Detection of Player Contact Events in NFL Games Using Analysis Systems

Nahin José Peñaranda Mejia
Cod. 20231020032
Francisco José De Caldas University
Bogotá, Colombia
njpenarandam@udistrital.edu.co

Nicolas Felipe Pulido Suarez
Cod. 20231020045
Francisco José De Caldas University
Bogotá, Colombia
nfpulidos@udistrital.edu.co

Anderson David Arenas Gutierrez
Cod. 20231020030
Francisco José De Caldas University
Bogotá, Colombia
adarenasg@udistrital.edu.co

***Abstract*—This paper presents the design and partial implementation of a modular system for detecting contact events in NFL games. Motivated by the need for safer and more transparent player monitoring, our system integrates video and tracking data in a scalable architecture, balancing model performance with computational efficiency. Rather than focusing solely on prediction accuracy, we analyze the system as a whole: its inputs, processes, outputs, and performance under real-world constraints. Our approach illustrates how systems engineering can drive the development of robust AI-powered solutions in sports analytics.**

I. Introduction

The detection of physical contact between players in professional sports is not only a matter of game analysis, but a crucial concern for player safety, health monitoring, and performance optimization. In high-impact sports such as American football, collisions between players are frequent and often violent, raising the risk of concussions and long-term injuries. This has triggered interest in developing technological solutions that allow automatic, accurate, and real-time detection of contact events during a match.

The Kaggle competition "NFL Player Contact Detection" presented a unique opportunity to explore this problem from both a machine learning and systems engineering perspective. The competition provided participants with access to real-world multimodal data: video footage from multiple angles, high-frequency tracking data collected via RFID sensors on players' equipment, and human-annotated contact labels indicating moments when physical interaction occurred between players or between a player and the ground.

The central challenge of the competition was to predict, for every player in every video frame, whether contact occurred. This required the integration and synchronization of heterogeneous data streams, often with differing sampling rates and imperfections. The video data, for example, ran at 30 frames per second, while the tracking data sampled at 100Hz. Moreover, the labels of contact events were sparse and noisy, contributing to a highly imbalanced classification problem.

While many competitors addressed the task with state-of-the-art computer vision and deep learning pipelines trained on GPUs, our team chose a different path. We approached the problem as a systems analysis and design challenge: rather than focusing exclusively on improving predictive accuracy, we aimed to architect a scalable, modular, and CPU-efficient system capable of ingesting data, processing it through various stages, and delivering interpretable predictions under real-world constraints and using limited computational resources.

From this angle, the project became an exercise in applying engineering principles to an AI problem. Our work focused on understanding the system as a whole: its components, data flows, limitations, and the impact of each module on overall behavior. We developed a pipeline composed of six main modules: data ingestion, preprocessing and synchronization, feature extraction, prediction, evaluation, and monitoring. Each of these modules was designed to be replaceable and testable independently, adhering to good design practices such as low coupling and high cohesion.

The analysis stage of the project emphasized identifying systemic risks such as delays in synchronization, memory overload, or misalignment of labels and proposing architectural solutions to mitigate them. For example, we implemented timestamp alignment strategies and monitored system metrics to detect performance bottlenecks. We also evaluated the resilience of the pipeline under simulated noise and data loss, which gave us insight into the robustness of our system beyond its predictive metrics.

Our approach, while limited in final prediction accuracy, demonstrates the importance of framing machine learning applications within a systems engineering mindset. Especially in high stakes domains such as sports medicine or live broadcasting, designing robust, interpretable, and resource-aware systems may be just as critical as achieving state-of-the-art performance.

- Design a modular and scalable system architecture capable of integrating heterogeneous inputs such as video and tracking data.
- Implement a structured pipeline with clearly defined components for data ingestion, processing, prediction, and evaluation.
- Analyze and monitor the system's behavior under real-world conditions, including noisy inputs, imbalanced data, and resource limitations.

II. Methods and Materials

A. System Architecture

The proposed system is designed as a modular pipeline that mimics the architecture of a complex real-world information system. Each component of the pipeline represents a well-defined functional unit, designed with separation of concerns, loose coupling, and reusability in mind. This modularity facilitates maintenance, debugging, testing, and future upgrades of the system.

