Machine Learning for NFL Pass Play Predictions

Andrew Tweedell[1]

[1]Khoury College of Computer Science
Northeastern University, Boston MA

Abstract

The purpose of this project was to build and evaluate models that could use high resolution NFL football player tracking data that could predict which receiver on the field the quarterback would throw to based on the play spatial and global characteristic at 0.5 second before the ball was thrown. We used a mix of traditional and deep learning approaches, as well as attempted to apply novel graph based methods to the problem. Regardless of methods used and despite the relatively balanced class outcomes, we were not able to achieve the predictive capabilities of previous methods. Our Multi Layer Perceptron performed the best with a test accuracy of 53.5% compared to the previous records of 59.8%. Despite their expressive capacity, the graph methods performed much worse, with the hypothesis that graph level classifications have poorly inductive capabilities with such highly dynamic data sets. Further areas of research are suggested.

Introduction

In 2020, the National Football League (NFL) started releasing all their game play data to the data community to help generate actionable insights about American football (https://www.kaggle.com/competitions/nfl-big-data-bowl-2021). One particular analytical capability coaches, fans, and betting companies are keen to understand is the prediction of targeted receivers in the game. This is likely due to the out sized impact that passing plays have on game outcomes. Thus, the purpose of this project is to develop and evaluate models to predict intended offensive player target during pass plays using traditional, deep and graph learning methods. The projects seeks a data-driven approach to answer "Given the picture presented to a quarterback, who would he choose to throw the ball to?" We will use the time point 0.5 seconds prior to the release of the football by the quarterback to build a data set off relevant spatial relations. Previous research typically uses the moment of ball release to predict targets; however, certain features like quarterback orientation are highly correlated and could cause misleading performance.

Game, play, player, and positional features derived from NFL Next-Gen Stats player tracking data were leveraged for this project. We assess the accuracy of a model that tries to predicted offensive player the pass will be thrown to. This work will contribute to the growth of vector and graph based methodologies for use in the sports world. It will provide insights into specific player and game characteristics that should be accounted for when predicting sport outcomes. This paper is divided into 5 sections; 1) Background, 2) Related Work, 3) Project Description, 4) Empirical Results, and 5) Discussion and Conclusions.

Background

Through the Big Data Bowl, the NFL started releasing all their game play data to the data community. This includes player tracking data which has since spurred many novel applications. More recently, this has lead to many data scientists to try to predict outcomes of passing plays because of their out sized impact on game outcomes. Understanding the complex factors that go into what makes a certain play successful could be extremely useful to those that evaluate players, or develop offensive and defensive strategies. Estimating the probability of an offensive player to successfully catch a ball thrown to them is a major focus for many groups. However, estimating completion probability is post-hoc in nature and not truly predictive. They assume the target is known already and often time include data up to the moment of catch. To create an accurate statistical depiction of the probability of completion, you also need to know what the probability of the quarterback throwing to that particular receiver is. Pompeu da Silva (2022) and Burke (2019) are a few of the only authors that actually take probability of passing to a particular player into account for completion probability.

Moreover, many models only look at game states (snapshots of a particular play) and not the relationship between players or entities on the field. Vector representations of individual play data is limited in its flexibility or expressiveness when it comes to relationships between players or between features of the game play. Very interesting graph methodologies have been developed recently that could harness the expressiveness that these data structures have. Figure 1 is taken from Xenopoulos and Silva (2021) and gives a visual depiction of the differences in how sports data might be structured in vector form versus graph form. As can be seen, graphs allow for a non-Euclidean data structure that more accurately represents player-player interactions during team sports. Also graphs offer more expressiveness based

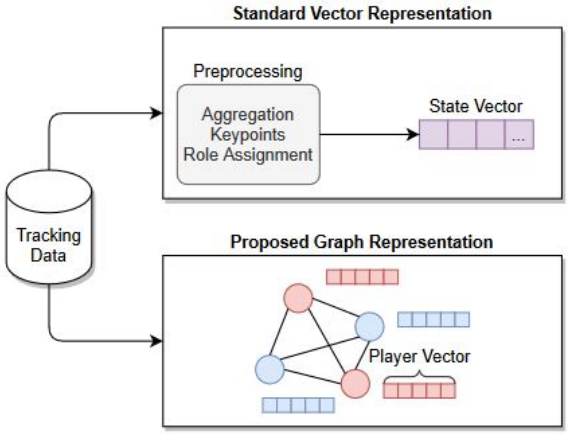

Figure 1: Vector versus graph data structure for team sports.

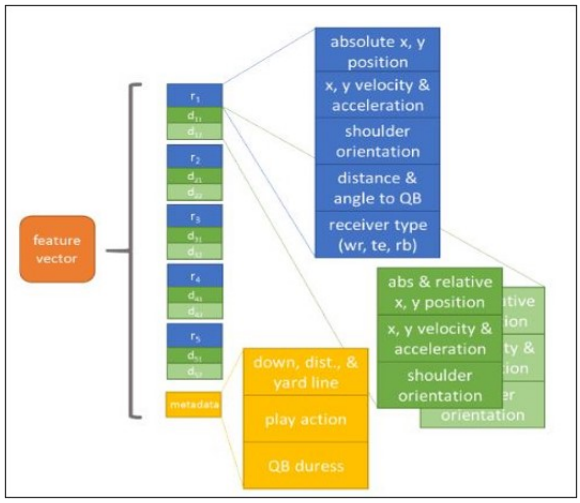

Figure 2: DeepQB Vector Data Structure.

on connected-ness between and within hierarchical features, which is how team sports data is usually structured. Nodes in graphs are usually depicted as individual players or entities, like the ball. Edges between nodes constitute their relationship to each other, be it spatial or semantic. Many multi-agent problems (robotics, team performance) use distance-derived features to construct game state graphs. If spatio-temporal data is provided, then these methods may offer a unique solution to predicting who a quarterback will throw to, given the players, their location, and their features.

