Human Pose Estimation using Machine Learning

A Project Report

submitted in partial fulfillment of the requirements

of

AICTE Internship on AI: Transformative Learning
with
TechSaksham – A joint CSR initiative of Microsoft & SAP

by

JEET BHARDWAJ

Jeet90008@gmail.com

Under the Guidance of

P.Raja

Master Trainer, Edunet Foundation

ACKNOWLEDGEMENT

I would like to take this opportunity to express my deep sense of gratitude to everyone who helped me directly or indirectly during this project work.

First of all, I want to thank my supervisor, [P.Raja sir], for being a great mentor and the best adviser I could ever have. His advice, encouragement and critics have been a source of innovative ideas and inspiration and it was the key behind the successful completion of this project. The confidence shown in me by him was the biggest source of inspiration for me. It has been a privilege working with him for the last few weeks. He always helped me during my project and in many other aspects related to the program. His talks and lessons not only helped in project work and other activities but also made me a more responsible professional.

ABSTRACT

The aim of this project was to create a web-based application capable of live pose detection by using **Mediapipe** and **Flask** as the core technologies. The main objective was to identify and match live body poses captured through a camera feed with pre-stored custom poses that were pre-defined.

The project involved leveraging **computer vision** and **machine learning** techniques to calculate joint angles from the live video input and compare them against a database of saved poses. This matching process required precise analysis of the skeletal structure and movement patterns in real-time, which was achieved through Mediapipe's pose estimation capabilities. Flask, on the other hand, acted as the backbone of the application, providing a seamless interface for users to interact with the system.

The results demonstrated accurate real-time detection and pose matching, making it a valuable foundation for various domains like fitness tracking, healthcare solutions, and sports analytics. For example, the application could potentially be used in workout monitoring to ensure correct posture, to help patients perform prescribed exercises correctly.

While the project successfully showcased a practical implementation of live pose detection, there were also limitations that left room for improvement. For instance, the system might not handle extremely complex poses or unusual lighting conditions as effectively. Future enhancements could include integrating more advanced features like dynamic pose tracking over sequences, personalized feedback mechanisms, or even combining AI models to improve the accuracy and robustness of the detection process.

Overall, this project was a significant step towards utilizing real-time pose detection in real-world applications and opens up exciting possibilities for further development in this area.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Problem Statement:

The project aims to solve the challenge of identifying and matching human body poses in real time. This problem is significant in fields like fitness training, physiotherapy, and gaming, where accurate body movement analysis is critical. This project addresses these challenges by offering a system that can detect and compare body poses with high accuracy, making it easier to improve safety, performance, and user engagement in these areas.

1.2 Motivation:

This project was chosen to explore the intersection of computer vision and human-computer interaction. The potential applications in healthcare and sports motivated me in the development of this project, as it can be highly impactful in those fields.

1.3 Objective:

1. To implement real-time pose detection using Mediapipe.
2. To enable storage and matching of custom poses.
3. To provide a simple, user-friendly interface for live pose detection.

1.4 Scope of the Project:

The scope of this project involves creating a functional prototype capable of detecting and matching body poses in real time. The system is designed to focus on basic joint angle calculations for pose recognition and matching using Mediapipe and Flask.

While it provides an efficient solution for pose detection and storage, it does not incorporate advanced machine learning algorithms for pose classification, which limits its scalability for more complex use cases. The project is targeted primarily at developing a foundational system that can later be expanded for applications in fitness, sports, and healthcare, where real-time feedback on poses is essential.

Furthermore, the system's reliance on a web camera and lightweight computations ensures that it is accessible and can be implemented on standard devices without specialized hardware.

CHAPTER 2

Literature Survey

2.1 Literature or Previous work in this Domain

Over the years, numerous studies have investigated pose detection and recognition using various **computer vision** and **machine learning** techniques. These studies have laid the foundation for real-time pose estimation by focusing on the accurate detection of human skeletal structures from video feeds. Among the tools available, **Mediapipe** has emerged as one of the most efficient libraries for real-time pose estimation. Its lightweight design and optimized performance make it significantly faster and more accessible compared to traditional methods. Unlike many other frameworks, Mediapipe can run efficiently on consumer-grade hardware, such as laptops and smartphones, without requiring expensive GPUs.

