NFL Play Classification Using GNN and Player Tracking Data

Carlos E. G. R. Souza[1], **Eduardo J. S. Luz**[1], **Vander L. S. Freitas**[1]

[1]Postgraduate Program in Computer Science
Federal University of Ouro Preto (UFOP)
Ouro Preto – MG – Brazil

Abstract. *This study addresses the classification of offensive plays in the National Football League (NFL) as either pass or rush, utilizing official NFL datasets from 2022-2023 season. The central hypothesis posits that the relative spatial positioning of players on the field, along with their interactions, directly influences the offensive strategy. To investigate this, we generate graph-based representations connecting players, which serve as input for a Graph Convolutional Network (GCN) for the subsequent classification of these graph structures. The empirical results demonstrate that the GCN-based framework outperforms conventional machine learning architectures, namely Random Forest and Multilayer Perceptron.*

1. Introduction

The use of data analysis and machine learning in sports prediction, especially in dynamic and complex sports such as American football, is gaining significant attention [Bunker and Susnjak 2022]. NFL games involve strategic territorial conquest through "drives", which are sequences of plays divided into three phases: pre-snap, snap, and post-snap. The pre-snap phase, where teams line up, is particularly critical as it often reveals strategic intent. The two primary types of offensive play are the pass (the quarterback throws the ball) and the rush (the quarterback hands the ball off for a run).

Anticipating opponent actions in the pre-snap phase is strategically important, leading to numerous studies focused on play prediction [Fernandes et al. 2020, Ötting 2021, Teich et al. 2016, Goyal 2020, Heiny and Blevins 2011]. Although some works, such as [Xenopoulos and Silva 2021], utilize tracking data to build graphs to predict yards gained, our work explores a different hypothesis: that the relative positioning and interactions of the players on the field directly influence the offensive strategy. We believe that this suggests an underlying graph structure that can be a valuable input for predictive models.

This leads to our central research question: *Can Graph Neural Networks (GNNs), combined with tracking data, outperform traditional machine learning models in classifying NFL plays as pass or rush?* Our experimental results suggest that this strategy indeed enhances predictive performance by effectively capturing the spatial and relational dynamics among players.

2. Related works

Our literature review, conducted using Google Scholar, IEEE Xplore, and SpringerLink with the query *("nfl" OR "national football league") AND "tracking data" AND "play prediction"*, revealed several trends.

Many studies employ traditional machine learning methods such as artificial neural networks, logistic regression, and gradient boosting, for play prediction, often achieving good accuracy. For instance, [Fernandes et al. 2020] used decision trees with various features for 72.3% accuracy, while [Teich et al. 2016] and [Goyal 2020] found decision trees and neural networks to be strong performers (summarized in Table 1). A distinct approach by [Ötting 2021] utilized Hidden Markov Models (HMMs) to account for the time-series nature of play data.

Table 1. Summary of related works indicating the methods, datasets, and the accuracy achieved.

Study	Method	Accuracy	Dataset
[Fernandes et al. 2020]	Decision trees	72.3%	2013-2017[1] Madden ratings[2]
[Ötting 2021]	HMM	71.6%	2009-2018 [3]
[Goyal 2020]	Logistic regression Neural network Decision tree Random forest	71.1% 75.7% 71.5% 75.3%	2009-2018[3]
[Heiny and Blevins 2011]	Discriminant analysis	40.38%	2005-2006[4]
[Biro and Walker 2022]	Reinforcement learning (MDP)	-	2017-2018[5] 2019 [6]

Tracking data has been used in NFL research for various objectives, such as evaluating quarterback performance [Reyers and Swartz 2023] and deriving new player evaluation metrics [Eager et al. 2022].

Regarding graph-based modeling in NFL, while some studies exist, none directly align with our goal of leveraging tracking data to uncover latent features from on-field player relationships for play prediction. Previous graph applications include representing league teams based on game outcomes [Xia et al. 2018], linking teams and quarterbacks for performance prediction [Salim and Brandao 2018], and using GNNs with Next Gen Stats to predict yardage gains [Xenopoulos and Silva 2021].

