RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

Go, change the world®

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Project Report

On

PERSONAL YOGA-TUTOR

Submitted in partial fulfilment of the requirements for the V Semester

ARTIFICIAL NEURAL NETWORK AND DEEP LEARNING

AI253IA

By

1RV22AI055	**Shreyas Jain**
1RV23AI402	**K. Preethi**
1RV23AI404	**Roopa Iranna Bagalkoti**

Department of Artificial Intelligence and Machine Learning
RV College of Engineering®
Bengaluru – 560059

September 2024

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Bengaluru– 560059

CERTIFICATE

This is to certify that the project entitled "Personal yoga-Tutor" submitted in partial fulfillment of Artificial Neural Networks and Deep Learning (21AI53) of V Semester BE is a result of the bonafide work carried out by Shreyas Jain(1RV22AI055), K.Preethi(1RV23AI402), Roopa Iranna Bagalkoti(1RV23AI404) during the Academic year 2024-25

Faculty In charge

Date :

Head of the Department

Date :

RV COLLEGE OF ENGINEERING®

(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Bengaluru– 560059

DECLARATION

We, Shreyas Jain(1RV22AI055), K.Preethi(1RV23AI402) and Roopa Iranna Bagalkoti(1RV23AI404) students of Sixth Semester BE hereby declare that the Project titled **"Personal Yoga-Tutor "** has been carried out and completed successfully by us and is our original work.

Date of Submission: **Signature of the Student**

ACKNOWLEDGEMENT

We are profoundly grateful to our guide, **Dr. Somesh Nandi**, Assistant Professor, RV College of Engineering, for his wholehearted support, valuable suggestions, and invaluable advice throughout the duration of our project. His guidance and encouragement were instrumental not only in the successful completion of the project but also in the preparation of this report. We also extend our special thanks to **Dr. Anupama Kumar** for her invaluable insights, support, and constructive feedback, which significantly contributed to the improvement of our work.

We would like to express our sincere thanks to our Head of the Department, **Dr. Satish Babu**, for his constant encouragement and for fostering an environment of innovation and learning that greatly aided our progress.

We extend our heartfelt gratitude to our beloved Principal, **Dr. K. N. Subramanya**, for his unwavering appreciation and support for this Experiential Learning Project, which motivated us to give our best

Lastly, we take this opportunity to thank our family members and friends for their unconditional support and encouragement throughout the project. Their backup and motivation were crucial in helping us overcome challenges and successfully complete our work.

ABSTRACT

Yoga has gained worldwide recognition for its physical, mental, and therapeutic benefits. However, incorrect posture and alignment during practice can lead to inefficiencies or even injuries. This project introduces a **real-time Yoga Pose Detection System** using **deep learning and computer vision** techniques to assist yoga practitioners in performing poses accurately. The system employs **MediaPipe BlazePose** for **pose estimation**, extracting key body landmarks from video frames, which are then classified using a **K-Nearest Neighbours (KNN) model** to identify different yoga poses.

The primary objectives of this project include:

- **Real-time pose detection** and classification of common yoga postures.
- **Providing corrective feedback** on posture alignment and movement to prevent injuries.
- **Facilitating remote yoga instruction**, allowing users to practice without an in-person trainer.
- **Tracking user progress** over time to enhance learning outcomes.

The system architecture consists of **pose detection, feature extraction, classification, and feedback mechanisms**. The **model achieved a 99.55% accuracy** in classifying yoga poses, demonstrating its efficiency in real-world applications. Future enhancements include **multi-person detection, dataset expansion, wearable device integration, and augmented reality (AR)-based feedback** to further improve user experience.

This project aims to bridge the gap between **traditional yoga practice and AI-driven assistance**, making yoga more accessible, engaging, and precise for practitioners of all levels.

TABLE OF CONTENTS

Chapter 1 INTRODUCTION

1.1 Introduction to the topic

Project Description:

The project focuses on the development of a Yoga Pose Detection System using Deep Learning techniques to identify and track yoga poses. With the increasing global interest in yoga, this system aims to provide a solution that leverages computer vision and machine learning technologies to enhance the practice and teaching of yoga. It will use real-time image or video analysis to assess the alignment, posture, and movement of a person performing yoga poses, offering real-time feedback for self-correction, injury prevention, and progress tracking. By accurately detecting the various poses in yoga, the system can facilitate remote instruction, personalized yoga programs, and potentially aid in advancing scientific understanding of the benefits of yoga on physical and mental wellbeing.

Objectives of the Project:

The primary objectives of the Yoga Pose Detection system are as follows:

- Develop a machine learning model capable of detecting and classifying common yoga poses accurately.
- Analyze yoga practitioners' movements, providing real-time feedback on posture and alignment.
- Enable remote yoga instruction, allowing users to practice yoga without the need for an instructor physically present.
- Track users' progress by detecting improvements in pose accuracy over time.
- Prevent injuries by providing corrective feedback to ensure proper alignment during poses.
- Design a user-friendly interface for both beginners and advanced practitioners to interact with the system, allowing them to receive customized yoga routines based on their individual performance and progress.
- Contribute to scientific knowledge on the mind-body benefits of yoga, by analyzing and presenting the empirical data collected from yoga practitioners.

Theory and Concept Relevant to the Project:

The Yoga Pose Detection system combines the following key theories and concepts:

1. **Deep Learning & Convolutional Neural Networks (CNNs):** Deep learning, particularly CNNs, will be used for analyzing images and detecting features relevant to various yoga poses. The system will be trained on a dataset of yoga poses to recognize the key features that define each pose.

2. **Computer Vision:** Computer vision techniques will be employed to analyze images or videos captured of the yoga practitioner, detect the position of body joints, and match these positions with predefined yoga poses.
3. **Pose Estimation:** This involves the use of algorithms such as OpenPose or PoseNet that detect human body parts (e.g., head, shoulders, arms, legs, etc.) in images, helping to identify the posture and alignment of the practitioner.
4. **Human-Computer Interaction (HCI):** The interface will allow users to interact with the system and receive real-time feedback on their pose alignment.
5. **Mental & Physical Health Benefits of Yoga:** The project draws from research that highlights the benefits of yoga, such as improved physical flexibility, strength, and mental relaxation, and its potential in reducing stress and anxiety. The system will aim to reinforce these benefits through accurate pose detection and tailored feedback.
6. **Reinforcement Learning for Personalized Feedback:** Using reinforcement learning, the system will adapt to provide personalized feedback for users, rewarding progress and guiding them towards better posture and form.

