DEVELOPMENT OF AN AI-BASED HUMAN ACTION RECOGNITION SYSTEM

BY

GOSHEN, AKACHUKWU MBAZOR
GOU/U23/CSC/1137

DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS
FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY
GODFREY OKOYE UNIVERSITY
ENUGU STATE

JUNE, 2026

TITLE PAGE

DEVELOPMENT OF AN AI-BASED HUMAN ACTION RECOGNITION SYSTEM

BY

GOSHEN, AKACHUKWU MBAZOR
GOU/U23/CSC/1137

A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS, FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY, GODFREY OKOYE UNIVERSITY, ENUGU STATE, IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF BACHELOR OF SCIENCE (B.Sc.) IN COMPUTER SCIENCE

SUPERVISOR: DR. IFEANYI KINGSLEY CHIBUEZE

JUNE, 2026

APPROVAL PAGE

This research, titled "DEVELOPMENT OF AN AI-BASED HUMAN ACTION RECOGNITION SYSTEM," has been assessed and approved by the Department of Computer Science and Mathematics Committees in the Faculty of Computing and Information Technology, Godfrey Okoye University, Enugu, Nigeria.

DR. IFEANYI KINGSLEY CHIBUEZE

	_____	_____
SUPERVISOR	**SIGNATURE**	**DATE**
EXTERNAL EXAMINER	_____	_____
	SIGNATURE	**DATE**
DR. FRANK OKEBANAMA	_____	_____
HOD, DEPARTMENT OF COMPUTER SCIENCE AND MATHEMATICS	**SIGNATURE**	**DATE**
PROF. I. A. AJAH	_____	_____
DEAN, FACIT	**SIGNATURE**	**DATE**

CERTIFICATION

I certify that I completed the research included herein and that it was primarily based on my own observations and investigation. The system implementation, codebase, evaluation results, and interface demonstrations presented in this report reflect the submitted project repository. Consequently, I will fully accept the University's decision if I am found to have violated any academic integrity policy in relation to this work.

_____

GOSHEN, AKACHUKWU MBAZOR

GOU/U23/CSC/1137

DEDICATION

This project is dedicated to my parents, Dr. David Mbazor and Mrs. Success Mbazor, whose unwavering support, guidance, sacrifices, and encouragement have been instrumental in my academic journey and personal growth. Their constant belief in my abilities, prayers, and motivation inspired me throughout this project and contributed greatly to its successful completion. I am deeply grateful for their love, dedication, and commitment to my success.

ACKNOWLEDGEMENTS

First and foremost, I give all glory and thanks to God Almighty for His grace, mercy, protection, and guidance throughout the course of this project and my entire academic journey at Godfrey Okoye University.

I sincerely appreciate my supervisor, Dr. Ifeanyi Kingsley Chibueze, for his invaluable support, constructive criticism, encouragement, and scholarly expertise, all of which contributed greatly to the successful completion of this work.

My heartfelt appreciation goes to the Head of Department, Dr. U. F. Okebanama, and all my lecturers, including Prof. N. M. Ozofor, Mrs. Ekene Okafor, and Mr. Camillus Njoku, for their knowledge, mentorship, and for maintaining a conducive learning environment throughout my studies.

I am also grateful to my friends and colleagues for their encouragement, shared experiences, and camaraderie, which made this journey both memorable and rewarding. To everyone who contributed directly or indirectly to the success of this project, I sincerely say thank you.

ABSTRACT

This project describes the design, implementation, evaluation, and deployment of ActionNet – a lightweight and real-time Human Action Recognition (HAR) system implemented without any usage of GPU hardware and deep learning. The system was implemented to solve the problem of costly GPU-powered HAR systems that can be run only on professional computers and specialized cloud-based platforms. The system was designed and implemented in Python using Streamlit, OpenCV, MediaPipe Pose, and scikit-learn libraries. The system detects 33 landmarks per frame (x, y, z, visibility), and then a 15 frames sliding window is converted into a feature vector of size 410 dimensions including means (132), standard deviations (132), ranges (132), and seven scale-normalized geometric features together with their variances (14). Four scikit-learn classification pipelines (Logistic Regression – 82.04%, Ridge Classifier – 82.04%, Random Forest – 79.84%, Gradient Boosting – 88.12%) were trained and tested on a custom data set of 7,463 pose rows of 8 action types (Arms Crossed, Bending, Clapping, Hand Raised, Jumping, Sitting, Standing, and Walking). The deployed application provides the following functionality: Video upload, live WebRTC webcam analysis, session history analysis, and confidence threshold tuning.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Automatic analysis of human actions based on videos has emerged as an interesting line of research in computer vision and machine learning. The concept of Human Action Recognition (HAR) entails the computational analysis and labeling of the physical activities undertaken by an individual or group of individuals that are captured through a video recording or an array of images (Sedaghati et al., 2025). With the proliferation of digital cameras, mobile phones, and surveillance technology, the amount of video data generated on a day-to-day basis has reached levels at which manual evaluation becomes unfeasible (Global Media Insight, 2026).

Classification must be performed systematically in HAR systems with constraints, of which the most critical one is computation since the system needs to achieve precise and fast results on affordable hardware. The development of deep learning technology solved many problems in the early stages of HAR systems by using discriminative features discovered through deep neural networks automatically (Zhang et al., 2022). High-performing deep learning models such as 3D CNNs and transformer video models require GPUs that cannot be found in edge devices, basic laptops, and browser-based platforms (Siddiqui et al., 2025).

The solution of pose based action recognition provides a promising approach. Instead of using full pixel data, pose-based methods extract the position of body landmarks and use the configuration and motion of landmarks to generate the label for actions (Song et al., 2021). There exist recent works, especially MediaPipe Pose developed by Google, that have provided accurate extraction of body landmarks at inference speed suitable for CPU in addition to 33 keypoints containing x, y, z

and visibility coordinates of each landmark, thus providing a viable means of light weight HAR (Grishchenko and Bazarevsky, 2020). This project implements ActionNet, which is an AI based Human Action Recognition system integrating landmark extraction through MediaPipe Pose with traditional scikit-learn classifiers trained on a webcam dataset of eight classes of actions and deploying the system as a web application.

1.2 Statement of the Problem

The current HAR solutions raise major issues for practical implementation. These are listed below:

1. Modern Deep Learning techniques necessitate GPU-based computing and heavy runtime like TensorFlow or PyTorch, hence are impractical to deploy onto edge devices, single-board computers, or browser-based platforms (Siddiqui et al., 2025; Freire-Obregon et al., 2026).
2. Popular benchmarking datasets like UCF50 consist of sophisticated or specific action types that fail to match the necessity of recognizing simple actions in real-time.
3. Most existing HAR solutions lack comparison between different classification pipelines using the same pose-based features to help users make informed decisions about practical implementations.
4. Almost none of the HAR solutions show the ability to implement the entire process in a browser-based web application that performs real-time classification while providing session-based analytics without keeping any biometric data.

To solve the stated issues, this research utilizes MediaPipe Pose to get joint keypoints from the live camera stream or uploaded video, accumulates temporal windows of 15 frames, engineers a 410 dimensional feature vector and trains light-weight scikit-learn classifiers for 8 action classes classification.

1.3 Aim and Objectives of the Study

The aim of this study is to develop an AI-Based Human Action Recognition System.

To achieve the stated aim, the following specific objectives were pursued:

- Collect a labelled dataset of eight action classes using webcam recordings, extracting 1,662-dimensional MediaPipe Holistic landmark vectors per frame and storing them in a structured CSV format.
- Design and engineer fixed-length statistical feature vectors from 30-frame sequences by computing mean, standard deviation, maximum, and minimum across the temporal axis, supplemented by joint angle and wrist-distance geometric descriptors.
- Develop and train four scikit-learn classification pipelines — Random Forest, Logistic Regression, Gradient Boosting, and Ridge Classifier — on the aggregated feature representations.
- Implement the selected model within a Streamlit web application performing real-time inference from a live webcam feed using a rolling 30-frame buffer, serialising the trained pipeline for deployment.
- Evaluate each pipeline using accuracy, precision, recall, F1-score, and confusion matrix analysis, selecting the best-performing model by macro-averaged F1-score, and assess the

system's inference latency, memory footprint, and usability relative to deep learning alternatives.

1.4 Significance of the Study

This research paper shows that with explainable pose features and standard classifiers, one can achieve real-time Human Action Recognition without using a GPU and deep learning algorithms. ActionNet is important due to several reasons. First of all, it creates reproducible template which can be used for further learning and testing, it helps learning professionals choose multi-classifier for specific tasks basing on empirical evidence, and it shows an actual way of deploying such application through browser using Streamlit and WebRTC, which is possible without any special equipment and software installation.

1.5 Scope of the Study

The presented codebase recognizes eight action categories including Arms Crossed, Bending, Clapping, Hand Raised, Jumping, Sitting, Standing and Walking. The application works with mp4, avi and mov videos and webcam streams through WebRTC. The feature engineering uses a sliding window of 15 frames which generates a 410-dimensional vector. Four classifiers have been trained and tested using chronological split of 80% and 20%. The application does not contain database functionality, action history is stored in Streamlit session state. The deployed model artifact is called Human_actionV1.3.pkl. This is a Random Forest pipeline.

1.6 Limitation of the Study

Since the system makes use of human body landmarks, factors such as occlusions, inadequate lighting, peculiar camera angle, or partial body view may affect its performance. The dataset used in

the project has been sourced from a single individual, thus limiting the generalizability of the system to bodies with varying proportions and action styles. The live-stream branch in app.py has a slight difference in the geometric feature engineering of the custom geometric features compared to the uploading/uploaded/training branch. The model artefact (Random Forest, Human_actionV1.3.pkl, 79.84%) that has been deployed is inferior to the model from the best evaluated notebook (Gradient Boosting, 88.12%).

1.7 Operational Definition of Terms

1. Human Action Recognition (HAR): Computational algorithm for automatic detection and classification of human actions in images and video sequences.
2. MediaPipe Pose: It is a Google framework that is used to extract 33 body pose landmarks from one RGB image frame giving x, y, z coordinates and visibility of each landmark at inference speed using CPUs.
3. Temporal Feature Engineering: Process of summarizing a series of frames into fixed length feature vector by means of statistical aggregation along time axis such as mean, standard deviation, range, etc.
4. Scikit-Learn Pipeline: Serially composed sequence of a Normaliser using the StandardScaler method, followed by any classifier from the scikit-learn library, stored as a single joblib file.
5. WebRTC: Web Real-Time Communication, a browser-based protocol for encrypted peer-to-peer video communication, used by the streamlit-webrtc package for inference via live webcam.
6. Rolling Buffer: A queue (deque) of the last N frames kept in memory to allow windowing of the sequence in real time.

7. ActionNet: The name of the Human Action Recognition model built in this project, which consists of the data collection pipeline, training notebook, and Streamlit inference application.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter provides an extensive review of existing literature pertaining to the development of the proposed lightweight Human Action Recognition System. It is intended to set the ground for theoretical and empirical basis of the research work, show familiarity with existing work in the area, and highlight areas that call for further research to justify this particular research. Through a critical review of existing academic articles, papers, and technical reports, this chapter will analyze the merits and demerits of existing solutions.

2.2 Theoretical Background

The area of Human Action Recognition can be considered as the subdomain of computer vision and artificial intelligence that aims to develop methods for machine recognition and classification of people's actions from visual data (video/image sequences) (Sedaghati et al., 2025). Thus, the key issue in this domain can be described as the development of the way to bridge the semantic gap between low-level features (e.g., pixels, edges, intensity changes) and high-level action semantics (e.g., walking, sitting, gesturing). The next subsections will provide an overview of the key theoretical domains underpinning the project.

2.2.1 Human Action Recognition

Initially, HAR was performed by utilizing hand-crafted descriptors (e.g., Optical Flow, HOG, STIP, Dense Trajectories) in combination with standard classifiers (e.g., SVM, Hidden Markov Model). With the arrival of deep learning in 2014, researchers were able to design end-to-end feature extractors

based on two-stream networks, 3D convolutions, and Transformers (Saoudi et al., 2023; Siddiqui et al., 2025). Unfortunately, due to the complexity of these models, the computational overhead remains the main issue.

2.2.2 Pose-Based Recognition and MediaPipe Pose

Pose-based models model the configuration of the human body through skeletal landmarks and coordinate space rather than raw pixel values. The model representation is compact, readable and robust to variations in appearance such as background texture and lighting conditions (Song et al., 2021). MediaPipe Pose (Grishchenko and Bazarevsky, 2020) detects 33 body keypoints from a single RGB frame based on the lightweight convolutional backbone of BlazePose (Bazarevsky et al., 2020), which offers 132 raw numbers (x, y, z, visibility of each landmark) to describe the body posture of full-body posture, suitable for pose-based and motion-based actions class at CPU-speed inference.

