Beating the House with GANs

Sean Moriarity and Eric Sturzinger
Department of Electrical Engineering and Computer Science
United States Military Academy
West Point, New York
sean.moriarity@westpoint.edu, eric.sturzinger@westpoint.edu

***Abstract*—As the legalization of sports betting sweeps across the United States, bettors and sports books alike continue to search for an edge. With the recent increase in emphasis on data analytics in sports, the amount of publicly available sports data is massive and continues to grow. The massive amounts of available data indicate sports betting is a perfect application for machine learning (ML). However, due to the infancy of legal sports betting in the United States, the application of sophisticated ML models and their unique capabilities in sports betting remains largely unexplored. Sports books maintain an artificial edge when they set lines, leaving bettors at an inherent disadvantage. In this paper, we propose a new strategy to eliminate this disadvantage based on Generative Adversarial Networks (GANs) specifically to predict sporting event upsets. We then compare the efficacy of this approach to more traditional machine learning approaches and baseline sports betting approaches. Results show that this approach outperforms the baseline and machine learning approaches.**

***Index Terms*—generative adversarial network, sports betting, anomaly detection**

I. Introduction

The legalization of sports betting across the United States, coupled with the rise in the use of analytics in sports has created a wealth of publicly available data waiting to be exploited. Sports betting is an adversarial game between bettors and bookmakers, also called the *house*. A bookmaker proposes lines on games that represent the amount of money a bettor would win on those games proportional to their overall bet.

The goal of the bookmaker is to maximize the profitability of their book by proposing lines that minimize their potential payout and maximizes their *margin*. The margin is the advantage bookmakers maintain over bettors by proposing lines where the expected value of their payout is less than the amount of money they accept in the form of bets. Bookmakers remain profitable by adjusting their lines proportional to the money bet on different outcomes.

The margin is such that it is difficult for bettors to obtain profitable outcomes even when they can predict the outcome of sporting events with greater than over 50% probability. Because of this, bettors are forced to come up with strategies that identify the most profitable games on which to bet to maximize their returns and minimize their risk. The process of identifying profitable games is called finding an *edge*.

An edge is the discrepancy between the true probability of an outcome and the bookmaker's proposed probability of the same outcome. An edge can be used to place flat bets or combined with a more sophisticated betting strategy such as *Kelly criterion* to yield considerable profits. A flat bet is simply a fixed numerical bet on the outcome of a game. Traditionally, bettors bet in units where 1 unit is equivalent to 1% of their total bankroll. With this approach, bettors attempt to calculate their edge on certain lines, and then bet according to a minimum threshold.

A more sophisticated approach uses the Kelly criterion. The Kelly criterion is used in investment theory to determine the optimal percentage to allocate to a certain stock in one's portfolio based on risk and expected return. In the context of sports betting, the Kelly criterion can be used to determine an optimal bet based on the predicted probability of a team winning versus the odds proposed by a bookmaker.

Both betting strategies depend on the ability to accurately assess the true probability of a given team winning a game. Most machine learning approaches attempt to accurately predict the true probability of a team winning based on features such as team or player statistics. However, the bookmaker's margin is such that these approaches require a high accuracy threshold to be profitable. Additionally, the use of accuracy as an assessment of models in sports betting is not a valid metric of success because accuracy does not always correlate to profitability.

In this paper, we propose a new strategy for betting on moneylines using Generative Adversarial Networks (GANs) to correctly predict upsets. Upsets are especially appealing because they have a far greater payout than betting on the favorite. Strictly betting on upsets significantly lowers the threshold of accuracy required to yield profits.

The rest of this paper will be organized as follows. First, we will provide a brief overview of GANs and their more traditional applications before exploring the current state of the art of machine learning in sports betting. Next, we will describe the architecture used to predict upsets. Finally, we will assess the efficacy of the architecture compared to other novel approaches.

II. Background

GANs are a family of generative models that use adversarial training to generate samples that closely resemble training data [1]. One key difference of GANs as opposed to other generative models is that rather than sampling from a latent space, they train a generator to produce samples from transformations of random noise input. GANs have proven a viable option for producing realistic data. The most popular application of

GANs is in the generation of realistic looking images with Convolutional Neural Networks (CNN) [2]. However, GANs have a much broader range of applications.

Because GANs are able to accurately generate samples from training distributions, they can be used for a variety of other applications. The goal of generative modeling is to accurately recreate a probability distribution. This requires a model to learn the features and patterns of a distribution such that it would be difficult for a reliable generative model to generate samples with anomalous features. This is precisely the approach taken by [3]. Anomalies in a distribution are samples of data that deviate significantly from a predefined threshold of normal. The authors in [3] used GANs to learn a distribution of normal images and then calculated an anomaly score for new images based off how closely they resembled the learned distribution. Along with the more traditional ML domains such as vision, anomaly detection can also be used in sports betting as a means of identifying an edge over bookmakers to increase profitability.

