NFL Play Prediction and Visual Analysis

Mack Smith

Senior Project
Lipscomb University
5/23/2025

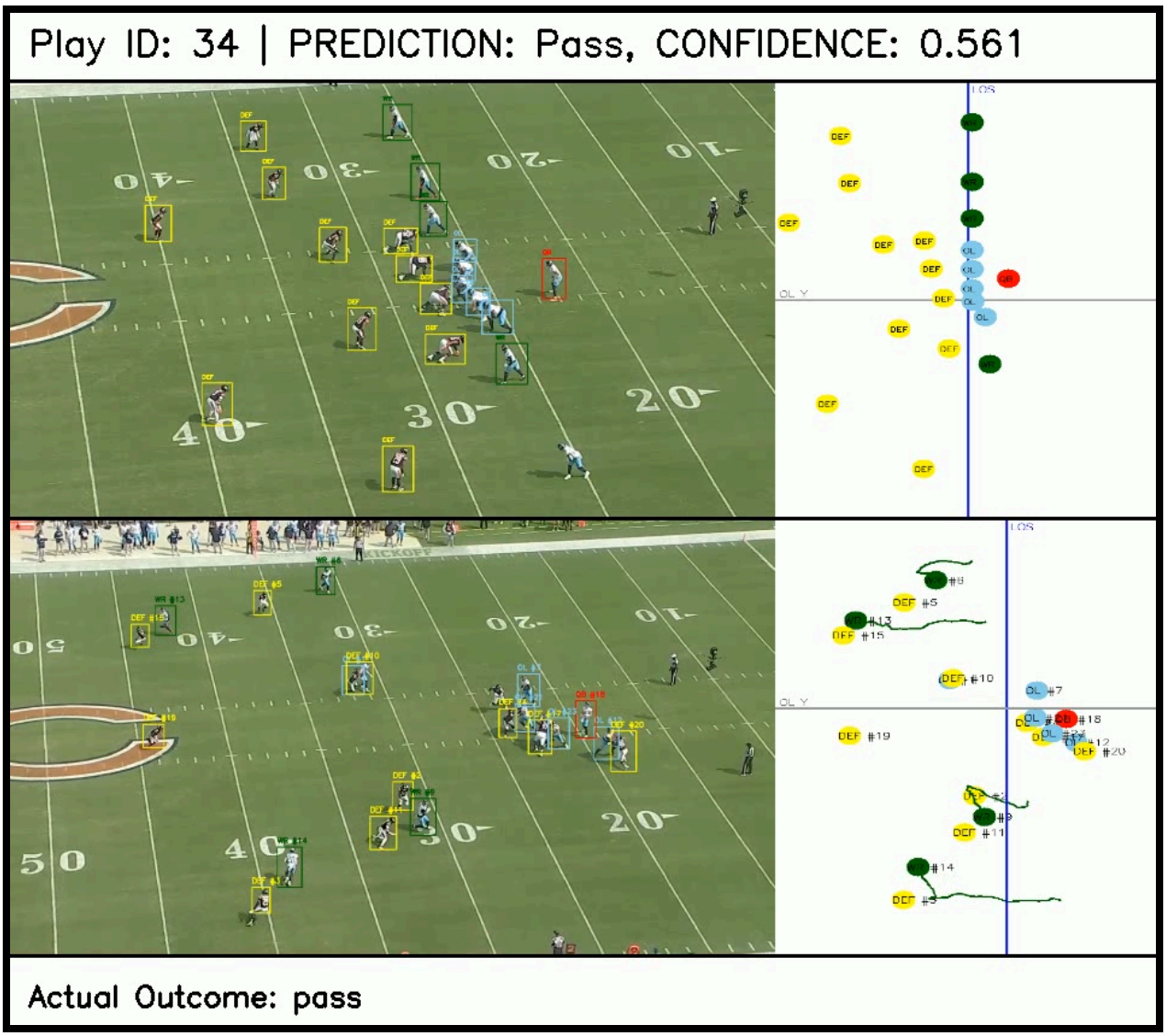


Table of Contents

I. Objectives _____ 2

II. Data Collection _____ 3

III. Data Processing and Manipulation _____ 4 - 6

IV. Play Type Prediction Model _____ 7 - 8

V. Position Identification _____ 9 - 10

VI. Player Tracking _____ 11 - 12

VII. Bird's Eye View _____ 13

VIII. Project Summary _____ 15

I. Objectives

The goal for this project is to generate a simulated live dashboard for analyzing Offensive plays in Football. This dashboard utilizes multiple machine learning and computer vision components to generate three core modules:

1. **Play Type Prediction Model**
    - A predictive model for classifying upcoming plays as either Run or Pass plays, using situational and historical game data.

2. **Position Identification**
    - Identify and visualize Offensive positions: QB, RB, WR, TE, OL as well as Defensive players, and Referees.
    - Generate a Pre snap bird's eye view of both the offensive and defensive formations.
    -
3. **Player tracking**
    - Assign tracking IDs to each identified player and follow their movements throughout the play.
    - Generate player paths for offensive skill positions (WR, RB, TE) to visualize route concepts and general offensive play design.

The final system provides a simulated continuous analysis for both the pre snap and post snap phases of an offensive play. It also supports iterative analysis, and recording features, for a series of plays, enabling both real time analysis and large scale analysis for pre recorded gamefilm.

II. Data Collection

For the play prediction model, I utilized play by play data for the entire 2024 Tennessee Titans offense. I modified data from nflverse, an open source resource for NFL data and analytics.

Open Source NFL game data from nflverse:

nflverse GitHub repository:

https://github.com/nflverse

NFL play by play data from nflverse (parquet):
https://github.com/nflverse/nflverse-data/releases/download/pbp/play_by_play_2024.parquet

This dataset provided more than enough information to support the development of a Pre-Snap Play Type Prediction Model. The final model was trained using the following *features:

- play_type (target)
- down
- ydstogo
- half_seconds_remaining
- score_differential
- yardline_100

* in addition to engineered features discussed on page 4

For the visual analysis modules, I used pre-recorded game footage. Each play was manually segmented into pre-snap and post-snap video clips for further processing.

III. Data Processing and Manipulation

Feature Engineering:

The final pre snap prediction model utilized two engineered features:

- pass_pct_last_20
- pass_pct_diff_10_vs_40

These two features were the most influential from over a hundred that were generated and tested.

The primary objectives when engineering features were:

1. Track play calling trends over multiple timescales.
2. Quantify shifts in the offensive game plan as the game progressed.

