PREDICTING FANTASY FOOTBALL RANKINGS

by

KENNETH STEADMAN

Advisor

JOSHUA ECKROTH

A senior research paper submitted in partial fulfillment of the requirements

for the degree of Bachelor of Science

in the Department of Mathematics and Computer Science

in the College of Arts and Science

at Stetson University

DeLand, Florida

Spring Term

2022

**Table of Contents**

[ABSTRACT 3](#_Toc101287435)

[1. INTRODUCTION 4](#_Toc101287436)

[2. PREVIOUS WORKS 5](#_Toc101287437)

[3. BACKGROUND 6](#_Toc101287438)

[Fantasy Football Overview 6](#_Toc101287439)

[Stats and Scoring 6](#_Toc101287440)

[Team Composition 7](#_Toc101287441)

[Neural Nets and Regression 8](#_Toc101287442)

[4. METHODOLOGY 9](#_Toc101287443)

[Dataset and Manipulation 9](#_Toc101287444)

[Network Architecture 10](#_Toc101287445)

[Train, Validation, Test 11](#_Toc101287446)

[Root Mean Square Error 11](#_Toc101287447)

[Random Forest 12](#_Toc101287448)

[Epochs and Learning rate 13](#_Toc101287449)

[Optimizer 13](#_Toc101287450)

[Loss Function 14](#_Toc101287451)

[6. RESULTS 14](#_Toc101287452)

[7. DISCUSSION 17](#_Toc101287453)

[8. MODEL AND DATASET ADJUSTMENTS 17](#_Toc101287454)

[9. CONCLUSION 19](#_Toc101287455)

[10. REFERENCES 20](#_Toc101287456)

# **ABSTRACT**

The first fantasy football leagues can be traced back to the 1960s. During this time getting the results of games and player performance were much more difficult than it is today. The popularity of fantasy football took off after the implementation of automation through the internet in 1997. Since then, the fantasy football market is now worth over 18.6 billion U.S. dollars and is only growing in popularity. The most important event in any fantasy football league is the draft. The draft is where the members of a league come together in-person or online and pick the players they want on their team. This part is crucial as the players selected during the draft will result in you winning or losing your league championship. However, Drafting is not as simple as it sounds and in order to come out of it with a championship caliber team, strenuous research and hard decisions will have to be made. The goal of this senior project is to help make this process easier by providing a neural net model that can predict the fantasy points of a player as accurately as possible using data from previous seasons. Currently not much is available in terms of draft help via AI, most apps have a list of players, and they are ranked by average draft position(the spot they are most often taken at based on all previous drafts). However, this way of ranking players does not provide the user enough information as higher average draft position does not always equate to higher fantasy output by that player. To tackle this problem, the model must take in all relevant stats, such as catches, targets, touchdowns, etc., to provide an accurate prediction of the players fantasy points.

# **INTRODUCTION**

Fantasy football has taken over the NFL fanbase and the gambling community by storm in recent years. This in turn has made fantasy football into a highly profitable market. Fantasy football is a game that is played alongside the NFL using a point system to rank real football players based on their in-game performances. Fantasy football players will join a league with friends, family, or random people and draft real NFL players for their fantasy team. Those teams will go head-to-head each week of the NFL season and in the end the player whose team won the most weekly matchups wins the league. It is highly competitive and because NFL players performances are volatile it makes fantasy football very unpredictable.

As an avid fantasy football player with firsthand knowledge of how difficult it can be to make the right decisions when drafting a team or even figuring out which player will play the best that week, having an accurate prediction of NFL players rankings at the end of the season would be extremely valuable. However, It is unlikely that a prediction could be accurate most of the time due to the many variables associated with playing football, such as injuries and game script(the way the real NFL game plays out). With this in mind, it does not change the utility of having such an insight. Alongside the numerous fantasy football tools and analytics, there is a place for AI to add another layer of resources to help make tough decisions come draft time or on a weekly basis a little easier.

AI is not the only way to solve the problem of predicting fantasy football rankings, however due to its ability to sift through large amounts of stats and learn correlations faster than people, AI is the perfect place to start for this task. AI, in this case can easily find a relationship between the raw in game stats and fantasy points quickly and effectively, which will be explained further in the rest of this report. This paper will go through previous works, the background and intricacies of the game of fantasy football, and how we worked with AI to make score predictions. The neural network architecture will be explained in detail. Then we will provide the preliminary results and conclude with the usefulness and future plans of this project.

