HIGH SCHOOL EDITION
Journal of Student Research
Volume 13 Issue 3 (2024)

Predictive Football Analysis

Anirudh Gadepalli

Cherry Creek High School

ABSTRACT

In the fast-paced landscape of sports, technology, and algorithms are becoming pivotal forces, creating a new era of performance and allowing athletes to rise to higher levels. Offenses and defenses are constantly evolving and transforming, and with the help of technology, teams are becoming better than ever, and coaching is trickier than ever. I present two algorithms to approach coaching sports. In the current football arena, accurately predicting the next play has been a longstanding challenge. Therefore, I created an optimized Random Forest Model (RFM) to anticipate what play a team might run, enabling coaches to strategize and teams to create better defenses. I also developed a second network using Deep Double Q-Learning (DDQN) to simulate an offense that a coach would call for his players.

Introduction

Football is very challenging regarding the physicality the players deal with and the crafting of strategies and management of entire teams that coaches contend with. Players are demanded of high physical strength, speed, and endurance and are tasked with making split-second decisions that can change the course of their games or seasons. On the coaching front, the challenges and complexities are similar to those of a high-stakes chess match, where coaches must anticipate moves, decipher the game's flow, and motivate their team. Football coaches navigate multiple challenges and manage diverse responsibilities ranging from strategic planning to injury reports. In a sport where success depends on many factors, coaches must be provided with the resources to deal with their problems properly. Technological advancements in strategic analysis, player performance tracking, and injury prevention have significantly enhanced the football landscape. I begin by reviewing current tech approaches before proposing a new solution that solves the problem of strategic analysis.

Current Models

Current state-of-the-art forecast systems for football are only used in the National Football League (NFL) and are not available commercially. This is the partnership between Amazon Web Services (AWS) and Next Gen Stats (NGS), which uses machine learning to provide cutting-edge real-time data on players' location, speed, and acceleration [1]. AWS NGS enables models to give statistics on metrics that are otherwise difficult to predict. This includes catch and pressure-probability, which enables high-quality, unique statistics to be taken, allowing analysts and teams to evaluate their players and performance and extract meaningful insights from the game.

Figure 1. NGS powered by NFL and AWS

For example, NGS's Pressure Probability model takes pass-rushing stats beyond the box score. Traditionally, pass-rush statistics have been limited to manual charting processes that can only be done at the play level [2]. However, Pressure Probability uses graph neural networks to treat each player as a node, capture their spatial relationships to identify rushers and blockers, and estimate pressure score and movement. Similarly, the NFL employs multiple models to take the analysis to the next level. This is one example of a solution in the current world.

Uniqueness

Although NGS offers player tracking and real-time statistics, it doesn't commercially provide the ability to predict upcoming plays and supply a team with the opportunity to formulate a strong defense. More importantly, NGS provides player tracking solely for NFL players with RFID chips ingrained in their helmets. This allows for models to work with very precise, specific data on every player on the field. This, however, isn't applicable in other levels of play. For example, high school athletes cannot wear technology on the field. Therefore, I aim to address this problem by developing machine learning models.

Methods

The objective of this research is to develop two models that would predict the next play based on the current state of the game. This includes the position of the ball on the field, the time remaining on the game clock, and the game's score. These factors prompt teams to approach the game with different tactics and formations, either running or throwing the ball. The goal is to consider all the factors and variables and speculate the most likely outcome the opposing team will execute. Note this is one of the primary reasons for the rise of machine learning, as the human mind simply cannot consider this multitude of variables. For this to work, I gathered a thorough data set that included multiple relevant variables, allowing the model to make the accurate predictions required for coaches to instruct a team properly.

To address these challenges, I developed two machine-learning models to help coaches manage their teams. My first objective was to create a model to predict, given information about the current state of the game (such as the position of the ball on the field, the time remaining on the game clock, and the score), what play the opposing team is most likely to execute (such as running vs. passing the ball). My second objective was to develop a reinforcement learning (RL) model that can generate its next play call by itself, taking into account the current state of the game (as in the first objective) and historical data on the effectiveness of each play in each situation to predict which play type is most likely to help the team gain yards and score points at any given time. The goal of this model is to provide the most optimal play to run and other insightful statistics that can help advance the offense.

