Baseball Injury Risk Analysis — Project Documentation

Table of Contents

Introduction

The "Baseball Injury Risk Assessment" project aims to provide an intelligent system capable of analyzing baseball player movements from video footage to assess potential injury risks. Leveraging computer vision and machine learning, this tool identifies risk-prone movements in joints such as the shoulder, elbow, and knee. By doing so, it empowers coaches, analysts, and athletes with data-driven insights to enhance safety, training efficiency, and injury prevention strategies.

This project focuses on post-game or training session analysis, using recorded videos to estimate poses, extract biomechanical metrics (specifically joint angles), and run them through a trained model to assess risk levels. The final product is an end-to-end system that accepts a user-uploaded baseball video and returns an overlaid version with pose landmarks, joint angles, and a predicted injury risk classification.

Problem Statement

In high-performance sports like baseball, athletes are at constant risk of injury due to repetitive motion and improper biomechanics. Traditional injury assessments rely heavily on manual video review or in-person analysis, which can be subjective, time-consuming, and error-prone. There is a growing need for automated, scalable systems that can process large amounts of video data to identify high-risk movements objectively.

Our project addresses this by developing an automated injury risk assessment tool that classifies movements as “safe,” “elbow risk,” or “shoulder risk” based on visual biomechanical analysis.

Project Objectives

- To extract pose landmarks from baseball videos using a robust pose estimation framework.
- To compute joint angles related to injury-prone areas (shoulder, elbow, knee).
- To train a machine learning model to classify injury risks based on joint angle patterns.
- To overlay results (pose + risk label) onto the video for intuitive feedback.
- To develop a web-based interface allowing users to upload videos and receive risk analysis.

Technology Stack and Rationale

Pose Estimation: MediaPipe

We chose MediaPipe as our primary pose estimation tool due to its ease of use, efficiency, and real-time performance. MediaPipe's pose model provides 33 body landmarks with consistent labeling, enabling accurate extraction of key joint coordinates.

Alternatives Considered:

- **OpenPose**: Although accurate, it was computationally heavy and difficult to set up for browser deployment.
- **AlphaPose**: Focused more on real-time multi-person tracking; overkill for our single-player sports context.
- **HRNet**: Very accurate but lacked the simplicity and browser efficiency that MediaPipe offered.

Model Training: TensorFlow (Keras)

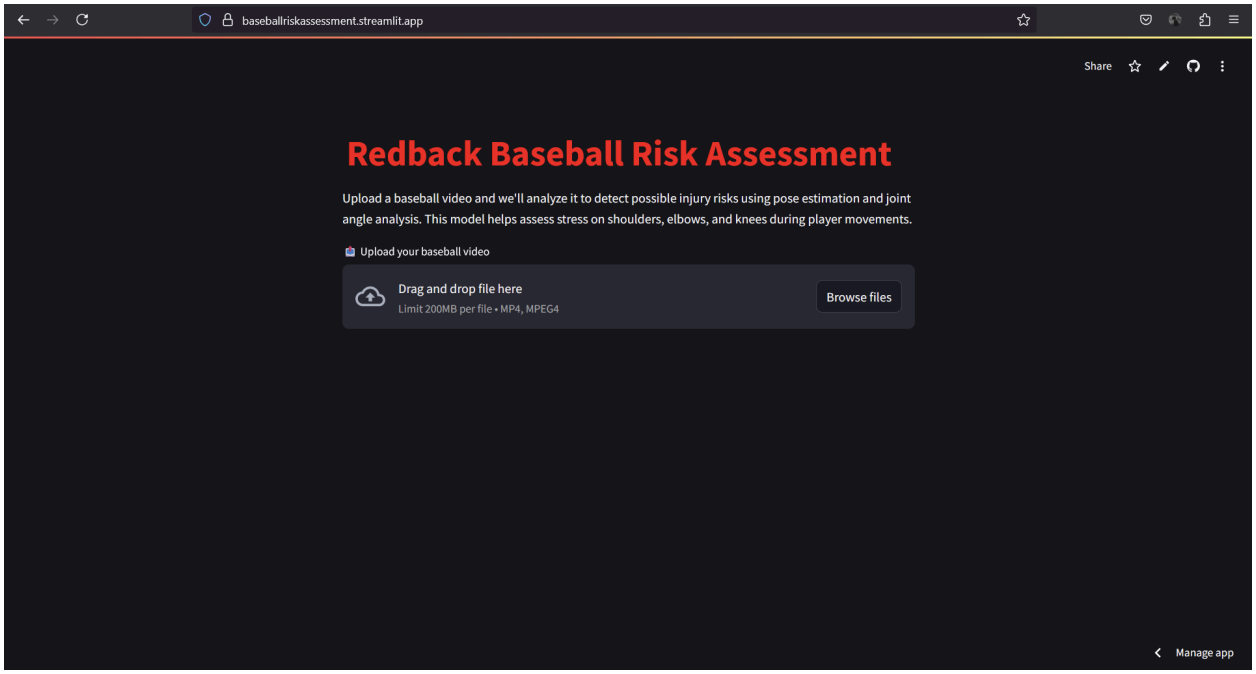
TensorFlow was used due to its compatibility with HDF5 (.h5) model saving/loading, ease of integration with web frameworks, and flexibility in neural network design.