Other notable frameworks, like **OpenPose** and similar deep learning models, have also been widely adopted for pose detection. However, while these systems offer high levels of accuracy and flexibility, they often come with the drawback of requiring substantial computational power. This makes them less suitable for applications where resources are limited or where real-time performance is critical.

2.2 Existing Models and Techniques

Current methods for pose detection frequently rely on complex machine learning algorithms that are computationally expensive. These systems often prioritize high accuracy and detailed analysis at the cost of speed and usability. Moreover, most applications are designed to recognize a set of predefined poses, making them less adaptable to scenarios where custom pose matching is needed.

Additionally, the technical complexity of these systems presents a significant barrier for non-technical users. Setting up and using such systems often requires specialized knowledge, which limits their usability in practical, everyday applications like fitness monitoring, physiotherapy, or gaming.

2.3 Gaps and Limitations in Existing Solutions

This project aims to bridge the gap by creating a **user-friendly interface** for live pose matching and storage. By leveraging the simplicity and efficiency of **Mediapipe** alongside the lightweight framework of **Flask**, the system provides a practical solution for real-time pose detection. Unlike computationally intensive methods, this system is designed to work seamlessly on standard consumer-grade hardware, eliminating the need for expensive setups or high-end computational resources.

The system is tailored to allow flexibility in defining and storing custom poses, making it more adaptable to diverse use cases. Its straightforward interface ensures that users, regardless of technical expertise, can leverage its capabilities for applications in fitness, healthcare, gaming, and beyond. By addressing these limitations, the project not only simplifies pose detection but also broadens its accessibility and potential impact.

CHAPTER 3

Proposed Methodology

3.1 System Design

The proposed system includes:

1. A camera interface for live video capture.
2. Pose detection using Mediapipe.
3. Custom pose storage in JSON format.
4. Real-time matching and feedback display.
5. A Flask-based web application for user interaction and customization.

.

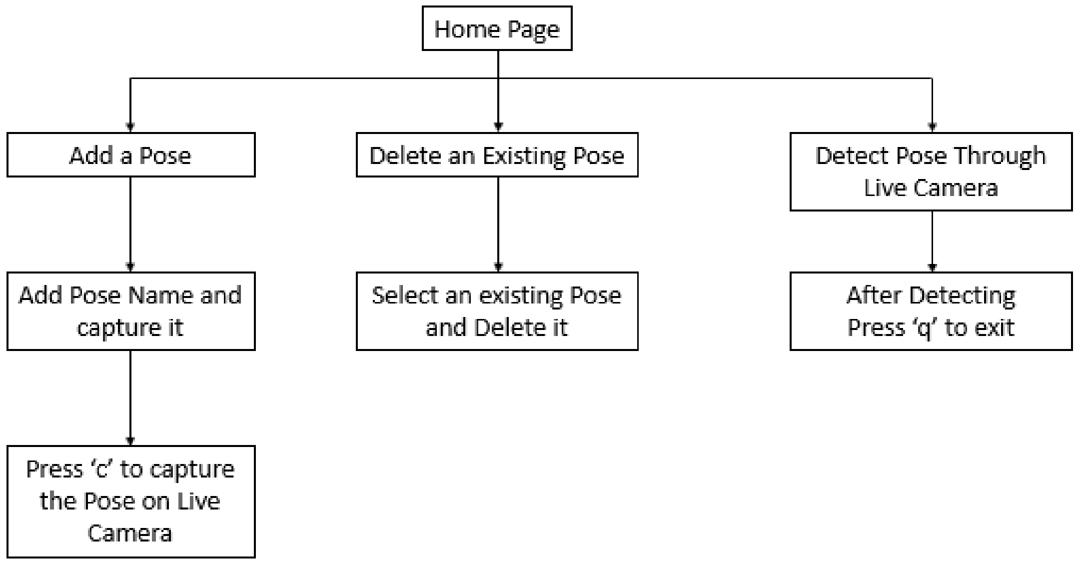

Fig.1

The system is divided into three main components:

1. Frontend: Provides a user interface for adding, deleting, and detecting poses.
2. Backend: Handles pose detection logic, custom pose storage, and matching algorithms.
3. Data Storage: Stores custom poses in a JSON file for easy retrieval and updates.

