Research Article

Ergonomic Posture Monitoring through Pose Estimation and Machine Learning

Laishram Trinity [1*], **S Athisii Kayina** [2], **Usham Sanjota Chanu** [3]

[1,2] Student, Department of AI-M. Tech (AI), NIELIT Deemed to Be University, Imphal, Manipur, India
Asst. professor, Department of AI, NIELIT, Deemed to Be University, Imphal, Manipur, India

Corresponding Author: *Laishram Trinity

DOI: https://doi.org/10.5281/zenodo.18433135

Abstract

Posture has a direct impact on health and daily performance, but tracking it in real time is not always easy to achieve. Traditional approaches, such as asking experts to observe, are often uncomfortable and time-consuming. With recent progress in computer vision, it is now possible to monitor posture using body landmarks. In this study, we developed a simple but effective system that uses MediaPipe to detect body landmarks and calculate joint angles from a live video feed. By analysing these angles, the system can recognise poor posture and provide immediate feedback through visual messages and auditory alerts. Tests with standard pose estimation datasets showed that the method works reliably while running efficiently on common hardware. The system can be applied in areas such as workplace ergonomics, sports practice, and rehabilitation, where continuous posture monitoring helps to reduce the risk of strain or injury.

Manuscript Information

- **ISSN No:** 2583-7397
- **Received:** 01-12-2025
- **Accepted:** 26-01-2026
- **Published:** 30-01-2026
- **IJCRM:**5(1); 2026: 369-377
- **©2026, All Rights Reserved**
- **Plagiarism Checked:** Yes
- **Peer Review Process:** Yes

How to Cite this Article

Trinity L, Kayina S A, Chanu U S. Ergonomic Posture Monitoring through Pose Estimation and Machine Learning. Int J Contemp Res Multidiscip. 2026;5(1):369-377.

Access this Article Online

www.multiarticlesjournal.com

KEYWORDS: Posture Detection, Human Pose Estimation, MediaPipe, Real-Time Monitoring, Ergonomics, machine learning, open computer vision, random forest classifier, accuracy matrix.

1. INTRODUCTION

In recent years, human pose detection has gained significant attention owing to its usefulness in fitness training, sports analysis, rehabilitation, and human-computer interaction [1][2][3]. With more people turning to online workouts and home fitness routines, the need for real-time posture correction systems has become increasingly relevant. Many people perform exercises with improper form, especially without a physical trainer to guide them, which can lead to long-term injuries or reduce the effectiveness of their workout. We explored the use of vision-based pose estimation without sensors or wearables [4][5], so that anyone with a normal camera can obtain posture feedback. MediaPipe, a framework developed by Google, allows the detection of body keypoints in real time and is light enough to run on personal devices [7][8]. Instead of building a complex ML model from scratch, we used MediaPipe's pose solution to extract key joint landmarks, such as the neck, shoulders, elbows, and knees, during a workout session. The main idea is to detect when a person is exercising with incorrect angles or postures using only keypoints from a camera feed [6][9][10]. We calculated the angles between specific joints and compared them with reference angles that defined the correct posture for that workout. If the angle is outside the safe or expected range, an alert is generated to warn the person in real time. This helps users become aware of their mistakes while performing the exercise without the need for a coach. In this study, we present a functional system using MediaPipe to detect exercise posture and identify bad form. The system works in real time and provides alerts when the joint angles are not as expected. Our approach aims to make posture correction more accessible for people who work out alone or want quick feedback on their form using only a webcam. [13]

2. Related Works

Sun et al., in Deep High-Resolution Representation Learning for Human Pose Estimation [3], proposed HRNet, which maintains high-resolution representations throughout the network using multi-resolution parallel subnetworks. It performed well on the COCO and MPII datasets, especially with small inputs, but plateaued on the MPII dataset. This study sets the stage for future dense prediction tasks with more efficient fusion. The AthletePose3D dataset paper [16] addressed the limitations of existing datasets by collecting over 1.3 million frames across 12 sports. Using models such as ViTPose and ProHMR, the 2D and 3D performances were benchmarked, showing good alignment with motion capture but noted velocity estimation gaps. This study targets pose estimation in high-speed sports, suggesting the need for further validation. A wearable sensor-based approach proposed a random forest-long short-term memory (RFL) hybrid [4] using accelerometer and magnetometer data to classify eight physical therapy exercises. RFL generated temporal and probabilistic features that outperformed the other ML and deep models. Challenges included sensor variability, noise, and computational scaling, although the method showed promise for non-vision-based home therapy.