1.2 Report Organization

The report is organized as follows:

1. **Introduction:** This section provides an overview of the project, the problem statement, and the importance of yoga pose detection. It introduces the goals and objectives of the project and the potential impact on yoga practice.
2. **Literature Review:** This section reviews previous research and technologies used for yoga pose detection, including computer vision, machine learning techniques, and the benefits of yoga for mental and physical health.
3. **System Design and Architecture:** This section details the technical architecture of the system, including the choice of algorithms, tools, and frameworks used (e.g., TensorFlow, OpenCV). It describes the pose detection process, data preprocessing, model training, and validation.
4. **Implementation:** This section describes the actual development process of the Yoga Pose Detection system. It outlines the dataset preparation, feature extraction, and training of the deep learning model. The implementation challenges and solutions are also discussed.
5. **Results and Evaluation:** This section presents the evaluation metrics used to assess the performance of the model, such as accuracy, precision, recall, and F1 score. It includes test results and comparisons with existing methods.
6. **Discussion:** This section explores the findings from the results, such as the potential applications of the system, the limitations, and areas for future improvement.
7. **Conclusion:** This section summarizes the findings of the project, its impact on yoga practice, and potential for future enhancements.
8. **References:** This section includes a list of all scholarly references, articles, books, and research papers used throughout the project.

Chapter 2 LITERATURE REVIEW

2.1 Literature survey:

1. **Santosh Kumar Yadav (2019)** - The study used **OpenPose** to identify key points in yoga stances. It utilized **CNN** for pattern recognition in individual frames and **LSTM** to memorize pose patterns. The combination of CNN and LSTM improved the accuracy of pose recognition and classification. This research highlights the integration of deep learning with yoga pose estimation systems.

2. **Cao et al. (2017)** - Introduced the **OpenPose** framework, which is widely used for pose estimation. OpenPose detects human body key points and facilitates the analysis of yoga poses. The framework significantly improves the accuracy of human pose recognition by detecting critical points, a foundational model for various yoga pose detection systems.

3. **Jothika Sunney (2022)** - Explored the use of the **MediaPipe BlazePose model** in real-time yoga/fitness applications. MediaPipe outperformed **OpenPose** by factors of 25-75 and provided 3D keypoint extraction without the need for depth sensors. However, the system has limitations in detecting multiple persons simultaneously. The paper's novelty lies in combining **BlazePose** and **XGBoost** for real-time yoga pose detection.

4. **BhattacharyaS (2022)** - Utilized **CNN** to detect 15 different yoga postures. A dataset of 10,000 images was augmented to train the model, achieving an accuracy of 91.7%. This study demonstrates the potential of CNNs in detecting complex yoga poses with high accuracy and provides a dataset for pose recognition.

5. **Sun et al. (2020)** - Implemented a yoga pose detection system based on **PoseNet** and **CNN** for real-time feedback during yoga practice. The system offered a significant advancement in pose detection accuracy and reduced the need for manual corrections during practice.

6. **Zhao et al. (2021)** - Proposed a **DeepPose** system combining deep neural networks for accurate pose estimation in yoga applications. The research suggests that **DeepPose** can be applied to various yoga poses for improving training and monitoring accuracy.

7. **Zhang et al. (2019)** - Developed a pose detection model for yoga that uses **Convolutional Pose Machines** (CPMs). The model shows promise in improving real-time feedback for yoga practitioners by tracking the alignment and angles of joints in dynamic poses.

8. **Tiwari et al. (2020)** - Investigated the use of **Recurrent Neural Networks** (RNNs) for modeling dynamic yoga poses. This research indicated the potential of RNNs to handle sequential data, which is crucial for continuous yoga movements.

9. **Chen et al. (2018)** - Introduced a hybrid deep learning model combining **CNN** and **RNN** for yoga pose detection. The system effectively categorized different yoga poses by integrating static and dynamic posture analysis, achieving high precision in classification.

10. **Ranjan et al. (2021)** - Focused on **Multi-task Learning** for pose estimation, enabling improved performance in multi-pose detection and real-time correction during yoga practice. The study suggests integrating multiple pose detection systems for better generalization.

11. **Liu et al. (2021)** - Studied the integration of **pose estimation models** with **interactive applications** to offer real-time yoga pose analysis. The research demonstrated how user interaction could enhance the accuracy and user experience of yoga training applications.

12. **Roy et al. (2020)** - Investigated **transfer learning** in yoga pose detection, using pre-trained models on large datasets to improve accuracy and performance for yoga applications. The study shows that **transfer learning** could significantly reduce the data requirements for pose recognition.

13. **Kumar et al. (2022)** - Proposed a system combining **pose estimation** with **motion tracking** for yoga practitioners. This hybrid approach aimed to provide detailed feedback on posture alignment and movement during practice.

14. **Almeida et al. (2019)** - Analyzed the use of **3D convolutional neural networks** for yoga pose recognition. Their findings support the idea that 3D CNNs can capture complex movements and alignments in yoga, which are essential for accurate pose detection.

15. **Singh et al. (2021)** - Investigated the use of **data augmentation techniques** to generate synthetic data for training deep learning models in yoga pose detection. The research demonstrated how augmenting the dataset can improve model performance, especially when training on smaller datasets.

16. **Nakamura et al. (2018)** - Proposed a **multi-view yoga pose detection system** that used multiple cameras to capture the yoga practitioner from different angles. This research shows the potential for improving accuracy by collecting 3D data from multiple perspectives.

17. **Gupta et al. (2020)** - Developed a **real-time yoga feedback system** based on pose recognition algorithms. The system tracks the accuracy of a practitioner's poses in real-time and provides corrective feedback, thus improving the practice experience.

18. **Patel et al. (2021)** - Introduced the use of **Edge Computing** for real-time yoga pose recognition, reducing latency and processing times in remote yoga classes. The system processes data directly on the edge device, enhancing the efficiency of yoga pose feedback.

19. **Srinivasan et al. (2020)** - Proposed a system combining **pose estimation** and **action recognition** for yoga applications. Their model is able to recognize dynamic yoga poses and evaluate practitioners' progress over time, offering personalized recommendations.

20. **Yang et al. (2021)** - Examined the application of **augmented reality** (AR) in yoga pose detection. The study suggested that combining AR with real-time yoga pose detection could enhance the teaching and learning experience by providing immediate visual feedback.

