RSET
RAJAGIRI SCHOOL OF
ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

Project Report On

Cricket Sensei: Intelligent Shot Analyzer

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

By

Nandana A Dev (U2103146)

Rohan Raghavan (U2103181)

Rohn Raphael (U2103183)

Shobin Shino Job (U2103196)

Under the guidance of

Mr. Sebin Jose

Computer Science and Engineering
Rajagiri School of Engineering & Technology (Autonomous)
(Parent University: APJ Abdul Kalam Technological University)
Rajagiri Valley, Kakkanad, Kochi, 682039
April 2025

CERTIFICATE

This is to certify that the project report entitled ***"Cricket Sensei: Intelligent Shot Analyzer"*** *is a bonafide record of the work done by* ***Nandana A Dev (U2103146)***, ***Rohan Raghavan (U2103181)***, ***Rohn Raphael (U2103183)*** *and* ***Shobin Shino Job (U2103196)*** *submitted to the Rajagiri School of Engineering & Technology (RSET) (Autonomous) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in "Computer Science and Engineering" during the academic year 2024-2025.*

Mr. Sebin Jose
Project Guide
Assistant Professor
Dept. of CSE
RSET

Ms. Sangeeetha Jamal
Project Coordinator
Assistant Professor
Dept. of CSE
RSET

Dr. Preetha K G
Professor & HOD
Dept. of CSE
RSET

ACKNOWLEDGMENT

We wish to express our sincere gratitude towards **Rev. Dr. Jaison Paul Mulerikkal CMI**, Principal of RSET, and **Dr Preetha K G**, Professor, Head of the Department of Computer Science and Engineering for providing us with the opportunity to undertake our project, "Cricket Sensei: Intelligent Shot Classifier".

We are highly indebted to our project coordinator, **Ms. Sangeeetha Jamal,** Assistant Professor, Department of Computer Science and Engineering, for her valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our project guide **Mr. Sebin Jose**, Assistant Professor, Department of Computer Science and Engineering, for his patience and all the priceless advice and wisdom he has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Nandana A Dev
Rohan Raghavan
Rohn Raphael
Shobin Shino Job

Abstract

CricketSensei is an AI-powered software innovation that aims at revolutionizing cricket coaching through new technologies like body landmarking and automated video analysis. The impact that this project will bring about will be the better batting training experience of cricketers by giving customized feedback on batting technique. Due to the dynamic nature of cricket, traditional methods of coaching seldom achieve the accuracy involved in the delivery of data-driven information in cricket. CricketSensei solves this problem by providing affordable, high-quality coaching to all levels of players beyond geographical or financial limitations.

By using body landmarking technology, it will track how accurate the batsman's posture, swings, and footworks are. Such analysis by CricketSensei can help in pointing out the improvement areas of each player, so that training for each player is tailored according to individual needs. The software will also provide visual feedback by way of extracted keyframes from the training videos such that players may compare their techniques against ideal models. A visual approach to learning will bring enhanced learning and retention in terms of effective batting practices.

CricketSensei is an attempt to make advanced coaching accessible to amateur and professional cricketers alike in an entertaining and easy-to-follow manner, hence helping the game develop and grow overall. In the process of filling the gap between aspiring cricketers and high-quality coaching, CricketSensei will bring in a new wave of skilled cricketers in the cricket training scene.

Contents

List of Abbreviations

1. CNN - Convolutional Neural Network
2. RNN - Recurrent Neural Network
3. VGG - Visual Geometry Group (refers to the network architecture developed by them, e.g., VGG-16)
4. LSTM - Long Short-Term Memory
5. KTH - Kungliga Tekniska Högskolan (Swedish for Royal Institute of Technology, referring to their action video dataset)
6. UCF - University of Central Florida (referring to the UCF-101 action recognition dataset)
7. MSE - Mean Squared Error
8. SSIM - Structural Similarity Index Measure

List of Figures

List of Tables

Chapter 1

Introduction

The "Cricket Sensei: Intelligent Shot Analyzer" project is an advanced tool designed to assist amateur cricketer batsmen in improving their batting skills. The user uploads a video of their shot and the system provides a real-time feedback on various aspects of the play like body posture, timing and swing mechanics. It compares the user's movements to professional cricketing standards using posture estimation and machine learning methods. This personal and data-based feedback enables the knowledge to enhance skill levels, achieving proper shot form, on which the system easily grants expert coaching access, making it all easier and self-led to become performance enhanced in general.

1.1 Background

Cricket is a sport cherished worldwide, with millions of aspiring players looking to improve their skills. However though, there are very few channels for amateur cricketers to have some professional tutors or a personalized teacher that could provide feedback and explain what the mistakes were and how to correct them and perfect shot techniques. This makes in-person coaching too expensive and time-bound, and traditional video analysis itself is not detailed enough to reveal specific biomechanical issues. In cricket performance analysis, emerging technologies like pose estimation and machine learning start to take on importance. Through these innovations, this present initiative democratizes high-quality coaching access that consequently makes an affordable, high-impact cricket training product feasible within an efficient user-friendly interface.

1.2 Problem Definition

Most amateur cricketers do not have access to real-time, personalized feedback that they might need to track and correct errors in their shot techniques. This hinders faster

improvement and wasted opportunities to hone performance. The project proposes developing a system to provide real-time feedback to the cricket players; it focuses more on improving batting technique based on the analysis of body posture and shot mechanics.

1.3 Scope and Motivation

1.3.1 Scope

Cricket Sensei is an app made for cricket batting analysis, from video capture and keyframe extraction to pose estimation and feedback generation. The system is designed to handle different types of cricket shots, offering feedback on stance, bat swing, and body alignment. With an intuitive user interface, it caters to players at different skill levels, ensuring a user-friendly experience. It sets the base for future improvements that could be based on incorporating even more advanced algorithms for pose estimation and expansion into other cricketing skills such as bowling or fielding analysis.

1.3.2 Motivation

This project's motivation comes from the ability to increase the population that can afford and benefit from professional cricket training. Many potential cricket players cannot afford or do not get chances for individual coaching. The gamers will be able to practice independently and make changes based on actionable insights through the automation of the feedback process. In addition, it also touches on the growing interest in technology-driven sports analysis, supports learning by yourself, and enables players to achieve at their best with relatively little time and cost.

1.4 Objectives

- Analyze cricket shots using video input and pose estimation techniques.
- Train a model using a large dataset of professional cricket shots to identify ideal shot poses and errors.
- Provide real-time, personalized feedback on aspects like stance, timing, and bat swing.
- Develop a user-friendly interface for video uploads, feedback delivery, and progress tracking.
- Enhance accessibility to high-quality, data-driven coaching for cricketers worldwide.

1.5 Challenges

- Video Quality Issues: If the user uploads low-quality videos, pose estimation might fail.
- Incorrect Keyframe Detection: Mistakes in identifying the main part of the shot could lead to incorrect feedback.
- Dataset Quality: Inaccurate or incomplete datasets can result in incorrect comparisons.
- Pose Estimation Failures: The model might fail to accurately detect body angles in challenging environments (e.g., bad lighting or occlusions).
- Real-Time Processing: Providing feedback in real-time can be resource-intensive and costly if high server capacity is needed.

1.6 Assumptions

- Clear and Stable Videos: It is assumed that users will upload videos that are stable and have minimal motion blur, ensuring the player's body movements are clearly captured. This is critical for the pose estimation algorithms to accurately detect and analyze body landmarks.

- Consistent Lighting: The project relies on videos recorded in environments with adequate and uniform lighting. Proper illumination ensures that the player's posture and key body points are well-defined, facilitating the precision of pose detection models like Mediapipe or OpenPose.

- Reliable Internet Connectivity: The assumption is that users have access to a stable internet connection. This is necessary for efficiently uploading videos to the server and for receiving timely, real-time feedback on their shot mechanics.

- PKnowledge of Shot Type: The system expects that the user is familiar with the specific type of cricket shot they are trying to analyze and improve. This knowledge allows the feedback to be more targeted and relevant, as different shots have distinct mechanics and postures.

1.7 Societal / Industrial Relevance

The "Cricket Sensei" project holds significant relevance both in society and the sports industry. For society, it democratizes access to professional-level cricket coaching, enabling more players to enhance their skills and pursue opportunities in the sport. In the industrial context, the project showcases the potential of integrating artificial intelligence and biomechanics, opening avenues for collaboration in sports technology and data-driven athlete development.

1.8 Organization of the Report

This report is structured into six chapters to systematically present the development of the project. Chapter 1: Introduction; outlines the background, motivation, objectives, and scope of the Smart Cricket Coach system.
Chapter 2: Literature Survey; reviews previous research and existing systems related to pose detection, video analysis, and sports training, providing insight into the current state of the art and identifying gaps that this project aims to fill.
Chapter 3: Requirements; discusses both hardware and software requirements essential for the successful implementation of the system, along with any constraints or assumptions considered.

Chapter 4: System Architecture; describes the overall design and architectural components of the system, including data flow diagrams and module-level breakdowns.
Chapter 5: System Implementation; explains the practical aspects of building the system, detailing how each component was developed and integrated.
Finally, Chapter 6: Results and Discussion; presents the outcomes of the implemented system, analyzes its performance, and discusses the effectiveness of the solution along with possible improvements.

1.9 Conclusion

In this chapter, the groundworks were laid for understanding the "Cricket Sensei: Intelligent Shot Analyzer" project. The project was begun by emphasizing its significance in providing accessible, high-quality coaching for cricketers through advanced video analysis and pose estimation technologies. The background section highlighted the gap in current training methods and the necessity of a data-driven approach to sports performance analysis. The problem that the project aims to solve was defined, outlining the specific objectives that drive our solution. The scope and motivation further emphasized the project's potential to revolutionize cricket training by offering an intuitive and effective self-guided learning experience.

Additionally, the various challenges that may arise during development were discussed. Finally, the societal and industrial relevance section showcased the broader impact of this work, positioning it as a valuable contribution to both individual athletes and the sports technology industry. This chapter has set the stage for a deeper exploration of the methodologies and technical details that will be covered in the subsequent sections of this report.

Chapter 2

Literature Survey

The field of sports performance analysis, particularly in cricket, has witnessed significant advancements with the integration of artificial intelligence, machine learning, and computer vision. Various researches have focused on using pose estimation and deep learning to analyze and improve sports performance. The present literature survey examines some of the key works and methodologies on which the "Cricket Sensei: Intelligent Shot Analyzer" is based. The work outlines the history of video analysis, with the usage of pose estimation models such as Mediapipe and OpenPose and techniques used for extracting meaningful insights from the movement of the users. There is also the keyframe extraction role, which is a vital step to make video analysis easier and focus only on relevant moments in a player's shot. With this knowledge from existing works and their shortcomings, this project aims to work around current methods and improve them even better so as to make it more accessible and possibly effective for cricket training.

2.1 Novel Keyframe Extraction Method for Video Classification[1]

This paper proposes a new method for improving video classification using Convolutional Neural Networks and Recurrent Neural Networks. While CNNs are excellent at spatial feature extraction in single frames, RNNs are particularly powerful in modeling sequential data; thus, the hybrid model is quite capable of capturing complex spatial-temporal patterns in videos. This work's main contribution is a novel keyframe extraction technique that uses an action template to pinpoint and concentrate on the most instructive areas of video frames while lessening the impact of distracting and dynamic backgrounds. This helps classify only relevant frames, hence drastically improving the overall accuracy. The study harnesses the ability of transfer learning by using a pre-trained CNN architecture like VGG-16 and uses advance sequence models, such as ConvLSTM and LSTM, in

video content analysis. Experiments conducted on the benchmark datasets such as KTH and UCF-101 prove the proposed method's strength and efficiency toward addressing the problems in video analysis and classification tasks.

2.1.1 Keyframe Extraction Using Action Templates

The video keyframe extraction technique is designed to extract the most informative frames of the video. Action area detection identifies active areas and provides additional meaning to video classification. The steps of this method can be defined as:

- **Action Template Identification:** First, each video frame is decomposed, and a central region in each frame is considered to be a candidate for the action area. The frame is divided into three possible templates, or regions, so as to identify the optimal area where actions are occurring. The template with the highest mean squared error (MSE) between consecutive frames is chosen because that area of visual change is higher, likely indicating an action.

$$\text{MSE}(X, Y) = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} [Y(i, j) - X(i, j)]^2$$

- **Action Location Specification:** Once the action template is identified, the exact region of interest in each frame is located by correlating this template across frames. A correlation coefficient is calculated to find the best matching region in each subsequent frame. This correlation process ensures that only relevant areas, where action is detected, are selected, filtering out background and other non-essential regions.

- **Similarity Calculation and Frame Filtering:** Regions of interest in consecutive frames are compared using the structural similarity measure (SSIM) to identify keyframes. Frames are picked based on whether their SSIM is within predetermined ranges that were empirically selected to capture action-related changes. Should there be insufficient frames within this threshold, the similarity range is extended to include additional frames.. The short set of keyframes produced by this selective procedure better captures the video than equally sampled frames.

$$\text{SSIM}(X, Y) = \frac{(2\mu_X\mu_Y + C_1)(2\sigma_{XY} + C_2)}{(\mu_X^2 + \mu_Y^2 + C_1)(\sigma_X^2 + \sigma_Y^2 + C_2)}$$

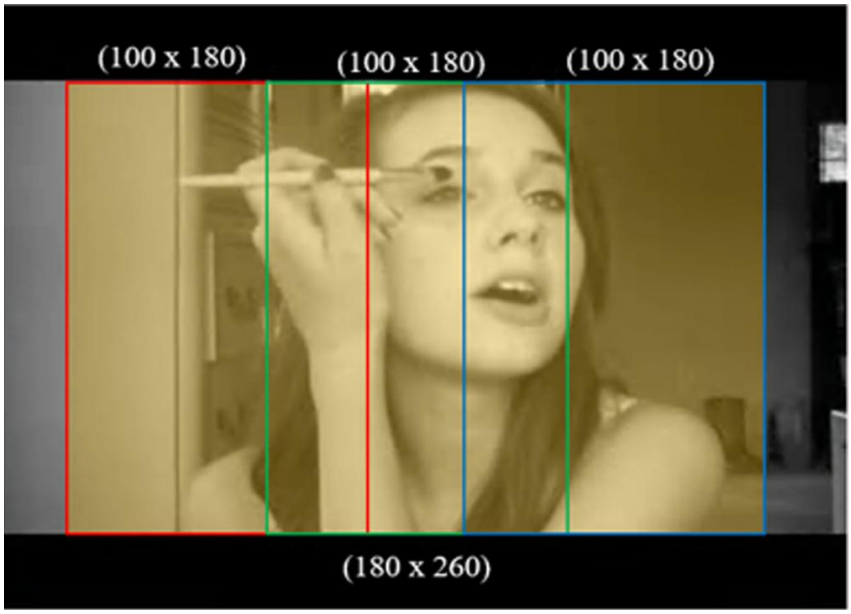


Figure 2.1: Defining an action template's potential locations. Three potential templates' borders are shown by red, green, and blue boxes

2.1.2 Deep Neural Network Architectures for Video Classification

The extracted keyframes are fed into a deep neural network architecture combining CNNs and RNNs, leveraging each network's strengths for video classification:

- **Feature Extraction with VGG-16:** Each keyframe's spatial properties are extracted using the VGG-16 network, which has been pre-trained on the ImageNet dataset. This CNN architecture captures intricate spatial details in each frame, serving as a foundation for understanding the visual content. The extracted features represent both objects and background information within frames, crucial for subsequent temporal analysis.