The applications of ML in sports betting is an understudied field. Most research emphasizes increasing accuracy as a reliable metric for tracking the performance of sports betting models; however, the bookmaker's margin is such that accurate models are not necessarily profitable. For example, Maymin [4] used neural networks to determine the profitability of betting on NBA games over the course of a season using different datasets. The results of the research show it was difficult to achieve profitability using standard box score statistics even if the model was accurate. The goal instead is to identify an edge rather than accurately predict every game. Hubáček, Šourek, and Železný [5] proposed a new approach using convolutional neural networks to extract individual player contributions to the outcomes of games. Additionally, they proposed techniques to decorrelate a bettor's model from the bookmaker's model. Finally, they proposed the use of various betting strategies such as portfolio-optimization to achieve profitability. The shift in emphasis on accuracy as a metric of performance to profitability ultimately yielded a more profitable ML model.

III. Upset Detection with GANs

GANs [1] are a framework for creating generative models using adversarial training. Two neural networks, a generator, G, and a discriminator, D, take part in a two-player minimax game. G is a mapping of some input z to a probability distribution p_g such that

$$G(z;\theta_g) = p_g \tag{1}$$

.

and D is a mapping of some input x such that

$$D(x;\theta_d) = P(x \epsilon p_{data}) \tag{2}$$

Theoretically, when the network converges, G can perfectly replicate p_{data} such that

$$G(z;\theta_g) = p_g = p_{data}. \tag{3}$$

One application of GANs is anomaly detection. Anomaly detection is the identification of patterns in data that does not meet a defined threshold of normal in the original distribution [6]. AnoGAN is an architecture proposed in [3] that utilizes GANs to compute anomaly scores for medical imaging to identify disease markers. The proposed model learns a representation of normal data i.e. it only trains on non-anomalous data points. Both the generator, G, and the discriminator, D, are trained to an equilibrium such that G improves at generating realistic images and D is better at identifying generated images. Once G and D are trained, a new image x can be mapped to an anomaly score according to the function:

$$A(x) = (1-\lambda)\cdot R(x) + \lambda \cdot D(x) \tag{4}$$

where $R(x)$ is defined as the *residual score* such that:

$$R(x) = \sum_{i=1}^{n} \mid x - G(z_i) \mid \tag{5}$$

and $D(x)$ is defined as the *discrimination score* such that:

$$D(x) = \sum_{i=1}^{n} \mid f(x) - f(G(z_i)) \mid \tag{6}$$

where z_i is a query that produces the most similar image to x. The function f is the output of an intermediate layer in the discriminator that represents the features of the image. λ is the weight assigned to each function in the anomaly calculation.

The anomaly score indicates how anomalous an image is, with high scores representing more anomalous images and low scores representing less anomalous images. Both the residual score and the discrimination score are calculated over hundreds of iterations. The residual score represents how difficult it is for G to reproduce x from the distribution it has learned. The discrimination score determines the difference between the features of x and the features of $G(z_i)$. Because the original network is trained only on non-anomalous data points, an anomalous data point would be difficult for G to accurately reproduce because it does not belong to the non-anomalous distribution learned by G.

In the context of sports betting, an upset occurs when an underdog beats a favorite. Considering upsets as anomalies, an architecture similar to AnoGAN can be applied to predict the likelihood that a game will end in an upset.

Given a matchup between two teams, Team A and Team B, a bookmaker presents two lines such that:

$$L_A = \frac{1}{P(A_{win})}$$
$$L_B = \frac{1}{P(B_{win})}$$

This type of bet is called a *moneyline*, where a bettor attempts to correctly choose the winner of a game. If the odds were fair, the probabilities of either Team A or Team

B winning would sum to 1; however, when creating the lines, bookmakers maintain a margin [5] such that:

$$M = \mid \frac{1}{L_A} + \frac{1}{L_B} - 1 \mid \tag{7}$$

The bookmaker's margin is such that they maintain an advantage over bettors, even when bettors are able to correctly predict the outcome of sporting events well over 50% of the time. This makes it extremely difficult to systematically develop a profitable betting strategy.

