

A

Project Report

On

“Cricket PoseNet : AI Shot Assistant”

By

Gyanendra Pratap Shukla (22721)

Mohd Umair (22733)

Shri Kant (22755)

Under the Guidance of

Prof. Samir Srivastava

Prof. Garima Yadav

For Partial Fulfilment of Award of Degree

In

Master of Computer Application

at



Department of Computer Science & Engineering

Kamla Nehru Institute of Technology, Sultanpur

Affiliated to Dr. A.P.J. Abdul Kalam Technical University, Lucknow

2023-2024

CERTIFICATE

Certified that **Gyanendra Pratap Shukla (22721)**, **Mohd Umair (22733)** and **Shri Kant (22755)** have carried out the project work presented in this project report entitled “**Cricket PoseNet : AI Shot Assistant**” for the award of Master of Computer Application(MCA) from Department of Computer Science and Engineering, Kamla Nehru Institute of Technology, Sultanpur, affiliated to Dr. A. P. J. Abdul Kalam Technical University, Lucknow.

This report is the record of the candidates own work carried out by them under our supervision and guidance. This project work is part of their Master of Computer Application curriculum. Their performance was excellent and we wish them good luck for their future endeavours.

.....
Prof. Samir Srivastava
(Supervisor)

.....
Prof. Awadhesh Kumar
(Head of the Department)

.....
Prof. Garima Yadav
(Supervisor)

ACKNOWLEDGEMENT

I started this project as a part of my course curriculum. It gives me great pleasure to present this project work conducted towards the fulfilment of the project titled Cricket PoseNet : AI Shot Assistant. I take this opportunity to thank those who have made efforts to ensure the success of the project. I extend my special gratitude to Prof. Samir Srivastava and Prof. Garima Yadav (Department of Computer Science and Engineering) who has been a source of motivation, encouragement, and guidance for doing this project. I express my warm wishes to the entire staff members for their assistance and kind guidance who helped me to find the project. I would like to express our thanks to Prof. Awadhesh Kumar, H.O.D. of Department of Computer Science and Engineering for their contributions and support towards this project. We also extend our thanks to Dr. R.K. Upadhyay, director of Kamla Nehru Institute of Technology for providing us with a conducive academic environment and for enabling us to pursue our academic aspirations. Additionally, we would like to express our gratitude to the entire staff of the department for their continuous support, guidance, and suggestions. Lastly, we would like to thank our parents, whose unwavering love and support have been a constant source of strength for us. We are forever grateful for their encouragement and guidance.

ABSTRACT

The "Cricket PoseNet : AI Shot Assistant" project aims to revolutionize cricket analysis and training through advanced computer vision and artificial intelligence techniques. This innovative system leverages cutting-edge technologies such as YOLO (You Only Look Once), ANN (Artificial Neural Networks) and Time-Distributed CNN (Convolutional Neural Networks), LSTM (Long Short-Term Memory) to precisely classify cricket shots from images and videos respectively. Additionally, the system incorporates Pose Detection algorithms to predict player positions and postures, enhancing the depth of analysis.

The data is collected from online sources and pre-processed through cleaning, resizing, and organizing. Similarly, an intuitive deep model is designed with a combination of Time-Distributed 2D CNN layers and LSTM cells for extracting and learning the spatiotemporal information from the input sequences.

The project's primary objectives include developing accurate shot classification models, ensuring real-time processing for applications, and providing actionable insights for cricket enthusiasts, coaches, and analysts. The system's architecture employs microservices for modularity and scalability, with components including a ReactJS for frontend, FastAPI and Spring Boot for backend services, MySQL for data storage, and TensorFlow and OpenCV for advanced image and video analysis. Key features of the system include high accuracy in shot classification, efficient processing times, user-friendly interfaces, and robust error handling mechanisms.

The results from system testing demonstrate the system's effectiveness in accurately classifying cricket shots across various scenarios, ensuring prompt response times, and providing an intuitive user experience. Future enhancements will focus on incorporating advanced algorithms, optimizing model architectures, and expanding training datasets to further enhance system capabilities and address emerging challenges in cricket analysis.

Overall, the "Cricket PosNet : AI Shot Assistant" project represents a significant advancement in sports analytics, offering a comprehensive solution for cricket shot classification, player analysis, and training support.

TABLE OF CONTENT

Chapter 1 : Introduction

1.1 Motivation	2
1.2 Problem Statement	2
1.3 Objectives.....	3
1.4 Summary	3

Chapter 2 : Literature Survey

2.1 Overview of Cricket Shot Classification.....	4
2.2 Deep Learning in Sports Analytics	4
2.3 Fast API and Its Applications.....	4
2.4 Spring Boot and Its Applications	5
2.5 MySQL Database	5
2.6 Computer Vision in Sports Technology	5
2.7 ReactJS and Tailwind CSS in Frontend Development.....	5

Chapter 3 : Methodology Used

3.1 Overview of Methodology	6
3.2 Datasets	6
3.3 Data Collection and Preprocessing	6
3.4 Deep Learning Models Architecture	7
3.5 Integration of FastAPI and Spring Boot.....	7
3.6 Frontend Development with ReactJS and Tailwind CSS.....	8

Chapter 4 : System Requirements Specifications (SRS)

4.1 Functional Requirements.....	9
4.2 Non-functional Requirements	10
4.3 Software and Hardware Requirements.....	11
4.4 Constraints.....	12

Chapter 5 : System Design

5.1 Overview Diagram	13
5.2 Workflow Diagram	15
5.3 Process Flow Diagram	16
5.4 ER Diagram.....	16
5.5 MVC Architecture.....	17

Chapter 6 : Implementation

6.1 Introduction	19
6.2 Frontend	19
6.3 Backend.....	36
6.4 Algorithm	44
6.5 Packages/Libraries Used	48
6.6 Testing	49

Chapter 7 : Result and Analysis

7.1 Accuracy.....	51
7.2 Classification Report	51
7.3 Confusion Matrix	52
7.4 Training and Validation Loss Curves.....	53
7.5 Analysis.....	54

Conclusion	55
-------------------	----

References	56
-------------------	----

Chapter 1

INTRODUCTION

Welcome to "Cricket PoseNet : AI Shot Assistant," your ultimate destination for cricket enthusiasts and passionate fans of the sport. Our innovative website is designed to revolutionize the way you experience cricket by bringing cutting-edge technology and real-time data analysis to your fingertips. It is not just another cricket website; it's a game-changer that combines the thrill of cricket with the power of artificial intelligence. With our state-of-the-art image and video recognition technology, we can tell you which shot a batsman has played in an uploaded image or video, providing an interactive and insightful experience like never before.

This project is dedicated to the precise classification of cricket shots, employing cutting-edge technologies like YOLO (You Only Look Once) and ANN (Artificial Neural Networks) for image classification, and Time-Distributed CNN (Convolutional Neural Networks) and LSTM (Long Short-Term Memory) for video analysis within the realm of advanced computer vision. A fundamental aspect of Computer Vision, known as Pose Detection, assumes a pivotal role in this endeavour. It involves the sophisticated prediction of the positions and postures of individuals or objects captured within images. The primary aim of this undertaking is to categorize cricket shots based on the specific poses adopted by players in the images and videos.

By harnessing the combined capabilities of Pose Detection and Deep Learning, we unlock the ability to not only identify the type of cricket shot being executed but also to derive invaluable insights for cricket analysis and training. It's akin to having a perceptive eye that can instantly recognize and categorize a wide array of cricketing actions.

This project employs a multi-faceted approach, encompassing deep learning methodologies, extensive data augmentation techniques, and rigorous model training procedures. This ensures that the model becomes adept at generalizing across various player poses and shot types. Through these techniques, the model achieves a high level of proficiency in recognizing and classifying cricket shots with a remarkable degree of accuracy. Ultimately, this project aspires to yield a robust and automated solution for cricket shot classification, poised to offer

indispensable support for coaches, analysts, and cricket enthusiasts seeking profound insights into the game.

1.1 Motivation

Our motivation for "Cricket PoseNet: AI Shot Assistant" is driven by a passion for cricket and a quest for innovative sports technology. We aim to revolutionize shot classification by leveraging AI and computer vision, providing accurate, real-time insights to enhance cricket analysis and enjoyment for players, coaches, analysts, and fans alike.

- **Passion for Cricket :** Driven by a deep love and enthusiasm for the sport.
- **Quest for Innovation :** Striving to explore new frontiers in sports technology.
- **Revolutionizing Shot Classification :** Using AI and computer vision to transform how shots are analyzed.
- **Real-time Insights :** Providing accurate and instant feedback for players, coaches, and analysts.
- **Enhancing Cricket Experience :** Improving the overall enjoyment and understanding of the game for fans.
- **Commitment to Excellence :** Dedicated to pushing the boundaries of sports technology for cricket's advancement.

1.2 Problem Statement

Manual shot analysis in cricket is prone to human error, lacks real-time insights, and can be overwhelming when dealing with vast amounts of data. The aim is to automate this process, providing a faster, more objective, and efficient means of shot classification.

- Prone to human error due to subjective judgement.
- Lack of real-time insights hinders timely decision-making during matches.
- Handling large volumes of data manually is time-consuming and error-prone.
- Inconsistencies in shot classification can lead to inaccurate performance analysis.
- Limited scalability for comprehensive shot analysis across various cricket scenarios.
- Improve efficiency by reducing manual effort and processing time.
- Scale the solution to handle diverse cricket scenarios and datasets effectively.

1.3 Objectives

- Develop a robust deep learning model capable of accurately classifying various cricket shots in images and videos.
- Implement a backend using FastAPI and Spring Boot to facilitate seamless communication between the frontend and the deep learning model for both image and video classification, as well as authentication and authorization functionalities.
- Utilize YOLO (You Only Look Once), Time-Distributed CNN(Convolutional Neural Networks), LSTM(Long Short Term Memory) for computer vision tasks, enabling precise shot localization and extraction in real-time.
- Create an intuitive and user-friendly frontend using ReactJS and Tailwind CSS to visualize and interact with the classification results for both images and videos.
- Perform comprehensive shot analysis to support coaches, players, and analysts in making informed decisions based on accurate and timely insights from image and video classifications.

1.4 Summary

The "Cricket PoseNet : AI Shot Assistant" project is dedicated to automating the classification of common cricket shots, including flicks, drives, pull shots, and cut shots. It covers the entire process from data collection to frontend visualization, leveraging advanced technologies like deep learning, computer vision, FastAPI, Spring Boot, YOLO, ReactJS, and Tailwind CSS. The project's objectives include developing a robust deep learning model for accurate shot classification in both images and videos, implementing a backend infrastructure for seamless communication and authentication, utilizing YOLO for precise shot localization, creating an intuitive frontend for interactive visualization, and conducting comprehensive shot analysis to support cricket coaches, players, and analysts. The ultimate goal is to enhance the cricketing experience by providing real-time, objective insights that drive informed decisions and improve overall performance.

Chapter 2

LITERATURE SURVEY

The literature survey serves as a guiding beacon, illuminating our path as we navigate the complex landscape of sports analytics, particularly in the domain of cricket shot classification and performance analysis. Drawing inspiration from prior research and exploration, our project is rooted in a rich tapestry of knowledge that spans decades of innovation and discovery in the field. By synthesising insights from existing literature, we aim to contribute to the ongoing evolution of sports technology, pushing the boundaries of what is possible in cricket performance analysis.

2.1 Overview of Cricket Shot Classification

Cricket shot classification involves identifying and categorising various shots played by batsmen during a match. Traditional methods rely on manual tagging, which is time-consuming and subjective. Recent advancements leverage machine learning and computer vision to automate this process, enhancing accuracy and efficiency.

2.2 Deep Learning in Sports Analytics

Deep learning has emerged as a powerful tool in sports analytics. In the context of cricket, it enables the creation of sophisticated models capable of recognizing patterns in player behaviour. The utilisation of deep neural networks for shot classification has shown promising results, allowing for real-time analysis and insights.

2.3 FastAPI and Its Applications

FastAPI, a modern web framework for building APIs with Python, facilitates the development of robust and high-performance backend systems. Its asynchronous capabilities are particularly advantageous for applications demanding quick responses, making it a suitable choice for the real-time requirements of this project.

2.4 Spring Boot and Its Applications

Spring Boot is utilized for various backend functionalities, including authentication, authorization, and handling database queries. Its robust features streamline these essential aspects of our cricket analytics platform, ensuring security and efficiency.

2.5 MySQL Database

MySQL serves as the backend database management system, facilitating data storage, retrieval, and management for our cricket shot classification and analysis platform.

2.6 Computer Vision in Sports Technology

Computer vision plays a pivotal role in sports technology, providing the ability to analyze visual data such as image feed. In cricket, computer vision techniques, including object detection and tracking, are instrumental in precisely identifying and extracting shots played by batsmen.

2.7 ReactJS and Tailwind CSS in Frontend Development

ReactJS, a JavaScript library for building user interfaces, combined with Tailwind CSS, a utility-first CSS framework, offers a modern and efficient approach to frontend development. The component-based structure of React simplifies the creation of interactive and dynamic user interfaces, while Tailwind CSS provides a utility-centric styling approach, enhancing frontend design flexibility.

Chapter 3

METHODOLOGY USED

3.1 Overview of Methodology

The development of the cricket shot classification system involves a systematic approach, encompassing data collection, preprocessing, model training, backend implementation, and frontend development. The methodology aims to seamlessly integrate deep learning, computer vision, and web technologies.

3.2 Datasets

Image Dataset : This dataset is a collection of images from the internet, played really amazing player (So, can hope for perfection in shots). Then the data was augmented to give rise to the data I provide in front of you. The dataset includes 5772 images that represent different classes of batting activities, including pull shot, sweep, drive, legglance-flick and cut-shot .

Video Dataset : The cricket video dataset is collected from YouTube and cricket-info websites. The dataset includes 722 videos that represent different classes of batting activities, including pull shot, bowled, reverse sweep, defence, and cover drive. The videos were all recorded at the same frame rate of 30 and had the same background, ground, pitch, and spectator accommodation. Each class contains 150 videos. The dimensions of each frame are 840 x 480.

3.3 Data Collection and Preprocessing

Data Source: Cricket shot datasets, containing labelled examples of various shot types, were collected from reliable sources. These datasets form the foundation for training the deep learning models.

Preprocessing: The collected data underwent preprocessing to remove noise and ensure uniformity. Image data for computer vision tasks was resized, normalised, and augmented to enhance the model's generalisation capabilities. The video editor tool was used to extract short video clips of 1 to 3 s in duration for each of these categories.

3.4 Deep Learning Models Architecture

Utilisation of YOLO for Object Detection in Sports Analytics : The adoption of the You Only Look Once (YOLO) algorithm for object detection has become increasingly prevalent in sports analytics. YOLO's real-time capabilities and accuracy make it ideal for extracting key points in images, particularly for dynamic events like cricket shots. This signifies a move towards more efficient and precise analysis methodologies in sports technology.

Artificial Neural Networks (ANN) for Shot Prediction in Image : Following the extraction of keypoints using YOLO, Artificial Neural Networks (ANN) are employed for shot prediction in images. By applying ANN algorithms to the extracted keypoints, the system can accurately predict the type of cricket shot played. This integration of YOLO and ANN enhances the classification accuracy and efficiency of the system.

Time-Distributed CNN and LSTM for Video Analysis : For video classification, a combination of Time-Distributed Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks is utilised. This approach allows for extracting and learning the spatiotemporal information from the input frame sequences, enabling the identification and classification of different shots over time. The integration of CNN and LSTM enhances the system's ability to capture temporal dependencies and patterns in video data.

Training: These models were trained on the labelled datasets, with a focus on achieving high accuracy and minimising false positives and false negatives in shot classification.

3.5 Integration of FastAPI and Spring Boot

FastAPI Backend : FastAPI was employed to develop the backend of the system. API endpoints were created to facilitate communication between the frontend and the deep learning model. Asynchronous features of FastAPI were utilised to ensure responsive real-time interactions.

Spring Boot Backend : Spring Boot is employed for personalised information delivery, authentication, and database management using MySQL. This integration ensures seamless communication and data management within the system.

3.6 Frontend Development with ReactJS and Tailwind CSS

The frontend interface is crafted using a powerful combination of ReactJS and Tailwind CSS, ensuring a contemporary and highly responsive user experience. Leveraging React Vite's swift build times and Tailwind CSS's utility-first approach, we enable rapid development and seamless customization of frontend components. This cohesive integration not only enhances the usability but also elevates the visual appeal of the application, resulting in a seamlessly immersive user experience.