Related Work

Using machine learning algorithms to predict football play outcomes is nothing particularly new. For instance, Horton (2020) used set-based learning on player trajectory to predict catch completion probabilities (Acc 73.1%). Deshpande and Evans (2019) developed a Bayesian non-parametric probabilistic model to estimate completion probabilities (MSE 0.09). Pompeu da Silva (2022) found a Random Forest Model to estimate pass completion probability the best compared to logistic regression, linear discriminant analysis, and others (AUC 0.88).

Burke (2019) uses Feed Forward Neural Network to estimate the probability of the outcomes on a pass play: complete, incomplete, or interception (Acc 74%). Of note, however, is that their neural network architecture also allowed for the estimation of the probability for each target to be thrown to. Their 'DeepQB' model vectorized geospatial characteristics of all players on the field. Ordering of inputs, i.e. where each receiver's information lies in the vector was given meaning and stayed consistent (See Figure 2.). The eligible receiving targets were labels 0 through 5 from shallowest to deepest on the field. The model is to predict that label from each play corresponding to which receiver was thrown to. They were able to get 59.8% accuracy from their models and as such we will emulate their methods in this project. However, Burke (2019) used the data up to the moment of the throw, which may make the problem easier as some features (quarterback shoulder orientation) may be highly correlated with the outcomes. Interestingly, Xenopoulos and Silva (2021) applied their methods of graph neural networks to predict expected yards gained from NFL running backs. Their graph regression based methods were able to outperform standard vector methodologies. Using graph based methods, nodes features such as geospatial location on the field, movement characteristics, and other demographics are convolved across neighboring nodes between their edges connecting them. The edge weights can be prescribed, as in the case of graph convolutional networks (GCN), where the authors used the inverse Euclidean distance. The edge weights can also be learned mathetmatically with an attention based network, the Graph Attention Network (GAT). Regardless, graph methodologies for graph level predictions (either classifying predicted receivers or regressing for rushing yards gained), a pooling layer is used at the end to aggregate node information together in order for a finally linear layer to perform the prediction. Although not passing plays, their approaches may be suitable for passing plays as well. A schematic of their general network flow for the Graph Attention Network is seen in Figure 3. The same workflow is taken for the Graph Convolutional Network except the attention weights ($e_{i,j}$) are not calculated but provided as a part of the model. This approach will be replicated for our current problem of predicting intended receiving targets.

Project Description

This section will describe the NFL data used in the analysis, as well as detail methods for exploratory data analysis. There were two different types of methods used in this project, vector-based machine learning (logistic regression, random forest classification, and multi-layer perceptrons), and graph-based machine learning (graph convolutional network, graph attention network, etc). Prior to any model training, data was split into training (80%) and testing (20%) sets to compare performance of all models.

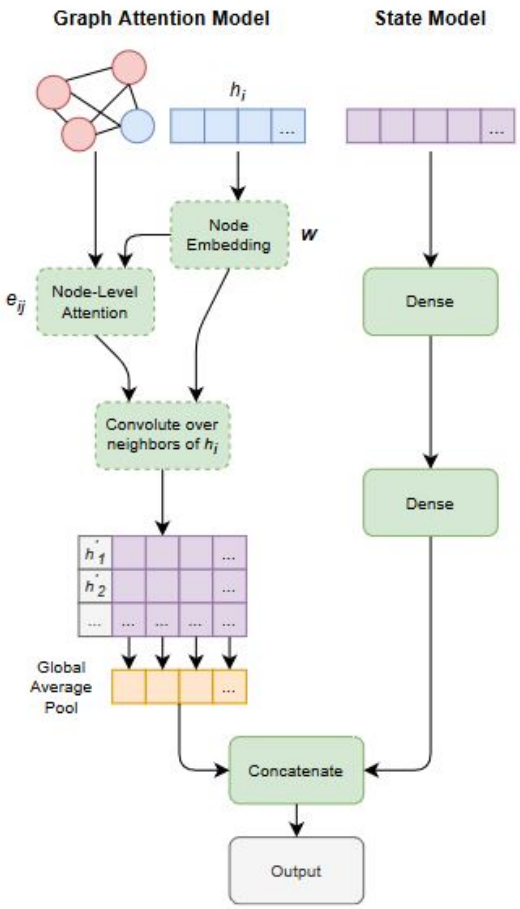

Figure 3: Graph Attention Network data flow and architecture.

Data

The data set that will be used for this project comes from the NFL Next Gen Stats player tracking data set and was released as a part of the 2021 NFL Big Data Bowl Kaggle Competition where every pass play from the 2018 NFL season is provided. The data set contains more than 19,000 pass plays where and event log will be transformed into the target label for which offensive player was targeted. The full data set contains hierarchical features sets. First, meta data for each game is provided. This includes features like the city the game was played, the weather, day, and time of day. Next, meta data was provided for each pass play within each game, with features like offense formation, what down, and what yard the play was on. Individual player data was also included with demographic, evaluation, and other metrics describing each player. Finally and most importantly, player tracking data is provided that includes moment to moment spatial data during the duration of the play, including where the player is on the field and what direction they are moving in. Event tags (e.g. ball snap, catch completed) are recorded along with the tracking data.

For the current analysis, each data point consists of a single play and the independent variables encompassed all the player, game, and field conditions for that play. The dependent variable is the specific receiver that the quarterback through the ball to on that play. For the exploratory data analysis and vector-based methods, this outcome variable corresponded to the order of the receivers based on their distance from the line of scrimmage, with the shallowest receiver getting a 0 designation and the deepest getting a 4 designation. This approach was replicated from Burke (2019), whose paper noted that strict ordering was needed to actualize stable results and predictions.