In conclusion, while traditional models like ANNs and Random Forests show strong accuracy, they often lack the ability to explicitly capture complex player interactions. This highlights a significant gap: the unexplored potential of using tracking data with graph-based models (GNNs) to understand on-field dynamics for enhanced play prediction. Our study aims to address this gap, leveraging the increasing availability of tracking data and initiatives like the Big Data Bowl to provide deeper insights.

[1] www.nflsavant.com
[2] www.maddenratings.weebly.com
[3] www.kaggle.com
[4] www.footballoutsiders.com
[5] www.github.com/maksimhorowitz/nflscrapR
[6] www.espn.com/nfl/stats

3. Problem Definition

We address the binary classification of NFL offensive plays as *pass* or *rush* using pre-snap player tracking data. This task is particularly challenging due to the large number of variables that may provide hints about the nature of the play. Factors such as the number of specific role players in the field, offensive formation, defensive coverage, on-field player statistics, current score, remaining time on the clock, player motion, and many other features must be taken into account to make an informed prediction.

Here we hypothesize that the relationships between players on the field, not only the static game statistics, are a fundamental aspect influencing the offensive strategy. The quarterback, often regarded as the tactical leader of the offensive unit, is responsible for making rapid and accurate decisions in high-pressure situations. To do so, he must assess the defensive alignment and anticipate the opposing team's strategy in a process known as the *pre-snap read*. In the brief moments before the snap, the quarterback evaluates critical aspects such as the defensive coverage, the players on the field, potential blitzes, and how defenders are matching up against wide receivers, in addition to contextual game information.

To incorporate these dynamics into the modeling process, we propose representing the play as a graph, as we formally define below.

3.1. Input Representation

A play P is modeled as a graph $G = (V, E)$, where: nodes V are players on the field, each with a feature vector $\mathbf{x}_i \in \mathbb{R}^d$. Features include position (x, y), speed s, acceleration a, orientation o, and metadata (e.g., position role, height). The graph edges E represent spatial connections between players. An edge e_{ij} exists if v_i and v_j are among the k-closest players (e.g., $k = 2$ - See Section 4.3) pre-snap. Finally, we also consider a global context $\mathbf{g}$ consisting of game state features (quarter, down, yards to go, score difference).

3.2. Output and Objective

The model predicts a binary label $y \in \{0, 1\}$, where $y = 1$ denotes a pass play and $y = 0$ a rush play. The objective is to learn a function $f : G \to \{0, 1\}$ that leverages graph structure, node features, and global context for classification.

3.3. Assumptions

The main assumptions are that only pre-snap data is used, as it encodes strategic intent. Moreover, ambiguous plays (e.g., scrambles, spikes) are discarded to ensure label reliability, and localized player relationships (e.g., blocks, coverage) are critical for classification.

3.4. Formalization

Given a dataset $\mathcal{D} = \{(G_1, y_1), \ldots, (G_N, y_N)\}$, we train a model f_θ parameterized by θ to minimize the cross-entropy loss:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log f_\theta(G_i) + (1 - y_i) \log(1 - f_\theta(G_i)) \right]. \tag{1}$$

This formulation models spatial interactions through graph connectivity, in contrast to prior work that flattens features into vectors. By integrating relational and contextual dynamics, the framework aims to uncover discriminative patterns between play types.

4. Materials and Methods

4.1. Data

A large volume of tracking data has become available for NFL games thanks to the collaboration between the NFL, Amazon Web Services, and other technology partners. These data are released annually through the *Big Data Bowl*, an initiative aimed at promoting innovation in sports analytics among students, professionals, and researchers.

