Yoga Pose Correction

Project submitted for the partial fulfillment of the requirements for the course

CSE 455 & CSE 455L: Artificial Intelligence

Offered by the

Department Computer Science and Engineering

School of Engineering and Sciences

Submitted by

1) Revanth Vangapandu, AP22110010109

2) Bandhavi Kanyadhara, AP22110010079

3) Swathi Chowdary Nagalla, AP22110010084

4) Vaishnavi Kolasani, AP22110010126

5) Rahul Chandra, AP22110010052

[Group Number: 09]

SRM University–AP

Neerukonda, Mangalagiri, Guntur

Andhra Pradesh – 522 240

[April 2025]

Contents:

1. Introduction

Over the last few years, the use of computer vision and artificial intelligence in fitness and well-being has seen remarkable momentum. Yoga poses detection and classification using Media Pipe, OpenCV, and machine learning is the focus of the present project. The main target is the creation of a system that can identify various yoga poses correctly from photos that can be further applied to real-time applications in feedback and correction.

The project starts with preprocessing a yoga dataset having 5 labeled yoga poses including Tree, Down Dog, Plank, Goddess, and Warrior II which is downloaded from Kaggle. The system gets 3D body landmarks (key points) for each image using Media Pipe Pose. The key points are used as input features to train a machine learning model. The features obtained are retained in a structured CSV file for model creation.

For the classification, a Random Forest Classifier is trained to output the right yoga pose given the landmark data. The system is tested for prediction accuracy and performance measures such as precision, recall, and F1-score. Using a good-enough model, this solution promises an application for pose recognition systems to support home fitness training, rehabilitation tracking, or support for yoga teaching.

Purpose:

The primary objective is to automatically recognize and classify yoga poses through human body KeyPoint extraction and machine learning, enabling users to practice yoga correctly with real-time guidance. At the current time when digital fitness and remote studies are increasingly prevalent, such AI-powered posture recognition systems have the potential to provide efficient and convenient solutions for self-practice.

The project leverages Media Pipe Pose, a robust library for human pose estimation, to find 3D key landmarks from every image in the yoga dataset. These key points represent important joints and body positions such as shoulders, hips, knees, and ankles. These landmarks are employed as features to train a Random Forest Classifier to identify various yoga poses based on their geometric arrangement.

2. Background

In recent years, Artificial Intelligence (AI) and computer vision has profoundly impacted the health and fitness space. With increasing usage of smart devices, wearables, and digital fitness applications, there is a definite need for smart systems that can track human movement, evaluate exercise quality, and give instant feedback.

Yoga, the ancient Indian art of enhancing physical, mental, and spiritual health, is now popular across the globe. Yet, to avoid injury and to acquire optimal benefits, proper alignment and form are essential in yoga. Without certified trainers, many practice incorrectly without realizing it. Thus, there is a demand for automatic pose correction technology that could assist practitioners in their daily practice particularly in home practice.

Why It Is Relevant

Imagine this scenario:

Someone starts doing yoga in the home with a YouTube video. They practice along, but their Down Dog pose is a bit off-center, shoulders are hunched over, and their spine is not neutral. Without feedback, this could result in shoulder strain or back pain.

Now, imagine a system that takes a picture of them with a webcam, compares their pose to a reference pose, and provides feedback in real-time such as:

"Straighten your spine" or "Push your hips higher."

This project is an attempt to make that vision a reality—by employing MediaPipe Pose to extract important landmarks/KeyPoint's and machine learning to classify and subsequently evaluate yoga poses. Such systems are especially valuable in:

- **Home workouts**
- **Online fitness platforms**
- **Elderly care fitness guidance**

Possible Ways to Solve the Problem

1. Rule-Based Pose Comparison

- Using angles between joints (like elbow, shoulder, knee) to compare against an ideal pose.
- Example: Comparing joint angles using cosine similarity or Euclidean distance.
- **Limitation**: Requires manual angle thresholding and lacks generalization.

2. Model Used: Random Forest Classifier

- Used a **Random Forest classifier** to identify yoga poses based on pose keypoints extracted using MediaPipe.
- Random Forest is a traditional machine learning model that operates by building an ensemble of decision trees and taking a majority vote for classification.