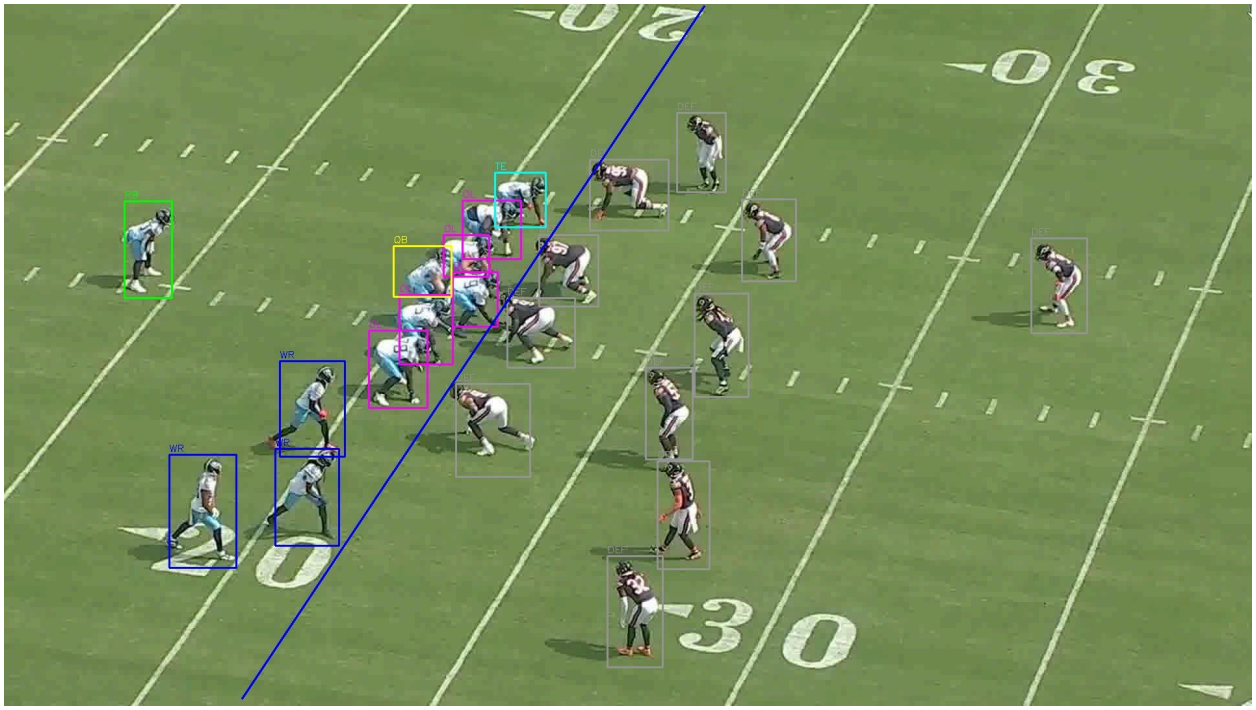
The feature, **pass_pct_last_20** represents a moving average for pass frequency over the last 20 play calls. This window captures around one third of an average game's snap count, providing a middle ground between short term tendencies and overall gameplan. While shorter timescales offer more precise insight into late-game play calling trends, this feature proved to be the most consistent across all game situations. In short, this feature provided additional insights into all available plays regardless of situation.

The feature, **pass_pct_diff_10_vs_40** captures a shift in pass frequency by comparing moving averages for the last 10 plays and the last 40 plays. With positive values representing a shift towards more passing, and a negative value representing a shift towards more running. Most of the time this feature will remain relatively neatureal, but it does an excellent job of signalling extreme shifts towards either run or pass.

I tested hundreds of combinations of features with similar formats to the above features, some of which provided better insights into game specific scenarios (early game and late game). A more focused play prediction model would benefit from more specific trend changes.

I also tested various engineered game scenario features related to game clock, down and distance, weather, and home field advantage. I believe that these features would be useful for more focused models as well, but they did not provide enough impact for the average play's situation.

Computer Vision:



*This is a manually labeled pre snap frame, using a python script to generate a json file with bounding boxes. The line of scrimmage label was not used for the final models.

In order to train a custom YOLO model, sample frames needed to be collected and manually labeled. For the position identification model, I chose to collect 5 frames per pre snap clip. Starting with the last available frame, moving backwards in intervals of 15 frames. This left me with, on average, 5 pre snap frames over the span of approximately 2 seconds of film.

This process allowed for two important concepts to be captured in virtually every play.

First, it allowed the model to view the final frame of the pre snap phase. This is the most important frame for determining formations and the final identification of positions. The model always receives a clear image of the final formation for the offense. If the model had been trained just on these frames, it would still achieve high levels of precision for identifying standard formations.

Second, the model also views four additional frames during the pre snap process. Which allows the model to pick up on pre snap motions and defensive shifts. Without these additional frames it would be extremely difficult to assign motioning receivers or running backs with accurate position labels.

The final iteration of the dashboard uses two YOLO models, with training data from the first 25 plays of the first game of the season. In total, this includes around 125 labeled frames, with around 2,750 total bounding boxes for players/referees.

