HUMAN ACTION RECOGNITION USING MACHINE LEARNING AND DEEP LEARNING

[1]M Nagaraj, [2]Ganti Priyanka, [3]Rakshita Basavaraj Laxani, [4]Bhumika
[1]Assistant Professor ,Department of CSE
Roohisafiya Allisab Walikar Computer Science and Engineering, Ballari
walikarroohi@gmail.com
[2]Computer Science and Engineering, Ballari
Priyankaganti2151@gmail.com
[3]Computer Science and Engineering, Ballari
rakshitalaxani2@gmail.com
[4]Computer Science and Engineering, Ballari
bhoomikapuladinni@gmail.com

Abstract

Human Action Recognition (HAR) is a computer-vision problem in which a system observes a person and identifies the activity being performed. This project develops a pose-based HAR prototype that combines Machine Learning (ML) and Deep Learning (DL) for nine predefined activities: Standing, Sitting, Hands Up, Walking, Jumping, Waving, Reading, Dancing and Lying Down. The webcam provides visual frames, MediaPipe Pose extracts 33 human body landmarks, and each landmark contributes x, y, z and visibility values. Consequently, every valid frame is represented by 132 numerical features. Random Forest is used as a frame-level ML baseline, whereas Long Short-Term Memory (LSTM) is used to learn temporal information from 30 consecutive frames. The final held-out evaluation produced 19.00% accuracy for Random Forest and 42.11% for LSTM. The trained LSTM is integrated with a SmartMotion web interface through a Flask prediction service, allowing live action prediction. The work provides an end-to-end pipeline covering data preparation, pose extraction, model training, evaluation and deployment.

Keywords

Human Action Recognition, Machine Learning, Deep Learning, Random Forest, LSTM, MediaPipe Pose, Pose Landmarks, Computer Vision, Temporal Sequence, Flask, SmartMotion, Real-Time Prediction.

1. Introduction

Human activities are naturally expressed through body posture and movement. In applications such as smart surveillance, education, fitness, assisted environments and human-computer interaction, recognizing an activity automatically can reduce the need for continuous manual observation. A video contains a large amount of visual information, but a recognition system needs a compact representation that can be processed consistently by a learning algorithm.

One possible approach is to classify raw images directly. However, raw images contain background, clothing, lighting and other visual factors that are not necessarily related to the action. A pose-based approach focuses on the geometry of the human body. Body landmarks provide a structured representation that can be converted into numerical features and reused by different models.

The distinction between posture and movement is important. Standing and Sitting can often be identified from a body configuration, while Walking, Waving, Dancing and Jumping depend strongly on changes across successive frames. Therefore, this project compares a frame-level Random Forest model with an LSTM model that observes a short temporal sequence.

The complete workflow begins with webcam capture, followed by pose landmark detection and numerical feature extraction. The prepared data are used to train and evaluate the models. After evaluation, the LSTM is connected to the Smart Motion interface so that the same general feature representation can be used for live inference.

2. Problem Statement

The problem is to automatically recognize a predefined human activity from webcam video. The system must convert the continuous camera stream into a consistent numerical representation, associate that representation with one of the target activity labels, evaluate the learned model on data not used for training and provide an understandable prediction during live operation.

A second challenge is temporal context. If each frame is considered independently, two different activities may produce similar body poses at a particular instant. A sequence model can use the order and change of poses to obtain additional information. The project therefore treats frame classification and sequence classification as related but distinct learning tasks.

3. Literature Review and Existing System

Human action recognition research includes image-based classification, video-based deep learning, skeleton and pose-based methods, and recurrent neural networks. Traditional ML methods can perform well when the input features are informative, but isolated-frame classification has an inherent limitation for activities whose identity is defined by movement. Recurrent sequence models address this by processing observations in temporal order.

Basic existing systems may depend on manual observation, isolated image recognition or a model that does not explicitly preserve temporal information. Such systems can also be affected by viewpoint, lighting, occlusion, subject distance and differences between the training and live environments. A pose-landmark representation reduces the dependence on raw appearance, while the LSTM provides explicit temporal context.

4. Research Motivation

The project is motivated by the need for a practical and understandable activity-recognition pipeline that can be demonstrated in real time. Using pose landmarks makes the learning input compact and interpretable, while comparing Random Forest and LSTM provides a direct way to study the benefit of temporal modeling. The SmartMotion interface further connects the research prototype to an actual user interaction workflow.

