 16)
# **Left Arm:** angle = detector.findAngle(img, 11, 13, 15)

Positions: The name itself tells what position does. They help us to get the position of a person which can be used to find the angle between the landmarks. It will be clearer when we perform actual positioning for our model.

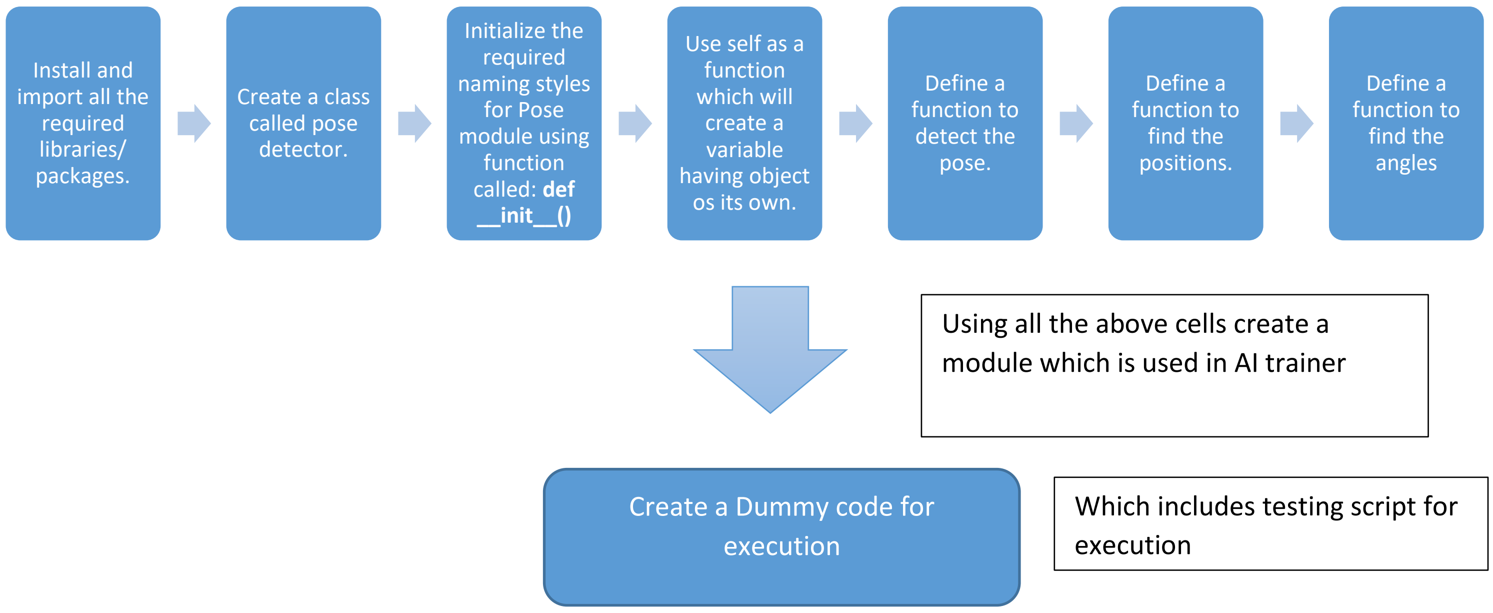
Angles: With the help of points and positions we find the angle between three desired points. They play an important role in Pose estimation as they help us to understand the Human pose better and we can actually implement a model using Opencv and get the desired results.

Based on these above mentioned points the Human position estimation is implementation for real time use also for an image and video data.

Let us now understand the required steps for AI personal Trainer.
Required Datafiles for the implementation:

1. Image
2. Video
3. Web camera

1) **Pose Module:** This module is responsible for all the pose estimation which we need for our AI Trainer. We are importing Pose Module to our AI trainer to perform AI trainer real time application.
Steps for implementation are as follows:



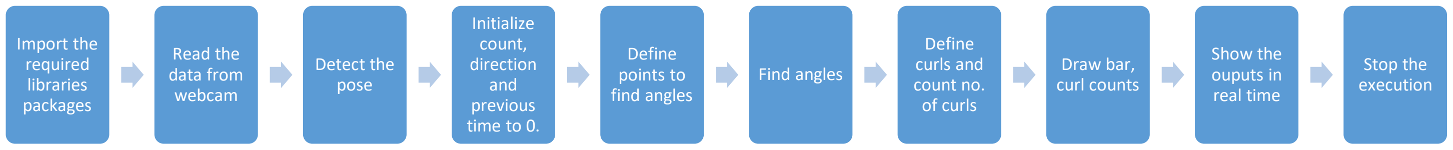
Install and import all the required libraries/ packages.
Create a class called pose detector.
Initialize the required naming styles for Pose module using function called: **def __init__()**
Use self as a function which will create a variable having object os its own.
Define a function to detect the pose.
Define a function to find the positions.
Define a function to find the angles
Using all the above cells create a module which is used in AI trainer
Create a Dummy code for execution
Which includes testing script for execution