Frontend: Streamlit

For deployment, we used **Streamlit Cloud** due to its simplicity in hosting Python-based web applications without requiring custom backend infrastructure.

Backend (Processing & Overlay): OpenCV

OpenCV was essential for video reading/writing, landmark overlay, and joint angle drawing. It allowed us to manipulate frames and write them back into video formats in real-time.



System Architecture

The application is structured with clear modular components, divided across folders and scripts to maintain readability, maintainability, and deployment compatibility. Here's a breakdown of the primary files and their responsibilities:

Main Functional Scripts

- **main.py**: This is the central script that handles end-to-end video analysis:
    - It takes the input video, extracts frames, uses MediaPipe for pose detection.
    - Calculates joint angles using geometric computation.
    - Predicts risk using the trained model.
    - Overlays the results and writes the annotated frames into a new video file.

- **streamlit_app.py**: Acts as the entry point for the web interface built using Streamlit Cloud. It handles file upload, triggers video processing, and renders the processed video for download via the browser interface.

Utility Scripts

- **utils/angles.py**: Contains the mathematical logic to compute joint angles using 2D coordinates. It implements the angle formula based on the cosine rule using NumPy.
- **utils/draw_overlay.py**: Handles the visual rendering of keypoints, skeletal lines, angle labels, and risk classifications directly onto video frames using OpenCV. This provides the visual feedback that users receive in their final downloaded video.
- **model/predictor.py**: Contains logic to load the trained Keras .h5 model and make predictions from extracted features (joint angles). It includes mapping from integer class IDs to readable labels like "elbow risk" or "safe".
- **model/trainer.py**: Defines the architecture of the neural network used for training on joint angle data. It reads data from a CSV file, preprocesses it, splits it into training and validation sets, and trains a lightweight classifier.
- **config.py**: Centralizes file paths, training constants (epochs, batch size), and directory references to improve maintainability and avoid hardcoding throughout the codebase.

- **dataset/label_mapper.py**: Maps class strings ("elbow", "shoulder", "safe") to numerical labels and vice versa to facilitate model training and prediction interpretation.

Dataset Development

To build a reliable classifier, we needed a dataset consisting of real-world baseball footage annotated with joint angles and associated injury risk labels. This phase involved three major steps: video selection, pose extraction, and angle-label pairing.

We began by sourcing four publicly available baseball videos from YouTube, focusing on different known movement patterns:

- **High-risk pitchers**: Drew Storen and Max Scherzer, known for aggressive throwing mechanics, were labeled as high risk (shoulder and elbow respectively).
- **Low-risk batters**: George Springer and Luis Arraez, representing more controlled and stable movements, were labeled as "safe".

Using MediaPipe Pose, each video was broken down frame-by-frame. For each frame where the pose was detected, the 2D coordinates of the relevant joints (shoulders, elbows, and knees) were extracted. These landmarks were then passed into the calculate_angle function, which computed six key joint angles:

- Left and Right Elbow
- Left and Right Shoulder
- Left and Right Knee

Each frame was tagged with the appropriate class label based on the known characteristics of the video (e.g., Max Scherzer — "elbow risk"). The data was compiled into a CSV file with columns:
[left_elbow, right_elbow, left_shoulder, right_shoulder, left_knee, right_knee, label]

This dataset served as the basis for supervised model training.