Graphs are a powerful way to analyze relationships between entities, as highlighted in [Cormen et al. 2022]. Given the nature of tracking data, it is particularly suitable for modeling interactions among players as graphs. This representation makes it possible to extract hidden but important relational features that may not be evident from raw positional data alone.

The dataset for the 2022-2023 NFL season is extensive and organized into multiple files, each corresponding to a distinct domain of game-related information. All relevant files used in this work are publicly accessible on the official 2025 Big Data Bowl Kaggle competition page[7]. These files are structured as follows: *Game data*: contains general information about each game, including the date, home and away teams, and final score; *Play data*: provides details about individual plays, such as the quarter, down, yards to go, possession team, offensive formation, defensive coverage, and play outcome; *Player-play data*: captures player-specific actions during a play, including route run, motion at the snap, pass rushing role, fumbles, and passing yards; *Player data*: includes player statistics such as height, weight, position and name; *Tracking data*: includes player tracking data on every play such as x position, y position, direction and speed.

In our study, we carefully selected fields that are likely to influence play classification. Fields that do not impact the quarterback's decision to call a pass or a run, such as the date of the play or the full name of the player, were excluded from consideration. The selected fields are: play data: *quarter*, *down*, *yardsToGo*, *possessionTeam*, *gameClock*, *absoluteYardlineNumber*, *offenseFormation*, *receiverAlignment*, *playClockAtSnap*; tracking data: *playDirection*, *x*, *y*, *s*, *a*, *dis*, *o*, *dir*; player data: *height*, *weight*, *position*; game data: *week*, *homeTeamAbbr*, *visitorTeamAbbr*, *homeFinalScore*, *visitorFinalScore*.

The original dataset has 15,416 plays with a proportion of 60.41% samples of pass and 39.59% of rush. Due to the imbalance, we randomly downsampled the majority class to ensure a fifty-fifty distribution. For each week of the season, *pass* and *rush* plays were separated, shuffled - with a fixed seed to ensure reproducibility -, and an equal number of samples from each category were selected based on the smaller class size. After this balancing process, the remaining plays were divided into 80% for training, 10% for validation and 10% for testing in a stratified manner, i.e., maintaining the same balance between pass and rush plays in both sets. Table 2 shows the final sets for our experiments.

[7]Official 2025 Big Data Bowl competition page: `https://www.kaggle.com/competitions/nfl-big-data-bowl-2025`

Table 2. Numbers of samples in each set.

	Training set	Validation set	Test set	Total
Pass	4.882	610	611	6.103
Rush	4.882	610	611	6.103
Total	9764	1220	1221	12.206

4.2. Data Preprocessing

Our data preprocessing involved several key steps to prepare the raw NFL data for model training. First, we carefully selected relevant files, discarding the player-play data to prevent data leakage since it contained information that could reveal play outcomes (e.g., rushing yards, pass completion).

A crucial derived feature is the score difference between the possession and defense teams, calculated from plays data and games data. This variable is important because it influences offensive strategy.

Next, we focused on labeling plays as either *pass* or *rush*, a process that required careful inference since direct labels were not available. We excluded special play types like *qbSpike*, *qbKneel*, *qbSneak*, and *R* (Scramble) due to their unique nature or ambiguity. Plays are primarily labeled based on the presence of *passLocationType* or *rushLocationType*; ambiguous cases were further resolved using *passResult* or discarded.

We addressed missing values in *receiverAlignment*, *offenseFormation*, and *playClockAtSnap* by filling them with default values (*"EMPTY"* or *0*). Additionally, we performed several data transformations: converting *playDirection* to angles, *gameClock* to total seconds, and player height/weight to metric units.

A particularly useful derived feature is the total distance a player traveled before the snap, which offers insights into formations and defensive schemes. After these transformations, both the tracking data and the players data were merged using *nflId*.

Finally, all categorical variables are encoded using *LabelEncoder* to avoid high dimensionality, and we calculated the Euclidean distance to the two nearest players for each player on the field.