IV. Play Type Prediction Model

Model Features:

***down**: Current down

***ydstogo**: Yards remaining for a first down

***half_seconds_remaining**: Seconds left in the current half

***score_differential**: Score difference between the teams

***yardline_100**: Distance from the opponent's end zone

pass_pct_last_20: Percentage of pass plays in the team's last 20 plays

pass_pct_diff_10_vs_40: Difference in pass percentage between the last 10 and last 40 plays

*Standard Scaled

Average Performance over 100 Random Seeds:

Accuracy: 0.6964

F1 Score: 0.7442

Precision: 0.7125

Recall: 0.7810

Macro F1 Score: 0.6844

Macro Precision: 0.6937

Macro Recall: 0.6834

Model Architecture:

XGBClassifier

n_estimators=200

max_depth=5

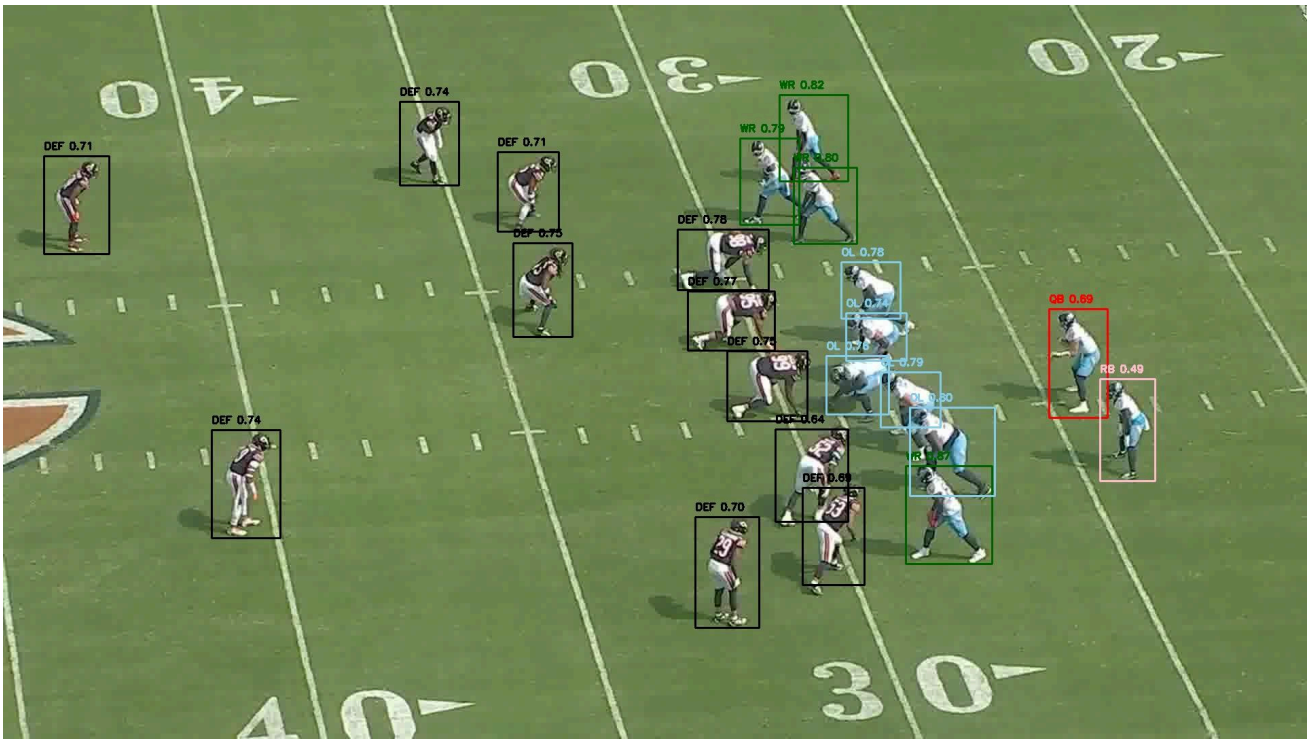
learning_rate=0.007,

eval_metric="logloss"

General Notes:

- 80:20 Train/Test split
- Slight class imbalance in favor of Pass, 55:45
- Running plays that result from QB scrambles are labeled as “run”, however, that contradicts the intent of the model. If QB scrambles were labeled as pass plays, model accuracy would likely improve.
- A Shotgun feature can boost this model’s accuracy ~2%, but was not implemented in this iteration.

