# A Mobile Feedback Framework for Pose-Based Cricket Fielding Technique Analysis

Himashi Kodithuwakku
*Informatics Institute of Technology*
Colombo, Sri Lanka
*University of Westminster*
London, England
himashi.20210243@iit.ac.lk

Nishantha J. Chandrasena
*Informatics Institute of Technology*
Colombo, Sri Lanka
nishantha.w@iit.ac.lk

***Abstract*—Cricket fielding analysis is a fine-grained sports-action recognition problem because several techniques contain similar upper-body configurations but differ in posture transition, hand position, and movement timing. This paper proposes a mobile feedback framework for pose-based cricket fielding technique analysis. The framework connects video capture, prototype processing, pose extraction, fixed-length landmark representation, hybrid CNN–LSTM inference, confidence reporting, and recommendation delivery. MediaPipe Pose represents each video as a temporal landmark sequence, and the classifier recognizes five techniques: High Catch, Orthodox Cup, Reverse Cup, Short Barrier, and Long Barrier. The prototype is evaluated through functional, timing-related, and model-performance evidence. Beyond reporting model accuracy, the paper clarifies the architectural bridge between isolated action-recognition models and deployable mobile feedback systems, while identifying the additional ablation, deployment-latency, biomechanical-feedback, and user-study evidence required for stronger empirical validation.**

***Index Terms*—pose-based action recognition, cricket fielding, mobile feedback framework, CNN-LSTM, sports analytics**

## I. Introduction

Fielding is a decisive component of cricket performance because catching, stopping, and body-positioning mistakes can directly affect match outcomes. From a computational perspective, cricket fielding is a challenging fine-grained action-recognition problem because the relevant movements are short, rapid, and often visually similar, while hand position, body height, and posture transition can change across a short video sequence. These properties make single-frame analysis insufficient and motivate temporal pose modeling.

A preliminary user survey supported the practical relevance of the problem. Most respondents considered fielding important, many practiced fielding only occasionally, and a large proportion received fielding feedback rarely or never. In addition, 75.2% of respondents indicated that an application capable of detecting fielding technique from video and providing feedback would be useful. The problem addressed in this paper is how to classify cricket fielding technique from short mobile videos and convert the result into feedback that players can understand during practice.

The research gap is that existing cricket analytics studies have focused mainly on batting, bowling, or match statistics, while fielding-technique analysis remains less developed. Related pose-estimation and CNN–LSTM studies show that body landmarks can support sport and exercise classification, but they do not provide an integrated workflow for cricket fielding that combines mobile capture, pose preprocessing, temporal classification, confidence reporting, and recommendation delivery.

To address this gap, this paper proposes Fielding Master, a mobile feedback framework for pose-based cricket fielding technique analysis. The framework accepts a recorded or uploaded fielding video, extracts MediaPipe Pose landmarks, converts the clip into a fixed-length $100 \times 33 \times 3$ landmark tensor, classifies the technique using a hybrid CNN–LSTM model, and returns a technique label, confidence percentage, and fielding-specific recommendation message through the mobile interface.

The work contributes a fielding-specific pose-recognition pipeline, a hybrid CNN–LSTM architecture for local pose patterns and temporal movement transitions, and a mobile feedback workflow for returning interpretable results. It also reports prototype-level validation covering functional requirements, timing-related requirements, and model performance.

The remainder of the paper is organized as follows. The Background and Related Work section reviews pose-estimation and sports-action recognition studies. The System Design and Architecture section presents the requirements, architecture, workflow, and feedback layer. The Machine-Learning Subsystem section describes pose representation, augmentation, and the CNN–LSTM architecture. The Implementation section summarizes the prototype, and the Experimental Evaluation section reports validation results. The paper then presents the Discussion, Future Work, and Conclusion sections.

## II. Background and Related Work

Markerless pose estimation is increasingly used in sports systems because it converts video frames into compact body-landmark representations. Fukushima et al. [5] validated OpenPose-based pose estimation against a VICON marker system across athletic movements, showing the potential of markerless motion capture for sports contexts. Siddiqui et al. [1] used MediaPipe keypoints for cricket batting-stroke

classification, demonstrating that pose-based features can support cricket analytics. MediaPipe and BlazePose also provide relevant foundations for real-time perception pipelines and body-pose tracking in lightweight application settings [8], [9]. Their work, however, focused on batting rather than fielding.