4.3. Graph Construction

To build the graphs for our model, we use the *networkx* library[8]. This is a well-documented and user-friendly Python package that makes it easy to create and manipulate graph structures and convert them into PyTorch tensors. Each graph consists of nodes that represent players on the field, and edges that represent the spatial relationships between them.

Every node corresponds to a player and is enriched with the following features: *club*: the team the player belongs to; *playDirection*: the player's orientation relative to the offensive play direction. If the play moves to the right, offensive players are oriented right, and defensive players left; *x*: the player's horizontal field position; *y*: the player's vertical field position; *s*: the player's speed; *a*: the player's acceleration; *dis*: distance traveled

[8] `https://networkx.org`

since the previous frame (in yards); *o*: the player's body orientation (in degrees); *dir*: the direction of the player's movement (in degrees); *height*: the player's height; *weight*: the player's weight; *position*: the player's field position (e.g., WR, RB, QB, CB); *totalDis*: total distance traveled by the player before the snap.

To define the graph connections, we leveraged local player interactions based on their proximity, regardless of their team affiliation. Initially, each player was connected with their n nearest neighbors, with n ranging from 2 to 10. This upper limit of 10 was chosen because it represents the maximum number of connections a player typically has with teammates on the field, although connections were established with any player on the field, allied or adversary. A visual representation of this $n = 2$ approach, where every player is connected with their two closest players on the field, can be seen in Figure 1. It is worth mentioning that the vertices may have more than two connections, because they could participate in the closest set to other vertices, besides having different closest neighbors of their own.

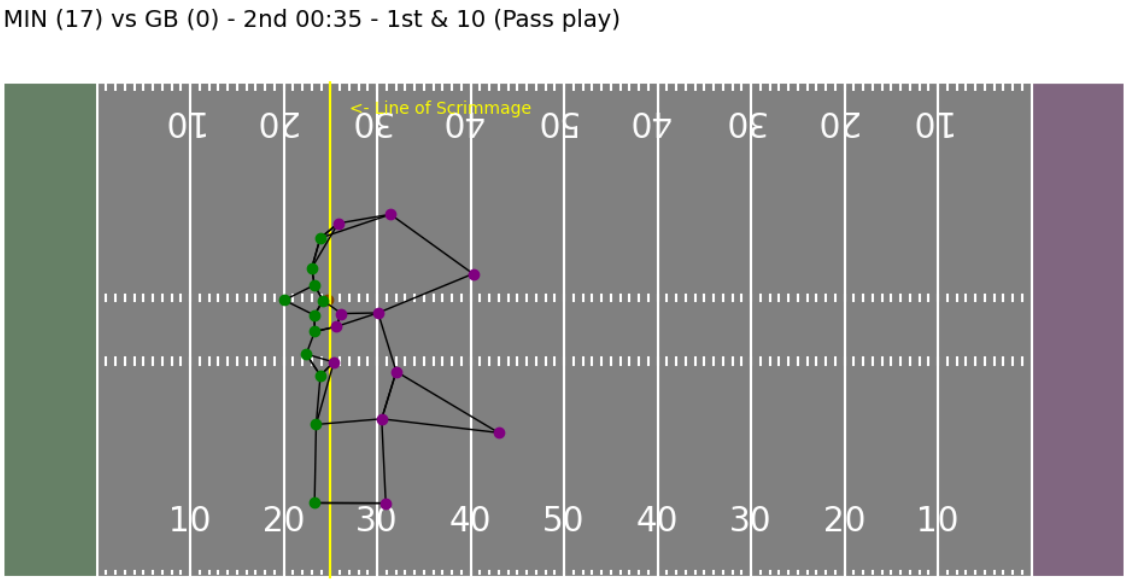

Figure 1. Graph representation of a play using the 2-closest strategy. The green vertices are players from Green Bay Packers (attack) and the purple vertices are players from Minnesota Vikings (defense).

