Tackling Big Data: Analyzing When, Where, and How Effectively Tackles Occur

By Teagan Milford, Rachana Aithal, Sanjhana Rangaraj, and Yolanda Ojeda

Image Credits: Adobe, Getty.

Abstract

“Down, set hike!”

This is a phrase that can be heard on football fields across the USA. It’s used (at least as a cliche) as a simple way to unify offensive players with regards to timing.

The more difficult part to predict is what comes after the snap.

In this article, we will analyze football data from the National Football League to glean insights into predicting characteristics of

plays. We were inspired by the NFL Big Data 2024 competition on Kaggle that provides a huge amount of data, courtesy of Next Gen Stats.

Let's see if we can "tackle" some insights out of this data!

***Keywords*:** NFL, Big Data, Predictive Modeling, Tackling, Ensemble Methods, Tackles, National Football League, Neural Network, Spatial Analysis, Convolutional Neural Network, Data Science, Football Strategy.

Introduction & Background

The National Football League (NFL) is a multi-billion dollar business, with the sum of total revenue earned by all the teams exceeding $18.6 billion in 2022. With over 17.5 million people on average watching each game, stakes are high for teams to perform well to retain sponsorships, advertising dollars, and strong fanbases. This, combined with the advent of sophisticated sports analytics systems, suggests the value that predictive modeling has for NFL teams. The massive amount of data available from NFL games allows for a seemingly endless amount of models to be made, with the main limitation being computing power.

Here, we seek to use predictive modeling to simulate the conditions of gameplay in order to identify strategic tactics for NFL coaches and their players. Specifically, we looked at tackling data to understand the impact that tackle occurrences, tackle location, and yards gained before a tackle have on the performance of offensive and defensive formations.

Data

In this report, we accessed data from the NFL Big Data Bowl 2024 competition on Kaggle, an annual competition that is hosted by the NFL. The competition overview notes the data as coming from Next Gen Stats. The 2024 theme of the competition is tackling, which is what inspired our methodology throughout our report process. No other outside data was accessed.

Description of Data

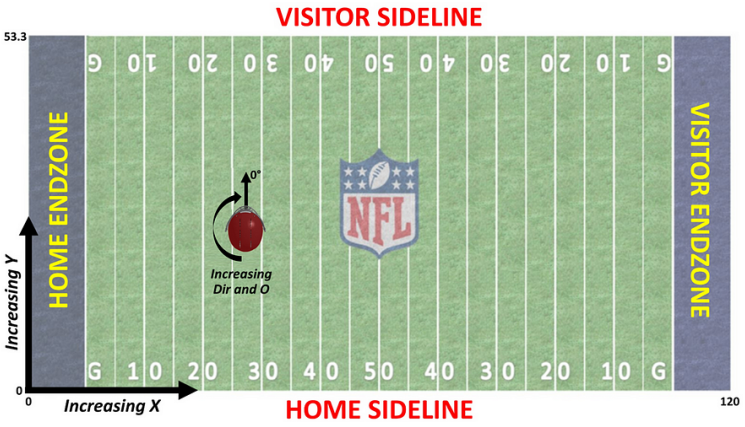
This page contains five different datasets, which are described below.

- *Game data*: Contains information about teams for each game. Columns include gameId, season, week, gameDate,

gameTimeEastern, homeTeamAbbr, visitorTeamAbbr, homeFinalScore, visitorFinalScore

- *Play data*: Contains play-level information from each game. This dataset has 35 columns, the most relevant including gameId, playId, ballCarrierId, playDescription, yardsToGo, possessionTeam, defensiveTeam, gameClock, passResult, passLength, playResult, offenseFormation, homeTeamWinProbabilityAdded, visitorTeamWinProbabilityAdded, expectedPoints
- *Player data*: Contains information about players. Columns include nflId, height, weight, birthDate, collegeName, position, displayName
- *Tackles data*: Contains player-level tackle information for each game and play. Columns include gameId, playId, nflId, tackle, assist, forcedFumble, pff_missedTackle
- *Tracking data*: Contains weekly player tracking data, spanning 9 weeks. Columns include gameId, playId, nflId, displayName, frameId, time, jerseyNumber, club, playDirection, x, y, s, a, dis, o, dir, event

GameId, playId, and nflId are the main identifiers that are used as keys to connect these datasets together. BallCarrierId is another term for nflId that specifically identifies which player is carrying the ball.



Football field as an x, y coordinate plane. Image Credit: Kaggle.

While most of these column names will be intuitive to understand, there are a few that warrant explanation.

Game data

- 'HomeTeamAbbr' and 'visitorTeamAbbr' refer to the shortened version of a team's name. For example, the Houston Texans would be referred to as *HOU*.

Play data

- 'PlayDescription' is a text description of the play that occurred. For example, a play involving ball carrier Jeffrey Wilson on the San Francisco 49s that gained 7 yards and ended with a tackle by William Gay and Justin Reid from the Kansas City Chiefs would appear as the following: "(15:00) J.Wilson left tackle to SF 32 for 7 yards (W.Gay; J.Reid)."
- 'PassResult' refers to outcome of the play, which can be *C* (complete pass), *I* (incomplete pass), *S* (quarterback sack), *IN* (intercepted pass), *R* (scramble), or null.
- 'OffenseFormation' is a text description of the formation that the possession team used in that play. Common values in this column are *SHOTGUN, SINGLEBACK*, and *I_FORM (I Formation)*.

