MCGILL UNIVERSITY

FINAL REPORT - GROUP 23

ECSE456 - DESIGN PROJECT 1

Picking Winners for Fantasy Basketball with Machine Learning

Group Members
Ege AYDEDE - 260576559
Stephen POOLE - 260508650
Florence REGOL - 260639261
Asher WRIGHT - 260559393

Supervisor
Prof. Ioannis PSAROMILIGKOS

December 1, 2017

Abstract

This project was motivated by the work of David Hunter et al. [5] where the objective was to predict winning teams in fantasy hockey using integer linear programming. In that project, we consider the problem of picking winners in fantasy basketball using machine learning. The object of the project is two-fold: First, to learn the principles of machine learning. Second, to apply this knowledge outside a theoretical framework, in order to build something tangible. In order to achieve these goals, the design process was broken into three steps: Research, data collection, and implementation. For the first half of this project, work was done in all three of these areas. Specifically, a significant portion of the research was completed, all the data required was collected, and a preliminary working neural network was built. Beyond this, a first iteration of the system architecture was developed. Reading, watching lectures, and following tutorials on project-relevant topics, allowed us to complete the work done thus far.

Acknowledgements

We would like to acknowledge and thank everyone who helped us with this project, including our supervisor, Professor Psaromiligkos, and two helpful talkers, Islay Wright and Ben Potter.

Contents

List of Figures

Abbreviations

Most of the terms used in this report are not abbreviated, and the material is likely familiar with most people. Some abbreviated terms that may not be clear are:

- **NBA**: National Basketball Association
- **NN**: Neural network
- **ML**: Machine learning
- **PG**: Point guard
- **SG**: Shooting guard
- **SF**: Small forward
- **PF**: Power forward

1 Introduction

The primary objective of this project is to build a system to predict winning line-ups of basketball players for daily fantasy basketball competitions. A winning line-up is any line-up that, upon being entered into the competition, yields more money than the entry fee (i.e. gains a profit). The goal of the project was to accomplish this using machine learning. We hypothesized this could be successful based on the assumption that there are hidden relationships between players on the same team, and the players on the opposing team. The project was separated into separate milestones that were intended to collectively allow us to build the final result. These milestones were:

1. Learning the necessary material regarding topics in machine learning
2. Collecting sufficient amounts of data to allow machine learning to actually take place
3. Finding relevant tools and technology to actually implement our final system.

The importance of this project has two components, one personal and one societal. The personal importance of this project has to do with the team creating it. This project has been, and will continue to be, a valuable learning opportunity. It allows the creators to learn more about data collection, machine learning, software projects, best practices, and engineering design. Machine learning is a field of significant interest to all of us. There is also a societal impact, tied to both the sports and academic communities. If we find that we are able to accurately and consistently predict winning line-ups for fantasy basketball, it may impact the way in which fantasy sports are played by everyone. Rather than rely on intuition, or optimization techniques alone, there may be more movement to use a neural network-based system if it can predict player performances with greater accuracy. This could not only change how fantasy competitions are played, but could be adapted to select which players should actually be played, in real life. Theoretically, a coach could use the same technique, although with a different metric (winning a game instead of maximizing fantasy points) to figure out who to play. Machine learning could help find patterns that we did not know exist. Finally, on the academic side, it could lead to more research on the applications of machine learning to sports betting.

2 Background

2.1 Inspiration

The inspiration for this project came directly from the paper, "Picking Winners" written by David Hunter, et al. The paper targets "top heavy" daily fantasy hockey competitions and uses the integer programming (IP) optimization technique to try to "beat" them. The optimization technique involved the researchers coming up with ideas about what *could* be beneficial for a line-up, building a model around the idea, and then optimizing and testing it. For example, one of their ideas was related to choosing defencemen on the same team. Upon reading this paper, we felt that the same goal could be achieved while allocating more to the computer. The neural net aimed to solve this.

2.2 Neural Networks

As we began exploring machine learning techniques, we decided, through the help of our research as well as our supervisor, that neural networks would allow for a good first implementation of our system. Neural networks are a set of interconnected nodes [2]. Inputs to the neural network are turned to outputs by passing them through a series of functions [2]. Figure 1 below gives an illustrative view of a neural network. The architecture of neural networks is inspired by the structure of a human brain, where the nodes represent neurons, and the links that connect the nodes represent synapses [2].

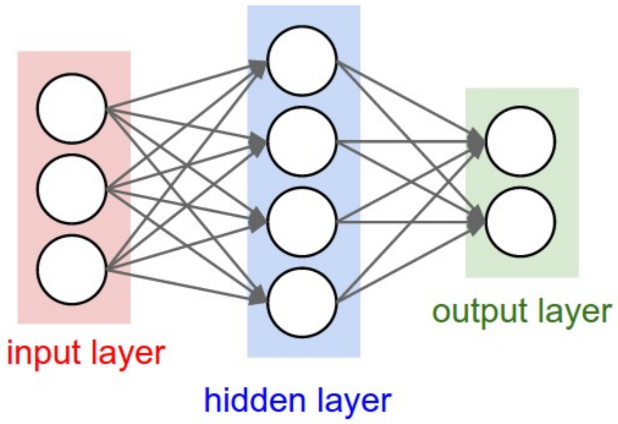


Figure 1: Illustration of neural network [2]

As shown in Figure 1, a neural network consists of an input layer, an output layer, and one or more hidden layers. Not surprisingly, the input layer takes in the inputs to the system, and the output layer yields the outputs of the system. For example, the input to a particular neural network might be two basketball teams, and the output might be the team we expect to win the match. The question remains, however, how the neural network figures out which output to generate based on a given input. This is done through the help of the hidden layer(s), as well as the links that connect all the nodes [4]. The combined effort of these two components are what allow learning to take place in a neural network.

In doing our research, we had to ask ourselves whether machine learning, and neural networks in particular, would be appropriate as a means of solving our particular problem. As we found while watching a lecture series on machine learning by Professor Yaser Abu-Mostafa, there are three conditions that should be met in order for machine learning to be considered a viable option [3]. First, there needs to be a distinguishable pattern between the inputs and the outputs of the system. For our problem, we hypothesized that this condition would be met since there should be a relationship between the players playing and the outcomes of the games. Secondly, this relationship should not be able to be trivially represented mathematically. This we also thought would be true since there are so many variables at play in any given basketball game. Finally, there needs to be enough data for the neural network to train with. Again, this condition was found to be satisfied.

Seeing as all three conditions described were satisfied, we deemed neural networks to be an appropriate solution for the matter. In theory, sufficient training of the network should allow learning to take place such that it be able to solve our problem of accurately predicting high-performing fantasy basketball lineups.

3 Requirements

3.1 Design Problem

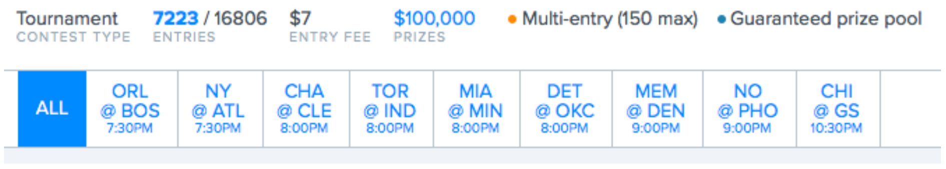
The problem we are trying to solve is to beat daily fantasy basketball competitions using machine learning. In order to fully understand this problem, it is important to detail daily, top-heavy, fantasy basketball competitions.