In addition to node-level features, the graphs include global attributes that represent the state of the game: *quarter*: current quarter of the game; *down*: current down; *yardsToGo*: yards needed for a first down; *possessionTeam*: team in possession of the ball; *gameClock*: time on the game clock at the snap; *absoluteYardlineNumber*: yard line where the snap occurs; *offenseFormation*: offensive formation; *receiverAlignment*: receiver alignment on the field; *playClockAtSnap*: seconds left on the play clock at the snap; *possessionTeamPointDiff*: score difference from the offense's perspective; *playResult*: play label—pass or rush.

4.4. Hyperparameters

The hyperparameters for all models like learning rate, number of epochs, number of layers and many others were selected through an iterative trial and error process using the

training and validation sets results as a guideline. Instead of using hyperparameter optimization tools, twenty five experiments with different combinations of hyperparameters were conducted to identify a configuration that yelded the best performance. This manual approach allowed for a more intuitive exploration of the model's behavior under different settings. The range explored for each hyperparameter is defined below:

- HIDDEN_CHANNELS: $\{32, 64, 128\}$
- HIDDEN_LAYERS: $\{1, 2, 3\}$
- LEARNING_RATE: $\{0.01, 0.001, 0.0001, 0.00001\}$
- DROPOUT: $\{0.2, 0.3, 0.4, 0.5\}$
- WEIGHT_DECAY: $\{0.005, 0.0005, 0.00005\}$
- RF_ESTIMATORS: $\{50, 100, 150\}$

4.5. Graph Neural Network (GNN) Architecture

The chosen model for the GNN was the Graph Convolutional Network (GCN). This choice was primarily motivated by the GCN's ability to effectively capture the spatial and relational dynamics between players, which can be naturally represented as a graph. GCNs are specifically designed to learn from such structures by aggregating information from neighboring nodes, enabling the model to recognize coordinated movements, formations, and other collective behaviors. Originally proposed by [Kipf and Welling 2016], the GCN architecture was selected for its simplicity and strong performance in learning node representations through local neighborhood aggregation. This characteristic aligns well with the structured nature of the data, which reflects player positions and interactions during plays. The GCN is formally defined by [Kipf and Welling 2016] as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right), \tag{2}$$

where:

- $H^{(l)} \in \mathbb{R}^{n\times d}$ is the node feature matrix at layer l, with n nodes and d features.
- $H^{(0)} = X$, the input feature matrix derived from tracking data (e.g., position, speed, direction), as described in Section 4.3.
- $W^{(l)}$ is the trainable weight matrix at layer l.
- $\sigma(\cdot)$ is a non-linear activation function (e.g., ReLU).
- $\tilde{A} = A + I$ is the adjacency matrix with added self-connections.
- $\tilde{D}$ is the degree matrix of $\tilde{A}$, with $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

The implemented GCN that generated the best results consists of two graph convolutional layers, each followed by a *ReLU* activation function. These layers progressively transform the node feature vectors, starting from the input dimension—equal to the number of features per node in the dataset—into a hidden representation of 64 dimensions, which is maintained constant across all hidden layers. Is valid to mention that to make it easier to deal, all the global graph attributes were added to every node in the graph so that these informations could easily be reached by the model. After the convolutional layers, the model applies a *global mean pooling* operation, which aggregates the node embeddings into a fixed-size graph-level representation. This readout strategy allows the model to summarize the entire play into a single vector, which is subsequently processed by a fully connected linear layer that maps it to a two-dimensional output corresponding to the possible classes (pass or run).

To train the model, we used the Adam optimizer with a learning rate of 0.001 and a weight decay of 5×10^{-5} for regularization. The batch size was established on the basis of available computational resources and empirical performance. All executions were consistently run for 500 epochs to ensure comprehensive testing. To mitigate overfitting, we incorporated a dropout layer with a dropout rate of 0.3 before the final classification layer.

