International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Advancements in Image Detection: A Comprehensive Approach to Object Localization and Classification Using Deep Learning Techniques

Sushmit Kadam

B.Tech Student, Department of Data Science, Mpstme, Nmims.

Abstract

Contemporary fitness gadgets create the demand for intelligent platforms to observe, teach, and provide real-time feedback as one is performing individual exercises. Therefore, this paper proposes an AI-powered personalized trainer that comes with the task of grading one of the simplest yet basic strength training exercises known as bicep curls. Using computer vision for the system, the user's posture and movement are tracked online, with emphases on main joints like shoulder, elbow, and wrist.

The AI trainer then uses the highly accurate landmark extraction from the user's body with the help of the Mediapipe library for proper measurement of angles. It evaluates the position of the elbow with respect to the shoulder and sends immediate feedback to the user by monitoring repetition speed. Correct form, key in the prevention of injuries and effectiveness during workouts, is envisioned to the user in real time, as any deviations fire up the proper feedback messages to motivate the user to engage in a better technique.

Moreover, the trainer includes a repetition counter that further tracks the performance of the user and adds more value to his workout. The system provides easy use of interface that visualizes the results for the users in order to motivate and engage themselves.

The project represents the integrated approach of computer vision, machine learning, and real-time feedback mechanisms that empower fitness enthusiasts toward an optimized routine that brings out the best results in an effective and safe manner.

Keywords: Fitness technology, AI-powered trainer, Bicep curls, Computer vision, Mediapipe, Body posture analysis.

Introduction

This report describes my capstone project in the domain of AI and Machine Learning: AI Fitness Trainer. In building up the AI Trainer, one received foundational knowledge in computer vision, studying core concepts and techniques that are to be used as the foundation for real-time body movement analysis. I developed expertise in pose estimation by tracking the key joints in the process of bicep curls using pre-trained models through hands-on experience with the Mediapipe library. While working on this project, there were a lot of challenges too which I overcame in the process of building the project.

Background

The fitness industry has taken a turn for the better, especially with technological advancements in the field of fitness tracking and personalized training. More people are becoming aware of their health, so there is always a growing demand for solutions that allow one to track his or her fitness metrics while giving real-time guidance and feedback. It is the integration of AI and computer vision into the fitness regime that can make workouts even more effective, personalized, and safe.

This technology is developed on a wider variety of systems that can track exercises, form analysis, and corrective advice. It is also part of the larger trend toward intelligent personal training systems individually tailored to the users' needs-from beginners up to professional athletes. Despite all these advances, there are still outstanding issues with the entire industry related to fitness technology.

Key Issues in the Industry

Body Movement Analysis: Accuracy and Precision: The first and foremost concern of this sector is how accurately AI-powered fitness systems can detect and interpret body movements. Proper form detection becomes critical and its correction as well, since minor inaccuracies may lead to ineffective workouts or even physical injuries.

User Engagement and Motivation: There is no lack of different applications and systems for fitness, but keeping the user's interest across a longer period of time remains a challenge. Users quit their exercise programs because they are not motivated, or they get no feedback on improvements, or they simply cannot notice any relevant improvement.

Accessibility and Usability: Another issue is the accessibility of these technologies to the widest range of users and their usability, independent from one's technical proficiency or physical fitness. Many users are discouraged from using these solutions by complicated interfaces or systems with heavy setup.

Cost and Affordability: High pricing for sophisticated systems of AI and machine learning normally acts as a barrier, allowing access only to more customers. Ensuring value for money with appropriate pricing of the technology remains a big challenge to industry growth.

Integration with existing ecosystems: Most of the enthusiasts in keeping themselves fit already use a variety of devices and applications. Seamless integration of new technologies with existing health and fitness ecosystems is often challenging, yet crucial to their success.

Potential for Growth

The AI-powered fitness technology market is going to see phenomenal growth due to growing demand for customized, data-driven workout solutions. Also, with the advancement of AI and computer vision, there comes a prospect of improving accuracy, accessibility, and engagement of the applications. The possibility of real-time feedback, personalized coaching, and the ability to seamlessly mesh these into existing health ecosystems could solve some of the key challenges these apps presently face: user motivation and injury prevention, to name just a couple. This would mean that, with the growth in the market, AI-powered systems could be deployed to optimize workout routines and reach fitness goals, hence becoming very important tools to drive the next wave of innovation in the fitness market.