3. KeyPoint Extraction + ML Classification

- Extract body landmarks using Media Pipe or Open Pose.
- Flatten the key points (x, y, z) and classify poses using models like:
    - Random Forest
    - SVM
    - K-Nearest Neighbors (KNN)
- **Pros**: Efficient, less data-hungry, interpretable.

4. 3D Pose Estimation and Skeletal Tracking

- An advanced extension to 2D KeyPoint detection is the use of 3D pose estimation, which enables the capture of depth information and full skeletal motion using multiple RGB cameras or depth sensors.

3. Proposed Approach

Algorithm: Yoga Pose Detection and Classification

Input: Yoga pose image (captured from a real time web cam with voice recognition “I am ready” or it automatically captures in 2 minutes)
Output: Predicted yoga pose class (Tree, Plank, Warrior II, Goddess, Down dog with correction using key points)

Step 1: Import necessary libraries (MediaPipe, OpenCV, pandas, sklearn, etc.)

Step 2: Load and pre-process yoga pose images

Step 3: For each image:

- Apply MediaPipe Pose to extract 33 body keypoints
- Store each landmark's x, y, z coordinates and visibility

Step 4: Convert the keypoint data into a structured CSV format:

- Each row = 1 image, Each column = 1 coordinate value
- Add a label column for the actual pose class

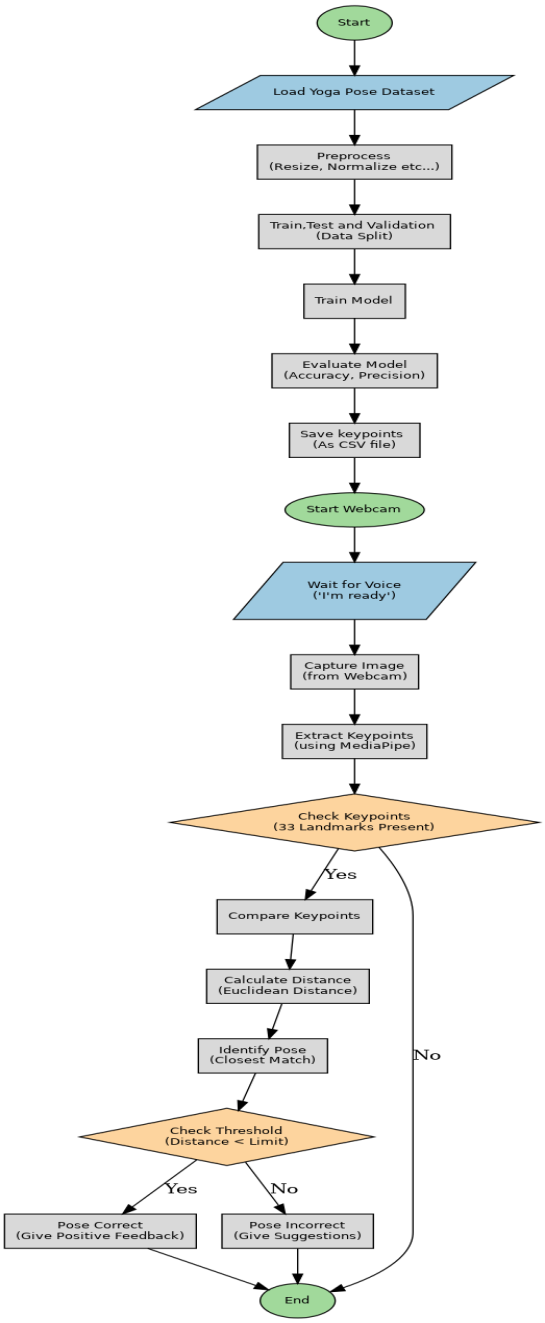
Step 5: Split data into training and testing sets

Step 6: Train a Random Forest Classifier using the training set

Step 7: Evaluate the model on the test set (accuracy, precision, recall)

Step 8: Input a new image and predict its pose using the trained model. Correct the Pose by giving the feedback.

