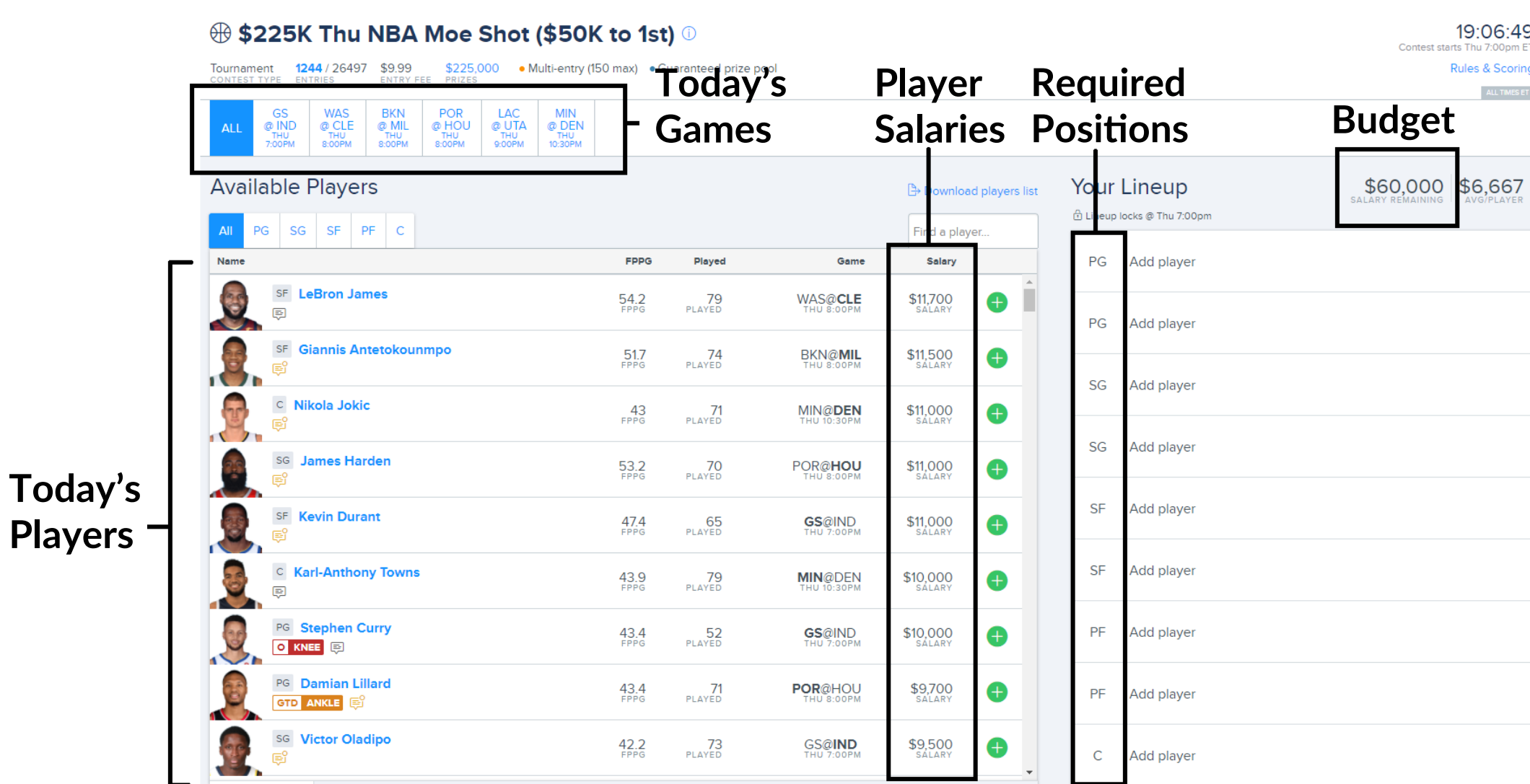


1 Objective & Background

The goal of this project was to make a machine learning system that could compete profitably in **daily fantasy NBA competitions**.

- **Daily** means the competitions happen every day that there are matches (not fantasy leagues)
- **Fantasy** means that they use a contrived points system
- **NBA** means that they are based on actual NBA games
- **Competition** means we are competing against real people

In these competitions, you have to select 9 basketball players of specific positions (e.g. center) who are playing tonight in real games, across any teams. Each player has a salary, and you have a budget. The goal of the competition is to maximize the amount of fantasy points your lineup gets.^[1]



Our goal was to build a machine learning system to predict these lineups and maybe even win some money (unlikely!)