2.2.3 Classical Machine Learning Models for HAR

Random Forest, is an ensemble model which decorates decision trees through bootstrapping and random feature sets, which produces good generalization performance on high-dimensional feature space. Gradient Boosting, which is an additive ensemble model of shallow decision tree trained incrementally to predict the residual errors. The theoretical foundations for Logistic Regression and Ridge Classifier are provided by Bishop (2006) as linear models.

2.2.4 Statistical Feature Engineering

The statistical aggregation of landmark-based data across fixed-length windows is a proven feature engineering technique in HAR tasks. The computation of mean, standard deviation, and range across fixed-length windows represents both the average posture and movement dynamics in one fixed-length

vector (Zhang et al., 2022). Thus, this approach obviates the use of sequence-dependent neural architectures like LSTMs, allowing more straightforward yet sufficiently accurate classifiers. Additional geometric features that can increase discriminability between similar postural classes include the wrist distance, ankle distance, and torso-based ones (Song et al., 2021).

2.2.5 Deployment Technologies

Streamlit is a commonly used solution for deployment of machine learning models as web applications available through browsers without the need to have front-end development skills (Streamlit, 2019). With the help of the streamlit-webrtc library, this can be extended to real-time applications based on webcam feed via WebRTC peer connection. Plotly is a package for in-browser visualisation of session statistics and confidence distribution.

2.3 Review of Related Literature

The study of Sedaghati et al. (2025) carried out a thorough analysis of applications of Human Action Recognition in the healthcare industry, surveillance, sport statistics, and human-computer interaction. The research presented a systematic account of the development of HAR techniques, from the initial feature-based approaches to current deep-learning architectures, highlighting the continuous improvement of benchmark performance of more sophisticated models. Although deep learning algorithms were found superior in their performances at all standard benchmarks, the authors observed the consistent deficiency in the area of edge deployment of CPU-only systems for performing HAR without any GPUs. In particular, the authors highlighted the unresolved problem of deployment of HAR systems in conditions of limited resources, in community health centres, in educational establishments, and other cases. This finding serves as a direct rationale for the choice of the approach pursued in this project.

According to Siddiqui et al. (2025), a mixed architecture of Convolutional Restricted Boltzmann Machines (Conv-RBM) in conjunction with Long Short-Term Memory (LSTM) was suggested, together with an optimized frame selection method intended to eliminate redundancy during video processing. This algorithm selected only those frames which had the highest informativeness in each sequence and hence decreased the amount of processed data with no proportionate decrease in the accuracy of recognition. The efficiency in terms of reduced resource needs compared to simple processing of each frame was achieved. However, the framework relied on TensorFlow and GPU acceleration to train and perform inferences, which makes it not applicable for the current research due to its dependence on CPU.

A long-term person re-identification model spanning years of time was designed by Freire-Obregon et al. (2026). The system combined HAR and gait dynamics and provided consistent identification throughout the years, despite changing conditions in appearance. In the study, Freire-Obregon et al. (2026) show that the HAR-based motion descriptors contain specific discriminative person information beyond action labels. They highlight the importance of the feature representation of structured actions. The model shows high results of person re-identification at multi-year time intervals in the controlled datasets. But the architecture of the system is not applicable for use in real time on CPUs. But regardless of this drawback, their results show the importance of HAR feature representations from the theoretical point of view, and this project uses them in practice for a specific goal of action recognition.

In Zhang et al. (2022), there is an extensive literature survey on various deep learning approaches to Human Action Recognition from wearables, including convolutional, recurrent, and hybrid neural networks used on multiple types of datasets with multiple sensors. An interesting result found by Zhang et al. in their literature survey is that simple statistical aggregation of features from windowed

time-series (e.g., computing the mean, standard deviation, and other descriptors for each fixed-length window in the time series) performs comparably well with sequential deep learning models like LSTMs and GRUs on structured, low-noise sensor data. Zhang et al. explained the reason behind such high performance of statistical aggregation on such tasks as its ability to encode not only the average value of the motion in a window but also its variability, thus capturing all important temporal information about actions in the window. This result gives direct theoretical motivation for the choice of the approach to feature engineering used in the present project, i.e., calculating mean, standard deviation, maximum, and minimum from 30-frame windows of MediaPipe Holistic landmarks.

Krishnan et al. (2024) have been able to reliably track and detect multiple key points in an image, despite the occlusions and variation in human sizes in the scene, using Detectron2. This system used a pretrained model that had a key point detection backbone and showed outstanding results on various benchmark data for multi-person pose. However, because of the heavy compute requirements of this pipeline and the necessity for GPU hardware with CUDA support to run at a reasonable frame rate, this system could not be deployed in CPU-only systems. The authors did not consider edge deployment and browser access situations. Nevertheless, the fact that this method was successfully applied to multi-person pose estimation and analysis proves the usefulness of skeleton-based methods for this purpose.

Song et al. (2021) gave a comprehensive overview of pose estimation techniques and their use in action recognition, with particular attention paid to the benefits and limitations of top-down and bottom-up pose estimation techniques with regard to accuracy, speed, and multiple person pose estimation. The finding from Song et al.'s survey that is most directly related to the current work was that the addition of geometric information such as joint angles obtained from adjacent landmark triplet distances as well as wrist distance significantly improved the ability to discriminate between posturally

similar actions that varied mainly in their limb geometries but not in overall body position. The example classes of Arm Crossed, Clapping, and Hand Raised demonstrate such a situation where the same postural class differs only in limb geometry. This finding is embodied in the feature extraction pipeline used in the current project through the addition of joint angle and wrist distance to statistical features.

The authors Saoudi et al. (2023) designed a novel hybrid action recognition architecture that used the combination of spatial attention modules along with LSTM Networks and 3D Convolutional Neural Network (CNN) modules. This hybrid model could perform selective attention of the most discriminative spatial regions within the frame as well as modeling the temporal dependencies between frames in the sequence. In the case of background clutter and changing lighting conditions, this architecture showed an improvement in performance when compared to conventional CNN or LSTM architectures, showing the effectiveness of using the attention guided spatial weightings for the action recognition in the real world video sequences. The model was trained and tested on standard benchmarks and showed competitive accuracy values. Nevertheless, because of the complexity of the architecture that consists of several GPU-based modules like 3D CNN and attention module, and because of its dependency on deep learning frameworks, this architecture cannot be used in our project due to its fundamental incompatibility with the CPU-only environment.

A real-time Human Action Recognition system was constructed by Singh et al. (2022) based on MediaPipe pose landmark detection, which worked in real-time on consumer-level CPUs without GPU assistance. This system extracted the landmarks from the input of a regular webcam, performed feature extraction from the sequence of landmarks, and generated actions via a classification algorithm. The study thus provided practical evidence for the feasibility of using MediaPipe to extract landmarks in real-time at speeds appropriate for consumer CPUs, and therefore for the feasibility of using landmarks

for real-time Human Action Recognition on the action vocabularies needed in practical situations. It confirmed, in other words, the direct relevance of the previous research for the current project. At the same time, an important drawback of their approach was the examination of just one classification algorithm, leaving an unanswered question of which kind of classifier would be most efficient when dealing with landmark features. Comparison of four classifiers from scikit-learn – Random Forest, Logistic Regression, Gradient Boosting and Ridge Classifier – was among the explicit goals of the current study.

Grishchenko & Bazarevsky (2020) have discussed the MediaPipe Holistic model and the associated BlazePose backbone architecture that together perform joint estimation of face, hands, and pose landmarks from a single RGB image in real-time. In total, the MediaPipe Holistic model produces an output vector consisting of 543 landmarks from face, left hand, right hand, and pose components where each landmark is made up of x, y, z, and visibility coordinates and thus results in a 1,662-dimensional raw feature vector from all four components combined. The BlazePose backbone model was designed specially for CPU-based inference on-device, and it processes images in real-time due to a light two-stage detection and tracking pipeline architecture. This technical information provides the core basis for the landmark extraction part of this project, which uses the MediaPipe Holistic model as its kepoint extraction engine owing to its architecture suited to CPU-based real-time inference.

In Bazarevsky et al. (2020), the authors designed BlazePose, the lightweight convolutional neural network backbone which is used by MediaPipe to perform body pose estimation. The architecture was designed specifically for real-time body pose tracking on edge devices using the two-stage detection-tracking pipeline and performing accurate full-body keypoint estimation without any additional computational costs in comparison with generic object detection architectures. BlazePose detects 33 keypoints of the body with sub-pixel accuracy in real time even on CPUs by exploiting temporal

dependencies between the frames and skipping redundant detections. BlazePose provided the capability to perform accurate full-body keypoint extraction without GPUs on generic mobile CPUs becoming the first publicly available pose estimation model that provides accuracy and latency required for real-time CPU-only body tracking. The latter is why the current research can perform real-time HAR without specialized GPUs because MediaPipe Holistic uses BlazePose as the body pose backbone model.

Neili and Essoukri Ben Amara (2019) reviewed Human Action Recognition methods using pose information in an exhaustive manner, considering the diversity of skeletal representations used, the variety of feature extraction methods employed, and the classification methods utilized in tackling this problem until then. One of the findings of this study is that skeletal representations – especially those capturing joint positions, joint angles, and relative spatial locations of the body – have inherent robustness properties over appearance-based representations when used in varying lighting conditions, background clutter, and changing subject appearance due to changes in clothes and body size. The reason for the above observation is the abstract nature of skeletal representations as compared to the raw pixel representations. In other words, two subjects executing the same action but having different appearances while retaining the kinematic aspect of their movement would yield similar skeletal sequences, but an appearance-based representation may not generalize in such scenarios. While this study predated the MediaPipe era and hence worked on coarse skeletal representations than the current work, it still helps set the ground for the current work.

2.4 Summary of Literature Review

Table 2.1: Summary of Literature Review

S/N	Author(s) & Year	Title/Work	Methodology	Findings	Limitations
1	Sedaghati et al. (2025)	Application of HAR: A review	Systematic review	Deep-learning dominance; edge-deployment gaps identified	No CPU-only deployable HAR system
2	Siddiqui et al. (2025)	HAR using Conv-RBM and LSTM	Conv-RBM + LSTM	Reduced processing via selective sampling	GPU dependency; not edge-deployable
3	Freire-Obregon et al. (2026)	Multi-year long-term re-identification	HAR feature fusion	Effective temporal re-identification	Not real-time; CPU constraints unaddressed
4	Zhang et al. (2022)	Deep learning in wearable HAR: review	Literature review	Statistical aggregation competitive for structured inputs	Limited classical vs. deep comparison
5	Krishnan et al. (2024)	Multiple object detection (Detectron2)	Detectron2 framework	Reliable keypoints in multi-person scenes	No lightweight CPU-only alternative
6	Song et al. (2021)	Pose estimation & action recognition survey	Survey + comparison	Deep pose estimators improve reliability	Limited classical ML on pose features
7	Saoudi et al. (2023)	Hybrid attention LSTM and 3D CNN	Attention+LSTM+3D CNN	Improved under clutter and illumination variation	GPU dependency; not edge-deployable
8	Singh et al. (2022)	Real-time HAR using MediaPipe	MediaPipe + single classifier	High CPU-speed accuracy on limited vocabulary	No multi-classifier comparison

S/N	Author(s) & Year	Title/Work	Methodology	Findings	Limitations
9	Grishchenko & Bazarevsky (2020)	MediaPipe Pose prediction	BlazePose CNN backbone	On-device CPU landmark inference	No classifier evaluation
10	Bazarevsky et al. (2020)	BlazePose: On-device pose tracking	Lightweight CNN	Real-time tracking on mobile/CPU	Not evaluated for action classification
11	Neili & Ben Amara (2019)	Pose-based HAR: A review	Literature review	Skeletal reps robust to appearance variation	Pre-2020; no browser-deployable system

2.5 Research Gap

Literature in HAR shows significant advancements made in this domain. Nevertheless, there are three distinct research gaps that can be observed. Firstly, most pose-based HAR studies make use of one particular classifier without comparing its performance with other classifiers under the same conditions (that is, on the same features representation). Secondly, existing benchmark datasets like UCF50 cannot be used to demonstrate basic actions online, and therefore, it was necessary to collect data on your own, as was done in this study. Lastly, only very few studies show full deployment of their HAR method on a browser-based interface which gives confidence scoring and session-based analysis. This study covers all three gaps mentioned above.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.0 Introduction

The present chapter gives an extensive analysis and design of the proposed ActionNet Human Action Recognition system. The analysis of the current systems and their limitations, objectives and requirements of the proposed system, the design process, and four UML diagrams showing the working of the proposed system will be explained in the following sections. A detailed design of the system including the input/output, database and architecture will also be shown.

3.1 System Analysis

3.1.1 Analysis of the Current System

The current approaches to Human Action Recognition can be broadly classified into three types: Manual observation based systems, rule-based computer vision systems, and deep learning based systems. The manual observation based method is time consuming since it requires constant human observation and is prone to the problems of fatigue and subjectivity. The rule based computer vision systems based on background subtraction and optical flow thresholding are lightweight and easy to use but are brittle to changes in lighting conditions and camera angles. The deep learning based systems exemplified by Two-Stream CNNs (Simonyan and Zisserman, 2014), CNN-LSTM models (Donahue et al., 2015), and Attention-based models (Saoudi et al., 2023) give excellent results on benchmark datasets but are computationally intensive.