3.2 Requirement Specification

3.2.1 Hardware Requirements:

- Web Camera
- Laptop/Desktop with a minimum of 4GB RAM
- Standard CPU (Intel i3 or higher)

3.2.2 Software Requirements:

- Python 3.9+
- Flask
- Mediapipe
- OpenCV
- JSON library for data storage
- Math library for angle calculations

CHAPTER 4

Implementation and Result

4.1 Snap Shots of Result:

1. Home Page

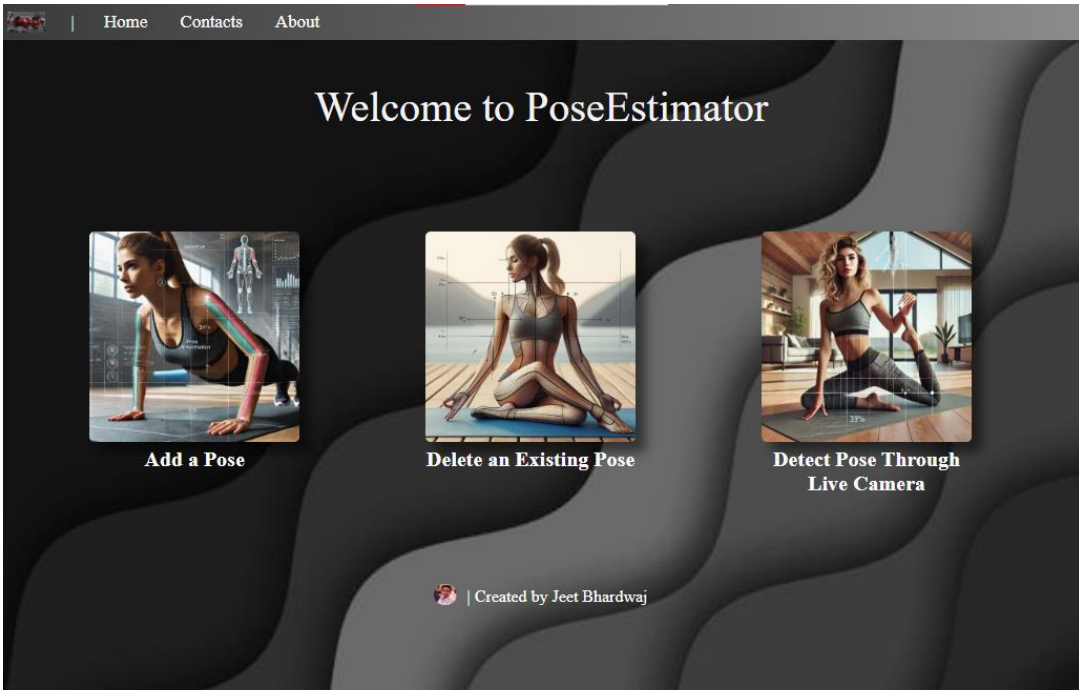

Fig.2

The home page serves as the central hub of the application. It provides three main options:

- **Add Pose:** Allows the user to add a new custom pose to the library.
- **Delete Pose:** Enables the user to remove an existing pose from the library.
- **Detect Pose:** Redirects the user to the live pose detection feature, where they can compare their live pose with the stored poses.