Chapter 4

System Requirements Specification (SRS)

The system requirements for the project encompass various functional and non-functional aspects, ensuring robust performance and user satisfaction.

4.1. Functional Requirements

- **Shot Prediction Functionality :**

- Users can submit image or video for shot classification. The input data is securely transmitted to the backend for processing.
- Users receive instant feedback on shot classifications
- Implement YOLO for shot extraction and keypoint detection in images.
- Apply ANN on keypoints to provide accurate shot predictions.
- Utilize Time-Distributed CNN and LSTM for shot prediction in videos.

- **User Personalization :**

- Develop user-specific systems using Spring Boot, allowing personalized data storage in MySQL.
- Enable signed-in users to access premium features such as video shot prediction.
- Utilize FastAPI to create APIs for seamless interaction with the system.

- **Analysis Functionality :**

- Create an analysis page using ReactJS and Tailwind CSS for intuitive user interfaces and comprehensive video analysis.

- **Implement Data Visualization :**

- Display shot classifications in a user-friendly and visually appealing format on the frontend.
- Include Box Plot and Keypoints to provide insights into the distribution of shot types.

- **Enable User Interface Customization :**

- Allow users to customize the theme and color scheme of the frontend interface using Tailwind CSS.

- **Password Change Functionality**

- Users should have the ability to change their passwords securely within the system.
- Implement a user-friendly interface for users to update their passwords.

4.2 Non-functional Requirements

- **Performance :** Ensure real-time or near-real-time shot classification in both images and videos. Handle multiple concurrent users efficiently without performance degradation.
- **Scalability :** Design the system to scale effectively, accommodating increased user loads without compromising performance.
- **Security :** Implement robust data security measures to protect user data and sensitive shot-related information.
- **Usability :** Develop an intuitive user interface for easy navigation and interaction with the system. Provide comprehensive user guides and documentation for understanding system functionalities.
- **Compatibility :** Ensure compatibility with various devices (desktops, tablets, smartphones) and browsers for seamless user experience. Maintain compatibility with external APIs for enhanced functionality.
- **Accuracy :** Strive for high accuracy in shot classification by regularly training and updating the deep learning models.

4.3 Software and Hardware Requirements

Software Requirements:

1. **Operating System** : x86/x64 based OS (Windows/Linux) (Recommended : Linux)

2. **Languages/Technologies Used** :

- Python : 3.10.12
- Java : 17.0.10
- FastAPI : 0.110.0
- Uvicorn : 0.29.0
- Ffmpeg : 6.1
- Ultralytics : 8.1.34
- Spring Boot : 3.22.0
- ReactJS : 18.2.0
- Tailwind CSS :
- CUDA Toolkit : 12.2
- CuDNN : 8.9.6
- TensorFlow : 2.15.0
- Keras : 2.15.0
- MySQL:8.0.22

3. **Integrated Development Environments (IDE) and Tools** :

- VS Code
- Eclipse
- Jupyter Notebook
- Google Docs
- Google Slides
- Git and github
- Canva
- Spring Tool Suite(STS)
- MySql Client
- Mysql Workbench

4. **Python Version (Recommended)** : 3.10.12 or greater

5. **pip Version** : 22.0.2 or higher

6. **Modules and Frameworks:**

- **YOLOv8 (You Only Look Once)** : For extraction of keypoints in images.

- **ANN (Artificial Neural Network)** : Applied on keypoints for shot prediction.
- **Time-distributed CNN and LSTM** : Used for video shot prediction.
- **FastAPI** : Creating APIs for the system.
- **MySQL** : Database to store details of users.
- **Spring Boot** : For user personalized features and data storage in MySQL.
- **ReactJS** : Frontend development for showing analysis.
- **Tailwind CSS** : Styling and designing user interfaces efficiently.

7. Development Environment

Virtualenv : Recommended for creating isolated Python environments, managing dependencies, and ensuring reproducibility of experiments.

Hardware Requirements:

- **Client's Side :**
 - Any computer or mobile device with an active internet connection and web browser
 - **RAM** : 2 GB
 - **Storage** : 16 GB
- **Developer's Side :**
 - **PC Type** : i5-10400 CPU@2.90 GHz
 - **Processor** : NVIDIA Ge-Force GTX 1650
 - **RAM** : 16 GB (Minimum Requirement: 8 GB)
 - **Storage** : 256 GB (Recommended Disk Space: 64 GB or higher)

4.4. Constraints

Data Quality: The accuracy of shot classification is dependent on the quality and diversity of labelled datasets used for training the deep learning model.

External APIs: The system relies on external APIs, such as Yolo for computer vision tasks. Any changes in these APIs may affect system functionality.

Chapter 5

SYSTEM DESIGN

5.1 Overview Diagram

The overview diagram illustrates the high-level architecture and components involved in the image/video analysis system.

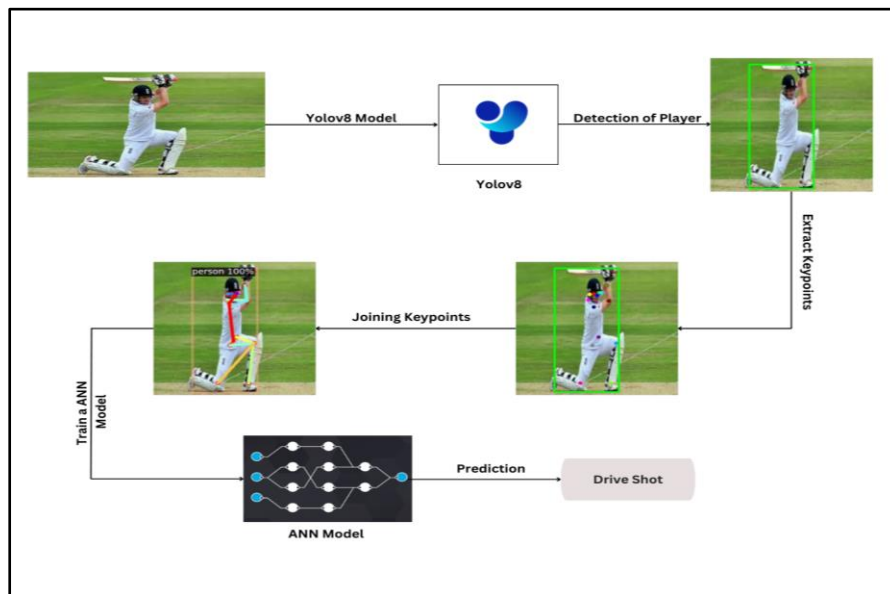


Figure 1 : Overview Diagram for Image Model

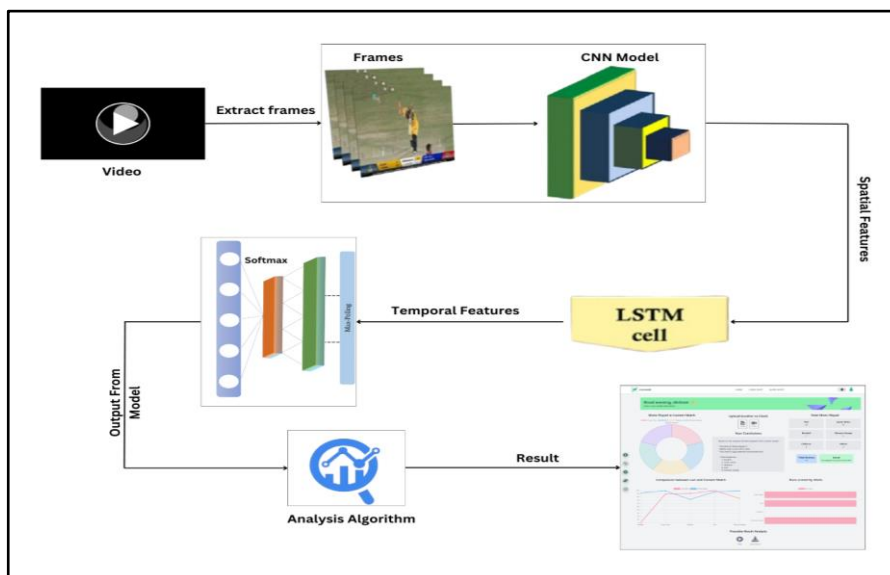


Figure 2 : Overview Diagram for Image and Video Model

This diagram provides a bird's eye view of the system, showcasing the interaction between various modules and their relationships.

A System Design involves several components that work together to achieve the objectives outlined. A high-level overview of the system design:

1. Data Collection and Preprocessing :

- Cricket shot datasets, containing labelled examples of various shot types, were collected from reliable sources.
- Annotate the data with labels indicating the type of shot (bowled, cover drive, pull, defence, reverse sweep).
- Image data preprocessing involved resizing, normalization, and augmentation to enhance model generalization.
- Video data preprocessing included resizing, extracting short clips of 1 to 3 seconds duration for each shot category.

2. Deep Learning Model Development :

- YOLO for Object Detection : Utilized for keypoint extraction in images, particularly for dynamic events like cricket shots.
- Artificial Neural Networks (ANN): Employed for shot prediction based on the extracted keypoints, enhancing classification accuracy.
- Time-Distributed CNN and LSTM : Used for video analysis to capture spatiotemporal information, enabling shot identification over time.
- Train the model using the annotated dataset to accurately classify cricket shots.

3. Backend Infrastructure :

- Implement a backend server using Fast API and Spring Boot technologies.
- Develop APIs for communication between frontend and backend, including endpoints for uploading images/videos, requesting shot classification, and retrieving analysis results.
- Implement authentication and authorization mechanisms to ensure secure access to the system.

4. Frontend Visualization :

- Develop a frontend interface using ReactJS for interactive visualization.
- Design user-friendly interfaces for uploading images/videos, displaying shot classifications, and visualizing shot analysis.
- Utilize Tailwind CSS for responsive and aesthetic UI design.

5. Comprehensive Shot Analysis :

- Develop algorithms to analyze shot data, including shot types, shot accuracy, shot probability, better shot, weak shot, etc.
- Provide visualizations and insights to users, such as shot distribution bar graph, performance trends, and comparative analyses.

6. Integration :

- Integrate all components into a cohesive system.
- Implement monitoring and logging mechanisms to track system performance and user interactions.

5.2 Workflow Diagram

The workflow diagram illustrates the workflow of tasks, actions, and decision points within the image shot classification system. It demonstrates how different components collaborate and exchange information during the classification pipeline.

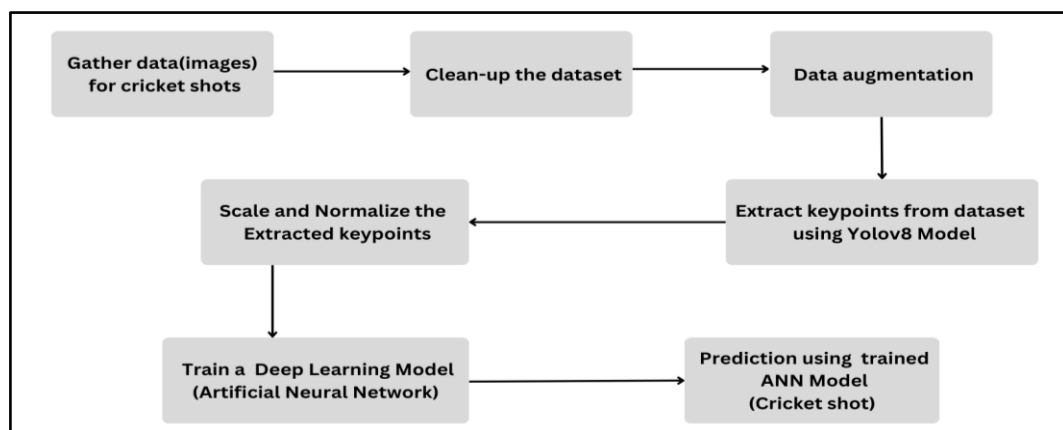


Figure 3. Workflow Diagram For Image Model

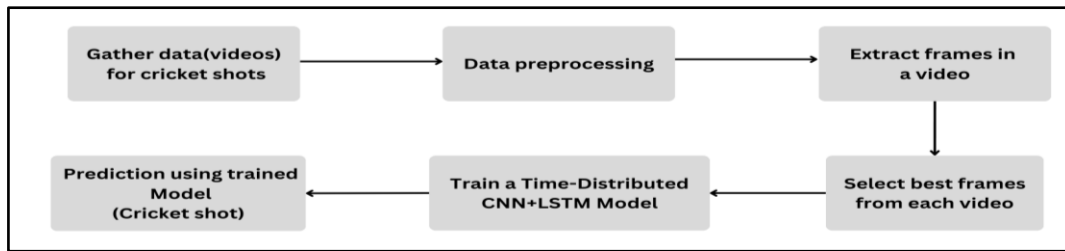


Figure 4 : Workflow Diagram For Video Model

5.3 Process Flow Diagram

The process flow diagram details the sequential steps involved in the image shot classification process. It outlines the stages from data ingestion to model training and inference.

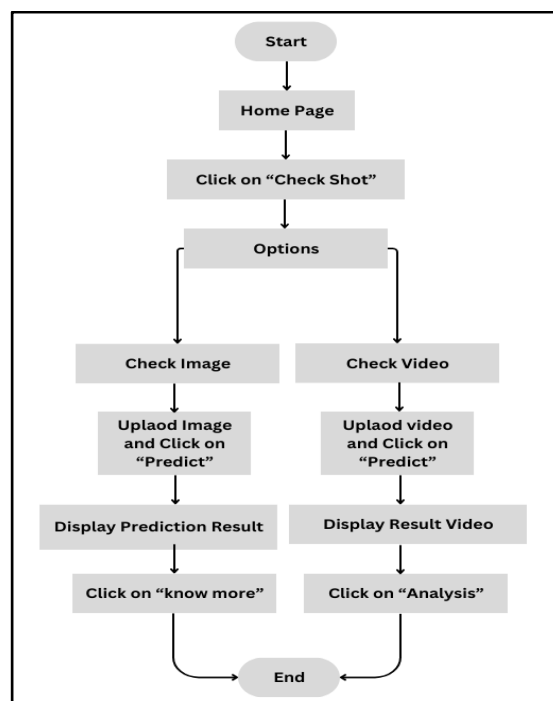


Figure 5 : Process Flow Diagram

5.4 ER Diagram

- The "User" table is connected to the "Image" table and the "Video" table through one-to-many relationships. This means that each user can have multiple images and videos associated with them, but each image or video belongs to only one user.

- The "User" table has attributes like id, username, password, email, etc.
- The "Image" table has attributes like id, user_id (foreign key referencing the User table), id,imageAddedDate, etc.
- The "Video" table has attributes like id, user_id (foreign key referencing the User table), id,videoAddedDate, etc.

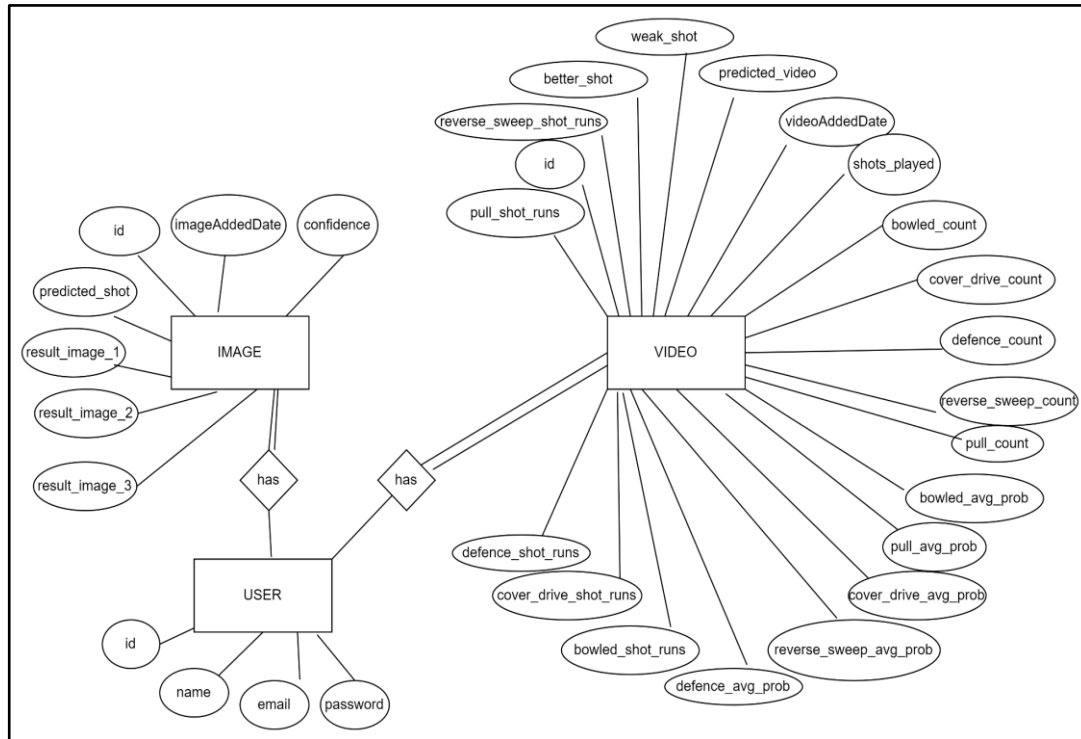


Figure 6 : ER Diagram

5.5 MVC ARCHITECTURE

- **Model :**
 - The Model represents the data and business logic of the application. It encapsulates the application's data and behavior, including data validation, manipulation, and storage.
 - The Model is responsible for interacting with the database, performing CRUD (Create, Read, Update, Delete) operations, and enforcing business rules.