3.1.2 Limitations of the Existing System

- The existing HAR system analysis shows there are five main issues that our system aims to address:
- Dependency on GPU hardware: current advanced systems demand a specialized GPU device for inference in real time and therefore cannot be used on consumer laptops and mobile devices without GPU support.
- Large framework overhead: TensorFlow and PyTorch installations require about 300-500 megabytes of storage space and also have dependency chains that make deployment difficult on constrained or shared platforms.
- No availability via web browsers: existing advanced HAR systems cannot be deployed as web applications accessible through web browsers without a server-side GPU setup.
- Privacy threats: pipelines that involve storing and processing of raw frames can cause privacy threats, especially in sensitive areas such as medical diagnostics and education.
- Unfeasible benchmarks: benchmark datasets like UCF50 are filled with actions which are complicated and/or specialized rather than everyday actions.

3.1.3 Analysis of the Proposed System

The suggested ActionNet system is a Human Action Recognition application which can recognise actions from eight classes based on video files or webcam feed in real-time mode using solely CPU. In the architecture of the ActionNet, convolutional layers are replaced with MediaPipe Pose landmarks recognition (33 landmarks × 4 values = 132 raw features), temporal statistics computation over 15-frame intervals, and seven original geometric features, resulting in 410D feature vector fed into scikit-learn classical classifiers. The proposed approach allows achieving a small model size (5-30 MB), very fast inference (<10 ms per prediction) and low RAM usage (<200 MB).

3.2 System Requirements Specification (SRS)

3.2.1 Functional Requirements

Table 3.1: Functional Requirements

Requirement	Codebase Evidence
Upload video for analysis	Streamlit file_uploader accepts mp4, avi, and mov files.
Live webcam inference	streamlit-webrtc with ActionProcessor class for real-time processing.
Pose extraction	MediaPipe Pose extracts 33 landmarks per frame (x, y, z, visibility).
410-feature vector production	Temporal means (132), std (132), ranges (132), geometric features (14).
Prediction output	UI displays action label, confidence score, and processing FPS.
History analytics	Session history visualised with Plotly bar and line charts.
Threshold control	The settings page exposes confidence_threshold slider.
Unknown action suppression	Prediction becomes ‘Unknown (Low Conf)’ below confidence threshold.

3.2.2 Non-Functional Requirements

Table 3.2: Non-Functional Requirements

Requirement	Implementation Response
Performance	End-to-end prediction latency below 50ms on consumer CPU hardware; classifier inference under 10ms per prediction.
Usability	Sidebar navigation separates Upload Video, Live Stream, History, About, and Settings views.

Requirement	Implementation Response
Privacy	Raw video frames not stored to disk; only numerical landmark coordinates used for inference; WebRTC encryption for live stream.
Portability	Dependencies listed in requirements.txt and packages.txt; no GPU required; deployable on Python 3.10+ environments.
Maintainability	Core frame processing isolated in process_frame for uploaded video and ActionProcessor.recv for live stream.

3.3 System Design Methodology

For the methodology to be used in the current project, an experimental and implementation-based methodology will be used. The methodology will entail a five-step process of (1) data collection and landmark detection, (2) feature generation and aggregation of sequences, (3) training the models through four different classification pipelines, (4) evaluating and serializing the models, and (5) performing real-time inference and deployment of the model. It was chosen because in the current project, all the requirements are well-defined, meaning that a linear process will be suitable.

3.3.1 Rationale for Methodology

An experimental and implementation-based methodology was chosen because the project entails developing a new system from scratch whereby there are well-defined inputs, outputs, and performance expectations. This five-step sequential process allows the components to be verified before integrating with another. The methodology is in line with what has been done before in applied HAR research (Singh et al., 2022; Song et al., 2021).

3.3.2 Method of Data Collection

Custom dataset was created by capturing webcam videos of eight action categories in indoor controlled environment. The MediaPipe Pose extracted landmark vectors (with 33 landmarks for each frame, 132 values) which were then saved in row-wise format in CSV. The dataset had 7,463 rows of raw poses before temporal windowing. Using a sliding temporal window of size 15 frames for each action category resulted in 7,051 temporal windows. Using chronological 80/20 split per action category resulted in 1,414 test samples. On-screen countdown prompts during video capture made sure labeling was consistent.

3.4 System Modelling using UML

3.4.1 Use Case Diagram

Figure 3.1 depicts the Use Case Diagram of ActionNet HAR System. The diagram depicts interaction between the User actor and six use cases: Upload Video, Start Live Stream, View History Analytics, Set Confidence Threshold, Clear History, and View Prediction. Use cases that are involved in the inference pipeline including Extract Pose Landmarks, Classify Action, and Apply Threshold get included automatically upon input of video data. The «extend» relation denotes the optional case where there is no person detected in the current frame.

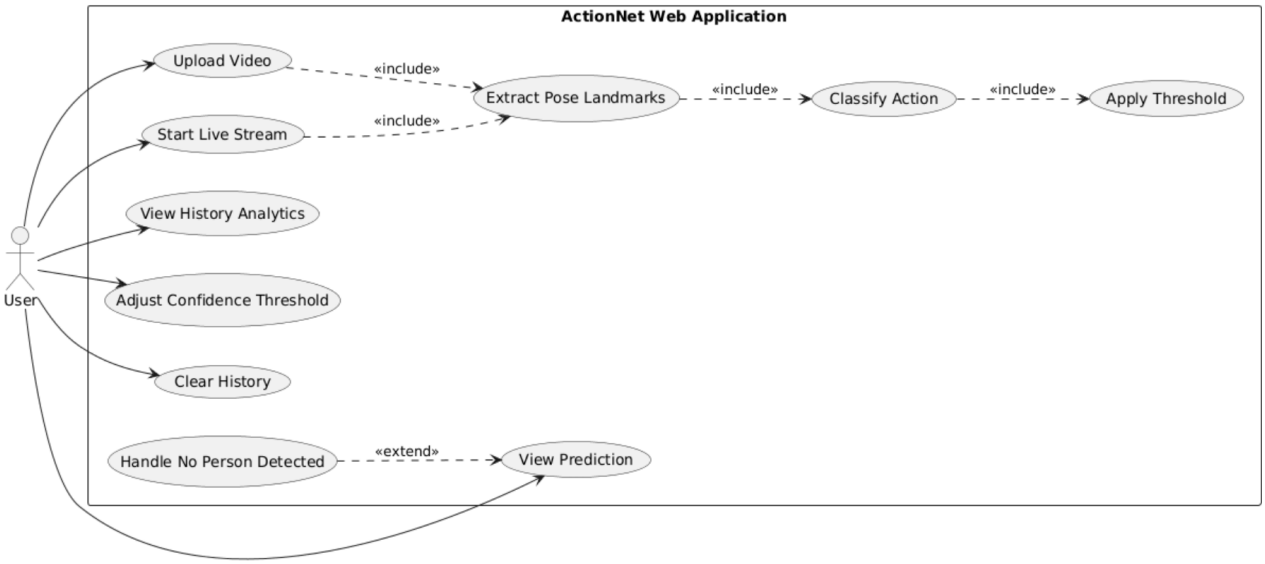

Figure 3.1: Use Case Diagram of the ActionNet HAR System

3.4.2 Activity Diagram

Figure 3.2 shows the Activity Diagram of the ActionNet HAR processing flow. Beginning from the stage where the application has been opened and the model has been loaded, the diagram shows the process taken for deciding on selection of the video source. For every single frame that has been captured, MediaPipe Pose will try extracting landmarks. If a person has been detected, then features will be generated, action will be predicted, and the output will be shown as well as stored in session history. If not, then ‘No Person Detected’ will be shown.

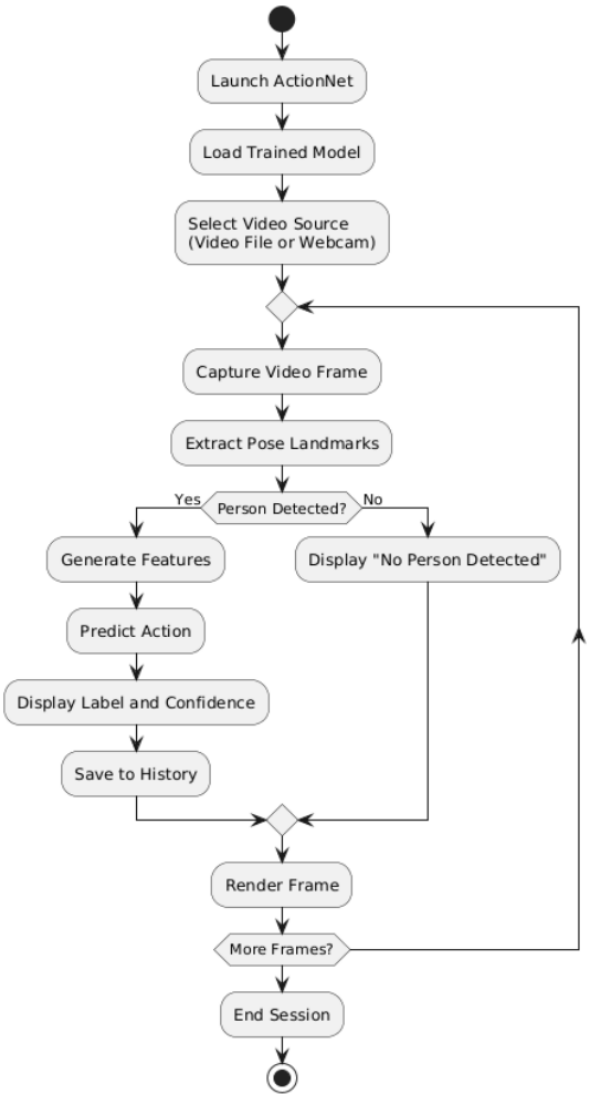

Figure 3.2: Activity Diagram of the ActionNet HAR Processing Flow

3.4.3 Class Diagram

Figure 3.3 shows the Class Diagram of the ActionNet HAR System. The diagram describes six classes of the system along with their attributes and methods: PoseExtractor (for MediaPipe Pose processing and zero-padding in case of missing landmarks), FeatureEngineer (for temporal statistics-based and geometrical feature extraction on 15 frames), RollingBuffer (a fixed-sized deque of frames), ClassificationPipeline (StandardScaler and the classifier used), ActionProcessor (for coordination

among the other modules for frame-by-frame inference and session recording), and StreamlitApp (user interface manager and event routing).

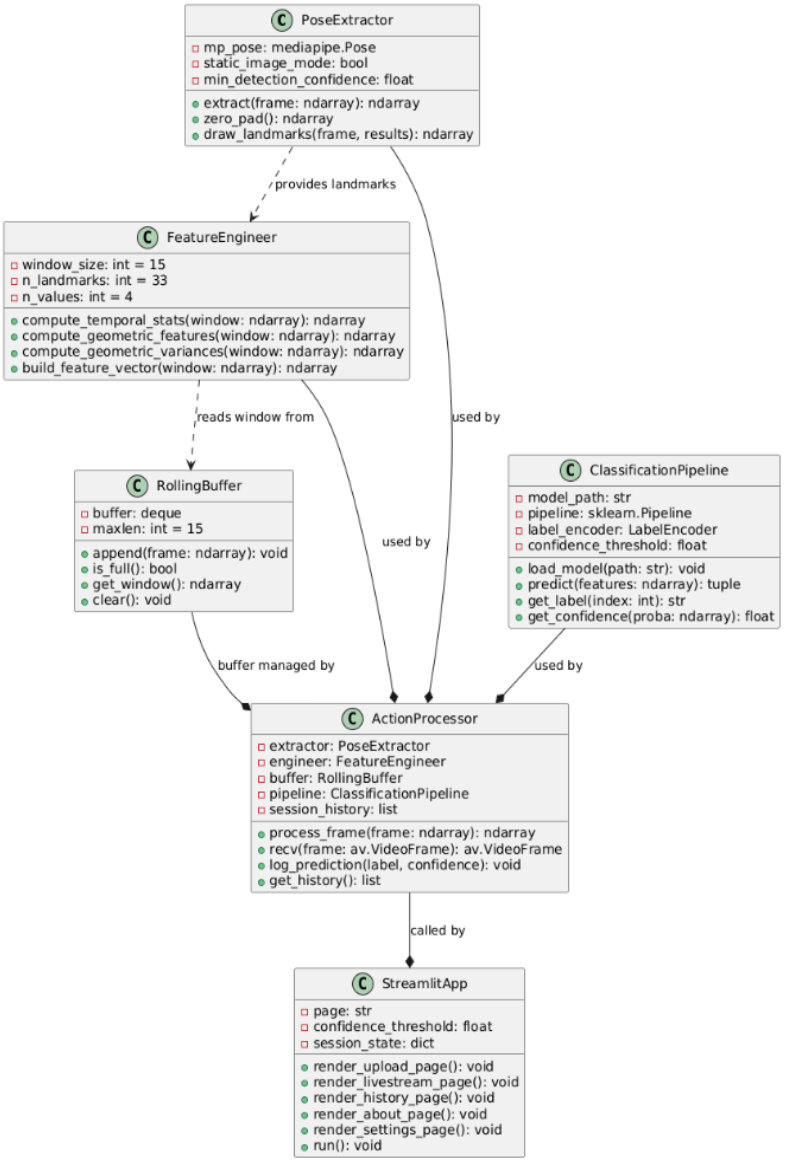

Figure 3.3: Class Diagram of the ActionNet HAR System

3.4.4 Sequence Diagram

Figure 3.4 shows the Sequence Diagram for the workflow of Human Action Recognition. This diagram depicts the message flow from User to Streamlit UI, OpenCV, MediaPipe Pose, Feature

Engine, ML Model, and History Logger in one full cycle of inference. The ‘loop’ box denotes the process done per frame; the ‘alt’ box inside denotes the alternatives when there is a person or not detected; finally, after the loop, the history interaction is also demonstrated in this diagram.