2. Adding a Pose

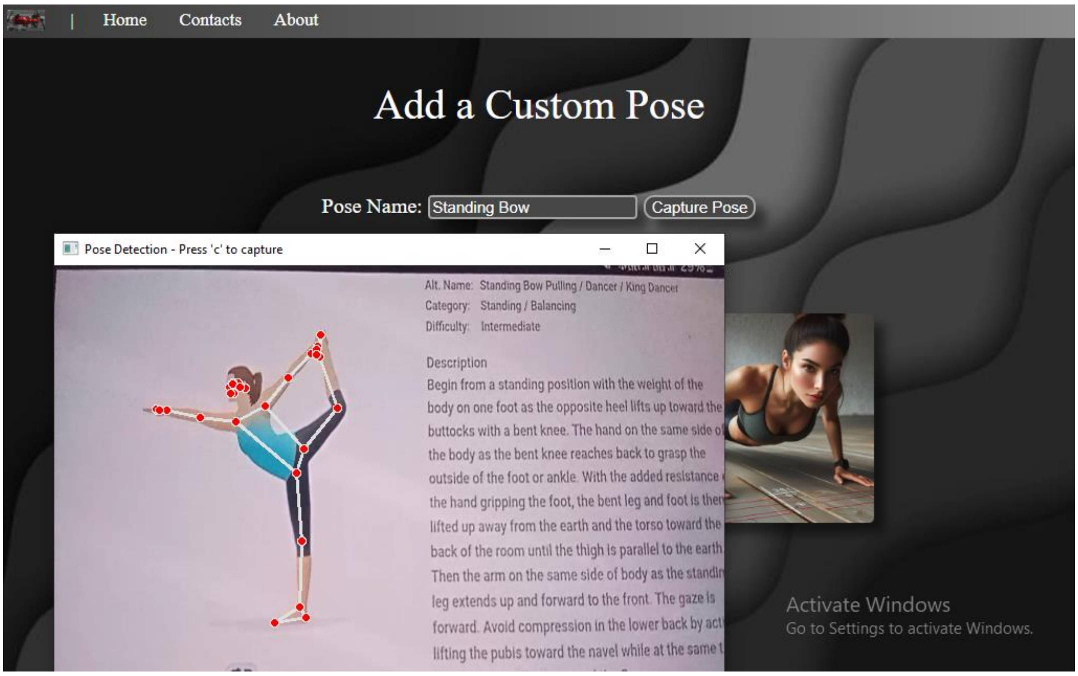

Fig.3

This snapshot captures the interface where users can add a new custom pose.

- Users are prompted to position themselves in front of the camera.
- Once the pose is captured, they can assign a name to it and save it in the library.
- The system records the joint angles and other pose-specific data in a JSON file, which will be used for future matching.