Testing script includes:

```
def main(): #Dummy code (testing script)
    cap = cv2.VideoCapture('curls.mp4') #reading the desired video
    pTime = 0 #previous Time
    detector = poseDetector() #detect pose
    while True:
        success, img = cap.read()
        img = detector.findPose(img) #detecting Pose
        lmList = detector.findPosition(img, draw=False) #detecting positions
        if len(lmList) != 0:
            print(lmList[14])
            cv2.circle(img, (lmList[14][1], lmList[14][2]), 15, (0, 0, 255), cv2.FILLED)
        cTime = time.time() #current time
        fps = 1 / (cTime - pTime)
        pTime = cTime
        cv2.putText(img, str(int(fps)), (70, 50), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 0), 3) #Text display
        cv2.imshow("Image", img) #Display image
        cv2.waitKey(1) #End

if __name__ == "__main__": #to execute main function as a module
    main()
```

2. AI Trainer: This module is used for executing for real time application as well as for video and image data. As mentioned earlier for this module we need to import Pose Module in this module as it contains basic requirements for pose estimation.

Steps for Implementing AI Personal Trainer:


Import the required libraries packages
Read the data from webcam
Detect the pose
Initialize count, direction and previous time to 0.
Define points to find angles
Find angles
Define curls and count no. of curls
Draw bar, curl counts
Show the ouputs in real time
Stop the execution

Code:

So in our project we need two ipynb files those are **AI trainer and Pose Module**.
In pose module, we have performed all the required steps to build a pose module and in AI trainer along with video and real time detection we have import the pose module which will help in implementing AI personal trainer real time application.