Thus, we have a multi-class classification problem where we are attempting to classify the receiver based on the characteristics of the play. All offensive players that can potentially receive the football from the quarterback are considered for the model. These players can be running backs (RB), tight ends (TE), full backs (FB), half backs (HB), or wide receivers (WR). For brevity through the rest of this paper, all of these positions will be referred to as 'receivers' or 'targeted receivers'.

A caveat to this data set, however, is that it does not include any data on the offensive or defensive linemen from the field. While they might not be directly involved in passing plays, their presence could have effects on who the quarterback can see or move to, thus influencing our models. Future data sets that include all offensive and defensive players on the field will likely yield better analyses.

Feature Engineering and Exploratory Data Analysis

For each play, different raw and hand crafted features were combined into a single vector form for ingestion into a logistic regression, random forest classification, and multilayer perceptron. The non-exhausted list of features included quarterback location, speed, and orientation, each receivers' location, speed, and orientation, the distances between each receiver and defenders near them, the types of formations the offense and defense lined up at, the week of the season, quarter of the player, down of the play, time left on the clock, score differential, and other. In total, there were 218 features used. Prior to classification, exploratory data analysis was performed wherein different variables were visualized to ensure the data aligned with our understanding of the rules of football. The data only consisted of NFL pass plays; however, we wanted to remove any abnormal data points resulting from non-traditional plays. These types of plays are edge cases where player positions might take actions contrary to their typical roles or when there are an irregular number of position types on the field. We also removed any play that did not result in the quarterback throwing the ball, such as during a quarterback sack. However, in-completions were kept in the data set because our interest is not in predicting catch completion but just predicting who the ball will be thrown to, which is not affected by in-completions. In total, 16,562 total pass plays were analyzed. First, as depicted in Figure 4, we plotted the frequency of position types that appeared in our data set. Because the data set focuses on passing plays, we confirmed that we indeed see high frequencies of wide receivers (WR) and corner backs (CB) because of

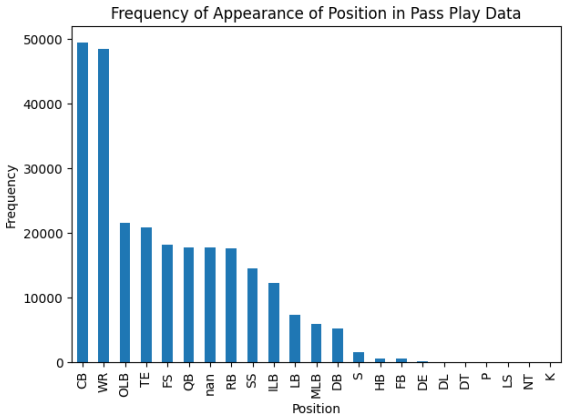

Figure 4: Player position in play frequency.

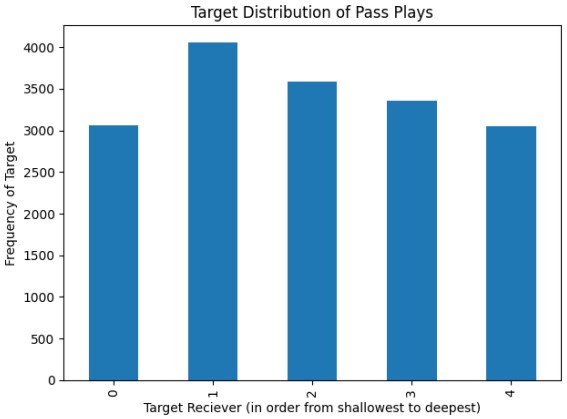

Figure 5: Class Distribution.

their specific role on passing plays. In addition, defensive positions are also included in the figure. Next, to check class imbalance, we calculate and plot the distribution of classes, or targeted receivers, within the data. As can be seen by Figure 5, there is only a slight imbalance with Target Receiver '1' or the second shallowest receiver, having approximately 1,000 more samples than the lowest class, the '0' or shallowest Target Receiver. Given the larger sample size of the data and the mild imbalance, only a stratified train-test split approach was taken to mitigate its effects. Additionally, the spatial density of positions types on the field during plays was played as a 2 dimensional histogram. For example, the difference in spatial density we see between Figure 6 and Figure 7 confirms our intuition about the roles of different positions in the game. Quarterbacks on passing plays (Figure 6.) tend to stay in the pocket (center of the field); while the wide receivers (Figure 7.) stay toward the outside edges of the field, as their name suggests.

Generalized Additive Models

As a part of the EDA to better understand the non-linear relationships between variables and the influence they have on pass outcomes, we fit various generalized additive models.

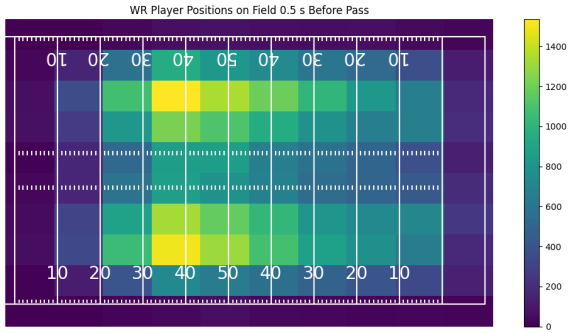

Figure 6: Player position specific spatial density on field.

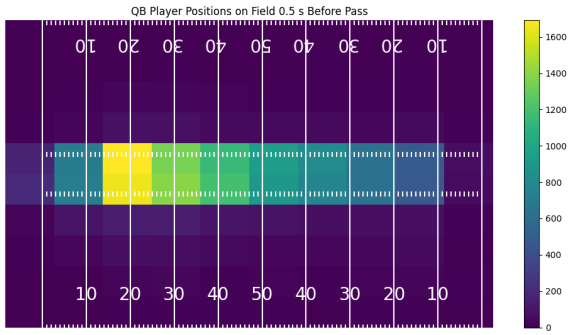

Figure 7: Player position specific spatial density on field..