3. Detecting Pose through Live Camera

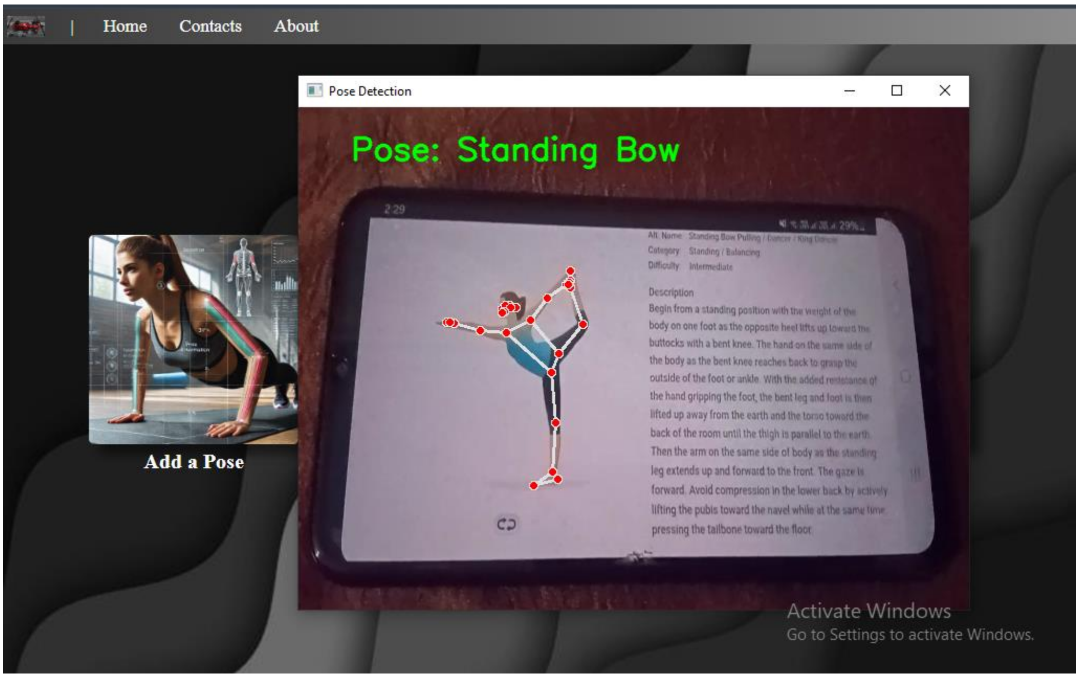

Fig.4

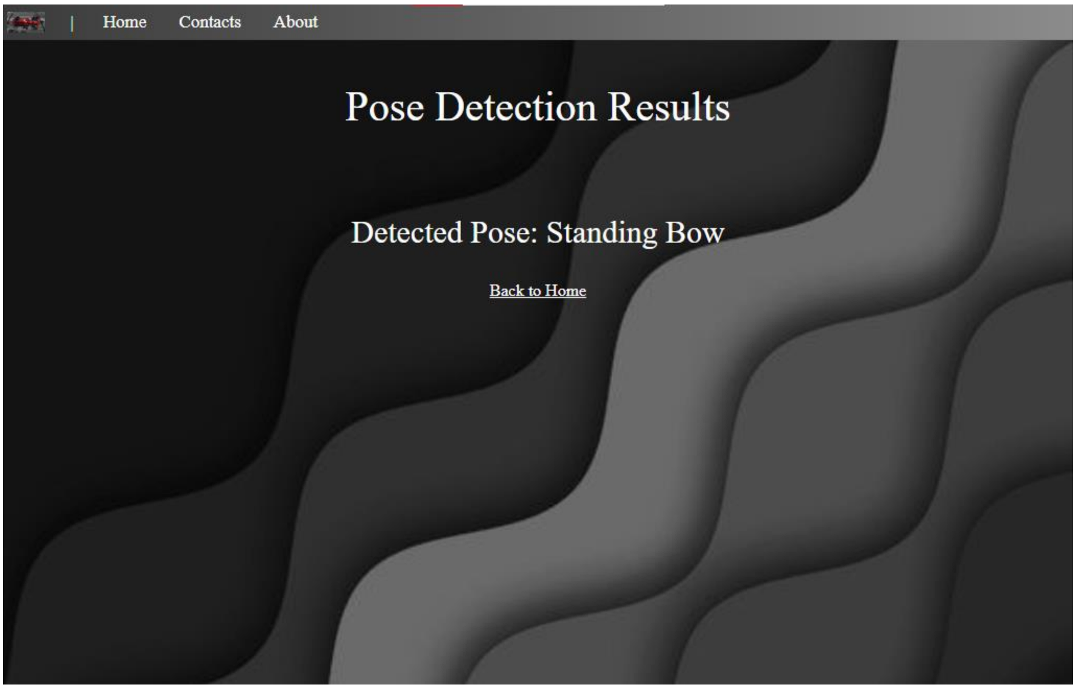

Fig.5

This snapshot illustrates the real-time pose detection feature:

- The camera captures the user's live video feed.

- The system detects the user's body pose using Mediapipe and calculates joint angles.
- It matches the live pose against the stored poses in the library and provides immediate feedback. For example, if a match is found, the system highlights the matched pose name and provides visual cues, such as marking matched joints in green.