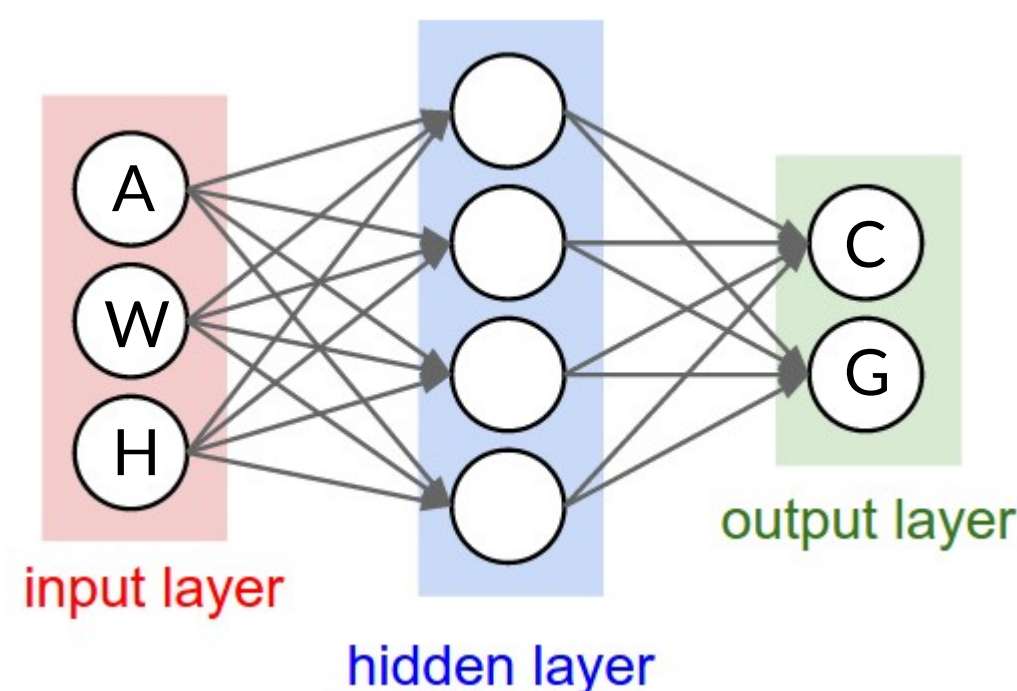
2 Neural Network

We decided to use a neural network (NN) to solve our design problem. A NN is useful under the following criteria, which our design problem met:

1. There has to be a distinguishable pattern between the inputs (player stats, games) and the outputs (lineups) of the system
2. The relationship cannot be trivially represented mathematically, otherwise no reason to use machine learning
3. There must be enough data to properly "train" the network

But what is a neural network?

Below is a graphic of a simple NN, with one hidden layer, two outputs, and three inputs.^[2] This network, for example, could be attempting to find the relationship between the age, weight, and height of a person and their country of birth and gender.



There are two main "stages" of a Neural network.

1. Feed forward, or how outputs are computed from inputs

When feeding an input through the network, every edge (line) is a weight to multiply the value by, and every node (circle) is an "activation function" on the incoming value.

2. Back propagation, or how the system learns

Once an input is fed through the network, the error can be calculated.