5. Objectives and Scope

Objectives:

- Develop an automated webcam-based human action recognition system.
- Recognize nine predefined human activities.
- Use MediaPipe Pose to extract standardized body-pose information.
- Represent each frame using 132 numerical pose features.
- Train Random Forest as the Machine Learning baseline.
- Train LSTM for temporal Deep Learning recognition.
- Evaluate the models using held-out data.
- Provide live prediction through the SmartMotion web interface.
- Compare frame-based and sequence-based approaches.
- Provide a modular foundation for future extension.

Scope: The completed prototype is focused on nine predefined activities and webcam-based recognition. It covers dataset preparation, pose extraction, model development, offline evaluation and live LSTM inference. The scope does not claim general recognition of every possible human activity.

6. Action Classes

The nine target classes are Standing, Sitting, Hands Up, Walking, Jumping, Waving Reading, Dancing and Lying Down. The selection contains both relatively stable postures and actions with visible movement. Standing, Sitting and Lying Down represent posture states. Hands Up and Reading provide recognizable body configurations. Walking, Jumping, Waving and Dancing introduce stronger temporal behavior. Having both static and dynamic classes is useful for comparing the two modeling strategies. A static posture may be recognizable from one frame, whereas a dynamic activity can require several observations. This distinction is one of the central ideas evaluated by the project.

7. Dataset and Data Preparation

The final training composition contains 8,482 samples. Approximately 900 own samples were used for each core class, while selected compatible public samples were added for Waving, Reading, Dancing and Lying Down. The project therefore combines project-collected examples with limited compatible public augmentation rather than treating every available public label as relevant.

The public augmentation source is Human Activity Pose Dataset v4, distributed under CC-BY-4.0. Only labels compatible with the project's target classes were selected. Training and test sources were kept separate so that the final evaluation could measure performance on held-out examples rather than simply reproducing training behavior.

For Random Forest, the final held-out evaluation contains 300 frames per class, giving 2,700 test frames. For LSTM, frame features are organized into ordered sequences of 30 frames. The final sequence data contain 7,839 training sequences and 2,439 test sequences.

8. Dataset Organization and Labeling

Samples are organized by activity class and associated with a target label. Maintaining separate training and testing sources makes the workflow easier to reproduce and allows class-wise analysis. Public augmentation is restricted to selected compatible activities so that the label definitions remain consistent.

The dataset design also reflects the difference between static and dynamic behavior. A static class may be represented by a stable body configuration, while a movement class needs examples showing changes in pose. This makes sequence construction particularly important for the LSTM pipeline.

9. Data Processing Flow

- Collect or load a labeled action sample.
- Read the corresponding image or video frame.
- •Detect the human pose using MediaPipe Pose.
- Extract x, y, z and visibility values from each landmark.
- Create the 132-feature frame representation.
- Associate the features with the correct action label.
- Store frame-level data for Random Forest.
- Group consecutive frames into 30-frame sequences for LSTM.
- Keep held-out data separate for final evaluation.

10. Data Quality Considerations

Pose-based learning depends on successful landmark detection. If a person is poorly visible, far from the camera or partially occluded, landmark quality can change. Differences in movement style can also make two examples of the same action look different. These factors are important when interpreting the final accuracy and motivate more diverse data collection in future work.

Pose-based learning depends on successful landmark detection. If a person is poorly visible, far from the camera or partially occluded, landmark quality can change Differences in movement style can also make two examples of the same action look different. These factors are important when interpreting the final accuracy and motivate more diverse data collection in future work

11. Pose Representation and Preprocessing

MediaPipe Pose detects 33 body landmarks. For each landmark, four values are retained: x-coordinate, ycoordinate, z-coordinate and visibility. The resulting representation is therefore $33 \times 4 = 132$ features per frame. This fixed representation provides a common interface between pose detection and the learning models.

The same feature structure is required during training and inference. A training sample that contains one frame is represented as a 132-value vector. For the LSTM pipeline, several such vectors are kept in temporal order. Thirty consecutive vectors create one sequence with input shape (30, 132). The sequence allows the model to observe how the pose evolves instead of observing only one instant.

Pose representation also reduces the amount of information that the learning algorithm must process compared with a raw image. The model focuses on body landmarks and their visibility rather than directly learning from every background pixel. This does not eliminate variation, but it creates a structured numerical input suitable for supervised learning.

12. Feature Extraction Process

- Receive an image/video frame from the input stream.
- Locate the person and estimate the body pose.