Flow Chart



4. Results & Discussion

```
import base64
import io
import time
import cv2
import numpy as np
import pandas as pd
import mediapipe as mp
from PIL import Image
from IPython.display import Javascript, display, Image as IPImage
from google.colab import output
from pydub import AudioSegment
import speech_recognition as sr

# --- JavaScript to Start Webcam ---
display(Javascript("""
async function startWebcam() {
    const div = document.createElement('div');
    const video = document.createElement('video');
    video.style.width = '500px';
    video.setAttribute('autoplay', '');
    video.setAttribute('playsinline', '');
    const stream = await navigator.mediaDevices.getUserMedia({video: true});
    video.srcObject = stream;
    div.appendChild(video);
    document.body.appendChild(div);

    window.takePhoto = async function() {
        const canvas = document.createElement('canvas');
        canvas.width = video.videoWidth;
        canvas.height = video.videoHeight;
        canvas.getContext('2d').drawImage(video, 0, 0);
        stream.getTracks().forEach(track => track.stop());
        div.remove();
        return canvas.toDataURL('image/jpeg');
    }
}
startWebcam();
"""))

# --- JavaScript to Record Audio ---
```

```
RECORD = """
const sleep = time => new Promise(resolve => setTimeout(resolve, time));
const b2text = blob => new Promise(resolve => {
  const reader = new FileReader();
  reader.onloadend = () => resolve(reader.result);
  reader.readAsDataURL(blob);
});
async function recordAudio() {
  const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
  const mediaRecorder = new MediaRecorder(stream);
  const audioChunks = [];

  mediaRecorder.ondataavailable = event => {
    audioChunks.push(event.data);
  };

  mediaRecorder.start();
  await sleep(6000);
  mediaRecorder.stop();

  await new Promise(resolve => mediaRecorder.onstop = resolve);
  const audioBlob = new Blob(audioChunks);
  const audioBase64 = await b2text(audioBlob);
  stream.getTracks().forEach(track => track.stop());
  return audioBase64;
}
recordAudio();
"""

def record_and_transcribe():
    audio_base64 = output.eval_js(RECORD)
    audio_data = base64.b64decode(audio_base64.split(',')[1])
    with open("user_audio.webm", "wb") as f:
        f.write(audio_data)

    audio = AudioSegment.from_file("user_audio.webm")
    audio = audio.set_frame_rate(16000).set_channels(1)
    audio.export("converted_audio.wav", format="wav")

    recognizer = sr.Recognizer()
    with sr.AudioFile("converted_audio.wav") as source:
        audio_data = recognizer.record(source)
        try:
```

```
            return recognizer.recognize_google(audio_data).lower()
        except:
            return ""

POSE_FEEDBACK = {
    "tree": {
        "keywords": ["hip", "knee", "foot", "arm", "shoulder", "balance"],
        "tips": [
            "Place sole of foot on inner thigh or calf, not knee.",
            "Keep hips squared and facing forward.",
            "Engage core muscles to maintain balance.",
            "Bring palms together at chest or overhead.",
            "Relax shoulders away from ears.",
            "Keep standing leg straight but not locked."
        ]
    },
    "downdog": {
        "keywords": ["back", "heel", "arm", "hip", "shoulder", "head"],
        "tips": [
            "Spread fingers wide and press palms firmly.",
            "Straighten arms without locking elbows.",
            "Lift hips high to create an inverted V shape.",
            "Lengthen spine and avoid rounding back.",
            "Reach heels toward the ground.",
            "Keep head between arms in line with spine."
        ]
    },
    "plank": {
        "keywords": ["core", "hip", "shoulder", "arm", "neck", "leg"],
        "tips": [
            "Keep hands under shoulders.",
            "Maintain a straight line from head to heels.",
            "Engage your core to prevent sagging.",
            "Avoid lifting hips too high or letting them drop.",
            "Keep neck neutral, gaze slightly forward.",
            "Tighten leg muscles for stability."
        ]
    },
    "goddess": {
        "keywords": ["knee", "hip", "foot", "arm", "core", "torso"],