Briefly, after choosing 1-3 features, we trained five models, in a one versus rest approach with one model per class with binary predictions. Class imbalance makes models unstable and not useful for inference, but these models can be useful to uncover trends in the data. For example, to understand the influence of what down it is on who the quarterback will throw to, we can model and compare the between the classes as seen in Figure 8. As can be seen from the figure on the right, when the plays are only at the first or second down, the unnecessary risk of throwing to a deep receiver is rarely taken by quarterbacks. Thus we see an overall lower log-odds for the deepest receivers until we get to the fourth down, where we see the log-odds increase as may be perceived risk is deemed worth it. On the left side, we note the opposite trend for the shallowest receivers. During the first or second down, they are seen as the safer choice, and thus we see them have higher log-odds during those plays. While it is not possible to show and discuss all relationships within the current data set, this highlights an important tool available for sports analytics. For instance, one could model and produce risk profiles for each quarterbacks on who they are willing to thrown to based on the situation (down, quarter, distance to go, etc).

Vector Based Modeling

Vector based modeling used the 218 features tabular data described previously.

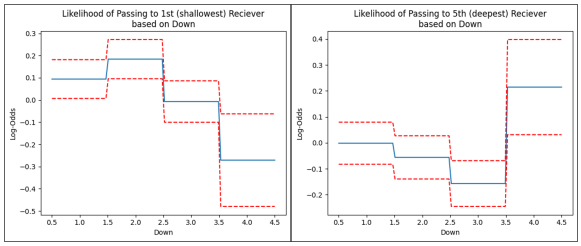

Figure 8: Generalized Additive Model outcome for likelihood of targeted receiver by down.

Logistic Regression Similar to the generalized additive models, a One versus Rest approach was taken in modeling the likelihood of a pass being thrown to each receiver as a function of the 218 features described earlier. An L1 regularization was taken in order to minimize the impact of non-relevant features on the final model. Additionally, a 5-fold cross validation was implemented to determine the best regularization strength.

Random Forest Classification For the Random Forest Classification model, a 5-fold cross validation approach was also taken with hyper parameter tuning for max depth, max features, and number of trees.

Multi Layer Perceptron A 3-layer feed forward neural network was constructed with 256, and 128 nodes in the first and second layer, respectively. Layers were batched normalized and there was a drop out layer before the final softmax activation function. The model was trained with a Cross Entropy Loss and an ADAM optimizer (learning rate = 0.005).

Graph Based Modeling

Prior to ingestion by graph models, the data structure had to be reconfigured. There are three different types of components to a graph; nodes, edges, and global components. For this project, nodes referred to players on the field and their features includes the players' age, weight, height, X, Y location, speed, acceleration, orientation, direction, and position. Edges refer to the relationships between the nodes (players). These features included their inverse distance, difference in speed, difference in orientation, and a binary designation for whether they were on the same team or not. For our purposes, all players have edges between each other, i.e. the graph is a fully connected directed graph. Global features refer to everything else outside of the player specific data; time of game, score differential, quarter, down, time Remaining. These graph structures can be visualized as overlays on player field position to give a better idea of how graphs are constructed. In Figure 9 , we see an example play around the 50 yard line with offense players in red and defensive players in blue. For graph deep learning methods, models were trained for 50 epochs on a training and validation set. Then they were assessed via accuracy, recall, precision, and F1 on the final hold out test set.

Graph Convolutional Network For the graph convolutional network, the inverse Euclidean distance between players was used as the edge weights ($e_{i,j}$) to normalize the convolution across edges. The equation to calculate each nodes new embedding with a size of the hidden dimension is seen below (Morris et. al. 2018).

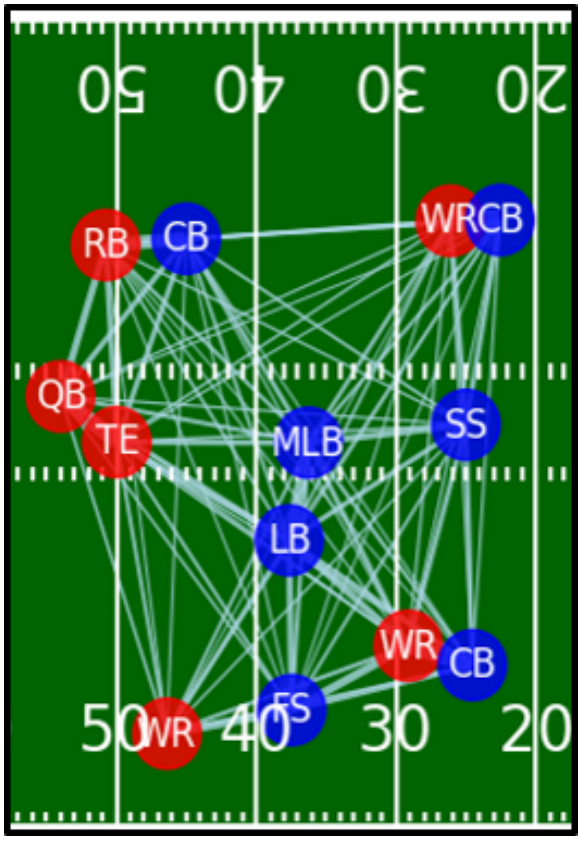

Figure 9: Example graph construction with field overlay.

$$x_i' = W_1 x_i + W_2 \sum_{j \in N_i} e_{ji} x_j$$

A single layer network was constructed, with a hidden dimension size of 128. Layer was batch normalized and had a drop out at 0.2. A global average pooling layer was inserted prior to concatenation with the global features. Again, an ADAM optimizer was used on a Cross Entropy Loss with a learning rate of 0.005. In total, this resulted in 25,656 trainable parameters.