CNN–LSTM models are commonly used when systems must learn both spatial configurations and temporal motion. Azmi Ulya et al. [6] used MediaPipe keypoints with CNN–LSTM layers for elderly exercise classification, while Vats and Mehta [3] applied a 3D-CNN–LSTM model to gym workout recognition. Emmenegger et al. [4] developed a video-based fencing classification and correction system, and Lobo et al. [2] explored yoga-pose correction using MediaPipe and machine learning. These systems show that action classification can be connected to corrective feedback, which is the central systems goal of Fielding Master.

Existing cricket analytics studies have mainly emphasized batting, bowling, or match statistics. The fielding problem requires a different system design in which players must be able to capture short practice videos, submit them for analysis, receive an interpretable technique label, and obtain feedback quickly enough to support repeated practice. The gap is not only the absence of a fielding classifier, but also the lack of an integrated mobile workflow for fielding-specific feedback.

Table I summarizes the main gap in representative pose-based and sports-analysis systems. The comparison highlights that related work usually focuses on a single task such as batting-stroke analysis, pose correction, workout recognition, or fencing feedback. Table II then summarizes how the proposed framework addresses this gap by connecting cricket-fielding recognition with mobile capture, temporal landmark modeling, confidence reporting, and recommendation delivery.

TABLE I
RELATED SYSTEMS AND REMAINING GAP

| Study/System | Domain | Main Gap for This Study |
|---|---|---|
| Siddiqui et al. [1] | Cricket batting | Uses pose features for cricket stroke analysis, but does not address fielding actions, fielding-specific technique labels, or player-facing mobile feedback delivery. |
| Lobo et al. [2] | Yoga | Demonstrates corrective pose feedback, but the movements are static or controlled and differ from rapid catching and ground-fielding actions. |
| Vats and Mehta [3] | Gym workout | Applies CNN–LSTM recognition to exercise movements, but does not provide cricket-specific recommendations or a mobile practice workflow. |
| Emmenegger et al. [4] | Fencing | Connects sport movement classification with correction, but the domain, technique vocabulary, and mobile video-capture needs differ from cricket fielding. |

TABLE II
PROPOSED FRAMEWORK SUMMARY

| Aspect | Proposed Framework |
|---|---|
| Domain | Cricket fielding covering High Catch, Orthodox Cup, Reverse Cup, Short Barrier, and Long Barrier. |
| Input representation | MediaPipe $100 \times 33 \times 3$ temporal landmark tensor generated from short uploaded or recorded fielding videos. |
| Model focus | Hybrid CNN–LSTM recognition of spatial posture cues, hand-position changes, body lowering, and temporal movement patterns. |
| System focus | Mobile capture, preview, authentication, prototype processing, invalid-video handling, and timing targets. |
| Feedback output | Technique label, confidence percentage, and fielding-specific recommendation message presented to the player after inference. |
| Validation scope | Functional verification, timing checks in the local prototype environment, and class-wise model-performance analysis. |
| Gap addressed | Connects cricket-fielding recognition with an end-to-end mobile feedback workflow that supports repeated practice by players and coaches. |

## III. SYSTEM DESIGN AND ARCHITECTURE

### A. Prototype Requirements

The proposed framework requires a mobile device with a working camera or access to stored fielding videos and sufficient storage for short practice clips. The processing environment requires Python-based video handling, OpenCV for frame extraction and resizing, MediaPipe Pose for landmark detection, and a trained CNN–LSTM model for inference. These requirements define the minimum prototype setup needed for local frame processing, landmark extraction, prediction, and feedback delivery.

The functional requirements define the minimum user workflow. The system must allow a player to upload an existing fielding video, record a new video using the mobile camera, preview the selected or recorded input before analysis, and create an account and log in. The analysis pipeline must accept the submitted video, predict the fielding technique, display a confidence percentage, and generate feedback and recommendations according to the prediction result. The main non-functional requirements are practical response time, invalid-video handling, usability for non-technical users, maintainable separation between mobile and machine-learning components, and basic account protection for uploaded videos and prediction results.

The model architecture also influenced these requirements. Because the mobile application sends short fielding clips rather than still images, the inference component must preserve temporal order while keeping the input size fixed. The processing workflow therefore converts each clip into a $100 \times 33 \times 3$ landmark tensor before classification. This representation allows the CNN–LSTM model to separate short-range pose patterns, such as hand placement or body height, from longer movement

transitions, such as lowering into a barrier or completing a catching motion. The architecture was therefore selected not only for accuracy, but also for compatibility with a repeatable mobile feedback pipeline.