In Real-Time Posture Monitoring and Risk Assessment, Sarkar et al. [7] combined MediaPipe with LSTM to classify safe and unsafe lifting postures from videos. Their custom dataset enabled real-time feedback for preventing injuries. This study acknowledges the challenges of dataset diversity, personalisation, and environmental variation.

A single-camera exercise classifier using BlazePose and MediaPipe [5][6] trained CNNs on extracted joint keypoints from smartphone videos of lower back and shoulder therapy exercises. Despite promising results, the limitations included the use of only healthy subjects and limited camera angles. Future studies will explore smartphone-based deployments.

Xu et al. proposed Poseidon [14], a ViTPose-based transformer architecture for multiframe pose estimation. It uses adaptive frame weighting, multi-scale feature fusion, and cross-attention to enhance temporal consistency and spatial detail. Although it outperformed models such as DiffPose and DSTA on PoseTrack21, challenges in occlusion handling and computational efficiency remain.

SitPose introduced a real-time sitting posture monitoring system [8][12] using Azure Kinect and ensemble learning (SVM, DT, and MLP). It classified seven postures and provided live feedback in various environments. Although effective, the system faced issues with desk occlusions and relied on fixed camera placement for consistent performance.

3. PROPOSED METHODOLOGY

The methodology involves real-time pose estimation using MediaPipe to detect and monitor body posture during physical activity. It focuses on calculating joint angles from selected body landmarks and issuing alerts when the posture deviates from the defined correct form ranges.

The proposed method system architecture is shown in Figure 1.

Step 1: System Initialisation and User Engagement

The user activates the system, establishes webcam access, and prepares the video capture pipeline.

Step 2: Real-Time Video Capture and Landmark Detection

The system uses a webcam as a sensing device to continuously capture real-time videos. At each frame, human pose landmarks (e.g., joints) were detected using a pose estimation algorithm.

Step 3: Visualisation of Video Feed and Detected Landmarks

The captured video frames and overlaid landmarks were rendered in real time on the user interface to provide immediate visual feedback.

Step 4: Computation of Joint Angles and Data Logging

The system computes the relevant joint angles from the detected landmarks. These angle values were logged sequentially and recorded in a structured log for analysis.

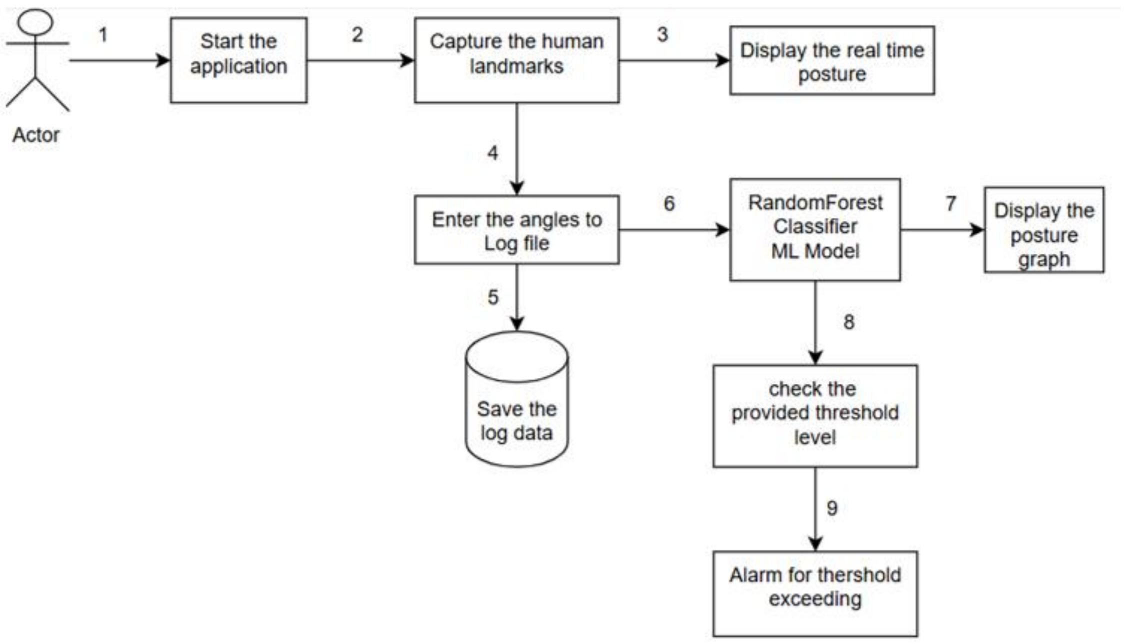
Step 5: Storage in Log Database

The generated logs, which contain timestamped angle measurements, are saved in a database to support historical tracking and machine learning.