3.1.1 Competition Choice

When most people think of fantasy sports, they think of creating a team at the beginning of the sport's season, and following this team through the season. The types of competitions and specific competition rules actually vary greatly, and rather than try to create a system that can handle all of these, we decided to target a specific subset. The competitions we will target must meet the following criteria:

- Daily competitions
- Low entry fee (< $10)
- High, top-heavy, prize pool
- Multiple entries allowed

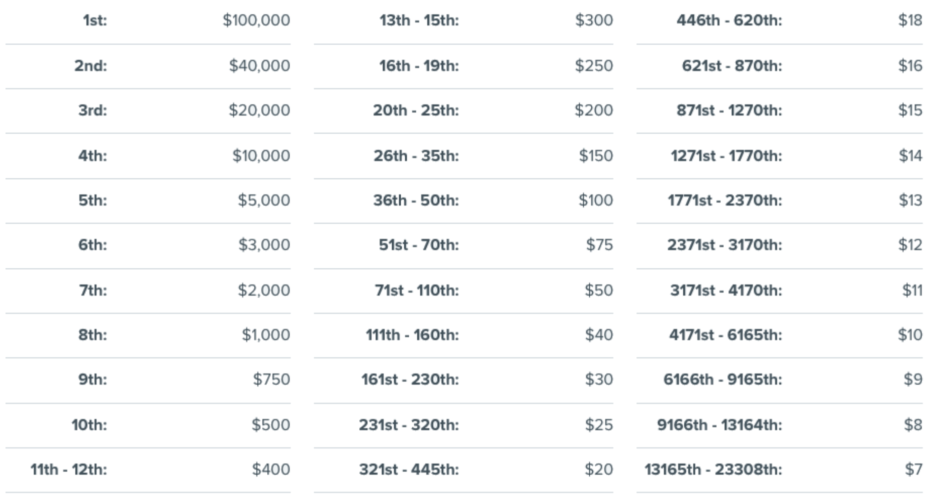
Daily competitions are different from the traditionally thought of fantasy season. They occur over a single day, where real basketball games are being played and involve only the games being played that day. A header for one of these daily competitions on FanDuel.com that meets our other criteria can be seen in figure 2. This header shows the low entry fee of $7, the high prize pool of $100,000, and that multiple entries are allowed. It also shows a list of the real games that are occurring that night.


Tournament CONTEST TYPE | 7223 / 16806 ENTRIES | $7 ENTRY FEE | $100,000 PRIZES | ● Multi-entry (150 max) | ● Guaranteed prize pool

ALL | ORL @ BOS 7:30PM | NY @ ATL 7:30PM | CHA @ CLE 8:00PM | TOR @ IND 8:00PM | MIA @ MIN 8:00PM | DET @ OKC 8:00PM | MEM @ DEN 9:00PM | NO @ PHO 9:00PM | CHI @ GS 10:30PM

Figure 2: Daily NBA fantasy competition header

The third criterion in the list specifies a "top-heavy" prize pool. This was one of the conditions of the Picking Winners paper, and is one of ours, as well. The meaning of top-heavy is that most of the prize-pool winnings go to the top contenders. This is different from a 50-50 contest, where the top 50% of competitors are given double their entry fee. With top-heavy competitions, "contests give a disproportionately high fraction of the entry fees to the top scoring lineups." [5]. The entry fees for these competitions are usually less than $10, and the top placing competitors usually win thousands of dollars. We aim for these types of competitions because we aim to submit many different lineups, each with a high probability of success. The payout structure of a competition with a $4.44 entry fee and $400,000 prize-pool can be seen in Figure 3. One can see that the difference between first and tenth place is 200 times, whereas the difference between tenth and one-hundredth is only 10 times.



1st:	$100,000	13th - 15th:	$300	446th - 620th:	$18
2nd:	$40,000	16th - 19th:	$250	621st - 870th:	$16
3rd:	$20,000	20th - 25th:	$200	871st - 1270th:	$15
4th:	$10,000	26th - 35th:	$150	1271st - 1770th:	$14
5th:	$5,000	36th - 50th:	$100	1771st - 2370th:	$13
6th:	$3,000	51st - 70th:	$75	2371st - 3170th:	$12
7th:	$2,000	71st - 110th:	$50	3171st - 4170th:	$11
8th:	$1,000	111th - 160th:	$40	4171st - 6165th:	$10
9th:	$750	161st - 230th:	$30	6166th - 9165th:	$9
10th:	$500	231st - 320th:	$25	9166th - 13164th:	$8
11th - 12th:	$400	321st - 445th:	$20	13165th - 23308th:	$7

Figure 3: FanDuel daily fantasy competition payout

The goal is for one or two of our entries to be on the top-heavy side of a competition (not every competition), offsetting the losses of the other entries. Another reason we feel that obtaining a high ranking entry is feasible is because we believe that our system, if successful, will return uncommon lineups that other methods (intuition or optimization) will not return.

3.1.2 Entries

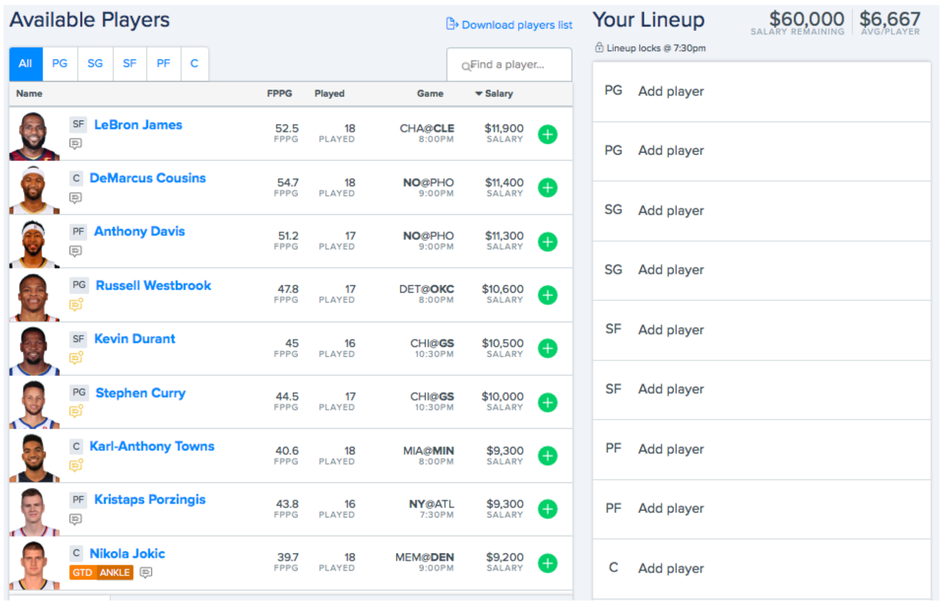
After selecting the competition to enter, the participant must create an entry, or lineup. This lineup must comprise players that are playing in the real games on that day. Additionally, the lineup must have a certain amount of specific basketball positions. A common requirement is that the lineup has to have two point guards (PG), two shooting guards (SG), two power forwards (PF), two small forwards (SF), and one center (C). Note that this may not be representative of a real basketball lineup. Figure 4 shows the lineup selection interface for FanDuel.com. On the right side, one can see the required positions for a given competition, along with the lineup's budget, in the top right corner. This budget is the amount of money each contender is allowed to spend on their lineup. On the left side, one can see the price of each player, along with their position, and their average fantasy performance for the season, on which the price is dependent.

3.1.3 Scoring

The participant's goal is to have the lineup with the highest score. The score is based on how well their selected players perform in real life, and is based on the amount of free throws, three pointers, field goals, blocks, steals, assists, and turnovers each player gets. A sample scoring system for competitions on FanDuel can be seen in Figure 5.