Unresolved Issues and Emerging Opportunities:

While significant advancements have been made in yoga pose detection, several issues remain unresolved:

1. **Multi-Person Detection**: Most systems still face challenges in detecting and tracking multiple individuals in a single frame. This is particularly important for group yoga sessions or for practitioners in busy environments.

2. **Real-Time Feedback**: Providing accurate, real-time feedback for dynamic poses remains a challenge. The system needs to evaluate continuous movements with high precision to guide practitioners effectively.

3. **Dataset Limitations**: The lack of large, diverse datasets with a wide range of yoga poses continues to hinder the development of robust pose detection models. More datasets and varied yoga poses are needed to improve model accuracy and generalizability.

4. **Improved Accuracy and Precision**: While models like **MediaPipe BlazePose** have shown promising results, integrating multiple machine learning algorithms to enhance accuracy and performance for yoga poses still presents a challenge.

Emerging opportunities in the field include the use of **multi-modal sensors**, **3D pose estimation**, and **deep reinforcement learning** to improve accuracy, real-time feedback, and personalized yoga experiences.

Conclusion of the Literature Survey:

The literature survey shows that significant progress has been made in the field of yoga pose detection, with various models such as **OpenPose**, **BlazePose**, and **CNNs** being applied to this domain. However, challenges such as multi-person detection, real-time feedback accuracy, and dataset limitations remain. This motivates the need for a comprehensive solution that combines **machine learning** and **computer vision** techniques to provide accurate yoga pose recognition and real-time feedback. The motivation for this project arises from the opportunity to advance yoga practice by improving pose detection accuracy, enabling personalized instruction, and offering real-time guidance to practitioners.

Objectives of the Project:

1. **Develop a deep learning-based yoga pose detection model** that accurately classifies different yoga poses in real-time.
2. **Integrate computer vision and machine learning techniques** to provide real-time feedback on posture alignment and movement.
3. **Design an intuitive user interface** for yoga practitioners to receive corrective feedback and track their progress over time.
4. **Investigate the potential for multi-person detection** and improve model robustness to handle dynamic environments.

2.2 Existing and Proposed System

Problem Statement:

Yoga pose detection remains a complex challenge, particularly in real-time applications, due to the need for accurate posture recognition and alignment feedback. Existing systems have limitations such as detecting only a single individual, lack of accurate real-time feedback, and insufficient datasets for training models with diverse poses. Furthermore, the scalability of these systems for multi-person yoga sessions and dynamic poses is often constrained by the technology used.

Scope of the Project:

The scope of this project is to design and implement an advanced **yoga pose detection system** using deep learning techniques to provide real-time feedback for yoga practitioners. The system will focus on:

- Enhancing the accuracy of yoga pose recognition using the latest models like **MediaPipe BlazePose**.
- Integrating **machine learning algorithms** to offer real-time feedback on posture alignment and movement.
- Addressing multi-person detection challenges, allowing the system to work in environments where multiple people are practicing yoga simultaneously.
- Collecting a new dataset for yoga poses to improve the robustness of the model and expand its capabilities.
- Developing a user-friendly interface for practitioners to visualize their posture, receive corrections, and track progress over time.

Methodology Adopted in the Proposed System:

The methodology adopted in the proposed system is divided into several phases:

1 **Data Collection and Preprocessing**:
   - o Collect images and videos of different yoga poses to create a dataset. Data augmentation techniques such as rotation, scaling, and flipping will be applied to enhance the dataset and ensure diversity. o Key points for each pose will be labeled, and the data will be organized in CSV or JSON format for ease of processing.

2 **Pose Detection Model Selection**:
   - o **MediaPipe BlazePose** will be used for pose estimation due to its efficiency and the ability to extract **3D key points** without the need for depth sensors.
   - o Additional algorithms, such as **Convolutional Neural Networks (CNN)** and **Long Short-Term Memory (LSTM)** networks, will be employed for pose classification and recognition.

3 **Pose Recognition and Feedback**: o The pose data will be passed through a classifier (e.g., **XGBoost**) for yoga pose recognition.
   - o **Real-time feedback** will be provided by comparing the detected pose with the reference pose and calculating the difference (i.e., angle errors and alignment).

4 **Integration with User Interface**:
   - o A simple **user interface (UI)** will be developed to visualize the detected poses and feedback, allowing users to track their progress.
   - o The system will output suggestions for pose corrections, such as adjustments in body alignment, posture, or balance.

5 **Multi-Person Detection**:
   - o A multi-person pose detection model will be implemented to enable tracking of multiple yoga practitioners in a single frame.
   - o Techniques such as **tracking algorithms** (e.g., **SORT** or **DeepSORT**) will be used to handle multiple individuals.

6 **Performance Evaluation**: o The system's performance will be evaluated based on **accuracy, precision, recall**, and **F1-score**. o The model will be tested on a new dataset of yoga poses to ensure generalization and robustness.

Technical Features of the Proposed System:

1. **Real-Time Yoga Pose Detection**: The system uses **MediaPipe BlazePose** for **real-time pose estimation**, providing accurate keypoints for human joints and limbs. The system detects poses in real-time, ensuring prompt feedback for practitioners.

2. **Pose Recognition Accuracy**: The system employs deep learning techniques such as **CNNs** and **LSTMs** for high-accuracy pose recognition. The trained models will be able to classify various yoga poses with high precision and offer corrective feedback on posture alignment.

3. **3D Pose Estimation**: Unlike traditional 2D pose detection systems, the proposed system leverages **3D pose estimation** through **BlazePose**. This allows for more accurate tracking of body positions in space, providing a better understanding of posture alignment and form.

4. **Real-Time Feedback for Yoga Practitioners**: The system offers real-time feedback on posture alignment, providing suggestions to improve the pose. The feedback will be based on the deviation of the detected pose from the reference pose, helping practitioners adjust their movements accordingly.

5. **Multi-Person Detection**:The system will support **multi-person detection**, allowing it to handle scenarios where multiple yoga practitioners are visible in the frame. The system tracks each individual separately and provides feedback specific to each practitioner's pose.

6. **User Interface (UI)**:The proposed system will include an intuitive user interface that allows practitioners to visualize their poses, see real-time feedback, and track progress over time. The interface will display the detected pose alongside the expected correct pose and highlight areas that need improvement.

7. **Dataset Expansion**:The system will contribute to the existing yoga pose detection community by adding a more comprehensive and diverse dataset, covering a wide range of yoga poses. This dataset will improve the accuracy and generalization of the model.