Image Detection

Introduction

Image detection, sometimes referred to as object detection, refers to locating and classifying objects pres-

ent within an image or a frame of a video using computer vision. The goal of image detection algorithms is the identification and location of multiple objects present within an image and assigning them special labels or classes. Several of these applications demonstrate a growing interest in the application of this technology to several fields ranging from self-driving cars, through surveillance systems, face recognition, medical imaging, to virtual reality.

Image detection generally consists of the following major steps:

1. **Image Preprocessing:** Generally speaking, the input image is preprocessed to enhance features and suppress noise. Standard image preprocessing includes resizing, normalization, and reduction of noise.
2. **Feature Extraction:** Edges, corners, or textures are extracted as features from the image in order to represent something special in the image. Feature extraction will enable an algorithm to identify objects based on some specific patterns.
3. **Object Localization:** It addresses the exact location of the object in an image. During this step, the coordinates of the bounding box of the object are identified, indicating where the object is situated within the image.
4. **Object Classification:** Once localization has occurred, objects detected need to be classified into a category or class. This classification labels each detected object according to what the object represents, for instance, car, person, dog.
5. **Post-processing:** Non-maximum suppression is one of the post-processing methods applied to refine the object detected, removing duplicate detections while improving the general accuracy of the test results.
6. **Output:** The bounding boxes of the detected objects and their corresponding class labels will be the final output, which clearly represents the identified objects in an input image.

In modern times, image detection makes use of deep learning models, especially CNNs, to learn and extract the relevant features from an image automatically. Such models have increased the accuracy and speed of object detection tasks in reality: examples include Faster R-CNN, YOLO, and SSD.

Libraries Used

1. **OpenCV:** It is the most popular open-source library of computer vision and machine learning software. It provides tools for real-time image and video processing, hence very important in capturing video frames, color space conversion, and image manipulation.
2. **Mediapipe:** This is a Google library for obtaining ready-to-use, high-quality implementations of various computer vision tasks, including human pose estimation. The Pose module from Mediapipe has been employed in this project for real-time pose detection.
3. **NumPy:** The NumPy library is among the most efficient libraries for numerical computation in Python. It provides support for very large multi-dimensional arrays and matrices, along with a set of mathematical functions operating on these arrays. NumPy is used in this project to efficiently perform array operations for various computations that involve pose estimation and angle calculation.
4. mp_drawing and mp_pose-these are specific modules in the Mediapipe library that are used for drawing the landmark on the detected human pose, mp_drawing, and for accessing the pose landmark, mp_pose. These will visualize the detected body joints and provide an easy way to analyze the pose.
5. **Streamlet:** Streamlit offers a simple, Python-script-based way to create web applications. You can build an interactive application using familiar Python scripting.

These collectively allow this project to process video input in real time, estimate the human pose, calculate the joint angles, assess the form of exercises, and provide instantaneous feedback to users.

Code:

```
import cv2
import mediapipe as mp
import numpy as np
mp_drawing = mp.solutions.drawing_utils
mp_pose = mp.solutions.pose
```
✓ 11.1s

```
%pip install mediapipe opencv-python
%pip install streamlit
import streamlit as st
!pip install jupyter
```

In this segment, the necessary libraries are imported. cv2 is OpenCV, used for image and video processing, mediapipe is used for human pose estimation, numpy is employed for numerical operations, mp_drawing and mp_pose are specific modules from mediapipe used for drawing landmarks and accessing pose landmarks, respectively. Also streamlet is used for deployment.

```
# Define Streamlit app title and description
st.title("Bicep Curl Counter with Pose Estimation")
st.write("Real-time curl counter with form feedback.")
```

```
# Main Streamlit app logic
def main():
    # Your Streamlit app content goes here
    st.title("Bicep Curl Counter with Pose Estimation")
    st.write("Real-time curl counter with form feedback.")
```

This segment defines the streamlet app title and description to be displayed as output.

```
def calculate_angle(a,b,c):
    a = np.array(a) # First
    b = np.array(b) # Mid
    c = np.array(c) # End

    radians = np.arctan2(c[1]-b[1], c[0]-b[0]) - np.arctan2(a[1]-b[1], a[0]-b[0])
    angle = np.abs(radians*180.0/np.pi)

    if angle >180.0:
        angle = 360-angle

    return angle
```
✓ 0.0s

This segment defines a function calculate_angle that computes the angle between three points using trigonometric calculations. This function is used to determine the angles at the shoulder, elbow, and wrist joints during the bicep curl exercise.

```
cap = cv2.VideoCapture(0)

# Curl counter variables
counter = 0
stage = None

# Previous angle for speed evaluation
prev_angle = None
```