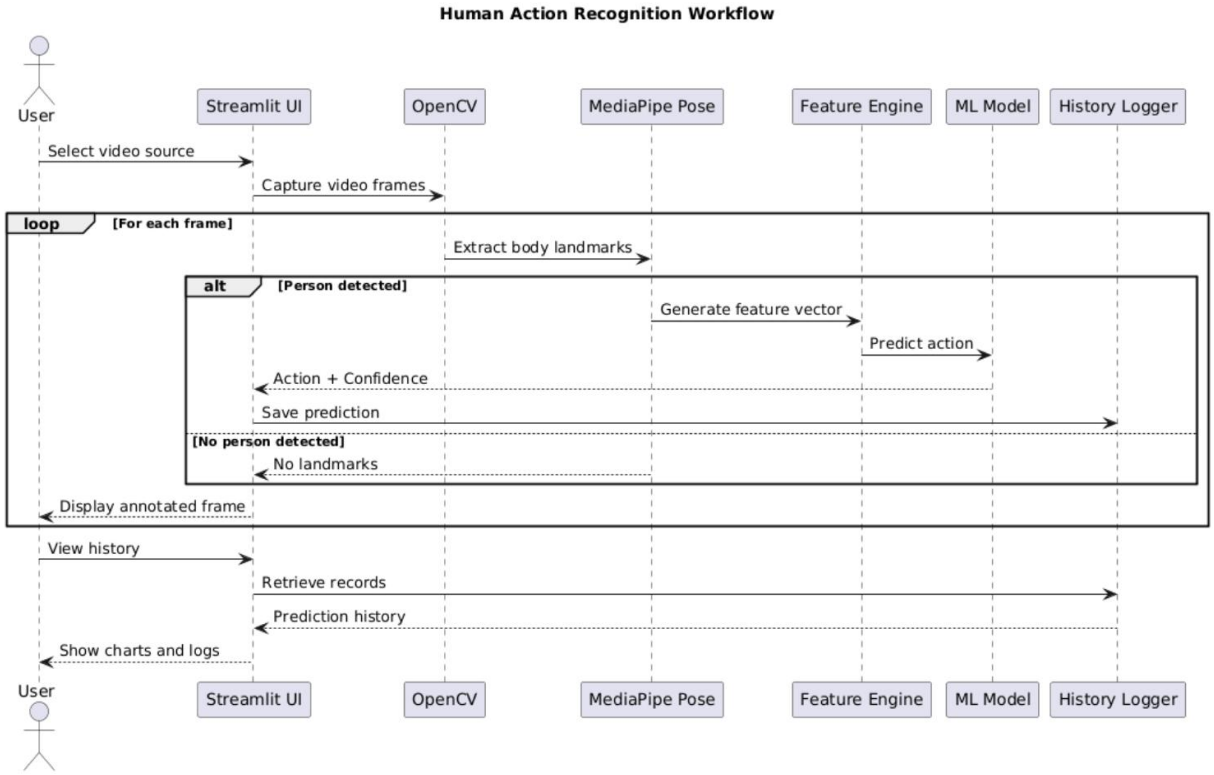

Figure 3.4: Sequence Diagram — HAR Workflow

3.5 System Design

3.5.1 Input Design

The key system inputs are as follows: (1) the uploaded video file in one of the supported formats (mp4, avi, or mov) through Streamlit file uploader in the Upload Video view; and (2) the video stream captured from the webcam using streamlit-webrtc in the Live Stream view. In the process of collecting the data, the system also receives the string input label that denotes the action class that needs to be

recorded. Input validation is used to ensure only supported video formats are accepted and webcam access is given explicitly through the WebRTC permission dialog in the browser.

3.5.2 Output Design

The key system output is the prediction of the action class rendered in high-contrast text overlay on the live or uploaded video stream in the Streamlit web application interface along with the confidence score and FPS indication. In the process of off-line training of the model, the system also produces the serialised model pickle files and the classification report containing precision, recall, F1-score and confusion matrix heatmaps.

3.5.3 Database Design

The system does not use a relational database management system. In the data gathering step, the collected data is saved in a structured CSV flat-file where each row contains 132 landmarks (each landmark containing four values), action class name, and the sequence number. When inferring, action history at session level is stored in Streamlit session state (Python dictionary) and not saved on the hard drive.

Table 3.4: Database Schema — Landmark Dataset (CSV Structure)

Field Name	Data Type	Example Value	Description
sequence_id	Integer	001	Unique identifier for each 15-frame recording sequence
action_label	String	“Bending”	Action class label for the recorded sequence
frame_number	Integer	7	Frame index within the sequence (0–14)

Field Name	Data Type	Example Value	Description
lm_0_x … lm_131	Float	0.5423	132 normalised landmark coordinate values from MediaPipe Pose (33 landmarks × 4 values)

3.5.4 System Architecture Design

The System Architecture of ActionNet is shown in Figure 3.5 as a five-tier structure, which includes the following tiers: (1) Input Tier — Video Stream from Webcam or an uploaded video; (2) Presentation Tier – Streamlit GUI for user interaction and visualisation; (3) Data Processing Tier – Frame Extraction with OpenCV, MediaPipe Pose (33 landmarks), and Feature Extraction Module (410 dimensional feature vector); (4) Intelligence Tier – Trained machine learning model (Human_actionV1.3.pkl); and (5) Output Tier – Recognition results (Predicted action, confidence score, activity logs).

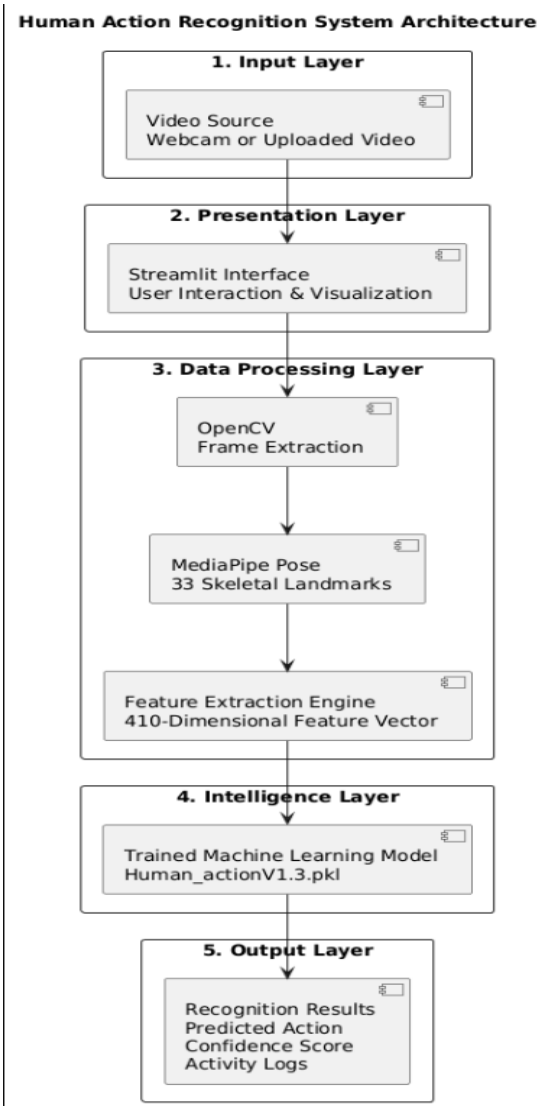

Figure 3.5: System Architecture Diagram — ActionNet (5-Layer)

Figure 3.6 presents the HAR Processing Pipeline as a horizontal flow: Video / Webcam → Frame Resize → MediaPipe Pose → 15-Frame Buffer → 410 Features → Model Prediction → Action + Confidence.

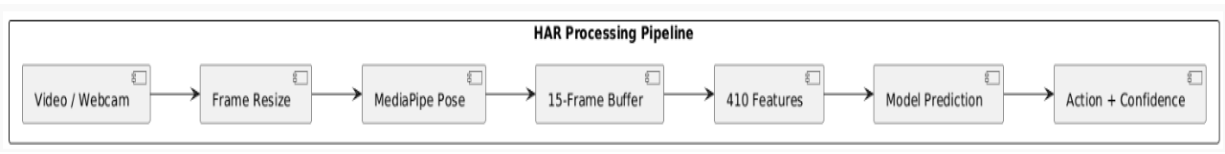

Figure 3.6: HAR Processing Pipeline

Table 3.3: Actual Feature Vector Composition

Feature Group	Count	Source / Description
Temporal mean	132	Mean of 33 pose landmarks × 4 values (x, y, z, vis) across 15 frames
Temporal standard deviation	132	Motion variability per landmark value across the 15-frame window
Temporal range	132	Peak-to-peak (max − min) landmark movement across 15 frames
Custom geometric means	7	Wrist distance, ankle distance, hand positions, torso metrics, hip position
Custom geometric variances	7	Variance of the seven custom geometric features across the window
Total	410	Combined 410-dimensional model input vector

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.0 Introduction

Chapter 4 discusses the implementation of the proposed ActionNet Human Action Recognition system including development environment, dataset, training and evaluating the model, integrating the modules of the system, testing and evaluating procedures, as well as the results presentation and discussion.

4.1 Development Environment and Tools

4.1.1 Programming Language and Frameworks

The ActionNet Human Action Recognition system was completely developed using Python 3.10 because of its rich set of machine learning and computer vision frameworks. The complete set of technologies used in the development process is summarized in Table 4.1.

Table 4.1: Technology Stack and Libraries Used

Tool / Library	Role in Codebase
Python 3.10	Core programming environment
Streamlit	Browser-accessible web application user interface
streamlit-webrtc	Real-time encrypted webcam streaming via WebRTC peer connection
OpenCV (cv2)	Video capture, frame resizing, landmark overlay drawing
MediaPipe	Pose landmark extraction — 33 body keypoints per frame
NumPy / Pandas	Feature vector computation and DataFrame preparation
scikit-learn	Model pipelines, StandardScaler, classifiers, evaluation utilities

Tool / Library	Role in Codebase
Plotly	Interactive history and confidence visualisation charts
Joblib	Model serialisation and loading (.pkl artifacts)
Twilio	Optional TURN server token for WebRTC in restricted networks

4.1.2 Development Environment and Hardware Requirements

The system was designed and evaluated on Windows 11 using Python 3.10 with venv, 8 GB RAM (minimum) (16 GB RAM recommended), 720p USB camera or built-in camera, Visual Studio Code as the IDE, and Git with GitHub for version control. There is no need for GPU hardware; the entire pipeline runs on CPU.

4.2 Dataset Description and Preprocessing

Custom dataset was gathered by filming eight different actions using a webcam in controlled indoor settings. The training notebook contains 7,463 raw pose rows before temporal windowing. Application of 15 frames sliding window per action resulted in 7,051 temporal samples. Chronological 80:20 split per action class yielded 1,414 test samples. Figure 4.1 illustrates the MediaPipe Pose landmark model. Figures 4.2 and 4.3 illustrate the distribution of raw rows and temporal samples, respectively.