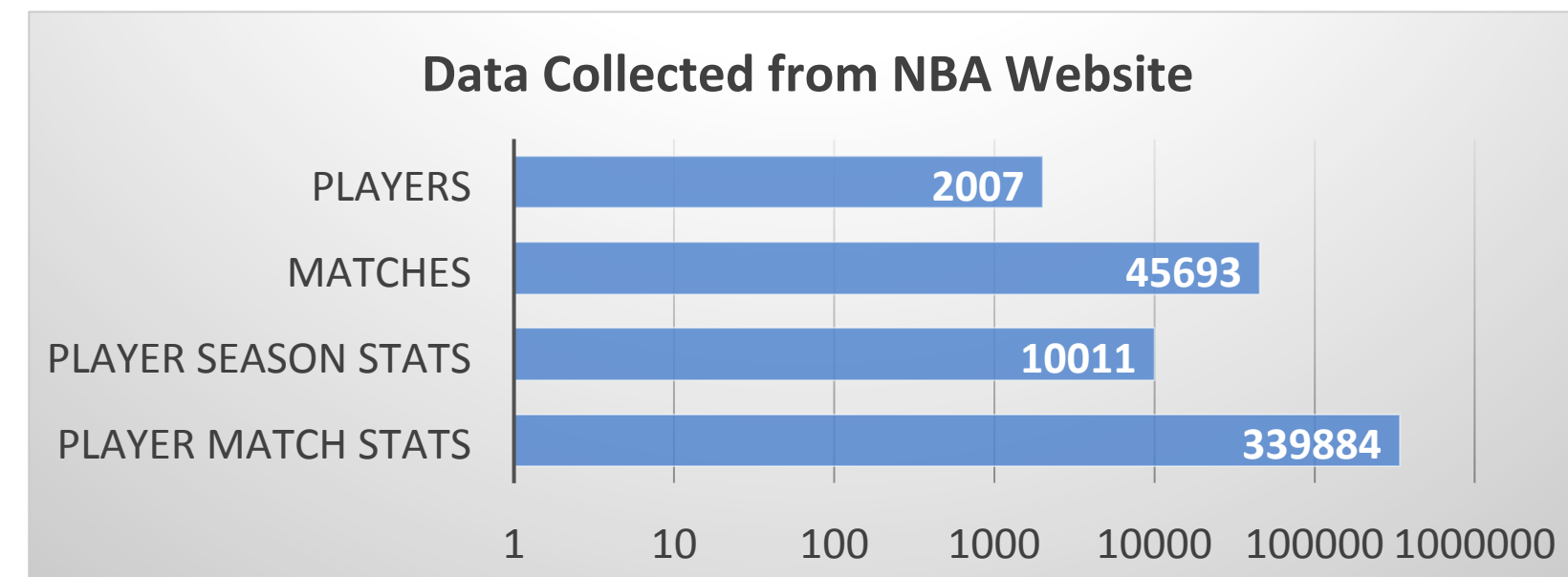
^[3]

$$E = \frac{1}{2} \sum (y - y')^2$$

Where y is the actual output and y' is our feed-forward prediction. The goal is to adjust the weights of the edges to reduce this error. This is done by "backpropagating" the error through the network. Then, the weights are adjusted, and the network's performance will increase.

3 Data Collection

With the model for our system chosen, we could begin collecting our data. The inputs to our system are NBA statistics. We built a scraper to obtain data from the NBA website (chosen for reliability). We scraped partial data from games back to 1970, and complete back to 2004:



We used Python to power our scraper



We used BeautifulSoup^[4] to retrieve the NBA website

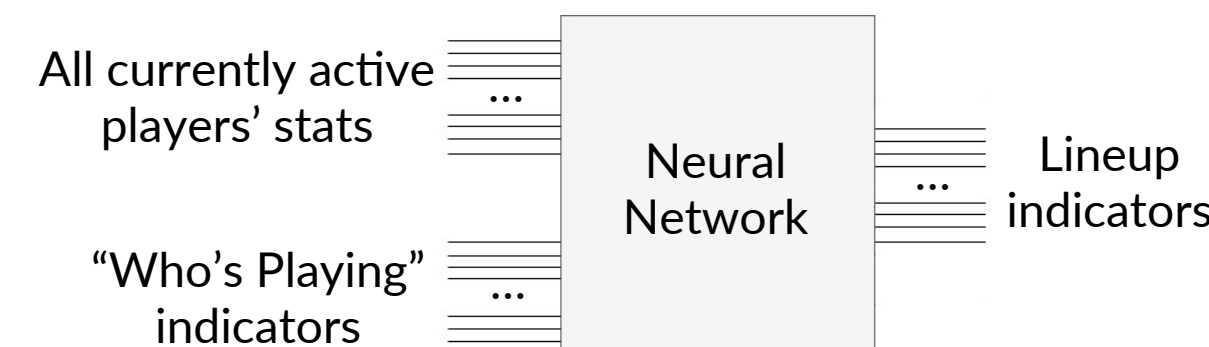


We used Selenium^[5] to control the NBA website



4 System Architecture

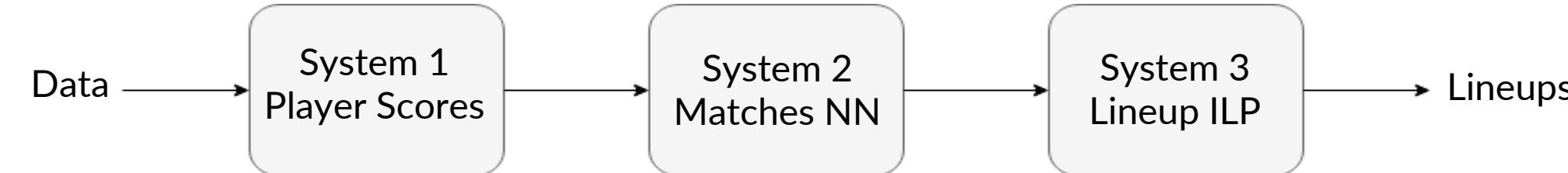
Our first architecture was a NN with multiple inputs for all active NBA players: their stats, and an "are they playing tonight" indicator. The NN had one output per player: a value indicating if the player should be in the lineup.



This preliminary model had limitations:

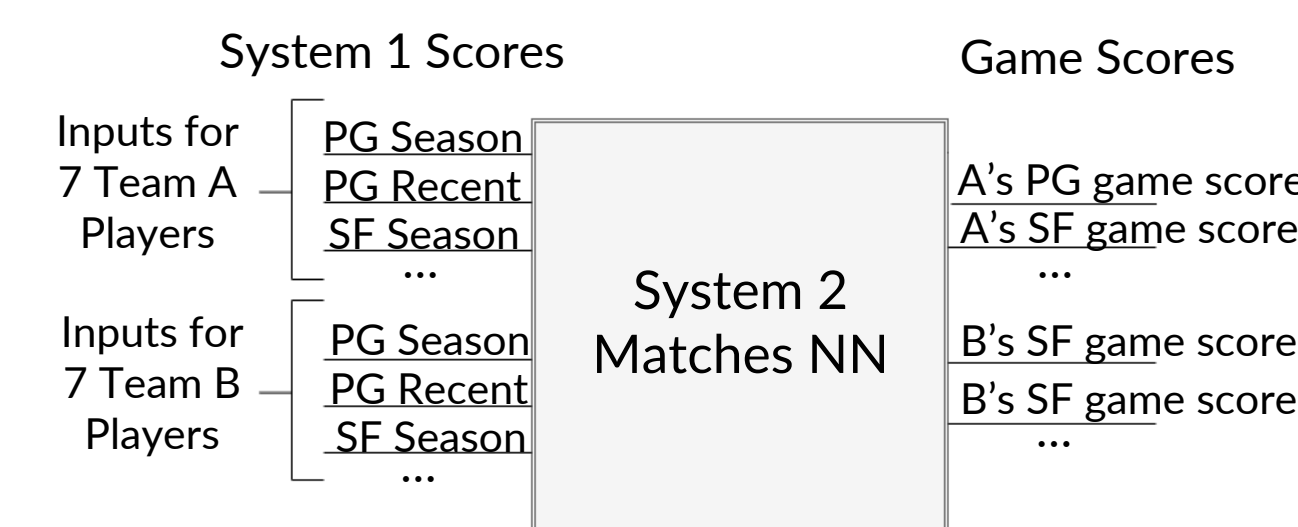
1. It would be tough for the network learn what the inputs mean
2. Active players by nature mean we can't use data from old games
3. The network will find correlations between unrelated players
4. The salary constraint is not handled

Instead, we spent time on architecture, and came up with a much better second solution, composed of 3 Systems:



System 1 is unrelated to the neural network and computes the players' "scores" with a simple function. The scores roughly represent how good the players are at scoring fantasy points.

System 2, the neural network, now computes the "game scores" of players: a prediction of how many fantasy points they will get in a specific game. The NN is trained and tested on single matches, with far fewer inputs (42). The inputs are the players' season scores and their recent scores, both computed by System 1. These inputs are sorted by position, as can be seen below:



This network is not tied to specific players. We lose this information, but in turn we get to train this on all NBA matches, not just the ones with active players. As well, now that we test matches individually, they are independent, and there is no correlation. We have solved the first three limitations!

System 3 solves the fourth limitation: the salary constraint problem.