It is important to note that both models rely on the same type of training data: historical play-by-play data from past football games indicating, for each play, what the current state of the game was, what play got called, and what resulted from the play (e.g., the number of yards and points gained).

Dataset Used for Both Objectives

I selected the NFL Play by Play Data 2009-2018 to train both models [3]. While play-by-play data is available via open-source software, it's not freely available for research. Therefore, a group of statistical researchers at Carnegie Mellon University developed a tool called nflscrapR, an R-package used to output clean datasets [4]. This dataset comprises 255 columns (features) and almost 400,000 rows. Since this dataset contains NFL historical data from 2009 to 2018, it provides a decade of high-quality, reliable data from the most credible source of football in the entire world. It can be assured that this data is accurate and the best representation for coaches as they will be using robust networks trained on data from the highest level of play.

Exploratory Data Analysis

During the data exploration stage, I used the Pandas Profiling Report to generate insightful visualization and statistics of my data. This helped me identify patterns, make critical observations, and understand what data I was working with. For example, it can be seen in the heatmap in Fig [5] that the strongest correlation of variables is between the yards gained from a play and whether or not it happened to be a first down pass. This is logical, as if a down results in a significant gain in yards, it is expected to become a first down.

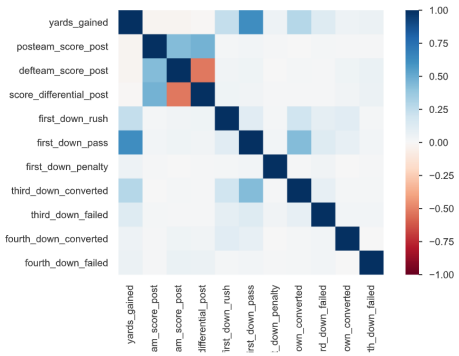

Figure 5. Heatmap of specific features

Data Engineering

For both models, I extracted the relevant features contained in the dataset to determine the current state of the game. This consisted of: [‘total_away_score’, ‘posteam_score’, ‘defteam_score’, ‘score_differential’, ‘yardline_100’, ‘down’, ‘goal_to_go’, ‘ydstogo’, ‘ydsnet’, ‘quarter_seconds_remaining’, ‘half_seconds_remaining’, ‘game_seconds_remaining’, ‘quarter_end’]. For the next-play-prediction model, I used the play type as the target variable, which took on one of 10 play types: pass, run, qb spike, qb kneel, field goal, kickoff, extra-point, and the end of play. Note that “end of play” includes sacks, scrambles, and stoppage-of-plays like timeouts and penalties. For the reinforcement learning play-caller model, I measured the outcome of each play with a custom formula taking into account the number of yards and points gained from calling that play:
f(reward) = yard_reward + score_differential_reward + first_down_conversion_reward + did_score

yard_reward = rewards[‘yards_gained’]
score_differential_reward =
{0.1, rewards[‘score_differential’] > 0
{0, otherwise
first_down_conv_reward = 7.5 * (rewards[‘first_down_rush’] + rewards[‘first_down_pass’])
This formula calculated the reward for every play, which would be later used to train the model.

Next-Play-Prediction Model

To complete the feature extraction for the classification, I needed a robust and scalable model for prediction. Since there were no current base models in this field to tweak due to the uniqueness of this project, I decided to make my model from scratch and not simply tune another model. This allowed for efficient testing of my models run on powerful hardware, allowing me to train these models from scratch and giving me the flexibility to adjust the models for their specific tasks. However, because I was creating these new models, I needed to fine-tune the parameters and increase the precision of the training. This prompted me to use hyperparameter tuning through the GridSearchCV class provided by scikit-learn. Hyperparameter tuning involves finding the best parameters for a model to maximize its performance and accuracy. Since I have capable hardware for evaluating the model, I used GridSearchCV to find its optimized parameters.