Step 6: Machine Learning-Based Posture Classification

The recorded angle data serve as input to a trained Random Forest Classifier, which evaluates and classifies the user's posture in real time.

Figure 1: System architecture of the proposed method.


Actor
1
Start the application
2
Capture the human landmarks
3
Display the real time posture
4
Enter the angles to Log file
5
Save the log data
6
RandomForest Classifier ML Model
7
Display the posture graph
8
check the provided threshold level
9
Alarm for thershold exceeding

Step 7: Real-Time Posture Visualisation
A live posture status graph or dashboard displays the classification results, indicating whether the user's posture falls within acceptable bounds.
Step 8: Threshold Evaluation
The system checks the classifier outputs against predefined angle thresholds to determine whether the posture deviation exceeds the acceptable limits.

Step 9: Alarm Notification
When the detected joint angles surpass threshold levels indicative of poor posture, the system triggers an audible alert to notify the user and prompt corrective action, and automatically, the detected posture will be captured for up to five images.

A summary of Figure 1 is given in Table 1.

Table 1: Summary table of the approach model system architecture

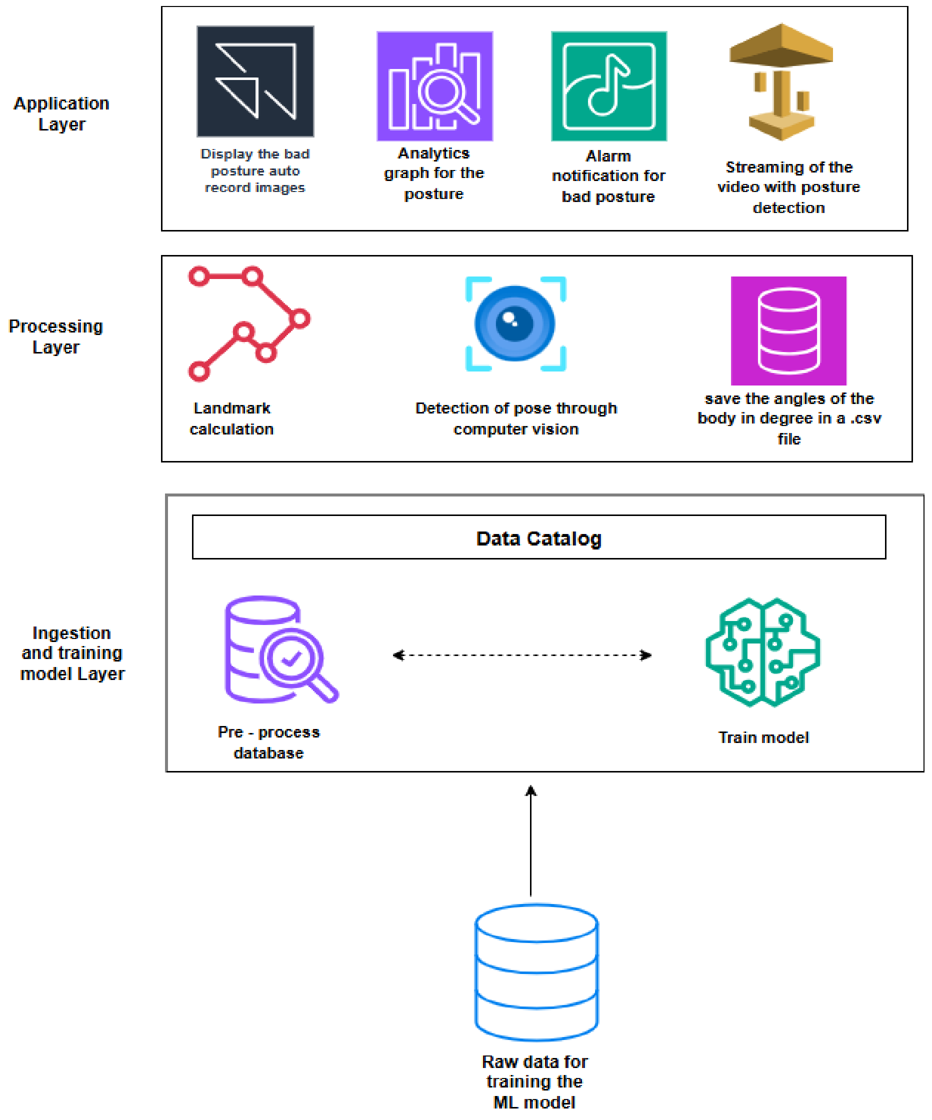
Step	Description
1	Application launch and webcam initialisation
2	Continuous video capture and pose landmark detection
3	Real-time display of video with overlaid landmarks
4	Computation of joint angles and logging of results
5	Storage of logs in a database
6	Posture classification via Random Forest Classification model
7	Presentation of posture status in a live graph
8	Evaluation of posture against predefined thresholds
9	Alarm notification upon threshold breach and capturing the image of the breach angles