3.2 Requirements and Constraints

In order to solve the described problem, we had to comply with everything that was mentioned above. However, there are other requirements and constraints which are given below.



Available Players

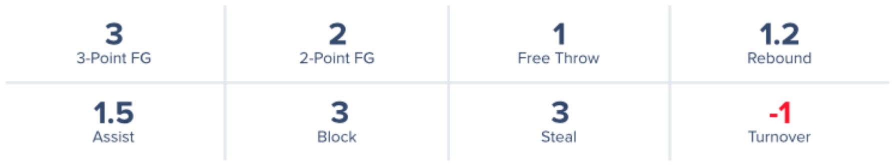
Name	FPPG	Played	Game	Salary
SF LeBron James	52.5 FPPG	18 PLAYED	CHA@CLE 8:00PM	$11,900 SALARY
C DeMarcus Cousins	54.7 FPPG	18 PLAYED	NO@PHO 9:00PM	$11,400 SALARY
PF Anthony Davis	51.2 FPPG	17 PLAYED	NO@PHO 9:00PM	$11,300 SALARY
PG Russell Westbrook	47.8 FPPG	17 PLAYED	DET@OKC 8:00PM	$10,600 SALARY
SF Kevin Durant	45 FPPG	16 PLAYED	CHI@GS 10:30PM	$10,500 SALARY
PG Stephen Curry	44.5 FPPG	17 PLAYED	CHI@GS 10:30PM	$10,000 SALARY
C Karl-Anthony Towns	40.6 FPPG	18 PLAYED	MIA@MIN 8:00PM	$9,300 SALARY
PF Kristaps Porzingis	43.8 FPPG	16 PLAYED	NY@ATL 7:30PM	$9,300 SALARY
C Nikola Jokic GTD ANKLE	39.7 FPPG	18 PLAYED	MEM@DEN 9:00PM	$9,200 SALARY

Your Lineup $60,000 SALARY REMAINING $6,667 AVG/PLAYER

Lineup locks @ 7:30pm

PG	Add player
PG	Add player
SG	Add player
SG	Add player
SF	Add player
SF	Add player
PF	Add player
PF	Add player
C	Add player

Figure 4: Fantasy basketball competition lineup



3 3-Point FG	2 2-Point FG	1 Free Throw	1.2 Rebound
1.5 Assist	3 Block	3 Steal	-1 Turnover

Figure 5: Fantasy basketball competition scoring

3.2.1 Functional Requirements

- The system should be able to use all previous NBA data
- The system should be able to output lineups for daily fantasy competitions based on which teams are playing, how much each player costs, and which positions are needed
- The system should output a projected score for each lineup

3.2.2 Usability Requirements

- The system should use a MySQL database
- The database should be secure and have personalized credentials for all users
- The system source code should be version controlled

3.2.3 Technical Requirements

- The system should be able to train on the order of magnitude of minutes
- The system should be modular, allowing for quick and easy changes

- The system should allow NN parameters and other parameters (data amount) to be changed quickly for testing

3.2.4 Constraints

- The system must not require any expensive services
- The system must be trainable on our personal computers
- The system must be completed within eight months (two university terms)

4 Design and Results

There are two major sections into which the design solution can be divided. These two section are data and implementation.

4.1 Data

The data section of our design solution includes anything relevant to data, including collecting, storing, and accessing the data.

4.1.1 Collecting Data

Initially we had two candidates for the primary source of obtaining all nba player and team related statistics. These two sources were:

- stats.nba.com
- basketball-reference.com

Both sources contained similar amounts of information on player matches, player season averages, teams, matches, with basket-ball reference being slightly more detailed in aspects such as player season averages and player matches. While both sources were credible, we decided to pursue stats.nba.com as the primary source as we felt the official statistics platform of the NBA would have more reliable data compared to one that is not affiliated.

The collection of data from stats.nba.com is done by finding a webpage with pertinent data, scraping that data using Python, and storing it in a MySQL database. We automated the process of scraping the data by writing python scripts that iterate through all web pages of a specified type and searching through HTML tags to retrieve the desired data. This usually entails locating and HTML table, iterating through it's rows and columns, and retrieving stats for points, assists, rebounds, etc. The two fundamental tools used in the scraping process were the python libraries:

- BeautifulSoup4
- Selenium

BeautifulSoup allows for objectification of HTML tags on a web page such that can be accessed through Python and Selenium automates the usage of a chosen browser. While, in most cases, BeautifulSoup alone would be enough to simply access the HTML elements of a given web page, stats.nba.com generates all of the data that we need through JavaScript. This causes a problem for us, as BeautifulSoup has a feature to wait until a page has fully loaded, but this does not take into account JavaScript components. When Selenium is used to automate browser control, we can specify that a page is not fully loaded until all of it's JavaScript components have fully loaded.

4.1.2 Storing Data

On the topic of data storage we, once again, had 2 major choices with their respective advantages and drawbacks. The methods of storing data were:

- local database, local files
- remote database

The method of using a local database or local files was enticing due to the lack of potential bottleneck associated with the connection bandwidth of a remote database. However, we knew that the convenience afforded by using a remote database would be more beneficial in the initial data collection process.

While employing the methods to collect data from stats.nba.com, we knew that there existed the potential for database schema changes as we discovered what was available to us and what we wanted to collect. In the case of a local database this would have created an inconsistency in the local copies for each group member.

Another benefit to using a remote database, at least initially in the data collection process, is the ability to update one persistent database in parallel. Which is to say, multiple people can be using to multiple tables at once to insert or delete entries as more data was collected or data was determined to be obsolete. This convenience is more beneficial than the cost of overhead in using a remote database for as long as data is still being collected.

4.1.3 Accessing Data

Since we decided to employ a remote database for data storage each group member is able to access the data through ssh. Each member has their own credentials through which they can access they database from any location that has access to wi-fi.

4.2 Implementation

We began designing the software architecture before finishing the aforementioned database design. Our initial design was a single neural network that took in, as input, all of the stats of every active NBA player. Additionally, it had an extra input for every player which was set to 1 if the player was playing tonight, and 0 if the player was not. The goal of this neural net was to output the best lineup for the competition. A block diagram of this system can be seen in Figure 6.

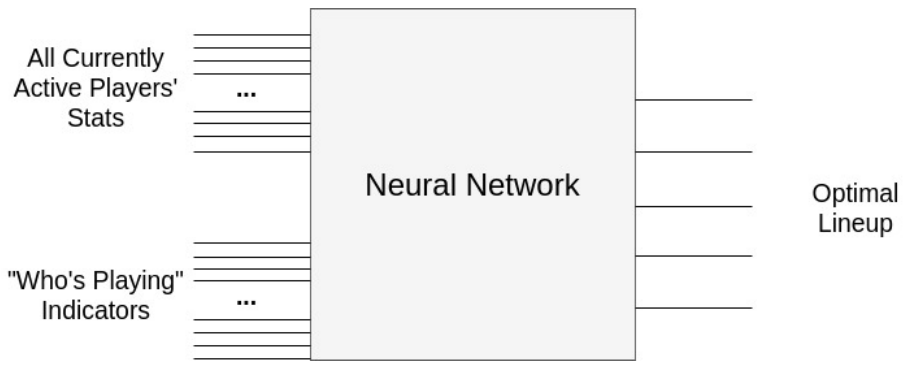

All Currently Active Players' Stats
...
"Who's Playing" Indicators
...
Neural Network
Optimal Lineup