- **View :**

- The View represents the presentation layer of the application. It's responsible for rendering the user interface and displaying data to the user.
- The View receives data from the Controller and presents it to the user in a visually appealing and interactive manner.

- **Controller :**

- The Controller acts as an intermediary between the Model and the View. It handles user requests, processes input, and updates the Model accordingly.
- The Controller receives HTTP requests from the client, invokes appropriate methods in the Model to perform business logic or data manipulation, and then selects the appropriate View to render the response.
- The Controller also handles form submissions, validates user input, and manages the flow of control within the application.

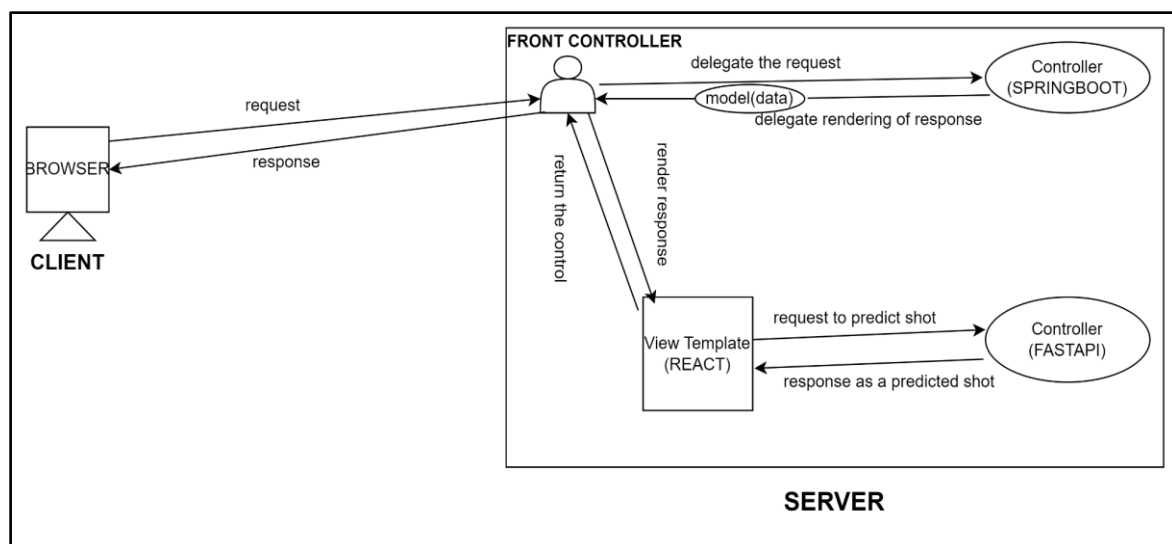


Figure 7 : MVC Architecture

Chapter 6

IMPLEMENTATION

6.1 Introduction

Our project utilizes microservices architecture for modularity, scalability, and efficient development. This involves breaking down the application into independent services with well-defined purposes like object detection and shot prediction. These services communicate via lightweight APIs, enabling faster development cycles and easier maintenance. Microservices also facilitate horizontal scaling, allowing resource-intensive services to scale independently. This ensures a robust and adaptable system for cricket shot analysis.

6.2 Frontend

The frontend of our project is a meticulously crafted React application, boasting a robust architecture and a plethora of noteworthy features. Tailwind CSS takes center stage for styling, evident in the index.css file where custom scrollbar styling elevates the sophistication of the user interface.

Major parts and components :

1. Home Page:

- Acts as the welcoming gateway to our application.
- Showcases a striking banner, informative sections detailing checkshots, cricket ground specifics, and a dedicated team section.

2. Checkshot Pages (Image and Video):

- Empowers users to conduct shot analysis for both images and videos.
- Facilitates input of shot-related data and provides insights and feedback.

3. Learn Shot Page:

- Delivers comprehensive insights into various cricket shots.
- Presents detailed descriptions, techniques, and animated GIFs for shots such as straight drive, off drive, on drive, cover drive, square drive, and backfoot drive.

4. Sign In and Sign Up Pages:

- Streamlines user authentication and account creation processes.

5. Personalized Dashboard Page:

Provides a tailored post-login experience for each user, featuring:

- **Profile Page:**

- Presents personalized account information and searched shots.
- Showcases insights into previously analysed videos, such as the number of different shots analysed and their accuracy through interactive charts.

- **Activity Page:**

- Provides a snapshot of the user's recent interactions within the application.
- Displays the four most recent image searches and three most recent video searches for quick reference.

- **Settings Page:**

- Offers personal information like email and username.
- Features a user-friendly interface for updating profile information and changing passwords.

- **Help Page:**

- Offers comprehensive guidance on utilizing the application's features for image and video search.
- Includes contact information for customer support or assistance.

6. Header and Footer:

- Ensures consistent navigation throughout the application with a header and a theme toggle for seamless transition between light and dark modes.
- Upon logging in, the header dynamically adapts to reflect the user's authentication status, featuring an avatar symbol for personalization
- Footer incorporates a subscription form and copyright information.

Header Component

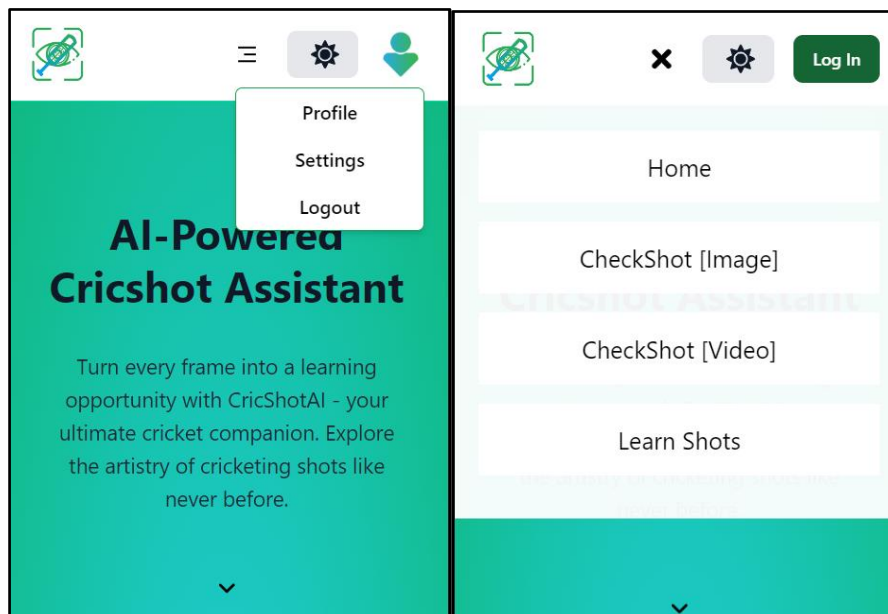
The Header component is an essential part of the cricshotAI website, facilitating navigation and theme-switching functionalities.



Key Features:

- **Responsive Design:**

- Adapts gracefully to different screen sizes, featuring a collapsible menu for smaller screens.
- Prioritizes a mobile-friendly experience with a responsive design



- **Navigation:**

- Primary navigation links include "Home," "Check Shot," and "Learn Shots."
- Links are designed to close the menu on mobile devices, enhancing usability.

- **Authentication:**

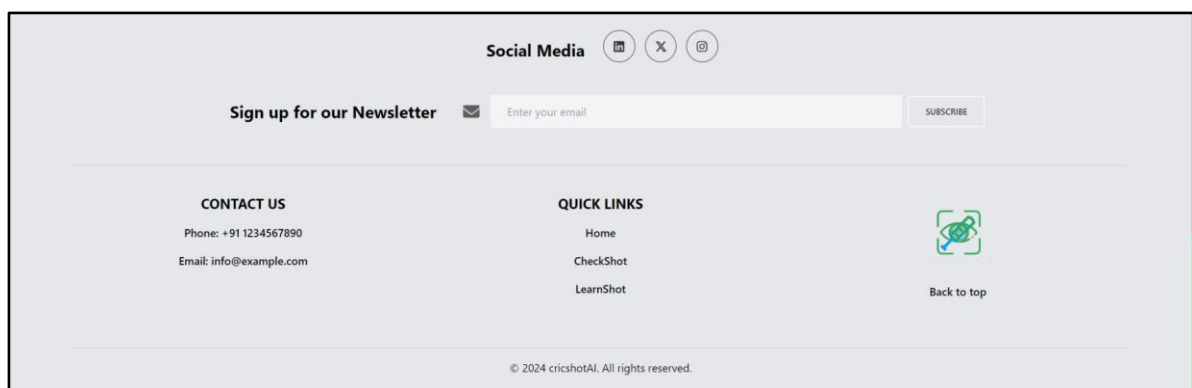
- Implements user authentication functionalities such as checking if the user is signed in and fetching current user details.
- Provides a "Log In" button for unsigned users and a dropdown menu for signed-in users with options for profile, settings, and logout.

- **Theme Switching:**

- Integrates the ThemeButton component, allowing users to toggle between light and dark themes.
- Enhances user customization based on their theme preference.

Footer Component

The Footer component provides subscription functionality, social media links, contact information, and quick links.



Key Features:

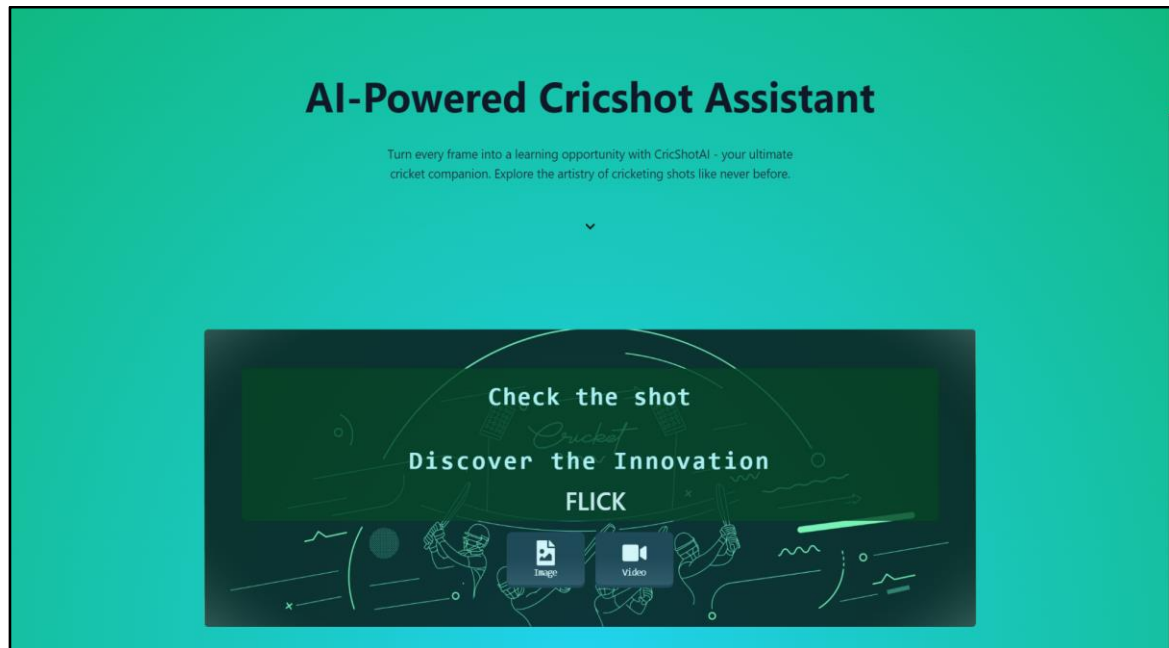
- **Social Media Links:** Displays icons for LinkedIn, Twitter, and Instagram.
- **Subscription Form:** Allows users to subscribe to newsletters with email validation.
- **Toast Notifications:** Provides dynamic feedback messages for subscription success or failure.
- **Contact and Quick Links:** Includes contact details and links to essential sections like Home, CheckShot, and LearnShot.
- **Back to Top Button:** Enables users to quickly navigate to the top of the page.

Home Page

The homepage welcomes users with dynamic text animations, scroll-triggered effects, and informative sections on AI shot recognition, cricket shots, FAQs, and team profiles. These elements combine to create an engaging and informative landing page experience, inviting users to delve deeper into the application's offerings with style and substance,

Hero Section Component:

Functionality : The Hero Section component is the focal point of the homepage, featuring dynamic text animations and scroll-triggered effects to engage users from the outset. Its captivating visuals and seamless navigation set the tone for exploring the website's offerings and features.

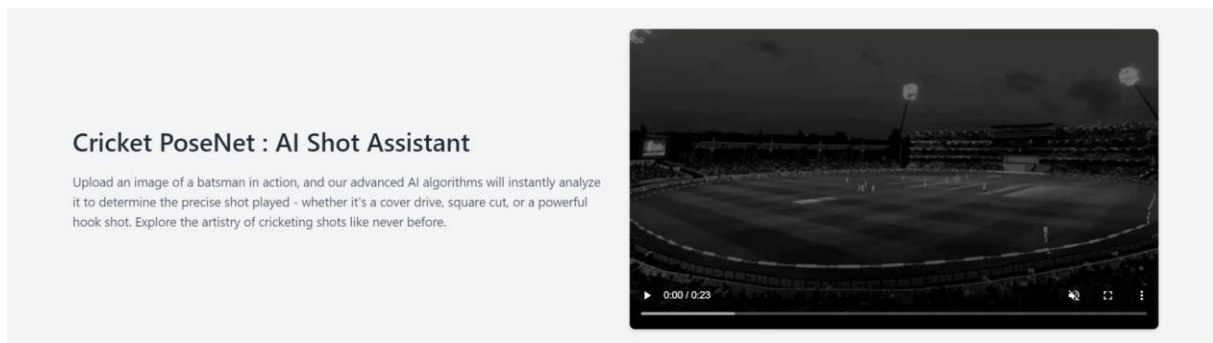


Key Features:

- **Dynamic Text Animation:** Displays cricketing shots dynamically with a typing effect.
- **Background Image:** Utilizes a changing background image based on the theme.
- **Scroll Animation:** Implements scroll-triggered animations for a visually appealing experience.
- **Shot Exploration:** Provides options to explore cricket shots through images and videos.
- **Responsive Design:** Ensures optimal viewing across various screen sizes.

About CheckShot Component:

Functionality : Provides information about Cricket PoseNet AI shot recognition system.

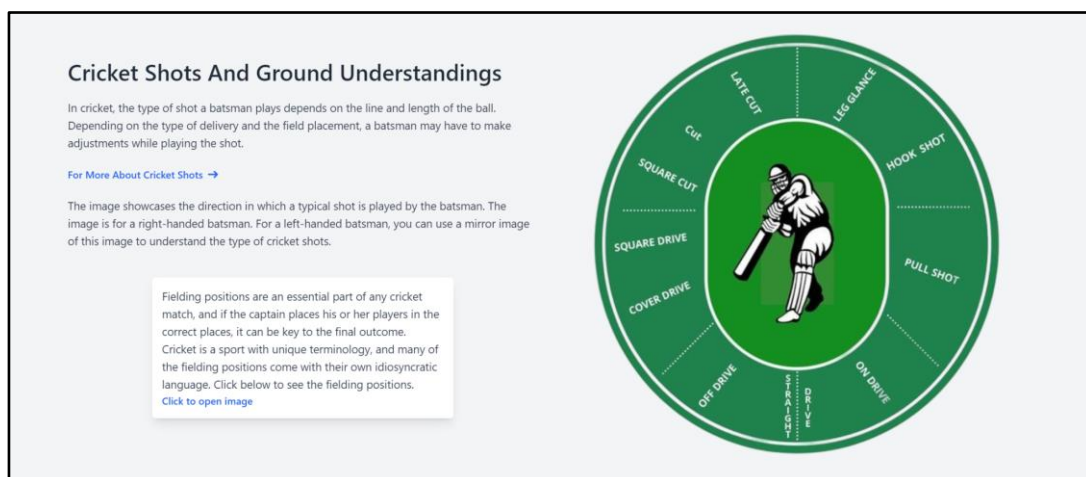


Key Features:

- **AI Shot Assistant :** Features a video showcasing AI shot recognition.
- **Text Content:** Explains AI functionality, emphasizing the ability to analyze a batsman's pose.