# **PREVIOUS WORKS**

In 2015, Roman Lutz of University of Massachusetts Amherst published a paper in which he used AI to predict fantasy football scores of Quarterbacks with limited data. The two methods he used to predict the scores were support vector regression and a Neural Network. His root squared mean error on all the data in his dataset was 7.815, mean average error was 6.238, and lastly his mean relative error was 0.453[1] As mentioned, this research focused solely on the quarterback position whereas we are expanding to all positions excluding kicker and defense due to dataset limitations.

Another paper in 2015 aimed to find the optimal fantasy football team. Paul Steenkiste of Stanford University focused on a type of fantasy football that is played daily instead of the standard season long league. He used three models: Linear Regression, Random Forests, and Multivariate Regression Splines, using every player’s statistics from the previous weeks and their opponent as the input which resulted in around 3,200 datapoints. All three models were able to accurately predict weekly quarterback rankings, predicting five out of seven of the top ranked quarterbacks that week. However, the models as expected were not able to predict stats like fumbles or interceptions very accurately due to the randomness of their occurrence. Also, the models overvalued two quarterbacks who were performing well during the season but had a bad week leaving them dead last in the true ranking but in the top five in the models predictions.[2]

# **BACKGROUND**

## Fantasy Football Overview

Fantasy football is a game in which a group of people come together and construct teams from current NFL players and then face off head-to-head on a weekly basis. There are many different ways to play fantasy football, from season long leagues to daily play-ins. There are numerous modes with different rules and roster constructions. The most popular type of fantasy football is called a redraft league. A redraft fantasy football league is where rosters only last for a year, so when the new NFL season starts all the league members come together and draft brand-new teams. There are leagues in which you keep the team you drafted called dynasty leagues where you revitalize your roster by drafting incoming rookies. Leagues can also vary in size, but they cannot be too large due to the limited NFL player pool. The standard size is 10 to 12 teams to allow for an even talent distribution as the more teams there are, the less top tier players per team which leads to a less enjoyable experience.

## Stats and Scoring

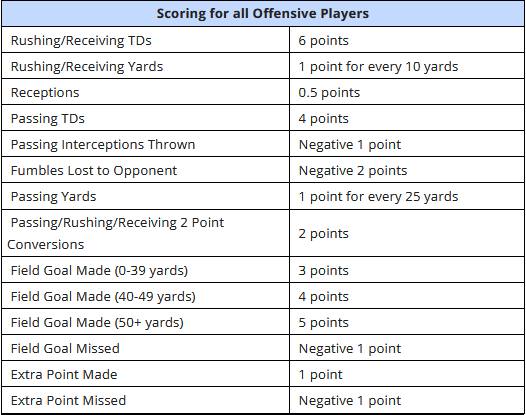
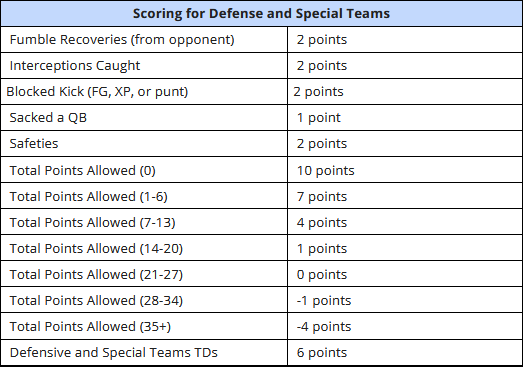
The three main scoring formats of fantasy football are half PPR, standard, and full PPR. PPR stands for points per reception. Standard does not reward players for receptions, half PPR gives half a point per reception, and full PPR gives a full point per reception. The scoring format is completely preferential; however, many fantasy football enthusiasts prefer half PPR compared to the others due to its ability to make a more even scoring system and not have one position over power the others. A running back who gets a lot of receptions would be more valuable in full PPR than any other position and half PPR limits that allowing for other positions to shine. This scoring system is important to our model, as different systems will make a player have vastly different fantasy scores at the end of the season. The current dataset provides fantasy points in PPR, so the neural net model as of now will only be able to predict scores based on that scoring system.