Given a bunch of players with scores, positions, and salaries, under positional constraints and a salary budget, pick a lineup to maximize the total score.

We solved this problem using a Python library called PuLP – a linear programming optimization solver.

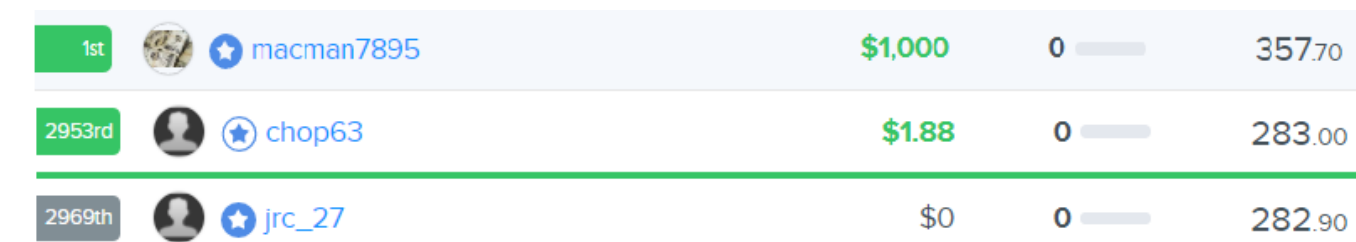
We were then able to get multiple lineups by adding random Gaussian noise to the outputs of the NN before feeding them to the LP problem.

5 Results

Defining a success metric for our lineups was nontrivial. The ideal metric is the profitability of the system, but this is incomputable without competition history. Thus, we selected a heuristic metric: the lineup's score relative to that of the best possible lineup that day.

$$Success(lineup) = \frac{lineup's\ fantasy\ score}{best\ lineup's\ score}$$

By looking at competitions, we determined that this heuristic implies profitability at around 80% and greater. In other words, if we can score 80% of the maximum possible score, we can win. The figure below shows this relationship, where the last winning score is roughly 79% of the maximum.



In order to improve this metric (started at ~60%), we performed cross-validation – a method of selecting the best hyperparameters of our NN. These hyperparameters include the number of hidden nodes, hidden layers, and the learning rate. This is a computationally heavy process, and so we rented a GPU from Paperspace.^[6]

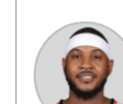
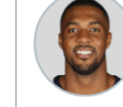
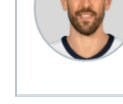
With cross-validation, we were able to hit a success metric of 82% - just above the minimum profitability line. Thus, we started joining competitions, and immediately ran into some initial issues:

- Non-updated database
- Players who didn't end up playing that night
- Accidentally ignoring players in some games
- Using the wrong data for a player