4.6. Baseline Models

To evaluate the effectiveness of the proposed GNN model, we implemented two traditional machine learning models as baselines: a *Multilayer Perceptron (MLP)* and a *Random Forest (RF)* classifier. These models were trained using the same set of features available to the GCN but without leveraging the relational (graph-based) structure of the data. For this purpose, all node-level features were flattened and concatenated with the global graph-level features into a single feature vector, which served as input to the baseline models.

The MLP was implemented using the *MLPClassifier* from the *scikit-learn* library. Its architecture consists of only one hidden layer, containing 32 neurons, and using the *ReLU* activation function by default. The model was trained with a maximum of 3000 iterations, using a fixed learning rate of 0.001. To prevent overfitting, we applied L2 regularization with a penalty term $\alpha = 5 \times 10^{-4}$.

The Random Forest classifier that reached the best results was also implemented using *scikit-learn*, with the number of estimators (*i.e.*, trees in the forest) set to 100. The model utilized default settings for other parameters such as the maximum depth and splitting criteria, which were empirically found to perform reasonably well on the dataset.

All baseline models were evaluated using the same training, validation, and test splits as the GCN to ensure a fair comparison. The performance of each model was assessed using classification accuracy and other standard evaluation metrics.

4.7. Model Evaluation

Four metrics are employed to evaluate the models. The first one is accuracy, which represents the proportion of correctly classified instances out of the total number of instances. The second one is precision which measures the proportion of correctly predicted positive instances among all instances predicted as positive. The third metric is recall that evaluates the proportion of actual positive instances that were correctly predicted. Finally, the F1-score gives the harmonic mean of precision and recall, providing a single metric that balances the trade-off between them.

5. Results

Our experiments were conducted on a system featuring an Intel i5-13450HX processor, 16 GB of RAM, and an NVIDIA GeForce RTX 3050 6GB GPU. The project's source code is publicly available at GitHub [9].

The results, summarized in Table 3 and illustrated in Figure 2, indicate that the Graph Convolutional Network (GCN) is the most effective model for this classification task.

[9] `https://github.com/CaduRomaniello/NFL-play-classification-GNN`

Table 3. Evaluation Metrics for Different Models

Model	Play type	Accuracy	Precision	Recall	F1-score
MLP	*pass*	0.750	0.768	0.717	0.742
	rush		0.735	0.784	0.759
RF	*pass*	0.742	0.765	0.699	0.731
	rush		0.723	0.786	0.753
GCN ($n = 2$)	*pass*	**0.782**	**0.828**	0.710	0.765
	rush		0.746	**0.853**	**0.796**
GCN ($n = 3$)	*pass*	0.763	0.797	0.705	0.748
	rush		0.736	0.820	0.776
GCN ($n = 4$)	*pass*	0.745	0.736	0.766	0.751
	rush		0.756	0.725	0.740
GCN ($n = 5$)	*pass*	0.748	0.784	0.684	0.731
	rush		0.720	0.812	0.763
GCN ($n = 6$)	*pass*	0.747	0.790	0.673	0.727
	rush		0.715	0.822	0.765
GCN ($n = 7$)	*pass*	0.741	0.756	0.714	0.734
	rush		0.729	0.769	0.748
GCN ($n = 8$)	*pass*	0.737	0.785	0.653	0.713
	rush		0.703	0.822	0.758
GCN ($n = 9$)	*pass*	0.734	0.783	0.648	0.709
	rush		0.700	0.820	0.755
GCN ($n = 10$)	*pass*	0.729	0.759	0.671	0.712
	rush		0.705	0.787	0.744