- **Temporal Analysis with LSTM and ConvLSTM:** The temporal dependencies between frames are analyzed using two variants of RNNs: LSTM and ConvLSTM. LSTM networks are capable of capturing long-term dependencies within sequences by maintaining memory states that can accumulate information over time. ConvLSTM is an extension of LSTM involving convolutional operations within its architecture. The convolutional layers of the ConvLSTM can preserve the spatial correlations across the frames amaking it particularly suitable for spatiotemporal

modelling. These architectures capture the sequential flow of actions that occurs in a video, and as such improve upon the recognition capability of complex activities.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o)$$

$$h_t = o_t \cdot \tanh(c_t)$$

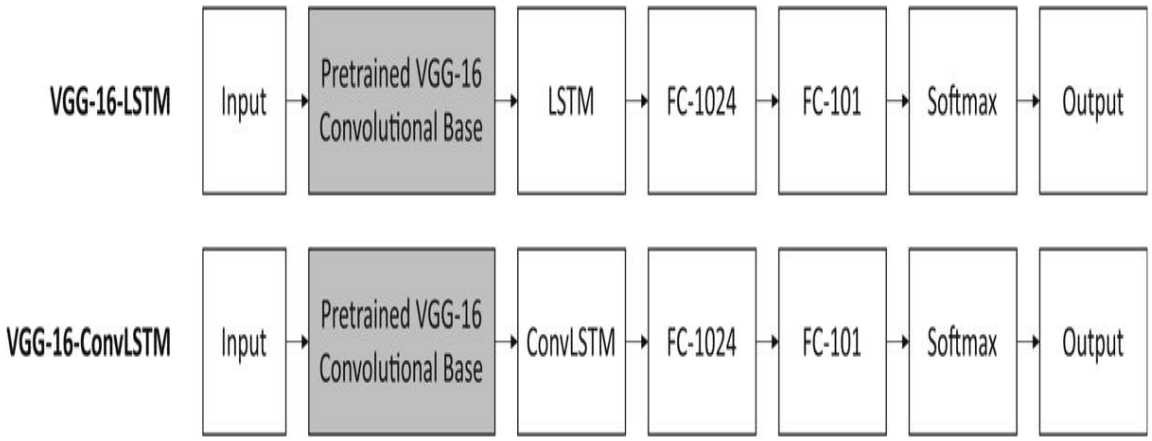


Figure 2.2: The network architectures utilized in the VGG-16-LSTM and VGG-16-ConvLSTM experiments

2.2 Shot-Based Keyframe Extraction Using Edge-LBP Approach[2]

The Binarized Edge Local Binary Pattern or the Edge-LBP approach is designed to capture the most informative frames within each shot by combining edge detection with Local Binary Pattern analysis. This method improves the process of keyframe extraction by focusing on textural and structural information in video frames. This method uses the following steps:

- **Edge Detection:** An edge detection filter is applied to each video frame to highlight the regions with sharp structural transitions. Such regions will be particularly helpful in isolating the significant changes between frames, which typically denote shot boundaries or moments of interest within a shot.

- **BELBP Feature Extraction:** This edge and texture information is then computed and used to encode in Binarized Edge Local Binary Pattern after it detects the edge. BELBP uses binary values in pixels which compare the difference of intensity values among neighboring pixels for edge detection alone:

$$\text{BELBP}_{(P,R)} = \sum_{p=0}^{p-1} S(G_p)2^p$$

where:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Here, g_c is the gray-level value of the central pixel, g_p is the gray-level value of neighboring pixels, P is the number of sampling points, and R is the radius.

- **Frame Similarity Calculation and Keyframe Selection:** Frames within a shot are compared based on their BELBP features. The similarity between frames is calculated using a distance metric, with frames showing the highest divergence selected as keyframes. This ensures that only frames representing unique visual moments within the shot are retained, minimizing redundancy and enhancing the summary.

$$\text{Distance}(X, Y) = \sum [\text{Edge-LBP}_X - \text{Edge-LBP}_Y]^2$$

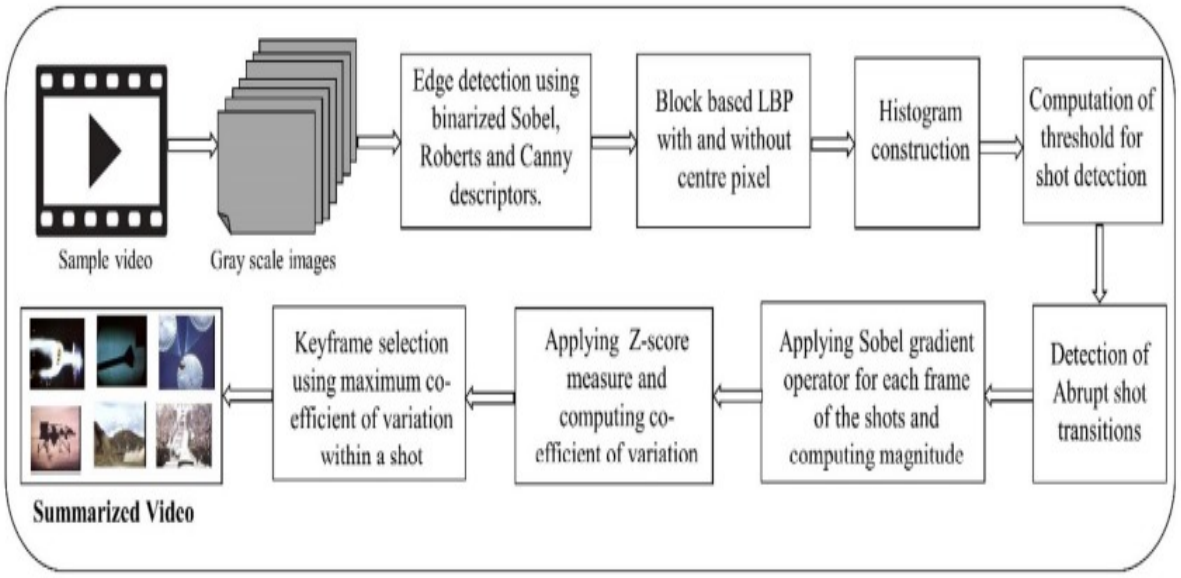


Figure 2.3: Framework of the proposed methodology.

2.2.1 Edge Detection Techniques Used in Edge-LBP

Overview: This section can provide details about the different edge detection methods used (e.g., Sobel, Canny, Roberts) and their specific roles in the Edge-LBP approach. Comparison of Edge Detectors: Explain why Sobel edge detection was particularly effective and provide comparisons of the performance outcomes with other edge detection techniques, which can add depth to the method evaluation. Equation and Filter Application: Include the mathematical formulation of each edge detector used, such as Sobel's gradient function and how it contributes to isolating key visual elements within the frames.

2.2.2 Evaluation of Edge-LBP Keyframe Extraction

The effectiveness of the Edge-LBP method is tested using a benchmark dataset for video summarization, comparing its performance with traditional LBP and edge-only methods:

- **Dataset and Keyframe Extraction Baselines:** The proposed method is evaluated on the SumMe and TVSum datasets, commonly used for video summarization. To compare performance, three methods were considered: (1) Edge-only, which selects keyframes based on edge differences; (2) LBP-only, which focuses solely on texture patterns; and (3) Uniform sampling, which selects frames at regular intervals.

- **Performance Metrics:** Keyframe extraction accuracy is measured by Precision, Recall, and F1 Score, assessing the representational quality of selected frames. These metrics indicate how effectively Edge-LBP captures the most informative frames.
- **Statistical Significance Testing:** To validate the results, a one-way ANOVA test was conducted, followed by Tukey's HSD test, to determine significant differences in performance across methods.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

2.3 Enhancing Cricket Performance Analysis with Human Pose Estimation and Machine Learning [3]

2.3.1 Introduction

This study combines a variety of machine learning and deep learning methods to extract features and classify a dataset. For precise analysis, the videos were pre-processed to have grain-free quality, 720p resolution, and lengths of 1.5–4 seconds. 17 important human movement points were extracted using the MediaPipe library, producing a novel dataset that was then further refined to eliminate noise. The dataset was split into train and test sets for analysis. Models such as LSTM, KNN, logistic regression, decision tree, SVM, and random forest were applied to classify cricket strokes, with hyperparameter tuning optimizing their performance for real-time evaluation.

2.3.2 Performance Matrices

The performance of machine learning algorithms is assessed in this study using a variety of performance matrices. Three common evaluation measures for machine learning classification tasks are precision, recall, and F1 score. In addition to these, the performance of machine learning models is assessed using the geometric mean of the standard evaluation matrix, Cohen's kappa, and log loss.

$$precision = \frac{true\ positives}{true\ positives + false\ positives}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

$$\kappa = \frac{p_0 - p_e}{1 - p_e}$$

$$G\text{-Mean} = \sqrt{\text{Sensitivity} \times \text{Specificity}}$$

$$LogLoss = -\frac{1}{n}\sum_{i=0}^{n}[y_i log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

$$accuracy = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ negatives + false\ positives}$$

Three distinct data splits—70:30, 80:20, and 90:10 for training and testing, respectively—are used to assess each performance metric. According to Table 3.2, the LSTM model based on deep learning has the lowest precision, recall, and F1 score value. When compared to all other machine learning models, the RF has the highest precision, recall, and F1 score across all data splits. On an 80:20 data split, RF obtains the best precision, recall, and F1 score.

Model	Precision			Recall			F1 Score		
	70:30	80:20	90:10	70:30	80:20	90:10	70:30	80:20	90:10
LSTM	0.409	0.452	0.470	0.476	0.502	0.527	0.476	0.502	0.527
LR	0.658	0.655	0.649	0.637	0.643	0.651	0.637	0.643	0.651
DT	0.948	0.951	0.970	0.947	0.956	0.968	0.947	0.956	0.968
SVM	0.863	0.867	0.873	0.842	0.846	0.857	0.842	0.846	0.857
KNN	0.982	0.989	0.988	0.984	0.989	0.989	0.984	0.989	0.989
RF	0.996	0.998	0.996	0.995	0.998	0.997	0.995	0.998	0.997

The RF model also performs better than all other models used on various performance metrics, such as the geometric mean and Cohen kappa score. The RF has the highest geometric mean score and Cohen kappa value on the 80:20 data split. On each data split, LSTM has the lowest geometric mean score and Cohen kappa value. Out of all the deployed machine learning models, the LSTM model has the largest log loss value. RF's log loss on an 80:20 split is 0.076.

2.3.3 Conclusion

The accuracy of k-fold cross-validation based on an 80:20 data split is 0.95, but the accuracy of the RF model is 0.997. Despite the distribution of video data, Random Forest model classified eight cricket shots with the best accuracy. This study emphasizes how new technologies like machine learning and computer vision have a big impact on sports. More precise forecasts of batsman strokes and other crucial elements of cricket can be anticipated as this technology develops. In the future, more strokes will be added to the video library.

2.4 Classifying Cricket Shots using CNN and GRU [4]

The CricShotClassify study is a hybrid deep learning model that combines Convolutional Neural Networks (CNNs) and Gated Recurrent Units (GRUs) for the classification of

cricket batting shots from video data. In this study, the CricShot10 dataset was developed with ten different cricket shots. The spatial feature extraction was performed by CNNs and GRUs that were used to capture the Temporal dependencies which will result in a strong classification architecture for complex cricket actions.

2.4.1 Data Preprocessing and Dataset Design

The first step was creating and preprocessing the CricShot10 dataset, designed for ten different cricket shot types; it was made to contain cover drive and hook shot, among others, all those recorded in various real-world conditions. To ensure that these sequences were split up without extra data added for maintaining temporal coherence, each video sequence was segmented into 15 representative key frames: this is to balance between computing efficiency and detail.

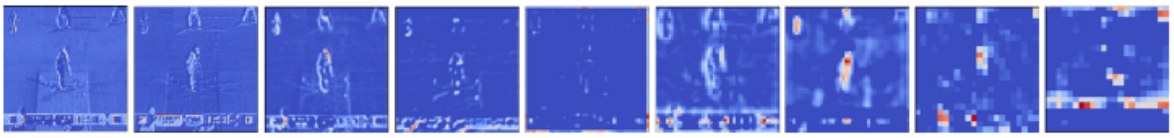
2.4.2 CNN Architecture for Spatial Feature Extraction

The CricShotClassify model uses CNNs to extract spatial features from each frame, including bat angles, body posture, and shot direction. Several pre-trained models were used as a starting point for feature extraction, namely VGG16, InceptionV3, and DenseNet169, since they performed well on general image classification tasks. Transfer learn- ing was applied, with certain layers frozen to retain generalized image features, while specific layers were left trainable to allow the model to learn the unique characteristics of cricket shots in CricShot10. VGG16 was selected as the primary architecture due to its proven performance in capturing essential spatial details for fine-grained classification. The selected model allowed the CNN layers to effectively detect spatial patterns in each frame, which are critical for identifying each shot type accurately.

2.4.3 GRU Architecture for Temporal Feature Extraction

To account for temporal dependencies across frames, a GRU layer was integrated following CNN. GRUs are optimized for sequential data, capturing long-term dependencies while remaining computationally efficient compared to traditional long short-term memory (LSTM) networks. The GRU layer processed the sequential frame-level features generated by the CNN component, analyzing frame progression to distinguish between similar-looking shot types. By understanding the temporal context of the movements,

(a) Sequential video frames



(b) Low-high feature maps

Figure 2.4: Feature extraction

such as bat swing and player posture across frames, the GRU enhances the model's ability to accurately identify shots based on both spatial details and temporal dynamics. Update, reset, and current memory gates are used by GRU. How much of the past should be carried over into subsequent time steps is determined by the model with the aid of the update gate. The model also determines the extent to which the new state value is a copy of the old state value in order to rule out the possibility of a vanishing gradient. The reset gate aids the model in determining how much of its history it should disregard. The current-memory gate calculates the value of the current state by combining the current input and the previous hidden state.

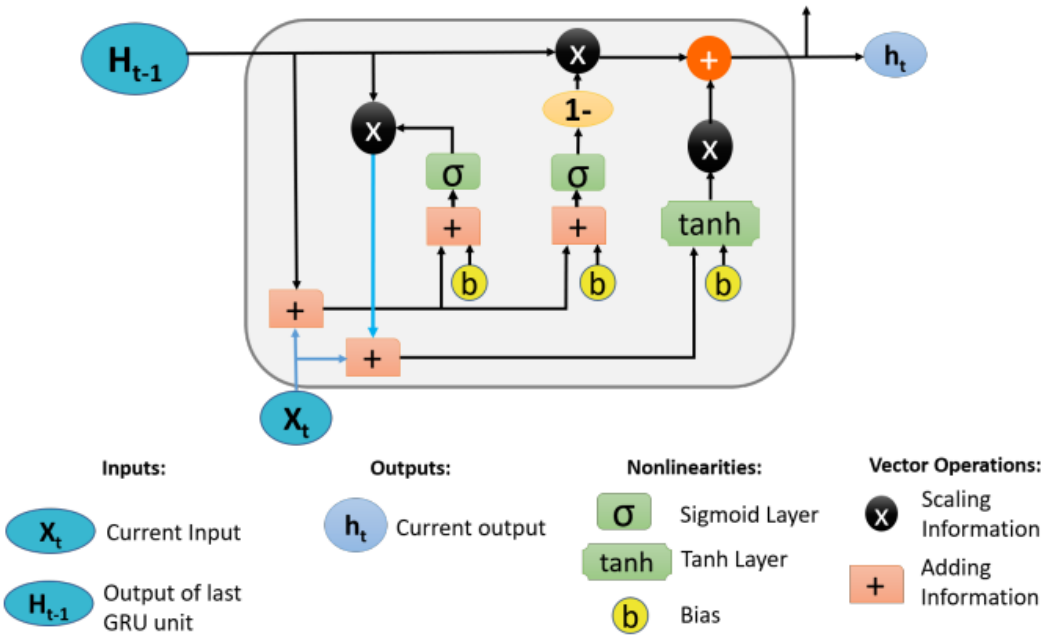


Figure 2.5: GRU Cell.