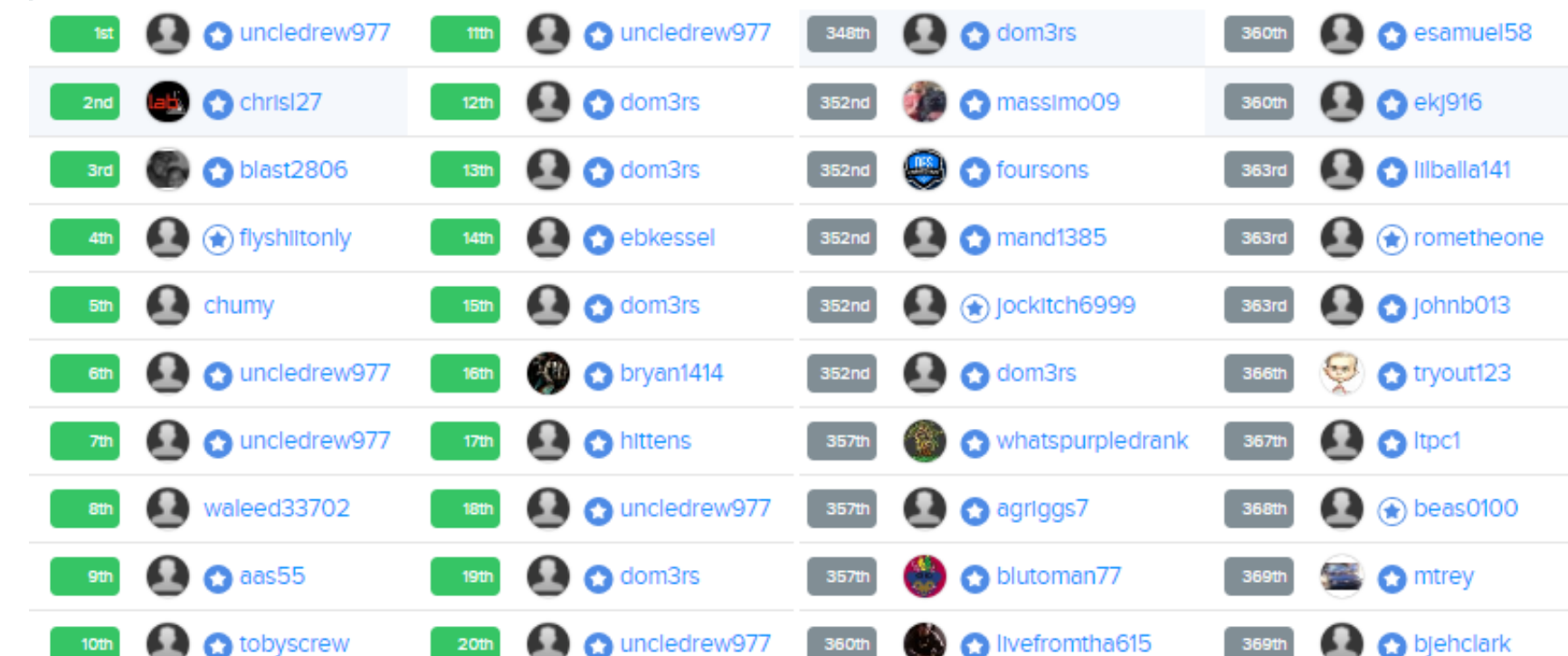
We solved these issues and continued testing. To our surprise, the system was somewhat successful. Below are screenshots of the competition summaries for all of the competitions we played in where all of our players played.

Date	Contest	Score	Opponent	Entry Fee	Winning
04/03	NBA Beat the Score: 260 (Beginners Only, \$50 Guaranteed)	303.8 (1 of 52)	Tournament	\$1	\$1.22
04/03	NBA Beat the Score: 275 (\$500 Guaranteed)	294.4 (1 of 595)	Tournament	\$1	\$1.04
04/01	\$2K Sun NBA Fadeaway (\$0.25 to Enter)	263.3 (3916 of 9580)	Tournament	\$0.25	\$0
04/01	\$2K Sun NBA Fadeaway (\$0.25 to Enter)	269.6 (3162 of 9580)	Tournament	\$0.25	\$0
03/30	\$5K Fri NBA Rabies Shot (\$5K Guaranteed)	324.4 (151 of 1340)	Tournament	\$4.44	\$10
03/28	\$15K Wed NBA Block (Single Entry)	319.8 (830 of 8929)	Tournament	\$2	\$4
03/28	\$15K Wed NBA Drizzle (Single Entry)	287.5 (5459 of 17857)	Tournament	\$1	\$0

Sometimes our predictions are neat: We make good picks that others miss! In our best lineup, our players were picked on average 7.5% by others. The nearest competitor in score averaged 27%. Three of these neat picks:

 DEN 126 @ OKC 125 REAL \$5,500 SALARY 2.9% OWNED	39.8
 MEM 97 @ UTA 107 REAL \$5,400 SALARY 2.5% OWNED	31.2
 MEM 97 @ UTA 107 REAL \$5,400 SALARY 2.5% OWNED	34.6

As well, one can see that, even in the competitions we lose, we do not perform that poorly (always top 50%). But who are we up against? Pros!



A white star in blue means they have been in over 1000 competitions and have won at least \$1000 across four contests.

A blue star in white means they have been in at least 500 competitions or have won at least \$1000 across four contests.

Looking through by hand, we found that ~85% of people have one type of these star in these competitions! This is true of the winners and the losers.

Our neural network was able to perform quite well against these experienced players, turning a profit in lineups that had all players active.