Figure 1: These tables display the most common scoring settings in a standard fantasy football league. **Left:** Offensive scoring settings**. Right:** Defense and Special Teams scoring settings. Retrieved from https://www.fantasypros.com/scoring-settings/

## Team Composition

Table

Description automatically generatedTeams are constructed through a draft process in which league members are assigned a randomly generated draft position. A normal league allows for seven different positions: A quarterback, running back, wide receiver, tight end, kicker, and defense. The neural net will focus on quarterback, running back, wide receiver, and tight end due to the limitations of the dataset. There is also a spot called a flex spot. This opens up an extra slot to a running back or wide receiver. Some leagues have multiple flex spots and even open it up to any position. Also, due to bye weeks, which are weeks in the NFL where teams get a week off for rest, and injuries, there must me backup players available. A typical league will give every team five bench spots to ensure every team will have a full starting lineup. Players on the bench receive scores but are not counted towards the weeks total, only fantasy points scored by players in the starting lineup will be added to the total fantasy points of that week.

Figure 2: Example of a fantasy football roster.

## Neural Nets and Regression

A picture containing dark, light, colorful

Description automatically generatedModeled after the way neurons work in the human brain, neural networks are interconnected nodes that overtime can find relationships and classify raw data. Using algorithms these artificial systems can learn and improve quickly and more effectively than people, making this is a perfect system for tackling the problem of predicting fantasy football player rankings. This task is also classified as a regression problem. We are using various in-game stats to find a linear relationship to the total fantasy points. We can use linear regression to assume that with a combination of stats that the total fantasy points can be calculated accurately. Refer to Fig.3, here we can see how a line can be drawn from various datapoints.

Figure 3: Visual representation of linear regression. Retrieved from https://medium.com/simple-ai/linear-regression-intro-to-machine-learning-6-6e320dbdaf06

# **4. METHODOLOGY**

## Dataset and Manipulation

We are using a dataset constructed by Fantasy Football Data Pros[2], which includes yearly fantasy football data going back to 1970 and weekly fantasy football data going back to 1999. The dataset consists of over 11500 rows and 34 columns which contain the relevant stats and the fantasy points. As shown in Figure 1, there are only a few in game stats that result in fantasy points. However, the dataset includes additional stats and classification data such as completions, age, position, targets, yards per route, and games played which are very valuable in training the neural net model. The model is currently trained on years spanning from 2000 to 2019.

Using all the years from 2000 to 2019, we split the dataset into two data subsets and extracted the relevant columns from the years prior to 2019(x values) to predict the fantasy points from 2019(y values). However, the data in its current form must be normalized to bring all the values from x to a range between 0 and 1 as the neural network works best when values are in this range. To do that we simply calculated the range by subtracting the minimum x values from the maximum x values, then taking x minus the mean and dividing that by the range which resulted in a normalized dataset. We also had to take in classification data which by itself cannot be fed into the model. To convert the classification data into numerical values, we converted them into onehot columns, which are encodings of the qualitative data that are now represented by ones and zeros. These values were then concatenated onto the existing dataset before normalization.

## Network Architecture

The neural network model consists of three layers known as the input layer, hidden layers, and output layers. The model itself is five lines of code, three linear functions with activation functions in-between. Refer to Fig.3, here we can see a visualization of the neural network with all the interconnected nodes and arrows showing the flow from the input layer to the output layer.

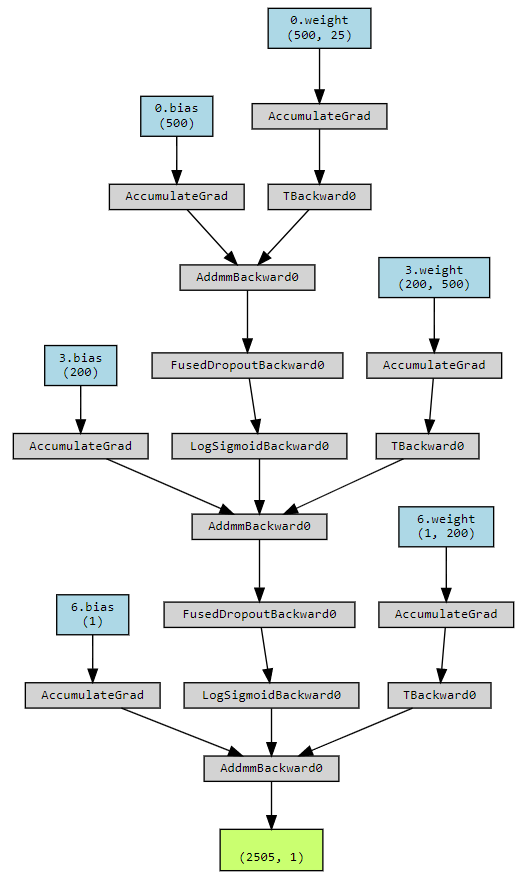


Figure 4: Neural net visualization. Rendered by TorchViz.[3]

]]

The neural network begins by taking in the inputs, which are the 25 columns in our training dataset. This information is then transferred to the hidden layers. In the connections between these layers, weights are assigned to each input and a bias is then added to each input after weights are multiplied. This weight is then sent to the activation function, which in our case is LogSigmoid. After it gets through the hidden layers, we configured the network to produce one output. The in features start at 25 then we output 500, 500 has be used as the in feature for the linear function then we output 200, then lastly the in features for the final linear function are 200 with the output being 1.