Equations 2.8 to 2.11 provide the equations for the update gate, reset gate, current memory state, and final memory, respectively.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t] + b)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

2.5 Summary and Gaps Identified

Paper	Advantages	Disadvantages
Novel Keyframe Extraction Method for Video Classification	Enhances classification accuracy by focusing on keyframes; effective on spatial-temporal video data	Limited robustness to dynamic backgrounds and high computational cost for deep neural network models
Shot-Based Keyframe Extraction Using Edge-LBP Approach	Captures both structural and textural information; efficient for keyframe selection	Dependence on edge-detection accuracy; limited generalization to diverse datasets
Enhancing Cricket Performance Analysis with Pose Estimation and ML	Effective in identifying key player movements; high accuracy with RF model	Poor performance of deep learning models (e.g., LSTM) compared to traditional ML; dataset dependency
Classifying Cricket Shots using CNN and GRU	Robust classification of cricket shots; optimized sequential feature learning	High training time; requires large datasets for training

Table 2.1: Summary of Literature Review

2.5.1 Gaps Identified

The following gaps were identified in the reviewed studies:

1. **Dataset Limitations:** Many studies rely on specific datasets that may not cover diverse cricketing scenarios, leading to limited generalizability.

2. **Pose Estimation Challenges:** Existing methods struggle in low-quality video settings or complex poses with occlusions.

3. **Real-Time Processing Issues:** High computational requirements hinder real-time analysis, especially for deep learning models.

4. **Shot Detection Granularity:** Methods often fail to capture subtle differences between visually similar cricket shots.

5. **Limited Integration of Features:** Current systems lack a unified approach combining pose estimation, keyframe extraction, and contextual feedback effectively.

This section highlights the limitations of existing works and provides a foundation for addressing these challenges in the proposed methodology.

Chapter 3

Requirements

3.1 Hardware Requirements

1. Storage: For storing videos, processed frames, and datasets.
2. Processor : Intel core i5 or AMD Ryzen 5
3. RAM : 8GB
4. GPU: NVIDIA GTX 1050 Ti or NVIDIA GTX 1650
5. Disk Space : 6GB

3.2 Software Requirements

Operating System: Windows/Linux/macOS

1. Development Tools: Python, TensorFlow, OpenCV
2. Pose Estimation: Mediapipe
3. Web/Frontend: HTML, CSS, Flask
4. Video Processing: FFmpeg

3.3 Functional Requirements

The key functional requirements of the system being developed are:

1. Enhanced User Feedback: The system should generate clear, informative, and personalized feedback for the user based on their performance. This includes identifying key mistakes in posture or movement and offering constructive suggestions to help improve their cricket shots.

2. Angle Deviation Report: For each key frame in the user's video, the system calculates the angles formed at important joints (e.g., elbow, shoulder, knee) and compares them with the angles from professional reference models. The deviation is then quantified and reported, helping the user understand exactly where corrections are needed.

3. Side-by-Side Comparison: To make the feedback more intuitive, the system provides a visual comparison between the user's recorded pose and the ideal reference pose. The user's frames are displayed alongside a reference video taken from the dataset, helping the user easily identify discrepancies in their body alignment and movement.

4. Pose Recommendation: Based on the identified deviations and performance metrics, the system offers recommendations for correcting the user's pose. These suggestions may include adjusting joint angles, improving stance, or refining motion flow, all tailored to mimic the form of professional cricketers.

Chapter 4

System Architecture

4.1 System Overview

Using the video provided by the user as input, the system calculates improvement by processing the video, identifying required body landmarks, and comparing these with the ideal and optimal landmarks calculated by processing the dataset. The output of the comparator is then passed to the feedback generator, which provides textual and video feedback highlighting the problem areas where improvement is necessary.

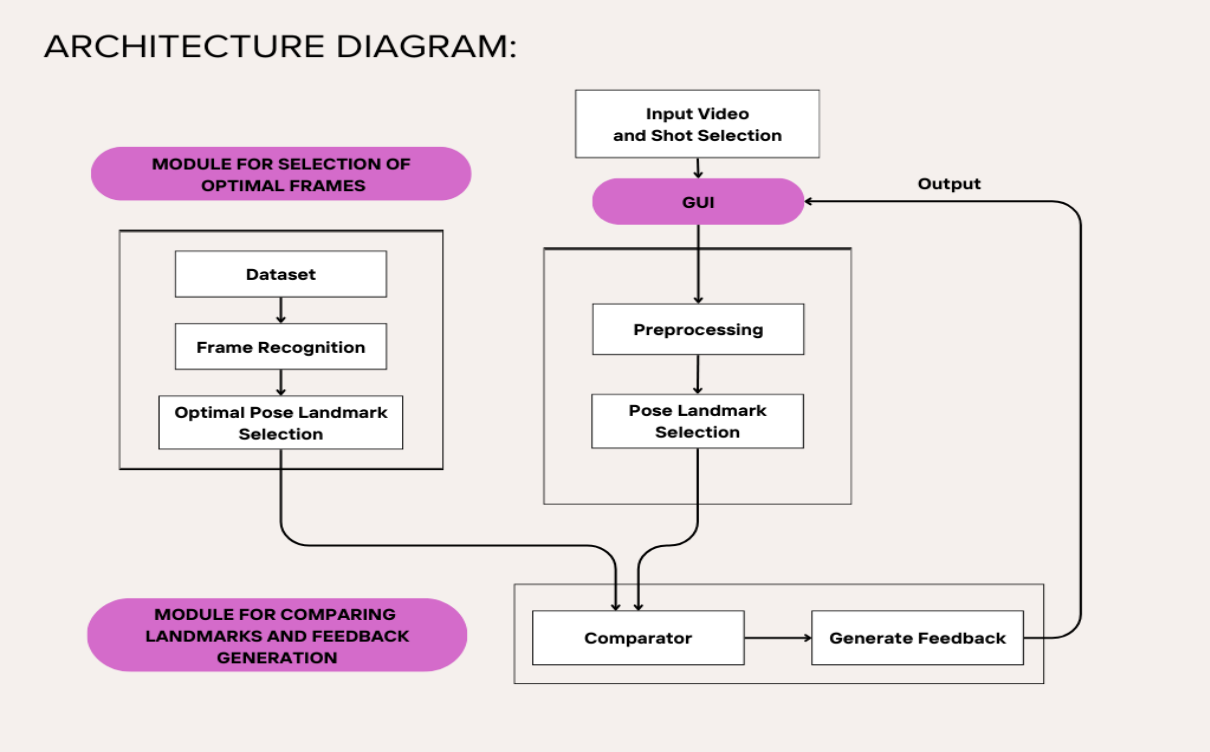


Figure 4.1: Architecture Diagram for CricketSensei

4.2 Component Analysis

4.2.1 Input Video and Shot Selection

The system's main feature is the user-friendly interface that allows users to upload videos of their cricket shots. Users need to ensure that the uploaded video contains only the part of the shot that they want to analyze. This approach helps focus on the key aspects of the shot, making the analysis more efficient and effective.

4.2.2 Preprocessing

The system divides the video into separate frames, eliminates any extraneous noise, and guarantees that every frame is the same size. This fundamental stage guarantees that the video is ready for more in-depth examination, laying the groundwork for precise findings.

4.2.3 Pose Landmark Selection

The system delves into the specifics of biomechanics at this point. It recognizes important bodily landmarks in every frame, including the shoulders, hips, knees, and wrists, using sophisticated algorithms. These landmarks offer a thorough picture of the player's stance and motions, which serves as the foundation for evaluation and comparison.[5]

4.2.4 Module for Selection of Optimal Frames

In addition to analyzing, the system also compares. A carefully selected dataset of expert cricket shots, which is the gold standard, powers this component. The ideal position and motions for every shot phase are determined, and the user's performance is assessed using these as a standard.[6]

4.2.5 Comparator

The actual analysis takes place at this point. The user's pose data is compared to professional players' benchmarks by the comparator. It detects variations in movements, alignments, and angles to identify the precise point at when things deviated off the path. The outcome is a detailed analysis of what's good and what needs improvement.

4.2.6 Feedback Generation

Where the system really excels is in feedback. It explains mistakes rather than merely pointing them out. For example, if the user's follow-through is incomplete or their backlift is too high, the system offers comprehensive, doable recommendations to address these problems. The user receives personalized, useful feedback that is intended to help them get better with each attempt.

4.2.7 Graphical User Interface (GUI)

Users may upload films, view analyzed frames, and see how their performance compares to the experts thanks to the interactive and user-friendly interface. With its clear annotations and side-by-side comparisons, the feedback is displayed visually, making it simple to comprehend and act upon.

4.3 Data Flow Diagram

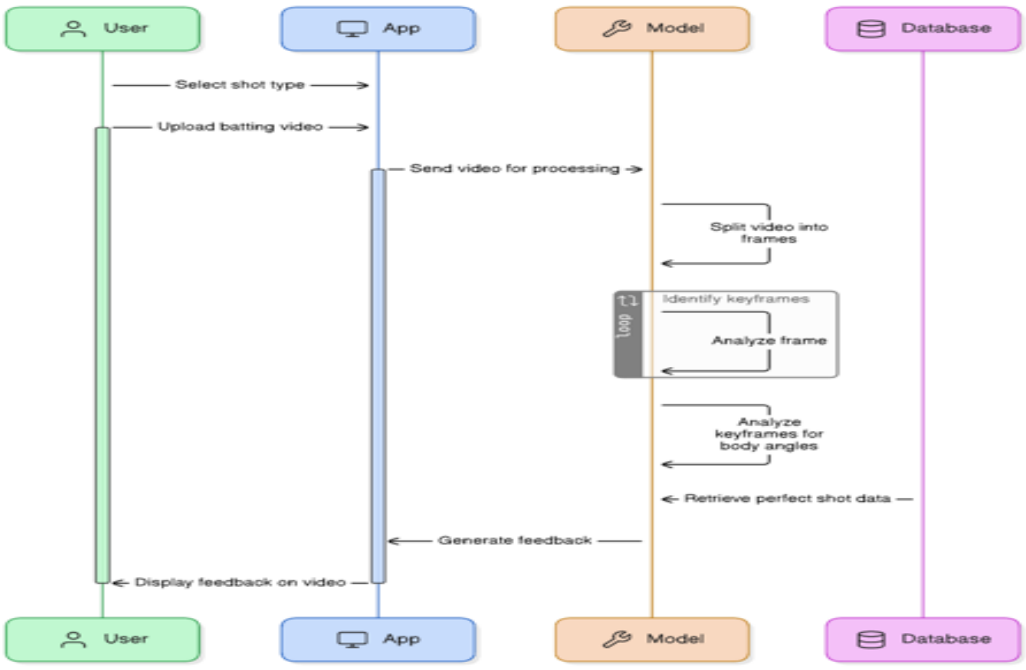


Figure 4.2: Data Flow Diagram for CricketSensei

4.4 Module Division

The system is divided into four distinct modules, each undertaking a key aspect of performance evaluation. It begins with the analysis of user video, where cricket shots are identified. This is followed by the ideal pose estimation module which provides benchmark poses for comparison. The comparison module determines how closely the user shot resembles the benchmark shots. Finally, the feedback generation module translates the results to actionable advice that enables users to refine their batting techniques effectively.

4.4.1 Analysis of User Video

In this module, the user uploads a video of their batting, which is processed to extract its frames. Pose estimation model namely Mediapipe is applied to these keyframes to identify body landmarks and calculate joint angles. This data represents the user's body posture during the shot.

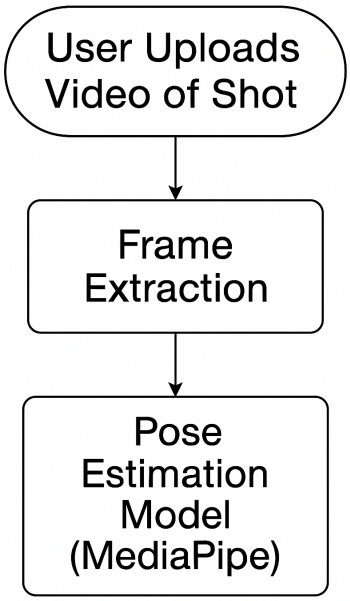


Figure 4.3: Analysis of User Video

4.4.2 Ideal Pose Extraction

A dataset of professional cricketers performing different shots (e.g. drive shot, pull shot) is used to train a machine learning model. The necessary ten frames containing the shot is extracted from each video of the dataset using a script code and is named as keyframes.The model learns the "perfect" body posture and angles for each shot type from these keyframes, which will be used to compare with user data.

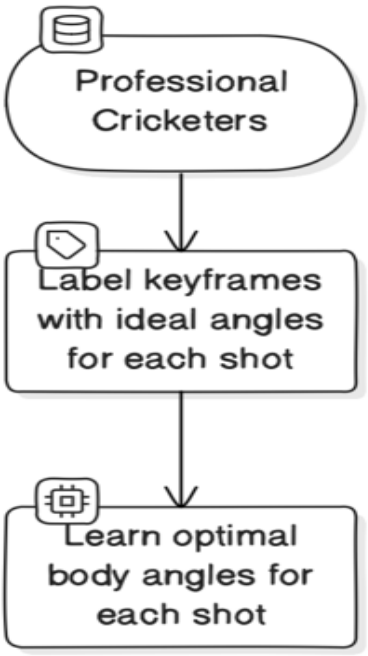


Figure 4.4: Ideal Pose Extraction

4.4.3 Comparison of Poses

The angle parameter data extracted from the user's video are compared to the optimal angles data from the trained model. This module identifies any deviations in the user's stance and suggests improvements by highlighting specific joints or angles that needs adjustment.