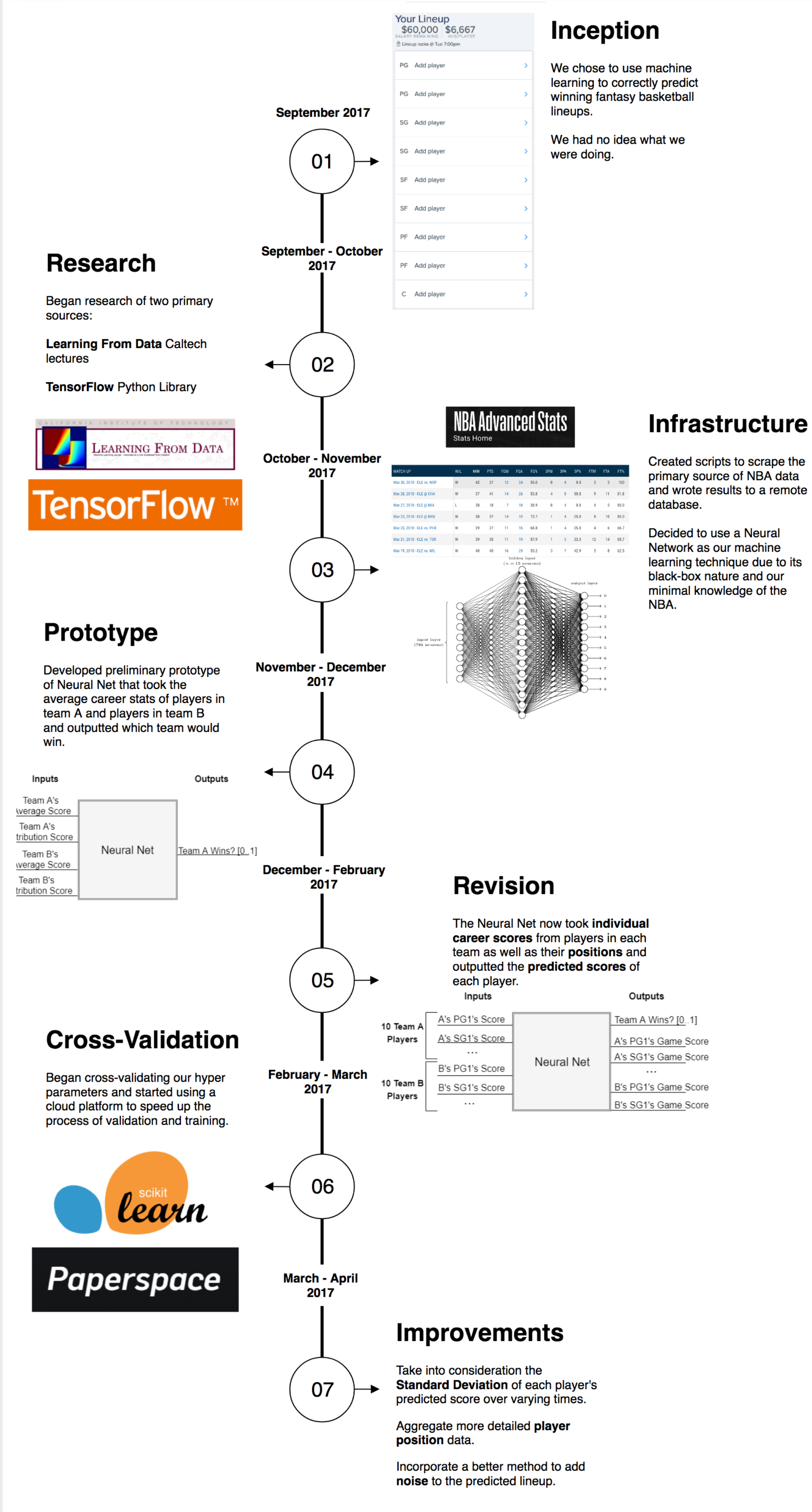
However, the system is still far from being reliably profitable, as its outputs are high in variance. There are still improvements to be made!

6 Future Plans

We did not think our network was going to be nearly as successful as it was, but it is still far from perfect, and not reliably profitable. We plan to continue to pursue and improve this project. Some ways to improve include:

- Add additional valuable features
- Define a better score function
- Perform more cross-validation (e.g. on data amount used)
- Performing Boosting and/or Bagging
- Factor standard deviation into LP
- Find more precise player positions

7 Timeline



Referenced within:
[1] <https://www.fanduel.com/contests>
[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] <https://www.crummy.com/software/BeautifulSoup/>
[5] <https://www.seleniumhq.org/>
[6] <https://www.paperspace.com/>