About Ground Component:

Functionality: Focuses on cricket shots and ground understandings.



Key Features:

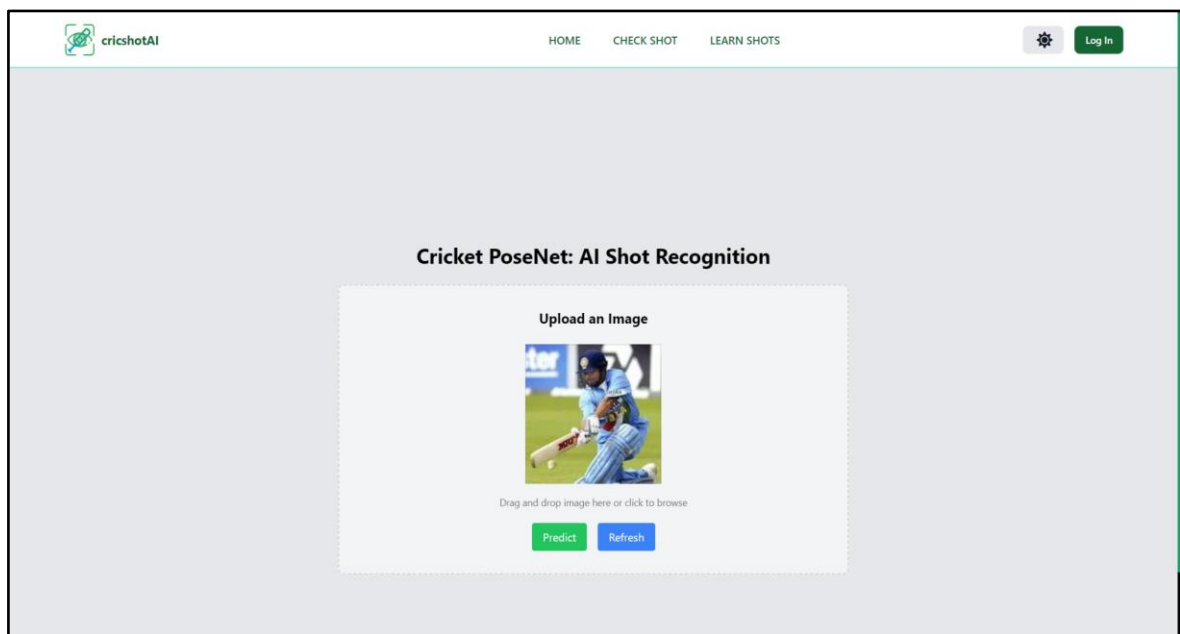
- **Shot Direction Image:** Displays an image depicting the direction of a typical shot for a right-handed batsman.
- **Fielding Positions:** Provides information about fielding positions with a clickable popup image.

CheckShot Image Page

The Checkshot component is a React component designed for a page where users can upload an image to predict and analyze cricket shots. Here are the key features:

1. Image Upload and Processing:

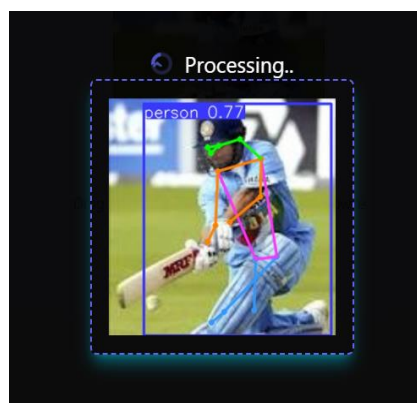
- Users can upload images interactively via drag-and-drop or a file input.
- Utilizes the Fetch API to send image data to an external prediction endpoint.
- Asynchronous handling ensures a smooth user experience during prediction.



2. Dynamic Image Display:

- Uploaded and default images are dynamically displayed.
- Slideshow-like behaviour showcases predicted shot images, changing every 3 seconds.

3. Modal for Predicted Images:



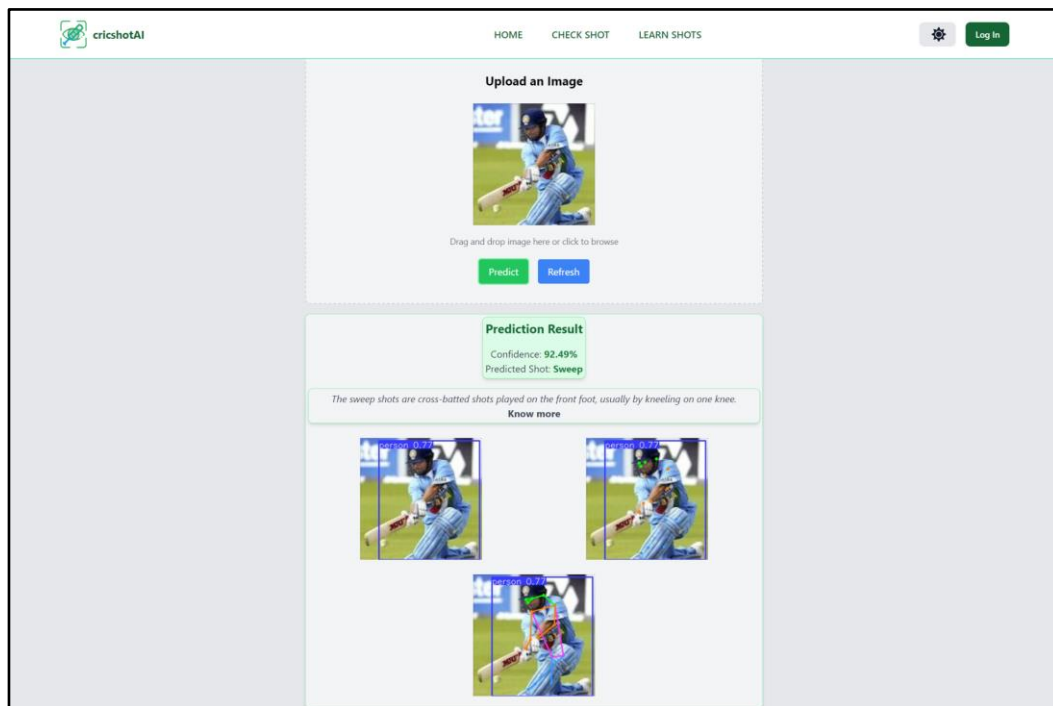
- When predictions are available, a modal overlay is triggered, offering a detailed view of the predicted cricket shots.
- Includes visual cues and a loading indicator for a seamless user experience.

4. Styling and Animations:

- Utilizes Tailwind CSS for modern and responsive design.
- AOS library enhances visual appeal with smooth transitions.

5. Additional Features:

- Displays confidence level for predicted age, adding transparency to predictions.
- Provides informative content about the predicted shot, including a description and a link for further learning.



6. User Interaction:

- Users can refresh the page to initiate a new prediction process.
- Presents a loader during image processing for responsive interface.

7. Responsive Design:

- Crafted to adapt gracefully to various screen sizes and resolutions.

8. Error Handling:

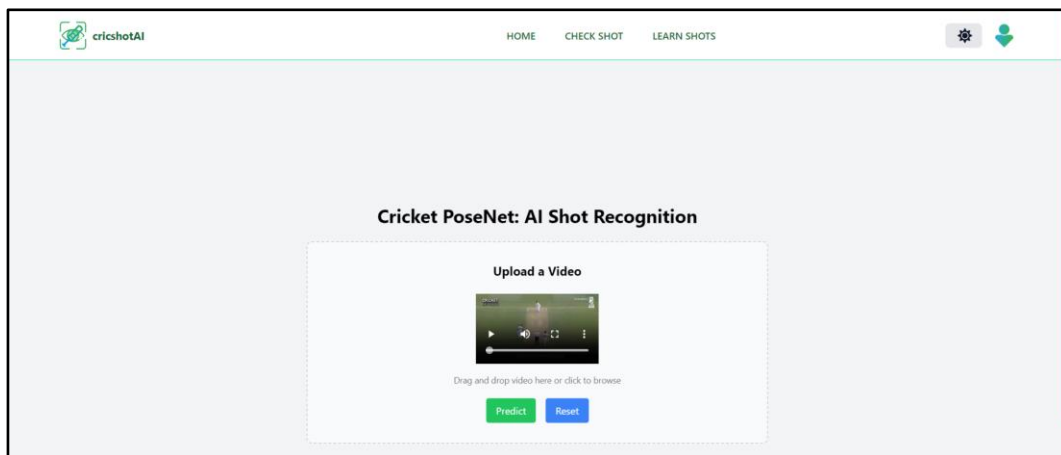
- Robust error-handling mechanisms include user alerts for specific scenarios such as invalid image uploads or API errors.

9. Integration with External Services:

- Seamless integration with an external API for image prediction demonstrates versatility and effectiveness in interacting with external services.

CheckShot Video Page

Checkshot Video is your go-to React component for analyzing cricket shots . Upload your videos, get predictions, and dive into detailed shot analysis effortlessly.

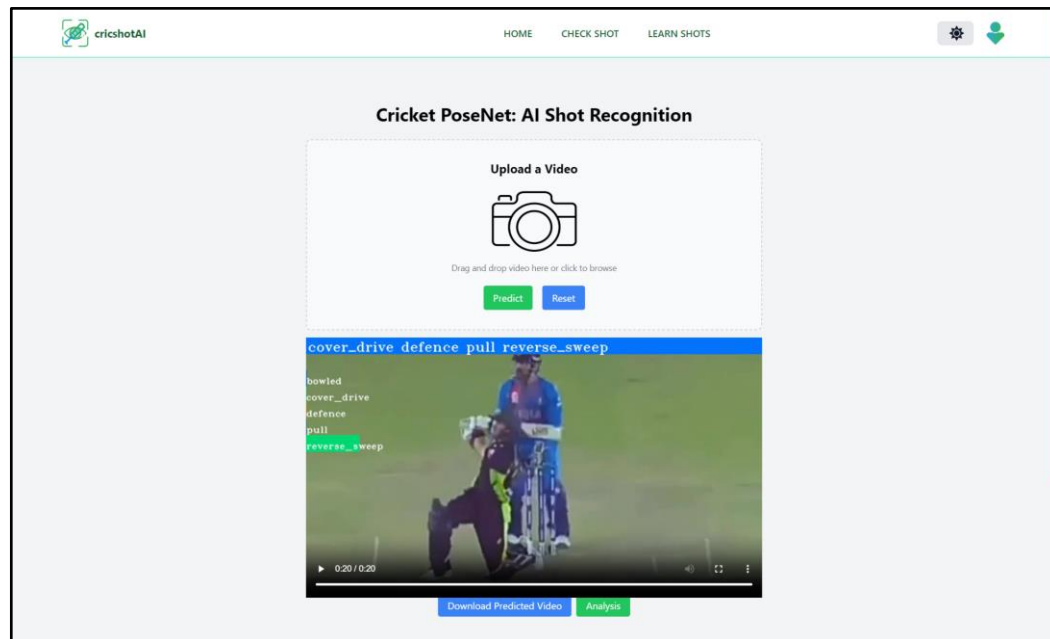


1. Event Handlers:

- Manages drag-and-drop, video and image uploads, prediction initiation, and result resetting.

2. Rendering:

- Displays the video upload area with drag-and-drop functionality.
- Buttons for predicting and resetting.
- Loader component for indicating processing.
- Display of predicted video with download and analysis options.

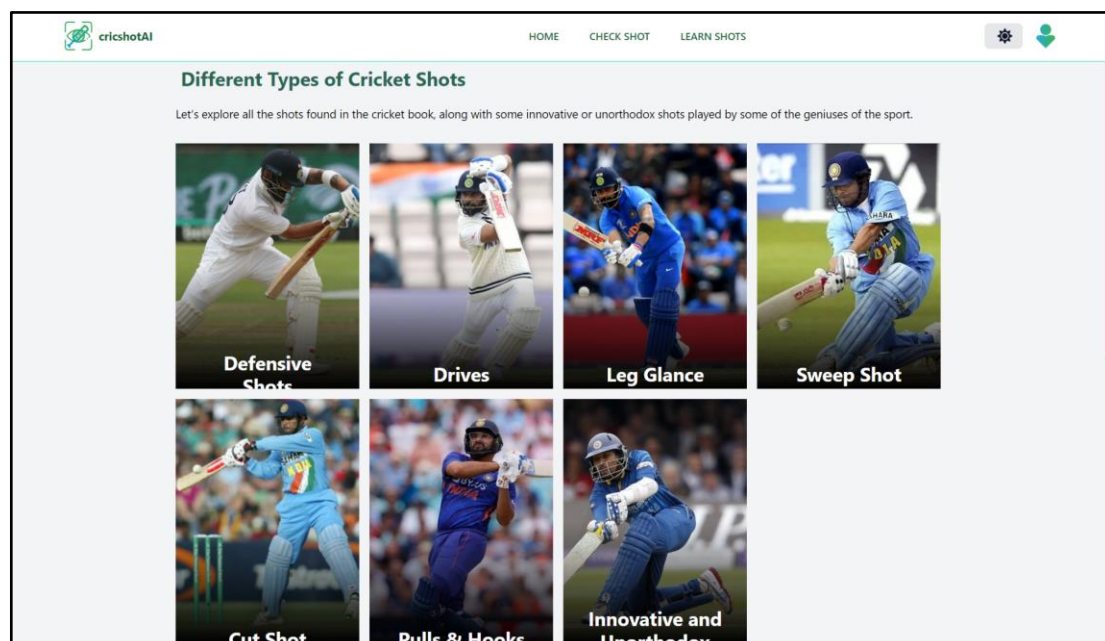


6.Backend Integration:

- Sends uploaded video to a prediction endpoint and handles the response.
- Saves predicted video data to the server for analysis using a custom service.

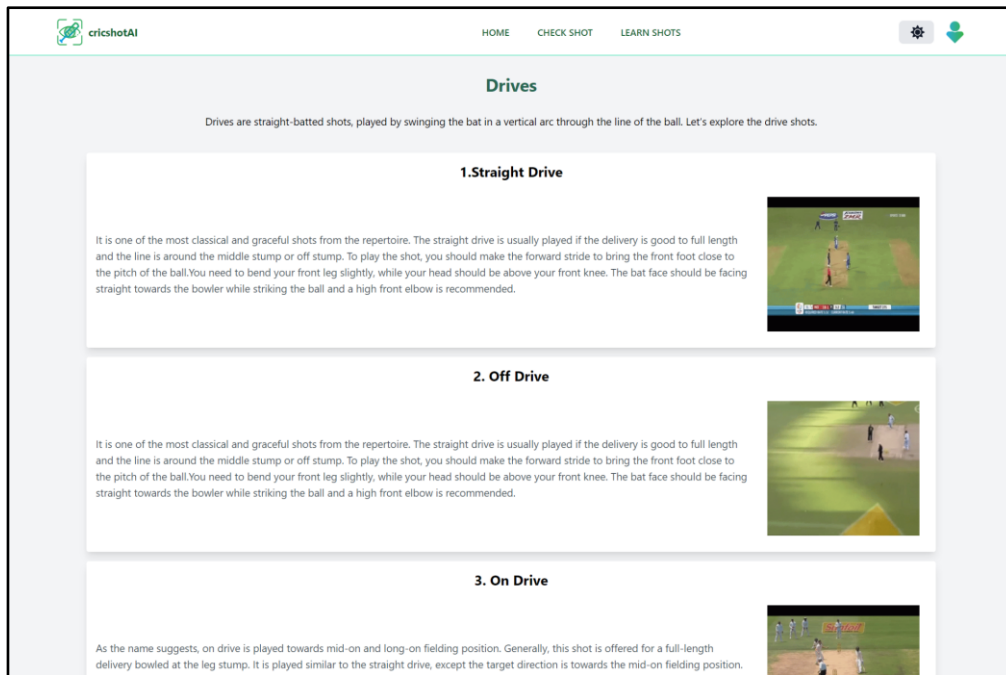
Shot Content Page

Functionality : This component serves as the main page for displaying various cricket shots.