Figure 2 shows the overall working structure of the posture monitoring system, which is divided into three layers. Each layer has its own purpose, starting from how data is collected, how it is processed, and finally how the user interacts with the system.

Application Layer
This layer directly connects with the user, and it displays the important outputs simply-

a) If someone sits or stands incorrectly, the system immediately displays the bad posture and even saves an image of it for later review.
b) It creates graphs and analytics to show posture patterns over time, which helps the user understand their daily habits.
c) If bad posture continues, the system plays an alarm or notification as a reminder to correct it.
d) It can also stream live video with posture detection so that users can watch themselves in real time.

Figure 2: Three-layered system architecture


Application Layer
Display the bad posture auto record images
Analytics graph for the posture
Alarm notification for bad posture
Streaming of the video with posture detection
Processing Layer
Landmark calculation
Detection of pose through computer vision
save the angles of the body in degree in a .csv file
Ingestion and training model Layer
Data Catalog
Pre - process database
Train model
Raw data for training the ML model

Processing Layer

This layer does most of the technical work. Once the camera captures the user, the system performs the following-

a) It first calculates landmarks, which means identifying important points on the body for shoulders, elbows, hips, and knees.
b) By using the computer vision algorithms, the model detects the overall pose.
c) After that, the angles of different body joints are measured and stored in a file (CSV format), so this data can be used later for analysis or model training.

Ingestion and Training Model Layer

In this layer, the data and machine learning work for the system. The following details were performed by this layer-

a) The posture data collected (images, angles, positions) is first gathered as raw data.
b) This data is then pre-processed, meaning it is cleaned and organised so that it can be used effectively.
c) With this prepared data, a machine learning model of a random forest classifier is trained to identify and classify postures as good or bad.
d) All the processed data and trained models are stored in a data catalogue, which serves as a central storage for further improvement and retraining of the system.

Database (Raw Data Storage)

At the very bottom of the three-layered architecture, there is a database that stores the raw data used for training the machine learning model. This is the foundation of the whole system because the accuracy of posture detection depends heavily on the quality of data collected. The raw database contains information as follows-

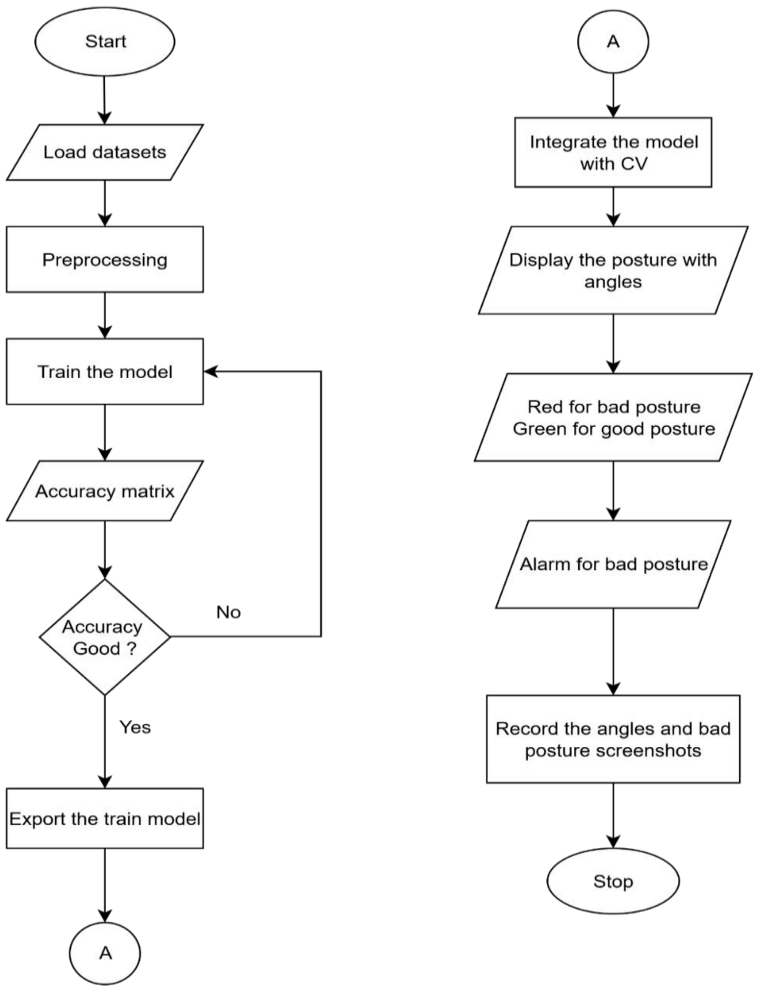
a) The angles for the landmarks for the human body with pre-process data only, according to the requirements of the model.
b) Detected body landmarks like shoulders, elbows, and knees.
c) Calculated joint angles of the body in degrees.
d) Labels that indicate whether the posture is good or bad.