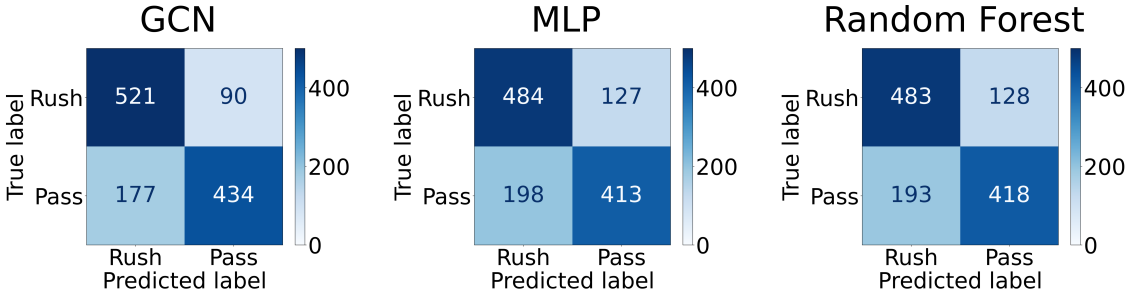

Figure 2. Confusion matrices for GCN, MLP, and RF models.

5.1. The impact of graph neighborhood

The GCN model with a neighborhood size of $n = 2$ achieved the highest performance across nearly all metrics. It reached an overall accuracy of 0.782, surpassing both the MLP (0.750) and Random Forest (0.742) baselines.

This model's strength lies in its nuanced understanding of play dynamics. It achieved an excellent recall of 0.853 for rush plays, meaning it correctly identified over 85% of all actual rushes, demonstrating a low rate of false negatives for this class. Simultaneously, it obtained a strong precision of 0.828 for pass plays, indicating that when it predicted a pass, it was correct nearly 83% of the time, thus producing few false positives

for passes. This balanced performance, also reflected in the highest F1-scores, shows its capability to effectively distinguish between the two most fundamental play types by leveraging the relational structure of the data.

A key finding from our hyperparameter search for this specific dataset was the performance degradation as the neighborhood size (n) increased. After the peak performance at $n = 2$, accuracy consistently declined. This trend strongly suggests the presence of over-smoothing.

For small n (like $n = 2$), the GCN aggregates features from immediate neighbors, capturing meaningful local interactions between players. However, as n grows, each node's representation becomes an average of a much larger and more homogeneous set of nodes in the graph. This smooths out the unique features of each node, making them too similar to distinguish effectively. In essence, $n = 2$ represents the best neighborhood for this dataset—capturing sufficient relational context without diluting the critical information that defines a specific play type. Further validation on other datasets is necessary to generalize these findings.

5.2. Comparison with Baseline Models

In contrast, the MLP and Random Forest models, while robust, treat the input data as a flat feature set, ignoring the inherent spatial relationships between players on the field. The MLP achieved a respectable accuracy of 0.750, but its confusion matrix shows a higher number of misclassifications (170 false rushes) compared to the GCN. The Random Forest model performed similarly.

Ultimately, the GCN's outperformance is not just marginal; it highlights a fundamental advantage. By modeling the players and their proximity as a graph, the GCN with $n = 2$ accesses a richer, more contextual layer of information, leading to a more accurate and reliable classification of complex NFL plays.

6. Conclusion and Future Work

In this study, we explored the application of Graph Neural Networks (GNNs) for classifying NFL plays, leveraging player tracking data to model spatial and relational patterns. Our approach, which represented each play as a graph connecting players to their two closest neighbors, demonstrated that GNNs significantly outperform traditional baseline models like MLPs and Random Forests. This result underscores the value of relational modeling for capturing the complex, dynamic interactions inherent in team sports.

Building on these promising preliminary results, which serve as a strong proof-of-concept, we have identified several key areas for future work to rigorously validate our hypothesis and enhance the model's capabilities.

First, to ensure a more robust and fair comparison, future iterations should benchmark the GCN against more sophisticated baseline models. A comprehensive evaluation must also include statistical significance testing to formally validate the performance differences and an analysis of running times to assess the computational efficiency of each approach. Furthermore, conducting ablation studies will be crucial to systematically understand the contribution of each model component, such as specific node features versus the graph structure itself.