Dataset Snippet

	A	B	C	D	E	F	G	H
1	filename	left_elbow	right_elbow	left_shoulder	right_shoulder	left_knee	right_knee	label
2	elbow_risk_0000.jpg	44.72	24.32	17.37	25.28	174.83	178.55	elbow_risk
3	elbow_risk_0001.jpg	44.32	23.51	16.07	24.58	175.59	179.47	elbow_risk
4	elbow_risk_0002.jpg	43.12	24	14.47	23.44	175.77	179.24	elbow_risk
5	elbow_risk_0003.jpg	45.57	25.17	17.74	24.25	174.81	178.4	elbow_risk
6	elbow_risk_0004.jpg	43.86	22.28	17.15	21.49	174.86	178.35	elbow_risk
7	elbow_risk_0005.jpg	45.86	23.81	16.1	21.02	174.17	176.78	elbow_risk
8	elbow_risk_0006.jpg	46.02	24.96	14.96	22.64	174.71	176.72	elbow_risk
9	elbow_risk_0007.jpg	48.55	26.31	16.09	22.98	175.62	176.71	elbow_risk
10	elbow_risk_0008.jpg	47.1	22.48	16.23	18.99	175.46	176.33	elbow_risk
11	elbow_risk_0009.jpg	48.46	22.06	16.3	18.31	175.31	176.86	elbow_risk

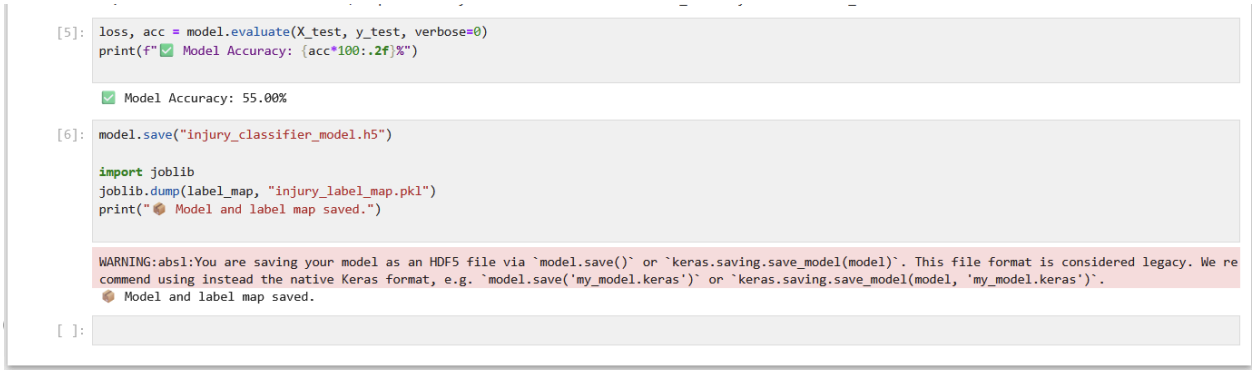
Model Training

The model is a lightweight feedforward neural network implemented in TensorFlow/Keras. It accepts six joint angles as input and outputs a probability distribution over three classes: safe, shoulder risk, and elbow risk. The architecture includes:

- An input layer of size 6 (corresponding to the six joint angles)
- One hidden layer with 64 neurons and ReLU activation
- A final softmax layer to classify into one of the three labels

The model was compiled with categorical crossentropy loss and the Adam optimizer. During training, the dataset was split into a 90/10 train-validation split. The model achieved high accuracy on the training data due to the clean angle-label pairing, although generalizability is expected to improve as more diverse data is introduced.

The model was then saved in .h5 format and loaded at runtime via the web application to classify incoming videos.



Pose Estimation Errors

MediaPipe Pose occasionally fails to detect landmarks in complex or occluded frames, leading to skipped frames or inaccurate overlays. This is especially problematic during fast pitching motions when limbs become blurred.

Small Dataset

Due to time constraints, the dataset was limited to four videos. This introduces a high chance of overfitting and limits the model's generalizability. In real-world deployment, the model might misclassify unfamiliar postures.

Resource Constraints

Deployment platforms like Render had memory limits that led to model crashes during video processing. Streamlit Cloud also limits file size and processing time, restricting real-time capability and necessitating short video lengths.