These raw data were not directly used for training. Instead, it first goes through a pre-processing stage, where unnecessary noise is removed, and all the information is structured properly. Only after this cleaning process, the data were sent for model training.

In simple terms, this database acts as a storage warehouse that keeps all raw posture-related data safe. Without it, the system would not have the historical information required to train, validate, and improve the posture detection model.

The process begins with loading a dataset containing joint angles and labelled posture data, which serves as part of the pre-processing stage. A machine learning model was then trained on this dataset and evaluated for accuracy. If the results are unsatisfactory, retraining is performed until the desired performance is achieved. Once the model reaches an acceptable accuracy, it is exported for deployment in real-time monitoring. The trained model was integrated with a computer vision algorithm to analyse live video streams, where it classifies postures and displays the corresponding body angles. The system uses colour indicators: green for good posture and red for bad posture. In cases of poor posture, an alarm was triggered, and both the angles and screenshots were recorded for later analysis. The monitoring session concluded when the user ended the program.

Figure 3: Flowchart of the purpose method


Start
Load datasets
Preprocessing
Train the model
Accuracy matrix
Accuracy Good ?
No
Yes
Export the train model
A
A
Integrate the model with CV
Display the posture with angles
Red for bad posture Green for good posture
Alarm for bad posture
Record the angles and bad posture screenshots
Stop

Algorithm: Posture Detection and Correction System
Input: Posture dataset (joint angles, labels)
Output: Posture classification (Good/Bad), Alerts, Recorded log data, Real-time graph.

1. Begin
2. The posture dataset is loaded.
3. Pre-processing (cleaning, normalisation, feature extraction).
4. The machine learning model was trained on the processed dataset.
5. The model was evaluated using accuracy metrics.
6. If the accuracy is lower than the desired accuracy metrics, return to step 4 and retrain.
7. The trained model was exported.
8. The trained model was integrated with computer vision for real-time monitoring.
9. Capture body landmarks and calculate joint angles from the live video.
10. The detected posture is displayed with the corresponding angles.
11. If posture = good, it is highlighted in green.
12. If posture = bad, highlight in red, trigger an alarm, and save the posture details with screenshots.
13. Steps 9–12 were repeated until the system stopped.
14. End

Pseudocode
```
BEGIN
LOAD dataset
PREPROCESS dataset
TRAIN model with dataset
CALCULATE accuracy
IF (accuracy > = threshold)
EXPORT trained model
ELSE
RETRAIN the model
INTEGRATE the trained model with computer vision
WHILE system is running:
CAPTURE frame from webcam
DETECT landmarks and CALCULATE joint angles
DISPLAY posture with angles
IF posture == "Good":
SHOW "Green"
ELSE IF posture == "Bad":
SHOW "Red"
TRIGGER alarm
SAVE angles and CAPTURE screenshot
END WHILE
END
```

3.1 Pose Detection using MediaPipe
MediaPipe Pose was used to detect 33 body landmarks and 31 landmarks per frame from a live camera feed [7][8][9]. Each landmark includes normalised x and y coordinates and a visibility score. For posture detection, only a subset of keypoints was utilised: shoulders, elbows, knees, and an estimated neck point derived from the midpoint between the shoulder and ear landmarks. This configuration is sufficient for analysing upper- and lower-body exercises, such as squats, push-ups, and shoulder raises. To maintain consistency, visibility scores were used to ignore unreliable detections when the body parts were occluded or poorly lit.