8. **Error Calculation and Correction**:For each detected pose, the system will calculate the error between the current pose and the ideal pose (e.g., joint angles, posture angles) and provide suggestions for correction.

9. **Integration with Machine Learning Models**:The model will integrate **XGBoost** and other machine learning algorithms to classify yoga poses based on 3D keypoints. This ensures improved accuracy and efficient performance for pose recognition.

10. **Data-Driven Insights**:The system will provide practitioners with data-driven insights, such as posture progress over time, consistency in alignment, and any potential risk areas for injury prevention.

2.3 Tools and Technologies Used

Platform / Tools Used in Implementing the Project

1. **Python Programming Language**:

Python is the core programming language for this project. Its flexibility, simplicity, and extensive ecosystem of libraries make it ideal for developing deep learning applications. Python is used for tasks such as data preprocessing, model training, and real-time predictions.

2. **TensorFlow**;

TensorFlow is an open-source machine learning framework widely used for creating and training deep learning models. In this project, TensorFlow facilitates the development of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to process and classify yoga poses. Its GPU support accelerates the training and inference processes.

3. **OpenCV (Open Source Computer Vision Library)**:

OpenCV is utilized for real-time image and video processing. It plays a vital role in capturing live video streams, preprocessing frames, and preparing the data for pose estimation. OpenCV ensures efficient handling of visual inputs, critical for real-time performance.

4. **MediaPipe**:

MediaPipe is a comprehensive framework for building multimodal machine learning pipelines. This project uses its BlazePose module for detecting human key points (landmarks) in real-time. MediaPipe generates 2D and 3D key point data, which forms the foundation for pose evaluation.

5. **Google Colab / Jupyter Notebook**:

These platforms are employed for developing, training, and testing models. Google Colab provides free access to GPUs, making it suitable for running computationally intensive tasks, while Jupyter Notebook enables interactive coding and visualization.

6. **Numpy & Pandas**:

Numpy: Provides support for numerical computations, including array manipulation and mathematical operations essential for preprocessing and feeding data into models.

Pandas: Facilitates data handling by organizing key point coordinates into tabular formats, enabling seamless storage, retrieval, and analysis of data stored in CSV files.

7. **Matplotlib & Seaborn**:

These libraries are used for visualizing data and analyzing the performance of models. Matplotlib provides flexibility in creating plots, while Seaborn simplifies the creation of attractive and informative visualizations.

8. **SoftMax Layer**:

The SoftMax layer, located at the output of the neural network, computes the probabilities of each yoga pose in real-time. This ensures accurate classification by identifying the pose with the highest likelihood.

2.4 Hardware and Software Requirements

Hardware Requirements

1. **Processor**:Minimum:

- Intel Core i5 or AMD Ryzen 5.
- Recommended: Intel Core i7 or AMD Ryzen 7 for handling computationally intensive tasks and real-time processing effectively.

2. **RAM**:

   - Minimum: 8 GB for basic tasks.
   - Recommended: 16 GB or higher to support smooth multitasking, including real-time data streaming and model inference.

3. **GPU**:

   - Minimum: NVIDIA GeForce GTX 1050 for basic deep learning tasks.
   - Recommended: NVIDIA RTX 3060 or higher for faster training and inference.

4. **Storage**:

   - Minimum: 256 GB SSD for operating system and basic files.
   - Recommended: 512 GB SSD or higher to store datasets, trained models, and related files efficiently.

5. **Webcam**:

   - A webcam is necessary for capturing live video streams of the user performing yoga poses. The recommended resolution is 720p or higher for better accuracy in pose estimation.

Software Requirements

1. **Operating System**:

   - The project can run on multiple operating systems, including:

     - Windows 10/11.
     - macOS.
     - Linux (Ubuntu recommended for better compatibility with deep learning libraries).

2. **Development Environment**:

   - **Python (3.8 or higher)**: All programming and execution are done using Python, leveraging its powerful libraries for deep learning and computer vision.

3. **Libraries and Frameworks**:

   - **TensorFlow**: For designing and training CNN and LSTM models. ○ **MediaPipe**: For real-time pose estimation. ○ **OpenCV**: For handling video capture and frame processing. ○ **Numpy & Pandas**: For data manipulation and preprocessing.

   - **Matplotlib & Seaborn**: For data visualization and performance analysis.

4. **IDE/Code Editors**:
    - PyCharm, Jupyter Notebook, or Visual Studio Code for writing and debugging Python code.
5. **Database Management**:
    - The project uses lightweight CSV files to store pose data, key point coordinates, and training results.
6. **Version Control Tools**:
    - **Git/GitHub**: For tracking changes, collaborative development, and managing project repositories.
7. **Additional Tools**:
    - Docker (optional): For containerizing the project and ensuring platform-independent deployment.
    - Virtual Environment: Virtual environments are used to manage dependencies and isolate project-specific libraries.

Chapter 3 SOFTWARE REQUIREMENT SPECIFICATIONS

3.1 Introduction

Definitions, Acronyms, and Abbreviations

- **MediaPipe**: A cross-platform library for real-time, multi-person keypoint detection used for human pose estimation.
- **CNN**: Convolutional Neural Network, a deep learning model specialized in extracting spatial features from images.
- **LSTM**: Long Short-Term Memory, a type of recurrent neural network suitable for analyzing sequential data like pose movements.
- **SoftMax**: A mathematical function that converts model outputs into probabilities for classification tasks, helping identify the most likely yoga pose.

Overview

The project focuses on creating a **real-time yoga pose detection system** designed to help users practice yoga accurately and safely. The system employs computer vision and deep learning techniques to analyze video inputs, extract key points from the user's body, and evaluate the correctness of yoga poses. The ultimate goal is to provide actionable feedback for improving yoga practice.

3.2 General Description

Product Perspective

The yoga pose detection system is a cutting-edge software application that integrates **computer vision** and **deep learning** technologies. It takes video input from a user's camera, analyzes it using advanced machine learning models, and provides real-time feedback. The system is envisioned as a standalone application but can be extended for integration with fitness platforms or wearable devices in the future. **Product Functions**

- **Pose Detection**: Identifies key body points such as shoulders, elbows, knees, and ankles in real-time using MediaPipe.
- **Pose Classification**: Uses CNN to analyze spatial patterns and LSTM for temporal pose sequence analysis.
- **Feedback Generation**: Provides corrective feedback to the user based on the detected pose and predefined accuracy thresholds.

- **Progress Tracking**: Optionally stores session data for users to track their improvement over time.