- Read the 33 landmark positions.
- Retain x, y, z and visibility for each landmark.
- Flatten the landmark information into a 132-value feature vector.
- Associate the vector with the activity label during dataset preparation.
- Reuse the same representation during live prediction.

13. System Architecture

Webcam → MediaPipe Pose Landmark Detection → 132 Features/Frame → 30-Frame Sequence → LSTM → 9Class Prediction → Smart Motion Web Interface.

The architecture can be divided into four logical stages. The input stage supplies the visual frames. The feature stage performs pose estimation and creates numerical landmark features. The model stage processes either individual frames or temporal sequences. The presentation stage displays the predicted action through Smart Motion.

The separation of these stages is useful because a change in one component does not conceptually require replacing the entire pipeline. For example, additional activities can be incorporated by extending the labeled data and model output space, while the pose representation can remain the common feature interface.

14. Methodology

The methodology starts with labeled action samples. Pose features are extracted from those samples and stored in a structured form. Random Forest uses individual frame vectors for its ML baseline. A separate sequence dataset is created for LSTM by grouping 30 consecutive feature vectors. Both models are evaluated on held-out data, after which the trained LSTM is connected to the Flask prediction service.

The methodology therefore contains two related paths. The first path asks whether a single pose can identify the activity. The second asks whether a short history of poses provides better evidence. The final comparison is based on held-out test performance rather than live confidence alone.

15. Key Processes

- Capture a frame from the webcam.
- Check whether a usable human pose is detected.
- Extract the 33 body landmarks.
- Convert the landmarks into 132 numerical values.
- Maintain the temporal window required by the LSTM.
- Prepare the model input in the expected shape.
- Run inference using the trained model.
- Map the model output to one of nine action labels.
- Return the prediction and confidence to the interface.

16. Modules Description

Dataset Module: Maintains labeled training and testing samples and supports selected public augmentation.

Pose Detection Module: Detects the body landmarks and supplies the structured pose representation. Feature Extraction Module: Converts 33 landmarks and their four retained values into the 132-feature frame vector.

Random Forest Module: Performs frame-level ML classification and supplies a baseline for comparison.

LSTM Sequence Module: Groups frames into 30-frame sequences and learns temporal relationships. Flask Prediction Module: Hosts the trained LSTM inference service and handles prediction requests from the web application.

SmartMotion Web Module: Provides the user-facing real-time interface and presents the predicted activity. Testing and Evaluation Module: Measures held-out performance using accuracy and class-wise precision, recall and F1-score.

17. Random Forest Model

Random Forest is the Machine Learning baseline. It receives the 132-feature representation of an individual frame and predicts an action class. This model is useful for comparison because it provides a conventional supervised-learning approach without explicitly representing a sequence of frames.

The final Random Forest evaluation achieved 19.00% accuracy on 2,700 held-out frames. The class-wise precision/recall/F1 values were: Standing 0.6119/0.1367/0.2234; Sitting 0.5256/0.1367/0.2169; Hands Up 0/0/0; Walking 0.2280/0.7600/0.3508; Jumping 0.0070/0.0033/0.0045; Waving 0.0471/0.0300/0.0367; Reading

0.0814/0.0233/0.0363; Dancing 0.1594/0.5467/0.2468; Lying Down 0.3929/0.0733/0.1236.

The uneven class-wise performance demonstrates the limitation of relying on one frame. Walking and Dancing obtained comparatively higher recall, while Hands Up and Jumping were especially difficult. A frame can provide useful posture information, but it cannot directly encode the direction or progression of movement.

18. LSTM Model

LSTM is the Deep Learning component because the identity of many human activities depends on temporal behavior. Each example contains 30 consecutive frames, with 132 features per frame. The input therefore has shape (30, 132). The model processes this short history and predicts one of the nine activity classes.

The final held-out LSTM evaluation achieved 42.11% accuracy on 2,439 test sequences. The improvement over Random Forest indicates that temporal context contributes useful information for this project. The result should nevertheless be interpreted as the performance of the completed prototype under the available data and evaluation conditions.

19. Why LSTM is Used

An action is often a pattern of change rather than a single posture. Walking involves repeated changes in leg and body configuration; waving involves a changing arm position; dancing can involve multiple successive body configurations; and jumping includes a progression from one posture to another. A 30-frame window provides the LSTM with a short movement history and therefore a more appropriate input for temporal recognition than an isolated frame.