In my case, I needed a model that could handle a large amount of data being processed and accurately forecast the corresponding play type for every play. Because of this, I chose to work with a Random Forest Model (RFM). RFMs work by ensembling multiple decision trees and averaging the data to make very low-variance predictions. RFMs are highly accurate and scalable and reduce overfitting, therefore making them perfect for this prediction task [6].

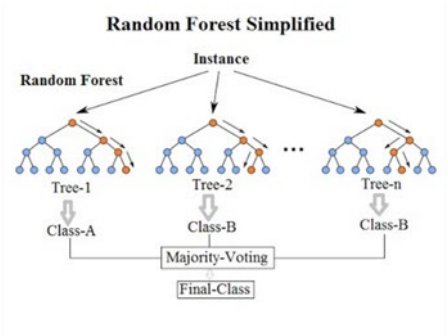

Figure 7. RFM architecture

Next, the RFM needed to be trained on the data. As mentioned before, the dataset being worked with is very large. Much of the data in the original dataset falls into two categories: unnecessary or unusable. Examples include statistics that rarely are caught, such as penalties, challenges, or replays, which might occur once or twice a game but aren’t important for the prediction of the next play. Even if they can provide insight, the rarity of these memorable plays causes them to be unimportant and dropped from the included data. Our dataset also contained many other types of less valuable information, such as the players, which was dropped for simplicity. The rest of the data, for example, the play type or the results, were taken in as a string. However, this data must be an integer, as that is the only way an RFM can take in the data. For this, the data was OneHotEncoded to convert the data into integer format properly. Then, the RFM was created using GridSearchCV to optimize its hyperparameters.

Finally, the RFM needed to be evaluated on its performance and accuracy to predict the upcoming play. Therefore, nested cross-validation was implemented to determine the validity of the results. Nested cross-validation involves an outer loop that iteratively splits the data into training and validation sets. Then, an inner loop further divides the training set into subsets for training and hyperparameter tuning. This method allows the assessment of the model and how it generalizes to new, unseen data.

Play-Caller Model

The play-calling objective of this project required a deep understanding of RL techniques. The main components of RL include a policy, reward, state, and action. Generally, RL works when an agent acts as the environment, resulting in a new state for the agent and the environment either rewards or punishes the agent based on the policy applied. This can be seen in Fig [8].

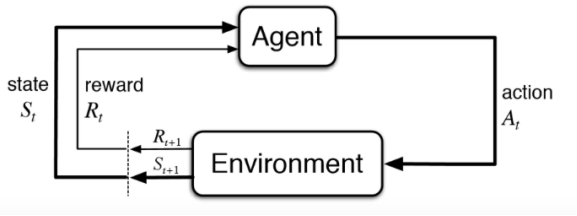

Figure 8. Policy-based RL

Q-learning is an RL algorithm designed to enable a policy for decision-making in an environment. Traditionally, the agent maintains a Q-table, with Q-values representing the reward for taking a specific action. The agent then targets to maximize the Q-values. Due to the limitations of a Q-table in environments with massive amounts of entries and the inaccurate estimation of the Q-values, neural networks are employed to predict them.

Furthermore, this nonlinear nature of neural networks is crucial for handling complex environmental relationships. Therefore, I will use a neural network approach to this task. Because of the number of dimensions our data has and the complex factors going into it, I decided to use a Double Deep Q-learning network (DDQN) to approximate the Q-function for the reward. DDQNs were introduced by Google DeepMind [9] to address the overestimation in a standard DQN, leading to suboptimal policies and slow convergence. DDQNs mitigate this issue with two separate neural networks: one for the action selection and one for the action evaluation. These are known as the target and online network. The target network periodically copies the weights from the online networking, effectively reducing the overestimation.