        "tips": [
            "Turn feet out to 45 degrees.",
            "Bend knees directly over ankles.",
            "Sink hips low as if sitting on a chair.",
```

```
            "Arms at 90 degrees, palms forward.",
            "Engage core to support lower back.",
            "Keep torso upright and gaze forward."
        ]
    },
    "warrior2": {
        "keywords": ["knee", "hip", "shoulder", "arm", "leg", "foot"],
        "tips": [
            "Bend front knee over ankle.",
            "Straighten back leg and ground heel.",
            "Open hips to the side.",
            "Extend arms parallel to the ground.",
            "Keep shoulders over hips.",
            "Engage core for balance and stability."
        ]
    }
}
def select_relevant_tips(pose_name, keypoints):
    selected = []
    if pose_name in POSE_FEEDBACK:
        tips = POSE_FEEDBACK[pose_name]["tips"]
        issues = np.random.choice(len(tips), size=3, replace=False)
        selected = [tips[i] for i in issues]
    return selected

def match_pose(pose_keypoints, csv_path="yoga_keypoints.csv"):
    df = pd.read_csv(csv_path)
    keypoint_cols = [col for col in df.columns if col != 'label']
    for col in keypoint_cols:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    sample_np = np.array(pose_keypoints).reshape(1, -1)
    db_np = df[keypoint_cols].values
    distances = np.linalg.norm(db_np - sample_np, axis=1)
    df["distance"] = distances

    min_distance = df["distance"].min()
    matched_pose = df.loc[df["distance"].idxmin()]["label"]
    return matched_pose, min_distance

# --- Main Execution ---
print("Listening for 'I'm ready' (2-minute timeout)...")
start_time = time.time()
ready_detected = False
```

```
for _ in range(20):
    transcript = record_and_transcribe()
    print(f"You said: {transcript}")
    if "i'm ready" in transcript or "im ready" in transcript or "i am ready" in
transcript:
        ready_detected = True
        break
    if time.time() - start_time > 120:
        break

if not ready_detected:
    print("Trigger phrase not detected.")
else:
    print("Trigger phrase detected. Capturing image...")
    photo_data = output.eval_js("takePhoto()")
    img_bytes = base64.b64decode(photo_data.split(",")[1])
    img = Image.open(io.BytesIO(img_bytes)).convert("RGB")
    img.save("captured_image.jpg")
    display(img)

    mp_pose = mp.solutions.pose
    mp_drawing = mp.solutions.drawing_utils
    image = cv2.cvtColor(np.array(img), cv2.COLOR_RGB2BGR)
    image = cv2.resize(image, (640, 480))

    with         mp_pose.Pose(static_image_mode=True,         model_complexity=2,
enable_segmentation=False, min_detection_confidence=0.5) as pose:
        result = pose.process(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
        pose_keypoints = []

        if result.pose_landmarks:
            for lm in result.pose_landmarks.landmark:
                pose_keypoints.extend([lm.x,             lm.y,             lm.z])
            mp_drawing.draw_landmarks(image,               result.pose_landmarks,
mp_pose.POSE_CONNECTIONS)
        else:
            print("No person detected in the image.")
            pose_keypoints = []
    if len(pose_keypoints) == 99:
        try:
            matched_pose, distance = match_pose(pose_keypoints)
            score      =      max(0,      100      -      distance      *      10)
            if distance < 0.1:
```

```
                score = 100
                print(f"Detected Pose: {matched_pose.upper()} | Accuracy Score:
{score}/100 (Perfect Pose!)")
                suggestions = []  # No feedback for perfect poses
            else:
                print(f"Detected Pose: {matched_pose.upper()} | Accuracy Score:
{score:.2f}/100")
                suggestions = select_relevant_tips(matched_pose, pose_keypoints)
                print("Feedback based on your pose:")
                for tip in suggestions:
                    print("-", tip)
            display(IPImage(data=cv2.imencode('.jpg', image)[1].tobytes()))

        except Exception as e:
            print("Error during pose evaluation:", str(e))
    else:
        print(f"Expected 99 keypoints, but got {len(pose_keypoints)}.")
```

