AI ASSINMENT

Project Title: Yoga Pose Detection and Alignment Feedback

1.Objective:

The goal of this project is to create an AI-powered tool that helps yoga practitioners by detecting their poses in real-time using computer vision. The tool also provides feedback on the alignment of key body joints (e.g., shoulder, elbow, wrist) to ensure proper form, which is essential for safe and effective practice.

2. Approach

Overview of Approach:

This project combines pose detection and pose correction feedback by detecting key body landmarks in real-time using MediaPipe Pose. These landmarks are then used to calculate angles between joints and compare them to predefined thresholds for alignment. Feedback is provided on the video stream, guiding users to correct any misalignment.

Step-by-Step Process:

1. Pose Detection: The MediaPipe Pose model is used to extract key body landmarks from video frames. It detects 33 key points (e.g., shoulders, elbows, wrists, knees) on the human body.
2. Angle Calculation: The angles between key joints (such as the elbow, shoulder, and wrist) are calculated to detect misalignment.
3. Feedback Mechanism: Based on the calculated angles, feedback is provided to the user (e.g., "Straighten your arm" or "Elbow too straight!").
4. Real-Time Feedback: The feedback is displayed in real-time overlaid on the video stream, helping the user adjust their form.

3. Data Preprocessing

Since this is a **pose detection** task rather than traditional image classification or object detection, the main data preprocessing tasks involve:

- Frame Extraction: Extract frames from the uploaded video or webcam stream.
- Landmark Extraction: MediaPipe Pose provides real-time landmarks, which represent the key points on the body. These landmarks are normalised between 0 and 1 for consistency across different video resolutions.
- Angle Calculation: The key points are used to calculate angles (e.g., shoulder-elbow-wrist for elbow alignment).

Preprocessing Steps:

1. Video Frame Extraction: The video is read frame by frame using OpenCV's `cv2.VideoCapture()`. Each frame is processed independently for pose detection.
2. Pose Detection: MediaPipe is used to extract the key landmarks for each frame.

3. Key point Normalisation: Coordinates of the keypoints are normalized to the range [0, 1] to handle different video resolutions.
4. Angle Calculation: Angles between key joints (e.g., elbow angle) are calculated using a trigonometric formula based on the positions of the shoulder, elbow, and wrist.

4. Model Architecture

In this project, MediaPipe Pose is used as the core model for pose detection. MediaPipe Pose is a pre-trained model that uses a lightweight architecture to detect human poses in real-time.

MediaPipe Pose Model:

- Input: RGB image (frame from the video).
- Output: 33 key points (landmarks) corresponding to various body parts.
- Architecture: MediaPipe Pose uses a machine learning-based solution that outputs a set of 2D coordinates for each of the detected landmarks.

Pose Correction (Angle Calculation) Logic:

For pose correction, angles between key points are calculated:

- Example: Elbow angle is calculated between the shoulder, elbow, and wrist joints using the `atan2` function to calculate the angle between two vectors.
- Thresholds: Defined thresholds for the elbow angle to determine if it needs adjustment (e.g., elbow angle < 70 degrees means "Straighten your arm!").

5. Results

Model Performance:

- Pose Detection Accuracy: MediaPipe Pose is known for its high accuracy in detecting body landmarks in real-time, with an average error of around **3-5 pixels** for 2D pose estimation.
- Pose Correction Accuracy: The angle calculation method is simple but effective. The alignment feedback is based on angles between joints, and the feedback is shown in real-time.

Real-Time Feedback:

The system works in real-time, providing immediate feedback to the user on their pose. The video is processed frame by frame, and the results are shown with the detected landmarks and feedback overlayed.

Visual Example:

- Before Correction: The video shows the user performing a yoga pose with feedback indicating "Elbow too bent."
- After Correction: The user adjusts their pose, and the feedback changes to "Good elbow position!"

You can include screenshots or video stills here to demonstrate how the feedback looks during the pose correction process.

6. Next Steps

1. **Expand Pose Recognition**:
   - Current Limitation: The system provides feedback based only on the elbow angle. The next step would be to include more poses and calculate additional joint angles (e.g., knee angle, wrist angle) for a more comprehensive feedback system.
   - Pose Classification: Train a classification model to identify different yoga poses (e.g., Downward Dog, Tree Pose) from the detected keypoints and provide targeted feedback.
2. **Real-Time Integration**:
   - Web Application: Integrate this solution into a web application (using **Flask** or **FastAPI**) to allow users to upload videos and get feedback in real-time.
   - Mobile Application: Convert the solution to work with **TensorFlow Lite** or use MediaPipe's mobile version for real-time pose detection and feedback on mobile devices.
3. **Model Optimization**:
   - Improve Accuracy: While MediaPipe provides good accuracy, further fine-tuning can be done using custom datasets of yoga poses to improve detection in various conditions (e.g., different lighting, angle of view).
4. **User Interface**:
   - Streamlined Feedback: Implement a user interface with clearer feedback (e.g., "Good Job!" or "Try again") and instructions.
   - Progress Tracking: Allow users to track their improvement over time by storing and analyzing their pose alignment score
   -
5. **Integration with Other Wearables**:
   - Integrate the pose detection system with fitness trackers or smartwatches to provide a holistic wellness experience by combining pose feedback with heart rate, calories burned, and other data.

7. Dependencies

The project relies on the following libraries:

- MediaPipe: For real-time pose detection.
- OpenCV: For video frame processing and display.
- NumPy: For mathematical operations such as angle calculation.
- Streamlit (optional): For web interface creation.

Requirements.txt:

Txt

```
opencv-python
mediapipe
numpystreamlit
```

8. Conclusion

This project demonstrates the use of computer vision for real-time feedback on yoga poses. By leveraging MediaPipe Pose and angle calculations, we can guide users in adjusting their form, helping them perform yoga poses safely and effectively. Future improvements include expanding the feedback system to cover more poses and integrating with other fitness data.