4. Deleting a Pose

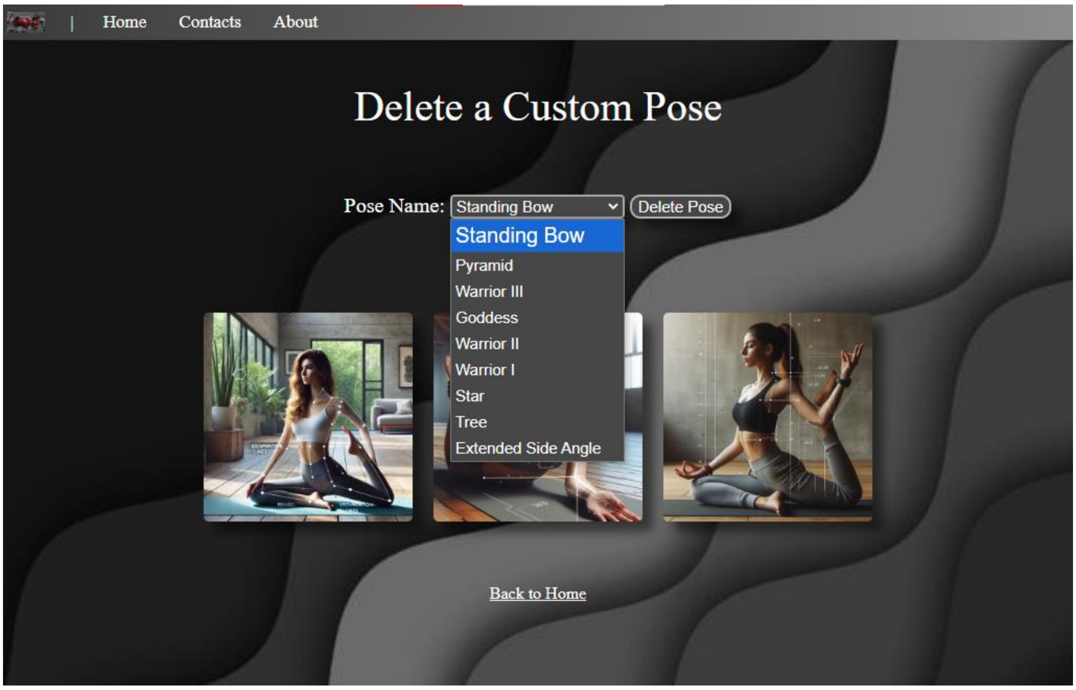

Fig.6

This snapshot shows the interface for managing stored poses.

- The user selects the pose they want to delete from a dropdown list or a displayed table of poses.
- Once confirmed, the pose is permanently removed from the library, and the JSON data is updated to reflect the changes.
- This feature ensures flexibility in maintaining the custom pose library.

4.2 GitHub Link for Code:

https://github.com/GitHub-JeetBhardwaj/PoseEstimator

CHAPTER 5

Discussion and Conclusion

5.1 Future Work:

Provide suggestions for improving the model or addressing any unresolved issues in future work. The current project lays a strong foundation for real-time pose detection but also opens up avenues for further development and enhancements.

- **Enhancing Pose Angle Detection**: One challenge that remains is the identification of poses with similar joint angles, particularly in cases where the angles of both shoulders, both knees, neck and hip are the same. Future versions could incorporate additional angles, critical joints, to improve pose differentiation. By introducing more angles into the system, it would be able to distinguish between similar poses with greater accuracy, ensuring more reliable pose classification in dynamic movements.

- **Advanced Pose Classification Using Deep Learning**: Future iterations could integrate deep learning models to classify and recognize more complex poses beyond simple matching. This would enable the system to handle dynamic movements and sequences, making it more versatile for applications like dance training or sports analysis.

- **Mobile Device Compatibility**: Enhancing the system interface to support mobile devices is a key goal. By optimizing the application for mobile platforms, users could access pose detection capabilities on the go, making it more accessible for fitness enthusiasts, trainers, and patients.

- **Cloud-Based Remote Pose Detection**: Developing a cloud-based version of the system would allow users to perform pose detection remotely. Trainers, therapists, or coaches could monitor and guide users in real-time, regardless of their location, broadening the system's reach and usability.

- **Motion Tracking and Real-Time Corrections**: Adding motion tracking capabilities to analyzes continuous movements and provide real-time corrections would further enhance the system's functionality. This feature would be especially beneficial for activities involving fluid motion, such as yoga, martial arts, or sports.

5.2 Conclusion:

This project successfully developed a lightweight, real-time pose detection system using Mediapipe and Flask. It demonstrates the potential for practical applications in fitness, healthcare, and other domains where pose analysis is essential.

The simplicity of the system ensures that it is accessible to users with minimal technical expertise, while its modular design allows for future expansion and customization. The project serves as a robust starting point for implementing more advanced pose detection features, such as dynamic motion analysis, cloud integration, and mobile compatibility.

The results highlight the effectiveness of this approach in creating a user-friendly and efficient solution. With further development, the system has the potential to revolutionize pose detection applications, improving safety, performance, and user engagement in diverse fields.

REFERENCES

[1] [1] Ming-Hsuan Yang, David J. Kriegman, Narendra Ahuja, “Detecting Faces in Images: A Survey”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Volume. 24, No. 1, 2002.

[2] [2] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields," CVPR, 2017.

[3] [3] Mediapipe Documentation, "Pose Module," Google Developers, 2023.