## Train, Validation, Test

It is important that we have data left over that the model has not trained itself on. This is achievable by splitting the dataset into three subsets: Train, Validation, and Test. Using the train\_test\_split function from sklearn[4], we split the data into an 80/20 split with 80 percent being the training subset and 20 percent being the validation and testing subsets. The 20% was split evenly between validation and testing subsets making them 10 percent each. The reason for splitting the data is for evaluation. A testing dataset allows us to validate the efficacy of the neural network model by giving it data it has not seen before therefore providing a more accurate evaluation of the model. Validation in this task was used along side the training dataset in the neural network to compare training results to the results from data not used during training. This comparison is shown per epoch in Fig. 7.

## Root Mean Square Error

Root Mean Square Error(RSME) is calculated as the root of the mean of the squared differences between the predicted values from the model and the actual values. In a 2014 paper, T.Chai and R. R. Draxler compared root mean square error to mean absolute error and in their findings, they concluded that RSME is more appropriate to use than the MAE when the errors follow a normal distribution. They also note the advantage of RSME is the avoidance of using absolute value which can be undesirable.[5] We have decided to use RSME to determine the errors in our neural net model to show the true number of fantasy points the model is off by and to use as a figure to compare to previous works. Fig. 5 shows how each are calculated.

A picture containing text, watch, clock

Description automatically generatedA picture containing text

Description automatically generated

Figure 5: Equations for RSME and MAE

## Random Forest

In an effort to compare our model to other known methods of generating predictions, we decided to look at random forests. Random forests are a combination of decision trees that can be modeled for prediction tasks like the one we face with fantasy football points. Random forests are also capable of handling large datasets due to its handling of variables. We are using the random forest classifier module from scikit learn. This module will allow us to run the same data into a different method of prediction generation to have a comparison to our neural network. Below in figure 8 is a diagram of how random forests work.

Diagram

Description automatically generated

Figure 6: Random Forest Visualization. Retrieved from https://towardsai.net/p/machine-learning/why-choose-random-forest-and-not-decision-trees

## Epochs and Learning rate

Epochs and learning rate are essential parts of the training process of neural networks. Epochs are the number of cycles set for the model to go through the training set, with the goal being that the model learns and improves upon each cycle. We set the epochs in our neural net model to 150 as that was all that was needed before the change in loss and error became miniscule per epoch and adding more epochs would have been excessive and would not have made much of an impact. Epochs and learning rate work together in tandem, learning rate is what determines how fast the model goes through the training set. However, faster does not mean better in this case as it has a trade-off, the higher the learning rate the more likely you end with a sub optimal set of weights and overshooting. The lower the learning rate the more accurate the weights are however it is much slower and requires more epochs. For our model we settled on a learning rate of around 0.01 which was slow enough to prevent overshooting and fast enough to finish in 150 epochs. These two hyperparameters are very easy to tweak and they must be tuned to ensure the best possible accuracy is reached.

## Optimizer

We used an optimizer function called Adam. Adam was first introduced in 2015 in a research paper by Diedrick P.Kingma in collaboration with Jimmy Lei Ba. It was proposed as a method for stochastic optimization. Stochastic optimization is an optimization method that use and generate random variables. Their findings concluded that Adam is computationally efficient and robust in convex and non-convex optimization problems.[6] When looking at a regression problem, there are two optimizer functions that are widely used: Adam and SGD(Stochastic Gradient Descent). After trying out both of these functions for this task, it was evident that Adam outperformed SGD as SGD caused a lot of overfitting and higher errors.

## Loss Function

MSELoss refers to a loss function that calculates the mean square error. This is closely related to the RMSE mentioned earlier. The result of MSELoss is squared so what RSME does is perform a square root on MSELoss. Both of these functions essentially serve the same purpose, as an indicator of how well the model is performing. The square root of MSELoss gives a number that is reasonable in terms of the amount of fantasy football points scored in a season rather than a number in the thousands. Fig 6 shows the formula for MSELoss, which finds the mean of the difference between the true and predicted values then squares it.