Figure 6: First design iteration

After discussing this design within our group, and with our supervisor, we realized it had some limitations. First, this system would only use data from active players. This was not taking advantage of the many decades of basketball data that we had access to. Second, it would also likely require a massive amount of data to be able to learn properly. This is because, generally, the amount of data required to learn adequately is correlated with the amount of inputs and complexity of inputs. Third, there would be many unused inputs each time the net is used. Since there are usually only about four NBA games each night, there are only about 80 players active on a single night. There are about 500 total active players, which means that there would be about 420 "unused" inputs. Although this is not explicitly a problem, it makes it seem like there would be a better way to structure the design. The final limitation of the initial neural network that we thought of was that it would be prone to discover "bad" correlations. Since the net only knows which players are playing, and not on which team, it would be bound to find correlations between players that are not playing against each other. To explain, say that NYK is playing against BOS in one game, and GSW is playing against CLE in another. The neural network may find a relationship between someone on BOS and on GSW, even though they are not playing against each other, and games are independent. Taking these limitations into account, we decided to come up with an improved architecture that was more flexible and required less data to train. We decided on a three step process, of which only one of the steps would involve machine learning, to start. The three steps are to calculate the player scores, use them in a NN to find good players, and then use a third system to choose lineups. An overview of this software architecture can be seen in Figure 7.

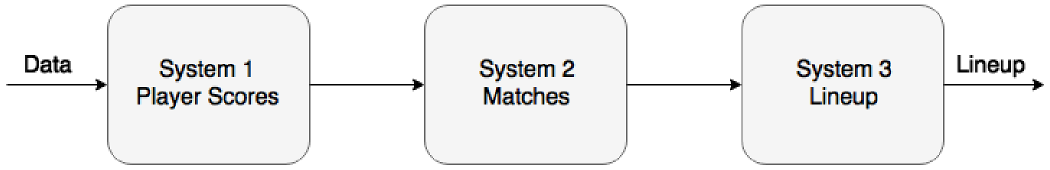


Figure 7: Redesigned System Architecture

These three systems will now be looked at in detail.

System 1: Calculating Player Scores

The first step of the design solution is to calculate a score for each player who has played in the NBA. This score is ideally supposed to be a representation of how good the player is, "in a vacuum". That is to say, how well a player will perform if we ignore the match-specific variables like who is on his team, or who he is playing against. Getting a score that represents this is not an exact science. We were initially going to use player rankings given by experts, but could not find historical data for this. Thus, we knew we had to come up with a system of our own to generate these scores. Since we wanted to spend more time focusing on the second system (NN), we decided to initially use a simple equation here based on the player's season statistics, as opposed to something like a neural network. The first equation we decided to use was actually the equation for getting fantasy basketball scores for season fantasy basketball. We postulate that there is a clear correlation between the amount of fantasy points a player would get, and how good of a player they are. Although this is a simplification, it is a good starting point, and due to the modularity of our design, we can always improve it or change it, as long as the inputs and outputs stay the same.

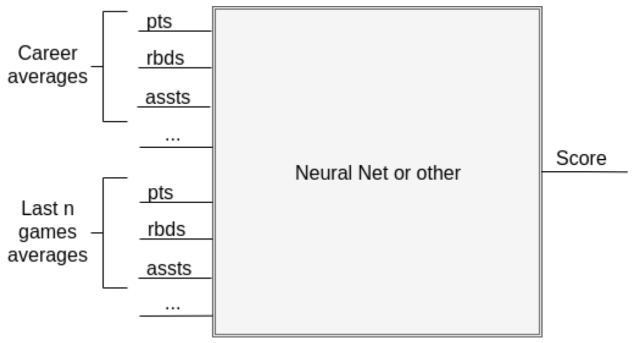


Figure 8: Player scores system

System 2: Matches Neural Network
The scores for every player, obtained in the first step, are then used as inputs for the matches neural network, the portion of the implementation side of the project that has seen the most development in the first half of the project timeline. The matches neural net puts two teams of players, each represented by a score, "against each other". That is to say, the first ten inputs are used for Team A's players, and the next ten are used for Team B's players. The output of the net is then a "fantasy game score" for each player that should be higher for players who would get higher fantasy scores. The idea behind this net is that, since all basketball matches follow the same rules, with the only (or at least most impactful) variable being who is playing on each team, we can train this net on every match that has happened in the past 50 years, and still use it on matches today. This net has a minimal amount of inputs, none of which will be unused. This means it needs less data to train successfully. This design can be seen in Figure 9.

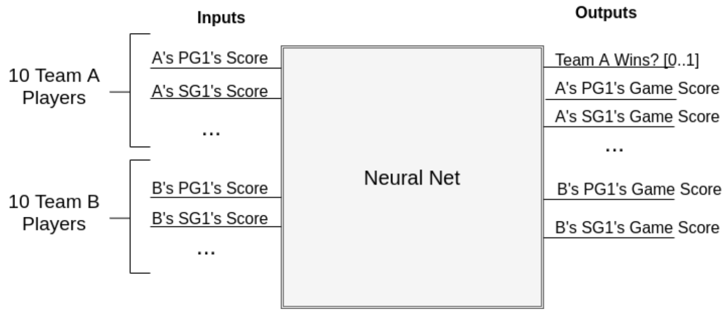


Figure 9: Desired complete matches neural network

To minimize the debugging phase of this NN, we decided to start with a simple version, and then iterate towards the final version. The more simplified version (the first prototype) output which team would win, instead of the players' game scores. This neural net was thus not applicable for fantasy basketball, but was used instead to become familiar with TensorFlow and best practices. In building this early prototype, we decided to start with a further simplification, namely to only have four inputs: the average player scores and their distribution for each team. Thus, the diagram in Figure 9 was simplified to the one seen in Figure 10.

The justification behind the inputs and outputs is that the average scores of the players on each team should help to determine which team will win or lose. We found that this neural network was better than random chance at predicting which team would win, with around a 63% success rate (see the Results subsection for more detail).

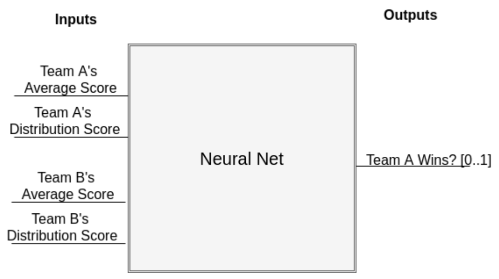


Figure 10: Matches neural network with averaged player scores

We then undid this "averages" simplification, and reverted to having separate inputs for each player. However, since teams do not always play ten players, we used seven instead. We ordered the inputs for each team by the score, so that the first input corresponded to the player on Team A with the highest score. We felt that having ordered scores would help the net learn better, as it would likely learn to put a heavier weight on the higher scores (better players have more impact). This was found to be true in a couple of tests where we used randomized inputs, which were found to be less successful. Using ordered player scores, we had a success rate of around 64%. The diagram for this net can be seen in Figure 11.

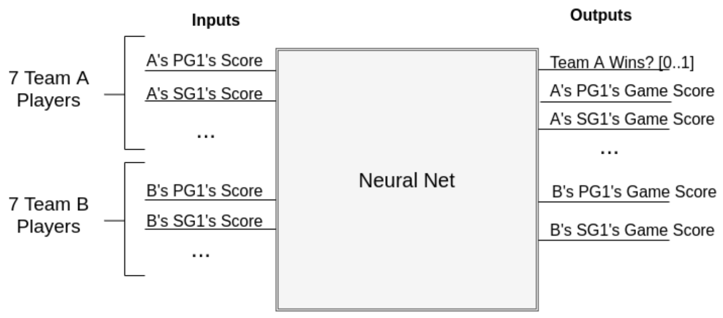


Figure 11: Matches neural network with ordered player scores

We decided to iterate towards our final goal once more, and did so by ordering the players on each team by their basketball position. The basketball positions given by our source are guard (G), forward (F), and center (C). As not each team plays the same amount of players in each position, we had to determine what a representative combination of the above positions would look like. This analysis will be discussed in the Data Results section, below. Ultimately, we decided that a representative set for which we had data would be one center (first input), two guards (second and third inputs), and two forwards (fourth and fifth inputs). This net was the same as the initial plan in Figure 9, except with the single winning output rather than the player game scores.

