NBA Play by Play Prediction

Frederick Zhang[2], **Brawner Quan**[2]
[2]Department of Computer Science
Tufts University
Medford, MA 02155
{frederick.zhang, brawner.quan}@tufts.edu

Abstract

Many Machine Learning techniques have been used to attempt to predict the outcome of NBA games. Before the Deep Learning Revolution, previous research has investigated classic machine learning methods such as regression models, SVM, and various simple Feed Forward Neural Networks in search of improving the state of the art in predicting the outcome of NBA games. Taking inspiration from Football Match Prediction using Deep Learning [6], we explore a LSTM based approach to predict the outcome of NBA games and achieve an 79.59% accuracy in predicting the outcome of NBA games.

1 Introduction

Professional sports are a global phenomenon. Just the NBA alone is a multi-billion dollar industry, with revenue exceeding 7.9 billion dollars in the 2019-2020 season alone [4]. Furthermore, the sports betting industry in the United States is massive. For example, in New Jersey alone, 6 billion dollars were wagered on sports in 2020 [2]. Crucial to both the NBA and sports betting is accurate match outcome predictions, and it remains an unsolved problem, as the highest accuracy the state of the art has achieved in Match Prediction is 74% [5].

The current research on predicting NBA outcomes (the winner of the game) using machine learning relies on techniques that pre-date the deep learning revolution. Linear Regression, Logistic Regression, SVM, primitive Feed Forward Neural Networks and decision trees have been used independently and combined to predict match outcomes [1] [3]. In addition, Bayesian based models have also been explored [3].

Since the Deep Learning revolution, architectures and concepts such as Autoencoders, RNNs, CNNs, and Attention mechanisms have improved the state of the art in most areas of active machine learning research such as Natural Language Processing and Computer Vision. However, in our search for references in this field, we found research in applying deep learning to predict the likelihood of a NBA player's success [8] and basketball shot trajectories [9], but discovered no papers applying deep learning to predicting the outcome of NBA games.

Thus, because of deep learning based innovations in sequential modeling, which is a task that classical machine learning methods struggle with, we chose to explore applying a sequential deep learning architecture, namely a LSTM, to predicting the outcome of NBA games. Based of findings from **Raj (2017)**, we also explore incorporating CNNs into our sequential architecture [7].

In this paper, rather than predict statically based off of all knowledge known prior to a game, we leverage the sequential nature of LSTMs to process the first 150 plays in a NBA game and combine that with CNNs to make a prediction on the outcome of a given NBA game.

2 Related Works

There have not been published improvements to the state of the art for predicting NBA game outcomes since 2013. Furthermore, none of the currently published papers use deep learning. However, this may be because models that incorporate deep learning beat the state of the art could also serve as profitable sports betting models, which, to maintain their edge over the rest of the market, have no incentive to be published publicly. **Štrumbelj and Vračar (2012)** achieved 70.69% accuracy by using Markov models to represent play by play data, suggesting that a more detailed and accurate simulation of a basketball game may lead to better predictions. [11]. **Loeffelholz, Bednar, and Bauer (2009)** achieved 74.33% accuracy by restricting player stats feature set to 4 features and feeding that into a simple Feed Forward Neural Network [5]. **Beckler, Wang, and Papamichael (2013)** achieved up to a 73% accuracy by creating a SQL database that allowed live updating of stats on a per game basis when training and testing, allowing current stats of rookie players in the league to be passed into the model [1].

3 Methodology

Our approach builds on top of [10] to include player information into each play-by-play feature vector. Instead of denoting part of the vector for each team we only give the play-by-play to the team and players relevant to the specific play-by-play. For each play-by-play we provide the team embedding for the team that committed the actions as well as the player embedding for up to 3 players that are involved in the action including main actor, assister of action, and opposer of the action. When not all 3 players are need we use a 0-vector for players not needed to represent the action (i.e. a shooting action when no one is present to assist). For the rest of the vector we include seconds left, quarter number, home team score advantage and 1-hot encoding of action type. To normalize the data we use min-max scaling so all entries of the input vector is between 0 and 1.