V. Position Identification



***Predicted Player Positions via custom YOLO model**

Position Model Summary:

This module utilizes the **YOLOv8 Medium (yolov8m)** architecture, this model size was the best option for balancing accuracy and speed (both in terms of training and inference). While a larger model would have provided a better performance on a more powerful GPU, the aim of this project was to simulate real time analysis, making the medium model the best choice.

Model Strengths:

- The model showed strong overall performance for all player detections, mean average precision (mAP@0.5) of .798 represents reliable localization and classification for most classes.
- The model was particularly useful for identifying Defensive players with .983 Precision and .993 Recall. This is by far the best performance for the model, which later provided an opportunity to construct accurate Line of Scrimmage overlays.

Model Limitations:

- The biggest challenge in this module was the model's tendency to generate false positives for the Quarterback position. While overall metrics for QBs were reasonably strong, with a precision of 0.842, this still allowed for a significant number of incorrect QB detections. When identifying shotgun formations, which rely on precise QB positioning, even a small number of false positives undermined accuracy
- Tight Ends were expected to be the most difficult classification, given the inherently ambiguous nature of the position. Proximity to offensive linemen and similar movement to Wide Receivers made it very difficult to distinguish this position on such a small training set. The result of this ambiguity is a very low recall of .357 for Tight Ends. Despite these challenges, the model has still achieved high precision, meaning the model is still accurate when choosing to identify Tight Ends. In short, the model was too conservative when identifying Tight Ends.
- Referees and on field cameras also caused problems for generating false positives for other positions, but this was to be expected on such a small training set.

Model Evaluation

Class	Instances	Precision	Recall	mAP@0.5	mAP@0.5:0.95
All	550	0.935	0.737	0.798	0.408
QB	22	0.842	0.971	0.924	0.408
RB	20	0.969	1.000	0.995	0.532
WR	70	0.982	0.871	0.988	0.597
TE	28	0.933	0.357	0.671	0.399
OL	126	0.835	0.965	0.938	0.431
DEF	274	0.983	0.993	0.988	0.468
REF	10	1.000	0.000	0.078	0.024

VI. Player Tracking

Player Tracking Model

In order to track players after the snap, I needed to create a separate, modified, version of the previous YOLO model. This new model utilized the same pre labeled images from the original model for training, but with a single class. All bounding boxes for players were converted to class: Player. This conversion was necessary because the previous model was depending on the last few frames of the pre snap formation to assign position labels, once the snap takes place, starting formation is irrelevant. In short, the position model was great at finding bounding boxes for players after the snap, but classifications were no longer reliable.

In order to get around this issue, we classify all players as a single class, and retrain the new model. Creating a model that only prioritizes player identification without the need to assign player roles, improving overall precision and recall for any given player.

Player Tracking Model Metrics

Class	Images	Instances	Precision	Recall	mAP@0.5	mAP@0.5:0.95
All	25	550	0.963	0.975	0.971	0.488

This model achieved excellent precision and recall for identifying players throughout the play. These consistent classifications allow for object tracking and the assignment of unique IDs (via DeepSORT). However, because the new model was no longer classifying by role, unique IDs needed to be associated with player positions another way.

Linking Role Labels with Tracking IDs

To associate role labels with unique IDs, both the position model and the tracking model were applied to the first 5 frames of each post-snap clip. The process works as follows:

1. The position model assigns position labels (QB,RB,WR,TE,DEF).

2. The tracking model detects all players and generates bounding boxes used by DeepSORT to assign tracking IDs.

3. For each detected object, the Intersection over Union (IoU) between the bounding boxes from both models is calculated.