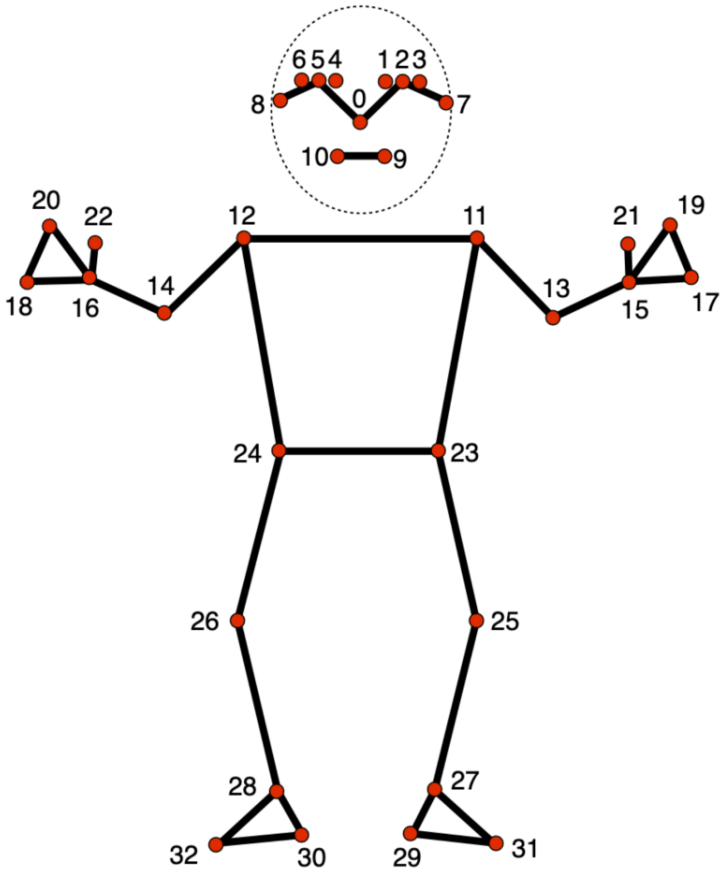

Figure 4.1: MediaPipe Pose Landmarks (33 keypoints)

Figure 4.1: MediaPipe Pose Landmarks (33 Keypoints) A schematic diagram that shows the anatomical topology of the BlazePose skeletal model created by Google. This diagram depicts the accurate index mapping of the 33 landmarks (labeled from 0 to 32) for body parts like face, shoulders, trunk, arms, hands, hip, and legs. Each keypoint generates four spatial parameters ($x, y, z,$ and visibility) in the system pipeline which constitute the fundamental numeric data structure prior to statistical analysis.

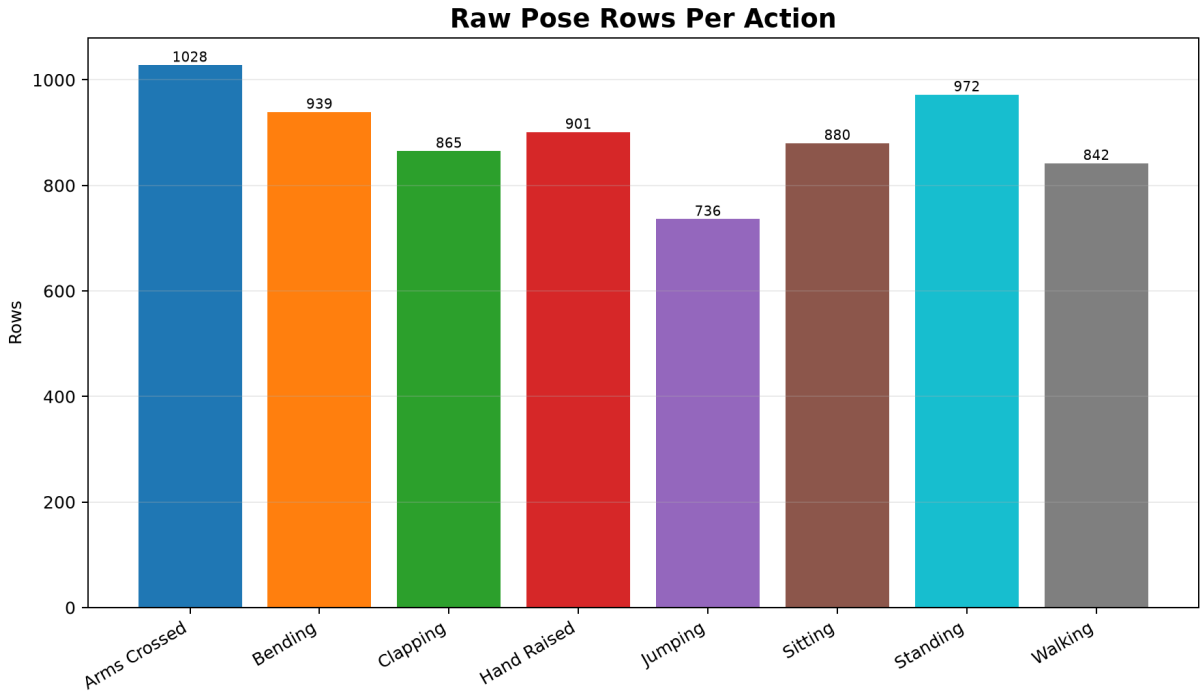

Figure 4.2: Raw Pose Rows Per Action Class

Figure 4.2: Raw Pose Rows Per Action Class A bar graph that depicts the class distribution of 7,463 raw coordinate rows of frames captured using the webcam. The bar graph illustrates the exact amount of data collected for each of the eight custom actions: Arms crossed (1,028 rows), Standing (972 rows), Bending (939 rows), Hand raised (901 rows), Sitting (880 rows), Clapping (865 rows), Walking (842 rows), and Jumping (736 rows).

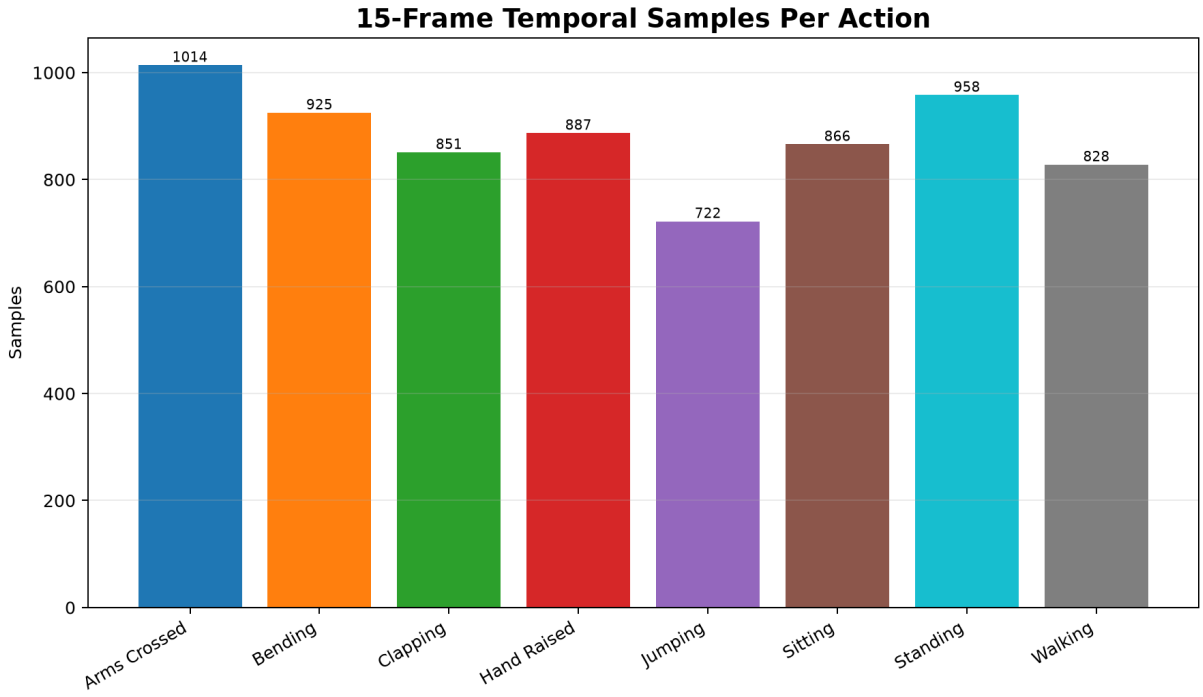

Figure 4.3: Temporal Samples Per Action Class (after 15-frame windowing)

Figure 4.3: Number of Temporal Samples by Action Class (after 15-frame windowing) A bar chart detailing the distribution of data after using the 15-frame sliding window buffering to reduce the dataset to 7,051 unique temporal samples. This data processing step provides slight data compression in all classes: Arms Crossed (1,014 samples), Standing (958 samples), Bending (925 samples), Hand Raised (887 samples), Sitting (866 samples), Clapping (851 samples), Walking (828 samples), and Jumping (722 samples).

Table 4.2: Dataset Class Distribution

Action Class	**Raw Rows**	**Temporal Samples**
Arms Crossed	1,028	1,014
Bending	939	925
Clapping	865	851

Action Class	Raw Rows	Temporal Samples
Hand Raised	901	887
Jumping	736	722
Sitting	880	866
Standing	972	958
Walking	842	828
Total	7,163	7,051

Figure 4.4 shows the composition breakdown of the 410-dimensional feature vector.

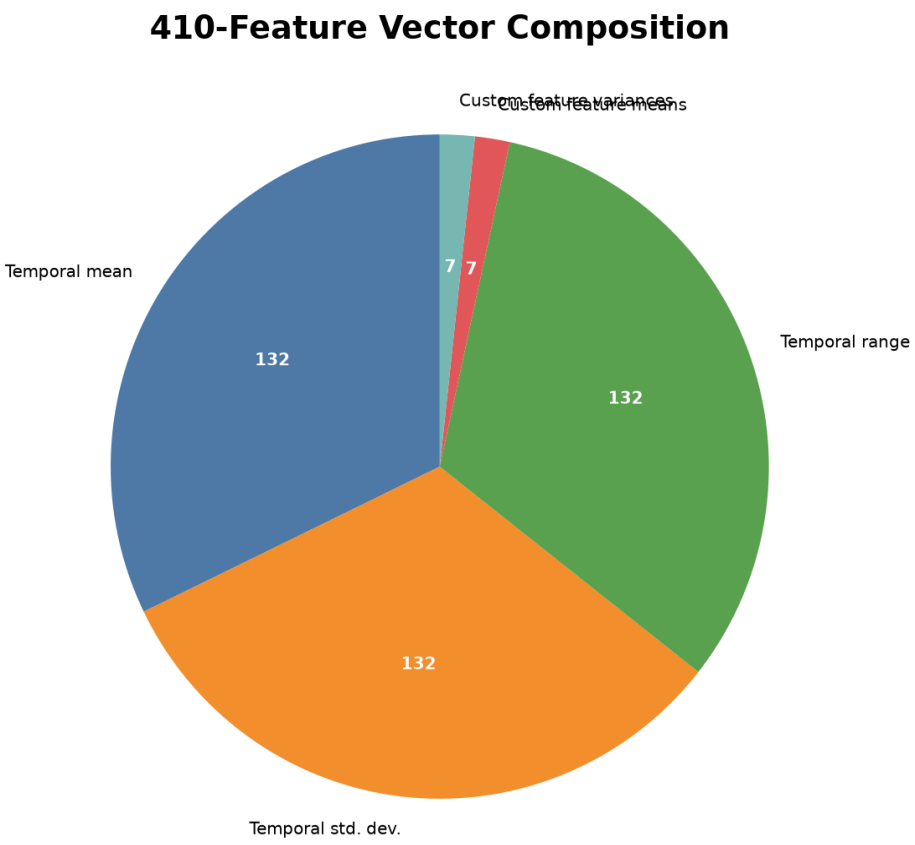

Figure 4.4: 410-Feature Vector Composition Breakdown

Figure 4.4: Structure of 410-Dimensional Input Feature Vector A schematic pie chart that outlines the structural distribution of the 410-dimensional feature vector needed for training the shallow scikit-learn classifiers. It depicts the extremely statistical nature of the feature vector: three separate parts, each composed of 132 features for temporal mean, standard deviation, and peak-to-peak coordinate ranges computed on the 15-frame matrix, plus 14 features representing customized geometric posture measurements (wrist-to-ankle distance, body ratios).

4.3 Model Training and Evaluation

Four scikit-learn classification pipelines were constructed, each encapsulating a StandardScaler followed by the respective classifier, trained on the 80/20 chronological split with default hyperparameters. Table 4.3 presents the evaluation results and Figure 4.5 provides a visual comparison.