Our embedding was trained end to end using a player embedding network that is shared and uses the same network to produce embeddings for all players present in the play-by-play and a team vector. We use an encoder to generate our embeddings. The encoder takes in player and team summary statistics over the prior years as a high dimension vector and represents it in a lower latent space. Player and Team embeddings were concatenated together with the rest of the original input vector and fed into the rest of our model for downstream prediction. All play-by-plays in a game were embedded to form sequential data across play-by-plays which we used in an LSTM model. The hidden output of the last time point of LSTM model was fed into Dense output layers to produce the final prediction.

To further enhance our model we wanted to explore using both CNN and LSTM layers together which found success for "biomedical text" domain in Raj 2017. We believe that capturing both short and long term context within play-by-play information [7]. We used 2D convolution and pooling to look at feature information across adjacent play-by-plays. Our convolution treated the each play-by-play as 1 dimension of convolution and the features in each play-by-play as another feature. We are only interested in neighboring play-by-plays and not neighboring features in our input convolution and pooling. Our model only looked a neighborhoods that pool the same features of neighboring play-by-plays. So each filter was a 1D filter which did not pool adjacent features in the same play-by-play. Additionally we also used pooling to reduce the number of play-by-plays to help improve downstream training for the predictive task.

One issue we were concerned with is feeding the results of LSTM layers into CNN since it may lose out on short local changes of play-by-play. Alternatively we were also concerned with using CNN layers first that fed into LSTM since pooling from CNN layers will reduce play-by-play size dimension which could lose long term dependencies. So we experimented with using 2 parallel layers of CNN and LSTM that take input directly from the emebedded input features and concatenate their output into a downstream predictive model. To concatenate CNN and LSTM output we had to structure CNN layer output to have a 1 channel output and LSTM to maintain the same number of output hidden units as the embedding feature size.

4 Experimental Setup/Results

For our model we used a 2 unit output layer with softmax and BinaryCrossEntropy loss. Our model was trained using default Adam optimizer (on TensorFlow) for 300 epoch with 0.9 train validation data set split. We selected for training the games that were in the 90 percent chronologically earliest games played in the 2015-2018 period and the validation using the last 10 percent of games. We used the first 150 plays in each game to predict the final outcome of the game deciding home or away team would win. Test data set was composed of play-by-play games in the 2018-2019 season. We collected our data from Basketball-Reference to scrape players and team statistics and play-by-play games was taken from a Kaggle Data set.

We used our training-validation split to evaluate model performance on accuracy to decide the best model architecture for our final model. We decided on 6 major architecture types for testing as follows:

LSTM Only Dose not use parallel LSTM and CNN layers. This architecture is composed of an end-to-end trained encoder that feeds sequentially down to a LSTM predictive layer followed by Dense output layers.

Conv + LSTM (With Pooling Size Reduction) Begins with encoder that feeds encoded output to a parallel LSTM and CNN layers. The output of the parallel LSTM and CNN layers are concatenated and fed sequentially down LSTM layers and Dense output layers. CNN layers use pooling with (2,1) stride so that the data is reduced along the play-by-play dimension by half.

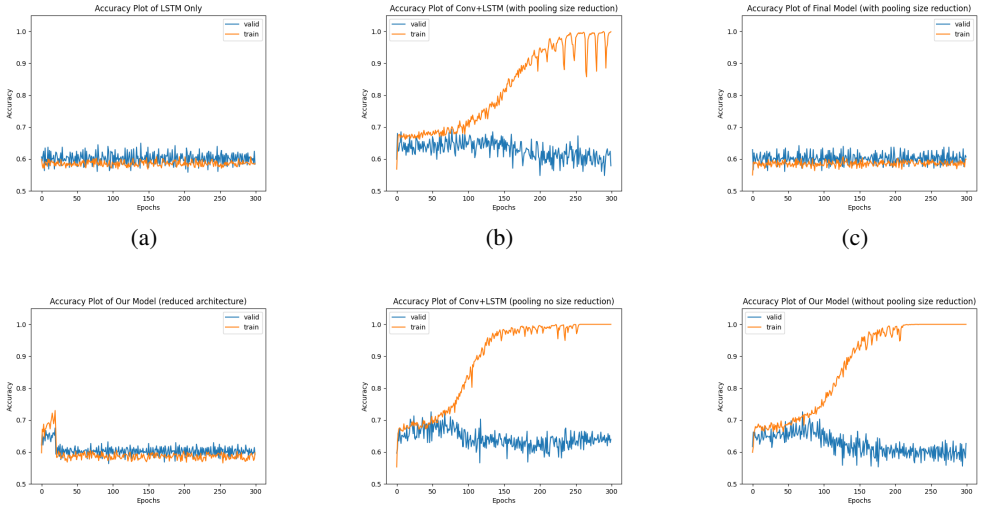
Final Model (With Pooling Size Reduction) Uses the Conv + LSTM (With Pooling Size Reduction) architecture except we interleave the sequential step after parallel LSTM and CNN layers with sequential sequence of LSTM and CNN layers before a final Dense output layer.