The final aspect of the neural network that we changed in hopes to see better results was the structure of the output. Instead of using a "digital" output, where the game was either won (1) or lost (0), we decided to use an "analog" output. In this case, winning a game by a high amount of points would be close to an output of 1, and losing by a high amount would be close to an output of -1, but values in between are also possible. We felt that this made the most sense, since we felt that our network should be trained on the point difference between teams to yield more confident wins and losses. To decide how to map the point difference in a game to [0,1], we first discussed with people familiar with the sport what the difference is between a team winning a game by 20 points versus winning by 2 points. We came to the conclusion that, generally speaking, if a team wins by 15 points, they won rather handily, and anything closer to 0 points could have gone either way. With this in mind, we decided to use a scaled arctangent function, such that, around a difference of 15 points, the output is around a value of 1, and that it tapers off after this. The equation used for this can be seen in (Eq 1). The shape of this mapping can be seen in Figure 12. One can see that this function can output a value greater than 1 (at a point difference of about 25 points), but this was not a great concern. Even with a point difference of 50 points, which is unlikely, the output is only 1.27.

$$Output = arctan(pointDifference/15) \tag{1}$$

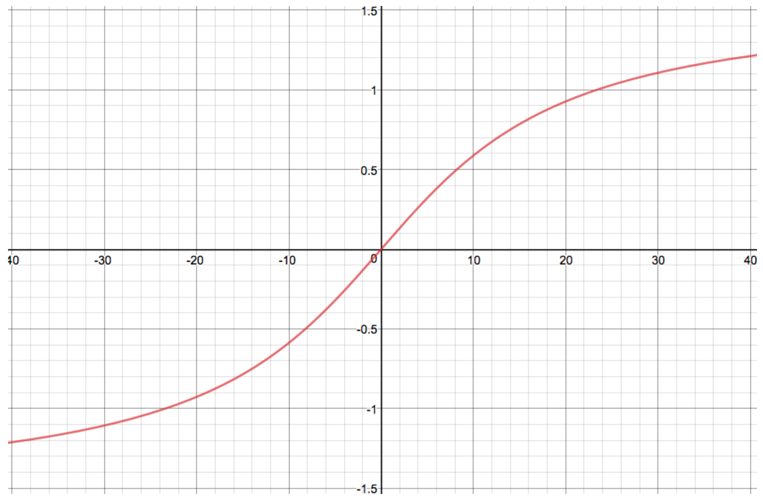


Figure 12: Shape of modified arctangent used for mapping point difference to output

The results of the final version of this prototype can be seen in the results section. This is not the version that will be used for our final project; the next step will be to change the outputs from which team wins to player game scores. This will be discussed in the Future Plans section.

System 3: Choosing Lineups
The choosing lineups system will be the system that actually outputs lineups to enter into the fantasy competition. Although this system has not been built yet, the inputs and outputs have been designed. It will take in the properties of the fantasy competition, such as the budget, the positions needed, and the teams playing, as well as all of the players' game scores from system 2. Using these inputs, it will output a set of lineups that meet all the necessary competition constraints (e.g. below the budget), that maximize the overall score.

4.3 Results

There are two types of results that will be given in this report. The first are those related to how much data was scraped from our source, or generated, and the second is related to the success rate of the neural network.

4.3.1 Data Results

We managed to scrape, collect, and otherwise compute a significant amount of data to be stored in our database. We were able to gather all data for every basketball game since 1979. We have data for 42000 matches, 10000 player-seasons (unique players for a specific season), 2000 players, and 36 teams, including some that are no longer active. For each match, there are about 15 players with statistics for that match. This is entered in our "player matches" table, and we currently have data in this table for about 25% of matches. This corresponds to 250,000 entries, and yields about 13000 inputs for our neural net. However, the different architectures which have been presented all impose different constraints on the inputs themselves. For example, the neural network with ordered player scores shown in Figure 11 requires the match input to have both teams with at least the minimum number of players (7 in this instance). The neural net with players ordered by position has even more constraints. Not only does it require a minimum number of players per team, but also it needs a minimum number of players per *position*. When performing an analysis on the different player positions per lineup, we found significant variation between teams. This was even further complicated as some players played multiple positions, denoted by two positions separated by a hyphen. For example, the position "F-G" meant that the player could either play a forward or a guard. We found 21 possible combinations of these positions that teams could play. As an example, playing "C, G, G, F, F, C-F" could be a single combination, whereas "F-G, C, C-F, F, F, G, G" could be another. We found that none of the possible 21 combinations had enough data to train our neural network on. Even our initially designed input format of requiring two guards, one center, and two forwards per team (meaning only five inputs per team) caused roughly 21% of inputs to be discarded, simply because these inputs did not meet the criteria of having these specific numbers of players in each positions.

To avoid losing significant amounts of input data due to constraints, we simplified our input scheme by combining the multi-position players with the single-position players. Our methods of combining these players is shown in Figure 13 below. In using this scheme, we were able to reduce the amount of lost data to a mere 11%.

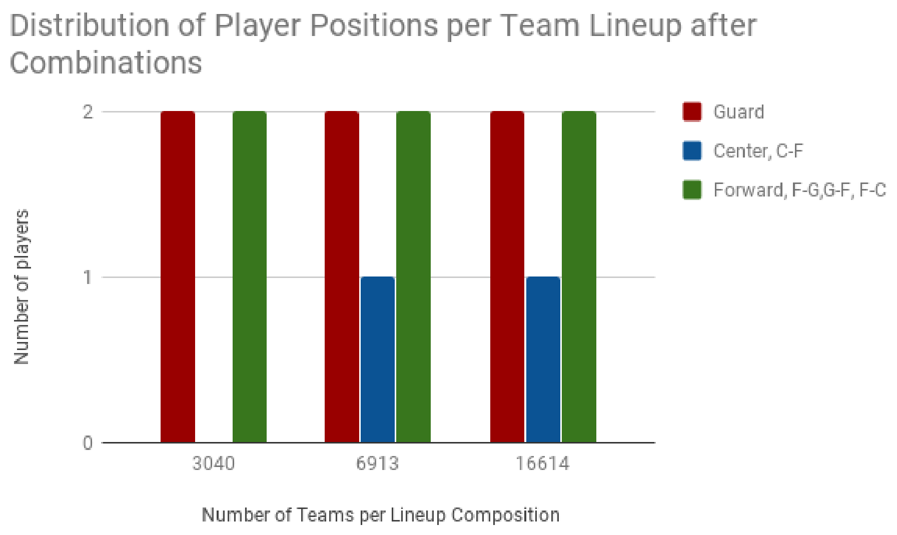


Figure 13: Team Compositions by Player Positions After Combinations

4.3.2 Implementation Results

As mentioned in the description of the neural network in the implementation section, we went through a few different architectures. Although the final architecture will be one that takes in the player scores in order of position, and outputs game scores for each of the players, this has not been built yet. However, for the first three architectures, we have collected some results. The main result collected is the win-loss prediction success rate, as related the amount of data used and the architecture of the inputs and outputs. All other ML parameters, such as learning rate and number of hidden nodes, remained unchanged. The neural network has only one hidden layer for simplicity. Furthermore, all accuracy results were obtained by using 70% of the input data to train the network, and 30% to test.