Tackles data

- 'Pff_missedTackle' refers to a binary value provided by Pro Football Focus (PFF) that indicates whether a player missed a tackle in that play.

Tracking data

- 'X' and 'y' represent the x, y coordinates of the field, with the x ranging from 0–120 yards and the y ranging from 0–53.3 yards.
- 'S' refers to speed in yards per second and 'a' refers to speed in yards per second squared. 'O' is the player orientation in that frame and 'dir' refers to the angle of a player's movement, both given as a value in terms of 0–360 degrees.
- 'Event' is a simple text description of the play details.

Connecting the datasets using these keys unlocked the potential to conduct feature engineering and create new, more robust dataframes.

Note: Due to computing limitations, we could not use all 9 weeks of tracking data. Instead, we worked with a random sample of 50,000 rows from the tracking data for week 1.

Note: For the sake of brevity we have opted to describe only the most relevant columns used in our project. More in-depth descriptions

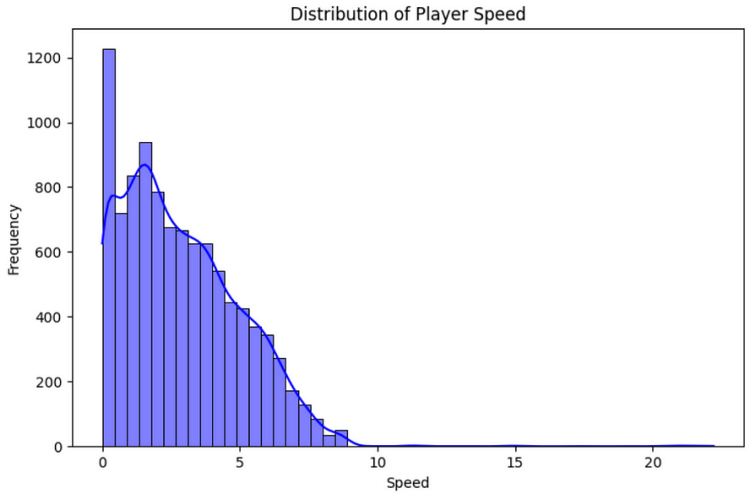
can be found on the NFL's Kaggle page.

Exploratory Data Analysis

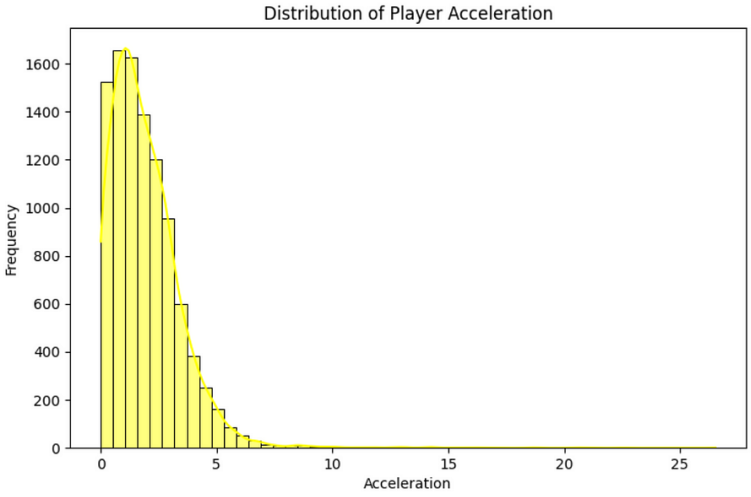
In order to better understand the data at hand, we conducted exploratory data analysis.

At first, we approached this broadly and made simple visualizations that were illustrative of the data more generally. We did this to see if any interesting or unexpected trends emerged that may inform our modeling going forward.

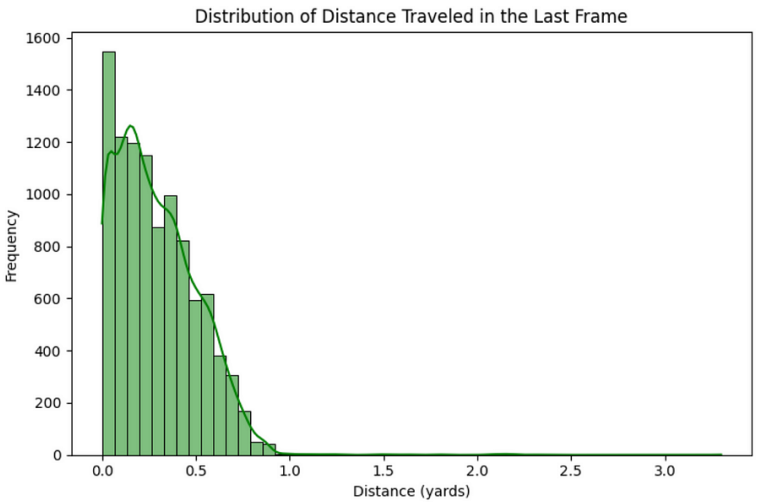
```
# setting up the subplots for visualization
fig, axes = plt.subplots(nrows=2, ncols=2, figsi

# distribution of 's' (speed)

sns.histplot(tracking['s'], bins=50, kde=True, a
axes[0][0].set_title('Distribution of Player Spe
axes[0][0].set_xlabel('Speed')
axes[0][0].set_ylabel('Frequency')

# distribution of 'a' (Acceleration)
sns.histplot(tracking['a'], bins=50, kde=True, a
axes[0][1].set_title('Distribution of Player Acc
axes[0][1].set_xlabel('Acceleration')
axes[0][1].set_ylabel('Frequency')

# distribution of 'dis' (Distance traveled in th
sns.histplot(tracking['dis'], bins=50, kde=True,
axes[1][0].set_title('Distribution of Distance T
axes[1][0].set_xlabel('Distance (yards)')
axes[1][0].set_ylabel('Frequency')

plt.tight_layout()
plt.show()
```