Conv + LSTM (Pooling No Size Reduction) Uses Conv + LSTM (With Pooling Size Reduction) architecture except CNN pooling layers use (1,1) stride and so data shape will not change after pooling.

Our Model (Pooling No Size Reduction) Uses Final Model (With Pooling Size Reduction) architecture except CNN pooling layers use (1,1) stride and so data shape will not change after pooling. This was the best model architecture we found.

Our Model (Reduced Architecture) Modifies Our Model (Pooling No Size Reduction) to use fewer layers and with smaller hidden units.

We experimented with all 6 models and used the plot of their accuracy scores on training and validation set across epochs Fig. 1 to decide the best architecture. We focused on the validation accuracy using the best validation accuracy it found across epochs and how long it maintained that high accuracy as a measure of how good the model performed. From our result we picked Our Model (Pooling No Size Reduction) as the best architecture Fig. 1f.



(a) (b) (c)

(d) (e) (f)

Figure 1: Architecture Selection

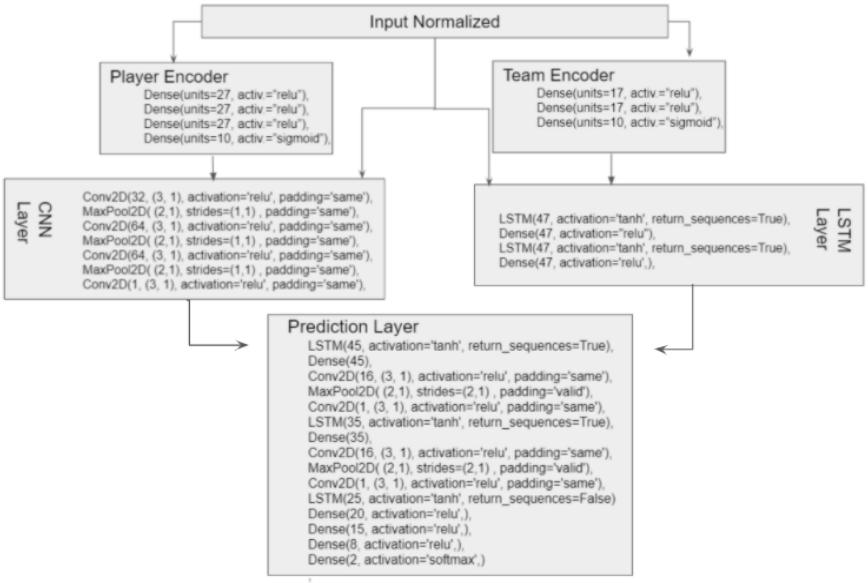


Figure 2: Architecture of Best Model Architecture

The final architecture of the best selected architecture is show in Fig. 2. Lastly we used the best architecture trained at epoch 300 using only training data set (validation data set not used to train) to evaluate how it performed on the testing data and compared the results reported in existing literature (Table 1). Our findings show that our results out perform other methods that presently exist for NBA game prediction.