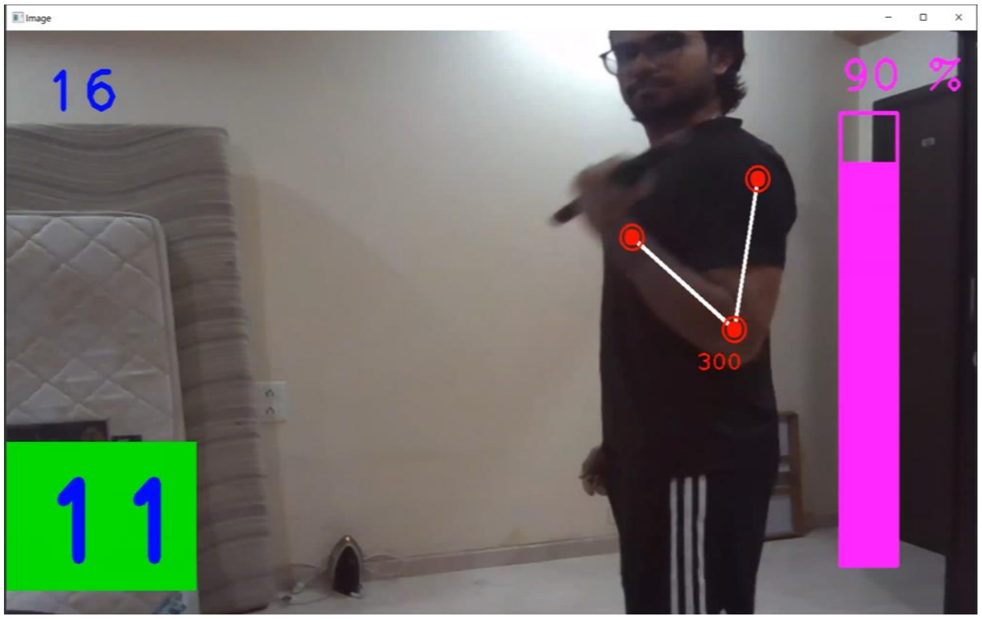
```
#For Ai personal Trainer the code is:
#importing all the required libraries
import cv2
import numpy as np
import time
import posemodule as pm #Pose Module

cap = cv2.VideoCapture(0) #0 for web cam . Also you can give the path containing the video file for capturing the vid
ep
detector = pm.poseDetector() #it will detect the pose
count = 0 #Initially the count is 0
dir = 0 #Direction
pTime = 0 #previous time
while True:
    success, img = cap.read()
    img = cv2.resize(img, (1280, 720)) #resize the video for better visuals
    # img = cv2.imread("AiTrainer/test.jpg") #this is used when we are passing image as an ip.
    img = detector.findPose(img, False) #It will detect the pose only once.False when we don't want to draw. The defau
lt value is True (i.e.we draw the image)
    lmList = detector.findPosition(img, False) #this will find the position(i.e. Landmark values). False is used when we
don't want to draw.
    # print(lmList) #Shows the landmarks in the form of points
    if len(lmList) != 0:
        # Right Arm
        angle = detector.findAngle(img, 12, 14, 16) #12,14,16 are the points on the right arm using mediapipe Pose land
marks to find the angle between them
        # # Left Arm
        #angle = detector.findAngle(img, 11, 13, 15,False) #Same as what we did for right arm.
        per = np.interp(angle, (210, 310), (0, 100)) # Convert (210,310) range into (0,100) in range
        bar = np.interp(angle, (220, 310), (650, 100)) #Convert (220,310) range into (650,100) in range for bars max valu
e is 650 and min value is 100.
        # print(angle, per)
        # Check for the dumbbell curls
        color = (255, 0, 255) #Purple
        if per == 100:
            color = (0, 255, 0) #Green
            if dir == 0: #going up
                count += 0.5 #Add 0.5 to the count
                dir = 1 #going down
        if per == 0:
            color = (0, 255, 0) #Green
            if dir == 1: #going up
                count += 0.5 #Add 0.5 to the count
                dir = 0 #going down
        print(count) #It will display the counts for dumbell curls (for going up and down) it counts it as 1 when it goes up
 and comes down .
```

```
# Draw Bar
      cv2.rectangle(img, (1100, 100), (1175, 650), color, 3)
      cv2.rectangle(img, (1100, int(bar)), (1175, 650), color, cv2.FILLED)
      cv2.putText(img, f'{int(per)} %', (1100, 75), cv2.FONT_HERSHEY_PLAIN, 4,
               color, 4) #using color so that we will understand when it had reached the max or min position

      # Draw Curl Count
      cv2.rectangle(img, (0, 450), (250, 720), (0, 255, 0), cv2.FILLED)
      cv2.putText(img, str(int(count)), (45, 670), cv2.FONT_HERSHEY_PLAIN, 15,
               (255, 0, 0), 25) #for displaying the count on the screen
   cTime = time.time() #current time
   fps = 1 / (cTime - pTime) # 1- current time/previous time
   pTime = cTime
   cv2.putText(img, str(int(fps)), (50, 100), cv2.FONT_HERSHEY_PLAIN, 5,
            (255, 0, 0), 5) #Displaying it on the screen
   cv2.imshow("Image", img) #Display
   key = cv2.waitKey(1)
   if key == ord('q'): #for stopping the implementation
      break
```