These lines initialize the video capture from the default camera and the variables counter for counting repetitions and stage to track the exercise stage (up or down). prev_angle stores the previous joint angle for evaluating repetition speed.

```
## Setup mediapipe instance
with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
    while cap.isOpened():
        ret, frame = cap.read()

        # Recolor image to RGB
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image.flags.writeable = False

        # Make detection
        results = pose.process(image)

        # Recolor back to BGR
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
```

In this segment, a mediapipe.Pose instance is created with specified confidence thresholds. Inside the loop, frames from the video capture are read. The frames are converted to RGB, processed using the Pose model, and then converted back to BGR for rendering.

```
# Extract landmarks
try:
    landmarks = results.pose_landmarks.landmark

    # Get coordinates
    shoulder = [landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].x,landmarks[mp_pose.PoseLandmark.LEFT_SHOULDER.value].y]
    elbow = [landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].x,landmarks[mp_pose.PoseLandmark.LEFT_ELBOW.value].y]
    wrist = [landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].x,landmarks[mp_pose.PoseLandmark.LEFT_WRIST.value].y]

    # Calculate angle
    angle = calculate_angle(shoulder, elbow, wrist)
```

- This part of the code extracts landmark coordinates for the left shoulder, elbow, and wrist joints from the processed pose results.

Using the calculate_angle function, the joint angle for the bicep curl is computed based on the positions of the shoulder, elbow, and wrist joints.

```
# Assess elbow and shoulder alignment
shoulder_x = shoulder[0]
elbow_x = elbow[0]
if shoulder_x > elbow_x:
    # Elbow is not aligned with the shoulder
    feedback_message = "Elbow Not Aligned"
    feedback_color = (0, 0, 255)
else:
    feedback_message = "Good Form"
    feedback_color = (0, 255, 0)
```

This segment assesses the alignment of the elbow concerning the shoulder. If the elbow is not aligned, a feedback message indicating improper alignment is generated, displayed in red. If the alignment is correct, a "Good Form" message in green is displayed.

```
# Assess repetition speed
rep_speed_feedback = ""
if prev_angle is not None:
    speed = abs(angle - prev_angle)
    if speed > 30:
        rep_speed_feedback = "Slow down!"
        feedback_color = (0, 0, 255)
```

This part evaluates the repetition speed by comparing the current joint angle with the previous angle. If the speed is too fast (angle change greater than 30 degrees), a "Slow down!" message in blue is displayed.

```
# Update previous angle
prev_angle = angle

# Visualize angle
cv2.putText(image, str(angle),
            tuple(np.multiply(elbow, [640, 480]).astype(int)),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)
```

The current angle is stored as the previous angle for the next iteration.

The calculated joint angle is visualized on the video feed by overlaying the angle value near the elbow joint.

```
# Curl counter logic
if angle > 160:
    stage = "down"
if angle < 30 and stage =='down':
    stage = "up"
    counter += 1
    print("Rep Count:", counter)
```

These lines implement the logic for counting repetitions. When the angle is greater than 160 degrees, the exercise is in the "down" stage. If the angle falls below 30 degrees and the exercise was in the "down" stage, it is considered a complete repetition, and the counter is incremented.

```
except:
    pass

# Render curl counter and feedback
cv2.rectangle(image, (0, 0), (225, 123), (245, 117, 16), -1)

# Rep data
cv2.putText(image, 'REPS', (15, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
cv2.putText(image, str(counter),
            (10, 80),
            cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 2, cv2.LINE_AA)

# Stage data
cv2.putText(image, 'STAGE', (65, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
cv2.putText(image, stage,
            (60, 80),
            cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255, 255), 2, cv2.LINE_AA)

# Feedback data
cv2.putText(image, feedback_message, (10, 110), cv2.FONT_HERSHEY_SIMPLEX, 0.5, feedback_color, 1, cv2.LINE_AA)
cv2.putText(image, rep_speed_feedback, (10, 140), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)
```

This try-except block ensures that if pose landmarks are not detected properly, the program continues without raising errors.