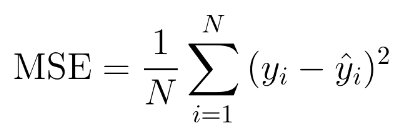
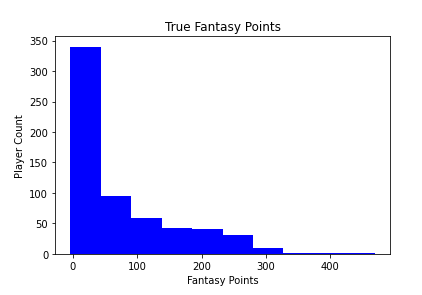
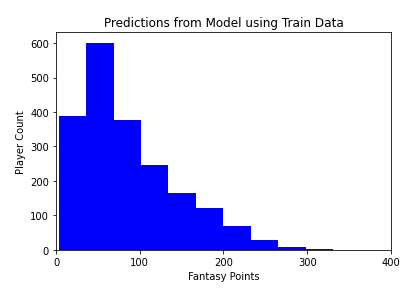


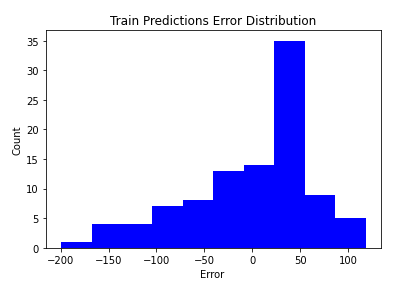
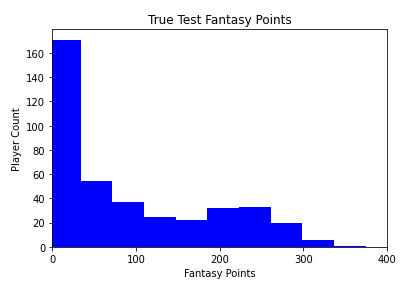
Figure 6: Equation for MSELoss. Retrieved from https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3

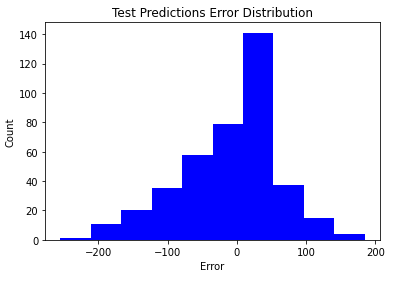
# Chart Description automatically generated**6. RESULTS**

Chart

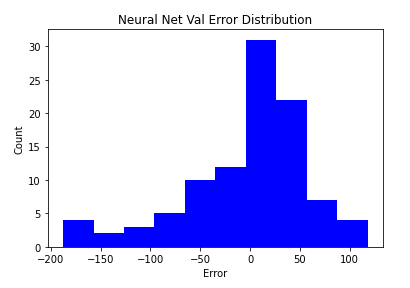
Description automatically generated

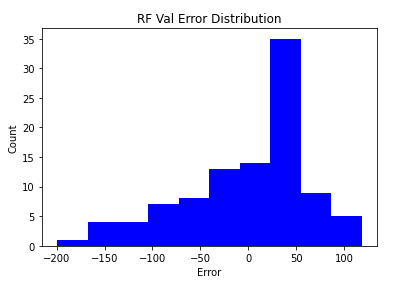




Chart, histogram

Description automatically generated





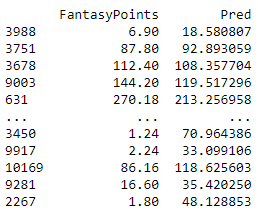


Figure 7: Graphs and table used for evaluation and to visualize the efficacy of the model. Rendered by Matplotlib.[7]

Referring to the first two graphs in Fig.7, these graphs represent the loss and error that was calculated by MSELoss and RMSE over the 150 epochs. The next set of graphs are histograms that show the real fantasy values in the 2019 subset compared to the predicted output generated from the neural network model as well as the error distributions. The histograms help visualize the disparity between the prediction from the model and the true fantasy points in the selected subset. The error distributions take the predicted values and subtract the true value. Refer to the table in Fig.7 which depicts the true fantasy points of the validation subset and the neural networks prediction in the other. Here we can also see the discrepancy in a raw data form where some seem pretty close, and others are off by a large margin. The last two histograms show the error distributions of the neural network validation distribution compared to the random forests error distribution.

|  |  |  |  |
| --- | --- | --- | --- |
| *Train Loss* | *Validation Loss* | *Train Error* | *Validation Error* |
| 4545.2095 | 3707.6855 | 67.4182 | 60.8908 |

Table 1: Shows the training results after 150 epochs. Loss refers to MSELoss and Error refers to Root Mean Square Error(RMSE).