The data sent between components is intentionally lightweight after preprocessing. The mobile side initially submits the selected or recorded video with the user/session context and prediction request. The processing layer then extracts frames, detects pose landmarks, normalizes the coordinates, and builds the fixed-length landmark tensor. After inference, the full video does not need to be returned to the mobile interface. Instead, the response can be represented as a compact result object containing the predicted class, confidence percentage, and recommendation text. This separation reduces unnecessary data transfer, keeps the user-facing response simple, and allows the trained model or recommendation rules to be updated without changing the mobile screens. It also supports invalid-video handling because the processing layer can return an error message when landmarks are missing, the clip is unclear, or the frame sequence cannot be converted into the required tensor.

### *B. Architecture and Workflow*

Fielding Master follows a tiered mobile feedback architecture designed around the implemented application workflow. The presentation tier provides the mobile user interface. The processing tier represents the prototype workflow used to receive video inputs, invoke preprocessing and inference steps, and format the response. Authentication is included as a prototype feature rather than as a production security implementation. The machine-learning tier performs pose extraction and CNN–LSTM prediction. The data tier stores user-related information, trained model weights, and fielding-class recommendations.

As summarized in Fig. 1, the workflow links mobile capture, pose extraction, temporal modeling, and feedback delivery. The player records or uploads a short fielding video, the mobile application passes the video to the prototype processing layer, frames are extracted and resized, MediaPipe Pose obtains 33 body landmarks per frame, the fixed-length pose tensor is passed to the trained CNN–LSTM classifier, and the predicted technique, confidence value, and recommendation message are returned to the mobile interface.

Fig. 2 shows how these functions are organized within the mobile application architecture. The architecture is therefore described as a mobile feedback framework rather than as a production deployment because the prototype demonstrates workflow-level connection between the mobile screens, processing pipeline, model inference, and feedback mapping. This separation of concerns supports maintainability because the mobile interface can evolve without changing the classifier, while the model can be retrained or replaced without redesigning the entire user interface.

The mobile frontend was implemented using React Native so that players could interact with the system through familiar phone-based screens for registration, video selection, camera recording, result viewing, and navigation. A prototype processing workflow was used to connect the mobile screens, authentication flow, and model-inference functions during development. OpenCV was used for video processing, MediaPipe Pose for landmark extraction, and Python-based machine-learning libraries for model training and inference. Android Studio Emulator and Expo Go supported mobile testing during development.

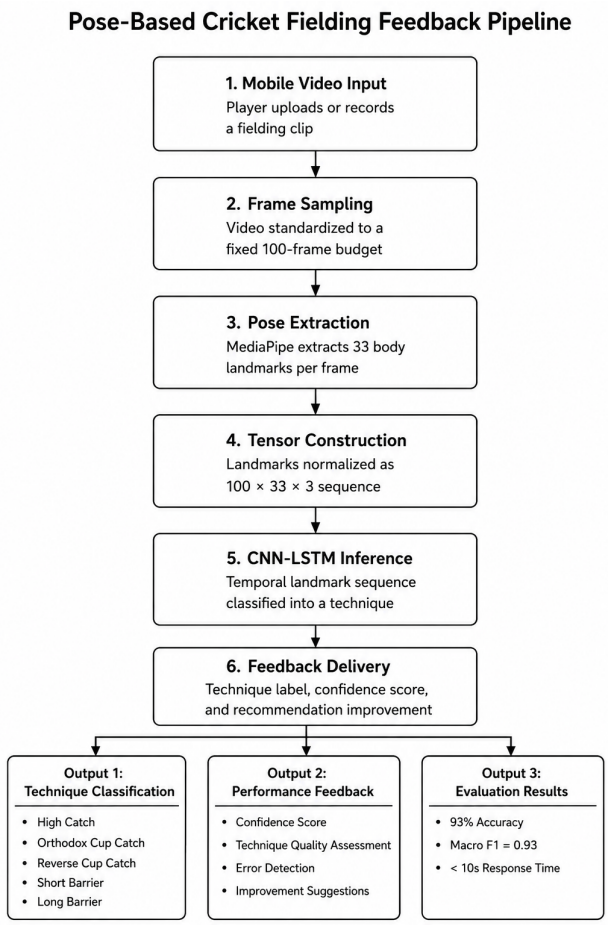


Fig. 1. End-to-end pose-based cricket fielding feedback pipeline used in the proposed framework.