Distribution of player speed, derived from tracking data.

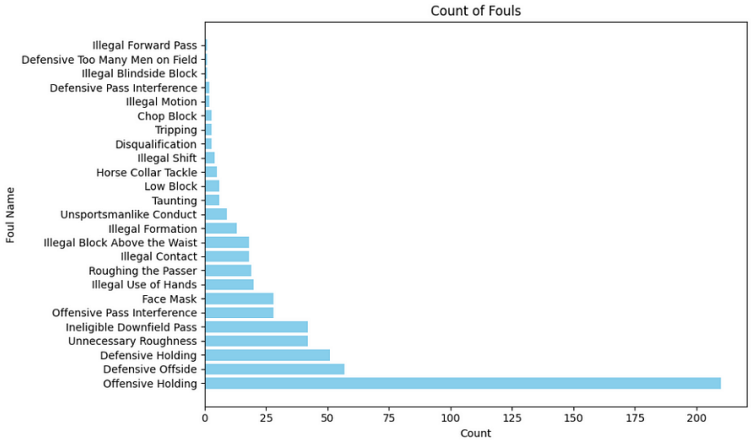


Distribution of player acceleration rates, derived from tracking data.



Distribution of the distance traveled in yards since the previous frame, derived from the tracking data.

```
foul_counts = plays['foulName1'].value_counts()

# convert to a dataframe
foul_counts_df = pd.DataFrame({'foulName1': foul

plt.figure(figsize=(10, 6))
plt.barh(foul_counts_df['foulName1'], foul_count
plt.xlabel('Count')
plt.ylabel('Foul Name')
plt.title('Count of Fouls')
plt.tight_layout()

plt.show()
```



Count of fouls by play type, derived from the play data.

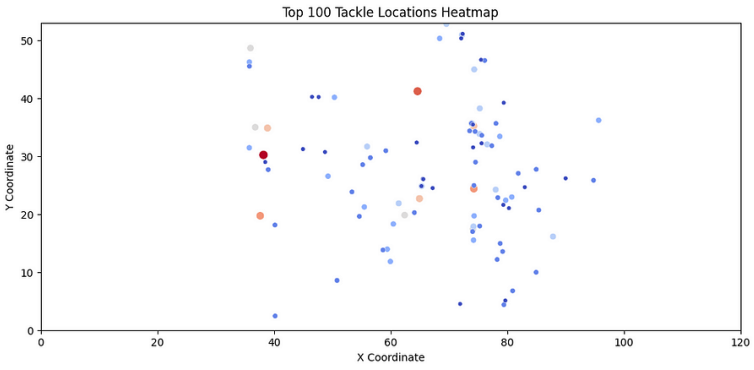
These visualizations give us an idea of some basic statistics from the datasets.

Then, considering that the theme of the Kaggle competition was tackling, we decided to look more specifically at the tackling data and its interaction with our other datasets.

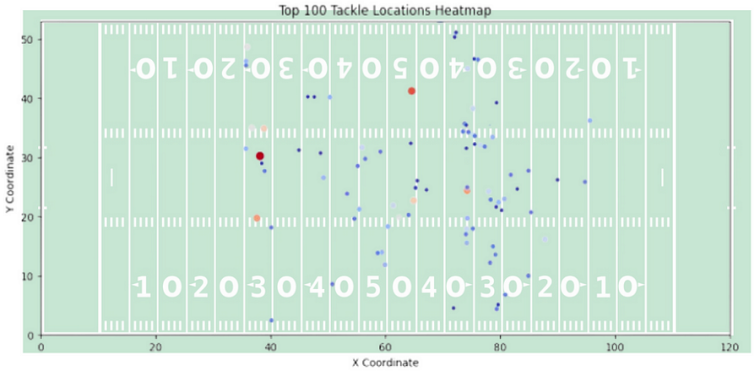
In the figure below, we plotted all occurrences in the tracking data for week 1 where the 'tackle' variable equals 1, indicating that a tackle occurred.

```
tackle_track = pd.merge(tackles, week1, on=['gam
tackle_track.head()
```

```
# filter the dataframe for instances where 'tack
tackles_df = tackle_track[tackle_track['tackle']

# get the top 100 locations for tackles & freque
top_tackle_locations = tackles_df['x'].value_cou

# create a dataframe with top 100 locations and
top_tackle_df = pd.DataFrame({'x': top_tackle_lo
                              'frequency': top_t

# create aspect ratio based on football field di
y_range = 53
x_range = 120
aspect_ratio = x_range / y_range

plt.figure(figsize=(12, 12 / aspect_ratio))  # s
sns.scatterplot(data=top_tackle_df, x='x', y='y'

plt.xlim(0, x_range)
plt.ylim(0, y_range)

plt.xlabel('X Coordinate')
plt.ylabel('Y Coordinate')
plt.title('Top 100 Tackle Locations Heatmap')

plt.show()
```



Top 100 tackle locations heatmap, derived from the tackles data and tracking data week 1.