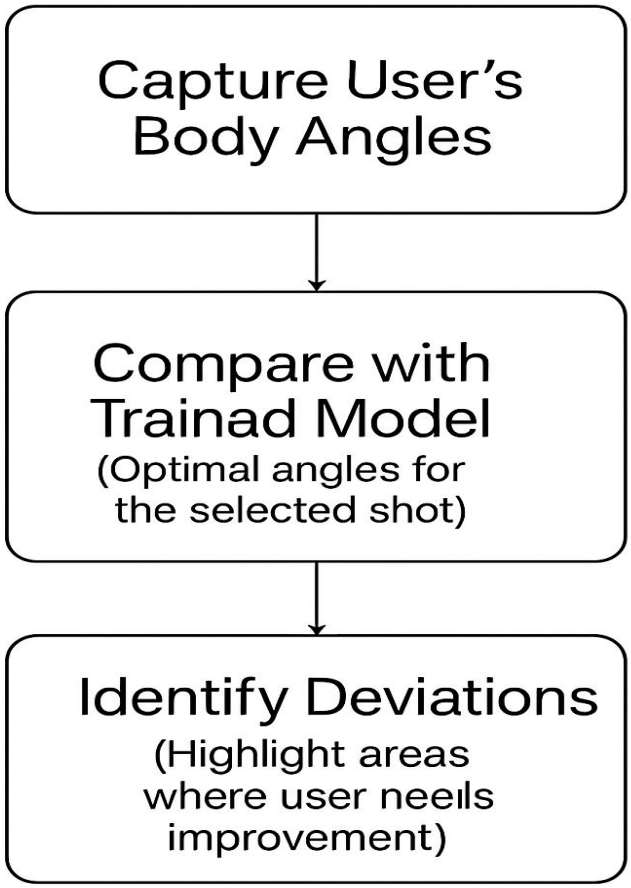


Figure 4.5: Comparison of User and Ideal Pose

4.4.4 Feedback Generation

This module provides feedback to the user based on the comparison. It generates visual feedback on the user's video, showing which parts of their body need to adjust. The feedback can be in the form of highlighted joints, angle adjustments, or text instructions on how to improve their stance.

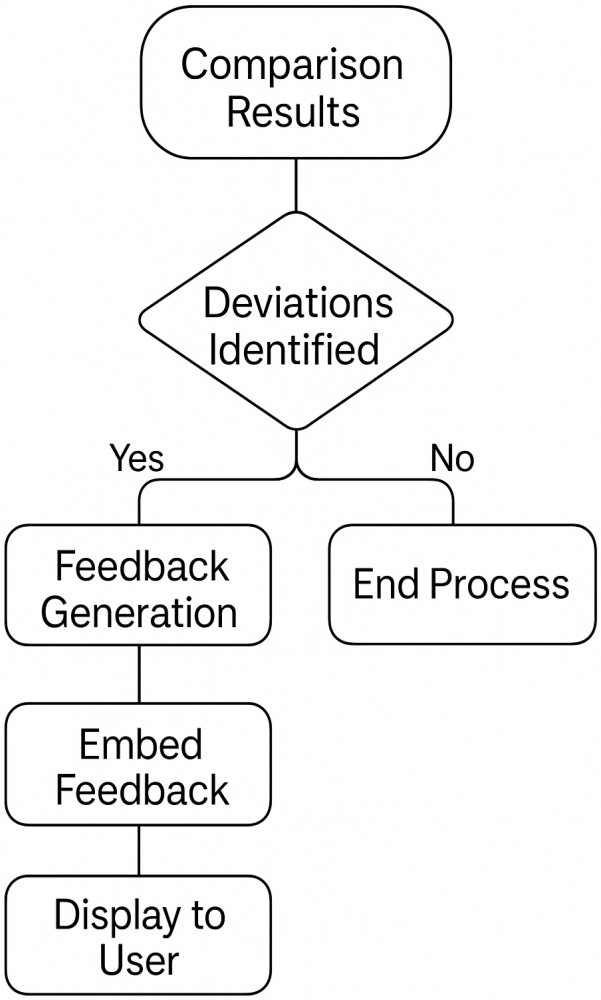
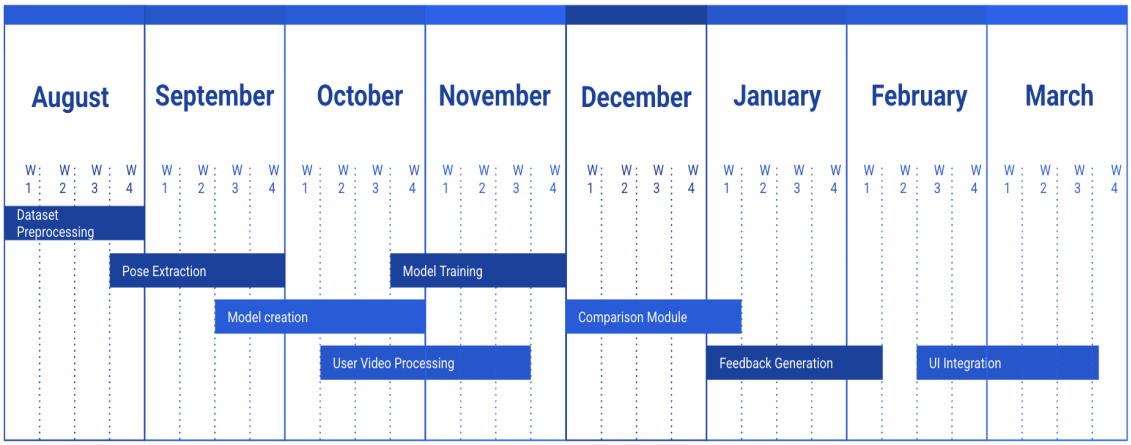


Figure 4.6: Feedback Generation

4.5 Timeline and Work Breakdown



Chapter 5

System Implementation

This chapter describes in detail the methodology used in this project and how it was implemented. The entire project was built using python. Flask was used for the backend server while HTML and CSS were used for creating the user-interface. Flask is a popular Python microframework for online application development that is lightweight, adaptable, and simple to use. HTML (HyperText Markup Language) is the standard markup language for constructing web pages, defining the structure and content of web documents and CSS is a style sheet language used to determine the appearance and styling of a document written in HTML.

5.1 Datasets Identified

CricShot10[4] was the dataset used. It is a video action recognition dataset consisting of 10 cricket batting shots, out of which, this project uses 5. This dataset was developed using the videos from YouTube. Video samples were collected considering all the three formats of cricket match: Test, ODI, and T20.

Figure 5.1: CricShot10 Dataset-hook shot(before preprocessing)

Figure 5.2: CricShot10 Dataset-hook shot(after preprocessing)

5.2 Proposed Methodology/Algorithms

5.2.1 Dataset processing

- Each video from the dataset is simultaneously cropped and trimmed to ensure that only the most important section is considered. This process makes sure that only the section where the batsman makes contact with the ball is included and unnecessary frames with extra people in the background is removed.

- Each video is cropped to a fixed standard of 10 frames with a resolution of 300×470 pixels, ensuring consistency across the dataset. The trimmed videos are then split into individual frames, and MediaPipe is used to extract body landmarks for each frame. From these landmarks, critical angles such as elbow, shoulder, knee, and hip are computed and stored in dataset_keyframes.json.

5.2.2 Processing of the User Video Input

- The video uploaded by the user is to be of the same format as that of the dataset, i.e, a standard of 10 frames with a resolution of 300×470 pixels. It should contain the frames where the bat makes contact with the ball and the follow-through.

- MediaPipe now extracts body landmarks such as angles for each frame. These extracted angles are stored in user_keyframes.json in the same format as the dataset, ensuring consistency in the upcoming comparison phase.

5.2.3 Feature Extraction

- To evaluate the user's shot against professional players, the system converts the extracted body angles into structured feature vectors. Each vector represents a single frame and consists of numerical values that correspond to key angles such as the elbow, knee, and hip.[7]

- The reference feature vectors of the dataset, stored in pose_tolerances.json, contain precomputed metrics such as the mean and standard deviation for each angle. By applying the same feature extraction method to the user's video, the system ensures

consistency in format, enabling an accurate comparison between the user's posture and the professional benchmark.

5.2.4 Model Training

The machine learning model is trained using the extracted features from the professional batsmen's shots. The feature vectors, along with their corresponding shot labels, are used to train a classifier. In this case, its RandomForestClassifier. The training process involves:

- Splitting the dataset into 80% training data and 20% testing data.
- Training the model on extracted feature vectors and their corresponding shot labels.
- Evaluating the model's performance using accuracy, precision, and recall metrics.
- Saving the trained model as pose_model.pkl for future predictions.

Once trained, the model can recognize whether a given shot conforms to professional standards.

5.2.5 Comparison and Evaluation

Once the user's feature vectors are extracted, they are compared against the reference dataset. The system combines the 10 frames into 3 sets containing 3,4 and 3 frames each respectively. Each of these sets is considered as a phase of the shot(initial, middle, end) and compared to a similar splitting of the dataset values. It then evaluates the user's shot by calculating the deviation of each body angle from the professional reference values. For each phase, the system classifies angle deviations into four categories:

- Perfect: Matches professional players' angles within a minimal margin.
- Great: Slight deviation but still within an acceptable range.
- Good: Moderate deviation that might affect shot efficiency.
- Needs Improvement: Significant deviation requiring correction.

Additionally, if a particular angle exceeds the pre-defined threshold, the system generates feedback indicating how and in what direction the user should move their body to correct their posture.

5.2.6 Feedback Generation

Using the trained pose_model.pkl, the system predicts whether the user's shot execution is technically sound. If the predicted class aligns with the ideal shot, the system confirms a correct form. Otherwise, the model suggests necessary improvements. Detailed feedback is generated for each key body angle, highlighting which angles deviate from the expected range. The system also provides targeted suggestions, such as:

- "Lower your elbow by 5° for better follow-through."
- "Increase your knee bend by 10° to maintain balance."
- "Your front leg is too straight; consider bending it slightly."

This feedback helps users understand specific corrections needed to improve their batting technique.

5.2.7 Output

To make the feedback more easy-to-understand, the system overlays visual corrections on one select frame from each phase from the user's video. Using OpenCV, the system dynamically draws lines and angles on the user's posture in real-time, showing both detected angles and the ideal angles for reference. The final output consists of:

- Annotated Video: The user's video is projected as 3 frames with feedback, displaying lines and angles indicating where all correction is required.
- Textual Report: A summary for each frame (each phase) is generated, detailing deviations and recommended corrections. By combining textual feedback with visual cues, the system ensures that users receive actionable insights to refine their shot technique.

5.3 User Interface Design

The user interface (UI) of the Smart Cricket Coach system is implemented using Flask for the backend and HTML, CSS, JavaScript for the front end The main components of the UI include:

- Video Upload and Selection: Users can either upload a pre-recorded video to the system.
- Pose Visualization: Once the video is processed, the UI displays the user's body posture with key points and lines overlaid on the frames. These landmarks are extracted using MediaPipe and show the joint positions and detected body angles.
- Comparison with Ideal Posture: The system presents a side-by-side view of the user's posture against an ideal reference shot from the trained dataset.
- Feedback Display: The UI provides necessary corrections in stance, foot positioning, and/or elbow position. Textual suggestions are shown alongside the important frames, guiding the user on how to align their posture more closely with the optimal form. The UI is designed to be simple yet effective, ensuring that players at any skill level can easily understand and implement the suggested corrections.

5.4 Implementation Strategies

The Smart Cricket Coach system leverages computer vision and machine learning to analyze a batsman's stance and shot execution. The implementation is divided into multiple stages, from dataset preprocessing to feedback generation. Various libraries and frameworks have been used for different tasks:

- OpenCV – For video processing, frame extraction, and visualization.
- MediaPipe – For detecting body landmarks and extracting joint positions.
- NumPy – For mathematical operations, including angle calculations.
- scikit-learn – For training and evaluating machine learning models.
- Tkinter – For building a simple GUI for video upload and result visualization.
- Joblib – For saving and loading trained models efficiently.

5.4.1 Dataset Processing

The dataset consists of videos of professional batsmen executing various cricket shots. These videos are segmented into individual frames and stored for further processing. This is done using the z shell, a powerful, interactive, and customizable Unix shell.

```
$ crop.zsh
#!/bin/zsh

# Input and Output Directories
input_dir="pull"
output_dir="output"

# Create output directory if it doesn't exist
mkdir -p "$output_dir"

# Crop Parameters (width:height:x:y)
crop_settings="300:470:480:100"

# Trim Parameters (in frames)
start_frame=15
end_frame=$((start_frame + 10))

# Function to round time values
round_time() {
    printf "%.6f" "$1"
}

# Process all videos in the input directory
for video_file in "$input_dir"/*; do
    filename=$(basename "$video_file")

    echo "Processing: $filename"

    # Get FPS using ffprobe
    fps=$(ffprobe -v error -select_streams v:0 -show_entries stream=r_frame_rate -o
```

```
    # Calculate and round trim times in seconds
    start_time=$(round_time "$(echo "$start_frame / $fps" | bc -l)")
    end_time=$(round_time "$(echo "$end_frame / $fps" | bc -l)")

    # Output path
    output_file="$output_dir/$filename"

    # Crop + Trim using FFmpeg
    ffmpeg -i "$video_file" -vf "crop=${crop_settings}" -ss "$start_time" -to "$end_

    echo "Done: $output_file"
done

echo "All videos cropped and trimmed"


# ffmpeg -i "shombincoverdrive.mp4" -vf "crop=300:470:480:100" -ss "" -to "$end_time
```

Figure 5.3: Code snippet for cropping and trimming

5.4.2 Processing of the Input Video and Feature Extraction

Before analysis, the user-uploaded video undergoes preprocessing, meaning the necessary frames are extracted and required body-angles are calculated.