The code renders feedback messages, repetition counts, and exercise stage information on the video feed. Feedback messages are displayed in the lower part of the video window.

```
        # Render detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=2),
                                  mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                                  )

        cv2.imshow('Mediapipe Feed', image)

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

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

✓ 12.3s

```
Rep Count: 1
Rep Count: 2
Rep Count: 3
```

In this segment, the code renders the detected pose landmarks on the video feed using mp_drawing.draw_landmarks. The video window is displayed as 'Mediapipe Feed'. The program continues to run until the 'q' key is pressed, after which it releases the video capture and closes all windows.

Output:

Repetition Count (REPS):

The number of completed bicep curl repetitions is displayed at the top left corner of the video feed. It starts counting when the user lowers their arm (angle > 160 degrees) and then raises it again (angle < 30 degrees) completing one full repetition.

Example: "REPS: 5" indicates the user has completed 5 full bicep curl repetitions.

Exercise Stage (STAGE):

The current stage of the bicep curl exercise is displayed at the top middle of the video feed. It shows either "up" or "down" based on the position of the user's arm.

Example: "STAGE: down" indicates the user is in the downward phase of the bicep curl.

Elbow Alignment Feedback:

Immediate feedback on the alignment of the user's elbow concerning the shoulder is displayed at the bottom left corner of the video feed. If the elbow is not aligned properly with the shoulder, it shows "Elbow Not Aligned" in red.

Example: "Elbow Not Aligned" indicates the user needs to adjust their elbow position.

Repetition Speed Feedback:

Feedback regarding the speed of the bicep curl repetitions is displayed at the bottom right corner of the video feed. If the user is performing the repetitions too quickly, it shows "Slow down!" in blue.

Example: "Slow down!" indicates the user needs to perform the exercise at a slower pace.

Visual Angle Representation:

The angle formed by the shoulder, elbow, and wrist joints is displayed near the elbow joint. This visual representation helps users understand the angle at which they are performing the bicep curls.

Real-time Pose Landmarks:

The video feed displays the user's real-time pose landmarks, including key joints like shoulders, elbows, and wrists, overlaid on the video. These landmarks provide a visual representation of the user's body posture and movement.

FUTURE SCOPE

The future looks bright for the AI Trainer project, which assesses biceps curls and provides real-time feedback in the domain of health and fitness technology. Possible future scopes for this project include:

Multi-Exercise Support: Extend the application by providing support for several exercises other than bicep curls; introduce various exercise forms and offer feedback on scores of exercises taken up by people to take up strength training and target a wide section of the audience.

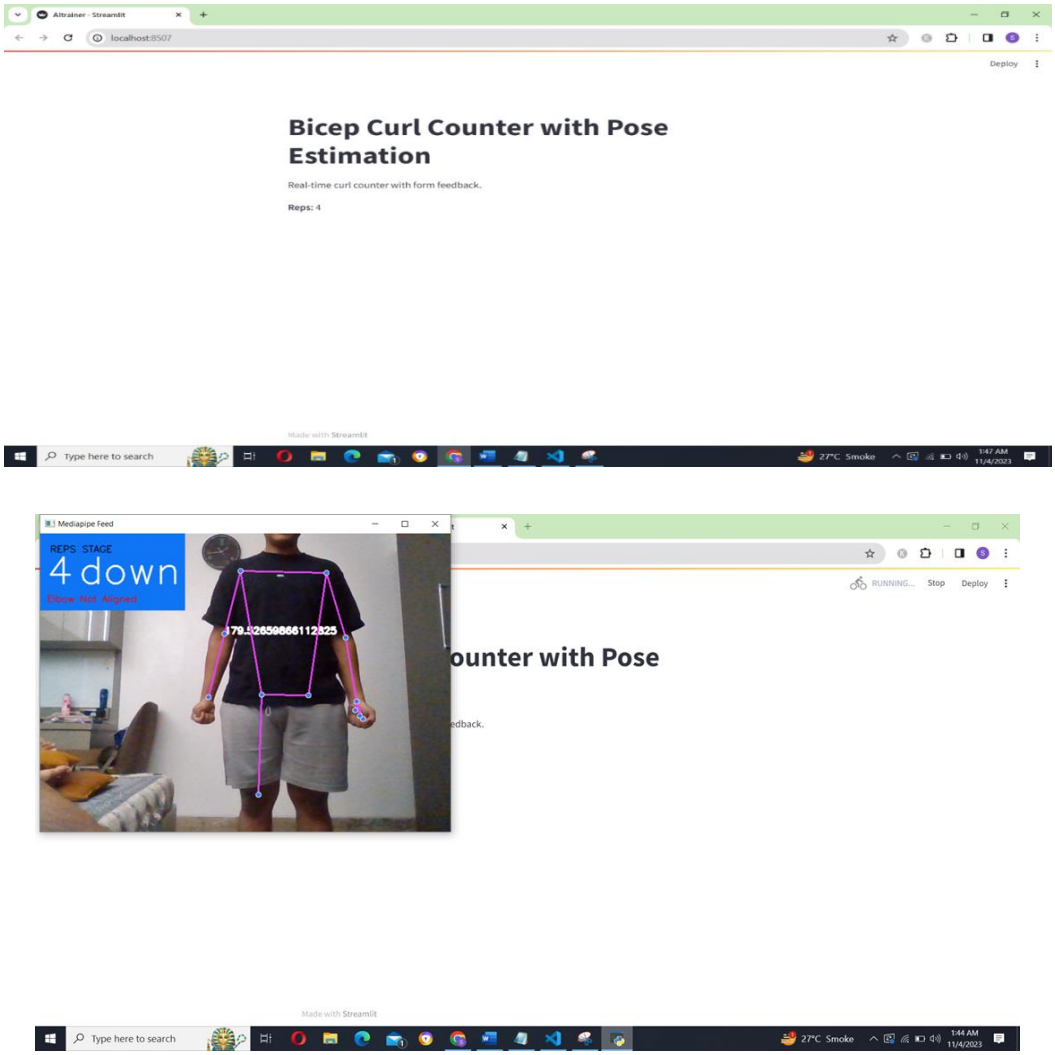
1. **Workouts that can be tailored:** Allow any user the capability to custom-make their workouts in the application. This allows the user to set goals, track progress, and receive tailored feedback based on individualized objectives.
2. **Mobile Application:** A mobile application of the AI Trainer should be developed, enhancing accessibility and allowing users to run workouts with just their smartphones or even tablets. The application can include value additions such as video tutorials, exercise history, and user profiles.