20. Model Comparison

Random Forest and LSTM solve related classification problems with different information. Random Forest uses a single frame, whereas LSTM uses a sequence. The final accuracies are 19.00% and 42.11%, respectively. The difference supports the hypothesis that temporal information is useful for human action recognition. It does not mean that every LSTM class is recognized equally well; class balance and data diversity remain important.

21. Smart Motion Web Integration

The trained LSTM model is integrated into the SmartMotion web interface through a Flask backend. The Flask prediction service is started using python lstm_server.py and runs locally on port 5000 at 127.0.0.1:5000. The SmartMotion frontend is started using npm start and runs on port 3000 at localhost:3000.

The integration separates the user interface from the model inference service. The frontend is responsible for the camera interaction and presentation, while the backend receives the prepared prediction input, runs the LSTM model and returns the result. This separation also makes the system easier to test because the interface and model service can be considered as individual components.

22. Live Inference Workflow

- Start the SmartMotion web interface.
- Provide the webcam stream to the recognition workflow.
- Detect the person's pose from incoming frames.
- Create the same 132-feature representation used by the model.
- Maintain enough consecutive frames for the LSTM sequence.
- Send the prepared prediction input to the Flask service.
- Run LSTM inference and obtain the predicted class.
- Return the action and confidence to SmartMotion.
- Display the prediction to the user.

23. Live Demonstration

The integrated system successfully generated live predictions during testing. A representative demonstration produced Reading with 94% confidence. This number is the confidence associated with that individual prediction. It is not the overall accuracy of the model. The final held-out LSTM accuracy remains 42.11%. Live behavior can differ from offline evaluation because the camera environment, subject position and movement style may not match the held-out data. Live demonstrations are therefore useful for verifying endto-end functionality, while controlled held-out evaluation is used to report model performance.

24. Functional Testing

- Webcam capture and frame acquisition.
- Human pose landmark detection.
- Generation of the 132-feature representation.
- Construction of the 30-frame LSTM sequence.
- Communication with the Flask prediction endpoint.
- Return of a model prediction from the backend.
- Display of the returned prediction in SmartMotion.
- Live testing with Sitting, Standing, Hands Up and Waving.
- Comparison of live behavior with offline model evaluation.

25. Testing Strategy

Testing is divided into data-level, model-level and integration-level checks. Data-level checks concern whether the expected pose features and labels are produced. Model-level evaluation uses held-out frames or sequences so that the reported results measure generalization. Integration-level testing checks whether the trained LSTM can be connected to the Flask service and whether the web interface can present the returned prediction.

This layered strategy is important because a model can be mathematically evaluable while the live application still has an integration problem. Conversely, a successful live demonstration does not by itself establish model accuracy. The completed project considers both functional operation and quantitative evaluation.

26. Results and Discussion

The final Random Forest result is 19.00% accuracy on 2,700 held-out frames, while the final LSTM result is 42.11% accuracy on 2,439 held-out sequences. LSTM provides the stronger overall result. The comparison is consistent with the nature of the problem because many actions contain temporal information that a single frame cannot represent.

Random Forest showed comparatively stronger recall for Walking and Dancing, but it failed to recognize Hands Up in the final evaluation and performed extremely poorly for Jumping. These results show that a model may learn useful patterns for some classes while failing to generalize across the complete action set.

LSTM performed particularly well for Reading and Sitting and obtained useful recall for Walking and Waving. However, Jumping and Lying Down obtained zero precision, recall and F1 in the final test. These weaknesses identify specific areas where more representative temporal training examples are needed.

27. Performance Summary

Model	Input	Test Set	Accuracy
Random Forest	132 features/frame	2,700 frames	19.00%
LSTM	30 × 132 sequence	2,439 sequences	42.11%

28. Class-Wise LSTM Results

Class	P	R	F1
Dancing	0.8966	0.1919	0.3161
Hands Up	0.1545	0.1328	0.1429
Jumping	0	0	0
Lying Down	0	0	0
Reading	0.8469	1.0000	0.9171
Sitting	0.7243	0.9114	0.8072
Standing	0.1343	0.0664	0.0889
Walking	0.3176	0.8192	0.4577
Waving	0.3415	0.6679	0.4519

29. Interpretation of Class-Wise Metrics

Overall accuracy alone can hide differences among individual classes. Precision measures how often predictions assigned to a class are correct, recall measures how much of that class is successfully found, and F1-score balances the two. Reporting these measures provides a clearer description of a multi-class action recognition system.