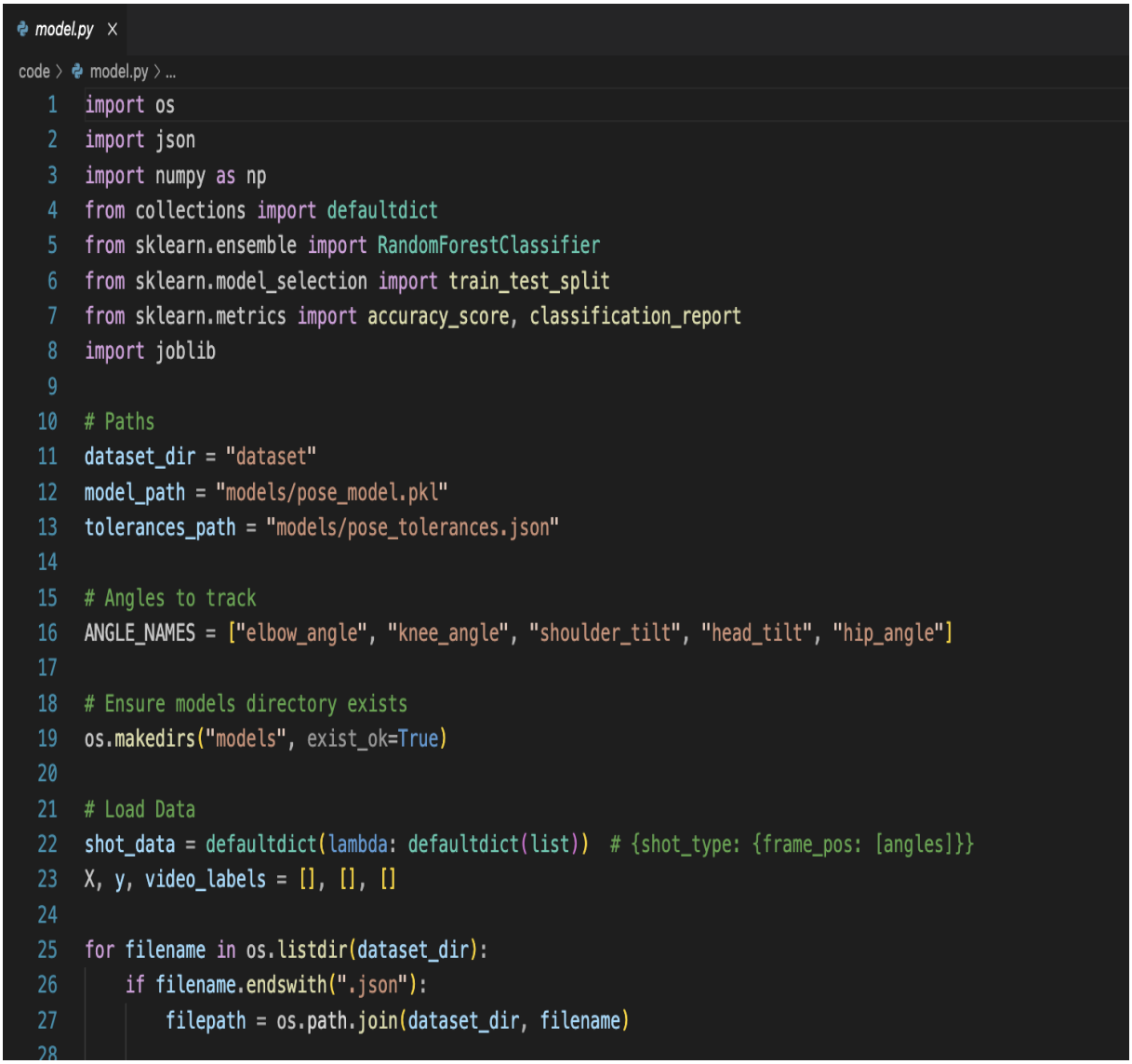
```python
def extract_keyframes(video_path, flag=1):
    cap = cv2.VideoCapture(video_path)
    keyframes = []
    frame_id = 0
    selected_frames = {2, 6, 9}

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        frame_id += 1
        image_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = pose.process(image_rgb)

        if results.pose_landmarks:
            keypoints = results.pose_landmarks.landmark
            angles = calculate_angles(keypoints)

            keyframes.append({
                "frame": frame_id,
                "angles": angles
            })

        if flag == 1 and frame_id in selected_frames:
            # Draw landmarks on the frame
            mp_drawing.draw_landmarks(
                frame, results.pose_landmarks, mp_pose.POSE_CONNECTIONS,
                mp_drawing.DrawingSpec(color=(0, 255, 0), thickness=2, circle_radius=3),
                mp_drawing.DrawingSpec(color=(255, 0, 0), thickness=2, circle_radius=2)
            )

            # Save the annotated frame
            frame_path = os.path.join(output_frames_dir, f"{frame_id}.jpg")
            cv2.imwrite(frame_path, frame)

    cap.release()
    return keyframes

def calculate_angles(keypoints):
    def get_angle(p1, p2, p3):
        v1 = np.array([p1.x - p2.x, p1.y - p2.y])
        v2 = np.array([p3.x - p2.x, p3.y - p2.y])
        angle = np.degrees(np.arccos(np.dot(v1, v2) / (np.linalg.norm(v1) * np.linalg.norm(v2))))
        return round(angle, 2)
```

Figure 5.4: Code snippet for processing the user's video

5.4.3 Model Training

A machine learning model (Random Forest) is trained on the extracted features to determine the ideal shot angles. Steps:

- Extract features from the dataset and create a labeled dataset.
- Train a supervised learning model on body angles.
- Save the trained model for real-time inference.



```
        try:
            with open(filepath, "r") as f:
                keyframes = json.load(f)

            if not keyframes:
                print(f"Skipping {filename} (Empty JSON)")
                continue

            # Extract shot type from filename (everything before "_")
            shot_type = filename.split("_")[0]

            # Process each frame
            frame_angles = []
            for frame in keyframes:
                frame_pos = frame["frame"]  # Get frame position

                if "angles" not in frame:
                    print(f"Skipping frame {frame_pos} in {filename} (Missing 'angles' key)")
                    continue

                try:
                    angles = [frame["angles"][angle] for angle in ANGLE_NAMES]
                    frame_angles.append(angles)

                    # Store frame data in shot_data per frame position
                    shot_data[shot_type][frame_pos].append(angles)

                except KeyError as e:
                    print(f"Skipping frame {frame_pos} in {filename} (Missing angle: {e})")
                    continue
```

```
            if not frame_angles:
                print(f"Skipping {filename} (No valid frames)")
                continue

            # Use mean values of angles across frames as a feature vector for the video
            mean_angles = np.mean(frame_angles, axis=0)

            X.append(mean_angles)
            y.append(shot_type)
            video_labels.append(filename)

        except json.JSONDecodeError:
            print(f"Skipping {filename} (Corrupt JSON)")

# Convert to numpy arrays
X = np.array(X)
y = np.array(y)

# Check if data is sufficient
if len(X) == 0:
    raise ValueError("No valid data found. Check dataset folder.")

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=40)

# Train RandomForest Model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```python
# Evaluate Model
y_pred = model.predict(X_test)

# Compute accuracy per shot type
shot_accuracies = defaultdict(lambda: {"correct": 0, "total": 0})

for actual, predicted in zip(y_test, y_pred):
    shot_accuracies[actual]["total"] += 1
    if actual == predicted:
        shot_accuracies[actual]["correct"] += 1

# Print overall accuracy
print(f"Overall Model Accuracy: {accuracy_score(y_test, y_pred) * 100:.2f}%\n")
print("Classification Report:\n")
print(classification_report(y_test, y_pred))

# Print per-shot accuracy
print("\nShot-wise Accuracy:")
for shot, counts in shot_accuracies.items():
    accuracy = (counts["correct"] / counts["total"]) * 100
    print(f"{shot}: {accuracy:.2f}% ({counts['correct']} / {counts['total']})")

# Save Model
joblib.dump(model, model_path)
print(f"Model saved to {model_path}")

# Save Pose Tolerances (Per Shot Type & Frame Position)
pose_tolerances = {}
```

```python
for shot_type, frames_data in shot_data.items():
    pose_tolerances[shot_type] = {}

    for frame_pos, frame_values in frames_data.items():
        shot_array = np.array(frame_values)  # Convert list to NumPy array

        if len(shot_array) > 1:
            weights = np.linspace(0.5, 2, num=len(shot_array))  # Weight later frames more

            pose_tolerances[shot_type][frame_pos] = {
                name: {
                    "mean": float(np.average(shot_array[:, i], weights=weights)),
                    "std": float(np.sqrt(np.average((shot_array[:, i] - np.average(shot_array[:, i], wei
                }
                for i, name in enumerate(ANGLE_NAMES)
            }

# Save updated tolerances
with open(tolerances_path, "w") as f:
    json.dump(pose_tolerances, f, indent=4)

print(f"Pose tolerances saved to {tolerances_path}")
```

Figure 5.5: Code snippet for model training

5.4.4 Comparison and Evaluation

The extracted features from the user's shot are compared against the ideal angles from the trained dataset. Steps:

- Retrieve body angles from the user's shot.
- Compute deviations from the ideal angles.
- Evaluate the correctness of posture and movement.

comparison.py

code > comparison.py > ...

```
78
79  # Compare user keyframes against reference tolerances
80  def compare_keyframes(ml_model, tolerances, user_keyframes, shot_type):
81      if shot_type not in tolerances:
82          sys.exit(1)
83
84      shot_tolerances = tolerances[shot_type]
85      frame_feedback = []
86
87      for frame_idx, user_frame in enumerate(user_keyframes):
88          if str(frame_idx + 1) not in shot_tolerances:
89              continue  # Skip if reference data for this frame is missing
90
91          frame_result = {"frame": frame_idx + 1, "analysis": {}, "issues": []}
92          frame_tolerances = shot_tolerances[str(frame_idx + 1)]
93          features = []
94
95          for param, user_value in user_frame["angles"].items():
96              if param in frame_tolerances:
97                  ref_range = frame_tolerances[param]
98                  deviation = abs(user_value - ref_range["mean"])
99                  quality, advice = get_quality_and_feedback(param, deviation, user_value, ref_range)
100
101                 frame_result["analysis"][param] = quality
102                 if advice:
103                     frame_result["issues"].append(advice)
104
105                 features.append(user_value)
106
107         # ML Model Prediction (1 = Correct, 0 = Incorrect)
108         prediction = ml_model.predict([features])[0]
109         frame_result["prediction"] = "Correct Form" if prediction == 1 else "Incorrect Form"
110
111         frame_feedback.append(frame_result)
112
113     return frame_feedback
```

Figure 5.6: Code snippet for model training

5.4.5 Feedback Generation

The system provides feedback based on detected deviations, suggesting improvements in batting posture. Steps:

- Highlight incorrect angles and suggest corrections.
- Generate visual overlays on frames from the video to guide the user.
- Provide textual and visual feedback.

comparison.py

code > comparison.py > ...

```python
# Load JSON helper
def load_json(file_path):
    with open(file_path, 'r') as f:
        return json.load(f)

# Save JSON helper
def save_json(data, file_path):
    with open(file_path, 'w') as f:
        json.dump(data, f, indent=4)

# Load trained model & tolerances
def load_models():
    ml_model = joblib.load(POSE_MODEL_PATH)
    tolerances = load_json(TOLERANCE_PATH)
    return ml_model, tolerances

# Get feedback based on deviation
def get_quality_and_feedback(param, deviation, user_value, ref_range):
    direction = "upwards" if user_value < ref_range["mean"] else "downwards"

    if deviation <= ref_range["std"] * 0.5:
        return "Perfect", None
    elif deviation <= ref_range["std"]:
        return "Great", None
    elif deviation <= ref_range["std"] * 1.5:
        return "Good", f"Adjust your {param.replace('_', ' ')} slightly {direction}."
    else:
        feedback = {
            "elbow_angle": f"Move your elbow {direction} for better alignment.",
            "knee_angle": f"Bend your knee {direction} to match the reference posture.",
            "shoulder_tilt": f"Adjust your shoulder tilt {direction} for better posture.",
            "head_tilt": f"Adjust your head tilt {direction} to improve alignment.",
            "hip_angle": f"Modify your hip angle {direction} for better posture."
        }.get(param, f"Adjust your {param.replace('_', ' ')} {direction} for better posture.")
        return "Needs Improvement", feedback
```

Figure 5.7: Code snippet for model training

5.4.6 Output

The final output consists of an overlay on the main frames from the video with detected angles and recommendations. The system visualizes these corrections. Steps:

- Overlay detected key points and angles on the user's video.
- Display suggested improvements on the screen.
- Allow users to analyze their shots.

```
# Generate and print phase-wise feedback in a readable format (only if issue appears in 2+ frames)
def print_phase_feedback(frame_feedback):
    phase_ranges = {
        "Starting Phase": range(0, 3),
        "Middle Phase": range(3, 7),
        "Ending Phase": range(7, 10)
    }

    print("\nGeneralized Feedback:")

    for phase, frame_range in phase_ranges.items():
        phase_issues = []

        # Collect issues from frames in the current phase
        for frame in frame_feedback:
            if frame["frame"] - 1 in frame_range:  # Convert to zero-indexed
                phase_issues.extend(frame["issues"])

        # Count occurrences of each issue
        issue_counts = Counter(phase_issues)

        # Display only issues that occur in **2 or more frames**
        filtered_issues = [issue for issue, count in issue_counts.items() if count >= 2]

        if filtered_issues:
            print(f"\n{phase}:")
            for issue in filtered_issues:
                print(f"  - {issue}")
```

Figure 5.8: Code snippet for model training

Chapter 6

Results and Discussions

In this chapter the results and discussions of the Cricket Sensei prototype's implementation will be examined. An outline of the results attained and in-depth explanations of the findings are given in the sections that follow.

6.1 Overview

The Cricket Batting Shot Analysis System is an AI-driven framework designed to analyze and enhance a batsman's technique using computer vision and machine learning. The system enables users to upload a video of their batting shot, which is processed through MediaPipe Pose to extract key skeletal landmarks. Critical joint angles—such as the elbow, knee, shoulder, and hip—are computed using vector mathematics, forming the basis for performance evaluation.

A Random Forest Classifier is trained on an annotated dataset of professional cricket shots, allowing the system to compare a user's pose with an optimal reference model. The extracted features are classified based on predefined pose tolerances, assessing whether the user's posture aligns with the ideal form. Deviations are identified, and the system generates automated, text-based feedback detailing necessary adjustments to improve technique.

Additionally, left-handed batsmen are detected and filtered to maintain consistency in analysis. By integrating pose estimation, statistical modeling, and machine learning, this system provides an efficient, data-driven approach to cricket training. It minimizes reliance on manual coaching, offering scalable, automated shot evaluation and personalized feedback to help cricketers refine their skills.

6.2 Model Performance

The proposed AI-based cricket training system was evaluated using a Random Forest classifier trained on pose angles extracted from batting videos. The model was tested on a dataset containing four distinct batting shots: cover drive, flick, hook, and pull shots. The system's effectiveness was assessed using classification accuracy, precision, recall, and F1-score.

6.2.1 Evaluation based on Performance Metric

The overall accuracy of the model was 85.61%, indicating a high capability in distinguishing between different shot types based on pose features. The detailed classification report is shown in Figure 6.1.

```
> python3 model.py
Overall Model Accuracy: 85.61%

Classification Report:

              precision    recall  f1-score   support

       cover       0.74      0.88      0.80        32
       flick       0.95      0.85      0.90        41
        hook       0.91      0.86      0.89        36
        pull       0.83      0.83      0.83        30

    accuracy                           0.86       139
   macro avg       0.86      0.86      0.85       139
weighted avg       0.86      0.86      0.86       139
```

Figure 6.1: Classification Performance of Random forest Model

From the table, flick and hook shots exhibit the highest precision (0.95 and 0.91, respectively), meaning they are less prone to false positives. However, the cover drive has a lower precision of 0.74, suggesting some misclassification.

6.2.2 Shot-wise Accuracy

In addition to standard evaluation metrics, shot-wise accuracy was computed to assess the system's consistency in identifying each type of shot. The results are summarized in

Figure 6.2.

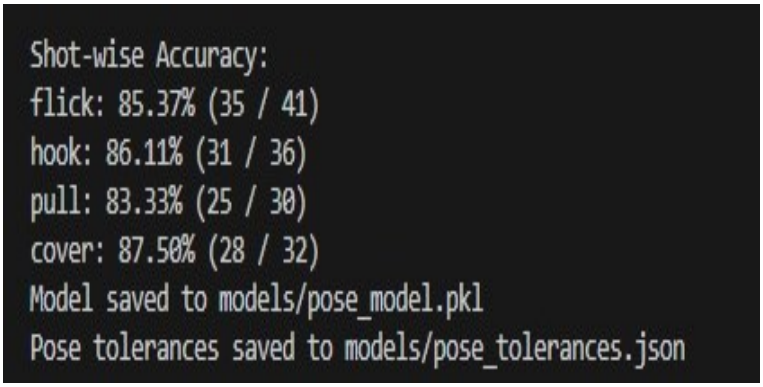


Figure 6.2: Shot-Wise Accuracy

The highest shot-wise accuracy was observed for cover drives (87.50%), while the lowest was for pull shots (83.33%).