4. The best IoU match determines which tracking ID is linked to which role label.

5. All Tracking IDs that were successfully matched to their respective roles during this process are then tracked for the remainder of the play, **any additional or unmatched detections are ignored.**

The end result for this process allows for the creation of player objects with unique tracking IDs. Each unique ID has an associated role and is color coded to match pre snap visuals. By limiting the tracking IDs to just objects with assigned roles, the model ignores a significant amount of clutter for generating visuals. However, this limiting filter also makes it more difficult to track players after temporary occlusions or missed detections, since any new or reappearing objects without an initially assigned role are ignored by the system. For the purposes of this project, I chose to prioritize cleaner visuals with fewer false positives, accepting a higher rate of false negatives as a trade-off.

VII. Bird's Eye View

Yardline Detection

To create a simulated bird's eye view of the football field, the first step is detecting the yard lines in the video frame. These yard lines provide fixed, vertical reference points that help define the horizontal scaling of the field. The number of detected yard lines is crucial for generating a layout with proper scaling, I set the minimum number of detected yard lines to 5, any play with less than 5 yard lines will not render a bird's eye view.

Canny Edge Detection

In the first few frames of the play, the system converts the frame to grayscale and runs Canny edge detection to extract strong edges. Then it applies Hough Line Transform (cv2.HoughLinesP) to detect straight line segments. The goal is to only identify vertical yard lines, so lines must pass certain thresholds for length and slope in order to qualify.

Once 5 yard lines are detected within a frame, the perspective for the bird's eye view is locked. All other frames will use the same, locked, perspective for projecting players.

Player Projection

Once the field layout is established, the system begins projecting player positions onto the bird's eye view. For each player, the system calculates the center of the bounding box in the original video frame, and then scales these coordinates to the bird's eye view. Each Player's role and unique ID is then displayed on the bird's eye view.

There are two more important lines for scaling and centering the image. The system calculates the average x position and the average y position for all offensive linemen. This provides two lines, a vertical line representing the line of scrimmage, and a horizontal line that is used for centering the image. Once these two lines are detected, we lock both lines to the bird's eye view and use their intersection to lock the center of the image (only for the pre snap visual, post snap plays do not have a locked center).

To ensure the pre snap visuals align with real life NFL formations, we also use the line of scrimmage to calculate which side is offensive and which side is defensive, and 'force' players to appear on the correct side of the ball. Without this forcing method, slight distortions can sometimes make WRs appear a few feet beyond the line of scrimmage. The post snap visuals ignore this aspect of the line of scrimmage, providing slightly more accurate player paths.

Player Path Tracking

For post snap visuals, the system tracks and visualizes the paths for all potential receivers (RB,WR,TE). On pass plays, this captures the routes for each receiver in real time. This functionality can be easily tuned to track any role's movement for specific analysis.

Each player assigned a unique has their x position, y position, and role stored, for every frame, in a dictionary. Only the specified roles have their paths drawn to the bird's eye view in order to reduce visual clutter.

VIII. Project Summary

This project aimed to build a complete pipeline for offensive play analysis using a combination of machine learning, computer vision, and custom visualization tools. The final result of this project is an automated dashboard that provides play type prediction, a real time visual for player positions and pre snap formations (with an attached bird's eye view), and post snap player tracking (with an attached bird's eye view).

The play prediction model uses situational features and engineered trends from play-by-play data to forecast whether a given play will be a run or a pass. The position identification model uses a custom-trained YOLOv8 architecture to detect and classify players by position (QB, RB, WR, TE, OL, DEF, REF) based on pre-snap video footage. The player tracking module expands this functionality into the post-snap phase by simplifying classification, enabling path tracking for individual offensive players across time.

In the end, the full pipeline delivers a functional prototype of a live analysis dashboard that can process and visualize a play from its formation through to its execution.