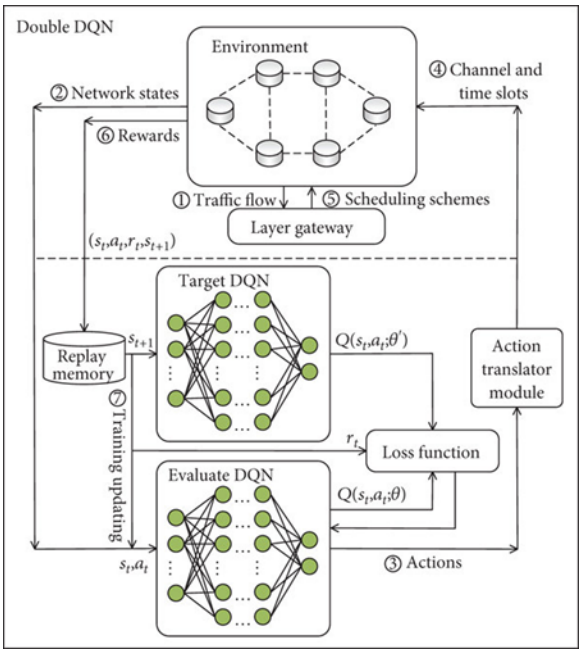

Figure 10. DDQN model architecture

DQNs are specifically shaped through a famous dynamic programming equation called the Bellman Equation. When we need to calculate the loss of the model, we need to take the difference between the Q-target and the current Q-value. This can be modeled with the equation shown in Fig [11].

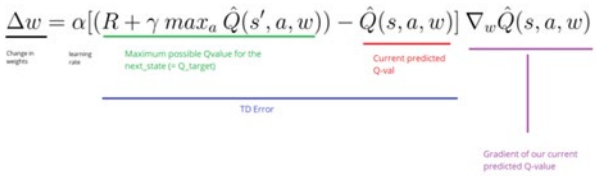

Figure 11. How to calculate loss

However, the Q-target value is unknown; it needs to be estimated. From here, the Bellman Equation comes into play, as the Q-target can be predicted, as shown in Fig [12].

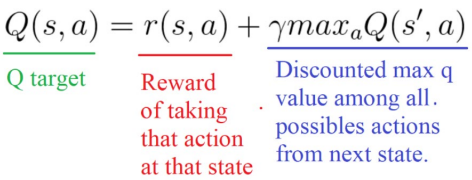

Figure 12. Bellman Equation

The Q-target value is predicted using neural networks. Multiple model architectures were used for the target network, with 2 examples below.

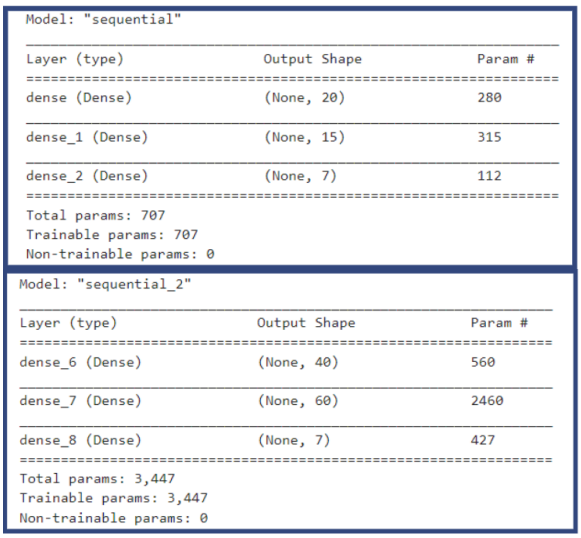

Figure 13. 2 trial target NN architectures

These neural networks and the DDQN model were trained using Pytorch's neural network library. The data was split into training batches, and using mean squared error loss (MSE) and the Adam optimizer, the model was trained by trialing multiple different epochs.

Results

Prediction

Initially, the play-prediction model achieved relatively high accuracy. On seen data, the play prediction had an accuracy of ~99.97%. Of course, this was on training data, so while it doesn't represent the model's factual accuracy, this value must be extremely high to show the model is learning from the data. On unseen data, the play-prediction model had an accuracy of ~85.21%. This accuracy was found through cross-validation by splitting the data into seven folds, iterating through, and creating new training-validation pairs. The average accuracy of each fold was taken, giving that result. Note this model was HP-tuned using GridSearchCV.