6.2.3 Model Deployment and Pose Tolerance Estimation

To enhance real-time feedback, the trained model was saved as pose_modelṗkl, allowing for easy integration into a cricket coaching system. Furthermore, pose tolerances were calculated for each shot type, capturing the mean and standard deviation of key pose angles. These tolerances, stored in pose_tolerances.json, will enable the system to provide more personalized feedback by identifying deviations from ideal shot execution.

6.2.4 Key Findings

- The Random Forest classifier performed well, achieving over 85% accuracy.
- The cover drive had the highest accuracy (87.50%), but lower precision compared to other shots.
- The model's precision and recall scores indicate strong differentiation between shots, with minor misclassifications in closely related shots.
- The saved pose tolerances will be useful for feedback generation, enabling personalized corrections for cricketers.

6.3 Feedback Generation

6.3.1 Overview of Shot Analysis Module

The shot analysis module of the cricket training system provides an intuitive and effective way for users to receive feedback on their batting techniques. Users upload a video containing their shot, which is then analyzed in comparison to an ideal reference player's shot. The process is structured into three major phases:

- Starting Phase (Frames 1-3)
- Middle Phase (Frames 4-7)
- Ending Phase (Frames 8-10)

Each phase is assessed separately, with key body landmarks extracted and evaluated for deviations from the reference player's optimal positions. The system provides feedback to help users adjust their posture, knee angle, shoulder tilt, and other critical aspects of their stance and movement.

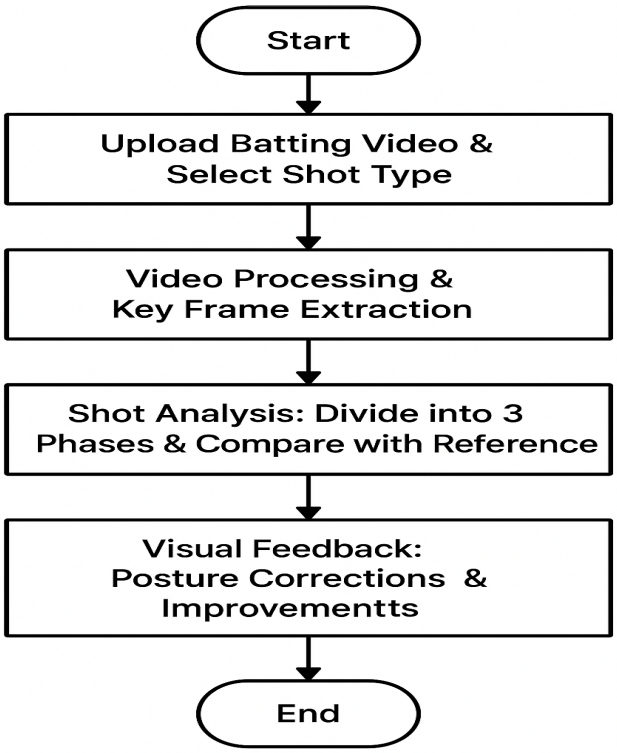
6.3.2 Implementation of Analysis

To achieve this functionality, three core files—analyzer.py, comparison.py, and model.py—were developed and integrated.

- analyzer.py is responsible for extracting body landmarks from the user's uploaded video. This is achieved using MediaPipe Pose Estimation, which identifies key points such as shoulders, elbows, knees, and hips.
- comparison.py performs a detailed comparison between the user's shot and the ideal reference shot. It calculates angular deviations and positional mismatches to generate actionable feedback.
- model.py acts as the backbone of the system, handling reference data derived from over 100+ professional cricket shots and ensuring accurate comparisons.

6.3.3 User Interface and Experience

The system provides a user-friendly interface, as illustrated in the images below. The process flow is structured as follows:



Below is the user interface at different stages:

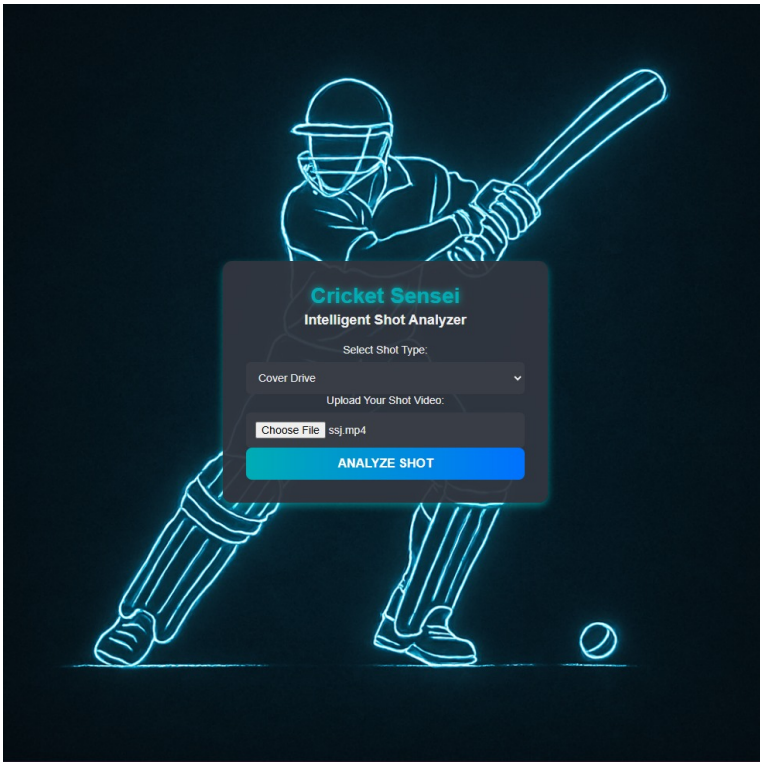


Figure 6.3: User Uploading Video and Selecting Shot Type

Figure 6.4: Processing Phase

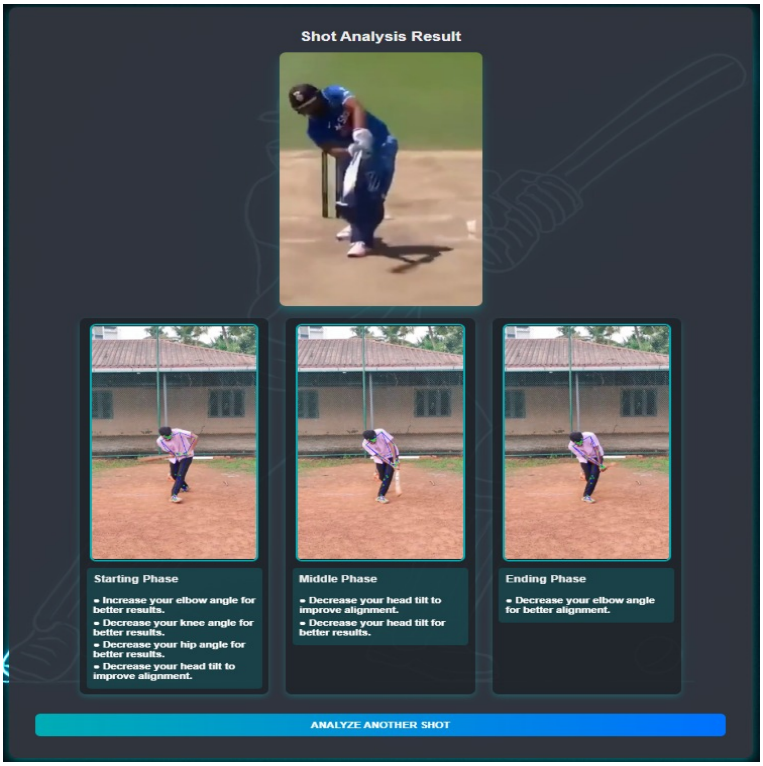


Figure 6.5: Final Analysis and Feedback Display

6.3.4 Discussion of Results

The system successfully analyzes the shot and provides meaningful feedback based on biomechanical analysis. Some key observations include:

- Knee Angle Adjustments: Many users have been observed to have excessive or insufficient knee bending, affecting their stability during shots.
- Shoulder Tilt: Proper shoulder alignment is critical for shot balance, and the system effectively highlights deviations in the user's posture.
- Head Positioning: The head should remain stable and aligned with the shot; misalignment often leads to mistimed shots, which is detected and corrected.
- Elbow and Hip Angles: These parameters also play a vital role in evaluating shot quality. Deviations in elbow angle may indicate improper arm extension, while hip angle affects weight distribution and rotation efficiency.

Five Key Parameters: The core of the biomechanical analysis focuses on five parameters namely elbow_angle, knee_angle, shoulder_tilt, hip_angle, and head_tilt, each offering

critical insights into the overall posture and technique of the batsman during each shot phase.The modular design of the system ensures that it can be expanded to include additional shot types and customized feedback mechanisms in the future.

Chapter 7

Conclusions and Future Scope

CricketSensei has the potential to enhance the player's performance by providing more personalized and data-driven feedback.Also, the system aims to detect the shot that is to be trained. The system offers real-time insights into shot mechanics of the player. The player will be able to make quicker adjustments with live feedback systems. With increasing technological opportunities CricketSensei can be used to make more tailored, specific and effective coaching for the players. Integrating evolved techniques and more AI models, this system can identify areas of improvement and to refine their techniques resulting in even greater breakthroughs in sports performance analysis.

The future scope of the project aims to enhance the accuracy of cricket shots performed by it's users and to broaden it's reach among the aspiring cricket enthusiasts. Some of them include the addition of player height to further personalize the feedback, making it accessible to batsmen who are both right and left handed, detect any missed shots to determine shot accuracy as well and also enable live feedback during training sessions to provide immediate responses. Furthermore, applying this system to other sports such as football, tennis and badminton could prove to be helpful for players of all athletic disciplines.

References

[1] R. Savran Kızıltepe, J. Q. Gan, and J. J. Escobar, "A novel keyframe extraction method for video classification using deep neural networks," *Neural Computing and Applications*, vol. 35, no. 34, pp. 24 513–24 524, 2023.

[2] H. Nandini, H. Chethan, and B. Rashmi, "Shot based keyframe extraction using edge-lbp approach," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 7, pp. 4537–4545, 2022.

[3] H. U. R. Siddiqui, F. Younas, F. Rustam, E. S. Flores, J. B. Ballester, I. d. l. T. Diez, S. Dudley, and I. Ashraf, "Enhancing cricket performance analysis with human pose estimation and machine learning," *Sensors*, vol. 23, no. 15, p. 6839, 2023.

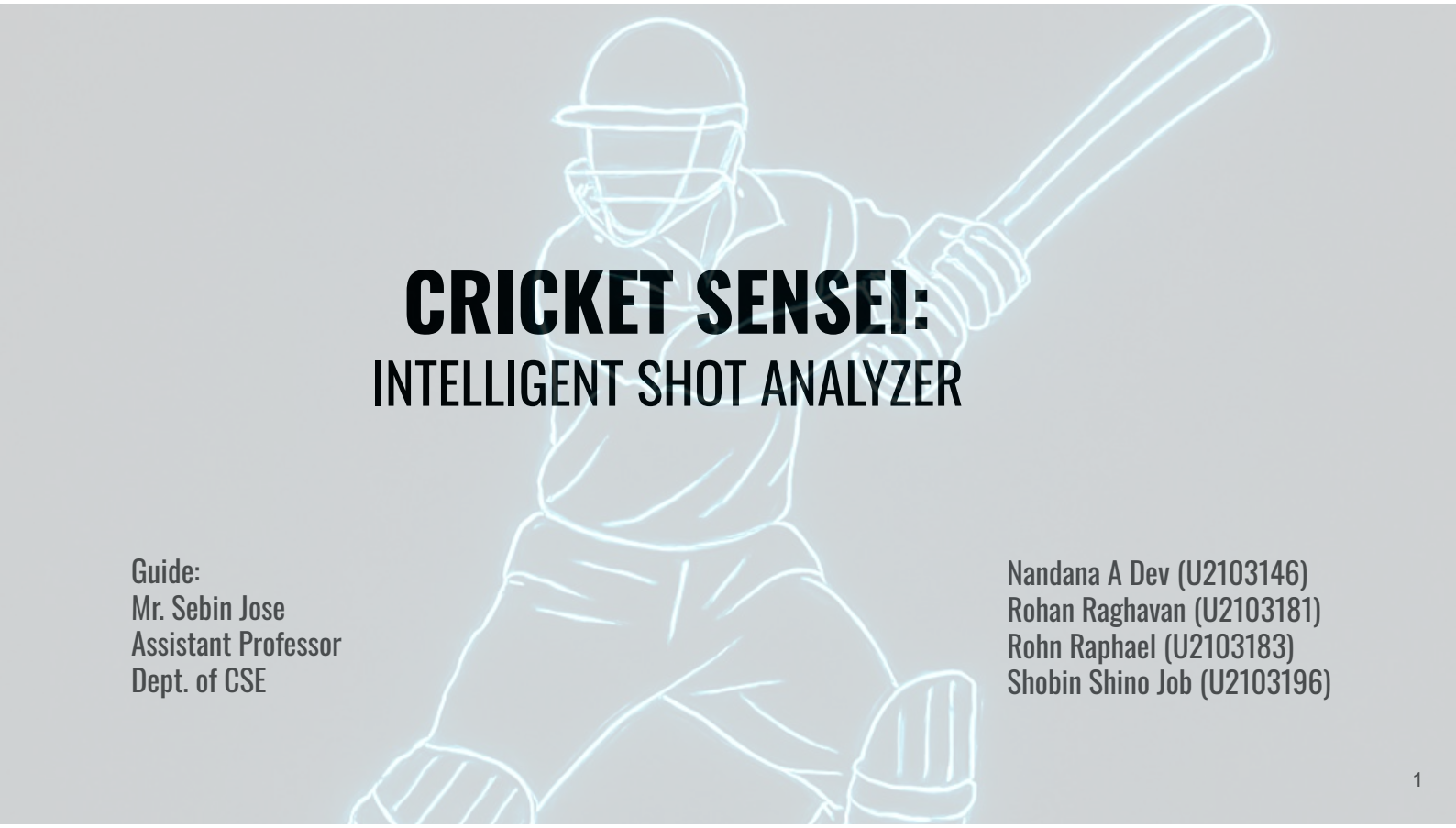
[4] A. Sen, K. Deb, P. K. Dhar, and T. Koshiba, "Cricshotclassif: An approach to classifying batting shots from cricket videos using a convolutional neural network and gated recurrent unit," *Sensors*, vol. 21, no. 8, p. 2846, 2021.

[5] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2021.

[6] R. Singh and R. S. Anand, "Key frame extraction from video using entropy and local features," *Multimedia Tools and Applications*, vol. 80, pp. 3191–3212, 2021.

[7] N. Srivastava and P. Singhal, "Human pose estimation and action recognition using deep learning: A survey," *Multimedia Tools and Applications*, vol. 81, pp. 879–909, 2022.

Appendix A: Presentation



CRICKET SENSEI:
INTELLIGENT SHOT ANALYZER

Guide:
Mr. Sebin Jose
Assistant Professor
Dept. of CSE

Nandana A Dev (U2103146)
Rohan Raghavan (U2103181)
Rohn Raphael (U2103183)
Shobin Shino Job (U2103196)

CONTENTS

INTRODUCTION

- Cricket Sensei is a smart coaching tool designed for enhancing batting performance.
- Helps players improve their batting techniques through real-time, personalized feedback.
- Users upload a video of their shot to receive instant analysis and suggestions.