User Characteristics

The product is tailored for:

- **Yoga Enthusiasts**: Individuals practicing yoga for personal fitness and well-being.
- **Fitness Trainers**: Professionals monitoring and guiding clients remotely.
- **Rehabilitation Centers**: Physiotherapists helping patients recover through guided yoga sessions.

Users are expected to have:

- Basic knowledge of yoga postures.
- Access to a device with a camera for video input.

General Constraints

- The system requires a well-lit environment for optimal pose detection.
- Performance may decline if multiple users are in the frame or if body parts overlap significantly.
- Real-time processing demands moderate to high computational power.

Assumptions and Dependencies

- The user remains within the camera's field of view throughout the session.
- A stable internet connection may be required for updates and cloud-based processing (if implemented).
- Relies on pre-trained machine learning models and labeled datasets for accurate pose classification.

3.3 Functional Requirements

Introduction

The system functionality is categorized into three primary operations: input, processing, and output, ensuring seamless interaction with the user.

Input

- **Video Feed**: A live video stream captured from the user’s camera.
- **Threshold Parameters**: Adjustable metrics defining pose accuracy for feedback generation.

Processing

- Key body points are detected using the **MediaPipe** library.

- Pose classification is performed using a **CNN-LSTM hybrid model** to ensure spatial and temporal analysis of user movements.
- Corrective feedback is generated based on predefined pose alignment criteria. **Output**
- The detected yoga pose, along with its probability score.
- Feedback such as “Adjust your posture” or “Pose is correct” displayed in real-time.

3.4 External Interfaces Requirements

User Interfaces

- **Graphical Interface**: Displays real-time video feed, detected key points, pose classification results, and corrective feedback.
- **Controls**: Start/stop video feed, select yoga poses for practice, and adjust pose accuracy thresholds.

Hardware Interface

- **Camera**: A high-definition camera for accurate key point detection.
- **Processing Unit**: A device with a minimum configuration of 8GB RAM and a GPU for real-time processing.

Software Interface

- **MediaPipe Library**: For detecting and tracking key body points.
- **TensorFlow/Keras**: For implementing and training deep learning models.
- **CSV File Storage**: For saving and analyzing session data.

3.5 Non-Functional Requirements

- **Performance**: Must process and classify poses within 100 milliseconds per frame for real-time feedback.
- **Reliability**: Provide accurate feedback consistently across various lighting conditions and poses.
- **Scalability**: Easily extendable to include additional yoga poses in future versions.
- **User-Friendly**: Intuitive interface requiring minimal setup or technical expertise.

3.6 Design Constraints

Standard Compliance

- Adheres to established frameworks like TensorFlow and Keras.

- Compliant with privacy standards, ensuring video data is processed securely and not stored unless explicitly permitted by the user.

Hardware Limitations

- Requires a modern processing unit with GPU capabilities for efficient operation.
- May not perform optimally on low-powered devices.

3.7 Other Requirements

- **Customization**: The user can define their own accuracy thresholds for pose feedback.
- **Integration Potential**: Can be paired with fitness trackers or smart mirrors for enhanced functionality.
- **Platform Independence**: Designed to be compatible with Windows, macOS, and Linux platforms.
- **Portability**: Potential to develop a mobile application version for smartphones and tablets.

Chapter 4 Architectural Design of the Project

4.1 Problem Specification

The primary goal of the project is to develop a real-time yoga pose detection system that utilizes computer vision and deep learning techniques. The system will accurately identify and evaluate yoga postures, providing feedback to the user for self-correction and improvement. This will aid in reducing injuries, improving posture accuracy, and encouraging consistent yoga practice.

Block Diagram

The block diagram illustrates the sequential workflow of the **Yoga Pose Tutor System**, highlighting the key technologies and methods used in each stage. Below is a detailed description of each step:

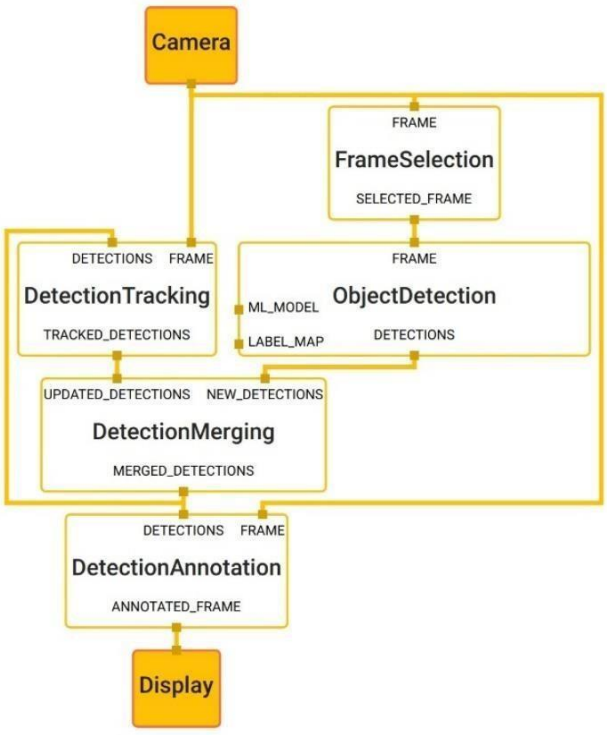
1. **Captures Video Feed from a Camera** ○ The system starts by capturing a real-time video feed from the user's camera. This input serves as the raw data for further processing.
    - **Technology Used**: Standard webcam or phone camera integration.
2. **Uses MediaPipe to Detect Key Points and Preprocess Data** ○ The captured video is processed using **MediaPipe**, which identifies key points (joints of the body) and extracts the required data for pose estimation. The data is then preprocessed to ensure accuracy in the subsequent steps.
    - **Technology Used**: MediaPipe framework for key point detection.

3. **CNN Extracts Spatial Features from Frames** ○ A **Convolutional Neural Network (CNN)** is applied to the processed frames to extract spatial features, which capture the structure and position of the user's body in the image. These features are critical for understanding body alignment.
    - **Technology Used**: CNN architecture for feature extraction.
4. **KNN Classifier Predicts the Yoga Pose** ○ The extracted features are passed to a **K-Nearest Neighbors (KNN)** classifier. This machine learning algorithm predicts the user's yoga pose based on pre-trained data.
    - **Technology Used**: KNN classifier for pose prediction.
5. **Provides Feedback Based on Accuracy Thresholds** ○ The system evaluates the predicted yoga pose against predefined accuracy thresholds and provides feedback to the user. This feedback ensures that the user can correct their posture and improve their practice.
    - **Technology Used**: Threshold-based feedback mechanism.