Reading achieved 0.8469 precision, 1.0000 recall and 0.9171 F1, indicating strong recognition in the held-out evaluation. Sitting also performed strongly with 0.7243 precision, 0.9114 recall and 0.8072 F1. Walking and Waving showed useful recall but lower precision, indicating that some predictions for those classes were confused with other activities.

Dancing had high precision but low recall, showing that predictions labeled Dancing were often correct while many actual Dancing

sequences were not detected. Hands Up and Standing had relatively low overall classwise scores. Jumping and Lying Down had zero values in the final LSTM evaluation, which is an important limitation rather than a result to hide.

30. Error and Generalization Factors

Recognition performance can change when the testing environment differs from the training environment. Camera angle can change the apparent body configuration. Subject distance can affect landmark visibility. Lighting can affect the quality of pose detection. Body orientation, occlusion and individual movement style can also change the extracted features.

Some activities may overlap during transitions. A person moving from Standing to Walking, for example, can produce intermediate poses that do not clearly belong to one static category. Similar ambiguity can occur between Sitting and Lying Down or between different arm movements. These cases explain why real-time recognition should be interpreted with awareness of the conditions under which the model was developed.

31. Existing System vs Proposed System

Existing/manual approach: Human observation or isolated-frame processing is used to determine the activity. Manual observation requires continuous attention, while isolated-frame recognition cannot directly represent the progression of movement.

Proposed approach: The system automatically extracts body pose, creates a standardized numerical representation, provides a frame level ML baseline, adds a temporal DL model, evaluates the models on heldout data and exposes the trained LSTM through SmartMotion.

The proposed architecture therefore combines automation, structured feature extraction, comparative modeling, quantitative evaluation and real-time presentation. The modular separation also allows individual components to be improved without redesigning the entire concept.

32. Reliability and Evaluation Considerations

The project retains separate held-out data for evaluation. This is important because measuring performance on the same examples used for training can give an overly optimistic view of generalization. Accuracy summarizes the overall proportion of correct predictions, while precision, recall and F1 provide class-level information.

The final reported accuracy values should therefore be read together with the class-wise results. The LSTM's 42.11% accuracy is the final held-out performance of the completed prototype. The 94% Reading value observed during live testing is a single confidence score and should not be presented as model accuracy.

33. Practical Applications

A pose-based action-recognition system can support predefined activity monitoring in smart classrooms, fitness demonstrations, assisted environments, interactive applications, basic surveillance prototypes and human-computer interaction. The current project is restricted to nine trained classes, so these applications represent possible uses of the approach rather than claims of universal activity recognition.

34. System Requirements

Software: Python-based ML/DL environment, MediaPipe Pose, TensorFlow/Keras for LSTM, Scikit-learn for Random Forest, NumPy and Pandas for data processing, Flask for the prediction service and the SmartMotion web frontend. Hardware: a computer capable of running webcam capture, pose estimation and model inference in real time.

35. Design Advantages

- Pose features provide a compact structured input.
- The same representation can support both frame-level and sequence-level models.
- Random Forest provides a simple baseline for comparison.
- LSTM explicitly models temporal information.
- The Flask layer separates inference from the interface.
- SmartMotion provides a practical live demonstration.
- Held-out evaluation provides a more meaningful performance estimate.

29. Conclusion

The Human Action Recognition Using ML and DL project establishes an end-to-end framework for recognizing predefined human activities from webcam video. The system

transforms visual observations into structured pose features using MediaPipe Pose, uses Random Forest as a framelevel Machine Learning baseline and uses LSTM for temporal Deep Learning recognition. The final held-out results are 19.00% accuracy for Random Forest and 42.11% for LSTM.

The comparison demonstrates the value of temporal information for activities whose identity depends on movement across frames. The SmartMotion and Flask integration further demonstrates how the trained model can be used for live prediction. Although the present prototype does not achieve uniform recognition across all classes, it provides a clear technical foundation for improving the dataset, temporal representation and model architecture.

30. Future Scope

- Increase the number and diversity of training samples for all nine action classes.
- Collect more variations in camera viewpoint, lighting, distance, body shape andmovement style.
- Improve the representation and modeling of temporal movement.
- Evaluate stronger sequence-learning architectures and hybrid approaches.
- Add additional human activities after collecting suitable labeled data.
- Use suitable data augmentation and preprocessing to improve generalization.
- Optimize real-time inference and deployment for smoother live recognition.
- Add activity history, analytics and visualization features.
- Explore smart classroom, fitness, assisted-living and human-computer interactionuse cases.