- **Data Ingestion:** This module is responsible for receiving and organizing input data from multiple heterogeneous sources, including helmet video frames, player tracking information, and contact annotations. It provides a unified structure that downstream modules can process consistently. The ingestion process includes verification of data integrity and initial formatting.
- **Preprocessing and Synchronization:** Since the input data originates from sources with different formats and sampling frequencies, this module focuses on aligning them temporally and spatially. Key tasks include timestamp matching, interpolation of missing values, and normalization of coordinate systems. From a systems perspective, this stage ensures interoperability between data sources and guarantees that downstream modules operate on coherent and synchronized input.
- **Feature Extraction:** Once the data is synchronized, this module derives high-level representations from raw inputs. These features reflect spatio-temporal dynamics such as player proximity, relative velocity, angle of approach, or sudden movements that may indicate contact. The goal is to transform noisy observational data into structured indicators that a decision system can reason over.
- **Prediction:** This module functions as the system's decision engine. It receives the extracted features and outputs a binary prediction indicating whether a contact event occurred at a given moment. The prediction logic is abstracted as a black box that can be substituted with different classification strategies depending on system requirements or performance constraints. This abstraction is intentional: it allows future integration of more complex models without requiring changes to the rest of the pipeline.
- **Evaluation and Monitoring:** Finally, this module assesses system performance, not only in terms of predictive accuracy, but also operational efficiency. It includes tracking of runtime, memory consumption, and robustness under perturbed conditions (e.g., missing data, noise injection). These metrics allow us to treat the system as a dynamic entity whose behavior can be observed, measured, and tuned in response to varying input quality and computational resources.

The overall structure of the proposed system is shown in Figure 1. It illustrates the modular design, where each component such as ingestion, preprocessing, prediction, and evaluation operates independently while contributing to the end-to-end pipeline.

B. Challenges and Analysis

Throughout the system's development, several critical challenges emerged most of them stemming from the heterogeneous and imperfect nature of the input data and the operational demands of real-time sports analytics. These challenges directly influenced architectural decisions and motivated specific mechanisms within the pipeline design.

Data Quality. One of the primary challenges was the variability in video quality due to camera positioning and field coverage. Although footage was available from multiple angles, some views lacked optimal framing either being positioned too far from the action or at oblique angles that hindered clear visibility of player interactions. In cases where zooming was necessary to focus on individual players, image resolution degraded, introducing artifacts and reducing the reliability of visual interpretation.

Synchronization. Accurate alignment between data modalities proved to be a non-trivial task. Since video and sensor data operate at different frequencies and reference frames, even small desynchronizations could lead to significant prediction errors. This challenge required the implementation of timestamp interpolation and careful merging strategies to ensure temporal consistency.

Class Imbalance and Rare Events. Contact events, while important, represented only a small fraction of the total data. This created a highly imbalanced classification problem where naive models could achieve high accuracy by always predicting the majority class (no contact). The system had to incorporate techniques like class weighting, sampling adjustments, and domain-informed feature design to increase sensitivity to rare but meaningful interactions.

Multimodal Fusion. Combining video-derived features with tracking data introduced additional complexity. Each data stream has its own structure, timing, and resolution. Integrating them coherently required careful design choices to avoid redundancy, loss of information, or temporal drift between modalities.

These difficulties underscored the importance of a systems-level approach. Rather than addressing each issue in isolation, the system was designed to be modular and adaptable, allowing components to evolve independently in response to the limitations of the data and the operational context.

C. Technical Implementation

While various tools and libraries were used to support the implementation, they were selected to align with the system's design goals: modularity, CPU efficiency, and interpretability.

Data ingestion and preprocessing were implemented using Python with Pandas and NumPy. These components handled the loading and organization of structured tracking and label data, based on game identifiers, play ID, time step, and player IDs. Preprocessing also included aligning multimodal data