Play Calling

The output the DDQN provided was a vector of each of the 7 Q-Values of each play type for every single play. These Q-values were numerical values that represented the predicted result of each play. To simplify this result, the softmax of every vector was taken. This provided a list of percentages, which could be thought of as the likeliness of the model to select that play, which means that the highest percent value would be chosen. To find the accuracy of the DDQN, this largest value was compared to what the coach called in that situation. The DDQN called accurately what the coach called only 4% of the time. After analyzing this result, it was found that the field goal column was called over 350k+ thousand times out of the roughly 400k total plays. This means that the play-caller model was significantly biased towards field goals, as shown in Fig [14].

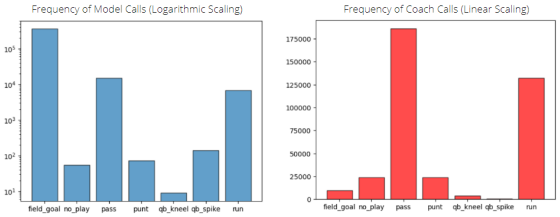

Figure 14. Frequency of Model Calls vs. Coach Calls

A random sample example that the model called is depicted below, with the pie chart representing the percentages of the model's confidence in each play.

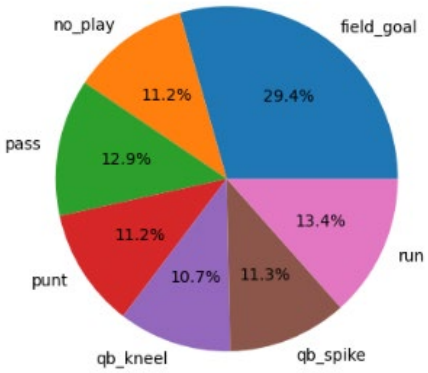

Figure 15. Pie chart of confidences of sample play

Discussion

Next-Play-Prediction Model

The outcomes from the RFM prediction model show that artificial intelligence can forecast the next play called in a football game. This reveals a strong correlation between the state and historical data of a football game with the play a coach would call. Practically speaking, it also suggests that models such as the one I have developed can help coaches predict which plays will be called by their opponents in a particular situation. This can help the coach adjust their strategy for that particular play type, for example, by adopting a defensive strategy better suited to running plays in cases where the model predicts that the opposing team will likely rush. While this model demonstrates promising results, these could be improved through more extensive hyperparameter tuning and training. Nevertheless, a larger dataset would not likely enhance the model as the current dataset used contained a surplus of information collected from over a decade of NFL games. As mentioned before, the dataset used contained almost 500,000 rows with over 250 features. Although some information was dropped due to compatibility, it is worth noting the model had plenty of data to work with. It's also important to recognize that at a certain point, increasing the volume of data may not lead to significant improvements in a model's performance and will gradually plateau.

Football coaching is already a progressively challenging process with large rosters, susceptibility to injury, and practical strategies from opponents. However, now, with AI, it is possible to simplify this process and aid coaches in this challenging field.

Play-Caller Model

The results of the DDQN reveal that artificial intelligence can simulate an offensive coordinator or a head coach. Using a reward-based neural network with reinforcement learning can simulate an offensive coordinator or the head coach calling plays for a team. This model, however, needs much tweaking and improvement. We can see from the initial results the model is extremely biased toward field goals. This is caused by the vast data imbalance in the NFL on field goals. The extremely high field goal completion rate in the NFL is causing the model to believe that any player can hit a field goal from any spot on the field. Because of the more intelligent strategy in the NFL, teams don't have players kicking field goals from far out, which increases the completion rate. Furthermore, on top of this, there is a tremendous skill gap between the NFL and a high school or college kicker. Therefore, this model needs to be tweaked to fix this. The first step that can be done to do this is to drop many of the field goals completed. This will balance out data and force the model to believe that field goals are not as likely as it thought. Secondly, the reward function can be tweaked. Every time the player misses a field goal, the model will be severely punished. This can also be done when every time a player hits a field goal, the model is rewarded much less than previously. These tweaks are minute but can majorly change performance.