The first architecture that we tested was the one shown in Figure 10. Because the input of this neural network consisted solely of averages and variances for each team, we were able to use all of our data (100%) to train and test the network. The highest testing accuracy was around 63% for this architecture, as can be seen in Figure 14, which shows the success of the neural net versus the epoch number, or the number of times the entire data set has been passed through the neural network. As the epoch increases, the accuracy should converge. It does not always converge asymptotically, and in these cases, the method for determining when a network has reached its true accuracy rate is when the training accuracy separates from, and surpasses, the testing accuracy. In these cases, the neural network is overfitting the training data, and achieving higher accuracy rates, but it is not representative of a real relationship.

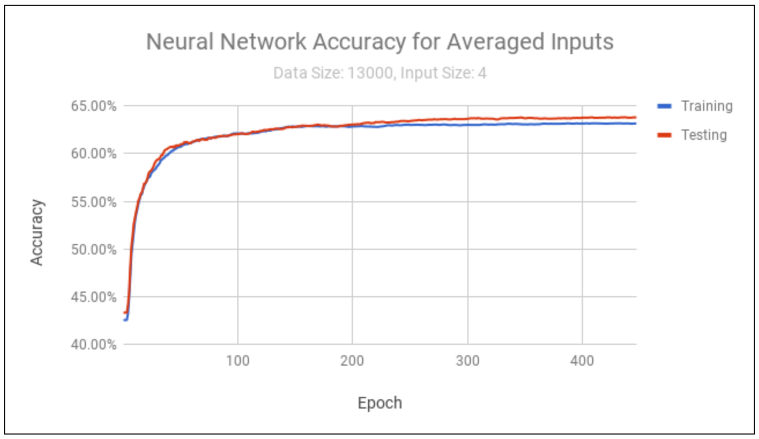


Figure 14: Accuracy of network for averaged player scores inputs

After testing the averages, we tested the version of the neural network where the inputs were the actual player scores on each team, ordered from highest to lowest (previously shown in Figure 11). We tested this with two different configurations; first by taking the top five players on each team, and then the top 7 players on each team. The accuracy for both of these scenarios can be seen in Figure 15 and Figure 16. One can see that having more scores as inputs does increase the accuracy, and by a significant margin in this case. However, it is also worth noting that this increase is also tied to the increase in data between the two cases (9000 vs. 10000). When both are set to use the same amount of data, the improvement is less significant.

As mentioned earlier, in an attempt to further increase the accuracy, we switched to having 5 inputs (or players) on each team, but ordered by their position. To reiterate, we grabbed the highest two guard (G) scores, the highest two forward (F) scores, and the highest center (C) score,

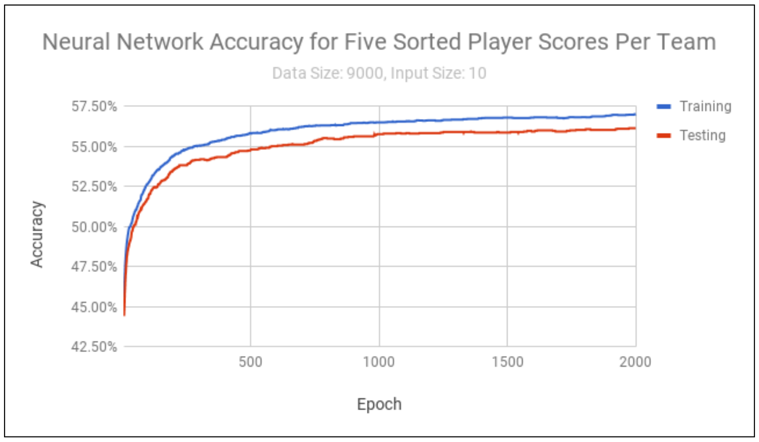


Figure 15: Accuracy of network for 5 player scores inputs per team

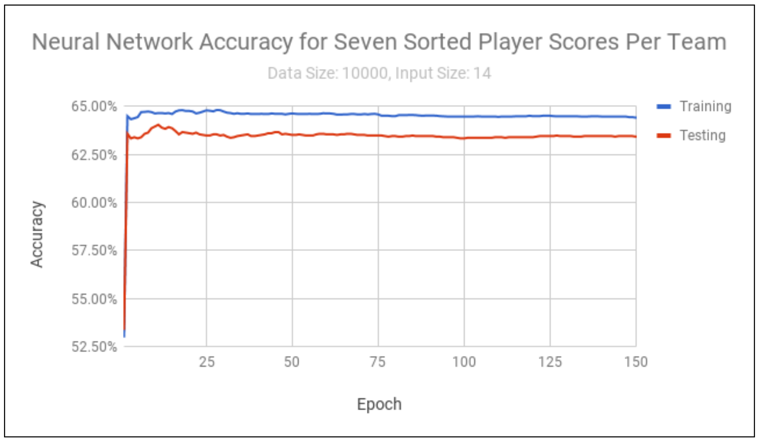


Figure 16: Accuracy of network for 7 player scores inputs per team

and used these as the inputs per team, in that order. The accuracy for this network can be seen in Figure 17. One can see that this did not increase the accuracy of the net by a significant amount. However, we believe that this is the best structuring of inputs to use, as it allows the network to learn that the first position is always tied to a guard, which may have more or less impact than other positions (as opposed to using ordered scores). We believe that the reason we did not see a significant increase in the accuracy is because, with only 5 players per team, and sorted by position, we often cut out some top players. Ideally, we would be able to select a more representative position set (as opposed to G, G, F, F, C) that would allow us to use more inputs (players) for the net.

In addition to changing the structure of the inputs to try to obtain higher accuracy rates for our neural network, we also worked on changing some of the other parameters. First, we tried to check the impact of using more or less data. We found that the impact could be quite significant, and that, usually, more data led to a higher accuracy rate. An extreme version of this can be seen in Figure 18. However, there was one case in which only using the most recent two thousand games actually yielded a higher accuracy. This could just be that the sample size was too small to gather any meaningful conclusions from, but it could also imply that using data from ten years ago for current games may be a bad idea, as basketball may have changed too much. This is something

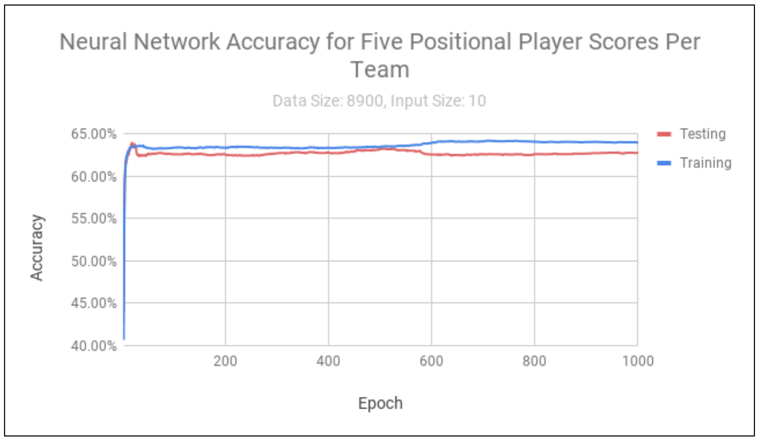


Figure 17: Accuracy of network for positionally ordered score inputs

that will be explored when we build the next and final neural net architecture.