DOWNDOG POSE

Listening for 'I'm ready' (2-minute timeout)
You said: i am ready
Trigger phrase detected. Capturing image...

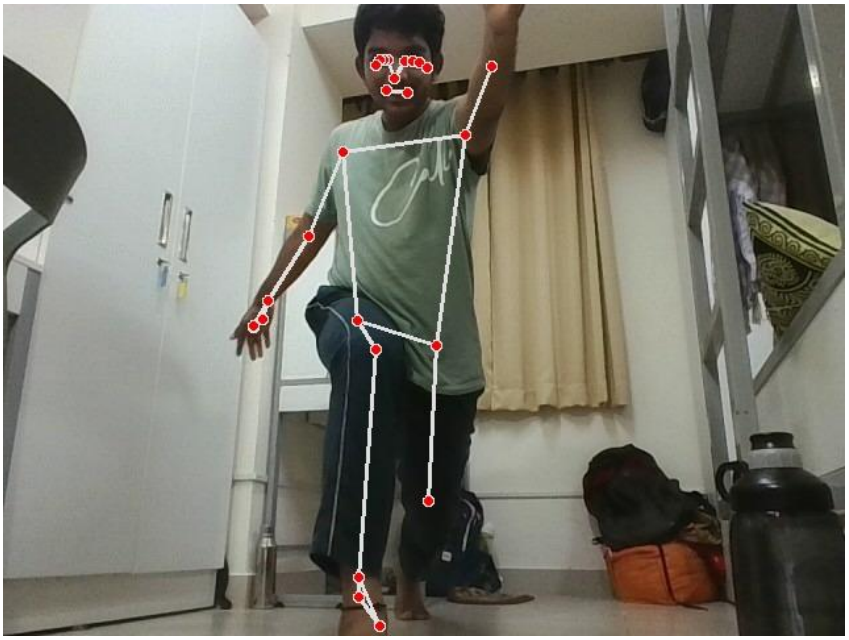
Detected Pose: DOWNDOG | Accuracy Score: 93.28/100
Feedback based on your pose:
-Reach heels toward the ground.
- Lift hips high to create an inverted V shape.
- Straighten arms without locking elbows.

WARRIOR II POSE

Listening for 'I'm ready' (2-minute timeout)
You said: i am ready

Trigger phrase detected. Capturing image...

Detected Pose: WARRIOR2 | Accuracy Score: 83.86/100
Feedback based on your pose:
- Engage core for balance and stability.
- Straighten back leg and ground heel.
- Extend arms parallel to the ground.



GODDESS POSE

Listening for 'I'm ready' (2-minute timeout)
You said: i am ready

Trigger phrase detected. Capturing image...

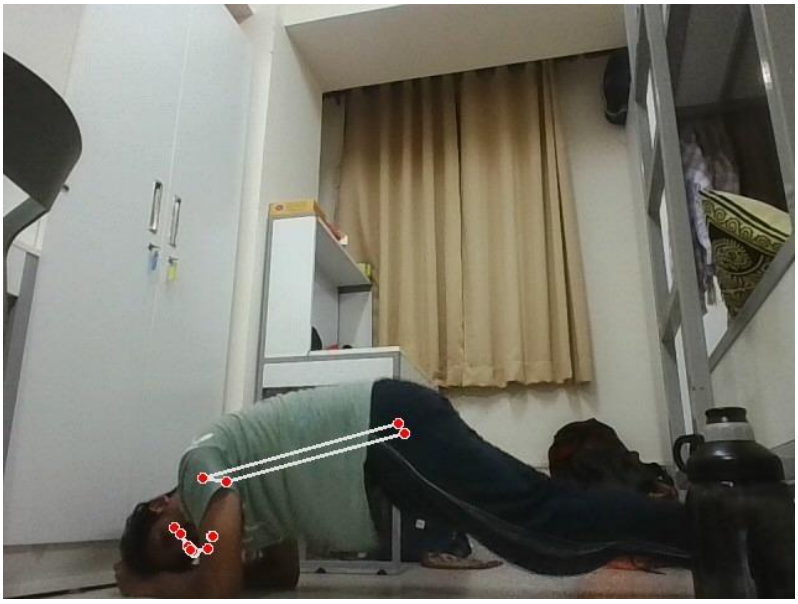
Detected Pose: GODDESS | Accuracy Score: 89.82/100
Feedback based on your pose:
- Keep torso upright and gaze forward.
- Arms at 90 degrees, palms forward.
- Turn feet out to 45 degrees.