PROBLEM DEFINITION

Amateur cricketers lack access to real-time, personalized feedback on their batting techniques, making it difficult to identify and correct flaws in posture, stance, and swing. This results in slower skill development and limits their ability to reach professional standards independently.

NOVELTY AND INNOVATIVENESS

- Pose Estimation in Sports Feedback :
  - The use of pose estimation (via MediaPipe) to analyze cricket shots brings a unique, data-driven approach to sports training.
- Self-Coaching Platform with UI Integration :
  - Seamless user interface with backend processing, allowing users to upload videos and receive feedback on their shot.

LITERATURE SURVEY

Paper Title	Methods Used	Results	Advantages	Disadvantages
Sen, Anik, Kaushik Deb, Pranab Kumar Dhar, and Takeshi Koshiba. "CricShotClassify: An Approach to Classifying Batting Shots from Cricket Videos Using a Convolutional Neural Network and Gated Recurrent Unit" Sensors 21 (2021), no. 8: 2846	• Extraction of spatial-temporal features from joints • GRU for temporal modeling • CNN for spatial feature extraction • Transfer learning based VGG16 model	• Pretrained VGG16 model achieved highest F1 score of 86%.	• Automatic feature extraction using CNN • Capturing temporal dependencies using GRU • Improved accuracy using transfer learning	• Dependency on accurate frame selection. • Requires careful tuning. • Limited generalization across players.
Savran Kızıltepe, R., Gan, J. Q., & Escobar, J. J. "A novel keyframe extraction method for video classification using deep neural networks". Neural Computing and Applications, 35(34), (2023)	• Action Template Recognition • Action Template Identification • Action Location Specigication • Keyframe Extraction • Keyframe Selection • VGG-16-ConvLSTM	• ConvLSTM(2) and LSTM(2), which used the proposed method, achieved accuracy scores of 88.15% and 83.10%, respectively, surpassing the previous method of selecting one frame per second.	• High accuracy • Effective feature extraction • Robust classification • Automated diagnosis • Scalable architecture	• High computational cost • Requires large labeled dataset • Longer training time • Risk of overfitting • Limited generalization to unseen diseases

Paper Title	Methods Used	Results	Advantages	Disadvantages
Nandini, H. M., Chethan, H. K., & Rashmi, B. S. "Shot based keyframe extraction using edge-LBP approach". Journal of King Saud University-Computer and Information Sciences, 34(7) (2022)	• The paper proposes an edge-based Local Binary Pattern (LBP) approach for efficient keyframe extraction. • It uses shot boundary detection to segment videos into shots. • Within each shot, Edge LBP features are computed to identify representative frames.	• The approach significantly reduces the number of frames required for analysis. • Extracted keyframes retain high visual and informational relevance. • It achieved better precision and recall compared to traditional histogram and entropy-based keyframe selection methods	• Computationally efficient, suitable for real-time applications. • Reduces redundant frames, enabling faster pose estimation.	• Might struggle in motion-heavy shots where edge orientation changes rapidly. • Doesn't directly incorporate semantic understanding (like recognizing a cricket shot).
Siddiqui, H.U.R., Younas, F., Rustam, F., Flores, E.S., Ballester, J.B., Diez, I.D.L.T., Dudley, S. and Ashraf, I. "Enhancing cricket performance analysis with human pose estimation and machine learning". Sensors, 23(15), p.6839. (2023)	• Applied machine learning algorithms including RF, SVM, KNN, DT, LR, and LSTM networks to classify eight cricket strokes: pull, cut, cover drive, straight drive, backfoot punch, on drive, flick, and sweep.	• The RF model achieved an outstanding accuracy of 99.77%, outperforming other algorithms.	• The RF Model approach provides precise classification of multiple cricket strokes.	• The accuracy of pose estimation and subsequent analysis may be affected by the quality of the input video footage. • Limited to visible landmarks

OBJECTIVES

- To develop an ML-based system that analyzes cricket batting techniques using video input from users.
- Train the model on a video dataset of professional cricket shots to recognize and device ideal shot mechanics.
- To allow users to choose from four different cricket shots — Cover Drive, Pull Shot, Flick Shot, and Hook Shot — and receive pose-specific feedback tailored to each style.

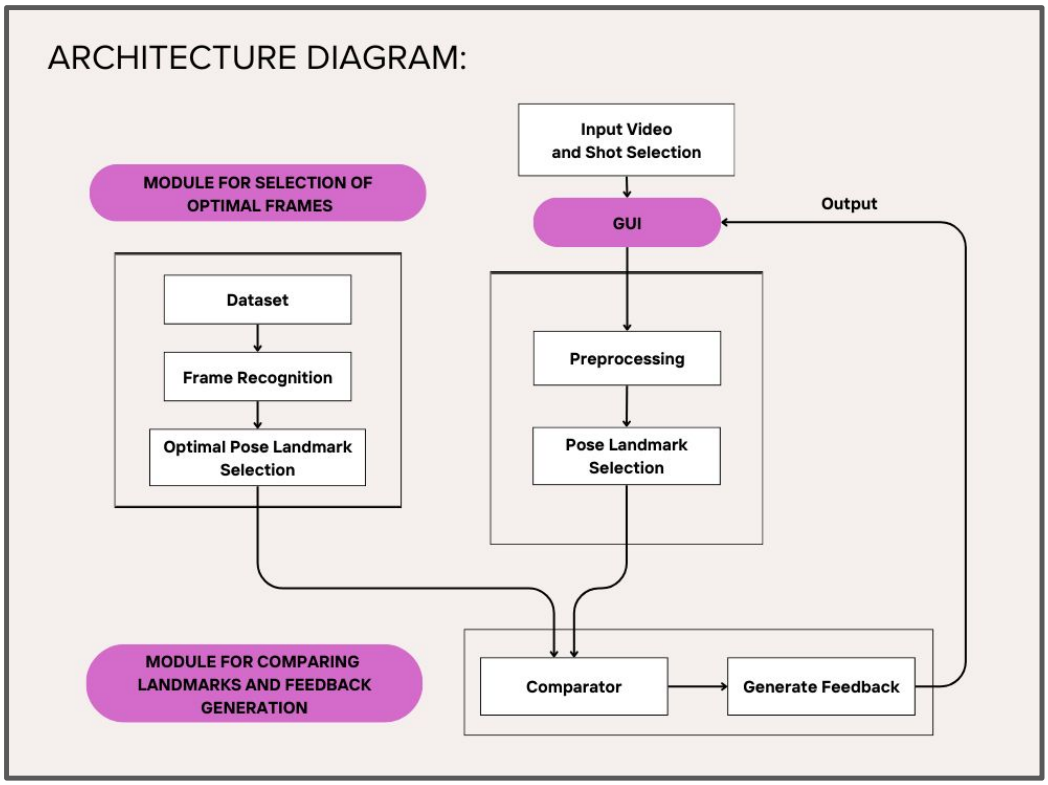
APPLICATIONS & BENEFICIARIES

- Performance Evaluation in Academies :
  - Monitor player improvement over time.
  - Enables objective comparison of a player's technique.
- Remote Sports Training for Aspiring Cricketers:
  - Cricketers in underserved areas can upload videos and receive expert-like analysis.
  - Eliminates the need for a human coach.

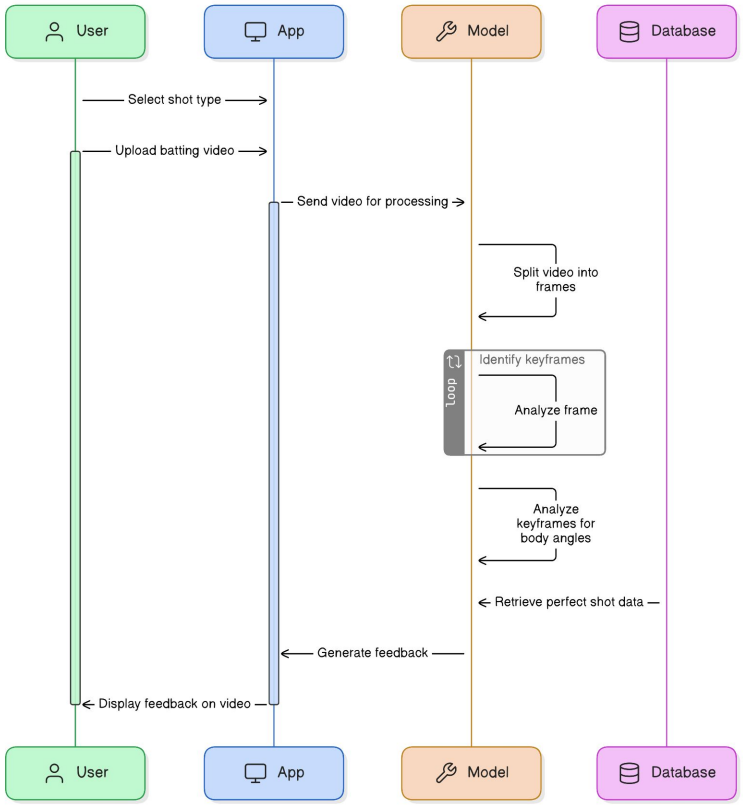
APPLICATIONS & BENEFICIARIES

- Cricket Coaches and Trainers :
  - Provides an assistive tool for evaluating players more efficiently.
  - Saves time by highlighting key issues automatically.

SYSTEM ARCHITECTURE



SEQUENCE DIAGRAM



METHODOLOGY

1. Dataset Preprocessing
2. Ideal Pose Extraction
3. Comparator
4. Feedback Generation

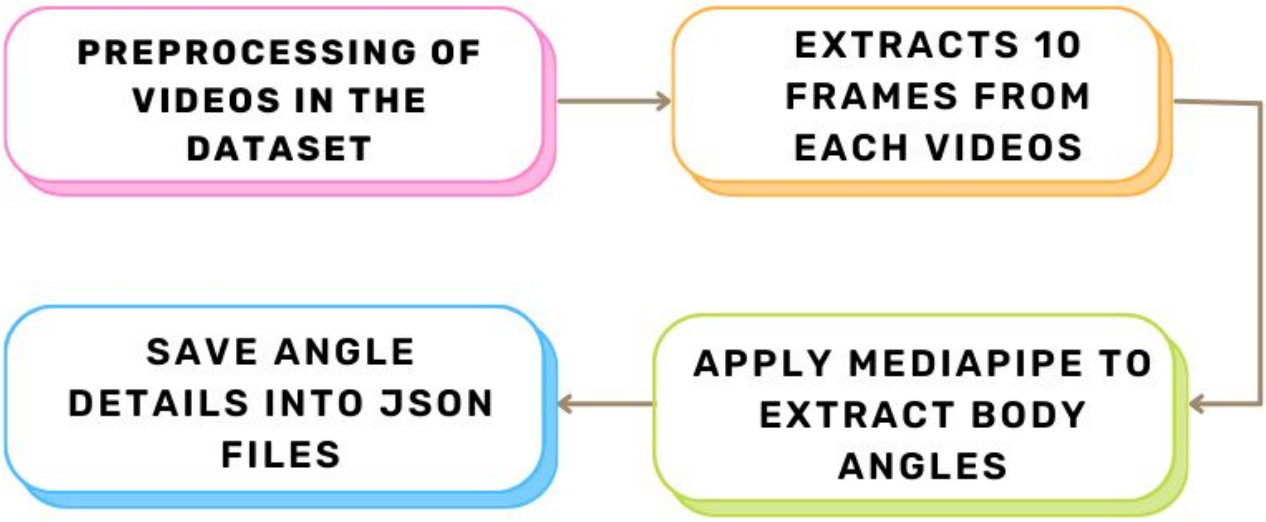
DATASET PREPROCESSING

- The dataset was received directly from the authors of CricShotClassify[1] after formal request via email.
- Contains around 180+ videos per shots with a total of 10 shots.
- Preprocessed and reduced to 100+ videos of 4 shots - Cover Drive, Hook, Pull and Flick.

DATASET PREPROCESSING

- Carefully curated the dataset by selecting only videos demonstrating technically correct cricket shot executions.
- Excluded videos featuring left-handed batsmen and spin bowlers .
- Developed a script to automate the trimming and cropping of videos, isolating the batsman from the full frame.

POSE FEATURE EXTRACTION



POSE FEATURE EXTRACTION

- The video is trimmed down to only the necessary 10 frames from which the ideal pose for the shot is identified using the body angles extracted from each frame.
- The model learns the accepted range of angles for each frame of the shot which will be used to compare with user data.
- These extracted parameters are sent as input to the next module.

ANALYZER

- From each video of the preprocessed video dataset, the different body angles is extracted using MediaPipe and stored into another textual dataset as .json files.

- This dataset is used to train the model to find the mean features and the deviation from the mean positions which gives us the range of the ideal pose.

MODEL

- Algorithm : RandomForestClassifier
- Random Forest is a powerful ensemble learning algorithm used for both classification and regression tasks. It works by constructing a large number of decision trees during training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

MODEL

- Hyperparameters:
  - RandomForestClassifier:
    - No. of trees=100
    - Random State=42
  - Pose tolerance weights:
    - np.linspace(0.5,2,...) - Giving later frames more weightage.

MODEL

- Data Splitting:
  - 80% Training Set
  - 20% Testing Set (evaluated after training)
- Input : .json files containing body angles of each professional player in the dataset
- Output : .json file storing mean & standard deviation of ideal keyframes

MODEL EVALUATION

- **Precision**: Ratio of correctly predicted positive observations to the total predicted positive observations, measuring the accuracy of positive predictions.
- **Recall**: Ratio of correctly predicted positive observations to all actual positive observations, measuring the model's ability to identify all relevant instances in a dataset.
- **F1 score**: Harmonic mean of precision and recall, providing a balanced measure of a model's accuracy when there is an uneven class distribution.
- **Support**: Number of data samples in the dataset for each class.

COMPARATOR & FEEDBACK GENERATOR

- Deviation is calculated using the difference in values of features extracted from user's pose and the reference pose's mean.
- Different feedback is generated according to the magnitude of deviation

User Interface

- **Intuitive Video Upload:**
  - Clean and simple interface allowing users to upload their cricket shot videos (MP4, AVI) with minimal steps.
- **Real-Time Visual Feedback:**
  - Displays frames from the user's video with overlaid pose landmarks, angle lines, and annotations on key joints for easy understanding.
- **Dynamic Feedback Panel:**
  - Automatically generates and displays shot evaluation results—highlighting areas of improvement like elbow position or follow-through posture.
- **Responsive Layout:**
  - Works seamlessly across desktops and tablets, ensuring usability across devices for players and coaches.