About Shot Page

The combination of the ShotCard component and the About Shots page provides a comprehensive and visually appealing presentation of various cricket shots.



Key Features :

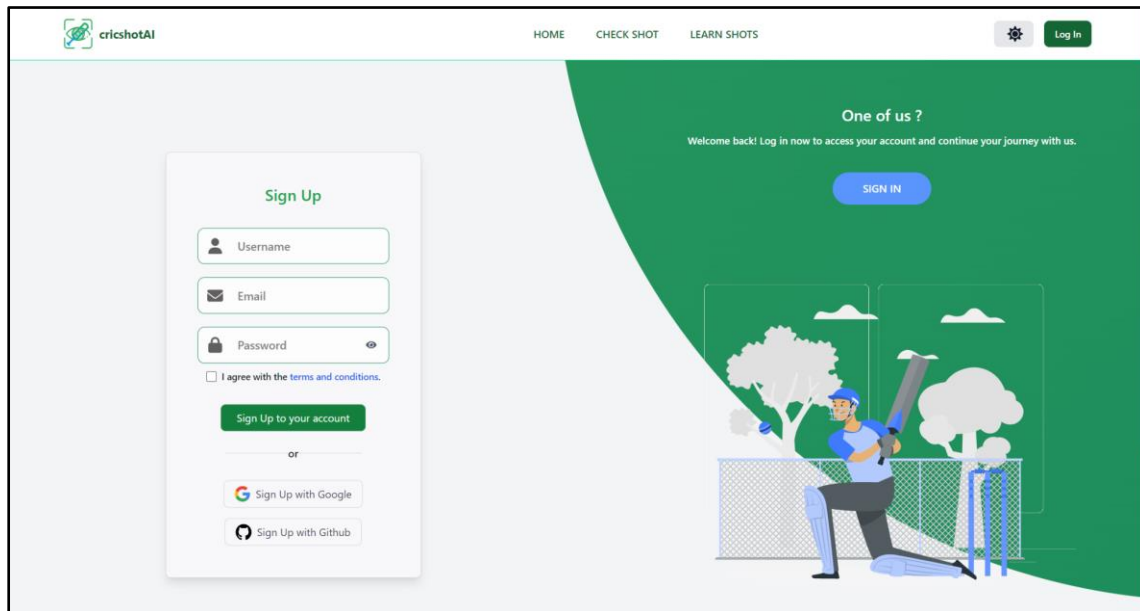
- **Educational Content:** Clear and informative content introducing various cricket shots. Descriptions provide insights into shot techniques.
- **Animation Integration:** Overall page animations with AOS library. Provides a cohesive and visually pleasing experience.
- **Clear Card Structure:** Each shot card has a well-defined structure. Title, text description, and GIF are presented in a readable format.

Sign In and Sign Up Page

Sign up Component :

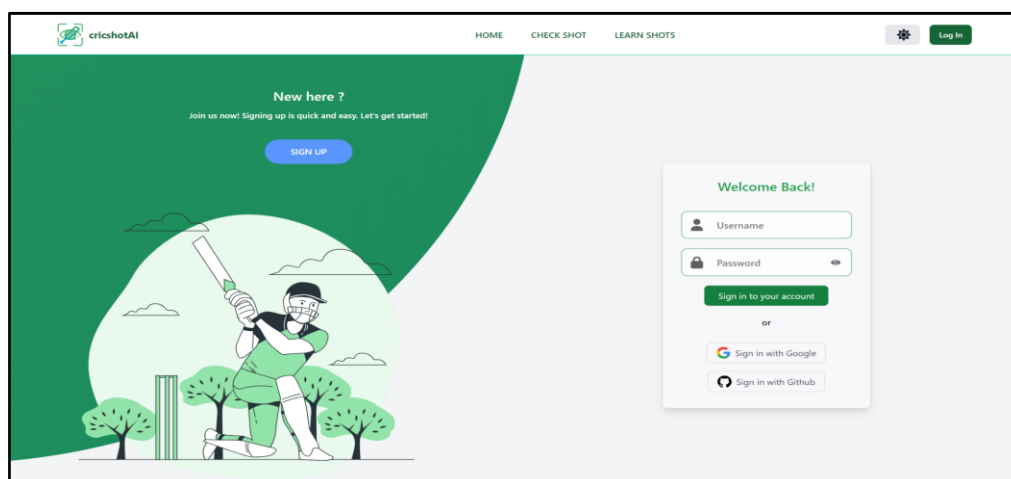
- Input validation guarantees the accuracy and security of user-provided data during registration.
- Password strength requirements enhance account security by enforcing complex password criteria.

- Error handling provides informative feedback on registration failures, guiding users through the process.
- Integration with social media platforms enables users to register using their existing accounts.



Sign in Component :

- Input validation ensures the integrity and security of user credentials.
- Password visibility toggle enhances user convenience while maintaining security.
- Seamless integration with social media platforms like Google and GitHub for alternative authentication methods.
- Customizable error handling provides clear feedback on authentication issues.
- Responsive design ensures compatibility across various devices and screen sizes.



Sign in & up Component :

- Presents clear options for users to either sign in or sign up for an account, enhancing usability.
- Integrates both signin and signup functionalities into a single, visually appealing interface.
- Seamless integration with social media platforms offers alternative authentication methods.
- Responsive layout ensures optimal user experience across different devices and screen sizes.

User's Dashboard

The Dashboard serves as a central control panel for users to manage activities, settings, and access assistance. A persistent Sidebar allows seamless navigation, while the Profile page offers insights into performance metrics. The Activity page provides detailed reports on historical engagements, and the Settings page enables account customization. Lastly, the Help page offers guidance and support resources.

Sidebar Component :

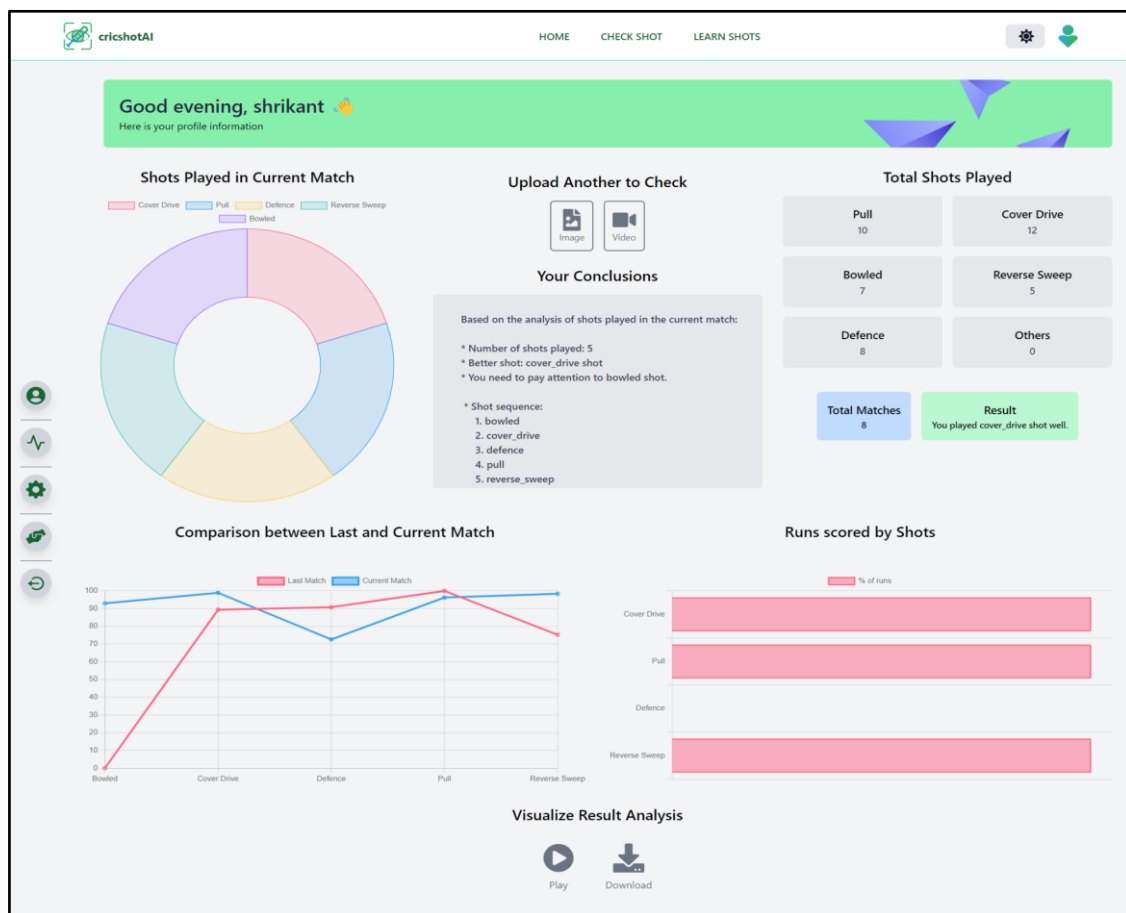
The Sidebar component plays a crucial role in facilitating navigation within the application by providing users with quick access to various sections and features.

Key Features :

- **Functionality:** Facilitates toggling sidebar visibility and handles logout functionality upon user interaction.
- **Icons and Links:** Offers easy access to profile, activity, settings, help, and logout pages through corresponding icons and text links.
- **Responsive Design:** Adapts to varying screen sizes.
- **Visual Enhancements:** Hover effects and tooltips for better interaction.

Profile Page

The Profile page is a central hub in the dashboard, offering users a comprehensive view of performance metrics, shot analysis, and match statistics. It helps users track progress, identify improvement areas, and make informed decisions to enhance their sporting performance.



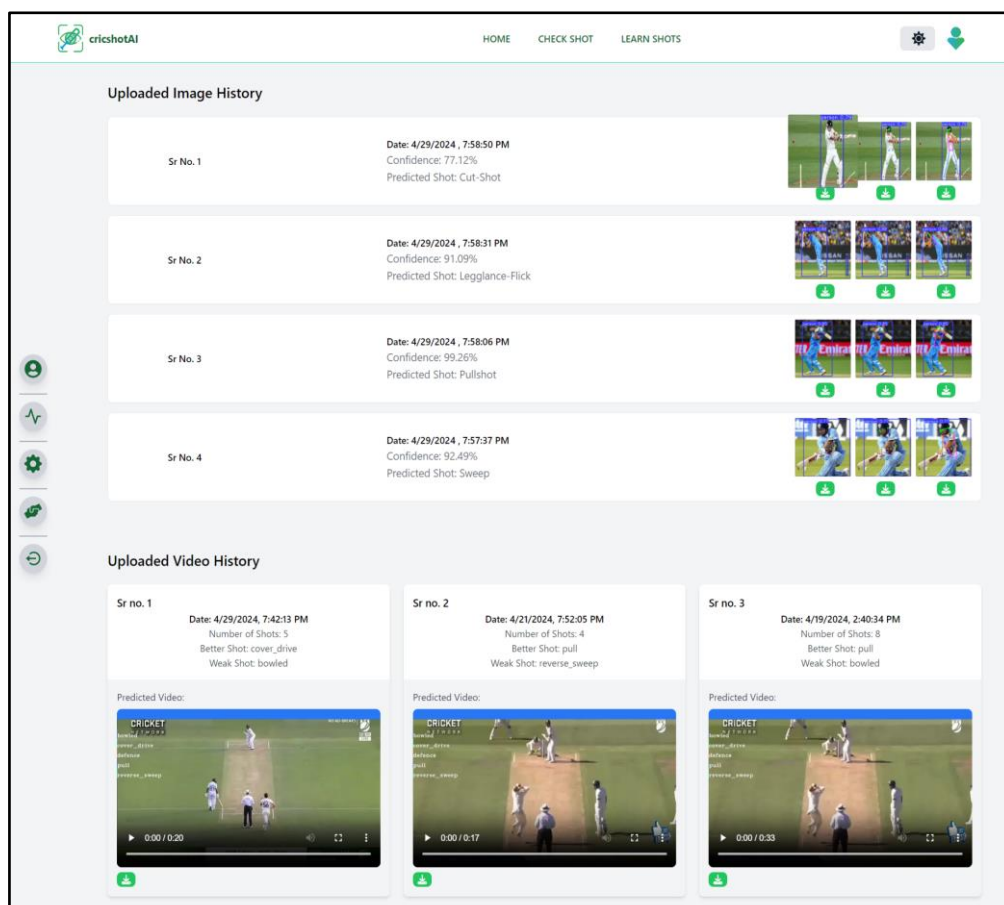
Page Overview :

- **User Data Retrieval :** Several useEffect hooks are employed to fetch user-specific data such as total matches played, shots played (e.g., pull, cover drive, defence), and video analysis data from the backend using service functions.
- **Shot Analysis :** The Profile page displays visualizations of shot analysis data, including doughnut charts (DoughC) representing shots played in the current match, and vertical bar charts (VerticalC) illustrating runs scored by shots.

- **Sidebar Navigation :** The SideBar component provides users with navigation options to access different sections of the dashboard, enhancing user experience and ease of navigation.
- **Welcome Message :** The Welcome component dynamically displays a personalized welcome message, incorporating the user's name and current time, to create a welcoming atmosphere for users.
- **Conclusion Display :** The Profile page presents users with textual conclusions based on shot analysis, displaying observations, better shot recommendations, and shot sequences extracted from video analysis data.
- **Video Playback and Download :** Users have the option to play and download predicted video clips,

Activity Page

The Activity page stands as a foundational element of the application, offering users a comprehensive overview of their historical image and video activities.



1. Image Activity Component :

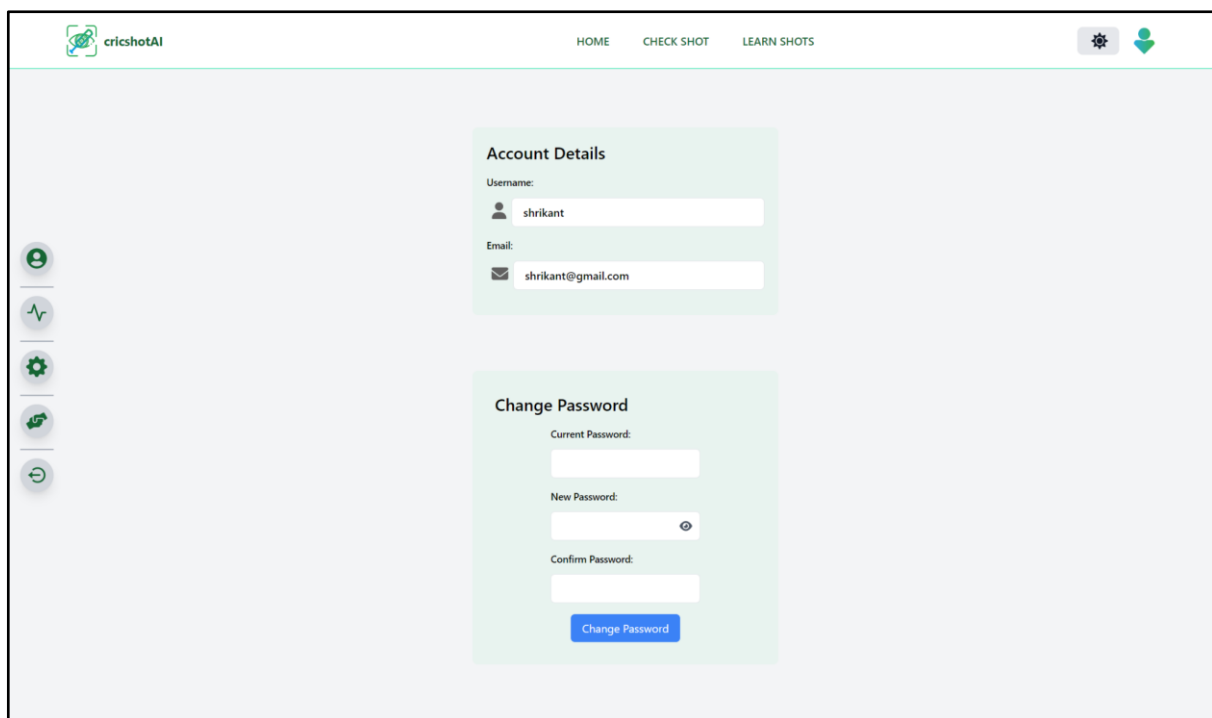
- Central to the Activity page is the ActivityImg component, responsible for presenting users' historical image activities.
- Uses React hooks (useState, useEffect) to retrieve and format data, showing image search history with metadata (date, confidence level, predicted shot type).