Data Definition/Dictionary

- **Input Data**: Real-time video feed containing human poses in yoga postures.
- **Output Data**: Predicted yoga pose, feedback on alignment, and performance metrics.
- **Key Points**: Body landmarks (e.g., joints, limbs) detected by the MediaPipe library.
- **Thresholds**: Predefined angles and positions for each yoga posture to evaluate accuracy.

Module Specification



1. **Pose Detection Module**: Detects key points using MediaPipe.
2. **Feature Extraction Module**: Extracts spatial features using CNN.
3. **Pose Classification Module**: Combines temporal and spatial features for classification using LSTM.
4. **Feedback Module**: Compares user pose against thresholds and provides real-time corrections.

Assumptions Made

- The user is visible to the camera with minimal obstructions.
- The environment has adequate lighting for accurate pose detection.
- The dataset is representative of the yoga poses being practiced.
- The model is trained on a sufficient number of labeled images and sequences to ensure robustness.

4.2 Data Flow Diagram

Level 0:

- **Process**: The system takes video input, detects poses, and provides feedback.
- **Entities**:
  - Input: Camera Feed
  - Output: Yoga Pose Prediction & Feedback

Level 1:

The Level 1 Data Flow Diagram (DFD) provides a detailed view of the processes involved in the Yoga Pose Tutor system. It elaborates on the main processes that interact with the user to capture video data, detect body key points, classify yoga poses, and provide feedback.

1. **Process 1.1: Capture Camera Feed** ○ **Description**: This process receives a real-time video feed from the user's camera.

   The input video is broken down into individual frames for further processing.

- **Input:** Video (Real-time camera feed). ○ **Output:** Frames (Processed images of the video feed).

2. **Process 1.2: Key Point Detection** ○ **Description:** This process extracts critical key points of the user's body from the captured frames using pose estimation techniques like MediaPipe or OpenPose.

   These key points represent body joints and are saved for further analysis.

   - **Input:** Frames (Captured from Process 1.1).
   - **Output:** Key Points (Coordinates representing body posture).

3. **Process 1.3: Pose Classification** ○ Description: This process uses a deep learning model (e.g., CNN and LSTM) to classify the detected key points into a specific yoga pose. It determines which yoga pose is being performed by the user.

   - **Input**: Key Points (Detected in Process 1.2). ○ **Output**: Pose (Yoga pose classification result).

4. **Process 1.4: Feedback Mechanism** ○ **Description:** This process evaluates the accuracy of the performed yoga pose compared to a predefined correct pose. It provides real-time feedback to the user on their pose alignment and correctness, ensuring proper posture during practice.

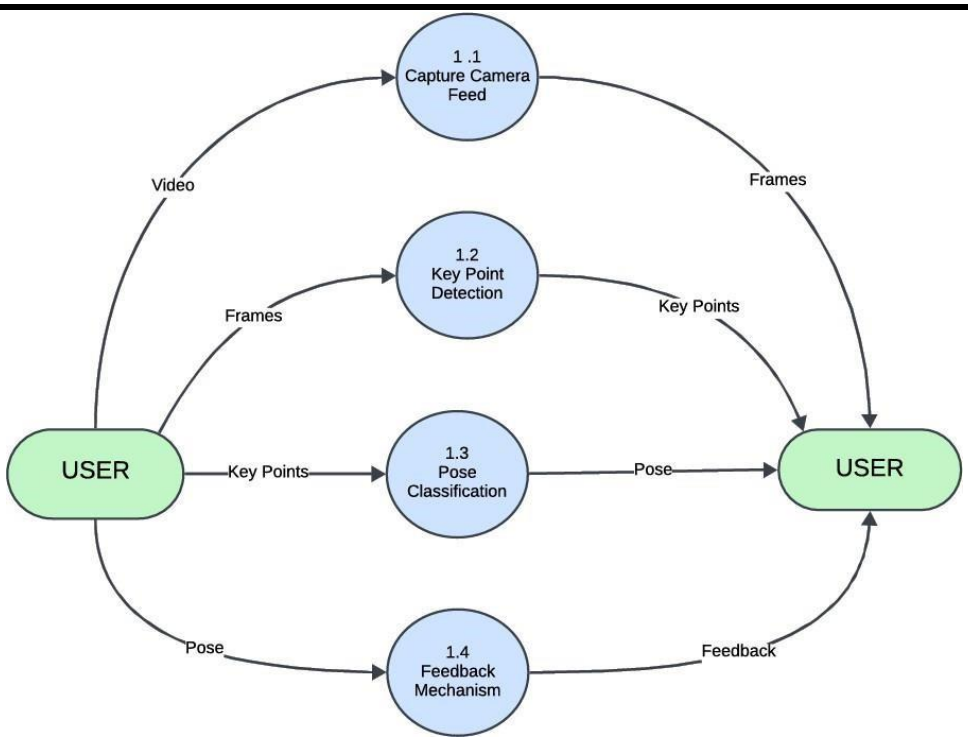
   - **Input:** Pose (Classified in Process 1.3).
   - **Output**: Feedback (Accuracy and suggestions to improve the pose).

External Entities:

- **User:** Interacts with the system by providing the camera feed and receiving feedback on their yoga pose.

Data Flow:

- **Video:** Sent from the user to the system for pose detection.
- **Frames:** Sent to the key point detection process for analysis.
- **Key Points:** Used for pose classification to identify the yoga pose.
- **Pose**: Provided to the feedback mechanism to generate user-specific suggestions.
- **Feedback:** Delivered back to the user for posture improvement



Description of the ANN/DL Architecture/Algorithm Used

1. **CNN (Convolutional Neural Network)**:
   - Extracts spatial features from individual video frames. ○ Key Layers:
     - Convolutional layers for feature detection.
     - Max-pooling layers to reduce dimensions.
     - Fully connected layers for feature representation.
2. **KNN Classifier**:
   - Outputs probabilities for each yoga pose.
   - Selects the pose with the highest probability as the final prediction.
3. **Training Approach**:
   - Dataset: Preprocessed and augmented yoga pose dataset.
   - Loss Function: Categorical cross-entropy. ○ Optimizer: Adam for efficient convergence.

Chapter 5 IMPLEMENTATION

The yoga pose detection project utilizes MediaPipe for extracting pose landmarks from video frames and a Convolutional Neural Network (CNN) for classifying the detected poses. MediaPipe's Pose module identifies key body landmarks, such as joints and angles, which are then preprocessed and fed into the CNN model for training and prediction. The model categorizes the input into predefined yoga poses, achieving high accuracy in real-time detection.