Edge betting is a way to counter the bookmaker's margin. The edge is defined as the difference between proposed odds and the true probability of a team winning. So, if the proposed odds on Team A winning were 1.95, but a bettor predicts the true probability of Team A winning is 0.60, then the bettor's predicted edge would be:

$$E = P - \frac{1}{L} = 0.6 - \frac{1}{1.95} \approx 0.0872 \tag{8}$$

If a bettor is able to identify discrepancies in the proposed odds versus the true outcome of a game, they have an advantage over the bookmaker and can capitalize on the advantage to produce a profitable outcome. Considering that upsets occur when a team with odds corresponding to a less than 50% chance of winning wins, upsets are games where the bookmaker was incorrect in predicting the outcome of the game. This means the odds for upsets are favorable to bettors. In order to identify potential upsets, I propose a model architecture based on AnoGAN that predicts the likelihood of an upset based on an upset score.

First, rather than supplying the generator, G with random noise and generating an image, a vector x of normalized team statistics is passed to G to generate odds. In this respect, G acts as the bookmaker, learning to reproduce the distribution of odds assigned by real bookmakers. The proposed odds and team statistics are separately passed to D which determines whether the assigned odds and accompanying statistics were produced by a real bookmaker, or were produced by the generator. Theoretically, G should be able to perfectly replicate the distribution of odds assigned to games by real bookmakers. Fig. 1 shows a basic overview of the network design.

Both G and D are trained only on games where the bookmaker correctly predicted the outcome. Following from [3], this forces G and D to learn only the distribution of non-anomalous games. A game is considered to be non-anomalous when the outcome of the game follows from the bookmaker odds.

To determine if a game is an upset, the upset score is calculated in the same manner as in [3] using Eq. 4 and the trained G and D. G is repeatedly queried with a vector x of normalized team statistics to produce predicted odds on a provided matchup. The upset score indicates the amount of difficulty the network has reproducing the true odds for the provided matchup. A high upset score indicates that the bookmaker odds from a matchup do not follow from the distribution of correctly predicted games. Thus, a high upset score means the bookmaker is more likely to be incorrect about the outcome of a matchup.

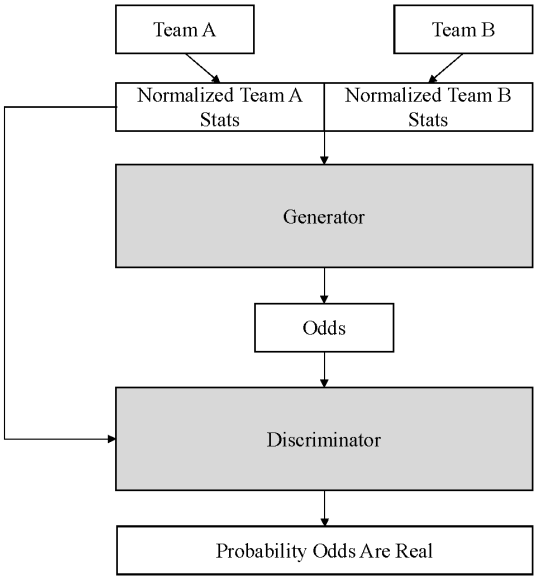

Fig. 1. Upset Detection Network Overview

IV. Experiments

In order to verify the efficacy of anomaly detection using GANs as a profitable approach to sports betting, we tested the proposed model using NBA moneylines. We collected historical outcomes and odds from every regular season NBA matchup since 2007. In total, the dataset consisted of 11754 games or 23508 different moneylines. We then collected a total of 87 different team statistics for each team from the official NBA website. We arranged the matchups in a CSV such that each line was formatted in the following order:

- Proposed Winner Statistics
- Proposed Loser Statistics
- Proposed Bookmaker Odds
- Actual Outcome

We normalized team statistics and converted the proposed bookmaker odds from the American odds system to the Decimal odds system according to the following equation:

$$D(Odds) = \begin{cases} \frac{100}{|Odds|} + 1, & Odds < 0 \\ \\ \frac{|Odds|}{100} + 1, & Odds \geq 0 \end{cases} \tag{9}$$

The actual outcome of a moneyline was a binary 1 or 0 indicating whether or not the proposed outcome actually occurred.

We split the dataset into training, validation, and test sets. We then created 15 random samples of 5120 matchups from the test set which represents 15 full seasons of NBA matchups.

Using the dataset, we established three naïve baselines: flat bets on favorites only, flat bets on underdogs only, and flat

bets on random underdogs. We determined whether a team was a favorite or underdog according to their odds, with odds greater than 2.0 indicating a team was an underdog and odds less than 2.0 indicating a team was a favorite. We used a flat bet of 1 unit to indicate 1% of a bettor's bankroll spent on each matchup.