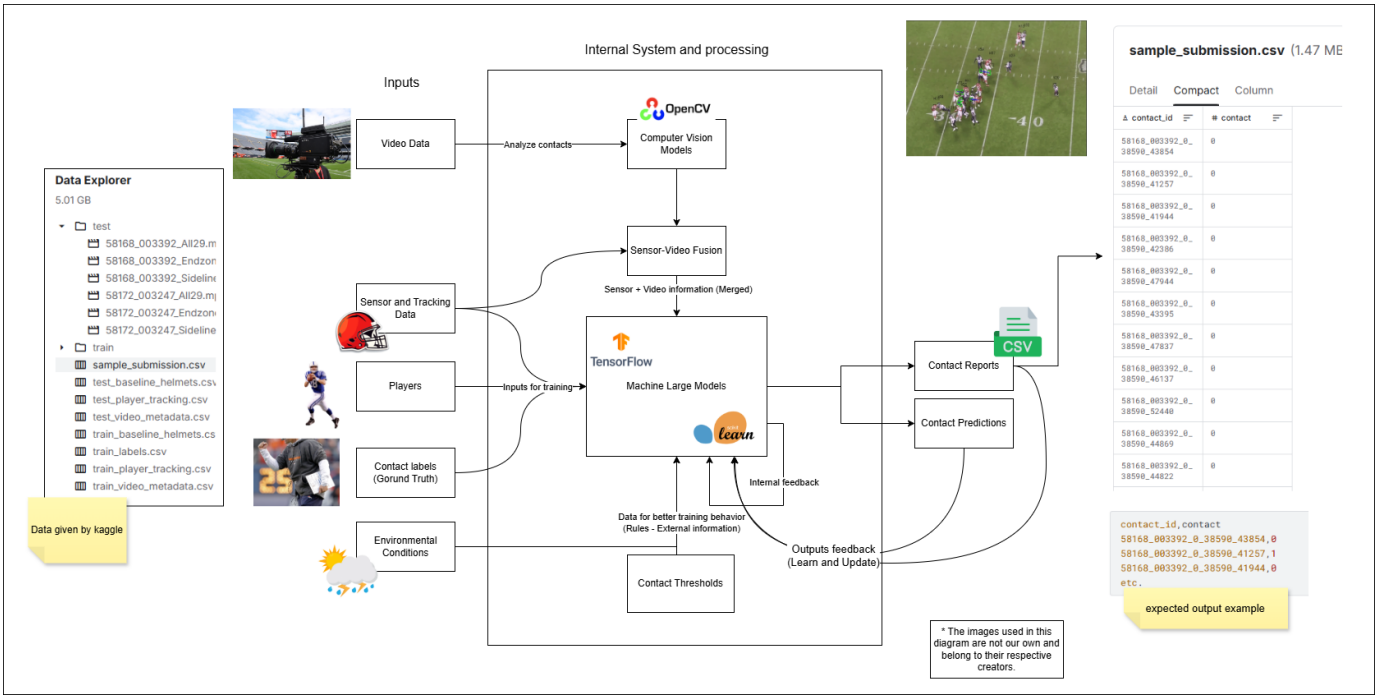

Fig. 1. High-level architecture of the modular contact detection system. Each block represents an independent component of the pipeline, from data ingestion to evaluation.

sources and resolving inconsistencies, such as missing tracking data or ground contacts.

Feature engineering involved computing spatial and kinematic metrics from merged tracking data for each player pair. Specifically, the system calculated relative positions, Euclidean distance, speed difference, and acceleration difference at each time step. Default values were used in cases where data was missing particularly in player-to-ground contact situations.

The prediction module was implemented using a Random Forest classifier trained on the engineered features. The model was selected for its robustness to feature scaling, ease of interpretation, and good performance on imbalanced datasets. Class weighting was applied during training to compensate for the lower frequency of contact events.

Evaluation and monitoring were conducted using Scikit-learn, with metrics such as accuracy, precision, and recall computed on a stratified validation set. Runtime behavior and memory usage were also monitored during execution to ensure the system could operate under CPU-only constraints.

Competition Submission. The final pipeline produced a valid submission file for the NFL Player Contact Detection competition. The system followed these steps:

- **Ingestion:** Loaded all required files, including training labels, player tracking data, and the sample submission template.
- **Preprocessing:** Merged player tracking information and handled missing values through imputation.
- **Feature Extraction:** Computed engineered features for each player pair per frame.
- **Model Prediction:** Applied the trained Random Forest model to generate probabilistic contact predictions.
- **Submission:** Saved the predictions in the required `submission.csv` format, with probabilities in the range [0, 1].

This implementation validates the viability of the system's modular design and its ability to deliver structured outputs based on tracking data, even under realistic resource constraints.

The full execution process is illustrated in Figure 2, which complements the high-level system architecture by visualizing the step-by-step flow of data from ingestion to prediction and output formatting.