Top 100 tackle locations heatmap overlayed on a football field, for visual comparison.

These visualizations give us an idea of the most prevalent tackle points on the field. The layout of the points suggests that there may be interesting insights to be had for both offensive and defensive strategy related to the coordinate locations in which tackles tend to occur.

In considering this, we must also be cognizant of the relationship of the data. We cannot suggest causation between tackle location and the offensive or defensive formations from this data, because it is unclear which caused the other.

With this in mind, we moved forward with pre-processing our data and conducting feature engineering.

Data Pre-Processing

In total, the original datasets had over 20,000,000 rows, making it computationally expensive and near impossible to process with our personal computers. Because of this, we opted to conduct random sampling on week 1 of the tracking data to decrease our data size to just 50,000 rows.

Feature Engineering

We leveraged insights from previous winners of the NFL's Big Data Bowl competitions to help us with completing this project. These insights informed our decisions of what features we should derive from our datasets.

Prior to conducting modeling, we added the following features:

- toRight
- isPlayLeftToRight
- isRusher

- isOnOffense
- yardIndexClipped
- yardIndex
- player_minus_rusher_Sy
- player_minus_rusher_Sx
- player_minus_rusher_y
- player_minus_rusher_x
- y_std
- x_std
- sx
- sy
- dir_rad
- dir_std

Many of these variables were informed by the solution for the 2020 competition called 'The Zoo' and a GitHub page that reproduces that solution. We also opted to transform the plays that move from right to left so that all of the plays we analyzed move from left to right, for simplicity. Thus, the data would be reshaped to be more consistently defense v. offense, making it more straight forward for the model to follow.

Modeling

Our study focuses on predicting three fundamental aspects—whether a tackle occurs, where a tackle occurs, and the yardage gained by the offense prior to the tackle occuring—related to tackles, contributing to a more comprehensive understanding of defensive dynamics.

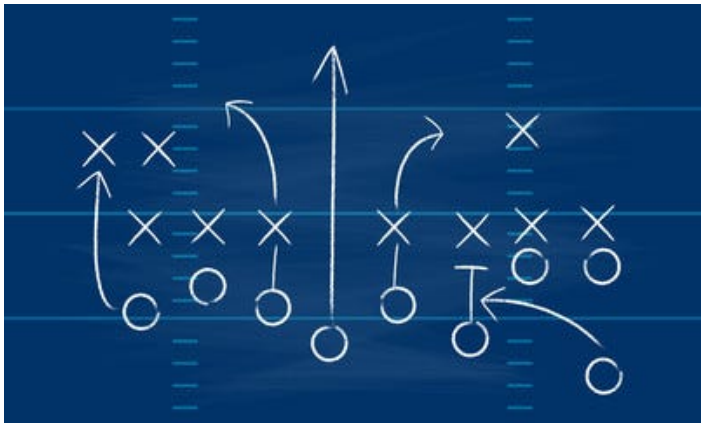


Image Credit: Adobe.

Tackle or no tackle?

The first dimension of our predictive modeling involves determining the likelihood of a tackle occurring during a given play. This binary classification task involves training a model on a rich dataset of historical plays, encompassing diverse game scenarios, player interactions, and environmental factors. By considering variables such as player positions, team strategies, and recent play sequences, our model aims to provide a nuanced prediction of whether a tackle will transpire, aiding teams in preemptive defensive planning.

Logistic regression: The logistic regression model achieved an accuracy of 0.58, indicating that it correctly predicted the outcome (tackle or not) approximately 58% of the time. However, this is a relatively low accuracy, suggesting that the model might not be capturing the underlying patterns in the data well.

Decision tree: The decision tree model performed better than logistic regression with an accuracy of 0.80271. It shows a good balance between precision and recall for both classes. Class 0 (not tackle) has a precision of 0.76, meaning when the model predicts not a tackle, it is correct 76% of the time. The recall of 0.79 suggests that it captures 79% of all actual not tackle instances. Class 1 (tackle) has a higher precision of 0.84, indicating that when the model predicts a tackle, it is correct 84% of the time. The recall of 0.81 suggests that it captures 81% of all actual tackle instances. The F1-score considers both precision and recall, providing a balanced measure, and it is higher for class 1.

Accuracy	Class	Precision	Recall
0.8	0	0.76	0.79
	1	0.84	0.83

Decision tree classification accuracy metrics.

Random Forest: The model performed exceptionally well with an accuracy of 0.88345. It exhibits perfect precision (1.00) for class 0, indicating that when the model predicts not a tackle, it is correct 100% of the time. However, the recall of 0.72 suggests that it misses 28% of actual not tackle instances. Class 1 has a high precision of 0.83, indicating that when the model predicts a tackle, it is correct 83% of the time. The recall of 1.00 suggests that it captures all actual tackle instances. The F1-score for both classes is high, reflecting a good balance between precision and recall.