2. Video Activity Component :

- The ActivityVid component handles video activities.
- Uses React hooks and backend calls to retrieve and display users' last four video matches with metadata (date, number of shots, best shot, weak shot) and embedded video previews.

Setting Page

The Setting page is a key part of the application, allowing users to manage account details and enhance security.



The screenshot displays the 'Setting Page' of the 'cricshotAI' application. The page features a light blue background with a white sidebar on the left containing five circular icons: a user profile, a heart rate monitor, a gear (settings), a document, and a refresh symbol. The main content area is divided into two sections. The top section, titled 'Account Details', contains two input fields: 'Username' with the value 'shrikant' and 'Email' with the value 'shrikant@gmail.com'. The bottom section, titled 'Change Password', contains three input fields: 'Current Password', 'New Password' (with an eye icon for toggling visibility), and 'Confirm Password'. A blue button labeled 'Change Password' is positioned at the bottom of this section. The top navigation bar includes the 'cricshotAI' logo, the text 'HOME CHECK SHOT LEARN SHOTS', and two icons: a gear and a green download arrow.

1. Account Details Form :

- a. Displays user information like username and email using React functional components.

2. Change Password Form :

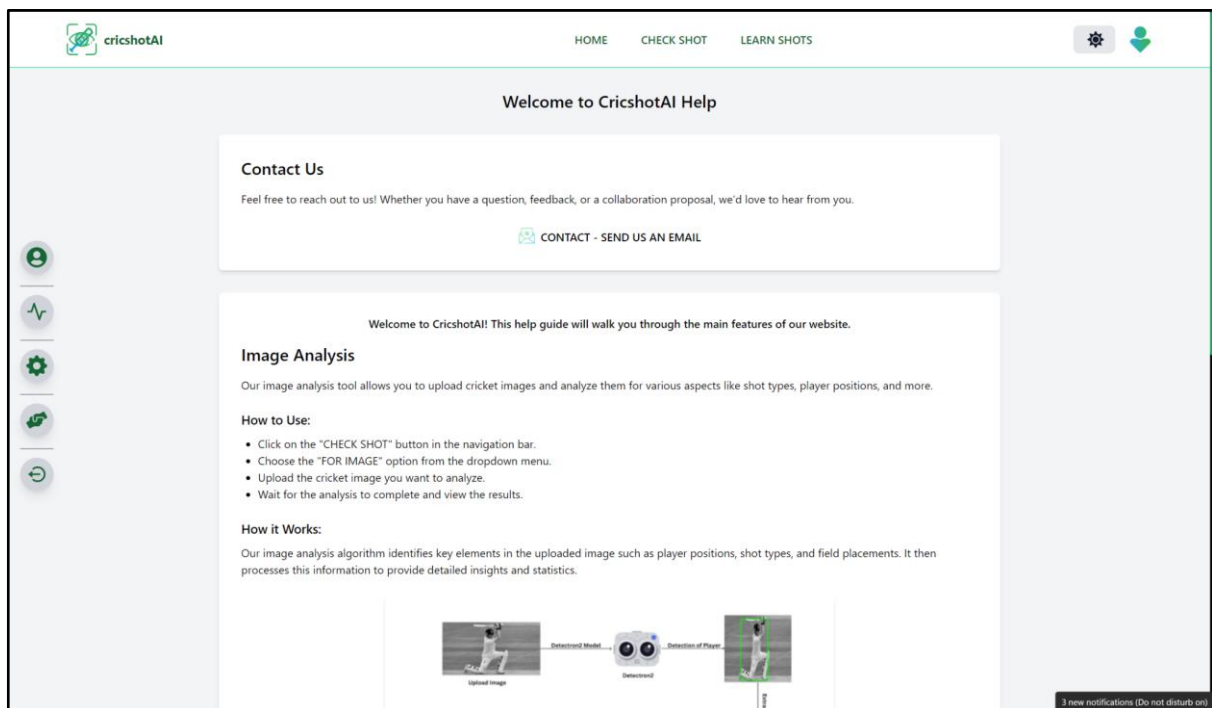
- a. Enables secure password changes with state management using useState hooks.
- b. Features a dynamic toggle for password visibility to enhance user convenience.

3. User Authentication and Password Change:

- a. Facilitates secure access to account details using the getCurrentUserDetail function.
- b. Handles password updates via the handleChangePassword function, ensuring secure backend service calls and providing user feedback and error handling.

Help Page

The Help page provides essential guidance for using the platform effectively. Key sections include



1. Contact Us Section:

- Direct communication with administrators via email for inquiries and feedback.

2. Image Analysis Guide:

- Step-by-step instructions for uploading cricket images and interpreting analysis results.

3. Video Analysis Guide:

- Detailed steps for uploading cricket videos and understanding analysis insights..

4. How It Works Illustrations:

- Visual aids in the analysis sections to explain how the algorithms work, enhancing user comprehension and engagement.

6.3 Backend

Section 1: FastAPI Implementation

Code 1 : app.py

Purpose: This snippet initializes a FastAPI application and defines routes for handling requests for Deep Learning Models related prediction.

Explanation:

- Imports necessary FastAPI modules and dependencies.
- Configures CORS settings to allow specified origins.
- Loads pre-trained Deep learning models and scaler for shot recognition.
- Defines routes for the home page and predictions.
- Reads uploaded image or video, processes them, and makes predictions using the model.

```

# Prediction route (For Image)
@app.post("/predict")
async def predict(file: UploadFile):
    # Read and process the uploaded image using OpenCV
    content = await file.read()
    nparr = np.frombuffer(content, np.uint8)
    img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

    img = cv2.resize(img, desired_resolution,
                     interpolation=cv2.INTER_LINEAR)

    # call prediction function
    predicted_shot, confidence, res_img_1, res_img_2,
    res_img_3 = prediction_on_image(img, i_model, loaded_scaler)

    if predicted_shot == None:
        return #Error for invalid image

    # Create JSON response with individual paths
    response_json = {
        "predicted_shot": predicted_shot,
        "confidence": f'{confidence:.2f}%',
        "result_image_1": res_img_1,
        "result_image_2": res_img_2,
        "result_image_3": res_img_3,
    }

    return response_json

```

Code 2 : i_pred.py

Purpose: Contains functions for keypoint extraction and image manipulation using YOLOv8 and ANN.

Explanation:

- Imports YOLOv8 for the keypoint and ANN for prediction model.
- Defines function to make prediction on image and create annotated images with bounding boxes and keypoints.
- Utilizes the function to process images and extract necessary information for prediction.

Code 3 : v_pred.py

Purpose: Contains function for video manipulation using CNN and LSTM network.

Explanation:

- Imports Time-Distributed CNN and LSTM for video classification.
- Defines function to make predictions on video and create analysed video.
- Utilizes the function to process video and extract necessary information for prediction.

Section 2: Data Augmentation

Code 4 : data_augmentation.ipynb

Purpose: Augments the dataset by flipping and rotating images.

Explanation:

- Imports required libraries for image augmentation.
- Loads unaugmented images from a specified directory.
- Applies flipping and rotation to images, creating augmented copies.

```
# Initialize lists to store augmented images and labels
aug_images = []
aug_labels = []

# Loop through each image in the data
for idx in range(len(images)):
    # Fetch an image and its corresponding label
    img = images[idx]
    lab = labels[idx]

    # Flip the image horizontally (1 indicates horizontal flip)
    img_flip = cv2.flip(img, 1)

    # Append the augmented image and label to the list
    aug_images.append(img_flip)
    aug_labels.append(lab)
```

- Displays sample images before and after augmentation.
- Saves augmented images in a new directory for further use.

Section 3: Deep Learning Models Development

Image Model

Code 5 : i_model.ipynb

Purpose: Develops and trains an Artificial Neural Network (ANN) model using TensorFlow.

Explanation:

- Imports necessary TensorFlow libraries and sets up the ANN model architecture.

- Setup YOLOv8 and Extract keypoints on image.

```
# function that extracts the keypoints for an image
def extract_keypoints(img):

    # make predictions
    outputs = keypoint_model.predict(img, conf= 0.75, imgsz=(288,288),
    keypoints = outputs[0].keypoints.xy.cpu().numpy())

    if(len(keypoints)>0):
        # fetch keypoints of a person with maximum confidence score
        kp = keypoints[0]
        # convert 2D array to 1D array
        kp = kp.flatten()
        # return keypoints
        return kp
```

- Normalize keypoints.
- Splits the dataset into training, validation and Test sets.
- Defines architecture and optimizer, loss function.

```
# defining the model architecture
model = Sequential([
    Dense(128, input_shape=(34,), activation='relu'),
    Dense(256, activation='relu'),
    Dense(128),
    LeakyReLU(0.01),
    Dense(5, activation='softmax')
])
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Model Summary
model.summary()
```

- Trains the model, iteratively updating weights and monitoring losses.

```
# Model Training

start_time = time.time()

history = model.fit(x_train,y_train, epochs=25, batch_size=32, validation_data=(x_val,y_val),
                    callbacks = tf.keras.callbacks.EarlyStopping(patience = 3, monitor = 'val_loss',
                    restore_best_weights=True))

end_time = time.time()

# Calculate the training time
time_taken = end_time - start_time
tf.print(f"Time taken for video: {time_taken:.4f} seconds")
```

- Create a function for Prediction on Some Images

```

# create subplots with 4 rows and 4 columns
fig, ax = plt.subplots(nrows=4, ncols=4, figsize=(15,15))

# randomly sample indices
ind = random.sample(range(len(images)), 16)

for row in range(4):
    for col in range(4):
        img_idx = 4 * row + col # Calculate the index for the ind list
        img = cv2.cvtColor(images[ ind[img_idx] ], cv2.COLOR_BGR2RGB)
        actual_label = labels[ ind[img_idx] ]
        predicted_label = prediction(img)
        # display image
        ax[row, col].imshow(img)
        # set title
        ax[row, col].set_title(f"Actual: {actual_label}\nPredicted: {predicted_label}")
        # Turn off axis
        ax[row, col].axis('off')

```

Video Model

Code 6 : v_model.ipynb

Purpose: Develops and trains a Time-Distributed Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) model using TensorFlow.

Explanation:

- Imports necessary TensorFlow libraries and sets up the CNN and LSTM model architecture.
- Function to Extract Frames from video.
- Load video Dataset from disk by creating own input Pipeline.

```

# To create own input pipeline for video data
class FrameGenerator:
    def __init__(self, path, n_frames, shuffle = False):

        self.path = Path(path) # Convert the path to a Path object
        self.n_frames = n_frames
        self.shuffle = shuffle
        self.class_names = sorted(set(p.name for p in self.path.iterdir() if p.is_dir()))
        self.class_ids_for_name = dict((name, idx) for idx, name in enumerate(self.class_names))

    def get_files_and_class_names(self):
        video_paths = list(self.path.glob('*/*.mp4'))
        classes = [p.parent.name for p in video_paths]
        return video_paths, classes

    def __call__(self):
        video_paths, classes = self.get_files_and_class_names()

        pairs = list(zip(video_paths, classes))
        if self.shuffle:
            np.random.seed(42)
            np.random.shuffle(pairs)

        for video_path, name in pairs:
            video_frames = frames_from_video(video_path, self.n_frames, interval=5)
            label = self.class_ids_for_name[name] # Encode labels
            label = tf.keras.utils.to_categorical(label, num_classes=len(self.class_names)) # One
            yield video_frames, label

```

```
# Decide output shape
output_signature = (tf.TensorSpec(shape = (18, 199, 199, 3), dtype = tf.float32), #video frames
                    tf.TensorSpec(shape = (5), dtype = tf.int16))                # labels

# Create Dataset
dataset = tf.data.Dataset.from_generator(FrameGenerator(path, n_frames=18, shuffle=True), output_signature
```

- Batch dataset and speed up loading.
- Splits the dataset into training, validation and Test sets.
- Similarly, defines architecture and optimizer, loss function.
- Trains the model, iteratively updating weights and monitoring losses.

Section 4 : Spring Boot Implementation

Authentication and Authorization

Authentication and authorization are crucial components of modern web applications, ensuring that only authenticated users gain access to authorized resources.

Authentication verifies the identity of users, typically through credentials like usernames and passwords. Once authenticated, users are granted access to the application's functionalities.

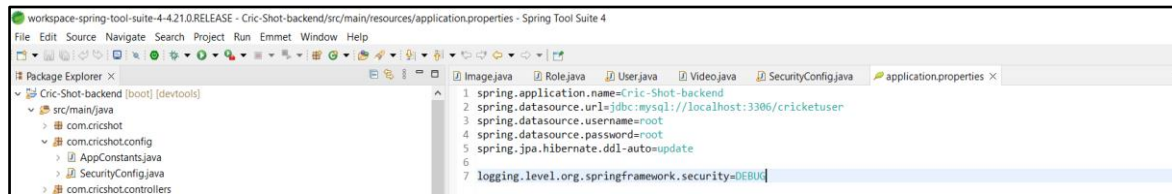
Authorization determines the level of access granted to authenticated users based on their roles, permissions, or other attributes. It ensures that users can only perform actions or access resources that they are explicitly allowed to.

MySQL Integration

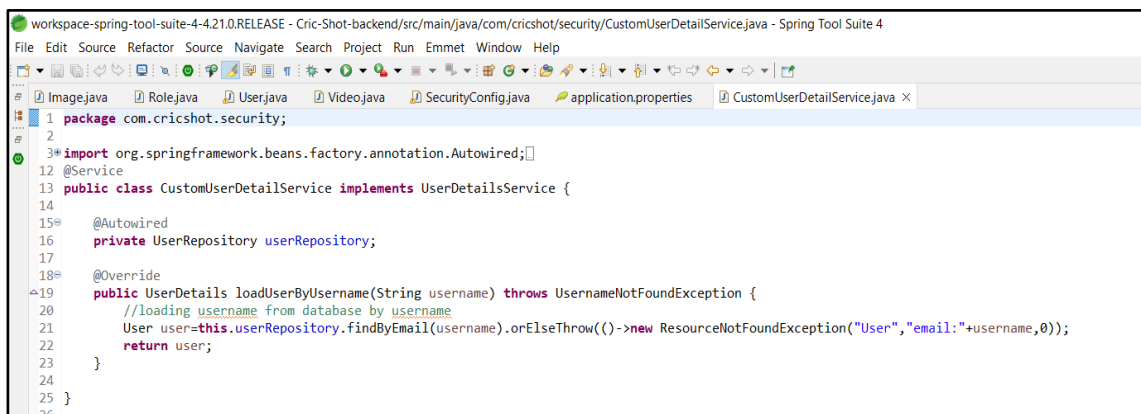
Integrating MySQL with Spring Security for authentication and authorization involves configuring the application to use MySQL as the data source for storing user credentials, roles, and permissions. Here's an overview of how to integrate MySQL with Spring Security :

- **Database Setup** : Create tables in your MySQL database to store user information, roles, and permissions.

- **Configure Data Source :** Configure the data source in your Spring Boot application to connect to your MySQL database. This involves specifying the database URL, username, password, and any other necessary properties in the application.properties file.



- **Define User and Role Entities :** Define JPA entity classes for users and roles, mapping them to the corresponding database tables. Make sure to include necessary attributes such as username, password (hashed), roles, etc. Annotate these classes with @Entity, and specify the table name using @Table.
- **Implement UserDetailsService :** Interface to load user details from the database during authentication. This involves retrieving user information (username, password, roles) from the database based on the username provided during login.



- **Configure Authentication Manager :** Configure the AuthenticationManagerBuilder in your Spring Security configuration class to use your custom UserDetailsService for authentication. This involves specifying the UserDetailsService implementation and configuring password encoding (e.g., BCryptPasswordEncoder) for secure password storage and validation.
- **Secure Application Endpoints :** Use HTTP security configurations to specify which endpoints require authentication and authorization, and define access rules based on user roles and permissions.

- **Test Authentication and Authorization :** Test your authentication and authorization setup by logging in with valid user credentials and accessing secured endpoints. Verify that users are authenticated successfully and granted access based on their roles and permissions.
- **Handle Access Denied and Authentication Failure :** These handlers use to handle cases where users are denied access to secured resources or authentication fails. These handlers can return appropriate error messages or redirect users to the login page.