### *C. Feedback and Recommendation Layer*

The feedback layer converts the classifier output into information that is useful during practice. After the CNN–LSTM model predicts a fielding class, the system returns three user-facing elements, namely the predicted technique label, a confidence percentage, and a recommendation message linked to the predicted class. This structure prevents the output from being only a technical class name and instead presents it as a short coaching-style response.

In the current prototype, recommendations are class-level messages rather than full biomechanical corrections. For catching techniques, the feedback can remind the player to maintain hand position, watch the ball, and stabilize the body before contact. For barrier techniques, the feedback can emphasize

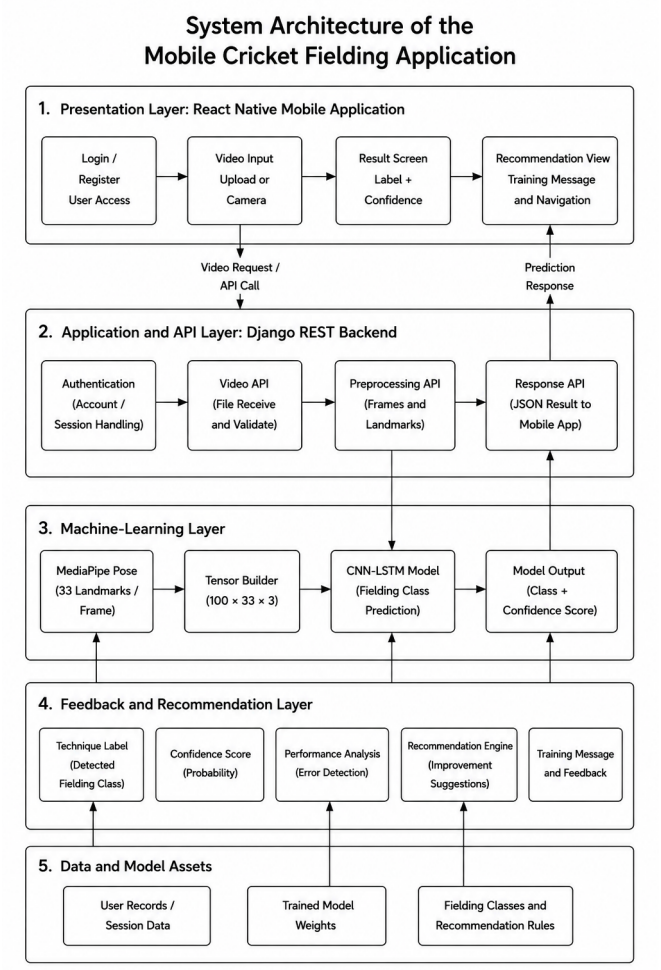


Fig. 2. System architecture of the mobile cricket fielding feedback application.

lowering the body, aligning the legs, and keeping the hands ready to stop the ball. The confidence value also supports interpretation because a high-confidence result can be treated as a stronger recognition outcome, while a lower-confidence result indicates that the video may need to be recorded again from a clearer angle or with better lighting. This feedback layer is therefore central to the framework because it connects pose-based recognition with practical fielding improvement.

## IV. Machine-Learning Subsystem

### A. Dataset and Classes

The classification subsystem was trained on a custom cricket fielding dataset because no suitable public fielding-technique dataset was available. The dataset contained five foundational fielding classes, namely High Catch, Orthodox Cup, Reverse Cup, Short Barrier, and Long Barrier. These classes were selected because they represent common catching and ground-fielding actions used during cricket practice, while also containing visually similar movements that make classification non-trivial. The catching classes require the model to distinguish hand position and upper-body configuration, whereas the barrier classes require attention to body height, leg alignment, and stopping posture.

Data came from participant-recorded fielding clips and expert-validated online cricket videos. The participant recordings provided controlled examples of the target techniques, while online videos increased variation in camera angle, background, player appearance, and movement speed. A cricket coach validated the selected class definitions and helped confirm that the collected examples represented the intended techniques. This validation was important because incorrect class labels would directly affect both classifier training and the usefulness of the generated feedback.