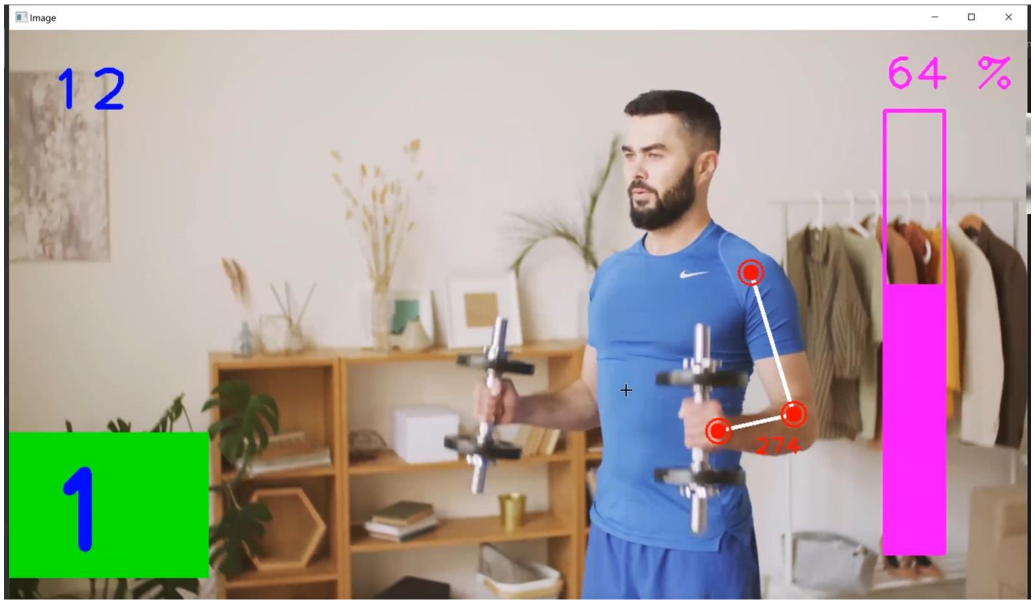
Naming styles required for Pose Module:
STATIC_IMAGE_MODE: If set to false, the solution treats the input images as a video stream. It will try to detect the most prominent person in the very first images, and upon a successful detection further localizes the pose landmarks. In subsequent images, it then simply tracks those landmarks without invoking another detection until it loses track, on reducing computation and latency. If set to true, person detection runs every input image, ideal for processing a batch of static, possibly unrelated, images. Default to false.
MODEL_COMPLEXITY: Complexity of the pose landmark model: 0, 1 or 2. Landmark accuracy as well as inference latency generally go up with the model complexity. Default to 1.
SMOOTH_LANDMARKS: If set to true, the solution filters pose landmarks across different input images to reduce jitter, but ignored if static_image_mode is also set to true. Default to true.
ENABLE_SEGMENTATION: If set to true, in addition to the pose landmarks the solution also generates the segmentation mask. Default to false.
SMOOTH_SEGMENTATION: If set to true, the solution filters segmentation masks across different input images to reduce jitter. Ignored if enable_segmentation is false or static_image_mode is true. Default to true.
MIN_DETECTION_CONFIDENCE: Minimum confidence value ([0.0, 1.0]) from the person-detection model for the detection to be considered successful. Default to 0.5.
MIN_TRACKING_CONFIDENCE: Minimum confidence value ([0.0, 1.0]) from the landmark-tracking model for the pose landmarks to be considered tracked successfully, or otherwise person detection will be invoked automatically on the next input image. Setting it to a higher value can increase robustness of the solution, at the expense of a higher latency. Ignored if static_image_mode is true, where person detection simply runs on every image. Default to 0.5.

```
#For pose module the code is:
import cv2 #for opencv
import mediapipe as mp #to get pose estimation
import time #Time
import math #mathematical function

class poseDetector():#create objects and methods for pose detection
#self is when we want variables to have its own object.
    def __init__(self, static_image_mode=False, model_complexity=False, smooth_landmarks=True,
                 min_detection_confidence=0.5, min_tracking_confidence=0.5): #Initalization

        self.static_image_mode = static_image_mode
        self.model_complexity = model_complexity
        self.smooth_landmarks = smooth_landmarks
        self.min_detection_confidence = min_detection_confidence
        self.min_tracking_confidence = min_tracking_confidence
        self.mpDraw = mp.solutions.drawing_utils #for drawing utils
        self.mpPose = mp.solutions.pose #Pose

        self.pose = self.mpPose.Pose(static_image_mode=self.static_image_mode, model_complexity=self.model_com
plexity,
                                     smooth_landmarks=self.smooth_landmarks,
                                     min_detection_confidence=self.min_detection_confidence,
                                     min_tracking_confidence=self.min_tracking_confidence)

    #For finding pose
    def findPose(self, img, draw=True): #Draw= True to draw on the image
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) #Converting the BGR color to RGB
        self.results = self.pose.process(imgRGB) #detection of pose in RGB
        if self.results.pose_landmarks: #Results
            if draw: #Draw landmarks
                self.mpDraw.draw_landmarks(img, self.results.pose_landmarks,self.mpPose.POSE_CONNECTIONS) #D
rawing points with connections ie. lines of landmarks.
        return img #return image

    #For finding positions
    def findPosition(self, img, draw=True):
        self.lmList = [] #landmarks list
        if self.results.pose_landmarks:
            for id, lm in enumerate(self.results.pose_landmarks.landmark): #to get the loop count use enumerate
                h, w, c = img.shape #image shape of h=height, w=width, c=channel
                # print(id, lm)
                cx, cy = int(lm.x * w), int(lm.y * h) #lm.x * w will give x of our landmark similarly lm.y * h will give y of
 our landmark
                self.lmList.append([id, cx, cy]) #
                if draw:
                    cv2.circle(img, (cx, cy), 5, (255, 0, 0), cv2.FILLED) #Draw circle (255, 0, 0) is blue color
        return self.lmList #returns lmlist
    def findAngle(self, img, p1, p2, p3, draw=True):
        # Get the landmarks
        x1, y1 = self.lmList[p1][1:] #getting the landmarks at the p1 and slicing it from point 1 till the end
        x2, y2 = self.lmList[p2][1:] #getting the landmarks at the p2 and slicing it from point 1 till the end
        x3, y3 = self.lmList[p3][1:] #getting the landmarks at the p3 and slicing it from point 1 till the end
        # Calculate the Angle
        angle = math.degrees(math.atan2(y3 - y2, x3 - x2) -
                             math.atan2(y1 - y2, x1 - x2)) #Using math function calculate the angle between this landmarks in 
degrees.
        #when the angles are negative in that case use this condition ie. (360- the value) to solve the negative angles.
        if angle < 0:
            angle += 360
        # print(angle) #printing the angle
        # Draw
        if draw: #for making sure that we are getting the points correctly.
            cv2.line(img, (x1, y1), (x2, y2), (255, 255, 255), 3) #for drawing a line between x1,y1 and x2,y2. 255 is used 
for white colour and value of 3 is used for thickness.
            cv2.line(img, (x3, y3), (x2, y2), (255, 255, 255), 3)
            cv2.circle(img, (x1, y1), 10, (0, 0, 255), cv2.FILLED) #for drawing a circle, 10 represents the size and 0,0,25
5 is used for red colour with thickness of 2
            cv2.circle(img, (x1, y1), 15, (0, 0, 255), 2)
            cv2.circle(img, (x2, y2), 10, (0, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), 15, (0, 0, 255), 2)
            cv2.circle(img, (x3, y3), 10, (0, 0, 255), cv2.FILLED)
            cv2.circle(img, (x3, y3), 15, (0, 0, 255), 2)
            cv2.putText(img, str(int(angle)), (x2 - 50, y2 + 50), #x2 is the middle point for which we don't want angle exa
ctly the the point so we subtract 50 so that we get values near to the center points.
                        cv2.FONT_HERSHEY_PLAIN, 2, (0, 0, 255), 2) #0,0,255 is used for red color , FONT_HERSHEY_
PLAIN is used to provide fonts with thickness of 2
        return angle #returns angles

def main(): #Dummy code (testing script)
    cap = cv2.VideoCapture('curls.mp4') #reading the desired video
    pTime = 0 #previous Time
    detector = poseDetector() #detect pose
    while True:
        success, img = cap.read()
        img = detector.findPose(img) #detecting Pose
        lmList = detector.findPosition(img, draw=False) #detecting positions
        if len(lmList) != 0:
            print(lmList[14])
            cv2.circle(img, (lmList[14][1], lmList[14][2]), 15, (0, 0, 255), cv2.FILLED)
        cTime = time.time() #current time
        fps = 1 / (cTime - pTime)
        pTime = cTime
        cv2.putText(img, str(int(fps)), (70, 50), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 0), 3) #Text display
        cv2.imshow("Image", img) #Display image
        cv2.waitKey(1) #End
if __name__ == "__main__": #to execute main function as a module
    main()
```