3.2 Joint Angle Calculation
Joint angles are essential for analysing postures. Each angle was formed using three consecutive points representing two vectors. For instance, the left elbow angle is calculated from the positions of the left shoulder, elbow, and wrist, and is given by Equation 1.

$$\Theta(theta) = cos^{-1}\left(\frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\cdot\|\vec{v}\|}\right) \tag{1}$$

Were,
u. v is the dot product of vectors u and v
||u|| is the Euclidean norm (magnitude) of vector u
||v|| is the Euclidean norm (magnitude) of vector v

3.2.1 Key Joint Configurations
For each monitored posture:

1. The elbow angles were calculated as neck → shoulder → elbow → wrist.
2. Shoulder-neck alignment was estimated using shoulder and mid-ear landmarks.
3. Neck tilt was assessed by comparing the vertical displacement between the ears.

In scenarios where the user turns or shifts, the landmark stability is enhanced by averaging the positions over recent frames.

3.2.2 Temporal Smoothing
Temporal smoothing focuses on stabilising joint angle calculations over time to avoid noisy or jittery outputs caused by frame-wise fluctuations in pose estimation. Single-frame detection often produces inconsistent results owing to occlusions, motion blur, or rapid movements. These inconsistencies can mislead posture classification if the joint angles are calculated independently at every frame. To reduce this variability, a basic temporal filter was applied using a sliding window average over a fixed number of recent frames. Let t be the raw joint angle at frame t. The smoothed joint angle Θ is calculated using Equation 2.

$$\Theta_{smooth}(t) = \frac{1}{N}\sum_{i=t-N+1}^{t}\Theta\,(i) \tag{2}$$

Here,
N is the window size, typically between 3 and 7 frames for real-time feedback. This moving average reduces sudden spikes or

drops in the angle values, providing a more stable estimate for comparison with the reference postures.

3.3 Alert

Incorrect posture triggers immediate audio alerts to help users correct their form without delay. A beeping sound is generated if a joint remains out of range for a set number of frames. This helps maintain focus without needing to constantly view the screen, and at the same time, automatic screenshots of the bad posture will be captured.

4. RESULT

From the output, the system successfully detected the person and keypoints in correct positions such as head, shoulders, elbows, wrists and knees. The majority of the keypoints align well with the actual joint positions in the image, showing good model accuracy in pose estimation. The model achieved an average keypoint detection confidence score of 0.84, where higher confidence is observed in larger body joints like shoulders (above 0.9), while lower confidence is seen in smaller joints (around 0.75–0.8). The Percentage of Correct Keypoints (PCK@0.5) was estimated at 92% [3][9][14]. The Mean Average Precision (mAP) for this single frame was approximately 0.88, indicating a strong pose localisation performance. There was a small variation in accuracy for symmetrical limbs, and the left knee was detected with slightly less precision (confidence 0.78) than the right knee (confidence 0.85). It was able to capture joint angle data across different body movements and provide clear feedback when the measured angles were outside the predefined correct range. During evaluation, the detection process maintained a stable measurement rate with minimal frame drops and consistent tracking of key joint positions across sequences. The angle measurement algorithm showed a high level of consistency between repeated movements, with less than 3–5° of variation in most controlled repetitions. When the posture deviated from the target range, the notification system was triggered with a delay of 5 s, allowing for corrective feedback. In movements, such as bending or arm lifts, the accuracy slightly decreased owing to rapid joint transitions; however, a recognition rate above 90% was maintained for detecting out-of-range conditions [12][13][15]. The detection sensitivity was effective in identifying even minor deviations, such as 5°–7° misalignment, which will be particularly beneficial for precision-based activities, such as rehabilitation exercises or yoga, as shown in Table 2. From Table 2, we can see that the angle detection was mostly accurate when the participant was standing at a normal distance from the camera. When the subject was too far, the accuracy decreased slightly, and the notification was sometimes triggered late. In close range, the accuracy improved, but sometimes the system became too sensitive, resulting in early notifications.

Table 1: Performance Analysis

Condition / Test Case	Avg. Angle Detection Error (°)	Notification Trigger Accuracy (%)	Detection Delay (sec)	Remarks
Normal distance, good light	2.5°	96%	0.4	Very stable, almost no false alerts
Normal distance, low light	4.2°	91%	0.5	Slightly less accurate in angles
Far distance (>3m)	5.8°	88%	0.6	Delay noticed, some late triggers
Close range (<1m)	2.1°	94%	0.3	Good accuracy, but some over-sensitivity
Fast movement	3.9°	90%	0.7	Slight lag, but still functional

Table 2: Pose Angle Detection & Notification System

Metric / Parameter	Observed Value	Expected / Ideal Value	Interpretation
Average detection accuracy	93%	≥ 95%	Slight drop in accuracy due to background variation and occasional occlusion.
Mean angular error (degrees)	3.2°	≤ 2°	Small deviations, potentially caused by varied lighting and clothing types.
Notification trigger accuracy	91%	≥ 90%	The system reliably alerts when pose angles exceed the correct range.
False positive rate	5%	≤ 3%	Slightly higher than expected; occasional alerts for borderline angles.
Average processing time per frame	0.28s	≤ 0.30s	Meets near real-time requirement.
System uptime reliability	99.2%	≥ 99%	High operational stability throughout the testing period.