Each class contained approximately 250 short video clips, giving a total dataset size of about 1,250 clips. The clips were kept short so that each sample focused on one fielding action rather than a long sequence containing unrelated movement. The dataset was divided into training, validation, and test subsets using an 80/10/10 split. The training set was used to learn the CNN–LSTM parameters, the validation set supported model selection during experimentation, and the independent test set was used to report final classification performance.

### B. Pose Representation

Each video is converted into a fixed-length pose sequence before classification. MediaPipe Pose extracts 33 body landmarks from each frame, and each landmark is represented by $(x, y, z)$ coordinates. A fielding clip is therefore treated as a temporal sequence of body configurations rather than as a set of raw image pixels. This representation reduces dependence on background, lighting, and clothing compared with RGB video, while preserving the posture and movement information needed for fielding classification.

Let $P_t = \{(x_{t,j}, y_{t,j}, z_{t,j})\}_{j=1}^{33}$ denote the full set of landmarks extracted at frame $t$, where $j$ indexes the body landmark. The first processing step normalizes each landmark coordinate so that the classifier focuses on relative body posture instead of the player's absolute position in the frame. The normalized landmark $\hat{p}_{t,j}$ is computed as follows.

$$\hat{p}_{t,j} = \frac{p_{t,j} - p_{t,c}}{s_t + \epsilon}, \tag{1}$$

where $p_{t,j}$ is the original coordinate vector of landmark $j$ in frame $t$, $p_{t,c}$ is the body reference point, taken as the hip-center location, $s_t$ is a body-scale estimate such as torso or shoulder–hip distance, and $\epsilon$ is a small constant used to avoid division by zero. Subtracting $p_{t,c}$ centers the pose around the player, while dividing by $s_t$ reduces the effect of camera distance and player size.

Because uploaded or recorded clips can contain different numbers of frames, the second processing step converts every clip to a common temporal length. For a clip with $T$ extracted frames, temporal sampling maps the original sequence to a fixed length $L = 100$ as follows.

$$X_i = \hat{P}_{\lfloor i(T-1)/(L-1) \rfloor}, \quad i = 0, \ldots, L-1. \tag{2}$$

Here, $X_i$ is the $i$th sampled normalized pose frame used by the model, $\hat{P}$ is the normalized pose sequence, and the

floor operator selects the nearest corresponding frame from the original clip. This equation preserves the order of the movement while producing the same input size for every example. The resulting tensor $X \in \mathbb{R}^{100\times33\times3}$ is used as the model input, where the dimensions correspond to 100 sampled frames, 33 landmarks, and three coordinate values per landmark.

### C. Augmentation and Hybrid CNN–LSTM Architecture

To address limited training data, six keypoint-level augmentation methods were used in the training pipeline, namely Gaussian noise injection, random joint dropout, temporal jitter, spatial scaling, horizontal flipping, and sequence resampling. These operations simulate landmark noise, occlusion, camera-distance variation, left–right orientation changes, and timing differences. The augmented landmark tensor is then passed to a hybrid CNN–LSTM architecture. The CNN block first acts as a feature extractor over the normalized pose sequence, while the LSTM block receives the extracted feature sequence and models the order of movement across the 100 sampled frames. A dense classification layer with softmax activation finally maps the learned representation to the five fielding classes.

The architecture is suitable for fielding because each class combines posture and motion. For example, High Catch and cup-catching techniques depend strongly on hand position and upper-body configuration, whereas Short Barrier and Long Barrier also depend on body lowering, leg alignment, and the transition into the stopping posture. A CNN-only model may capture local landmark patterns but can miss how those patterns evolve over time. An LSTM-only model can model sequence order but may be less effective at extracting compact spatial patterns from the landmark tensor. Combining the two components allows the model to learn both types of evidence before producing the final prediction.

The final CNN–LSTM model consists of two 3D convolutional blocks followed by one LSTM layer, as summarized in Fig. 3. The first Conv3D layer uses 32 filters with a $3 \times 3 \times 3$ kernel, same padding, and L2 regularization of 0.001. It is followed by batch normalization, ReLU activation, and MaxPooling3D with pool size and stride $(1, 2, 1)$. The second Conv3D block repeats the same structure with 64 filters. The feature map is then reshaped to $(100, -1)$ so that the temporal dimension is preserved for sequence modeling. Before the recurrent layer, dropout of 0.3 is applied. The LSTM layer contains 64 units and returns only the final hidden representation for classification. The classifier then uses a dense layer with 64 neurons, ReLU activation, L2 regularization of 0.001, dropout of 0.2, and a final dense softmax layer with five output neurons for the five fielding classes.