Relational Graph Convolutional Network A relatively new graph algorithm was implemented in the current project, taking advantage of the bipartite nature of sports teams. This means that we can categorize the edge types between players to indicate whether those players are on the same team or not. Relational Graph Convolutional Networks (RGCN) uses a similar equation as the GCN above, however it can learn separate weight matrices for each edge type. This is intuitive as we might expect the before between two players to be different whether they are on the same team or not. This allows the network to accounts for that. A single layer network was constructed, with a hidden dimension size of

128. Layer was batch normalized and had a drop out at 0.2. A global average pooling layer was inserted prior to concatenation with the global features. Again, an ADAM optimizer was used on a Cross Entropy Loss with a learning rate of 0.005. In total, this resulted in 25,656 trainable parameters.

Graph Attention Network The graph attention network is similar to the graph convolutional network; however, the edge weights are learned through training as 'attention'. Thus, nodes with higher degrees of similarity result in having higher attention coefficients between them, implicitly. The equation below is taken from Veličković et. al. and calculates the new node embedding from attentional layers.

$$x_i' = \alpha_{i,i}\Theta x_i + \sum_{j \in N_i} \alpha_{i,j}\Theta x_j$$

where

$$\alpha_{i,j} = \frac{exp(LeakyReLu(a^T[\Theta x_i || \Theta x_j]))}{\sum_{k \in N_i} exp(LeakyReLu(a^T[\Theta x_i || \Theta x_j]))}$$

A single layer network was constructed, with a hidden dimension size of 128. Layer was batch normalized and had a drop out at 0.2. A global average pooling layer was inserted prior to concatenation with the global features. Again, an ADAM optimizer was used on a Cross Entropy Loss with a learning rate of 0.005. In total, this resulted in 22,072 trainable parameters.

Empirical Results

Vector Based Modeling

First, for the logistic regression, we plots the relative feature importance for each of the five models to understand what characteristics were important for determining how the quarterback would react to different scenarios. Interestingly, as can be see from Figure 10, , for the shallower receivers the biggest influence on the probability of getting thrown to was their distance down the field. This is in contrast to the deepest receiver, whose main influences were the formations the offense took at the line of scrimmage. This may indicate that shorter throws to shallower receivers could be opportunistic in nature while deeper throws may be more designed and planned out ahead of time.

The hyperparameter search yielded the best model with a high regularization penalty of C = 100, meaning many variables were likely zero-ed out. The model displayed mild over fitting with a training accuracy of 52% and a test accuracy of 48%. Recall, precision, and F1 scores are reported and compared in Table 1. Regardless, it can be seen from the confusion matrix in Figure 11, that the model was more successful at classifying for shallower targets than deeper targets.

For the random forest classification model, the hyperparameter search yielded a deep tree structure with max depth of 14, max features were calculated with square roots, and 200 trees were used to predict outcomes. The random forest model greatly over fit as the training accuracy reach 99% while the test accuracy achieved 52%. This is no surprising for random forest models. Recall, precision, and F1 scores

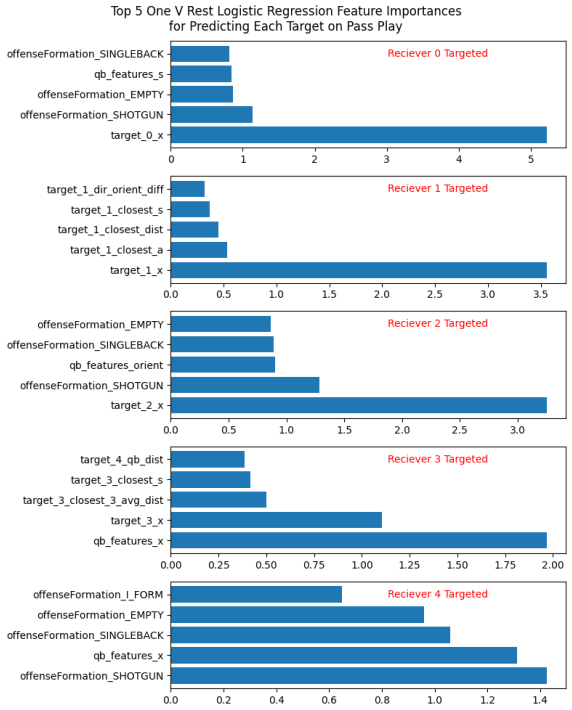

Figure 10: Logistic Regression Feature Importance.

are reported and compared in Table 1. The confusion matrix mirror the confusion matrix from the logistic regression.

The MLP produced the overall highest performance of any models in the project, with a training accuracy of 83% and a test accuracy of 54%. The Train-Validation curve in Figure 14 does indicate the model is over training. Adjusting the complexity of the model by changing the number of layers and number of hidden nodes did not significantly change the outcomes of the model. The confusion matrix (Figure 13.) also seemed to indicate it had a more even predictive capability across classes.

Graph Based Modeling

First, the graph models performed quite poorly compared to the vector based models. The more general GCN only achieved train and test accuracies of 48% and 29%, respectively. The learning curve in Figured 15 indicates over fitting as the model continued training. The validation data set performance plateaued after approximately 15 epochs. The model did even worse on the deeper receiver classifications than the previous vector models; although shallower receiver classification performed similarly.

The RGCN, while also including ability to distinguish between edge types, only performed 1% better than the GCN, with a train and test accuracies of 49% and 30%. The learning curve and confusion matrix mirror that of the GCN as well. This might indicate there weighting matrices are similar between the models.