We utilized cross-validation to fine-tune the Random Forest model hyperparameters. The optimal values were determined for n_estimators, max_features, and max_depth. This rigorous process helps ensure the model's generalization ability by preventing overfitting and improving performance on unseen data

Accuracy	Class	Precision	Recall
0.88	0	0.87	0.75
	1	0.83	0.91

Random forest classification accuracy metrics.

Ensemble Methods: The AdaBoost classifier, initialized with AdaBoostClassifier(random_state=42), and the XGBoost classifier, initialized with xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss'), demonstrated high accuracy in our tests. The use of use_label_encoder=False in XGBoost addresses a deprecation issue, ensuring compatibility with current standards. The choice of 'logloss' as the evaluation metric aligns with the binary classification nature of our task, focusing on logarithmic loss to evaluate model performance.

Through a comprehensive grid search, we fine-tuned the hyperparameters of these classifiers, exploring various combinations to optimize their performance on the training dataset. The subsequent training and evaluation process, conducted via the train_and_evaluate_classifier() function, revealed that both

classifiers achieved identical accuracy scores and produced similar confusion matrices.

This uniformity, particularly in the absence of false negatives, hints at potential data or model issues like class imbalance or overfitting, which warrant further investigation.

	Accuracy	AUC ROC
AdaBoost	0.907	0.897
XGBoost	0.907	0.936

AdaBoost and XGBoost accuracy metrics.

Both classifiers exhibit identical accuracy and confusion matrices. However, the XGBClassifier has a higher AUC_ROC score, indicating a slightly better capability in distinguishing between classes.

The metrics indicate overall correctness across both classes, translating to the model accurately predicting the outcome in approximately 91 out of 100 cases. For Class 0 (Non-Tackle), although it cannot be directly calculated from the provided data, the model shows a high number of true negatives (10,707) compared to false positives (2,763), suggesting good precision in correctly identifying non-tackle instances. The model achieves a perfect recall of 100%, indicating it correctly identified all instances of non-tackles without any false negatives.

For Class 1 (Tackles), the precision for this class is high. The absence of false negatives in the confusion matrix implies a high accuracy when the model predicts a tackle, although the exact value is not specified. While the specific value is not provided, the recall for tackles is suggested to be high, evidenced by a large number of true positives (16,326) and the absence of false negatives.

The model boasts a high AUC_ROC score of 93.60%. This value is crucial as it measures the model's ability to distinguish between the classes. An AUC_ROC score close to 1 is indicative of a model's strong predictive capability, and in this case, it highlights the model's robustness in differentiating between tackles and non-tackles.

In summary, the XGBClassifier demonstrates a high level of accuracy, complemented by excellent recall for both classes and good precision, particularly for non-tackles. Its high AUC_ROC score

further cements its status as a reliable and effective tool for classification tasks in this context.

The identical confusion matrices, especially the absence of false negatives, could signal issues like class imbalance or model overfitting. Also, given the small number of estimators and low learning rate, the models are relatively simple. While this might be suitable for the dataset, there is usually a balance to be struck between model simplicity and the complexity of patterns it can learn.

While both ensemble models demonstrate similar accuracy, the XGBClassifier shows a marginally superior ability to distinguish between classes. The identical performance across multiple metrics raises questions about the dataset and model training approach. It's important to investigate further to ensure that the models are effectively capturing the underlying patterns without overfitting or underfitting.

	Accuracy	AUC ROC
Light GBM	0.907	0.929

Light GBM accuracy metrics.

LightGBM: LightGBM, short for Light Gradient Boosting Machine, is an open-source, high-performance gradient boosting framework that employs decision tree algorithms. This tool is renowned among data scientists for its exceptional efficiency, scalability, and performance, particularly with large and complex datasets. Despite its strengths, LightGBM can overfit on smaller datasets and demands intricate hyperparameter tuning, a process that can be complex. Additionally, the interpretability of LightGBM models is lower compared to simpler models like linear regression or decision trees. Nonetheless, LightGBM achieves a commendable accuracy of 0.9069, making it competitive with other ensemble methods.