Entity Classes

Image :

- The Image class represents images stored in the system.
- It contains attributes such as imageAddedDate, confidence, predicted_shot, and result_image_1, result_image_2, result_image_3 for storing image data.
- Each image is associated with a User entity through a Many-to-One relationship, indicating which user uploaded the image.

```
package com.cricshot.entities;

import java.util.Date;

@Entity
@Table(name="image")
public class Image {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private Date imageAddedDate;
    private String confidence;
    private String predicted_shot;
    @Lob
    @Column(columnDefinition = "LONGBLOB")
    private byte[] result_image_1;
    @Lob
    @Column(columnDefinition = "LONGBLOB")
    private byte[] result_image_2;
    @Lob
    @Column(columnDefinition = "LONGBLOB")
    private byte[] result_image_3;
    @ManyToOne
    @JoinColumn(name = "image_user_id")
    private User user;
```

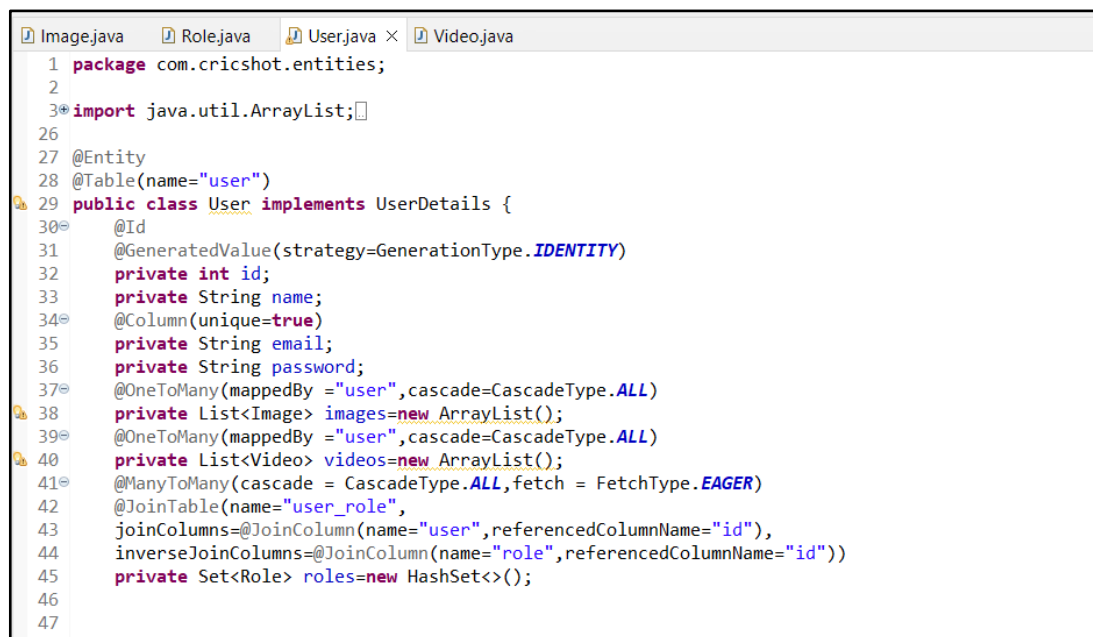
Video :

- The Video class represents videos stored in the system.
- It includes attributes like videoAddedDate, shots_played, and predicted_video for storing video-related data.

- Similar to Image, each video is associated with a User entity through a Many-to-One relationship.

User :

- The User class represents users of the system.
- It includes attributes such as name, email, and password for storing user information.
- Users can have multiple images and videos associated with them, forming One-to-Many relationships.
- Additionally, users can have multiple roles associated with them through a Many-to-Many relationship.



```

1 package com.cricshot.entities;
2
3 import java.util.ArrayList;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27 @Entity
28 @Table(name="user")
29 public class User implements UserDetails {
30     @Id
31     @GeneratedValue(strategy=GenerationType.IDENTITY)
32     private int id;
33     private String name;
34     @Column(unique=true)
35     private String email;
36     private String password;
37     @OneToMany(mappedBy = "user", cascade=CascadeType.ALL)
38     private List<Image> images=new ArrayList();
39     @OneToMany(mappedBy = "user", cascade=CascadeType.ALL)
40     private List<Video> videos=new ArrayList();
41     @ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
42     @JoinTable(name="user_role",
43         joinColumns=@JoinColumn(name="user", referencedColumnName="id"),
44         inverseJoinColumns=@JoinColumn(name="role", referencedColumnName="id"))
45     private Set<Role> roles=new HashSet<>();
46
47
48

```

6.4 Algorithm

The algorithmic approach adopted in the project includes:

YOLOv8 (You Only Look Once)

In addition to object detection, YOLO can also be adapted for pose estimation tasks. YOLO Pose Estimation involves detecting key body points, such as joints or keypoints, in an image or video frame. This can be useful in applications like human pose recognition, action recognition, and gesture recognition.

The process of YOLO keypoint extraction in this project can be broken down into several steps:

1. Imports the YOLO class from the Ultralytics library.
2. Loads a pre trained YOLOv8 model for keypoint detection (yolov8n-pose.pt).
3. Defines a function extract_keypoints that takes an image (img) as input.
4. Inside the function, it makes predictions using the loaded YOLOv8 model on the input image with certain parameters like confidence threshold (conf=0.75), image size (imgsz=(288,288)), and verbose mode off (verbose=False).
5. Focusing on the keypoints of a person with the maximum confidence score.
6. Flattens the 2D array of keypoints into a 1D array and returns the keypoints.

ANN (Artificial Neural Networks)

ANN is used for image classification tasks, aiding in identifying cricket shot types based on extracted features.

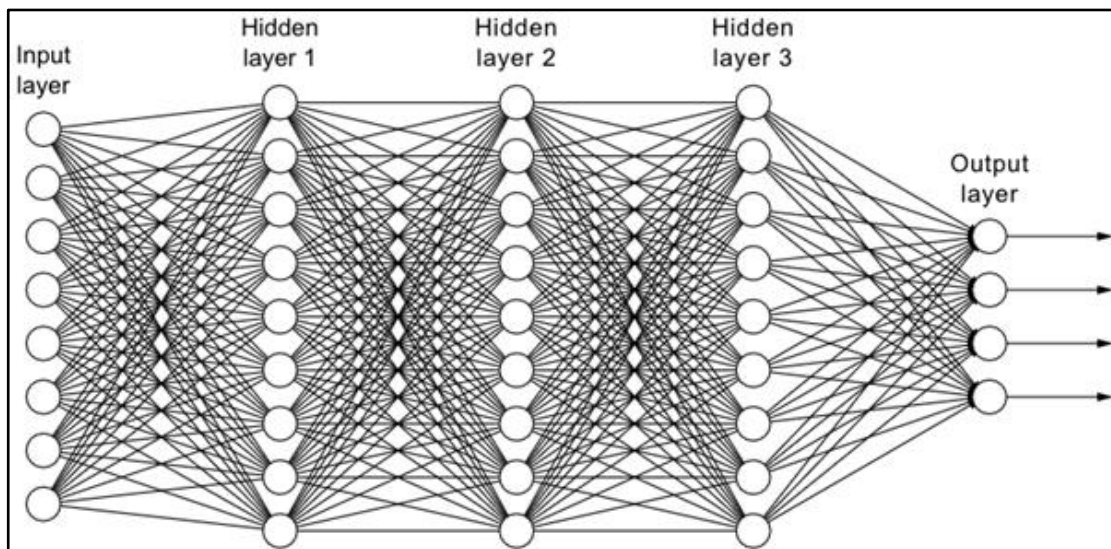


Figure 8 : Artificial Neural Network(ANN)

Here are the training steps in this project for Artificial Neural Networks (ANN):

1. Initialize a Tensorflow Sequential model with layers defined as follows:
 - a. Input layer with 34 input units.
 - b. Two Hidden layers with 128 and 256 units using ReLU activation.
 - c. Another hidden layer with 128 units and a LeakyReLU activation (leaky ReLU with alpha = 0.01).
 - d. Output layer with 5 units using softmax activation for multiclass classification.

2. Compile the model using the Adam optimizer, categorical cross-entropy loss function, and accuracy metric.
3. Fit the model to the training data for 25 epochs with a batch size of 32.
4. Utilize the validation data for validation during training.
5. Apply early stopping with a patience of 3 epochs and monitor validation loss to restore the best weights.

Time-Distributed CNN and LSTM

An intuitive deep model is designed with a combination of Time-Distributed 2D CNN layers and LSTM cells for extracting and learning the spatiotemporal information from the input sequences

The proposed model architecture consists of various layers including Time distributed 2D-CNN, Max pooling (MP), and fully connected layers. The model accepts a sequence of frames and is analyzed by the CNN network that extracts spatial features from the input frames. The LSTM layers are utilized to capture the temporal features and dependencies between the frames.

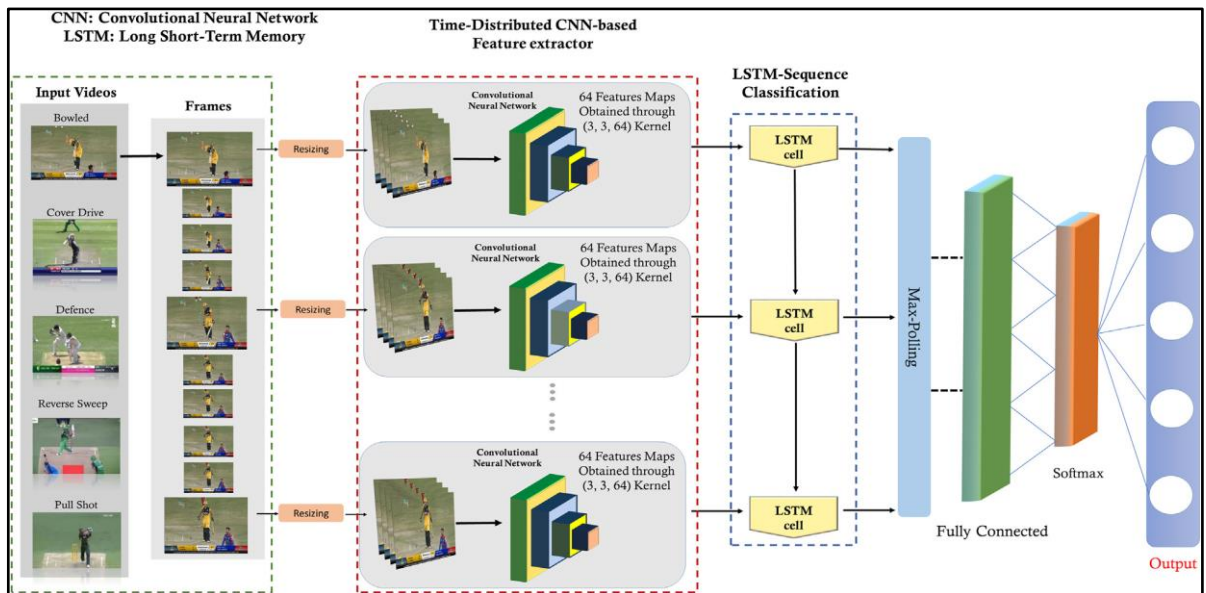


Figure 9 : An overall structure of the proposed model.

The model includes multiple convolutional layers. An activation function is used in each convolutional layer to introduce non-linearity to the output of each layer. The outputs of the last max-pooling layer in the CNN are flattened and fed to the LSTM layers, which capture the temporal dependencies in the sequence of frames.

Here are the training steps in this project for Time-Distributed CNN and LSTM network :

1. Model Architecture Setup: Define the `cnn_lstm_model(input_shape, num_classes)` function:
 - a. Input is of shape (e.g., (18, 199, 199, 3)) (18 frames, 199x199 pixels, RGB channels) and number of classes (e.g, 5) representing different cricket shots.
 - b. Input layer with the specified input shape.
 - c. Four TimeDistributed convolutional layers with increasing filters (16, 32, 64, 128) and apply kernel size (3, 3), strides (2, 2), padding='same' and ReLU activation in each layer.
 - d. After each convolutional layer, place a Time-Distributed MaxPooling layer with pool size (4, 4) for layers 1 and 2, (2, 2) for layers 3 and 4.
 - e. TimeDistributed Flatten layer to prepare the data for LSTM.
 - f. LSTM layer with 128 units and `return_sequences=False`.
 - g. Dense (fully connected) layer with 128 units and ReLU activation.
 - h. Output layer with 5 output units and softmax activation for multi-class classification.
2. Create the model using `cnn_lstm_model(input_shape=(18, 199, 199, 3), num_classes=5)`.
3. Compile the model using the Adam optimizer, categorical cross-entropy loss function, and accuracy metric.
4. Fit the model to the training data (`train_ds`) for 30 epochs with a batch size determined by the data.
5. Use the validation data (`val_ds`) for validation during training.
6. Apply early stopping with a patience of 3 epochs and monitor validation loss to restore the best weights.

6.5 Packages/Libraries Used

Key packages and libraries utilized in the implementation phase are:

- **TensorFlow** : Utilized for building and training deep learning models, including CNNs and LSTM networks, for image and video analysis.
- **OpenCV** : Used for image and video processing tasks, including keypoint detection with YOLO, object localization, and feature extraction.
- **FastAPI** : Chosen for creating RESTful APIs to handle user requests, serve model predictions, and interact with the backend services.
- **Spring Boot** : Implemented for backend logic, data processing, and integration with the MySQL database.
- **MySQL Connector** : Used for interacting with the MySQL database to store user profiles, uploaded data, model checkpoints, and analysis results.
- **ReactJS** : Employed for developing interactive and responsive frontend interfaces, including the user upload interface and result display.
- **Tailwind CSS** : Used for styling and designing frontend components to ensure a visually appealing and consistent user experience.
- **Scikit-learn** : Utilized for machine learning tasks, such as data preprocessing, feature engineering, and model evaluation.
- **Keras** : Integrated for building and training deep learning models, including ANN architectures for shot classification.
- **Numpy** : Utilized for numerical computations and array manipulations, essential for data processing and manipulation in deep learning workflows.
- **Ultralytics** : Integrated for enhanced object detection capabilities and model evaluation tools.
- **Matplotlib** : Used for data visualization, generating plots, and visualizing model outputs during analysis and debugging.
- **ffmpeg** : Integrated for handling audio and video processing tasks, such as video encoding and decoding.
- **Seaborn** : Utilized for statistical data visualization and enhancing the aesthetics of plots and charts.

6.6 Testing

Frontend Testing

Purpose: Manually evaluate the website's frontend by simulating user interactions and image/video uploads.

Explanation: Conduct manual testing to assess UI, interaction flow, and system responses in various scenarios.

Process: Testers navigate the website, perform interactions, and observe responses to identify any issues.

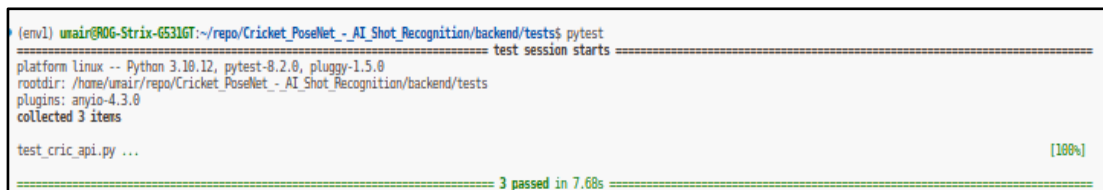
Backend API Testing (Pytest)

Purpose: Contains test cases to validate the functionality of the backend API endpoints.

Explanation:

- Imports necessary testing libraries and sets up fixtures for testing.
- Defines test cases for each API endpoint, including input validation, expected responses, and error handling(In test_cric_api.py file).
- Utilizes pytest framework to execute test cases and assert the expected behavior of the API endpoints .

Result of testing : run “pytest” command in terminal.



```
(env1) umair@06-Strix-6531G:~/repo/Cricket_PoseNet_-_AI_Shot_Recognition/backend/tests$ pytest
===== test session starts =====
platform linux -- Python 3.10.12, pytest-8.2.0, pluggy-1.5.0
rootdir: /home/umair/repo/Cricket_PoseNet_-_AI_Shot_Recognition/backend/tests
plugins: anyio-4.3.0
collected 3 items

test_cric_api.py ... [100%]

===== 3 passed in 7.68s =====
```

Test Cases

The test cases given below cover a range of scenarios and functionalities, ensuring thorough validation of the "Cricket PosNet: AI Shot Assistant" system's functionality, performance, and user experience.

- **Image Classification Test Cases :**
 - **Input :** An image containing a cricket shot.

- **Expected Outcome :** System accurately identifies and classifies the cricket shot type (e.g., cover drive, pull shot) with high confidence.
- **Video Analysis Test Cases :**
 - **Input :** A video containing cricket shots.
 - **Expected Outcome :** System detects and classifies shot transitions between different shot types (e.g., transition from drive to cut shot) accurately.
- **User Interaction Test Cases :**
 - **Scenario 1: Upload Interface:**

Input : User uploads an image or video through the frontend interface.