Lastly, and surprisingly given its capacity to learn, the GAT performed the worst of the graph methods with training and test accuracies of 36% and 28%. The learning curve and

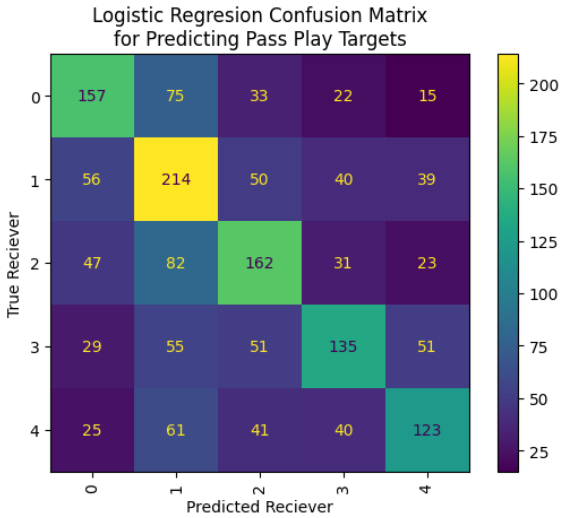

Figure 11: Confusion Matrix.

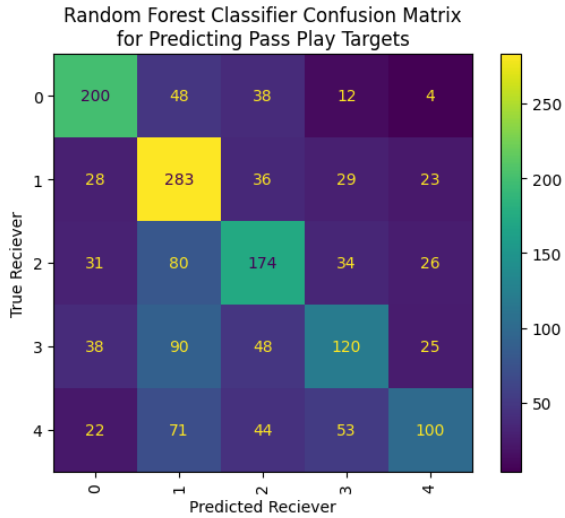

Figure 12: Confusion Matrix.

confusion matrix mirror that of the GCN as well. Similar to the RGCN, this might indicate there weighting matrices are similar between the models.

All performance metrics for all models are included in Table 1.

Conclusions and Discussion

The purpose of this project was to build and evaluate models that could use high resolution NFL football player tracking data that could predict which receiver on the field the quarterback would throw to. We used a mix of traditional and deep learning approaches, as well as attempted to apply novel graph based methods to the problem. Generalized additive models allowed us to uncover unique relationships between play characteristics that could have not been possible with linear methods. Regardless of methods used and despite the relatively balanced class outcomes, we were not able to achieve the predictive capabilities of previous methods. It is largely hypothesized that our attempt at predicting throws from data 0.5 second before the throw blunted our capacity quite a bit.

Vector based models tended to out perform our graph based models in their pure predictive capabilities, with the Multi Layer Perceptron performing the best overall. This higher performance comes at a higher cost, however. These methods require strict positional ordering and careful manual feature crafting to achieve such performances. While limited in the paper, outcomes from graph based learning can be used to explore much deeper latent relationships between players on the field due to the expressive and flexible nature of the graph. Other possible solutions to try in future research will be to construct the problem as node regression task to estimate probabilities across nodes, or use the attention coefficients from the GAT model themselves to predict quarterback to receiver connections. In addition, only fully connected graphs were used in this project, while there are an almost infinite number of ways to construct the graph based on context specific rules. For example, maybe the graph has a hub and spoke formation with only defensive players near offensive players have connections or just offensive players are connected. These are exciting questions that could be answered with graph methods.

Code Repository

The repository can be found at https://github.com/tweedell/DS5500_Project2 which contains the jupyter notebook where all code and data needed for training and analysis resides.

References

Burke, B. 2019. DeepQB: Deep Learning with Player Tracking to Quantify Quarterback Decision-Making Performance. *MIT Sloan Sports Analytics Conference*.

Deshpande, S. K.; and Evans, K. 2019. Expected Hypothetical Completion Probability. arXiv:1910.12337.

Horton, M. 2020. Learning Feature Representations from Football Tracking. *MIT Sloan Sports Analytics Conference*.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2018. Weisfeiler and Leman Go Neural: Higher-order Graph Neural Networks. *CoRR*, abs/1810.02244.

Pompeu da Silva, G. 2022. Frame by frame completion probability of an American football pass. *University of São Paulo Masters Thesis*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. arXiv:1710.10903.

Xenopoulos, P.; and Silva, C. 2021. Graph Neural Networks to Predict Sports Outcomes. In *2021 IEEE International Conference on Big Data (Big Data)*, 1757–1763.

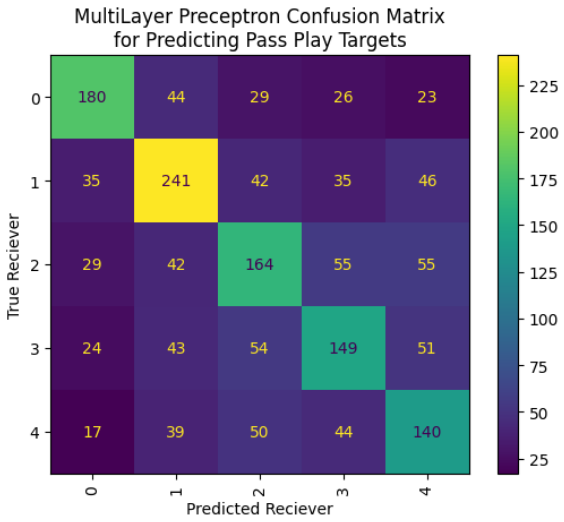

Figure 13: Confusion Matrix.