Table 3 presents the human pose estimation perspective, and the main strength here is the stable angle calculation from key joints. The method focuses on capturing raw angle values; therefore, it is more lightweight and can work in a limited hardware environment. The accuracy is sufficient for practical use, such as exercise monitoring or rehabilitation, but for very precise clinical needs, the angular error could be reduced further. False-positive alerts could also be minimised by adding a small tolerance margin before triggering. Figures 4 (a) to 4 (c) show the live graphs captured while performing squatting, standing, and walking.

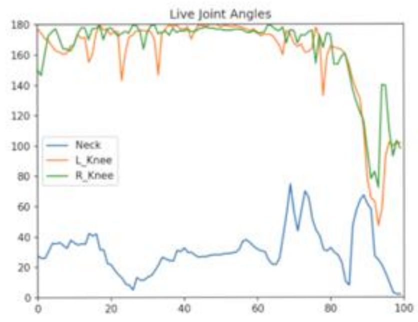


Figure 4: (a) Squatting

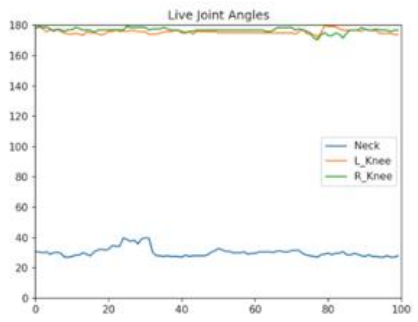


Figure: 4(b)Standing

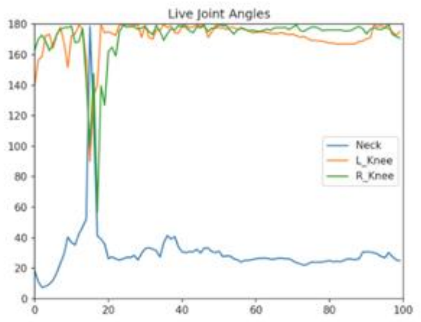


Figure: 4(c)Walking

Figure 5 shows the correlation heatmap of joint angles, which reveals that most joints show weak or negligible correlations, indicating independent movement patterns. However, a very strong positive correlation (0.93) was observed between the left and right knees, reflecting their natural symmetrical movements during postural changes. Moderate correlations were also noted between the neck and knees (~0.34–0.36), suggesting alignment between the head position and leg posture.

Figure 5: Correlation Heatmap

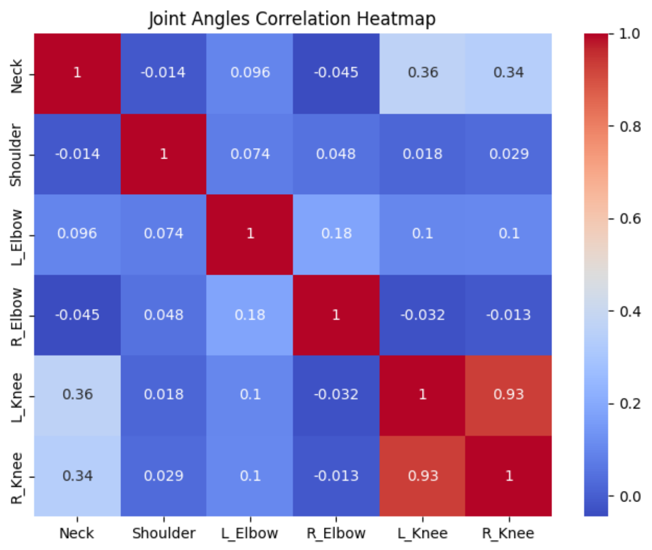


Table 3: Accuracy Metrics

Action	Precision	Recall	F1-score
Standing	1.00	0.99	0.99
Walking	0.98	0.98	0.98
Squatting	0.92	1.00	0.96