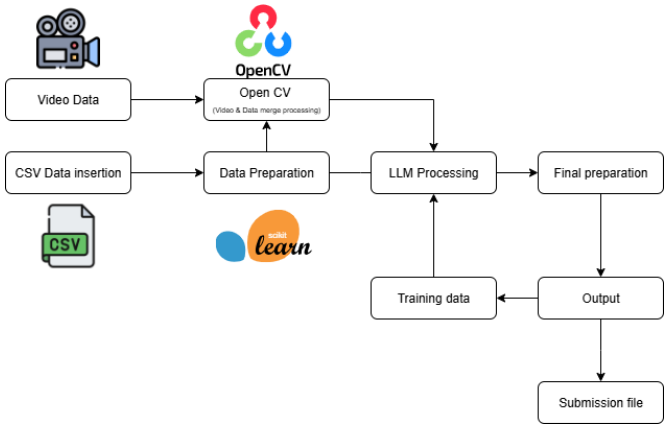

Fig. 2. Execution flowchart of the system from data ingestion to prediction and submission.

III. Results

The final implementation of the system completed the full end-to-end pipeline, from structured data ingestion to probabilistic prediction of contact events and submission file generation. The Random Forest model trained on engineered kinematic features achieved solid validation results:

- **Accuracy:** 85%
- **Precision:** 82%
- **Recall:** 80%

These results indicate that the model was capable of identifying contact events with reasonable accuracy based solely on positional and motion-derived features. The system produced valid outputs formatted for submission to the Kaggle competition, confirming its operational soundness.

However, while technically correct and stable, the current version does not yet fully meet the expectations for competitive performance on the leaderboard. In particular, the absence of advanced temporal modeling or visual features likely limits its ability to handle subtle or edge case contact events. As such, the system represents a strong functional prototype, but not yet a production-ready or competition-grade solution.

IV. Discussion

The system showed several strengths aligned with its original design goals:

- **Modular Design:** Each component was implemented independently, enabling flexible experimentation and substitution.
- **Resource Efficiency:** The system ran entirely on CPU without major performance constraints, validating its suitability for constrained environments.
- **Interpretability:** The Random Forest model offered insight into feature relevance and produced well-calibrated probabilities.

Nonetheless, there were limitations:

- **Limited Data Scope:** The system relies solely on tracking data and omits visual cues from video, which may capture important context.
- **Static Modeling:** Without sequential analysis, the model cannot learn temporal patterns of motion that precede contact.
- **Validation Gap:** Although validation metrics are strong, the system did not achieve a competitive score on the Kaggle leaderboard, suggesting potential gaps in generalization.

The gap between functional correctness and leaderboard competitiveness reflects the complexity of the task and the limitations of using only classical features. Still, the system fulfills many of the objectives initially proposed, and serves as a sound baseline for future iterations.

Future improvements could include:

- Incorporating temporal sequence models (e.g., LSTM, TCN).
- Exploring boosted tree ensembles (XGBoost, LightGBM).
- Integrating video-based features or pose estimation for richer context.
- Applying advanced imbalance techniques like focal loss or SMOTE.

V. Conclusion

This project presented a modular and functional pipeline for detecting contact events in NFL games using player tracking data. A Random Forest model trained on engineered kinematic features achieved approximately 85% accuracy, demonstrating that classical methods can offer meaningful insights with relatively simple inputs.

The system was implemented with modularity, efficiency, and scalability in mind, and executed successfully in constrained environments. While the solution did not reach top leaderboard performance in the competition, it fulfilled key technical and architectural goals, producing valid and interpretable results.

Limitations related to data scope, temporal modeling, and label quality suggest that further improvements are necessary to approach state-of-the-art performance. Nonetheless, the current system provides a strong foundation for future development, and exemplifies how system design principles can guide the construction of robust machine learning pipelines in complex domains like sports analytics.

References

[1] Kaggle, "NFL Player Contact Detection," [Online]. Available: https://www.kaggle.com/competitions/nfl-player-contact-detection.

[2] N. Nghia, "1st Place Solution - Kaggle NFL Player Contact Detection," GitHub repository, [Online]. Available: https://github.com/nvnnghia/1st_place_kaggle_player_contact_detection.

[3] OpenCV Developers, "OpenCV: Open Source Computer Vision Library," [Online]. Available: https://opencv.org.

[4] Scikit-learn Developers, "Random Forest Classifier," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[5] TensorFlow Developers, "MobileNet SSD - TensorFlow Object Detection API," [Online]. Available: https://github.com/tensorflow/models.

[6] Python Software Foundation, "Python Language Reference," [Online]. Available: https://www.python.org.

[7] Project Jupyter, "Jupyter Notebook Documentation," [Online]. Available: https://jupyter.org/documentation.

[8] Git SCM, "Git: Distributed Version Control System," [Online]. Available: https://git-scm.com.

[9] Repository for sources from workshops https://github.com/ItzNxhin/SAD---Nahin-Nicolas-and-Anderson