# **7. DISCUSSION**

As shown in Table 1, the results of the training yielded a training loss of 4545.2095, validation loss of 3707.6855, train error of 67.4182, and a validation error of 60.8908. The loss is the mean square error, while the error is the root of the mean squared error. These results tell us that on average our models’ predictions are off the true fantasy point value by an average of about 61 fantasy points. This number in relation to the game of fantasy football itself is too high to be considered useful. For example, the highest scoring fantasy player in 2019 ended the season with a little over 400 fantasy points. Being off by an average of 61 points means that the player rankings will drastically change and in turn render the predictions not accurate enough to be a useful insight. However, when compared to the random forest predictions, our neural network appeared to edge it out slightly having an average validation error of 48 points while random forest produced an average validation error of 52.

# **8. MODEL AND DATASET ADJUSTMENTS**

*Log Transformation*

In an attempt to improve the negative skew in our predictions, we decided to further manipulate the dataset by transforming the values with a log to achieve a more normal distribution. The model was routinely underpredicting and we believed that if we could get the inputs in a closer range it would result in a lower error. This produced values that were close together and it looked promising. However, after testing this proved not to be beneficial and ultimately caused the model to be unstable. We decided to revert back to the original method of normalization which transforms the inputs to a value between 0 and 1.

*Data Subsets*

After the log transformation attempt, another possible solution to our skew problem was to put more emphasis on the higher scoring players. We decided to cut all players who scored less than 50 fantasy points from the dataset to see if that would allow the model to more accurately predict those higher fantasy point values. However, again we were met with underwhelming results as the model was underpredicting just as much as before.

*Hyperparameter Search*

WANDB.AI is a powerful tool that can be used to fine tune the parameters of a neural network model. It can take in parameters like epochs, learning rate, optimizers, and loss functions to run multiple tests called sweeps to locate the best combination of parameters that meet the goal criteria. The goal criteria in this case was the lowest validation error that could be produced. This tool is great to use in general when designing and testing your model as it can be a method of improvement when other options do not seem to help improvement. In our case, after we went through the log transformation and data subsets trying to get to the bottom of our problem, we decided to take advantage of WANDB and it proved to be beneficial. Although, the improvement was miniscule as it only improved by a few points on average. WANDB gave us the learning rate and epoch value that produced the lowest validation error, which was 150 epochs and a learning rate of 0.016824394620345177. We plugged these into our existing neural network and we immediately saw an improvement.

# **9. CONCLUSION**

Creating a tool that can accurately predict fantasy points is extremely valuable. We took on the task of using a neural network model to see if it were possible to use AI to predict the yearly fantasy points in order to have an accurate way to rank players based on results from previous years. The model was able to predict fantasy points with an average error of 61 fantasy points. This, as mentioned, is not accurate enough to be useful when you take into account the points disparity between players and being off by such a large margin will produce wildly inaccurate rankings. However, this is a difficult problem for even AI to solve as there is numerous outside variables and external factors that go into an NFL players production. Any given week a player could sustain a season ending injury leaving your prediction useless. There are also suspensions, retirements, game script, and others that are just not able to be predicted in any accurate manner. While we did not reach our target goal of an error of 30 fantasy points, the process and experience of taking on a challenging task such as this has resulted in an informative and thorough examination of the benefits and limitations of using AI for problem solving and predictions.

# **10. REFERENCES**

[1] Lutz, R. (2015). Fantasy football prediction. *arXiv preprint arXiv:1505.06918*.

[2] Steenkiste, P. (2015). Finding the Optimal Fantasy Football Team.

[3] Scikit-learn: Machine Learning in Python, Pedregosa *et al*., JMLR 12, pp. 2825-2830, 2011

[4] Zagoruyko S. (2019). PyTorchViz. Github repository. https://github.com/szagoruyko/pytorchviz

[5] Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific model development*, *7*(3), 1247-1250.

[6] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[7] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90-95, 2007.