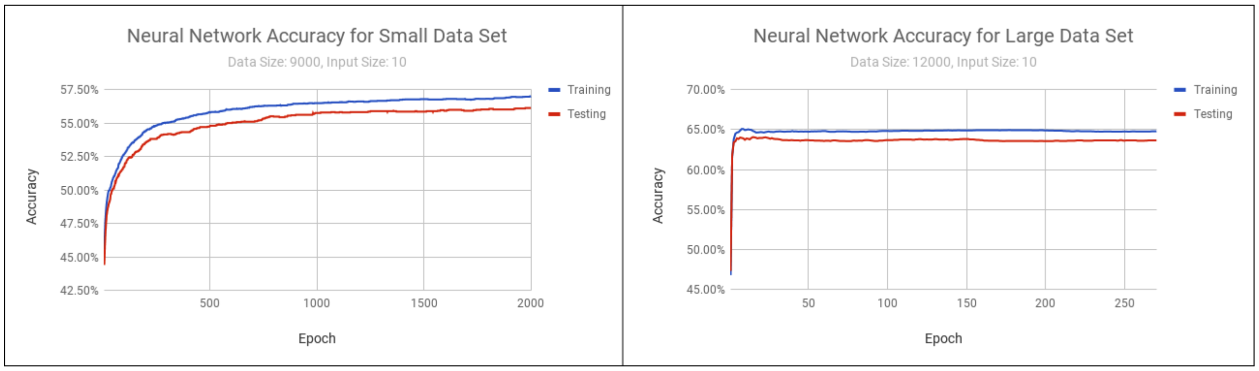


Figure 18: Accuracy of neural network vs. amount of data used

The next parameters that we wanted to alter were those related to the neural net's properties, namely the number of hidden layers, hidden nodes, and learning rate. We changed these by hand and saw changes to the net's accuracy, and were able to manipulate these to make minor improvements. We did not optimize these parameters thoroughly, as it was out of the scope of this semester, and since this neural net was only a prototype. As will be discussed in the Future Plans section, we did find a way to optimize these parameters, namely n-fold cross validation.

5 Future Plans

The design phase and data collection phase are mostly complete, but there is still a significant amount of building and testing that need to be completed before we have a full solution. What still needs to be done can be seen below.

5.1 Improved Neural Net

As mentioned in our Design and Results section, our current neural network takes in as input the scores of the players who are playing in a match. It then outputs which team it thinks will win the match, with a probability attached to the output. Although this was the final solution of this term, it is still not sufficient to solve our problem. We need to be able to output "game scores" for each player, instead of which team wins. The final neural network should take in the player scores, and output the player game scores. The distinction is that the game scores are related to how well the net thinks that the player will actually play in that game, which is ultimately all that matters for the fantasy competitions. Once we have these game scores, we will then pass these to the third system, our lineup chooser. Another improvement we can make with respect to the neural network has to do with the NN parameters. These parameters include the learning rate, the amount of hidden layers, the amount of nodes in each hidden layer, and the amount of data used. There is a technique called "n-fold cross-validation" which we plan to use to optimize these parameters. It involves testing different sets of values of these parameters to determine the values that yield the highest success rate.

5.2 System 3: Lineup Chooser

Currently, we do not have a system that chooses lineups out of the players playing. After modifying the neural network to output the game scores, we will have a list of players ordered by how well we think they are going to perform tonight. With no constraints, we could simply take the highest scoring players. However, as mentioned in the Requirements section, fantasy basketball imposes constraints on the lineup. The first constraint is the budget of the lineup. Each player will have a cost, and our system will have to be able to generate a high-scoring lineup while keeping the total cost of the players under the total budget. The next constraint is that we have to select players with specific basketball positions. Thus, the system will also have to take in these positions and be able to generate lineups with it. Finally, as there will likely be many possible lineups with similar estimated scores, the system will need to have a random aspect to it that allows it to generate different lineups. This system will likely initially be implemented as a simple algorithm, but may have to be adapted, and could even end up being implemented as another neural network.

5.3 Player Playtime

One important statistic that we are missing from our system is the player playtime, i.e. how much the player plays each game. This is clearly an important factor in determining his score, since even if a player has a high per-unit-time score (e.g. 5 free throws per minute), if that player only plays for a minute, his score will be low. Although we have this data in the database, we do not use it. The reason for not using it is that it cannot (currently) be used as an input for the neural network, since we do not have a way to predict how long a player will play for in a game in the future. That is to say, we would not be able to use our neural network for an upcoming game, as we would have missing data. There are two approaches to remedy this. The first is to create a system that estimates the playtime for a player for a future game. This system could either be a simple algorithm, like "average the playtime that player has received over the past n games". Alternatively, an argument could be made that this system could be implemented with some sort of predictive machine learning. The second approach would be to use the playtimes in calculating the player's career score. Instead of adding playtime as an input to the neural net on a per-match basis, we would change the players' scores that are used as inputs for the neural net to be multiplied by how much they played last season, or during the last n matches, on average. This is perhaps too gross a simplification, since the playtime may change on a more match-per-match basis, so both of these approaches will have to be explored.

5.4 Testing

Once the above tasks have been completed, we will be at the testing stage of our neural network. We will first test it for games in the past, and see how our lineups would have done in contrived fantasy competitions. The testing process is detailed as follows:

If, after performing our tests, we see that there is an adequate (i.e. profitable) success rate, we will move to test it on real fantasy competitions. However, if we see that there is not an adequate success rate, or if we want to improve the already adequate success rate, we will have to make adjustments. We already have many ideas for adjustments that we can make, including the following:

- **Improve player scores system (system 1):** Currently, our first system takes a player's statistics and outputs a score representing how good that player is, "in a vacuum" (i.e. not against any particular team). The current equation we use for this is just the fantasy scoring equation. Although there is certainly a correlation between fantasy score, and how good a player is, it is likely not the most accurate relationship. For one, the fantasy score does not take advantage of all of the information we have, such as the player's age, shot accuracy, the player's win-loss record, and the player's plus-minus, which is an indicator of how well a team does as a whole, when the player is on the court. The new system could either take all of these factors into account, or could outsource to use someone else's equation/algorithm. An argument for making the latter decision is that we do not know more about basketball than the experts do, and they can likely come up with better numerical representations of how good a player is than we can. One argument against this decision is that the information is hard to obtain, and in order to train the neural network on past games, we would need to know the details of the algorithm to be able to calculate the score for these older games (as opposed to just getting the output).

- **Improve player playtime:** As mentioned in the Player Playtime subsection above, our initial solution to playtime will likely be to make the assumption that a player's playtime during the previous year is a good estimation of their future playtime. We also mention that a better solution could be designed that takes advantage of the playtime per game data. Moving to this second solution could help improve results.

- **Obtain more accurate positional data:** Currently, our source, NBA stats, gives us three posible positions: guard (G), center (C), and forward (F). The positions required for fantasy competitions are more specific, such as shooting guard (SG), or small forward (SF). Collecting this more precise positional data could lead to a higher success rate.

6 Impact on Society and Environment

6.1 Use of Non-Renewable Resources

Seeing as our entire project is made up of software components, the environmental impact, in particular the use of non-renewable resources, is fairly minimal. Having said this, the training aspect of the neural network certainly requires processing power, which consumes energy. The amount of energy consumed by a CPU or GPU varies greatly, but, for high end hardware, the total power usage can reach 500 W [6]. As the amount of data the net is trained on increases, and the amount of times we retrain the net increases, so will the amount of time the computer runs for, and thus so will the total energy consumed. The amount of energy used by our system, and thus the total environmental impact, will be directly proportional to the amount of times the network is trained by all users. Theoretically, if this system was to be released to the public, then the accumulation of everyone training their respective neural networks could eventually become an environmental concern. With one thousand users training a net for an average of ten hours, we would see an energy consumption of about 18 GJ, which is equivalent to charging one million iPhones [7]. Although this may seem insignificant, with one million users, the energy used would be equivalent to charging one billion iPhones. Currently, since we are in Quebec, the energy being consumed is almost entirely made up of that generated with renewable resources. However, most of the planet's energy is produced via non-renewable resources[1]. Thus, if the project ends up growing to this scale, it would be important to continue to consider and evaluate the use of non-renewable resources.