3. **Data Analytics and Insights:** Integration of data analytics to track the pattern of the users' workouts through time. This would provide the user with insights into how they've progressed and which areas they might want to improve, and give suggestions on how to optimize their workout. Users could get a weekly or monthly report of their accomplishments in fitness and areas to focus on.
4. **Virtual Trainer Integration:** Integrate the feature of a virtual trainer that will be capable of natural language processing. Users can ask about their form, get well-descriptive answers, and immediate responses that make the interaction quite exciting.
5. **Gamification Elements:** Include some gamification elements in the workout session for making it more engaging. Incorporate challenges, rewards, leaderboards, and achievements that will motivate

users in trying to achieve their fitness goals.

6. **Wearable Device Integration:** Integrate the application with wearable fitness devices-smartwatches or fitness trackers-to capture more data on heart rate, calories burnt, and motion tracking. Use this information for in-depth analysis of the users' workout and overall health.
7. **Accessibility Features:** Embed the application with accessibility features such as voice guidance and visual cues for disabled persons. Ensure that the application caters to the needs of all people, irrespective of physical abilities.
8. **Collaboration with Fitness Professionals:** Team up with fitness trainers, physiotherapists, and other health professionals to validate the feedback provided by the app. Their input will enhance the helpfulness of the feedback because it would be done according to the best practices in training.
9. **Real-time Video Analysis:** Real-time video analysis will be integrated into the application by allowing users to record their workout sessions. The recordings can then be analyzed by the application, which would provide feedback and store the videos so that users can review their form and progress.

By developing all these future scopes, the AI Trainer project would be able to shape itself into a full-fledged fitness platform-offering users personalized, entertaining, and efficient exercise routines.

Interface:



Conclusion:

The AI Trainer project finally presented a full-scale and interactive solution for real-time bicep curl assessment with simultaneous feedback on form and repetition speed. This work has shown what technology can do, even within the context of fitness, by harnessing the power of computer vision and deep learning. The application, by using the Mediapipe library for pose estimation and OpenCV for video processing, was able to accurately estimate joint position, perform certain angle calculations between joints, and thus give useful tips to users for potential improvement in their workout.

Such feedback mechanisms, including repetition counting, form evaluation, and speed assessment, would ideally guide the users through proper alignment and speed to minimize the opportunity for injury while maximizing the effectiveness of their exercise routine. Visual cues, numerical feedback, and color-coded alerts make the interaction intuitive and informative for the user.

Above all, the AI Trainer project demonstrated not only the technical capabilities of computer vision and machine learning but also how technology can work for health, fitness, and wellness. As technology advances further, such applications might transform how people stay fit, making their journey with fitness more accessible and enjoyable for users of different levels and backgrounds.

References

1. https://www.youtube.com/watch?v=H7cGq0xIHbc
2. https://www.youtube.com/watch?v=PGsAsuwBdw0
3. https://medium.com/ai-techsystems/image-detection-recognition-and-image-classification-with-machine-learning-92226ea5f595#:~:text=Image%20or%20Object%20Detection%20is,the%20difference%20is%20rather%20clear.
4. https://viso.ai/computer-vision/image-recognition/
5. https://www.freecodecamp.org/news/how-to-detect-objects-in-images-using-yolov8/