5.1 Code Snippets

5.1.1. Importing Libraries

```
import cv2
import mediapipe as mp
import cv2
import os
import pickle
import pandas as pd

from landmark_plotting import plot_pose
```

Figure 5.1 Importing libraries

The following libraries were imported to facilitate the implementation of this project. (Figure 5.1) :

1. **Computer Vision (CV2):** Used for computer vision tasks, such as processing video streams and handling image input/output operations.
2. **Mediapipe:** A cross-platform framework for building multimodal applied machine learning pipelines. It was primarily utilized for pose estimation in this project.
3. **OS:** Facilitates interaction with the operating system, including file path manipulations and directory traversal.

4. **Pickle**: Enables the serialization and deserialization of Python objects, making it convenient for saving and loading intermediate data states.
5. **Pandas:** Provides efficient data structures for manipulating and analyzing data, particularly useful for handling pose data in tabular form.
6. **Landmark_plotting.plot_pose**: A custom module designed to visualize pose landmarks, aiding in the validation and interpretation of results.

2. Model Building and Training

```
KNN=KNeighborsClassifier(n_neighbors=5,p=2)


# training the model
KNN.fit(X_train,y_train)
```

Figure 5.2 KNN for Point Fitting

The extracted pose landmarks, converted into numerical points and stored in a CSV file, were used to train a K-Nearest Neighbors (KNN) classifier. The `KNeighborsClassifier` was initialized with 5 neighbors and Euclidean distance (`p=2`) as the metric. The model was trained using the `fit` method on the preprocessed training data (`X_train`, `y_train`), which contained the landmark coordinates and their corresponding yoga pose labels. This model effectively clusters the pose data points and classifies them into predefined yoga pose categories.

5.2.1 Data Preparation:

Pose landmarks of yoga poses were extracted using MediaPipe's Pose module and converted into a structured format (CSV file). Each row in the CSV represented a pose, with columns corresponding to key body landmark coordinates.

5.2.2 Model Selection:

K-Nearest Neighbors (KNN) classifier was chosen for its simplicity and effectiveness in clustering similar data points. The KNN model was initialized with:

- `n_neighbors=5`: Considering the 5 nearest neighbors for classification.
- `p=2`: Using the Euclidean distance as the metric for measuring proximity.

5.2.3 Training the Model:

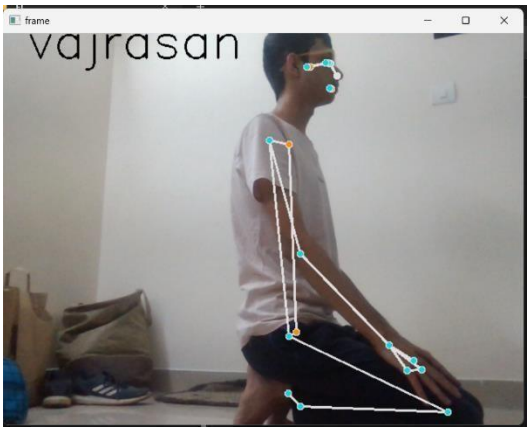
- The extracted landmark data (`X_train`) and their corresponding yoga pose labels (`y_train`) were used to train the KNN model.
- The `fit` method was applied to map the landmark points to their respective yoga pose classes.

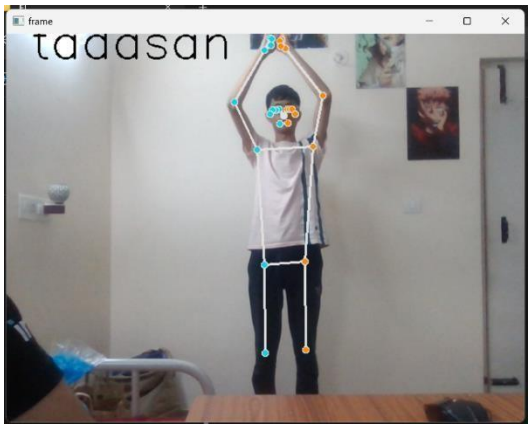
5.2.4 Pose Classification:

After training, the model could predict yoga poses by comparing the input pose's landmark points to its learned clusters and classifying it based on the majority class among the nearest neighbors.

3. Results and Discussions with screenshots

5.3.1. Results





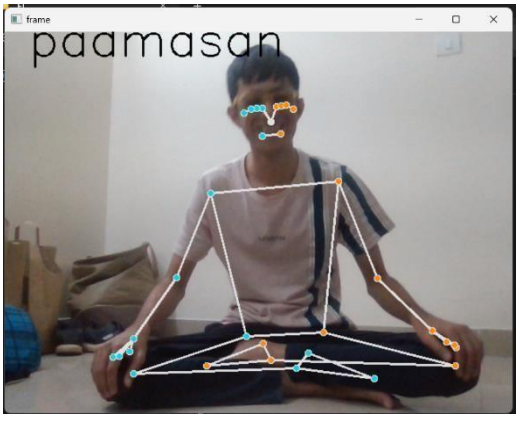


Figure 5.3 Posture Detection

The yoga pose detection model demonstrated successful identification and classification of various yoga poses. Based on the screenshots(5.2) , the following results were observed:

1. **Tadasana**: The model accurately identified the standing pose with arms aligned, reflecting correct landmark clustering and classification.
2. **Padmasana**: The model effectively recognized the seated pose with crossed legs, indicating robust performance on poses with symmetrical landmark points.
3. **Vajrasana**: The kneeling pose was correctly classified, showcasing the model's ability to distinguish poses involving lower body landmarks.
4. **Paschimottanasana**: The forward-bending seated pose was successfully detected, highlighting the model's precision in capturing complex poses involving bending and stretching.
5. **Padahastasana**: The model accurately identified the standing forward bend pose, indicating its reliability in classifying poses with close proximity between upper and lower body landmarks.

5.3.2 Accuracy Score and model saving

Figure 5.4 Accuracy and Model Saving

1. Model Accuracy:

The yoga pose detection model achieved an excellent accuracy of **99.55%**, demonstrating its high effectiveness in correctly identifying and classifying various yoga poses. (Fig. 5.4).

This near-perfect accuracy reflects the robustness of the extracted pose landmarks using MediaPipe and the efficiency of the KNN algorithm in clustering and classifying the data points into distinct pose categories.
The high accuracy ensures the model's reliability in practical scenarios, making it a strong candidate for real-world yoga pose detection and analysis applications.