We then ran the experiment with a standard Artificial Neural Network (ANN). The ANN was a basic fully connected feed-forward network that produces a probability that a given moneyline will win. The ANN consisted of 5 hidden layers of 128, 64, 32, 32, and 16 units. Layers 1, 3, and 4 used sigmoid activation and Layers 2 and 5 used ReLU activation. The output layer used sigmoid activation to produce a probability between 0 and 1 of a matchup resulting in a win. We used the Adam optimizer with Binary Cross-Entropy loss. We tested this approach on both flat bets of 1 unit on the predicted winning team and using Kelly's criterion to generate varying unit bets.

Finally, we tested using my proposed architecture.

The generator accepted a vector x of team statistics. It consisted of 2 dense layers of 128 and 64 units. Both layers used leaky ReLU activation [7] with learning rate $\alpha = 0.2$. The first hidden layer also used batch normalization [8] with $momentum = 0.8$. The final layer used sigmoid activation to output the odds of a matchup as a probability of the proposed winner winning the game.

The discriminator accepted two inputs: a vector x of team statistics and the odds proposed by the generator. It consisted of 2 dense layers of 128 and 64 units using leaky ReLU activation with $\alpha = 0.2$. Both layers also used a dropout of 0.2. The output of the second hidden layer was concatenated with the odds proposed from the generator and passed to the output layer which used sigmoid activation to generate a probability that the proposed odds came from a real bookmaker. Fig. 2 depicts this design.

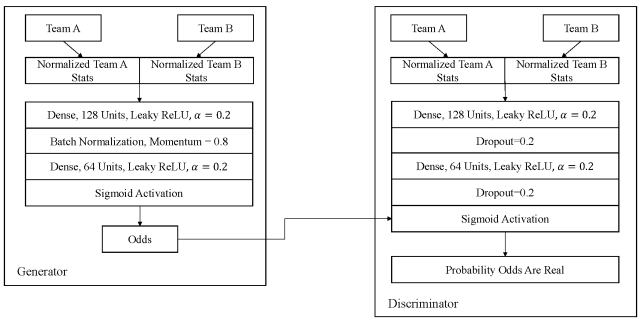

Fig. 2. Upset Detection Network Design

The proposed architecture was trained only on games that ended with the favorite winning. The discriminator was shown both valid odds and fake odds with its loss corresponding to its ability to recognize real and fake odds. The loss of the discriminator was propagated back to the generator to improve the generator's ability to generate odds. The joint model was trained using the RMSProp optimizer on batches of 64 matchups over 10000 epochs.

We computed upset scores using the output of the residual score of the generator and the discrimination score of the output of the first layer of the discriminator over the course of 500 iterations. We computed the upset scores of every underdog line in the random sample. We then used a flat bet strategy with a flat bet of 1 unit to only bet on games that met a certain upset score threshold. Because the profitability of the model depends heavily on the upset score threshold, we tested the 15 different random samples of seasons with 10 different thresholds ranging from 4.0 to 6.0.

Table I shows the 95% confidence interval of accuracy of each model over the course of 15 random seasons.

TABLE I
ACCURACY OVER 15 SEASONS

Strategy	Accuracy
Favorites Only	$(67.81 \pm 0.54)\%$
Underdogs Only	$(30.45 \pm 0.52)\%$
Random Upsets	$(30.28 \pm 2.20)\%$
ANN	$(63.72 \pm 0.51)\%$

Table II shows the 95% confidence interval of profits of each strategy over the course of 15 different seasons.

TABLE II
PROFITS OVER 15 SEASONS

Strategy	Profit (units)
Favorites Only	-116.06 ± 22.60
Underdogs Only	-117.03 ± 49.76
Random Upsets	-94.80 ± 48.08
ANN (Flat Bets)	-107.81 ± 29.20
ANN (Kelly Criterion)	-66.30 ± 15.43
Upset GAN*	1.98 ± 11.15

The profits shown in Table II for the anomaly detector reflect the outcomes of the most profitable upset score threshold over the course of 15 seasons ($t = 5.2$). Table III shows the 95% confidence interval of 10 different thresholds over the course of 15 seasons.

TABLE III
PROFITS OVER DIFFERENT THRESHOLDS

Threshold	Profit (units)
4.0	-4.17 ± 32.52
4.2	-0.73 ± 28.40
4.4	1.01 ± 27.32
4.6	0.51 ± 22.78
4.8	-0.25 ± 19.93
5.0	1.35 ± 14.38
5.2	1.98 ± 11.15
5.4	0.36 ± 8.86
5.6	-0.24 ± 7.41
5.8	-0.64 ± 4.49
6.0	-0.41 ± 4.19

The data shows that even the worst performing threshold significantly outperforms the best of the other models tested.