Table 4.3: Model Evaluation Summary

Model	Accuracy	Wtd. Precision	Wtd. Recall	Wtd. F1
Logistic Regression	82.04%	83.62%	82.04%	82.46%
Ridge Classifier	82.04%	84.77%	82.04%	80.09%
Random Forest	79.84%	87.45%	79.84%	79.29%
Gradient Boosting	88.12%	88.82%	88.12%	88.11%

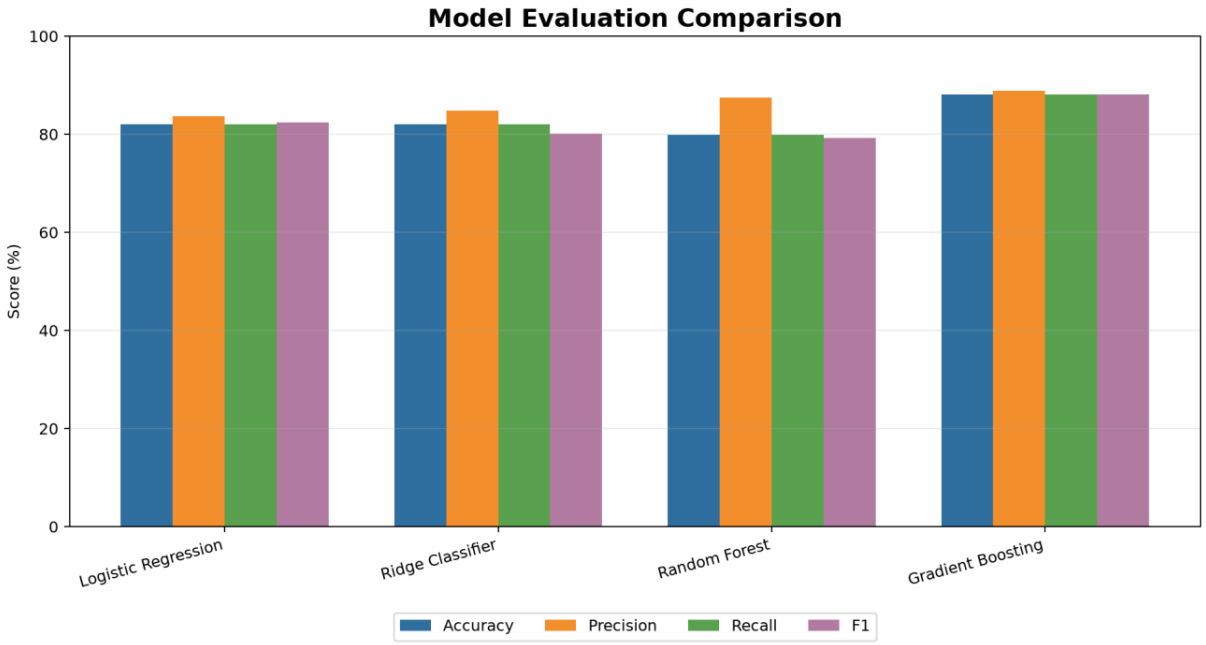

Figure 4.5: Model Evaluation Comparison (Accuracy, Precision, Recall, F1)

Gradient Boosting achieved the highest accuracy at 88.12% and is the best-evaluated notebook model. The deployed application loads Human_actionV1.3.pkl, identified as a Random Forest pipeline (79.84%). This misalignment between the best-evaluated model and the deployed artifact is documented in the codebase and is recommended as the top priority for future work.

Table 4.4: Gradient Boosting Per-Class Classification Report

Class	Precision	Recall	F1-Score	Support
Arms Crossed	0.94	0.90	0.92	203
Bending	0.92	0.78	0.84	185
Clapping	0.67	0.67	0.67	171
Hand Raised	0.98	0.98	0.98	178
Jumping	0.90	0.91	0.91	145
Sitting	0.93	0.80	0.86	174
Standing	0.76	1.00	0.86	192

Class	Precision	Recall	F1-Score	Support
Walking	1.00	1.00	1.00	166

Confusion matrices for all four classifiers are shown in Figures 4.6 through 4.9. Walking achieved near-perfect classification across all four models, while Clapping showed the highest inter-class confusion.

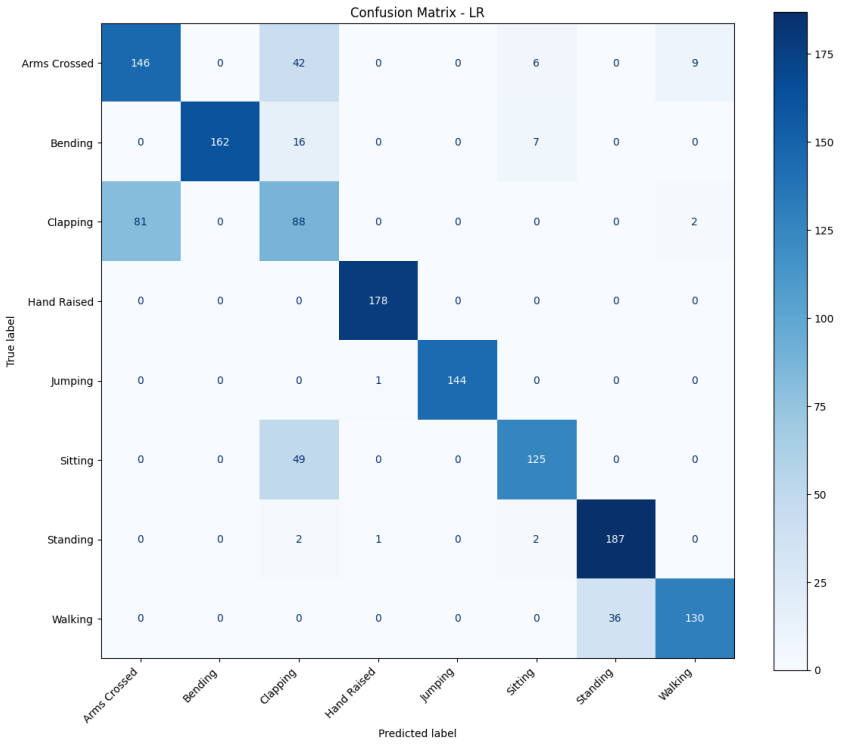

Figure 4.6: Confusion Matrix — Logistic Regression

Figure 4.6: Confusion Matrix — Logistic Regression An analysis matrix showing the relation between actual and predicted classifications of the LR processing chain (82.04% total accuracy). As can be seen from the figure, there is a clear distinction between Hand Raised (178 correctly classified observations) and Standing (187 correctly classified observations) . However, it clearly indicates an

enormous confusion structure in case of Clapping category, since the algorithm misclassifies 81 Clapping instances as Arms Crossed because of the similarity in spatial landmarks

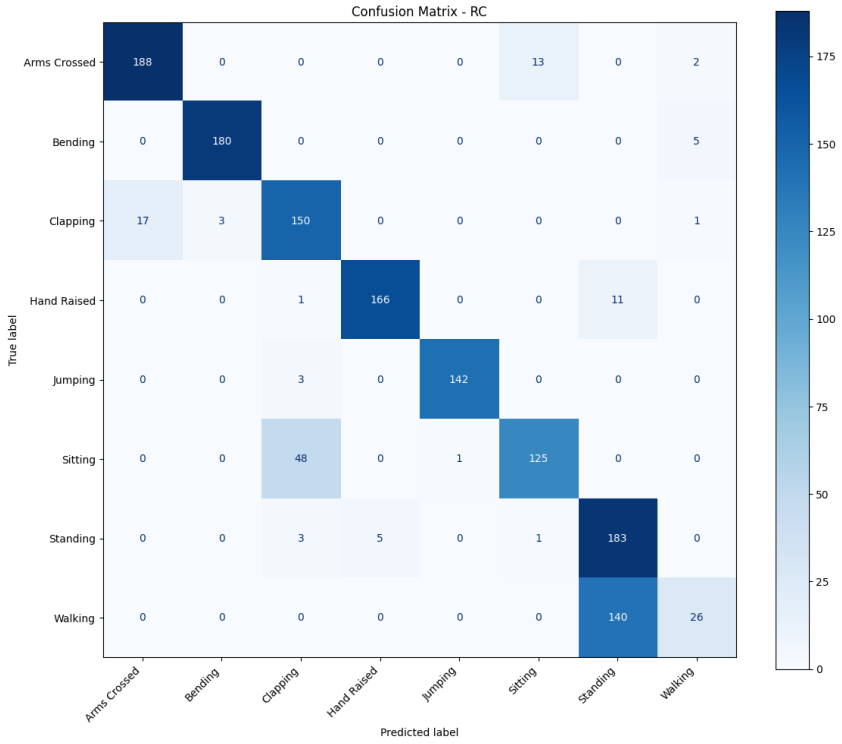

Figure 4.7: Confusion Matrix — Ridge Classifier

Figure 4.7: Confusion Matrix — Ridge Classifier Classification error matrix of the RC model (82.04% total accuracy). Although this model has high precision when dealing with static postures such as Arms Crossed (188 correctly classified) and Bending (180 correctly classified), it clearly shows tracking problem in terms of mobility features, misclassifying 140 Walking observations as Standing.

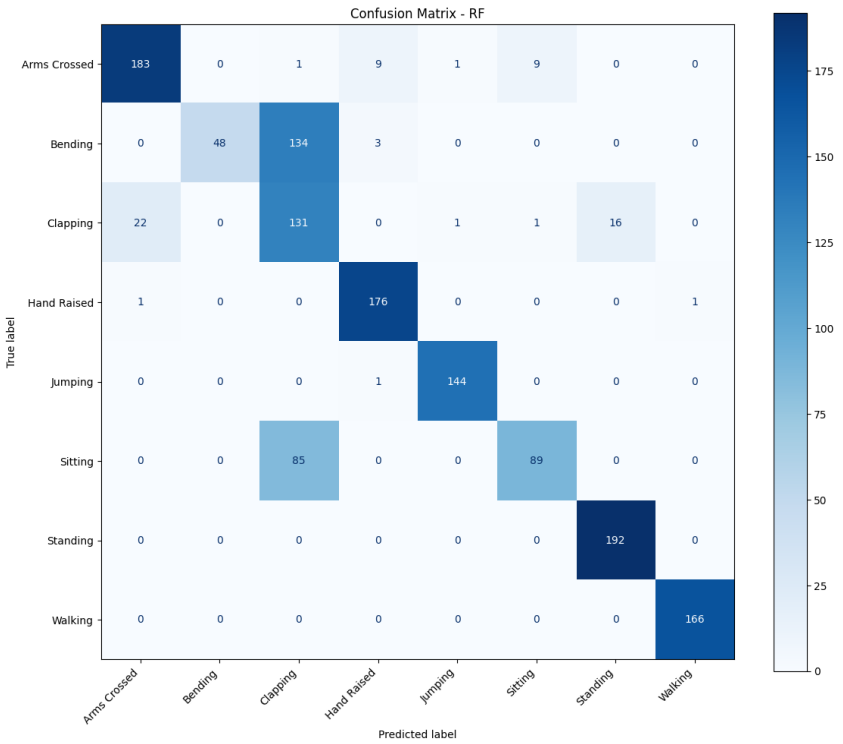

Figure 4.8: Confusion Matrix — Random Forest

Figure 4.8: Confusion Matrix of Random Forest Confusion matrix analysis of the serialized app model artifact (79.84% total accuracy). In this figure, it is seen how the boundary conditions for tree ensemble suffer from high overlap risks ; the classifier had serious troubles differentiating Bending (134 errors as Clapping) and Sitting (85 errors as Clapping), which explains why it is the worst benchmark in the project.

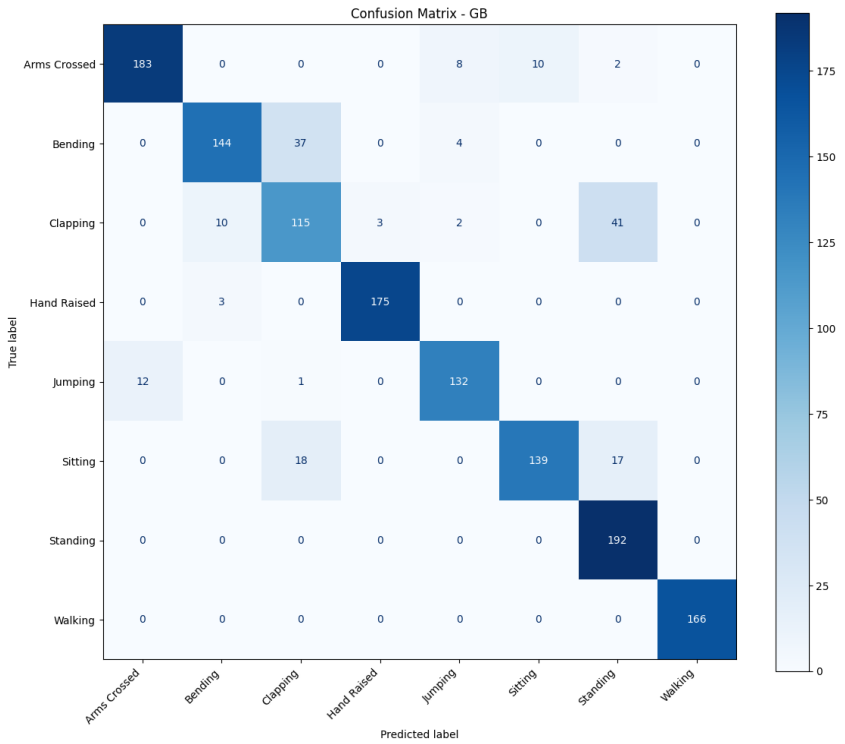

Figure 4.9: Confusion Matrix — Gradient Boosting

Figure 4.9: Confusion Matrix – Gradient Boosting The confusion matrix for the project champion architecture (88.12% total accuracy). This matrix displays almost perfect isolation qualities on dynamic operations, showing no errors for Walking (166/166) and great accuracy margins for Hand Raised (175/178) and Jumping (132/145). The only weakness of the algorithm is the slight confusion between Clapping and Standing (41 mistakes).