The convolutional component can be interpreted as learning local filters over neighboring time steps and landmark patterns. In this stage, the model identifies short-range posture and motion cues, such as hand position during catching or body lowering during barrier techniques.

$$h^{(l)} = \sigma(W^{(l)} * h^{(l-1)} + b^{(l)}), \tag{3}$$

where $h^{(l)}$ is the feature representation produced at layer $l$, $h^{(l-1)}$ is the previous layer representation, $*$ denotes convolution, $W^{(l)}$ is the learned filter bank, $b^{(l)}$ is the bias term, and $\sigma$ is a nonlinear activation function. This equation summarizes how the CNN transforms the normalized landmark tensor into higher-level movement features.

The LSTM component then models longer dependencies across the sampled sequence. This is important because fielding techniques are not defined only by a single posture, but also by how the player moves into and out of that posture. The gate operations are summarized as follows.

$$\begin{aligned} i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i), \quad f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \\ o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o), \quad c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \end{aligned} \tag{4}$$

where $x_t$ is the input feature at time $t$, $h_{t-1}$ is the previous hidden state, $i_t$, $f_t$, and $o_t$ are the input, forget, and output gates, and $c_t$ is the memory cell state. The operator $\odot$ denotes element-wise multiplication. The gates control how much new movement information is stored, how much previous motion context is retained, and how much information is exposed to the next layer. The final hidden state is mapped to the five fielding classes through a softmax classifier.

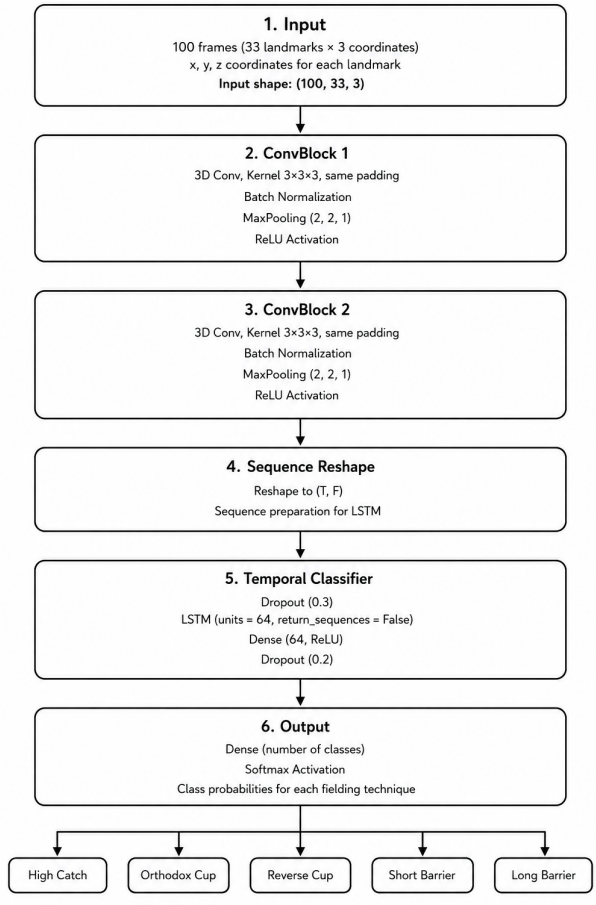


Fig. 3. Hybrid CNN–LSTM architecture used for cricket fielding-technique classification from MediaPipe pose-landmark sequences.

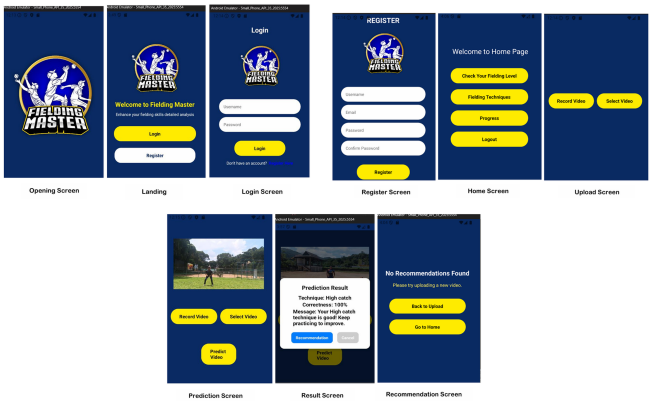


Fig. 4. Representative Fielding Master mobile application screens reconstructed from the updated opening, landing, login, registration, video-input, and result-output views.