Second, the graph construction methodology requires deeper exploration. Our use of k-nearest neighbors is simple, but the arbitrary choice of edges based solely on proximity ignores the inherent roles and hierarchies within a play. Future work should investigate graph structures informed by domain knowledge, for instance, by creating edges that explicitly connect the quarterback to all eligible receivers. Additionally, we plan to explore parameter-free graph construction methods, such as Delaunay triangulation, Gabriel Graphs, or Relative Neighborhood Graphs, to generate less arbitrary and more geometrically sound structures.

Finally, further methodological enhancements are needed. A more systematic hyperparameter tuning process using automated optimization tools, like Bayesian optimization, should be implemented to ensure the model is finely tuned. We also plan to train and test the model on the full, non-downsampled dataset. This will challenge the model to perform under realistic class imbalance conditions, providing a truer assessment of its behavior in real-world scenarios.

Pursuing these avenues will not only lead to more accurate and interpretable models but also significantly advance the application and understanding of GNNs in the nuanced field of sports analytics.

Acknowledgments

The authors thank the Minas Gerais State Research Foundation (FAPEMIG) for financial support. Furthermore, we acknowledge that ChatGPT [10] was used to help correct grammar errors.

References

[Biro and Walker 2022] Biro, P. and Walker, S. G. (2022). A reinforcement learning based approach to play calling in football. *Journal of Quantitative Analysis in Sports*, 18(2):97–112.

[Bunker and Susnjak 2022] Bunker, R. and Susnjak, T. (2022). The application of machine learning techniques for predicting match results in team sport: A review. *Journal of Artificial Intelligence Research*, 73:1285–1322.

[Cormen et al. 2022] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2022). *Introduction to algorithms*. MIT press.

[Eager et al. 2022] Eager, E., Brown, B., Chahrouri, G., Riske, T., Spielberger, B., Yui, L. S., and Seth, T. (2022). Using tracking and charting data to better evaluate nfl players: A review. In *2022 MIT Sloan Sports Analytics Conference*.

[Fernandes et al. 2020] Fernandes, C. J., Yakubov, R., Li, Y., Prasad, A. K., and Chan, T. C. (2020). Predicting plays in the national football league. *Journal of Sports Analytics*, 6(1):35–43.

[Goyal 2020] Goyal, U. (2020). *Leveraging machine learning to predict playcalling tendencies in the NFL*. Doctoral dissertation, Massachusetts Institute of Technology.

[Heiny and Blevins 2011] Heiny, E. L. and Blevins, D. (2011). Predicting the atlanta falcons play-calling using discriminant analysis. *Journal of Quantitative Analysis in Sports*, 7(3).

[10] `https://chatgpt.com`

[Kipf and Welling 2016] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[Reyers and Swartz 2023] Reyers, M. and Swartz, T. B. (2023). Quarterback evaluation in the national football league using tracking data. *AStA Advances in Statistical Analysis*, 107(1):327–342.

[Salim and Brandao 2018] Salim, M. O. and Brandao, W. C. (2018). Predicting the success of nfl teams using complex network analysis. In *International Conference on Enterprise Information Systems (ICEIS)*, volume 1, pages 135–142.

[Teich et al. 2016] Teich, B., Lutz, R., and Kassarnig, V. (2016). Nfl play prediction. *arXiv preprint arXiv:1601.00574*.

[Xenopoulos and Silva 2021] Xenopoulos, P. and Silva, C. (2021). Graph neural networks to predict sports outcomes. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1757–1763. IEEE.

[Xia et al. 2018] Xia, V., Jain, K., Krishna, A., and Brinton, C. G. (2018). A network-driven methodology for sports ranking and prediction. In *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE.

[Ötting 2021] Ötting, M. (2021). Predicting play calls in the national football league using hidden markov models. *IMA Journal of Management Mathematics*, 32(4):535–545.