Upset GAN also significantly outperforms the baseline that randomly selects upsets to bet on, indicating it is able to identify patterns in the data that indicate a game is profitable.

The data also shows how upset score threshold impacts the profitability of the betting strategy. A betting strategy that uses lower thresholds has more potential for profit, but it also has more potential for losses. Fig. 3 shows how profitability varies by threshold by season for thresholds of 4.0, 5.0, and 6.0.

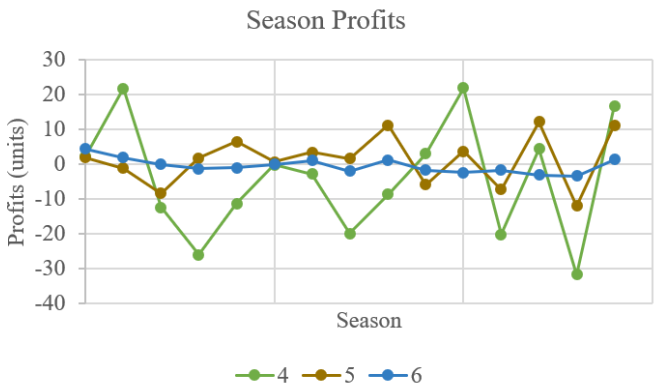

Fig. 3. Graph of Profits by Season

The chart shows how lower thresholds have the most potential for significant profits but also for significant losses. As upset score threshold increases, the profitability curve flattens and approaches zero. This is because higher thresholds are more selective on the games they bet on and thus bet on fewer games. This minimizes the risk with higher thresholds but also minimizes potential return. The data shows that the best performing threshold effectively balances potential returns and profitability.

Finally, the data verifies that accuracy is not a good indication of profitability when betting on NBA moneylines. The data shows the favorite wins roughly 67% of the time in the NBA. The lines are set such that betting only on the favorite will yield significant losses over the course of a season.

V. Future Work and Conclusions

This paper verifies the efficacy of anomaly detection using GANs as a profitable approach to sports betting. While the model is not significantly profitable, there is room for improvement. For example, the work done by Maymin [4] showed that box score statistics are not sufficient to produce profitable models. The model proposed in this paper could be significantly improved with the introduction of more detailed data.

Additionally, the model proposed in this paper does not account for the performance of individual players or the performance of a team across windows of time. The statistics used to predict games are averages of a team's performance throughout the entire season up to the game being predicted. This means the model is unable to account for "hot streaks" or the contributions of individual players when making predictions. The model could be improved using Recurrent Neural Networks (RNNs) to capture the performance of teams through time to better predict performance. Alternatively, a CNN could be used to identify local patterns in the matchups between individual players on teams and to account for mid-season trades and injuries.

Finally, this paper uses a flat betting strategy based on a threshold of upset scores. Further exploration of betting strategies according to upset scores are required to maximize the potential profitability of this method. For example, translating upset scores into probabilities to then be used in Kelly's criterion or in portfolio optimization could potentially lead to more profitable returns. Alternatively, reinforcement learning could be applied to teach an agent how to bet based off of signals from this model architecture.

References

[1] I. Goodfellow et al., "Generative Adversarial Nets," in Advances in Neural Information Processing Systems 27, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[2] A. Radford, L. Metz, and S. Chintala, "UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS," p. 16, 2016.

[3] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery," arXiv:1703.05921 [cs], Mar. 2017, Accessed: Apr. 09, 2020. [Online]. Available: http://arxiv.org/abs/1703.05921.

[4] P. Z. Maymin, "Wage against the machine: A generalized deep-learning market test of dataset value," International Journal of Forecasting, vol. 35, no. 2, pp. 776–782, Apr. 2019, doi: 10.1016/j.ijforecast.2017.09.008.

[5] O. Hubáček, G. Šourek, and F. Železný, "Exploiting sports-betting market using machine learning," International Journal of Forecasting, vol. 35, no. 2, pp. 783–796, Apr. 2019, doi: 10.1016/j.ijforecast.2019.01.001.

[6] F. Di Mattia, P. Galeone, M. De Simoni, and E. Ghelfi, "A Survey on GANs for Anomaly Detection," arXiv:1906.11632 [cs, stat], Jun. 2019, Accessed: Apr. 09, 2020. [Online]. Available: http://arxiv.org/abs/1906.11632.

[7] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," p. 6.

[8] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," arXiv:1502.03167 [cs], Mar. 2015, Accessed: Apr. 30, 2020. [Online]. Available: http://arxiv.org/abs/1502.03167.