PLANK POSE

Listening for 'I'm ready' (2-minute timeout)

You said: i am ready
Trigger phrase detected. Capturing image...

Detected Pose: PLANK | Accuracy Score: 87.76/100
Feedback based on your pose:
- Tighten leg muscles for stability.
- Keep neck neutral, gaze slightly forward.
- Avoid lifting hips too high or letting them drop.

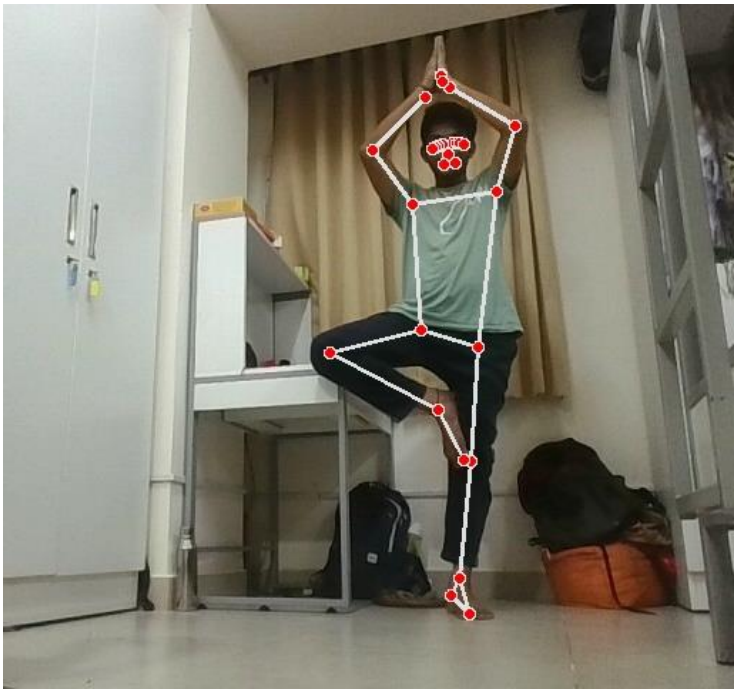


TREE POSE

Detected Pose: TREE | Accuracy Score: 89.03/100

Feedback based on your pose:

- Place your foot on your thigh or calf, not on your knee.
- Put your hands together at your chest or raise them above your head.
- Use your belly muscles to help balance.



5. Conclusion

This project demonstrates an effective approach to recognizing yoga poses using computer vision and machine learning. By leveraging Media Pipe Pose for landmark extraction and a Random Forest Classifier for pose classification, we created a system capable of identifying yoga postures with considerable accuracy based on human body key points.

The solution is lightweight, interpretable, and scalable making it suitable for real-time yoga assistance applications, especially for users who practice at home without instructor supervision. It bridges the gap between traditional yoga instruction and modern AI technology, contributing to the growing field of digital health and personalized fitness.

While the current Yoga Pose Corrector system demonstrates promising results, there are certain limitations that can be addressed in future improvements:

1. **Poor Image Quality**
   Low-resolution, blurry, dark, or overexposed images can significantly reduce the accuracy of KeyPoint detection. Enhancing preprocessing techniques or integrating image quality filters could help ensure better input quality for the model.
2. **Person is Partially Visible**
   When the subject appears too far from the camera or only partially in the frame, KeyPoint's may not be detected accurately or completely. This affects pose classification performance. Implementing automatic zoom adjustment or providing real-time framing guidance can mitigate this issue.
3. **Unusual Pose or Occlusion**
   Poses with occluded limbs (e.g., arms behind the back) or highly non-standard body positions can confuse the model, resulting in incorrect or missing KeyPoint's. Future work could include training the model on a more diverse dataset with occlusion handling techniques or using temporal smoothing in video streams.

6.References

Dataset Source

Niharika. (2020). *Yoga Poses Dataset*. Kaggle.

Available at: https://www.kaggle.com/datasets/niharika41298/yoga-poses-dataset