As mentioned before, coaching football is already challenging progress. With AI, it is now possible to aid coaches in this challenging field.

Conclusion

It's now possible for familiar football coaches to coach at a much higher level with the help of artificial intelligence. The results show a correlation between the state and the upcoming play. This justifies future research into creating more AI-powered tools to assist the 30K+ football teams, coaches, and millions of players nationwide. It should be important to note that this technology will **not** replace the coaching aspect of football. Instead, this should be used as a tool that would assist a coach in a game or the film room. By no means should this replace an athletics instructor, as any coach provides valuable insight that no technology can replicate. In the circumstances in which the model poorly predicts the opposing team, a coach's intuition is vital to hold the model accountable. Not only on the strategic aspect, but coaches are also critical in understanding player injuries, team chemistry, and player skill capacities, which, along with this model, can greatly be taken further than simply deepening teams' strategies. The teams who are in need in underprivileged communities who have an inherent disadvantage can use this to level the playing field. According to a study done in Texas by Julie Chang, aided by USA Today, wealthy schools with low poverty rates have a significantly higher win percentage than those schools with higher poverty rates. Reports have shown that these schools lack essential equipment, their helmets and pads are decades old, and their players barely eat anything before the games. Only 23% of low-income schools have won a state football championship in two decades, while the large high schools in Texas with the lowest poverty rate have won more than 59% [16]. The disparity in resources affects the outcomes on the field. Hence, these underprivileged schools and students must get the help they need. Since there is an obvious need for resources for these teams, there is a major opportunity for artificial intelligence to enter the sports industry, specifically traditional American football. This has the potential to save students time and money, and I look forward to seeing where future research leads.

Acknowledgments

I would like to thank my advisor for the valuable insight provided to me on this topic.

References

*Any images that are not listed were created by Anirudh Gadepalli, myself.
[1] A. Entin, “Tackling Next Gen Stats: How AWS is using AI to advance sports analytics with the NFL,” AWS, Sep. 20, 2023.
[2] “NFL NEXT GEN STATS,” Football Operations.
[3] M. Horowitz “NFL Play by Play Data 2009-2018”, Kaggle, Mar. 2019
[4] S. Carl, “An R package to quickly obtain clean and tidy NFL play by play data,” www.nflfastr.com. https://www.nflfastr.com/
[6] S. R, “Understand Random Forest Algorithms With Examples (Updated 2024),” Analytics Vidhya, Dec. 21, 2023.
[7] W. Koehrsen, “Random Forest Simple Explanation,” Medium, Aug. 18, 2020. https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d
[8] K. Chilamkurthy, “Off-policy vs On-Policy vs Offline Reinforcement Learning Demystified!,” Medium, Nov. 06, 2020
[9] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-Learning”, AAAI, vol. 30, no. 1, Mar. 2016.
[10] Wu, G. Zhang, J. Nie, Y. Peng, and Y. Zhang, "Deep Reinforcement Learning for Scheduling in an Edge Computing-Based Industrial Internet of Things," Wireless Communications and Mobile Computing, 2021. DOI: 10.1155/2021/8017334.
[11] T. Simonini, “Improvements in Deep Q Learning: Dueling Double DQN, Prioritized Experience Replay, and fixed,” freeCodeCamp, Jul. 06, 2018.
[12] T. Simonini, “Improvements in Deep Q Learning: Dueling Double DQN, Prioritized Experience Replay, and fixed,” freeCodeCamp, Jul. 06, 2018.
[16] J. Chang, “WHY HIGH-POVERTY SCHOOLS LOSE MORE FOOTBALL GAMES,” USA Today, Oct. 04, 2019.