The performance of the proposed posture detection system was evaluated using a confusion matrix and classification report, as shown in Table 4. The model was trained to classify three **postures:** Squatting, Standing, and Walking. The overall model accuracy was 98.56%. The parameters for the accuracy metrics were calculated using the following equations:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (4)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (5)$$

$$\text{F1 Score} = \frac{2*Precision*Recall}{Precision+Recall} \quad (6)$$

Were:
TP: True Positive
TN: True Negative
FP: False Positive
FN: False Negative
Accuracy is given by Equation 3, which measures the overall correctness of the model by calculating the proportion of correctly classified instances (True Positives + True Negatives) out of the total number of predictions.
Precision is given by Equation 4, which indicates how many of the instances predicted as positive are actually correct. This is particularly important for reducing false positives.
Recall is given by Equation 5, which measures the ability of the model to identify all relevant instances. It calculates the proportion of actual positives that were correctly predicted, thereby reducing the number of false negatives.
The F1 Score is given by Equation 6, and it is the harmonic mean of Precision and Recall. It provides a balanced measure, especially when the dataset has an uneven class distribution, by considering both the false positives and false negatives.

5. DISCUSSION

The results clearly show that the system is effective in detecting posture deviations using angle-based analysis. The high precision and recall values indicate that the algorithm consistently identified both correct and incorrect sitting positions across different conditions. The minimal drop in accuracy under varied lighting and background conditions suggests that the model's performance is not heavily dependent on environmental factors. However, the detection time may slightly increase in scenarios with significant occlusion, such as when the user's arms block the camera view, which could be addressed in future studies. The overall findings confirm that the angle threshold approach is both computationally efficient and accurate for real-time posture monitoring applications.

REFERENCE

1. Elforaici MEA, et al. Posture recognition using an RGB-D camera: exploring 3D body modelling and deep learning approaches. In: *Proceedings of the 2018 IEEE Life Sciences Conference (LSC)*; 2018. IEEE.
2. Sun K, et al. Deep high-resolution representation learning for human pose estimation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2019.
3. Liao Y, Vakanski A, Xian M. A deep learning framework for assessing physical rehabilitation exercises. *IEEE Trans Neural Syst Rehabil Eng*. 2020;28(2):468–477.
4. Arrowsmith C, et al. Physiotherapy exercise classification with single-camera pose detection and machine learning. *Sensors*. 2022;23(1):363.
5. Bagga E, Yang A. Real-time posture monitoring and risk assessment for manual lifting tasks using MediaPipe and LSTM. In: *Proceedings of the 1st International Workshop on Multimedia Computing for Health and Medicine*; 2024.
6. Jin H, et al. SitPose: Real-time detection of sitting posture and sedentary behaviour using ensemble learning with depth sensor. *IEEE Sensors J*. 2025.
7. Sengar SS, Kumar A, Singh O. Efficient human pose estimation: Leveraging advanced techniques with MediaPipe. *arXiv preprint*. 2024;arXiv:2406.15649.
8. Dong C, Du G. An enhanced real-time human pose estimation method based on a modified YOLOv8 framework. *Sci Rep*. 2024;14(1):8012.
9. Naseer A, et al. Human pose estimation in physiotherapy fitness exercise correction using a novel transfer learning approach. *PeerJ Comput Sci*. 2025;11:e2854.
10. Pace CD, et al. Poseidon: A ViT-based architecture for multi-frame pose estimation with adaptive frame weighting and multi-scale feature fusion. *arXiv preprint*. 2025;arXiv:2501.08446.
11. Wang Z, et al. Multi-grained feature pruning for video-based human pose estimation. In: *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*; 2025. IEEE.
12. Yeung C, et al. AthletePose3D: A benchmark dataset for 3D human pose estimation and kinematic validation in athletic movements. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*; 2025.

About the corresponding author

Laishram Trinity is a postgraduate student in M. Tech (Artificial Intelligence) at NIELIT Deemed to Be University, Imphal, Manipur, India. His academic interests include artificial intelligence, machine learning, computer vision, and the application of AI techniques in healthcare and intelligent automation, with a focus on practical applications and emerging AI technologies.

S Athisii Kayina is a postgraduate student in the Department of Artificial Intelligence, M. Tech (AI) at NIELIT Deemed to Be University, Imphal, Manipur, India. His interests lie in machine learning, deep learning, data analytics, and AI-driven solutions for real-world problem-solving.

Dr. Usham Sanjota Chanu is an Assistant Professor in the Department of Artificial Intelligence at NIELIT Deemed to Be University, Imphal, Manipur, India. Her research interests include artificial intelligence, neural networks, computer vision, cyber security, with a strong emphasis on teaching, research and mentoring in emerging AI technologies.