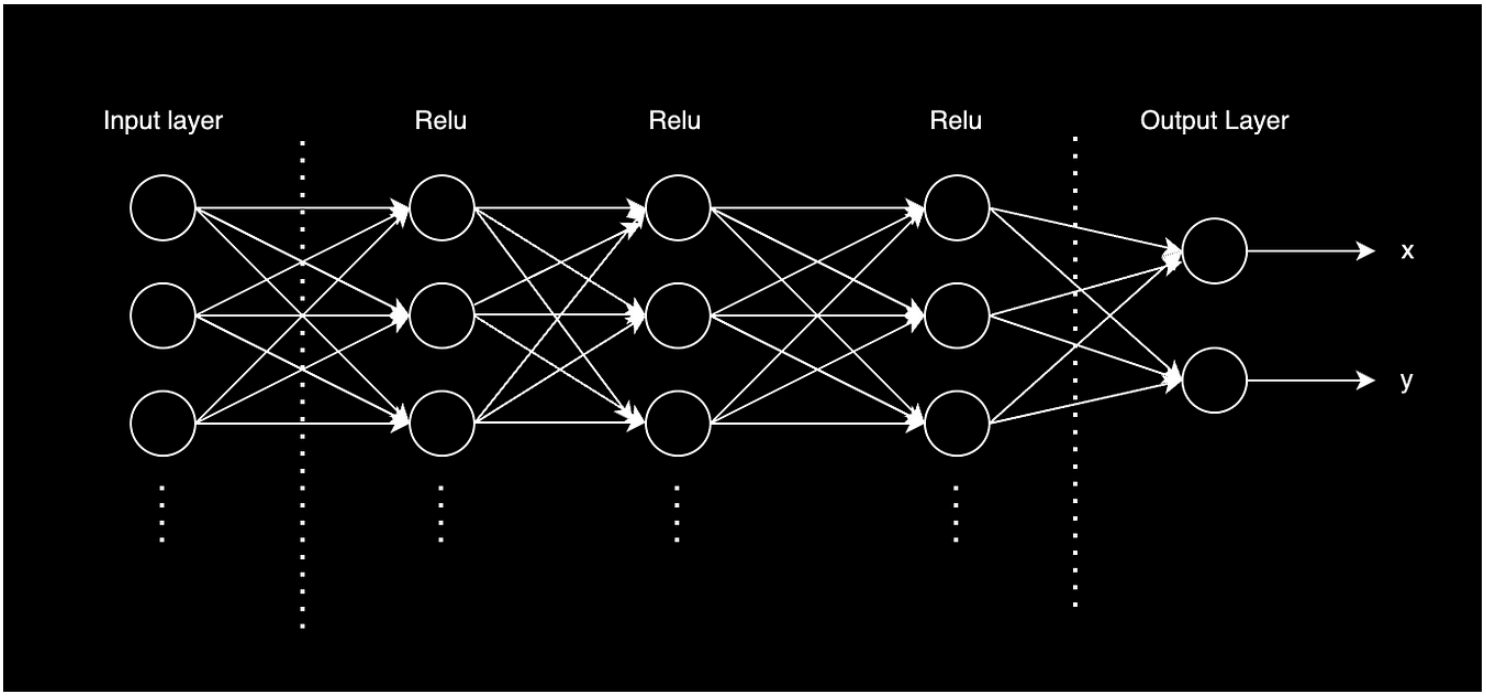
Where's the tackle?

Moving beyond the binary outcome, our second focus delves into the spatial dimensions of tackles. We seek to predict the precise location on the field where a tackle is likely to occur, capturing the intricate interplay of players and the strategic positioning of the ball. Utilizing advanced techniques such as spatial regression or convolutional neural networks (CNNs), our model aims to provide insights into the specific X and Y coordinates where tackles are more probable. This

spatial granularity contributes to a deeper tactical understanding, assisting teams in optimizing defensive formations.

Linear Regression Analysis: In our exploration, linear regression serves as an initial benchmark. The Mean Squared Error (MSE) for the linear regression model is calculated to be 545.45. While this provides a baseline understanding, we recognize the need for more sophisticated methodologies to capture the complexity inherent in spatial predictions.

Neural Network Implementation: Taking a leap into advanced methodologies, we employed a neural network tailored for spatial prediction. This neural network, consisting of three hidden layers, employs the Rectified Linear Unit (ReLU) activation function in each layer. The output layer, crucial for our spatial prediction task, is designed with two nodes to independently output the X and Y coordinates.



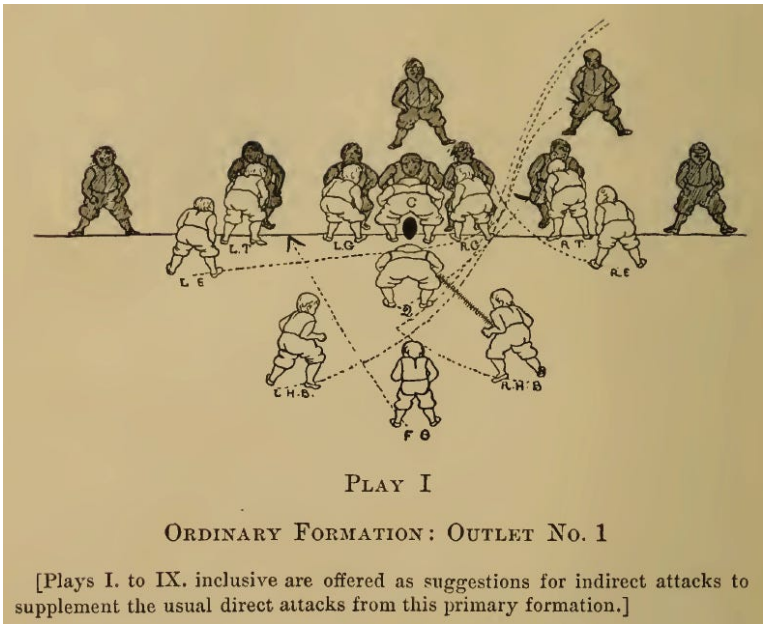
Architecture of the neural network.

Neural Network Performance: The neural network exhibits a significant improvement, achieving a substantially reduced MSE of 292.28. This enhancement underscores the ability of neural networks to discern intricate patterns within the spatial dimensions of tackles. The utilization of multiple hidden layers and the non-linear activation function enables the model to capture complex

relationships in the data, surpassing the capabilities of linear regression.