## V. Implementation

The mobile interface, illustrated in Fig. 4, supports account registration, login, video selection, camera recording, video preview, prediction request submission, and display of the returned result. This interface design is important for system adoption because the target users include players who may not have technical knowledge of machine learning.

During testing, the prototype connected the mobile screens with authentication, video submission, preprocessing, inference, and result display. The implementation keeps these responsibilities separate so that the interface, pose-extraction stage, classifier, and feedback mapping can be updated independently. The returned output combines the predicted class, confidence percentage, and recommendation message, allowing the model result to be presented as practical coaching support rather than as a label alone.

## VI. Experimental Evaluation

Prototype testing confirmed that the implemented system satisfied the defined functional, timing-related, and model-performance requirements. Users were able to upload videos, record videos, preview selected inputs, register and log in, receive fielding-technique predictions, view percentage output, and obtain feedback recommendations. Module and integration tests also verified mobile-to-processing communication and invalid-video handling through a popup alert. The system satisfied the two high-priority timing requirements defined for the prototype environment: video processing completed within the 3-second target, and prediction results were delivered within the 10-second target. Performance testing was carried out on a laptop with an AMD Ryzen 7 4800H processor and 16 GB RAM, which was sufficient for local frame extraction, pose estimation, and model prediction. The final classifier achieved 93% accuracy on the reported test set, and the macro F1-score was also 0.93, indicating that performance was not dominated by only one or two easier classes. As shown in Fig. 5, High Catch and Long Barrier achieved perfect recall, while Reverse

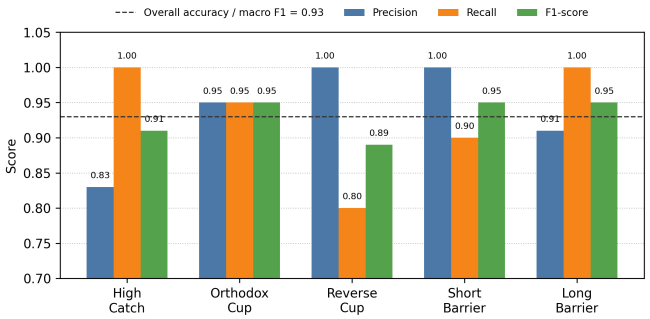


Fig. 5. Class-wise precision, recall, and F1-score of the CNN–LSTM fielding-technique classifier based on the prototype evaluation results.

Cup remained the most difficult class because of its lower recall and visual similarity to other cup-catching techniques.

Overall, the evaluation shows that the prototype met the defined functional workflow, handled invalid-video cases, and satisfied the timing targets in the local test environment. The CNN–LSTM classifier achieved 93% test accuracy, supporting the feasibility of the recognition component within the proposed mobile feedback workflow.

## VII. Discussion

The main engineering value of Fielding Master is the integration of a pose-based classifier into a usable mobile workflow. A standalone model can report accuracy, but a deployable training aid must also handle video capture, user authentication, interface communication, preprocessing, result formatting, and feedback presentation. The layered architecture supports these concerns while keeping the machine-learning subsystem replaceable.

Using prototype processing and inference simplifies mobile experimentation because the model does not need to run directly on the device. In a production deployment, this design would require reliable connectivity, service availability, and end-to-end latency testing. A future on-device version using TensorFlow Lite or other mobile-efficient model designs could reduce latency and improve offline usability, but would require optimization for mobile memory, battery use, and inference speed [10], [11].

Using pose landmarks instead of raw video improves privacy and reduces computational cost, but classification quality depends on accurate landmark extraction. Occlusion, poor lighting, fast movement, or unusual camera angles may reduce prediction reliability. The system should therefore guide users to record videos from suitable viewpoints and lighting conditions.

The dataset includes only five fielding techniques and a limited number of direct participants. Online videos introduce variation in camera angle and recording quality that cannot be fully controlled. The performance results are based on prototype testing rather than large-scale deployment across multiple phones, cloud regions, and mobile-network conditions. Therefore, the current evidence should be interpreted as

prototype-level validation rather than full empirical validation. Stronger evidence would require independent player testing, ablation experiments, realistic end-to-end latency studies, and user evaluation with players and coaches in field-practice settings.