2. Model Saving:

To make the yoga pose detection system reusable and efficient for future predictions, the essential components of the pipeline were saved using `.pkl` files:

- **Label Encoder**: The label encoder, which converts raw image data into numerical datapoints (landmarks), was saved as a `.pkl` file. This ensures that the preprocessing pipeline remains consistent and eliminates the need to reprocess the raw images each time.
- **Classifier**: The trained KNN classifier, responsible for classifying the yoga poses based on the extracted landmark points, was also saved as a `.pkl` file. This allows for quick and seamless loading of the model for inference without retraining.

The use of `.pkl` files ensures that both the preprocessing and classification stages are preserved in a ready-to-use format, significantly simplifying the deployment process.

This approach not only saves computational resources but also enhances the portability of the model, making it suitable for deployment in real-time applications like yoga tracking apps or fitness monitoring systems.

Chapter 6 CONCLUSION

This project introduces a comprehensive real-time yoga monitoring system leveraging advanced deep learning techniques. The integration of CNN and LSTM models, alongside the MediaPipe library, ensures accurate detection and analysis of yoga poses, enabling practitioners to receive real-time feedback on their postures. The system captures key body coordinates, processes them into image sequences, and evaluates them using a SoftMax classifier, providing users with actionable insights to improve their practice.

The proposed solution addresses challenges such as pose accuracy, alignment feedback, and user engagement, thereby reducing the risk of injuries and enhancing the overall yoga experience. Additionally, the system's threshold mechanism optimizes computational efficiency while ensuring precise feedback, fostering consistent and safe practice. This research lays a strong foundation for future advancements in pose detection and self-guided fitness programs, offering potential for applications in rehabilitation, fitness training, and beyond.

Chapter 7 FUTURE WORK

The current system demonstrates significant potential for real-time yoga pose detection but also highlights areas for further improvement and expansion. Future work will focus on the following aspects:

1. **Expanding Dataset**: The current dataset covers only six yoga asanas. Future research will involve expanding the dataset to include a broader range of yoga poses, capturing variations performed by individuals in diverse indoor and outdoor environments. This will enhance the generalizability and robustness of the model.
2. **Improving Pose Estimation Accuracy**: The performance of the models is heavily reliant on the accuracy of OpenPose for pose estimation. Addressing challenges such as overlapping individuals or body parts will involve exploring advanced pose estimation frameworks and algorithms that are robust in crowded or complex scenarios.
3. **Portable Device Implementation**: To enhance accessibility and usability, the system can be integrated into portable, real-time devices, such as wearables or standalone applications, enabling users to self-learn and receive immediate feedback during their yoga practice.
4. **Multi-Person Detection**: Future iterations of the system will address limitations related to detecting and assessing poses for multiple individuals in a single frame, making it suitable for group settings, such as yoga classes or workshops.
5. **Integration into Other Domains**: The methodologies developed for yoga pose detection can be adapted for applications in sports training, healthcare monitoring, surveillance, and rehabilitation, offering a versatile solution for position tracking and analysis across various fields.
6. **Real-Time Feedback Enhancement**: Incorporating advanced feedback mechanisms, including audio guidance and augmented reality overlays, will provide users with detailed, real-time corrective suggestions, improving the overall effectiveness of the system.

REFERENCES

[1] Liaqat,S.,Dashtipour,K.,Arshad,K.,Assaleh,K.,& Ramzan,N.(2021,April 1) A Hybrid Posture Detection Framework: Integrating Machine Learning and Deep Learning IEEE [2] Rajendran,A.K.,&Sethuraman,S.C.(2023 Februrary).A Survey on Yogic Posture Recognition.IEEE

[2] Swain,D.,Satapathy,S.,AcharayaB.,Shukla M.,Gerogiannis,V.C.,Kanavos &Giakovis,D.(2022,October31).Deep Learning Model for Yoga Pose Monitoring Algorithm

[3] Swain ,D.,Satapathy,S.,Patro,P.(2022,June 27).Yoga Pose Monitoring System using Deep Learning ,ResearchSquare

[4] Bhoite,P.,Kalekar,O.,Bhandari,S.,&Dhavas,N. (2023,April4).Yoga Posture Detection and Correction System International Reserch Journal of Modern Education and Technology

[5] Upadhyay, A., Basha, N. K., & Ananthakrishnan, B. (2023, February 17). Deep Learning-Based Yoga Posture Recognition Using the Y_PN-MSSD Model for Yoga Practitioners. Healthcare

[6] Kishore, D. M., Bindu, S., & Manjunath, N. K. (2022, September 5). Estimation of Yoga Postures Using Machine Learning Techniques. International Journal of Yoga

[7] Sunney, J. (2022). Real-Time Yoga Pose Detection using Machine Learning Algorithm.

[8] Bembde, M., Barude, S., Shinde, P., Thorat, T., & Thakar, D. (2022, July). Yoga Posture Detection and Correction System. International Journal of Advanced Research in Science, Communication and Technology, "www.ijarsct.co.in"

[9] Long, C., Jo, E., & Nam, Y. (2021, September 20). Development of a yoga posture coaching system using an interactive display based on transfer learning

[10] Narayanan, S. S., Misra, D. K., Arora, K., & Rai, H. (2021, May 13). Yoga Pose Detection Using Deep Learning Techniques. SSRN.

[11] Yadav, S. K., Singh, A., Gupta, A., & Raheja, J. L. (2019, May 9). Real-time Yoga recognition using deep learning. Neural Computing and Applications

[12] Kothari, S. (2020, May). Yoga Pose Classification Using Deep Learning (Master's thesis).

[13] D. Mohan Kishore, S. Bindu, Nandi Krishnamurthy Manjunath. (2022). "Estimation of Yoga Postures Using Machine Learning

[14] Kumar, D. (2020). "Yoga Pose Detection and Classification Using Deep Learning." In Proceedings of the IEEE International Conference on Artificial Intelligence and Machine Learning

[15] Murthy, N. A., Kumar, B. R. M., & Singru, N. M. (2015). "Real-time yoga posture recognition and correction system using Kinect sensor." International Journal of Computer Science and Information Technologies,

[16] Bhattacharya, S., Borkar, J. H., & Dnyaneshwar. (2021). "Yoga Pose Detection using Convolutional Neural Networks." Proceedings of the International Conference on Computer Vision and Pattern Recognition