31. Project Contribution

The project contributes a complete prototype rather than only an isolated classification model. It includes data preparation, pose landmark extraction, a common 132-feature representation, framelevel ML classification, temporal sequence construction, LSTM-based prediction, held-out evaluation and web integration. The SmartMotion demonstration connects the trained model to a usable realtime interface.

32. Final System Workflow

Input: Webcam video → Pose: MediaPipe Pose, 33 landmarks → Features: 132 values/frame → Temporal Window: 30 frames → Model: LSTM → Output: one of 9 action classes → Interface: SmartMotion.

33. Reference-Structure Coverage

The mentor reference paper is organized into Abstract, Keywords, Introduction,

Literature Review and Existing System, Objectives and Scope, System Architecture and Methodology, Key Processes, Modules Description, Results and Discussion, Conclusion, Future Scope and References. This paper follows the same major organization while adapting the subject from automated bus scheduling to human action recognition.

The reference also emphasizes the problem addressed, limitations of the existing approach, objectives, scope, architecture, modular design, operational steps, testing, measurable results, discussion, conclusion and future enhancements. These point categories have been retained and expanded with project-specific technical information.

34. Acknowledgement

The authors acknowledge the guidance and support of the project mentor and the Department of Computer Science and Engineering, RYMEC, Ballari, for providing the academic environment and resources required to complete this project.

42. Reference-Structure Coverage

The mentor-provided reference paper uses a clear project-paper structure: Abstract,

Keywords, Introduction, Literature Review and Existing System, Objectives and Scope,

System Architecture and Methodology, Key Processes, Modules Description, Results and Discussion, Conclusion, Future Scope and References. The present paper retains these major sections and adapts them to Human Action Recognition.

The same point categories are carried through the project-specific discussion: the problem is defined; existing/manual approaches are discussed; objectives and scope are stated; the architecture and methodology are explained; operational steps and modules are described; results are reported and interpreted; limitations are identified; and future improvements are proposed.

43. Final System Workflow

Input: Webcam video
↓
Pose Estimation: MediaPipe Pose — 33 landmarks
↓
Feature Extraction: 4 values per landmark = 132 features/frame
↓
Temporal Preparation: 30 consecutive frames
↓
Deep Learning: LSTM
↓
Classification: 1 of 9 predefined actions
↓
Presentation: SmartMotion Web Interface

44. Final Technical Summary

Classes: 9 — Standing, Sitting, Hands Up, Walking, Jumping, Waving, Reading, Dancing, Lying Down.
Pose: 33 landmarks; x, y, z and visibility; 132 features per frame.
Sequence: 30 frames × 132 features for LSTM.
Training composition: 8,482 final samples.
LSTM data: 7,839 training sequences and 2,439 test sequences.
Random Forest evaluation: 2,700 held-out frames; 19.00% accuracy.
LSTM evaluation: 2,439 held-out sequences; 42.11% accuracy.
Live demonstration: Reading with 94% confidence in one demonstration; this is not overall accuracy.

45. Acknowledgement

The authors acknowledge the guidance and support of the project mentor and the Department of Computer Science and Engineering, RYMEC, Ballari, for providing the academic environment and support required to complete this work.

46. References

[1] Breiman, “Random Forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[2] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9,no. 8, pp. 1735–1780, 1997.
[3] Google, “MediaPipe Pose Landmarker,” Google AI for Developers documentation.[4] TensorFlow, “Long Short-Term Memory and recurrent neural network operations,” TensorFlow documentation.
[4] Scikit-learn Developers, “Random Forest Classifier,” Scikit-learn documentation.
[5] Num Py Developers, “Num Py documentation.”
[6] Pandas Developers, “Pandas documentation.”[8] Pallets Projects, “Flask Documentation.” [9] Human Activity Pose Dataset v4, CC-BY-4.0.
[7] M. Nagaraj et al., “Automated Bus Scheduling and Route Management System for Educational Institutions,” mentor-provided reference paper.

47. Closing Discussion

The completed prototype demonstrates the full path from raw webcam input to a machine-learning decision. The structured pose representation makes the learning pipeline compact, while the Random Forest and LSTM comparison illustrates why model choice should reflect the nature of the data. The final results are reported transparently, including the classes where the model remains weak.

For a major-project base paper, this combination of problem definition, methodology, architecture, module description, quantitative evaluation, limitations and future scope provides a complete account of the implemented system while maintaining the organization of the mentor reference.