6.2 Environmental Benefits

If our technology was developed to the point at which it could be used with relatively low power requirements (e.g. no need for each user to train the neural network), then widespread use and adoption of our software could potentially reduce the amount of energy consumed by the fantasy sports community. Curretntly, fantasy sports participants spend significant amounts of time online trying to figure out how to improve their fantasy teams; some spend several hours a day doing such research [8]. If our system was adopted by a large enough portion of the people playing fantasy sports, and could be run by their computers easily, it could remove all the research and decision making from the hands of these participants. This could then in turn mean they will spend less time on their computers managing their fantasy teams, thus using less energy. This is a somewhat special case, as we would have to develop a version of our product that comes pre-trained, runs with low power, and is distributed in such a way that the user does not use more energy downloading and setting up the software than he/she saves with it. Currently, this is out of the scope of the project.

6.3 Safety and Risk

The most significant risk associated with our project will actually be posed to the fantasy sports industry itself. If we are able to successfully use machine learning to predict the optimal line-ups for fantasy sports, then the industry may switch from a human-human competition to an entirely ML one. After releasing the software, there could also be a high amount of people who were not previously playing fantasy sports who may now see it as profitable. These people may flood in and take over the game. After a while, however, it would no longer be profitable, since everyone playing would have the best, and equivalent, equipment. As well, more traditional fans may argue that it removes the fun or skill of fantasy sports, and may stop playing. If the project turns out to be exceptionally successful, and is released to the public, it could completely change the way that fantasy sports is played, and we see this as a risk to that industry. However, we also feel that it is unlikely that the system is able to outperform all humans, and is certainly unlikely that it would be adopted in such a way as to ruin the industry.

6.4 Benefits to Society

There are a few benefits to society that could result from this project, if it is successful. The first benefit of our project would be to hopefully help people become interested in neural networks and machine learning in general. We argue that this is important since more and more devices use machine learning techniques, and this would lead to a better understanding of the current technology. The second benefit is that it could help further the movement of data-backed sports hiring decisions, which is to use statistics for selecting players, rather than intuition. Michael Lewis writes about this method, and its success, in his book Moneyball [9]. The reason this is a benefit to society is because it could make the hiring process for sport players more blind, and more friendly towards minorities. The third benefit is related to expanding the list of domains to which machine learning techniques can be applied. If we are able to demonstrate a successful system for predicting optimal fantasy lineups, then others may try to apply a similar technique to a different problem. The problem could be something more meaningful, such as a pharmaceutical company determining how to select an optimal drug, which is an example mentioned in Picking Winners, by David Hunter et al [5].

7 Teamwork Report

7.1 How We Worked as a Team

In order to facilitate and improve the efficiency of teamwork within our group, we made use of several tools and strategies throughout the semester. For communication within the group, we used the Slack messaging app. For collaboration on code, we made use of git for version control, and github for code reviews. Weekly in-person meetings were had amongst all team members in order to come with fresh project ideas as a group, and to ensure that all of us were on the same page. Weekly in-person meetings with our supervisor were also had for similar reasons, and to ensure that we stayed on track throughout the semester. Without the use of these tools and strategies, the development process for this project would have been much more difficult.

7.2 Individual Contributions

Below we show the individual contributions of each person. Note, however, that much work was done as a group, so there are many areas that were worked on collectively by everyone.

7.2.1 Ege

- Worked on designing the database schema for the database that would house all the data being collected for the project.
- Implemented the scrapers that would retrieve the player and match data from stats.nba.com.

7.2.2 Stephen

- Worked on algorithms that compute player scores, both historical and for a given game, and worked on writing scripts to insert given data into database.
- Performed data integrity analysis to ensure that all data received made sense, and if something was wrong, sought to find out what was wrong with is and what was missing.

7.2.3 Florence

- Built preliminary neural network that predicts which team will win a given basketball match.

- Worked on overall architecture design of three-part prediction system.

7.2.4 Asher

- Worked on scripts to compute player scores, and scripts to connect tables in the database.
- Worked on data retrieval code from database to obtain data in a format appropriate for the TensorFlow neural network.
- Helped to implement scrapers for some player statistics.

7.3 Evaluation of Teamwork and Difficulties

The four of us worked very well together as a team and have yet to have any problems thus far in the project. The number of difficulties faced this semester were minimal since we worked so well as a group. We hope to continue this same good work through next semester.

8 Conclusion

The first half of our design project has been a success. Although we have not yet created a full solution to the problem of picking winning lineups for daily fantasy basketball competitions with machine learning, we are on track to do so with enough time to test and reiterate on our design. We were able to gather a sufficient amount of data to train and test a first prototype that was able to predict the outcome of a basketball game. We also found that we were able to increase the success rate of this prototype by adjusting ML parameters, by increasing the amount of data used for training, and by altering the fixtures to be more meaningful. Our best version was able to successfully predict the outcome of NBA games with 71% accuracy. As the ultimate goal of our project is to be able to generate lineups, and not to predict the outcome of games, we still have work to do. We will have to change the NN to output the players' game scores, and then subsequently create a system to take in these game scores, the cost of each player, and the lineup budget, and output optimal lineups. After doing this, we will have to test our system, and improve certain aspects of it, such as how we obtain player scores. The end result will hopefully be a working and well-tested system that is able to generate profitable lineups for fantasy basketball competitions.

References

[1] "How much of U.S. energy consumption and electricity generation comes from renewable energy sources?", U.S. Energy Information Administration, 2017. [Online]. Available: https://www.eia.gov/tools/faqs/faq.php?id=92.

[2] Karpathy, "Convolutional Neural Networks for Visual Recognition", Stanford. [Online]. Available: http://cs231n.github.io/neural-networks-1/.

[3] Abu-Mostafa, Yaser. "Learning From Data", Caltech, 2012. [Online]. Available: https://work.caltech.edu/telecourse.

[4] "A Basic Introduction To Neural Networks", University of Wisconsin-Madison. [Online]. Available: http://pages.cs.wisc.edu/ bolo/shipyard/neural/local.html.

[5] Hunter, David Scott, et al. “Picking Winners Using Integer Programming.” *ArXiv:1604.01455 [Stat.OT]*, 6 July 2016.

[6] Kaminski, J. (2009). "How much power does your video card use?." CNET. Available: https://www.cnet.com/news/how-much-power-does-your-video-card-use/.

[7] Mayo, B. (2015). iPhone 6s Plus battery rated 2750 mAH, 5% smaller capacity than iPhone 6 Plus. 9to5Mac. Available: https://9to5mac.com/2015/09/21/phone-6s-plus-smaller-battery/.

[8] YouGov. "How Much Time Do You Spend on Managing Your Fantasy Football Team?*." Statista - The Statistics Portal, Statista, www.statista.com/statistics/284718/amount-of-time-spent-on-managing-fantasy-football-team/, Accessed 30 Nov 2017

[9] Lewis, Michael. "Moneyball: The Art of Winning an Unfair Game". New York: W.W. Norton, 2003.