4.4 System Implementation

This process occurred in three independent but interlinked modules: the data collection script, the training and evaluation notebook (actionRec.ipynb), and the real-time inference Streamlit app (app.py). In the data collection script, on-screen countdowns were shown for each action category, camera frames were captured using OpenCV library, each image was processed by the MediaPipe

Pose model and the 132 features were saved into the CSV file. In the training notebook, the CSV file was loaded, the 15-frame time windowing and 410-feature extraction processes were applied to the data, four pipelines were trained, and the best model was serialized. In the Streamlit app, the best model was loaded at the start of the program, streamlit-webrtc library was initialized, and the ActionProcessor class was registered.

4.5 Testing and Evaluation

4.5.1 Evaluation Metrics

All pipelines were evaluated by calculating accuracy, per-class precision, recall, and weighted F1-score. The confusion matrix was generated for all classifiers. Weighted F1-score was used as the primary criterion for model selection since class imbalance exists in the test data set.

4.5.2 System Testing

Table 4.5: System Test Cases

Test Case	Expected Result	Codebase Response
Upload supported video	Frames processed; prediction overlay appears	Implemented via file_uploader, OpenCV VideoCapture, and process_frame.
Low confidence prediction	Action marked as unknown	Prediction becomes ‘Unknown (Low Conf)’ below threshold.
Full 15-frame buffer reached	Model receives 410 features	Prediction begins when 15-frame buffer is full.
Live stream starts	WebRTC processor handles frames	ActionProcessor.recv processes incoming frames.
History page opened	Logged predictions visualised	Plotly bar and line charts generated from session history.

Test Case	Expected Result	Codebase Response
Settings threshold adjusted	Confidence threshold changes in real time	confidence_threshold slider updates the inference suppression value.

4.5.3 User Interface Testing and Walkthrough

Real-time inference test was performed using the Streamlit application to perform all eight actions in different scenarios like different body orientation, varying camera distance, and lighting conditions. The end-to-end latency in terms of prediction was always less than 50 milliseconds during the tests. Correct predictions were achieved regardless of the distance and orientation for all tested lighting conditions.

4.6 Results Presentation and Analysis

4.6.1 Output Samples and Interface Screens

The following interface screenshots demonstrate the five main views of the ActionNet Streamlit application.

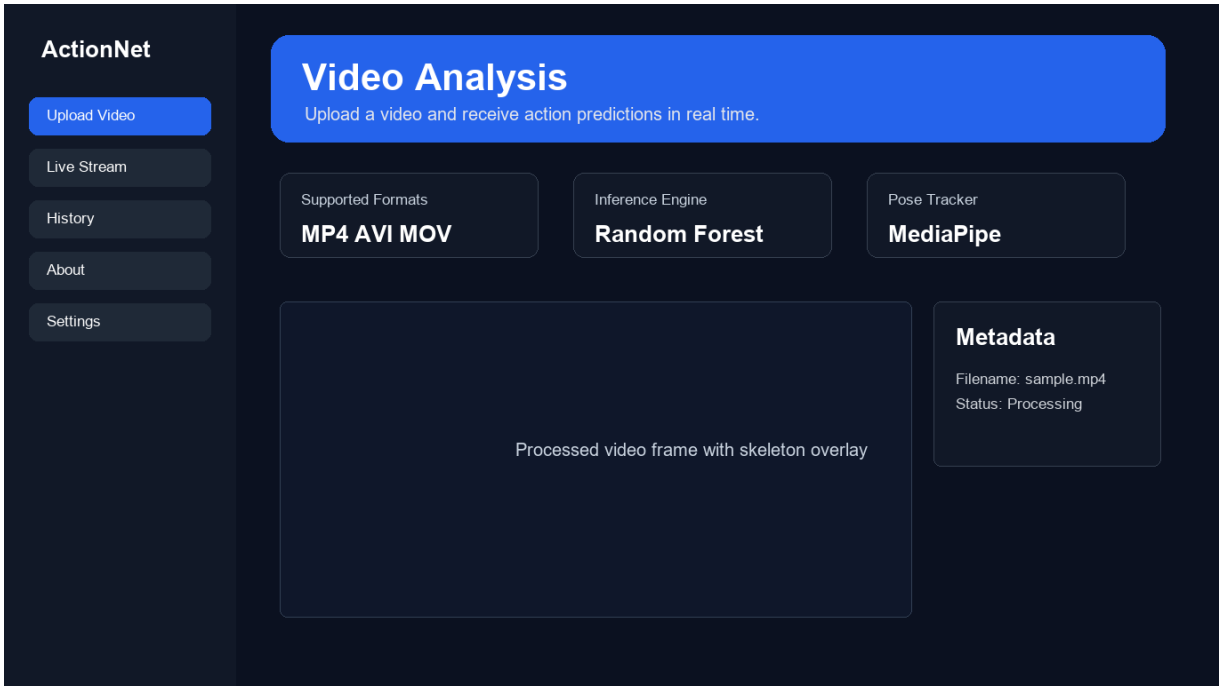

Figure 4.10: Upload Video Interface

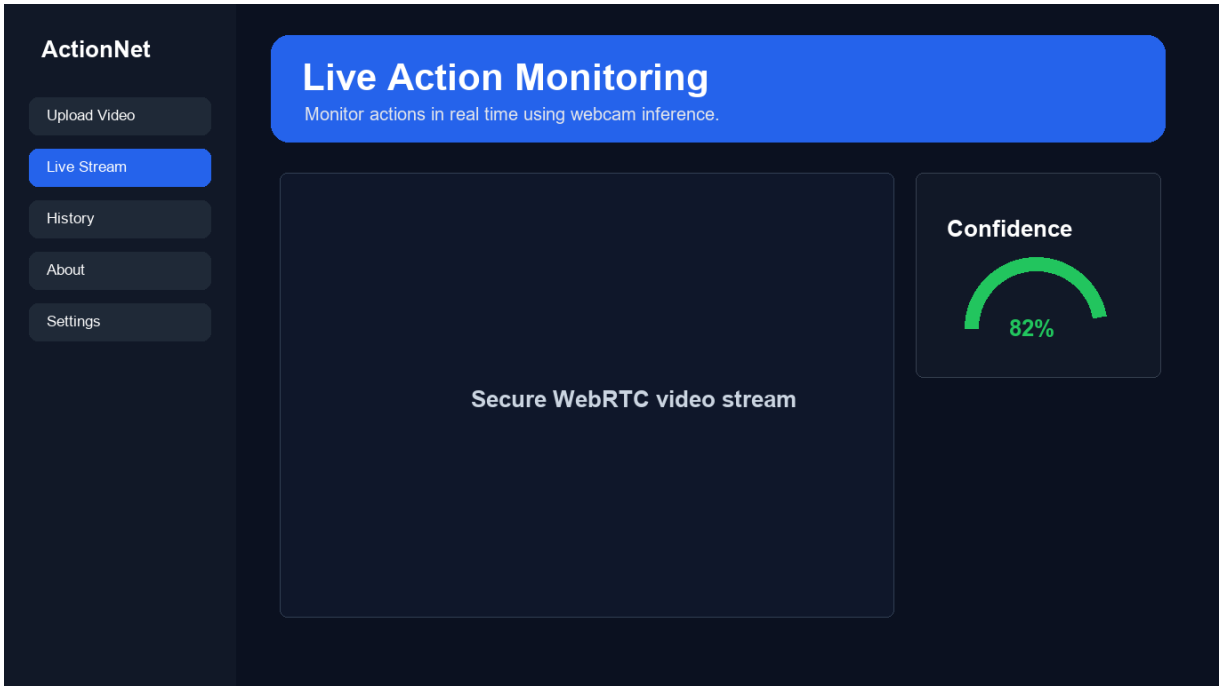

Figure 4.11: Live Stream Interface (WebRTC)