Table 1: Comparison of Test Set Accuracy

Method	Accuracy
Strumbelj 2011 [11]	0.7069
Loeffelholz 2009 [5]	0.7433
Beckler 2013 [1]	0.73
Experts [1]	0.71
Ours (Epoch 300)	0.7959

5 Discussion

In our Architecture selection training and validation curves suggests that their might be over fitting appearing in many curves where the training accuracy reaches 1.0 when validation starts to decline. We believe this is due to the fact that the validation set is the last games chronologically so we are validating on the last games in the season when player are most under pressure to succeed. Additionally games at the end are between the teams that lasted the furthest into the season and denote games between the best of teams and getting close to the last games left to win the championship. It is likely our validation set contains games that are much harder to test on. So our reduction in validation set over time is due to our model focusing less on predicting on the hardest games and more on achieve better average accuracy. This is supported since we achieve a 0.79 accuracy on test set but we do not reach above 0.7 on validation set at any epoch, implying that our validation set is harder to predict well on.

We also notice that the worst models are models that are very simple which is shown in Fig. 2a and Fig. 2d. These models are so simple they cannot even fit on the training data set. Play-by-play models cannot be accurately modeled using simple models and need more complex models. We also note an improvement between using only LSTM to downstream process CNN and LSTM layers (Fig. 2b and Fig. 2e) vs using both CNN and LSTM in the downstream processing (Fig. 2c and Fig. 2f). Including both LSTM and CNN shows a longer duration of higher validation accuracy suggesting that using CNN and LSTM can improve the performance of our model. Lastly we note that pooling without reduction in shape gives large improvement with Fig. 2c failing to fit on training data set and Fig. 2b showing little change in accuracy over epoch.

Our final test results greatly outperforms existing results in literature which suggests that the inclusion of play by play data with deep learning which we have not seen done in literature to be an important factor in generating good results.

6 Conclusion

In this paper we present the problem of using play by play data for accuracy prediction and show improved results by using CNN with LSTM. From our findings we believe that the use of play-by-play data for predicting games is an area that has not been fully explored and should be considered for future exploration.

Acknowledgments

We would like to acknowledge Kaggle data set retrieved from `https://www.kaggle.com/schmadam97/nba-playbyplay-data-20182019` as the source of play-by-play data as well as acknowledge basketball-reference.com as the source of team and player statistics. Furthermore, we would like to acknowledge Dennis Núñez Fernández for providing this NEURIPS paper template.

References

[1] Matthew Beckler, Hongfei Wang, and Michael Papamichael. Nba oracle. *Zuletzt besucht am*, 17(20082009.9), 2013.

[2] Tommy Beer. Gamblers in new jersey bet record $6 billion in 2020 and nearly $1 billion in december alone. *Forbes*, 2021.

[3] Ge Cheng, Zhenyu Zhang, Moses Ntanda Kyebambe, and Nasser Kimbugwe. Predicting the outcome of nba playoffs based on the maximum entropy principle. *Entropy*, 18(12):450, 2016.

[4] Forbes. National basketball association total league revenue from 2001/02 to 2019/20 (in billion u.s. dollars). 2021.

[5] Bernard Loeffelholz, Earl Bednar, and Kenneth W Bauer. Predicting nba games using neural networks. *Journal of Quantitative Analysis in Sports*, 5(1), 2009.

[6] Daniel Pettersson and Robert Nyquist. Football match prediction using deep learning. *Psychol. Sport Exerc.*, 15(5):538–547, 2017.

[7] Desh Raj, Sunil Sahu, and Ashish Anand. Learning local and global contexts using a convolutional recurrent network model for relation classification in biomedical text. In *Proceedings of the 21st conference on computational natural language learning (CoNLL 2017)*, pages 311–321, 2017.

[8] Vamsi Saladi. Deepshot: A deep learning approach to predicting basketball success.

[9] Rajiv Shah and Rob Romijnders. Applying deep learning to basketball trajectories. *CoRR*, abs/1608.03793, 2016.

[10] Petar Vračar, Erik Štrumbelj, and Igor Kononenko. Modeling basketball play-by-play data. *Expert Systems with Applications*, 44:58–66, 2016.

[11] Erik Štrumbelj and Petar Vračar. Simulating a basketball match with a homogeneous markov model and forecasting the outcome. *International Journal of Forecasting*, 28(2):532–542, 2012.