Results:



Output in real time



Output for Video

Demonstration: Attached link is to the video which we have recorded in real time. The video will show how it works in real time (i.e. detect the points, finds the angle, poses and so on). Hope this will help to have a clear understanding of our model.

Link: https://drive.google.com/drive/folders/19EbPcDKV6ApDLqIo-u3wGQ6ABZDlgA24?usp=sharing

Advantages of AI personal trainer:

- User friendly
- Low cost
- Improvement In Performance of a person
- No restriction of area and time
- Live counts

- Work out in the privacy and comfort of your own home.

Applications of AI personal Trainer:

Here are the top applications:

- AI-powered wristbands and gadgets
- AI-driven diet and nutrition apps
- AI-powered yoga suits to monitor movement and set accurate posture during an asana

Future Scope:

- Interactive Model who can correct us like a personal trainer in many different languages such as Hindi, Marathi, Bengali, etc.
- Application based model
- Application that can provide the user to track daily and monthly record of his fitness.
- A Customized diet plan according to the user needs.

Conclusion:

Artificial Intelligence has entered nearly every industry, health and fitness included. The technology enables people to stay healthy well by achieving workout goals without going into the fitness center. We barely have time to practice on a regular basis as our lives get busier. However, AI tries to overcome this problem in fitness. For this way we have implemented an AI based personal trainer for real time use. This model can be used for images and as well as for videos which are not in real time.

References:

1. https://google.github.io/mediapipe/getting_started/python.html
2. https://www.thetimes.co.uk/article/kaia-the-app-thats-a-pocket-personal-trainer-ptl6btrbf
3. https://api.time.com/wp-content/uploads/2020/03/gym-coronavirus.jpg?w=824&quality=70
4. https://cocodataset.org/#keypoints-2020
5. https://google.github.io/mediapipe/solutions/pose#python-solution-api
6. https://viso.ai/deep-learning/pose-estimation-ultimate-overview/
7. Open Pose: Real time Multi-Person 2D Pose Estimation using Part Affinity Fields: https://arxiv.org/pdf/1812.08008.pdf