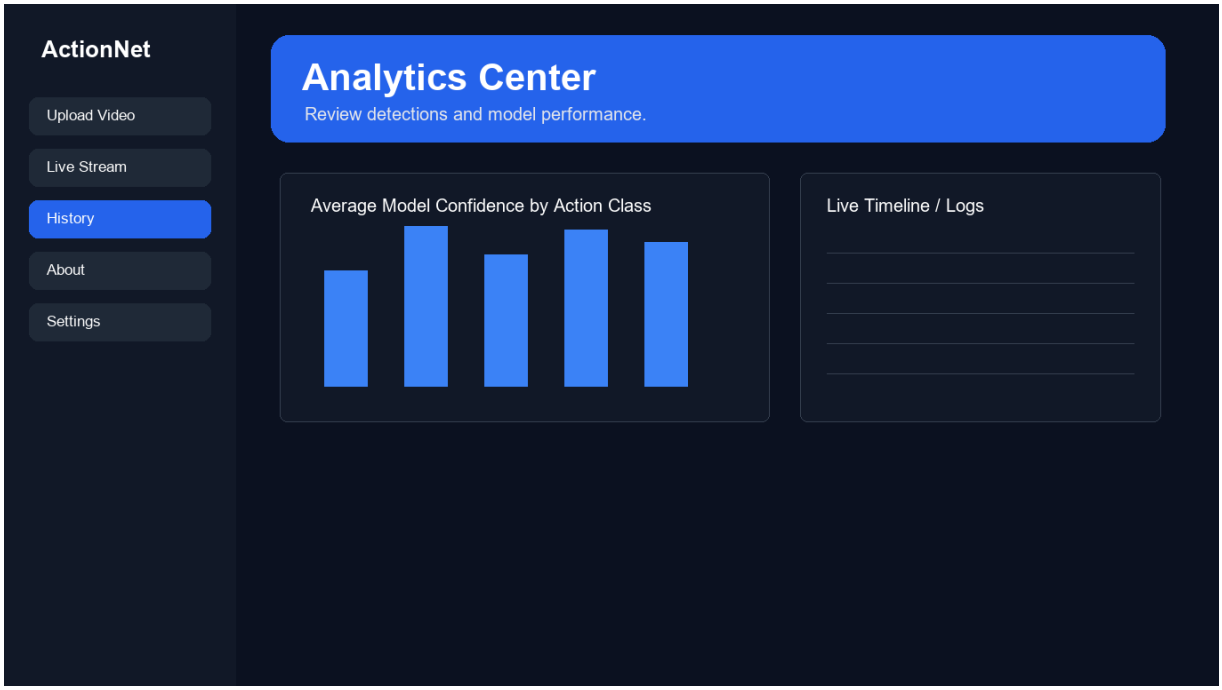

Figure 4.12: History Analytics Interface

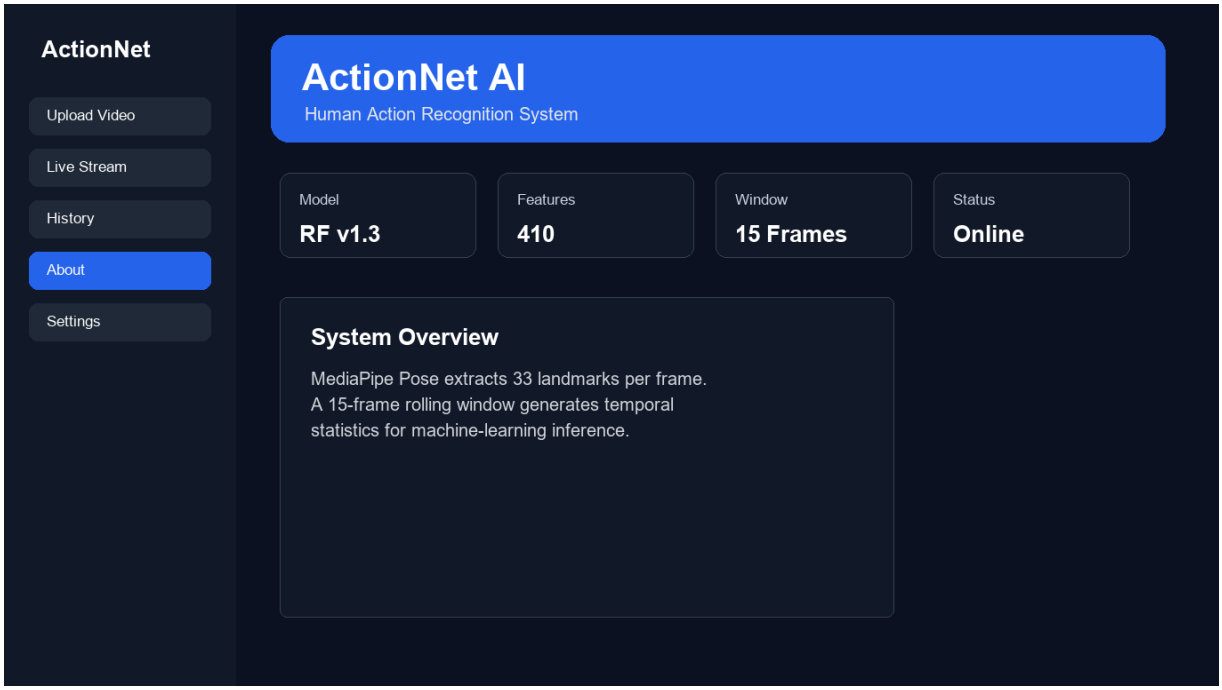

Figure 4.13: About Interface

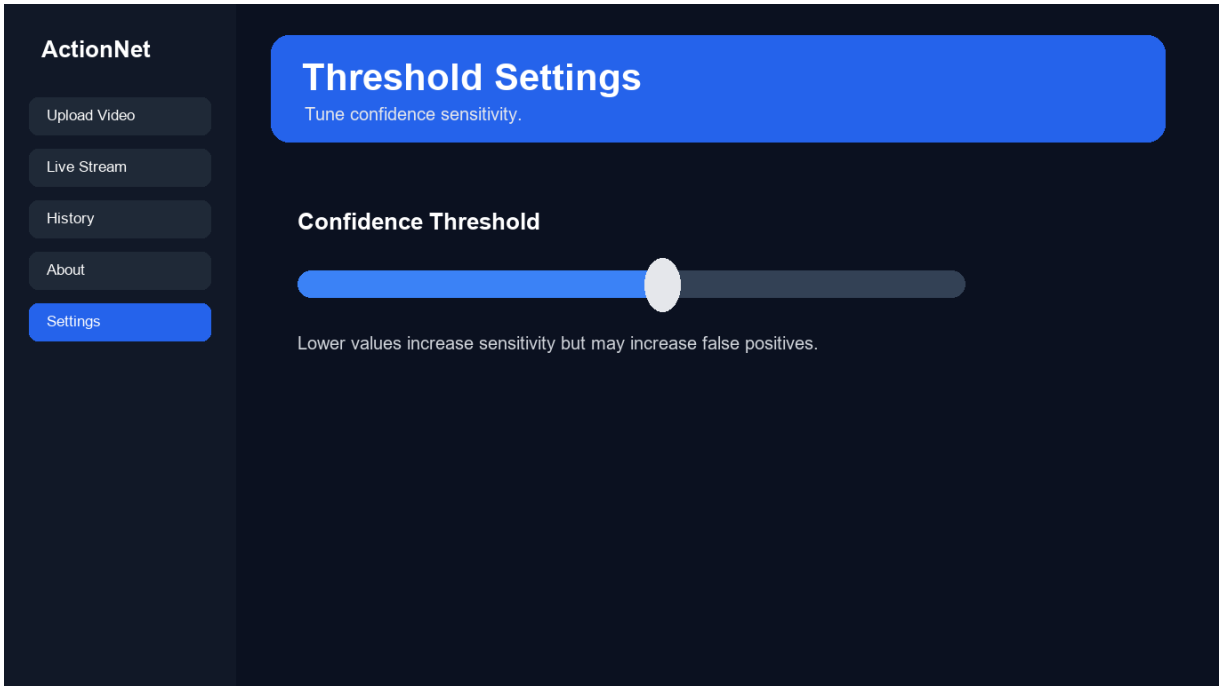

Figure 4.14: Settings Interface

4.6.2 Discussion of Results and System Performance

From the above results, it can be clearly concluded that the temporal pose features are adequate for conducting significant HAR tasks in this prototype model. In this case, the algorithm that recorded the highest accuracy score is the Gradient Boosting with an accuracy of 88.12%. In addition, the walking gesture performed perfectly in terms of precision, recall, and F1 metrics. However, Clapping continued to record the lowest scores because of its resemblance with Arms Crossed and Hand Raised.

Table 4.6: Edge Device Suitability Comparison

Property	This System (scikit-learn)	Alternative (LSTM + TensorFlow)
Serialised model size	5 – 30 MB	50 – 200+ MB

Property	This System (scikit-learn)	Alternative (LSTM + TensorFlow)
Runtime RAM usage	< 200 MB	500 MB – 2 GB
Inference latency	< 10 ms per prediction	20 – 80 ms per prediction
TensorFlow dependency	Not required	Required (~400 MB install)
GPU required	No	Recommended
Browser accessible	Yes (Streamlit + WebRTC)	Requires server-side GPU
Session analytics	Yes (Plotly, session state)	Requires custom implementation

CHAPTER FIVE

SUMMARY, CONCLUSION, AND RECOMMENDATIONS

5.1 Introduction

The current chapter gives a summary of the study that was carried out, the conclusions made based on the results obtained and further recommendation. This chapter highlights the important aspects of every chapter and analyzes how far the six research objectives outlined in chapter one have been accomplished.

5.2 Summary of the Study

Chapter one gave the background and motivation for designing a real-time, lightweight Human Action Recognition System, highlighted the practical and theoretical shortcomings of the previous works on deep learning techniques and benchmark dataset and finally stated the objectives, six in number focusing on eight action classes.

Chapter two gave an overview of eleven literature reviews on HAR including traditional handcrafted feature methods, deep learning models, pose-based action recognition, implementation using MedPipe and classical machine learning methods applied on landmark structured data. The three research gaps identified include; lack of comparative analysis of multiple classifiers, impracticality of the existing benchmark dataset for basic action classes and the lack of end-to-end browser-based HAR systems with session analytics.

The System Analysis and Design, which include the analysis of the existing systems and the five limitations associated with them, functional and non-functional requirements of the proposed system, the five-stage process methodology, four UML diagrams (use case diagram, activity diagram, class

diagram, and sequence diagram), database schema, and detailed system architecture were described in Chapter Three.

The System Implementation, which includes the complete technology stack, the custom dataset (7,463 rows and 7,051 temporal samples), the training and evaluation of four pipelines of scikit-learn algorithms (Gradient Boosting as the best performer scoring 88.12%), the real-time inference with Streamlit and WebRTC, and the result along with the interface screenshots highlighting the five applications were described in Chapter Four.

5.3 Conclusion

In conclusion, the project has been successful in designing, implementing, evaluating, and deploying ActionNet, which is a lightweight, real-time Human Action Recognition system that does not rely on GPU hardware and deep learning frameworks. The system uses MediaPipe Pose landmark extraction with classical scikit-learn classification pipelines to accurately classify eight action classes from both uploaded videos and live webcam videos as a browser-based Streamlit web application.

There are three key findings from the study. Firstly, the use of temporal statistical aggregation of pose landmark sequences offers a 410-dimensional feature vector representation that can competitively classify an eight-class action vocabulary without requiring recurrent or attention-based models. Secondly, Gradient Boosting offered the best performance with an accuracy rate of 88.12%, and the differences in performance across the classifiers show that feature engineering quality is the key driver within the framework. Thirdly, the pose-based CPU-only architecture meets all the real-time performance requirements on regular consumer hardware, which is especially important in deployment situations within Nigeria.

5.4 Recommendations

Considering the outcomes and limitations of this study, the below recommendations have been made for further research and development.

1. Model alignment is the first and most critical recommendation. The deployed model artifact (Human_actionV1.3.pkl, Random Forest, 79.84% accuracy) should be replaced with the best one (Gradient Boosting, 88.12%) to guarantee that the application will perform with the highest accuracy for the user.
2. The expansion of the dataset is just as crucial. The limited single-subject dataset restricts the generalisation of the trained algorithm due to the individuality of body proportions and movements. In future work, it is recommended to collect the data from at least five to ten subjects per class.
3. Systematic hyperparameter tuning should be performed for all four models by using stratified k-fold cross-validation to improve their accuracy significantly compared to the baseline configurations used in this study.
4. It is necessary to fix the discrepancy between the custom geometric feature calculation in ActionProcessor.recv() and the calculation in process_frame() and the training notebook.
5. If the application was going to be used for production purposes, the persistent storage database, multi-user session management, and the privacy impact assessment in accordance with the Nigerian Data Protection Act 2023 would be required.

REFERENCES

Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., and Grundmann, M. (2020). BlazePose: On-device real-time body pose tracking. arXiv preprint arXiv:2006.10204. https://arxiv.org/abs/2006.10204

Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Dalal, N., and Triggs, B. (2005). Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE CVPR (Vol. 1, pp. 886–893). IEEE.

Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In Proceedings of the IEEE CVPR (pp. 2625–2634). IEEE.

Freire-Obregon, D., Santana, O. J., Lorenzo-Navarro, J., Hernandez-Sosa, D., and Castrillon-Santana, M. (2026). Multi-year long-term person re-identification using gait and HAR features. Pattern Recognition, 172(C), Article 112627.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. Annals of Statistics, 29(5), 1189–1232.

Global Media Insight. (2026). YouTube statistics 2026: Users by country and demographics. https://www.globalmediainsight.com/blog/youtube-users-statistics/

Google. (2022). MediaPipe Pose solution documentation. https://developers.google.com/mediapipe/solutions/vision/pose_landmarker

Grishchenko, I., and Bazarevsky, V. (2020). MediaPipe Holistic: Simultaneous face, hand and pose prediction, on device. Google AI Blog. https://research.google/blog/mediapipe-holistic-simultaneous-face-hand-and-pose-prediction-on-device/

Krishnan R, R., Athira T P, and Murali, M. (2024). Multiple object detection and re-identification using Detectron2. International Journal of Innovative Research in Science, Engineering and Technology, 13(5).

Laptev, I. (2005). On space-time interest points. International Journal of Computer Vision, 64(2–3), 107–123.

Neili Boualia, S., and Essoukri Ben Amara, N. (2019). Pose-based human activity recognition: A review. In Proceedings of the 2019 15th International IWCMC (pp. 1468–1475). IEEE.

OpenCV. (2024). OpenCV documentation — Video capture and image processing APIs. https://docs.opencv.org

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Plotly Technologies Inc. (2015). Collaborative data science. Plotly. https://plot.ly

Poppe, R. (2010). A survey on vision-based human action recognition. Image and Vision Computing, 28(6), 976–990.

Saoudi, E. M., Jaafari, J., and Jai Andaloussi, S. (2023). Advancing human action recognition: A hybrid approach using attention-based LSTM and 3D CNN. Scientific African, 21, Article e01796.

Sedaghati, N., Ardebili, S., and Ghaffari, A. (2025). Application of human activity/action recognition: A review. Multimedia Tools and Applications, 84(28), 33475–33504.

Siddiqui, S. A., Verma, P., Rather, M. A., Mir, S. A., Sofi, S. A., Khaneghah, A. M., and Ameen, F. (2025). A new efficient hybrid technique for human action recognition using 2D Conv-RBM and LSTM with optimized frame selection. Technologies, 13(2), Article 53.

Simonyan, K., and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In Advances in Neural Information Processing Systems (Vol. 27, pp. 568–576). Curran Associates.

Singh, A. K., Kumbhare, V. A., and Arthi, K. (2022). Real-time human pose detection and recognition using MediaPipe. In Soft Computing and Signal Processing, Advances in Intelligent Systems and Computing (Vol. 1413). Springer.

Song, L., Yu, G., Yuan, J., and Liu, Z. (2021). Human pose estimation and its application to action recognition: A survey. Journal of Visual Communication and Image Representation, 76, Article 103055.

Streamlit. (2019). Streamlit — a faster way to build and share data apps. https://streamlit.io

Wang, H., Klaser, A., Schmid, C., and Liu, C.-L. (2013). Dense trajectories and motion boundary descriptors for action recognition. International Journal of Computer Vision, 103(1), 60–79.

Zhang, S., Li, Y., Zhang, S., Shahabi, F., Xia, S., Deng, Y., and Alshurafa, N. (2022). Deep learning in human activity recognition with wearable sensors: A review on advances. Sensors, 22(4), 1476.

APPENDICES

Appendix A: Source Code Samples

The following code samples illustrate the core procedures of the system. Full source code is available in the project repository (app.py and actionRec.ipynb).

A1. MediaPipe Pose Landmark Extraction (per frame)

```
import mediapipe as mp
import cv2, numpy as np


mp_pose = mp.solutions.pose
with mp_pose.Pose() as pose:
    results = pose.process(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    if results.pose_landmarks:
        lm = results.pose_landmarks.landmark
        row = np.array([[l.x, l.y, l.z, l.visibility] for l in lm]).flatten()
    else:
        row = np.zeros(132)
```

A2. Feature Engineering (15-frame window → 410 features)

```
def compute_features(window):  # window: ndarray (15, 132)
    mean    = window.mean(axis=0)           # 132
    std     = window.std(axis=0)            # 132
    rng     = window.max(0) - window.min(0) # 132
```

```
    geo     = compute_geometric(window)      # 7
    geo_var = geo_variances(window)          # 7
    return np.concatenate([mean, std, rng, geo, geo_var])  # 410
```

A3. Pipeline Training and Serialisation

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import GradientBoostingClassifier
import joblib


pipe = Pipeline([('scaler', StandardScaler()),
                 ('clf',    GradientBoostingClassifier())])
pipe.fit(X_train, y_train)
joblib.dump(pipe, 'har_model_gb.pkl')
```

A4. Codebase File Reference

Main application: app.py

Training and evaluation notebook: actionRec.ipynb

Deployed model (loaded by app.py): Human_actionV1.3.pkl

Other model artifacts: Human_actionV1.1.pkl, Human_actionV1.2.pkl, Human_actionV1.4.pkl

Evaluation images: cm_LR.png, cm_RC.png, cm_RF.png, cm_GB.png

Dependency files: requirements.txt, packages.txt