## VIII. Future Work

Future development should expand the class set to include throwing, diving, boundary fielding, and more advanced match scenarios. Ablation studies should compare the hybrid CNN–LSTM model against CNN-only, LSTM-only, and non-augmented baselines to quantify the contribution of spatial-temporal fusion and keypoint augmentation. Future deployment experiments should also measure end-to-end latency under realistic device and network conditions. The system should support progress tracking over repeated sessions so that players and coaches can monitor improvement over time, and on-device inference should be investigated to support offline use. The feedback layer should be improved with joint-angle, velocity, hand-separation, and posture-transition features so that recommendations become biomechanically specific rather than class-level advice. A pilot user study with players and coaches should evaluate usability, training value, and trust in the recommendations using established usability instruments such as the System Usability Scale [12].

## IX. Conclusion

Fielding Master was presented as a mobile framework for cricket fielding technique analysis and recommendation. The work addresses the gap between isolated action-recognition experiments and practical training tools by integrating mobile video capture, MediaPipe pose extraction, temporal landmark representation, CNN–LSTM classification, confidence reporting, and recommendation delivery. This integration is important because fielding improvement requires more than a predicted label; players and coaches also need interpretable output that can support repeated practice and technique correction.

The implemented prototype satisfied the seven high-priority functional requirements and the two timing-related non-functional requirements defined for the local test environment. The classifier achieved 93% test accuracy, indicating that pose-based temporal modeling can recognize the selected fielding techniques with promising performance. These findings suggest that a landmark-based CNN–LSTM model can be engineered into a usable mobile feedback pipeline when the interface, preprocessing stage, inference process, and feedback layer are designed together.

At the same time, the study remains a prototype-level validation rather than a full production deployment. Stronger evidence should be developed through larger and more diverse fielding datasets, independent player testing, ablation studies, realistic latency evaluation, and user studies with players and coaches. Future versions should also extend the recommendation layer with biomechanical measures such as knee flexion, elbow angle, hand separation, and posture transition. Overall, the study demonstrates a feasible foundation for mobile, pose-based cricket fielding support and identifies the next steps needed to move the framework toward broader coaching use.

## References


[1] H. U. R. Siddiqui et al., "Enhancing cricket performance analysis with human pose estimation and machine learning," *Sensors*, vol. 23, no. 15, p. 6839, 2023.

[2] A. Lobo et al., "Yoga correction using machine learning," in *Proc. 2022 Asian Conf. Innovation Technology (ASIANCON)*, 2022, pp. 1–6.

[3] S. Vats and S. Mehta, "Towards enhanced gym safety and efficiency: CNN-LSTM models for identifying leg builder workouts," in *Proc. ICCCNT 2024*, 2024, pp. 1–6.

[4] S. Emmenegger, M. Egli, and M. Pouly, "Mastering fencing techniques with machine learning: A video-based classification and correction system," in *Proc. 10th IEEE SDS*, 2023, pp. 120–127.

[5] T. Fukushima et al., "The potential of human pose estimation for motion capture in sports: A validation study," *Sports Engineering*, vol. 27, no. 1, p. 19, 2024.

[6] A. R. Azmi Ulya et al., "Elderly exercise activity classification based on pose estimation using CNN-LSTM," in *Proc. ISITIA 2024*, 2024, pp. 698–703.

[7] D. J. Vestly et al., "Parametric analysis of a cricketer's performance using machine learning approach," in *Proc. ICICCS 2023*, 2023, pp. 344–348.

[8] C. Lugaresi et al., "MediaPipe: A framework for building perception pipelines," in *Proc. Third Workshop on Computer Vision for AR/VR at CVPR*, 2019.

[9] V. Bazarevsky et al., "BlazePose: On-device real-time body pose tracking," in *Proc. CVPR Workshop on Computer Vision for Augmented and Virtual Reality*, 2020.

[10] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Operating Systems Design and Implementation (OSDI)*, 2016, pp. 265–283.

[11] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv:1704.04861, 2017.

[12] J. Brooke, "SUS: A quick and dirty usability scale," in *Usability Evaluation in Industry*. London, U.K.: Taylor & Francis, 1996, pp. 189–194.

[13] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7291–7299.

[14] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proc. AAAI Conf. Artificial Intelligence*, vol. 32, no. 1, 2018, pp. 7444–7452.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Machine Learning (ICML)*, 2015, pp. 448–456.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2015.

[19] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2014, pp. 568–576.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.