Expected Outcome : System processes the uploaded content promptly and generates accurate shot classification results.
 - **Scenario 2: Error Handling:**

Input : User provides invalid or corrupt input data.

Expected Outcome : System detects and handles errors gracefully, providing informative feedback to the user.
- **Integration Test Cases :**
 - **Scenario 1: API Endpoint Testing:**

Input : API requests sent to backend services for shot classification.

Expected Outcome : API endpoints respond correctly, serving predictions and results in the expected format.
 - **Scenario 2: Database Integration:**

Input : Data retrieval and storage operations in the MySQL database.

Expected Outcome : System seamlessly integrates with the database, ensuring data consistency and reliability.

Chapter 7

Result and Analysis

7.1 Accuracy

- **Training Accuracy**

The training accuracy measures the model's performance concerning the training dataset, showcasing how well it learned from the provided data. The achieved training accuracy for the image model was 95.24%, and for the video model, it was 98.42%, indicating their ability to effectively learn patterns and features within their respective training datasets.

- **Testing Accuracy**

The testing accuracy evaluates the model's performance on a completely unseen dataset. The testing accuracy obtained for the image model was 89.01%, and for the video model, it was 94.44%, signifying their ability to generalize well to unseen data, demonstrating robust performance beyond the training set.

7.2 Classification Report

The classification report for the data provides a breakdown of precision, recall, and F1-score for each class based on the model's predictions on the dataset.

- **Precision** : Precision measures the accuracy of positive predictions made by a model. precision answers the question, "Out of all the instances predicted as positive, how many are actually positive?"
- **Recall** : Recall measures the ability of a model to capture all positive instances in the dataset. In essence, recall answers the question, "Out of all the actual positive instances, how many did the model correctly identify as positive?"

- **F1 Score** : It provides a balance between precision and recall and is particularly useful when dealing with imbalanced datasets, where one class dominates the other. The F1 score ranges from 0 to 1, where higher values indicate better model performance.

Classification Report for image model

Training set Classification report:				Test set Classification report:			
	precision	recall	f1-score		precision	recall	f1-score
cut-shot	0.98	0.92	0.95	cut-shot	0.99	0.83	0.90
drive	0.91	0.95	0.93	drive	0.80	0.89	0.84
legglance-flick	0.93	0.97	0.95	legglance-flick	0.86	0.95	0.90
pullshot	0.96	0.94	0.95	pullshot	0.92	0.90	0.91
sweep	0.99	0.99	0.99	sweep	0.91	0.89	0.90

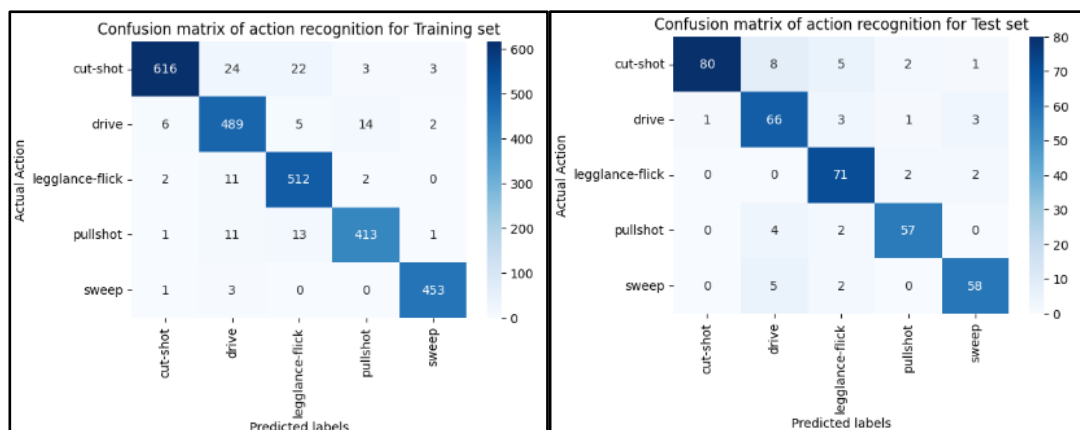
Classification Report for video model

Training set Classification report:				Test set Classification report:			
	precision	recall	f1-score		precision	recall	f1-score
Bowled	0.98	1.00	0.99	Bowled	0.89	1.00	0.94
Cover Drive	0.97	0.97	0.97	Cover Drive	1.00	0.87	0.93
Defence	1.00	0.99	1.00	Defence	1.00	1.00	1.00
Pull Shot	1.00	0.99	1.00	Pull Shot	1.00	1.00	1.00
Reverse Sweep	0.96	0.96	0.96	Reverse Sweep	0.83	0.83	0.83

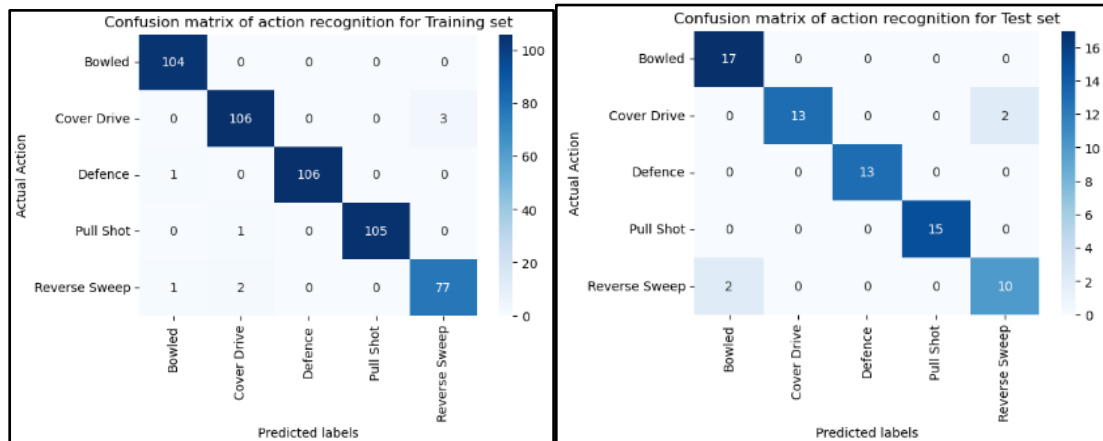
7.3 Confusion Matrix

The confusion matrix for the data visually represents the model's performance in terms of correctly and incorrectly classified instances for each class within the dataset.

Confusion Matrix for image model



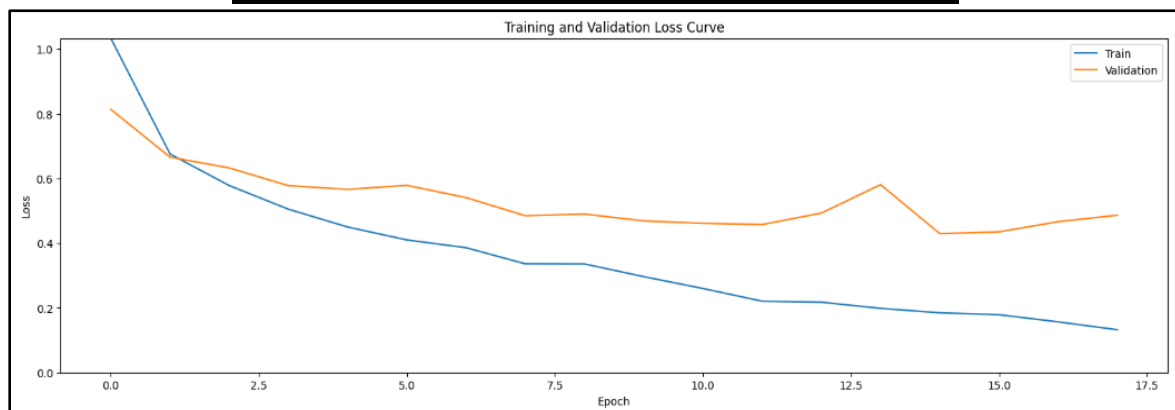
Confusion Matrix for video model



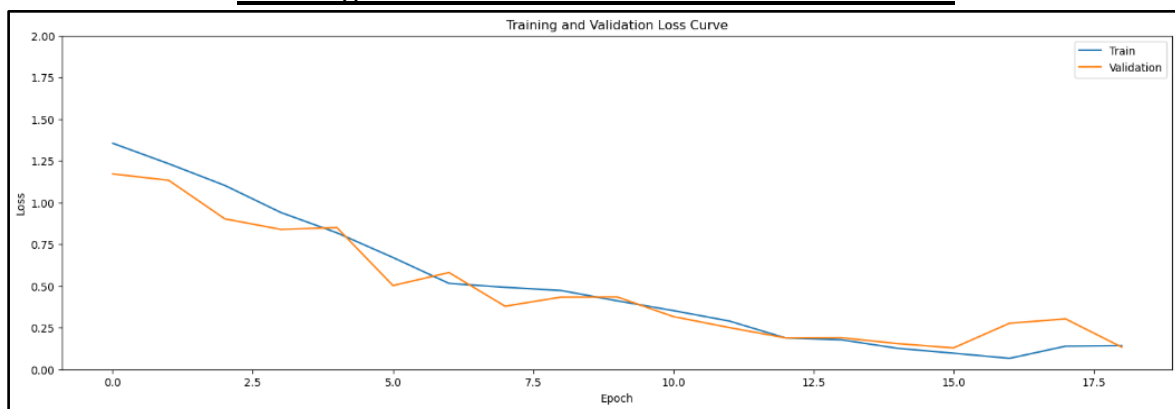
7.4 Training and Validation Loss Curves

The training and validation loss curves visualize the progression of loss values during the training process, indicating how well the model is learning from the data and whether it's overfitting or underfitting.

Training and Validation Loss Curve for image model



Training and Validation Loss Curve for video model



7.5 Analysis

The image and video models achieved high training accuracies, showcasing their capability to learn intricate patterns within their training datasets. The testing accuracies further validate the models' generalization capabilities, demonstrating their robustness beyond the training sets.

The classification reports and confusion matrices provide detailed insights into the models' performance across different datasets, while the training and validation loss curves visualize their learning progress and help assess potential overfitting or underfitting.

Overall, these evaluations indicate well-performing models for image and video shot analysis, demonstrating high accuracy and reliable performance across multiple datasets.

CONCLUSION

In this project, we delved into advanced computer vision and deep learning techniques to develop an AI-powered Cricket Shot Assistant. We began by curating a cricket shot image dataset comprising five categories: Bowled, cover drive, defence, pull shot, and reverse sweep. This dataset underwent preprocessing, including cropping and resizing, to prepare it for accurate classification. Our approach involved leveraging YOLO for keypoint extraction and employing an artificial neural network (ANN) for precise shot classification in images.

Additionally, we introduced a shots video dataset with the same five categories, which also underwent preprocessing for categorization. Subsequently, we designed an end-to-end time-distributed CNN-LSTM model. The time-distributed CNN facilitated frame-level feature extraction, while the LSTM layers learned sequence patterns and made predictions in the final stage of analysis.

Our experiments, conducted using both our collected dataset and publicly available datasets, demonstrated promising results in terms of accuracy, recall, precision, and F1 score. These results outperformed many reference models and baseline techniques commonly used for similar tasks.

Looking ahead, we plan to expand our dataset to include additional classes such as fast bowling (both fast and spin bowling), Fielder catches, and spin bowling. To improve real-time performance, we aim to experiment with various attention mechanisms, focusing on significant portions of frames for processing rather than analyzing the entire frame during the later stages of activity recognition. Furthermore, we intend to integrate our proposed method with people counting techniques to intelligently analyze crowd behavior and dense scenarios in a stadium setting.

REFERENCES

- [1] Foysal, Md. Ferdouse & Islam, Mohammad & Karim, Asif & Neehal, Nafis. (2018). ShotNet: A Convolutional Neural Network for Classifying Different Cricket Shots. https://www.researchgate.net/publication/328189966_ShotNet_A_Convolutional_Neural_Network_for_Classifying_Different_Cricket
- [2] Ahmad, Waqas & Munsif, Muhammad & Ullah, Habib & Ullah, Mohib & Alsuwailem, Alhanouf & Saudagar, Abdul & Muhammad, Khan & Sajjad, Muhammad. (2023). Optimized deep learning-based cricket activity focused network and medium scale benchmark. *AEJ - Alexandria Engineering Journal*. 73. 771-779. 10.1016/j.aej.2023.04.062.
- [3] Sen, Anik & Kaushik, Deb & Dhar, Pranab & Koshiba, Takeshi. (2021). CricShotClassify: An Approach to Classifying Batting Shots from Cricket Videos Using a Convolutional Neural Network and Gated Recurrent Unit. *Sensors*. 21. 2846. 10.3390/s21082846.
- [4] <https://detectron2.readthedocs.io/en/latest/tutorials/install.html> Additionally, consider consulting academic journals and research papers for more specific and upto-date information on computer vision, machine learning, and sports analytics in cricket.
- [5] Foysal, Md. Ferdouse & Islam, Mohammad & Karim, Asif & Neehal, Nafis. (2019). Shot-Net: A Convolutional Neural Network for Classifying Different Cricket Shots. 111-120. 10.1007/978-981-13-9181-1_10.
- [6] Russo, Ashraf & Kurnianggoro, Laksono & Jo, Kang-Hyun. (2019). Classification of sports videos with combination of deep learning models and transfer learning. 1-5. 10.1109/ECACE.2019.8679371.
- [7] Russo, Ashraf & Filonenko, Alexander & Jo, Kang-Hyun. (2018). Sports Classification in Sequential Frames Using CNN and RNN. 1-3. 10.1109/ICT-ROBOT.2018.8549884.
- [8] Khan, Zeeshan & Hassan, Muhammad & Farooq, Ammarah & Khan, Muhammad Usman. (2018). Deep CNN Based Data-Driven Recognition of Cricket Batting Shots. 67-71. 10.1109/ICAEM.2018.8536277.
- [9] Khan, Aftab & Nicholson, James & Ploetz, Thomas. (2017). Activity Recognition for Quality Assessment of Batting Shots in Cricket Using a Hierarchical Representation. *PACM Interact. Mob. Wearable Ubiquitous Technol. (IMWUT)*. 1. 62:1-62:31. 10.1145/3130927.

- [9] Semwal, Archit & Mishra, Durgesh & Raj, Vineet & Sharma, Jayanta & Mittal, Ankush. (2018). Cricket Shot Detection from Videos. 1-6. 10.1109/ICCCNT.2018.8494081.
- [10] Chen, Junqiao. (2023). Model Algorithm Research based on Python Fast API. *Frontiers in Science and Engineering*. 3. 7-10. 10.54691/fse.v3i9.5591.
- [11] A. Paszke et al. (2019) introduced PyTorch as a high-performance, imperative-style deep learning library, emphasizing its Pythonic programming, user control, and efficient support for hardware accelerators.
- [12] A.S, HARISH & N, AAGASH & A.S, VISHAL. (2023). CRICKET POSE PREDICTION USING DEEP LEARNING. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*. 07. 1-11. 10.55041/IJSREM26048.
- [13] The PyTorch team. Torch Script. <https://pytorch.org/docs/stable/jit.html>.
- [14] Justin Luitjens. Cuda streams. GPU technology conference, 2014.
- [15] Grossi, Enzo & Buscema, Massimo. (2008). Introduction to artificial neural networks. *European journal of gastroenterology & hepatology*. 19. 1046-54. 10.1097/MEG.0b013e3282f198a0.
- [16] R. Montoliu, R. Martín-Félez, J. Torres-Sospedra, A. Martínez- Usó, Team activity recognition in association football using a bag-of-words-based method, *Human Move. Sci.* 41 (2015) 165– 178.
- [17] L.N.N. Nguyen, D. Rodríguez-Martín, A. Català, C. Pérez- López, A. Samà, A. Cavallaro, Basketball activity recognition using wearable inertial measurement units, in: *Proceedings of the XVI International Conference on Human Computer Interaction, Interacción '15*, Association for Computing Machinery, New York, NY, USA, 2015.
- [18] M. Ullah, M. Mudassar Yamin, A. Mohammed, S. Daud Khan, H. Ullah, F. Alaya Cheikh, Attention-based lstm network for action recognition in sports, *Electronic Imaging 2021* (2021),302–1.
- [19] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, S.W. Baik, Action recognition in video sequences using deep bi-directional lstm with cnn features, *IEEE access* 6 (2017) 1155–1166.
- [20] A. Nadeem, A. Jalal, K. Kim, Automatic human posture estimation for sport activity recognition with robust body parts detection and entropy markov model, *Multimedia Tools Appl.* 80 (2021) 1–34.