Labeling Assumptions

Risk labels were manually assigned at the video level based on known athletes. However, individual frames within a video might not always exhibit high-risk mechanics, introducing potential label noise.

Key Features

Video-Based Risk Assessment

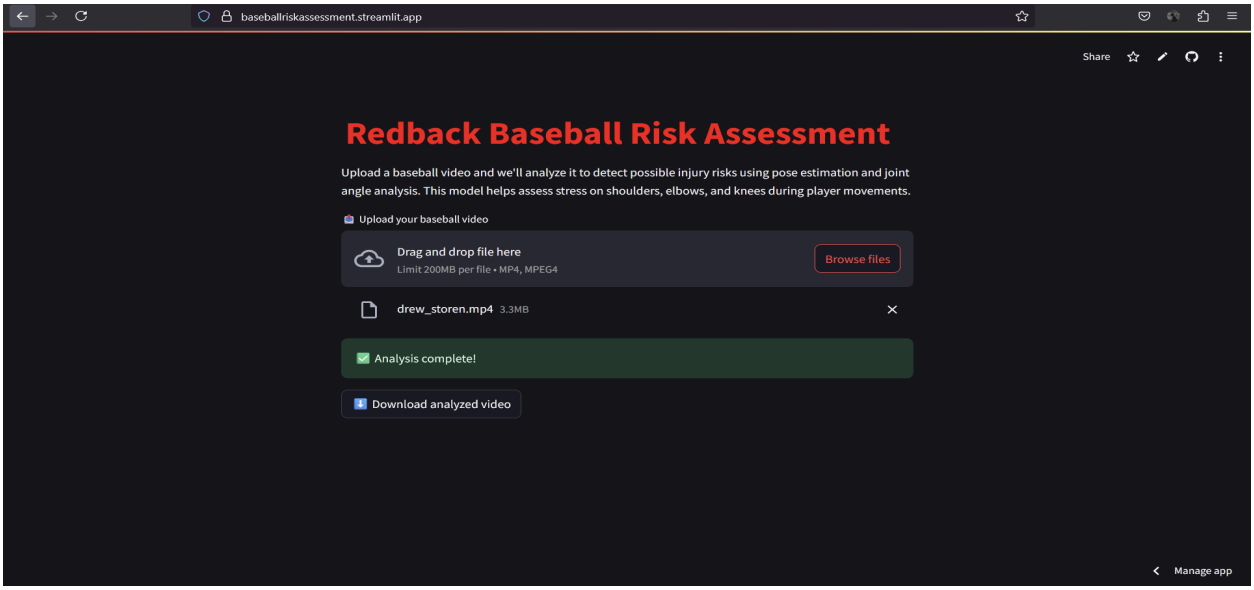
The core feature allows users to upload a standard baseball video and receive back a processed version with overlaid injury risk insights. This includes:

- Colored pose skeleton
- Text overlay indicating "safe", "elbow risk", or "shoulder risk"
- Joint angle values annotated per frame

Web Interface via Streamlit

We transitioned from Flask (Render-based deployment) to Streamlit Cloud for ease of hosting and real-time feedback. The interface is clean, mobile-friendly, and includes:

- File upload widget
- Spinner while processing
- Direct download of analyzed video
- Project description and usage instructions



Modular Design

Each component of the system is designed modularly, allowing easy future upgrades. For example:

- Swapping MediaPipe Pose with a more accurate model (like OpenPose)
- Replacing the classifier with an LSTM for temporal modeling
- Expanding dataset and retraining with new joints or sports

Future Work

- **Real-Time Feedback**: Deploying the model in a live setting using a webcam or live feed.
- **Expanded Dataset**: Including a larger variety of athletes and scenarios to improve robustness.
- **Injury Localization**: Not just labeling a risk, but identifying the exact joint under stress per frame.
- **Cross-Sport Generalization**: Extending the technique to sports like cricket, tennis, or football.

Conclusion

This Capstone project delivers a complete, functional pipeline for injury risk analysis in baseball using pose estimation and machine learning. It transforms passive video footage into a meaningful diagnostic tool, offering both visual and analytical insights. By making the system accessible through a web interface, we aim to democratize performance analytics for athletes, coaches, and healthcare professionals alike.

This project has also demonstrated the practical application of machine learning, software engineering principles, digital tools, and teamwork to address a real-world challenge — setting the stage for future expansion and industry use.