WORK DIVISION

Nandana A Dev	Rohan Raghavan	Rohn Raphael	Shobin Shino Job
• UI Integration • Comparison module	• User video processing • Feedback generation	• Dataset Preprocessing • Model Creation	• Pose extraction • Model Training

HARDWARE REQUIREMENTS

- Storage: For storing videos, processed frames, and datasets.
- Processor : Intel core i5 or AMD Ryzen 5
- RAM : 8GB
- GPU: NVIDIA GTX 1050 Ti or NVIDIA GTX 1650
- Disk Space : 15GB

SOFTWARE REQUIREMENTS

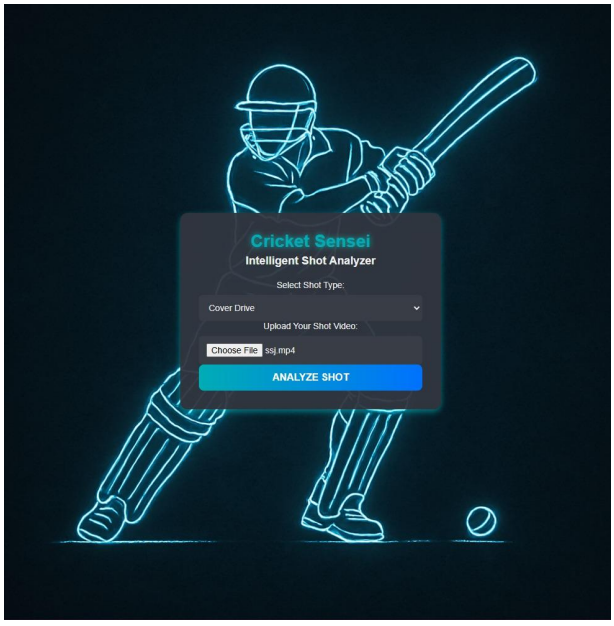
- Operating System: Windows/Linux/macOS
- Development Tools: Python, TensorFlow, OpenCV
- Pose Estimation: Mediapipe/OpenPose
- Web/Frontend: HTML, CSS, JavaScript, Flask/Django
- Database: MySQL for storing user videos and results
- Video Processing: FFmpeg

RESULTS

- Flick and hook shots exhibit the highest precision (0.95 and 0.91 respectively), meaning they are less prone to false positives.
- Shotwise accuracy of all shots are above 80% making it

```
❯ python3 model.py
Overall Model Accuracy: 85.61%

Classification Report:

              precision    recall  f1-score   support

       cover       0.74      0.88      0.80        32
       flick       0.95      0.85      0.90        41
        hook       0.91      0.86      0.89        36
        pull       0.83      0.83      0.83        30

    accuracy                           0.86       139
   macro avg       0.86      0.86      0.85       139
weighted avg       0.86      0.86      0.86       139


Shot-wise Accuracy:
flick: 85.37% (35 / 41)
hook: 86.11% (31 / 36)
pull: 83.33% (25 / 30)
cover: 87.50% (28 / 32)
Model saved to models/pose_model.pkl
Pose tolerances saved to models/pose_tolerances.json
```


RESULTS



RESULTS

RESULTS

FUTURE SCOPE

- Automating shot type detection and classification:
    - Implement machine learning classifiers to automatically recognize the type of cricket shot.
- Real-time feedback and live analysis integration:
    - Develop a live analysis module to provide instant feedback during practice sessions, making the system more interactive for real-time coaching.
- Expanding training datasets with professional footage:
    - Collaborate with cricket academies, coaches, and open-source sports data communities to build a more diverse and representative dataset of professional cricket shots for better accuracy and adaptability.

CONCLUSION

- The development of Cricket Sensei: Intelligent Shot Analyzer represents a meaningful step toward integrating AI techniques into sports training.
- Utilizing pose estimation and angular analysis, the system provides structured and personalized feedback to help players refine their cricket shots.
- During the course of the project, we addressed key technical challenges such as variations in landmark detection and efficient processing of video data. The resulting tool supports both individual skill development and offers potential as a data-driven aid for coaches and trainers in analyzing player performance.

REFERENCES

1. A. Sen, K. Deb, P. K. Dhar, and T. Koshiba, “CricShotClassify: An Approach to Classifying Batting Shots from Cricket Videos Using a Convolutional Neural Network and Gated Recurrent Unit,” Sensors, vol. 21, no. 8, p. 2846, Apr. 2021, doi: https://doi.org/10.3390/s21082846.
2. Siddiqui, Hafeez Ur Rehman, Faizan Younas, Furqan Rustam, Emmanuel Soriano Flores, Julién Brito Ballester, Isabel de la Torre Diez, Sandra Dudley, and Imran Ashraf. "Enhancing cricket performance analysis with human pose estimation and machine learning." Sensors 23, no. 15 (2023): 6839.
3. H. M. Nandini, H. K. Chethan, and B. S. Rashmi, “Shot based keyframe extraction using edge-LBP approach,” Journal of King Saud University - Computer and Information Sciences, Nov. 2020, doi: https://doi.org/10.1016/j.jksuci.2020.10.031.
4. R. Savran Kızıltepe, J. Q. Gan, and J. J. Escobar, “A novel keyframe extraction method for video classification using deep neural networks,” Neural Computing and Applications, Aug. 2021, doi: https://doi.org/10.1007/s00521-021-06322-x.

REFERENCES

5. Devanandan, M., Rasaratnam, V., Anbalagan, M. K., Asokan, N., Panchendrarajan, R., & Tharmaseelan, J. (2021, December). Cricket shot image classification using random forest. In 2021 3rd International Conference on Advancements in Computing (ICAC) (pp. 425-430). IEEE.
6. https://medium.com/@codetrade/implementation-of-human-pose-estimation-using-mediapipe-23a57968356b
7. https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker#:~:text=The%20MediaPipe%20Pose%20Landmarker%20task,with%20single%20images%20or%20video.
8. https://en.wikipedia.org/wiki/Random_forest


THANK YOU

Appendix B: Vision, Mission, Programme Outcomes and Course Outcomes

Vision, Mission, Programme Outcomes and Course Outcomes

Institute Vision

To evolve into a premier technological institution, moulding eminent professionals with creative minds, innovative ideas and sound practical skill, and to shape a future where technology works for the enrichment of mankind.

Institute Mission

To impart state-of-the-art knowledge to individuals in various technological disciplines and to inculcate in them a high degree of social consciousness and human values, thereby enabling them to face the challenges of life with courage and conviction.

Department Vision

To become a centre of excellence in Computer Science and Engineering, moulding professionals catering to the research and professional needs of national and international organizations.

Department Mission

To inspire and nurture students, with up-to-date knowledge in Computer Science and Engineering, ethics, team spirit, leadership abilities, innovation and creativity to come out with solutions meeting societal needs.

Programme Outcomes (PO)

Engineering Graduates will be able to:

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and Team work: Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.

10. Communication: Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Programme Specific Outcomes (PSO)

A graduate of the Computer Science and Engineering Program will demonstrate:

PSO1: Computer Science Specific Skills
The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas by understanding the core principles and concepts of computer science and thereby engage in national grand challenges.

PSO2: Programming and Software Development Skills
The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry.

PSO3: Professional Skills
The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur.

Course Outcomes (CO)

After the completion of the course the student will be able to:

Course Outcome 1: Model and solve real world problems by applying knowledge across domains (Cognitive knowledge level: Apply).

Course Outcome 2: Develop products, processes or technologies for sustainable and socially relevant applications (Cognitive knowledge level: Apply).

Course Outcome 3: Function effectively as an individual and as a leader in diverse teams and to comprehend and execute designated tasks (Cognitive knowledge level: Apply).

Course Outcome 4: Plan and execute tasks utilizing available resources within timelines, following ethical and professional norms (Cognitive knowledge level: Apply).

Course Outcome 5: Identify technology/research gaps and propose innovative/creative solutions (Cognitive knowledge level: Analyze).

Course Outcome 6: Organize and communicate technical and scientific findings effectively in written and oral forms (Cognitive knowledge level: Apply).

Appendix C: CO-PO-PSO Mapping

COURSE OUTCOMES:

After completion of the course, the student will be able to:

SL.NO	DESCRIPTION	Bloom's Taxonomy Level
CO1	Model and solve real-world problems by applying knowledge across domains (Cognitive knowledge level:Apply).	Level3: Apply
CO2	Develop products, processes, or technologies for sustainable and socially relevant applications. (Cognitive knowledge level:Apply).	Level 3: Apply
CO3	Function effectively as an individual and as a leader in diverse teams and comprehend and execute designated tasks. (Cognitive knowledge level:Apply).	Level 3: Apply
CO4	Plan and execute tasks utilizing available resources within timelines, following ethical and professional norms (Cognitive knowledge level:Apply).	Level 3: Apply
CO5	Identify technology/research gaps and propose innovative/creative solutions (Cognitive knowledge level:Analyze).	Level 4: Analyze
CO6	Organize and communicate technical and scientific findings effectively in written and oral forms (Cognitive knowledge level:Apply).	Level 3: Apply

CO-PO AND CO-PSO MAPPING

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	2	2	2	1	2	2	2	1	1	1	1	2	3		
CO2	2	2	2		1	3	3	1	1		1	1		2	
CO3									3	2	2	1			3
CO4					2			3	2	2	3	2			3
CO5	2	3	3	1	2							1	3		
CO6					2			2	2	3	1	1			3

3/2/1: high/medium/low

JUSTIFICATIONS FOR CO-PO MAPPING

Mapping	Level	Justification
101003/CS822U.1-PO1	M	Knowledge in the area of technology for project development using various tools results in better modeling.
101003/CS822U.1-PO2	M	Knowledge acquired in the selected area of project development can be used to identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions.
101003/CS822U.1-PO3	M	Can use the acquired knowledge in designing solutions to complex problems.
101003/CS822U.1-PO4	M	Can use the acquired knowledge in designing solutions to complex problems.
101003/CS822U.1-PO5	H	Students are able to interpret, improve, and redefine technical aspects for design of experiments, analysis, and interpretation of data, and synthesis of the information to provide valid conclusions.
101003/CS822U.1-PO6	M	Students are able to interpret, improve, and redefine technical aspects by applying contextual knowledge to assess societal, health, and consequential responsibilities relevant to professional engineering practices.
101003/CS822U.1-PO7	M	Project development based on societal and environmental context solution identification is the need for sustainable development.
101003/CS822U.1-PO8	L	Project development should be based on professional ethics and responsibilities.
101003/CS822U.1-PO9	L	Project development using a systematic approach based on well-defined principles will result in teamwork.
101003/CS822U.1-PO10	M	Project brings technological changes in society.
101003/CS822U.1-PO11	H	Acquiring knowledge for project development gathers skills in design, analysis, development, and implementation of algorithms.

101003/CS822U.1-PO12	H	Knowledge for project development contributes engineering skills in computing and information gatherings.
101003/CS822U.2-PO1	H	Knowledge acquired for project development will also include systematic planning, developing, testing, and implementation in computer science solutions in various domains.
101003/CS822U.2-PO2	H	Project design and development using a systematic approach brings knowledge in mathematics and engineering fundamentals.
101003/CS822U.2-PO3	H	Identifying, formulating, and analyzing the project results in a systematic approach.
101003/CS822U.2-PO5	H	Systematic approach is the tip for solving complex problems in various domains.
101003/CS822U.2-PO6	H	Systematic approach in the technical and design aspects provides valid conclusions.
101003/CS822U.2-PO7	H	Systematic approach in the technical and design aspects demonstrates the knowledge of sustainable development.
101003/CS822U.2-PO8	M	Identification and justification of technical aspects of project development demonstrates the need for sustainable development.
101003/CS822U.2-PO9	H	Apply professional ethics and responsibilities in engineering practice of development.
101003/CS822U.2-PO11	H	Systematic approach also includes effective reporting and documentation, which gives clear instructions.
101003/CS822U.2-PO12	M	Project development using a systematic approach based on well-defined principles will result in better teamwork.
101003/CS822U.3-PO9	H	Project development as a team brings the ability to engage in independent and lifelong learning.

101003/CS822U.3-PO10	H	Identification, formulation, and justification in technical aspects will be based on acquiring skills in design and development of algorithms.
101003/CS822U.3-PO11	H	Identification, formulation, and justification in technical aspects provides the betterment of life in various domains.
101003/CS822U.3-PO12	H	Students are able to interpret, improve, and redefine technical aspects with mathematics, science, and engineering fundamentals for the solutions of complex problems.
101003/CS822U.4-PO5	H	Students are able to interpret, improve, and redefine technical aspects with identification, formulation, and analysis of complex problems.
101003/CS822U.4-PO8	H	Students are able to interpret, improve, and redefine technical aspects to meet the specified needs with appropriate consideration for public health and safety, and the cultural, societal, and environmental considerations.
101003/CS822U.4-PO9	H	Students are able to interpret, improve, and redefine technical aspects for design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
101003/CS822U.4-PO10	H	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools for better products.
101003/CS822U.4-PO11	M	Students are able to interpret, improve, and redefine technical aspects by applying contextual knowledge to assess societal, health, and consequential responsibilities relevant to professional engineering practices.
101003/CS822U.4-PO12	H	Students are able to interpret, improve, and redefine technical aspects for demonstrating the knowledge of, and need for sustainable development.

101003/CS822U.5-PO1	H	Students are able to interpret, improve, and redefine technical aspects, apply ethical principles, and commit to professional ethics and responsibilities and norms of the engineering practice.
101003/CS822U.5-PO2	M	Students are able to interpret, improve, and redefine technical aspects, communicate effectively on complex engineering activities with the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
101003/CS822U.5-PO3	H	Students are able to interpret, improve, and redefine technical aspects to demonstrate knowledge and understanding of the engineering and management principle in multidisciplinary environments.
101003/CS822U.5-PO4	H	Students are able to interpret, improve, and redefine technical aspects, recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.
101003/CS822U.5-PO5	M	Students are able to interpret, improve, and redefine technical aspects in acquiring skills to design, analyze, and develop algorithms and implement those using high-level programming languages.
101003/CS822U.5-PO12	M	Students are able to interpret, improve, and redefine technical aspects and contribute their engineering skills in computing and information engineering domains like network design and administration, database design, and knowledge engineering.
101003/CS822U.6-PO5	M	Students are able to interpret, improve, and redefine technical aspects and develop strong skills in systematic planning, developing, testing, implementing, and providing IT solutions for different domains, which helps in the betterment of life.

101003/CS822U.6-PO8	H	Students will be able to associate with a team as an effective team player for the development of technical projects by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
101003/CS822U.6-PO9	H	Students will be able to associate with a team as an effective team player to identify, formulate, review research literature, and analyze complex engineering problems.
101003/CS822U.6-PO10	M	Students will be able to associate with a team as an effective team player for designing solutions to complex engineering problems and design system components.
101003/CS822U.6-PO11	M	Students will be able to associate with a team as an effective team player, use research-based knowledge and research methods including design of experiments, analysis, and interpretation of data.
101003/CS822U.6-PO12	H	Students will be able to associate with a team as an effective team player, applying ethical principles and committing to professional ethics and responsibilities and norms of the engineering practice.
101003/CS822U.1-PSO1	H	Students are able to develop Computer Science Specific Skills by modeling and solving problems.
101003/CS822U.2-PSO2	M	Developing products, processes or technologies for sustainable and socially relevant applications can promote Programming and Software Development Skills.
101003/CS822U.3-PSO3	H	Working in a team can result in the effective development of Professional Skills.
101003/CS822U.4-PSO3	H	Planning and scheduling can result in the effective development of Professional Skills.
101003/CS822U.5-PSO1	H	Students are able to develop Computer Science Specific Skills by creating innovative solutions to problems.

101003/CS822U.6-PSO3	H	Organizing and communicating technical and scientific findings can help in the effective development of Professional Skills..