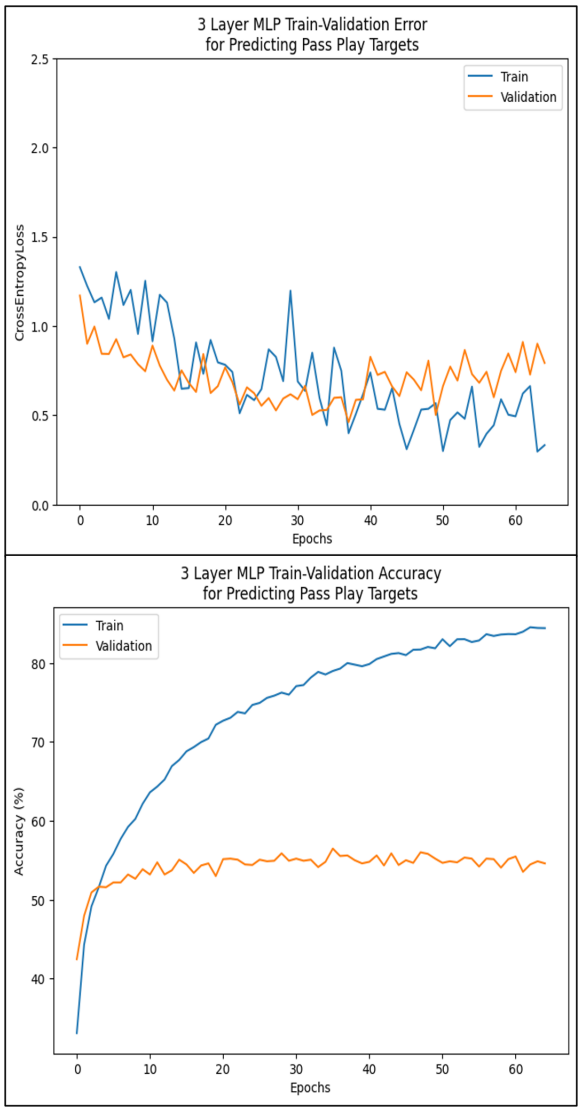

Figure 14: Learning Curve.

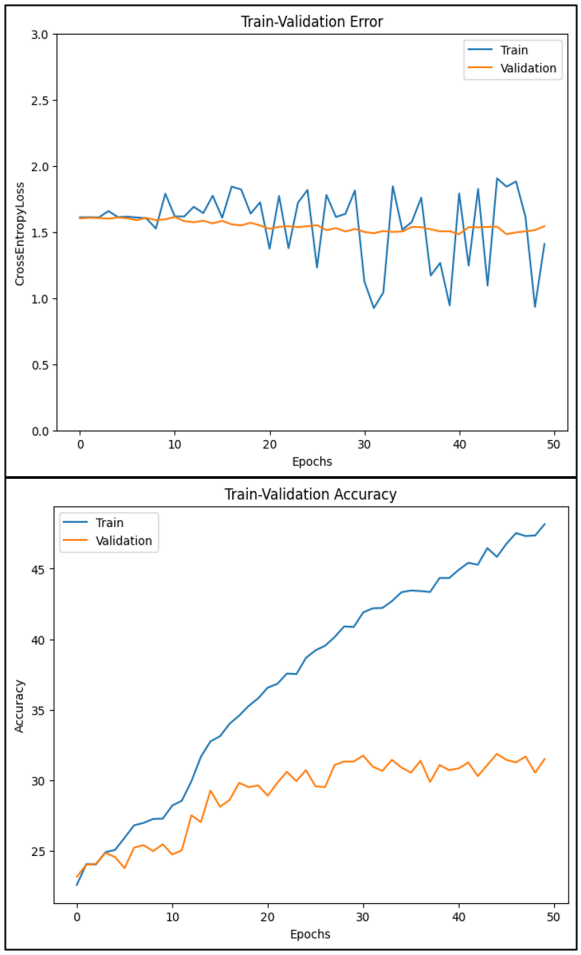

Figure 15: GCN Learning Curve.

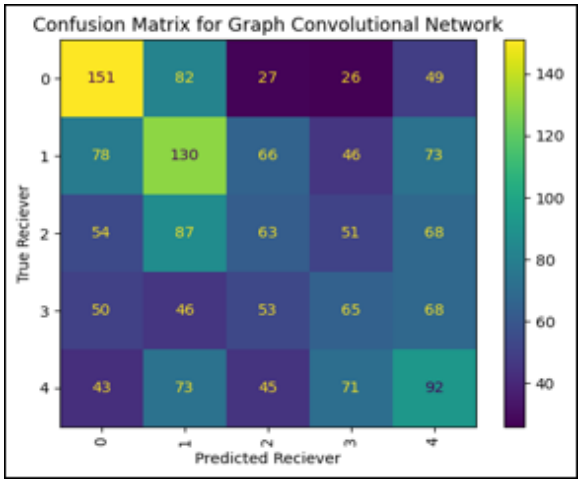

Figure 16: GCN Confusion Matrix.