What about yardage?

The third facet of our predictive modeling addresses the yards gained or lost before a tackle is executed. This regression task involves quantifying the distance covered by the ball carrier before encountering a defensive tackle. By incorporating features such as player speed, offensive line performance, and historical play sequences, our model aims to provide an estimate of the yards traversed, enabling teams to assess defensive vulnerabilities and optimize offensive strategies for maximizing yardage.



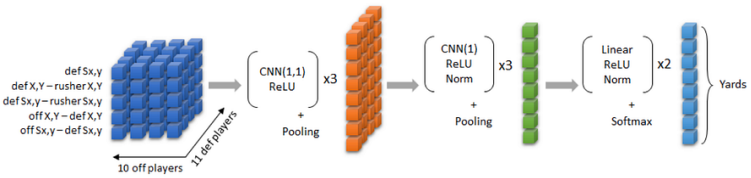
PLAY I

ORDINARY FORMATION: OUTLET NO. 1

[Plays I. to IX. inclusive are offered as suggestions for indirect attacks to supplement the usual direct attacks from this primary formation.]

Image Credit: Camp, Walter, and Lorin F. Deland, 'Football'.

Convolutional Neural Network: Convolutional Neural Networks (CNNs) are a class of deep learning models particularly effective in handling grid-like data, such as images or sequences. They have proven to be immensely successful in computer vision tasks, demonstrating an ability to learn hierarchical features and spatial relationships. CNNs are characterized by their use of convolutional layers, pooling layers, and fully connected layers.



CNN for predicting yards traveled. Image Credit: Kaggle.

```
def get_conv_net(num_classes_y):
    #_, x, y, z = train_x.shape
    inputdense_players = Input(shape=(11,10,10),

    X = Conv2D(128, kernel_size=(1,1), strides=(
    X = Conv2D(160, kernel_size=(1,1), strides=(
    X = Conv2D(128, kernel_size=(1,1), strides=(

    # The second block of convolutions learns th
    # For this reason the pool_size should be (1
    # (11, 1)
    Xmax = MaxPooling2D(pool_size=(1,10))(X)
    Xmax = Lambda(lambda x1 : x1*0.3)(Xmax)

    Xavg = AvgPool2D(pool_size=(1,10))(X)
    Xavg = Lambda(lambda x1 : x1*0.7)(Xavg)

    X = Add()([Xmax, Xavg])
    X = Lambda(lambda y : K.squeeze(y,2))(X)
    X = BatchNormalization()(X)

    X = Conv1D(160, kernel_size=1, strides=1, ac
    X = BatchNormalization()(X)
    X = Conv1D(96, kernel_size=1, strides=1, act
    X = BatchNormalization()(X)
    X = Conv1D(96, kernel_size=1, strides=1, act
    X = BatchNormalization()(X)

    Xmax = MaxPooling1D(pool_size=11)(X)
    Xmax = Lambda(lambda x1 : x1*0.3)(Xmax)

    Xavg = AvgPool1D(pool_size=11)(X)
    Xavg = Lambda(lambda x1 : x1*0.7)(Xavg)

    X = Add()([Xmax, Xavg])
    X = Lambda(lambda y : K.squeeze(y,1))(X)

    X = Dense(96, activation="relu")(X)
    X = BatchNormalization()(X)

    X = Dense(256, activation="relu")(X)
    X = LayerNormalization()(X)
    X = Dropout(0.3)(X)
```

```
        outsoft = Dense(num_classes_y, activation='s

        model = Model(inputs = [inputdense_players],
        return model
```

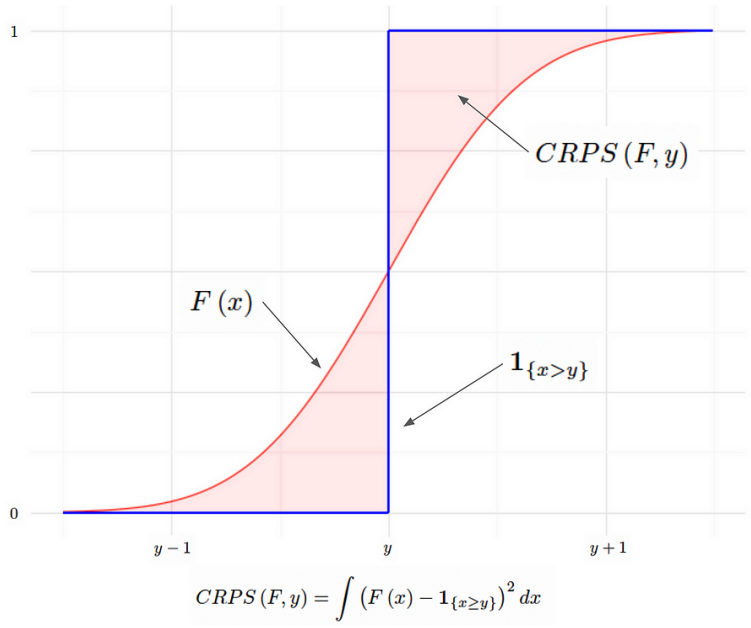
Predicting the outcome of a complex, fast-moving sport like football from raw player tracking data poses a unique machine learning challenge. This solution engineered the data and designed a CNN focused specifically on modeling the spatial relationships between players on a given rushing play.

The key preprocessing step was converting the x, y locations and speeds of all 22 players on the field into a tensor encoding interactions:

- The first dimension covers the 11 defensive players.
- The second dimension covers the 10 offensive skill position players.
- The 10 channels capture relative distances and speeds between players.

This aligns with the natural defense vs offense alignment and encapsulates how blockers and defenders work together or oppose the ball carrier. The output targets are the total yards gained discretized into bins.

The Continuous Ranked Probability Score (CRPS) loss function used in the prediction model provides strong training signal by comparing the full predicted probability distribution to the true outcome distribution. Unlike losses that just consider the accuracy of the argmax predicted class, CRPS integrates the squared error across all potential yard bins, rewarding models that assign high probabilities near the realized value and low probabilities further away. This tests the neural network's ability to reliably estimate the entire conditional distribution, not just the most likely bin. By directly optimizing well-calibrated probabilities during training, CRPS loss enables realistic uncertainty quantification and excellent performance at forecasting distributions in complex spatial prediction tasks like modeling football plays.



CRPS Scoring Function. Image Credit: Itamar Faran on Medium.

The first convolutional block in the model focuses on identifying key patterns of interaction between the offensive and defensive players. It picks up on movements, alignments, and reactions that influence the success of the play.

Pooling layers then condense the defensive side of the tensor into a summary vector. Further convolutional and dense layers combine to output a predicted probability distribution over the various yardage bins.

By structuring the data to take advantage of the spacial modeling capabilities of CNNs, the network could accurately forecast rushing yards without needing traditional formations or play-type features as inputs. The custom architecture provides an innovative template for learning from player tracking data.

The model demonstrates how thoughtfully engineering inputs tailored to deep learning algorithms unlocks considerable predictive signal. This homegrown spatial approach to turning raw football data into meaningful insights delivers results on par with world-class data science teams.

The reported mean validation loss of 0.0357 indicates a highly efficient convolutional neural network model for predicting rushing yards. A lower validation loss signals that the model generalizes well to unseen data, avoiding significant overfitting to the training examples.

The achieved score over 8-fold cross-validation implies the model can reliably predict the distribution of outcomes. Getting within a few yards on average is quite impressive given the spatial tracking data lacks traditional inputs like formations.

In summary, a mean CRPS in the 0.03–0.04 range enables quite granular yardage predictions. It demonstrates the network extracts meaningful signal from the raw tracking data without overfitting. The domain-specific architecture provides an efficient framework for football forecasting solely from player movements.

Results & Conclusion

Our work with the data and modeling suggests the following:

- Neural networks are effective at capturing the interactions between various features such as player positioning, ball movement, and team formations. Specifically, neural networks pick up on how these interactions influence likelihood and location of a tackle.
- Convolutional neural nets excel at analyzing spatial data. Our CNN effectively extracted and learned the spatial feature hierarchies between the data, identifying key patterns. It was most useful for accurately predicting yards gained in a play.
- Ultimately, the ensemble models outperformed the other models attempted, although overfitting may be at play.

Based on the predictive models we developed, given the proper data with available computational power, we can recommend utilizing ensemble methods to assist football coaches in optimizing resource allocation. By integrating insights from the ensemble models, coaches can gain a deeper understanding of team dynamics and performance, leading to more effective strategic planning for tackling in future games.

For future investigation into this data, we recommend using a distributed computing system so that all of the data could be utilized. We also think it may be interesting to filter further based on specific positions, player characteristics, and geographic or time conditions that could impact the gameplay. This is a very robust and complex dataset that could offer a lot of rich insights if the proper computing resources are available.

References

We leveraged several references for source code, including winning projects from previous NFL Big Data contests, to help us with this project.

NFL Next Gen Stats—Intro to Expected Rushing Yards

- Author: NFL.com
- Title: “Next Gen Stats: Intro to Expected Rushing Yards”
- Website: NFL.com
- URL: https://www.nfl.com/news/next-gen-stats-intro-to-expected-rushing-yards

Kaggle—Evaluating Tackle Difficulty

- Author: Kaggle (username: carrot1500)
- Title: “Evaluating Tackle Difficulty”
- Website: Kaggle
- URL: https://www.kaggle.com/code/carrot1500/evaluating-tackle-difficulty

GitHub—NFL Big Data Bowl 2020 Solution

- Author: Juan Camilo Campos
- Title: “1st Place Zoo Solution v2”
- Website: GitHub
- URL: https://github.com/juancamilocampos/nfl-big-data-bowl-2020/blob/master/1st_place_zoo_solution_v2.ipynb

GitHub—NFL Big Data Bowl 2021 Solution

- Author: Klordi
- Title: “Lordi NFL Big Data Bowl 2021 Solution”
- Website: GitHub
- URL: https://github.com/klordi/lordi_nfl_big_data_bowl_2021/tree/main

GitHub—NFL Big Data Bowl 2022 Solution

- Author: Sean Sullivan
- Title: “NFL Big Data Bowl 2022”
- Website: GitHub
- URL: https://github.com/seanwsullivan1/NFL-Big-Data-Bowl-2022