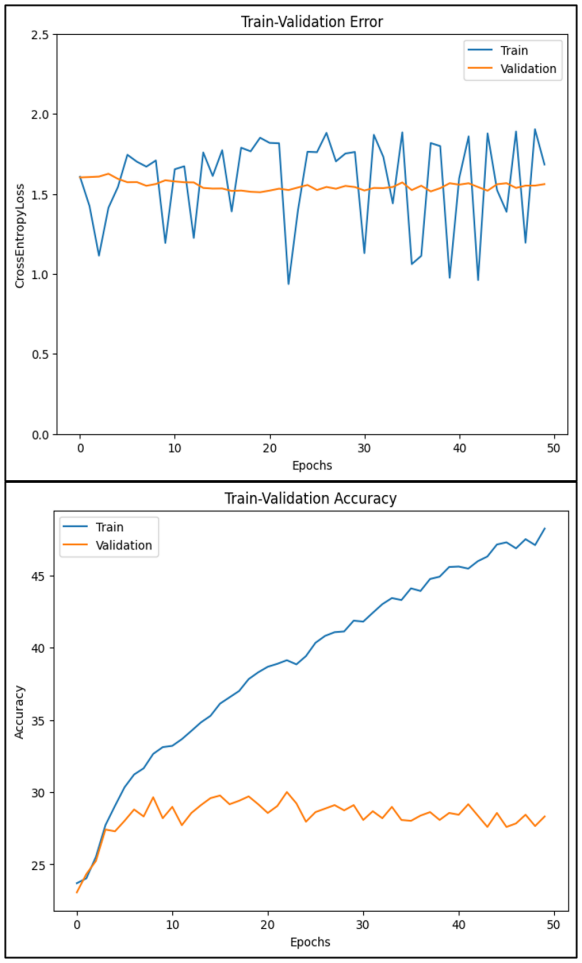

Figure 17: RGCN Learning Curve.

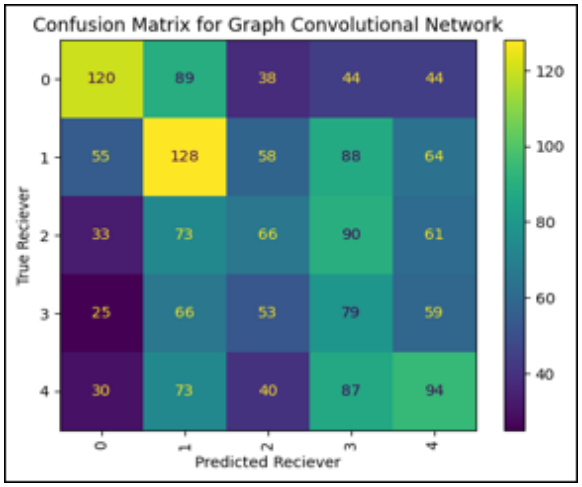

Figure 18: RGCN Caption.

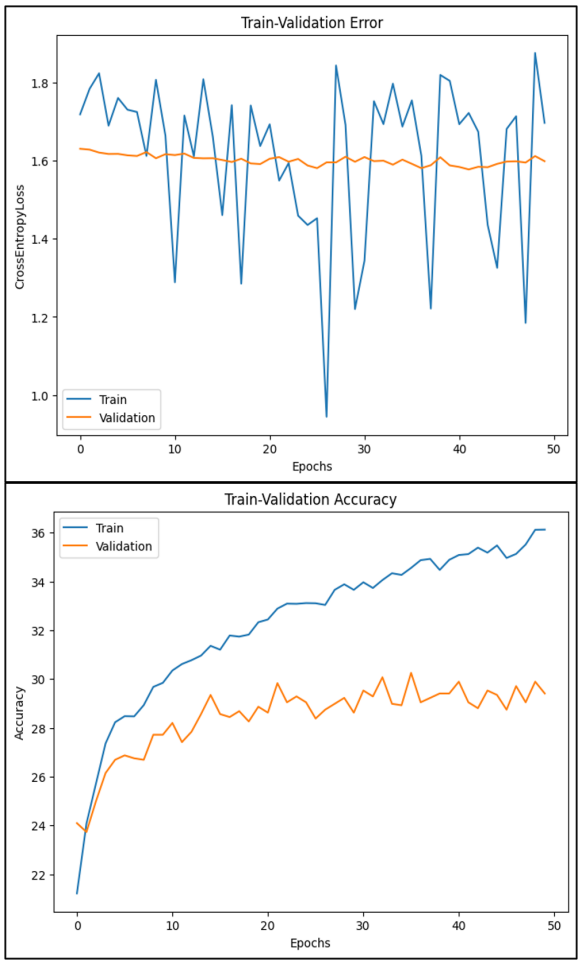

Figure 19: GAT Learning Curve.

Model	Train Accuracy (%)	Test Accuracy (%)	Test Precision	Test Recall	Test F1
Logistic Regression	51.7	48.2	48.0	47.0	48.0
RFC	**99.2**	52.4	**54.0**	**53.0**	52.0
MLP	82.9	**53.5**	53.0	**53.0**	**53.2**
GCN	48.0	29.0	30.0	29.0	29.0
RGCN	49.0	30.0	31.0	29.0	30.0
GAT	36.0	28.0	29.0	29.0	28.0

Table 1. Model Results

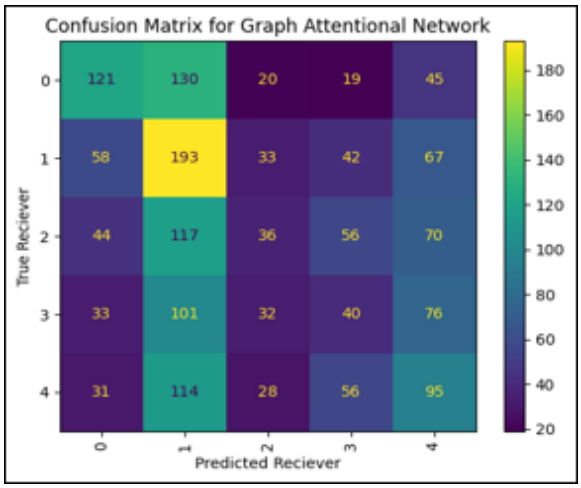

Figure 20: GAT Confusion Matrix.